Agora: Unified Framework for Crowd Simulation Research

by
Michelangelo Diamanti

Dissertation submitted to the Department of Computer Science
at Reykjavík University in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

June 13, 2023

Thesis Committee:

Supervisor:	Hannes Högni Vilhjálmsson, Professor, Reykjavík University (RU), Iceland
Examining Board:	Yngvi Björnsson, Professor, Reykjavík University (RU), Iceland
	Christopher Peters, Associate Professor, Kungliga Tekniska Högskolan (KTH), Sweden
External Examiners:	Nuria Pelechano, Associate Professor, Universitat Politecnica de Catalunya (UPC), Spain

Publishing Information

ISBN Print version:	978-9935-539-22-9
ISBN Electronic version:	978-9935-539-23-6

Author's ORCID:	0009-0004-0175-1989

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirements for the degree of Doctor of Philosophy.

Date of Signature

9. jun 23

Hannes Högni Vilhjálmsson
Professor, Reykjavík University (RU) Department of Computer Science,
Reykjavik, Iceland
Supervisor

9. júní 2023

Yngvi Björnsson
Professor, Reykjavík University (RU) Department of Computer Science,
Reykjavik, Iceland
Examining Board

26/05/2023

Christopher Peters
Associate Professor,
Kungliga Tekniska Högskolan (KTH), Sweden
Examining Board

26/05/2023

Nuria Pelechano
Associate Professor,
Universitat Politecnica de Catalunya (UPC), Spain
External Examiner

REYKJAVIK UNIVERSITY

Agora: Unified Framework for Crowd Simulation Research

Michelangelo Diamanti

June 13, 2023

Abstract

Crowd simulation focuses on modeling the movements and behaviors of large groups of people. This area of study has become increasingly important because of its several applications in various fields such as urban planning, safety, and entertainment. In each of these domains, the presence of virtual agents exhibiting realistic behavior greatly enhances the quality of the simulations. However, the inherently multifaceted and intricate nature of human behavior presents a unique challenge, necessitating the effective combination of multiple behavior models. This thesis introduces a novel theoretical framework for modeling human behavior in crowd simulations, addressing the unresolved issue of combining a plethora of behavior models, often developed in isolation. The proposed framework decomposes human behavior into fundamental driving stimuli, which are then represented graphically through the heatmap paradigm. Subsequently, the agent behavior is influenced by the heatmaps, which guide them toward attractive areas and steer them away from repulsive locations based on the encoded stimuli. A key advantage of this approach lies in the ability to combine heatmaps using well-defined color operations, effectively integrating different aspects of human behavior. Furthermore, the heatmap paradigm facilitates objective comparison of simulation output with real-world data, employing image similarity metrics to evaluate model accuracy. To realize this framework, the thesis presents a modular software architecture designed to support various tasks involved in crowd simulation, emphasizing the separation of concerns for each task. This architecture comprises a collection of abstract modules, which are subsequently implemented using appropriate software components to realize the underlying features, resulting in the Agora framework. To assess the ability of Agora to support the various tasks involved in crowd simulation, two case studies are implemented and analyzed. The first case study simulates tourists visiting Þingvellir national park in Iceland, examining how their behavior is influenced by the visibility of the surrounding environment. The second case study employs Agora to model the thermal and density comfort levels of virtual pedestrians in an urban setting. The results demonstrate that Agora successfully supports the development, combination, and evaluation of crowd simulation models against real-world data. The authoring process, assisted by Agora, is significantly more streamlined compared to its native counterpart. The integration of multiple models is achieved by combining the heatmaps, resulting in plausible behavior, and the model assessment is made convenient through the evaluator within the framework. The thesis concludes by discussing the implications of these findings for the field of crowd simulation, highlighting the contributions and potential future directions of the Agora framework.

Agora: Samræmd rannsóknarumgjörð fyrir mannfjöldahermun

Michelangelo Diamanti

June 13, 2023

Útdráttur

Mannfjöldahermun fæst við gerð líkana af hreyfingu og hegðun stórra hópa af fólki. Mikilvægi þessa rannsóknasviðs hefur vaxið stöðugt vegna hagnýtingar á margvíslegum vetvangi, eins og til dæmis á vetvangi borgarskipulags, öryggis og afþreyingar. Þegar sýndarmenni hegða sér á sannfærandi hátt, leiðir það til betri hermunar fyrir þessi notkunarsvið. En mannleg hegðun er í eðli sínu margbrotin og flókin og því er það sérstök áskorun við smíði sýndarmenna að sameina, með áhrifaríkum hætti, mörg mismunandi hegðunarlíkön. Þessi ritgerð kynnir nýja fræðilega umgjörð líkanasmíði mannlegrar hegðunar fyrir mannfjöldahermun, sem tekur á þeim óleysta vanda að sameina fjölda hegðunarlíkana, sem oft eru þróuð með aðskildum hætti. Umgjörðin brýtur mannlega hegðun niður í grundvallar drifáreiti, sem eru sett fram grafískt útfrá hugmyndafræði hitakorta. Sýndarmennin hegða sér síðan undir áhrifum frá hitakortunum, sem vísa þeim í áttina að aðlaðandi svæðum og stýra þeim burt frá fráhrindandi svæðum, útfrá hinu umritaða áreiti. Lykilkostur þessarar nálgunar er sá eiginleiki að geta blandað saman hitakortum með vel skilgreindum litaaðgerðum, sem eru þá í raun samþætting mismunandi hliða mannlegrar hegðunar. Hitakortshugmyndafræðin auðveldar ennfremur hlutlægan samanburð hermunarúttaks og raungagna með notkun myndsamanburðarmælinga, til að meta nákvæmni líkana. Varðandi útfærslu, þá kynnir þessi ritgerð einingadrifna hugbúnaðarhögun sem er hönnuð til að styðja við ýmsa ferla mannfjöldahermunar, með áherslu á aðskilnað helstu viðfangsefna hvers ferlis. Þessi högun inniheldur safn huglægra eininga, sem síðan eru útfærðar með viðeigandi hugbúnaðarhlutum, sem raungera undirliggjandi eiginleika. Útkoman er sjálf Agora umbjörðin. Tvö sýnidæmi eru útfærð og greind til að meta getu Agoru til að styðja við ýmis mannfjöldahermunarverkefni. Fyrra dæmið hermir eftir ferðamönnum sem heimsækja Þingvallaþjóðgarð, og skoðar hvernig hegðun þeirra verður fyrir áhrifum sýnileika umhverfisins sem umleikur þá. Seinna dæmið nýtir Agoru til að smíða líkan af hitauppstreymis- og þéttleikaþægindum hjá sýndarvegfarendum í borgarumhverfi. Niðurstöðurnar sýna góðan árangur Agoru við að styðja þróun, samþættingu og mat mannfjöldahermunarlíkana gagnvart raungögnum. Þróunarferlið er verulega þjálla með Agoru en með hefðbundnum aðferðum. Samþætting margra líkana tókst með blöndun hitakorta, möguleg hegðun var framkölluð og mat á líkönunum varð þægilegra með umgjörðinni. Ritgerðinni lýkur með því að fjalla um áhrif þessara niðurstaðna á svið mannfjöldahegðunar, með áherslu á nýstálegt framlag þessarar rannsóknar og mögulega framtíðarþróun Agora umgjarðarinnar.

Dedicated to my family,
my very own giants upon whose shoulders I stand to reach this milestone.

Acknowledgments

This Ph.D. journey, marked by discovery, learning, and a fair share of challenges and triumphs, wouldn't have been achievable without the constant support and wisdom of several individuals. I want to take this opportunity to express my gratitude to everyone who contributed to this process in myriad ways, big and small.

My deepest gratitude goes to my supervisor, Hannes Högni Vilhjálmsson, whose kind, optimistic, and resilient character proved instrumental. His contagious passion for science, research, and teaching motivates everyone around him, myself included, to reach for higher standards. He was a guiding force, providing positivity and demonstrating a unique ability to ingeniously turn difficulties into opportunities. His unwavering support has been invaluable and sets the bar for the researcher and person I aspire to become.

I extend my appreciation to my committee members, Yngvi Björnsson and Christopher Peters, who have offered invaluable insights and support from the very beginning of this endeavor. I am equally grateful to Nuria Pelechano, whose rigorous evaluation in the role of examiner enhanced the quality and depth of this research. The collective guidance of the committee has been crucial in the successful completion of my dissertation.

Many thanks to Claudio Pedica who provided invaluable hands-on supervision at the beginning of this journey. I am also grateful for the notable contributions made by Hallgrímur Andrésson, Sofia Basílio, Hrafnkell Þrastarson, and Jóhann Indriðason as interns during their summer projects. Finally, my gratitude goes out to all past and present members of the Center for Analysis and Design of Intelligent Agents, whose stimulating discussions have been a source of inspiration. Special recognition to Reykjavík University for its fantastic facilities and exceptional staff.

This work was made possible thanks to the funding provided by the Icelandic Research Fund. I am particularly thankful to them for their financial support through grant #174444-051, which supported the "Character Territoriality" project, and grant #217663-052, for "The Agora Framework" project.

During this journey, I've had the privilege of meeting countless amazing people who have contributed to making Iceland feel like a home away from home. Each of them has played a part in shaping this experience, and for this, I offer my sincere

thanks. This journey, like many others, had its fair share of hellos and goodbyes, but the impact of those shared experiences remains. I am profoundly grateful to Alessia, whose serendipitous entry into my life acted as a key for deciphering the complex pages of this story. Thanks to Giulio, Marco, and Judith, whose presence and friendship remained constant throughout this adventure. A big thanks to the first "Italian Gang", especially Federica and Dario, who formed in the spring and broke the metaphorical ice, setting the tone for my Icelandic stay. I can't forget my energetic and daring Skipholt flatmates, Ben, Frederick, and Henrik, whose spirit was a constant source of motivation. Equally appreciated are Valentina and Carlo, who always found their way back to Iceland. Finally, my heartfelt gratitude goes to the wonderful friends I've made during the last leg of this journey, who accompanied me during the final stretch of this rollercoaster: Riccardo, Camilla, Lena, Gabriele, Chiara, Lorena, Federica P., and Federica K.

While remarkable people in Iceland made it feel like a second home, my friends back in Italy brought the Icelandic spirit of independence and adventure into my original home, bridging the distance between the two. Despite the time apart, each reunion feels as if I've never left. For their enduring friendship, my deepest thanks go to: Gianmarco, Lorenzo, Alex, Aldo, Alessio, Cristina, Simone, Paolo, Chiara, Francesco, Davide, Nicole, Nicolò, Michela, Dennis, and Martina. I extend my appreciation to my university friends: Aurora, Brahim, Davide, Luca, and Christian. Despite being dispersed across various locations and life trajectories, they have remained in contact and been a constant source of support.

I owe a profound debt of gratitude to my family, my safe harbor in every storm. My deepest love and appreciation extend to my mother, Giovanna, and my grandmother, Domenica (Memma), whose selfless affection and unending support have driven me to pursue my goals, despite the distances they created. My brother, Jacopo, deserves special recognition for being a guiding light and forging a path for me to follow. Heartfelt thanks to Silvia, Marina, Ettore, and Maddalena for their unwavering support and infectious cheerfulness. I'm equally grateful to Paola, Sante, and Mirko, who have been my second family, offering deep care and support. My appreciation also extends to the international side of my family: Antonio, Ellen, Martino, Clara, and Sonia, who continually remind me that the world is vast and full of wonders worth exploring. I also want to express my deep gratitude to Anna, Riccardo, Michela, and Mirco, whose relationships stand as a testament to the strength and significance of familial bonds. Lastly, my thanks go to Bianca, Primo, their delightful family, and Sandro, for their constant care and kindness.

While my appreciation for those who helped me along the way knows no limits, alas, the space here does. So, goodbye and thanks for all the fish.

-Michelangelo

List of Tables

List of Figures

Contents

Chapter 1

Introduction

1.1 A Crowd Simulation Utopia

One of the remarkable capabilities of computer science is the ability to study and replicate reality through the use of simulations. In the realm of crowd simulation, the ultimate goal is to create realistic models that replicate the diverse and intricate behaviors of individuals as they interact with one another and their surroundings (see Figure 1.1).

Imagine a perfect world of crowd simulation, where a comprehensive library of human behavior components is made available to simulation designers. These designers would be tasked with creating a realistic model of a busy city center, filled with diverse individuals going about their daily routines. Each person in this virtual world would exhibit behaviors and characteristics that accurately reflect their real-world counterparts.

In this ideal scenario, simulation designers could effortlessly select and combine different aspects of human behavior from the extensive library. These behavioral components would be incorporated into the simulation, enabling the virtual agents to interact with one another and with the environment in a manner that closely mirrors reality. Each agent would be driven by internal motives, seamlessly integrating their needs, reasoning, and factors that attract or repulse them into their decision-making process. As a result, from the way they address their individual goals to how they avoid obstacles and navigate through bustling crowds, every aspect of their behavior would be customizable and cohesively integrated into the simulation.

The components would work together harmoniously, allowing designers to swap or modify individual elements without disrupting the overall simulation. They could experiment with different combinations of behaviors to study various scenarios, such as introducing a large group of tourists navigating an unfamiliar city or observing how virtual agents respond to unexpected events like an emergency.

In this perfect world of crowd simulation, the modular approach would allow for the creation of diverse and intricate simulations that could be easily customized and adapted to study a wide range of situations. The ability to combine various behaviors would result in a holistic model that closely resembles real human behavior, providing

researchers and practitioners with powerful insights into how people interact in complex environments.

This vision of an ideal world of crowd simulation highlights the potential impact that advanced and flexible simulation tools could have on the understanding of human behavior in various contexts. By striving towards this ideal, innovative approaches and frameworks could be developed that enable more accurate and comprehensive simulations of human interactions in crowded environments.

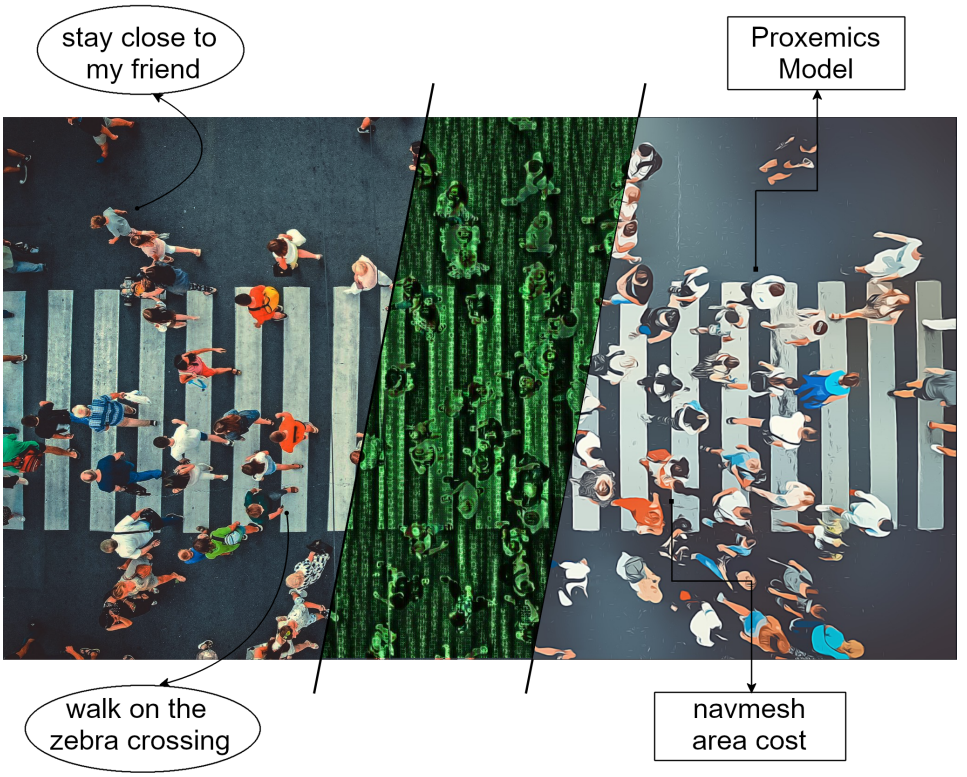

Figure 1.1: Left: People crossing the street in the real world; Middle: crowd simulation; Right: the resulting virtual environment/agents. People are guided by their internal reasoning and abide by specific rules. Agents do the same.

1.2 The Status Quo

Although the vision of an ideal crowd simulation world where people can effortlessly pick and choose behaviors and seamlessly integrate them into simulations is enticing, the current state of the field presents a different picture. The driving force behind the advancements in crowd simulation has been the study of human social behavior in the real world. In an attempt to replicate the complexities of human behavior,

the field has adopted a "divide and conquer" strategy by breaking down the problem into more manageable sub-problems. These sub-problems address specific aspects of human social behavior and translate them into computational models.

These models are built on precise techniques for guiding agents through environments while avoiding collisions, often involving numerous parameters that need calibration. The output of these models is then evaluated using well-defined metrics. In crowd simulation, all of these elements come together in interactive 3D virtual environments where animated agents move and behave like real people. Each model excels in its niche, showcasing remarkable techniques for simulating human behavior.

However, the field has not yet reached the ideal world of crowd simulation. One of the main reasons is that despite the existence of frameworks aiming to unite and standardize the field, many advancements remain confined to their sub-field. Additionally, each behavior model is closely tied to its underlying components and technologies, such as global pathfinding and local obstacle avoidance techniques, making it difficult to fairly compare different models addressing the same phenomenon. Furthermore, combining multiple behaviors into a unified model poses a significant challenge. There is currently no meaningful way to do so, and even if such a method existed, accommodating all possible combinations of components would be an arduous task.

1.2.1 Problem Statement

The problem this research seeks to address is the fragmentation and limited interoperability of existing crowd simulation models. Despite considerable advancements in the field, there is a lack of a unified approach that enables the seamless integration and combination of multiple behavior models. This fragmentation hinders the ability to create more accurate and realistic simulations of human behaviors and interactions, as well as complicating the fair comparison and evaluation of different models. The problem is further exacerbated by the closely tied nature of behavior models to their underlying components and technologies. The challenge, therefore, lies in developing a framework that can effectively combine diverse crowd simulation models, facilitate meaningful comparisons, and support the creation of more comprehensive and accurate simulations of human behavior in various scenarios and environments.

In light of the aforementioned problem, this research seeks to address the following research questions:

R.Q. I: What are the potential theoretical and technical challenges and limitations involved when attempting to integrate fragmented behavior models in crowd simulations?

R.Q. II: What paradigm can be employed to enable the seamless integration and combination of multiple behavior models in crowd simulations?

R.Q. III: Given a suitable paradigm for integrating multiple behaviors, what are the critical components and architectural principles that should be

considered when designing and implementing a system that leverages this paradigm?

R.Q. IV: How can the integration of multiple behavior models contribute to improved realism and accuracy in representing human behaviors and interactions in crowd simulations?

R.Q. V: How can objective evaluation methods be devised to assess the realism of various crowd simulation models, while maintaining a balance between evaluating individual behaviors at a local level and their wider interactions in diverse scenarios and environments?

This dissertation offers a new approach to tackle these challenges by introducing a framework that seeks to mitigate the existing limitations, promoting the development of more comprehensive and accurate crowd simulations.

1.3 A Hopeful Protopia

The journey toward an ideal world of crowd simulation is not an overnight transformation. Instead, the process will be characterized by incremental advancements, each contributing a piece to the puzzle and gradually enhancing the state of the field. This incremental progress leads us towards what is known as "protopia"[1] for crowd simulation.

1.3.1 Proposed Solution

This thesis proposes a crowd simulation framework that leverages heatmaps for modeling agent behaviors. A heatmap is a graphical representation of spatial data that uses color intensity to indicate the magnitude of a variable, enabling the visualization and analysis of complex patterns, relationships, or behaviors within an environment. The key idea behind the proposed approach is that certain components of human social behavior can be expressed through graphical spatial representations, of which heatmaps are an example. By encoding various aspects of human behavior like visibility, mental or emotional states, and other factors into heatmaps, it becomes possible to combine different models of human behavior, using a relatively simple set of mathematical blending operations, resulting in a more holistic simulation model.

The proposed framework, referred to as Agora, supports classic crowd simulation techniques, including those addressing high-level goal selection, global path planning adjustments, and local steering, as well as output evaluation. The software architecture of Agora addresses key requirements such as usability, modularity, scalability, and versatility, making it a powerful tool for researchers and practitioners alike.

By implementing the Agora framework, the goal is to make significant strides toward a more integrated, efficient, and realistic representation of human behavior in crowd simulations. As progress continues on this path, each incremental upgrade

[1] a neologism for a realistic, achievable, and better future. https://kk.org/thetechnium/protopia/

brings the field closer to the envisioned protopia, where the complexities of crowd simulation are better understood, managed, and harnessed for the benefit of various applications and domains.

1.4 Contributions and Thesis Structure

This section serves a dual purpose. Firstly, it underscores the significant contributions of this thesis to the field of crowd simulation, including the development and application of the Agora framework, and related academic publications. Secondly, it presents the thesis structure, providing a roadmap of each chapter to guide the reader through the progression of this research.

1.4.1 Contributions

This thesis has contributed significantly to the body of knowledge in the field of crowd simulation, with elements of this research presented in peer-reviewed publications. Notably, [33] presents a unique characterization of the current state of crowd simulation, identifying the existing limitations in current approaches. This work forms the basis of the extensive literature review of the Background Chapter 2.

Furthermore, as detailed in [34], the research has also contributed towards the integration of Menge and Unity 3D, enhancing Menge's modeling tools to include visual authoring GUIs for defining simulation scenarios and behavior. This aspect is discussed more in-depth in Chapter 5.

The primary output of this work, the Agora framework, has been employed in two unique projects (case studies). The first, explored in Section 6.1, investigates the impact of visibility on tourist behavior, hypothesizing that more visible locations attract individuals, particularly tourists visiting a location for the first time. This hypothesis was initially modeled as a native simulation (developed from scratch with Unity 3D), and later replicated with Agora to assess its ease of use. To validate the simulation output, a field study was conducted at Þingvellir National Park in Iceland, with the findings published in [106].

The second application of the Agora framework was in a simulation of an urban environment, wherein the behavior of the agents was influenced by their thermal comfort and crowd density comfort. The underlying model, derived from existing literature, was successfully replicated with Agora, illustrating its capability to conveniently adapt and enhance models from the literature. Further details on this case study are provided in Section 6.2.

1.4.2 Thesis Structure

Chapter 2 presents a comprehensive literature review that decomposes the field of crowd simulation into a set of sub-fields, **highlighting key limitations** that hinder faster advancements in the domain. This review sets the stage for the introduction of the proposed solutions. In Chapter 3, a **theoretical framework** centered around the heatmap paradigm for modeling human behavior in crowd simulations

is introduced. This framework enables the **combination of multiple models** and facilitates their evaluation through the heatmap approach. Following the theoretical framework, Chapter 4 describes the **system architecture**, and Chapter 5 provides a detailed **implementation** of the Agora framework. To demonstrate and evaluate the **practical applications** of the Agora framework, Chapter 6 describes two case studies that exemplify its capabilities and potential uses. Lastly, Chapter 7 discusses the results obtained in relation to the problem statement. Chapters 8 and 9 explore future works and conclusions derived from the contributions. These final chapters of the thesis position the approach within a broader context, considering general limitations and shedding light on the exciting opportunities the work unlocks for advancing the field of crowd simulation.

Chapter 2

Background

Crowd simulation is a growing field of research due to its potential applications in several areas, including, safety, urban planning, architecture, entertainment, and training of public emergency response personnel. However, these applications require agents capable of social reasoning to make the simulation model more believable. To this end, researchers employ a range of social behavior theories studied in psychology and sociology to generate and animate virtual human behavior in crowd simulations. Several navigation algorithms have also been developed to enable virtual humans to traverse space in increasingly realistic ways. Despite advances in simulation techniques providing ways of objectively calibrating parameters and validating simulation models, the field has become fragmented due to its reliance on different components from sub-fields. This makes it difficult to compare, combine, and ultimately predict the interactions between different models. To address this issue, there have been several attempts to create frameworks for decomposing the crowd simulation field into smaller sub-problems, such as Menge. However, these frameworks cannot still combine several social behavior models, which is essential for developing a unified framework. Further research is needed to create a unified framework that can integrate different social behavior models and provide a platform for the comparison and combination of existing models. Such a framework would allow for the development of more complex and realistic crowd simulations, thus advancing the field closer to reality.

This chapter presents an overview of the state of the art in crowd simulation, reviews its constituent domains, and discusses the main challenges in the field. The review of each subfield includes a description of the key concepts and techniques, as well as the existing literature and open research questions. It is concluded that, although the divide and conquer strategy has been successful in allowing research progress in each subfield, there is a need for a unified framework that can integrate these disparate subfields and provide a standard method for designing and implementing crowd simulation models. The chapter begins by examining human behavior, as it forms the essential foundation for accurately modeling crowd simulations.

2.1 Human Behavior

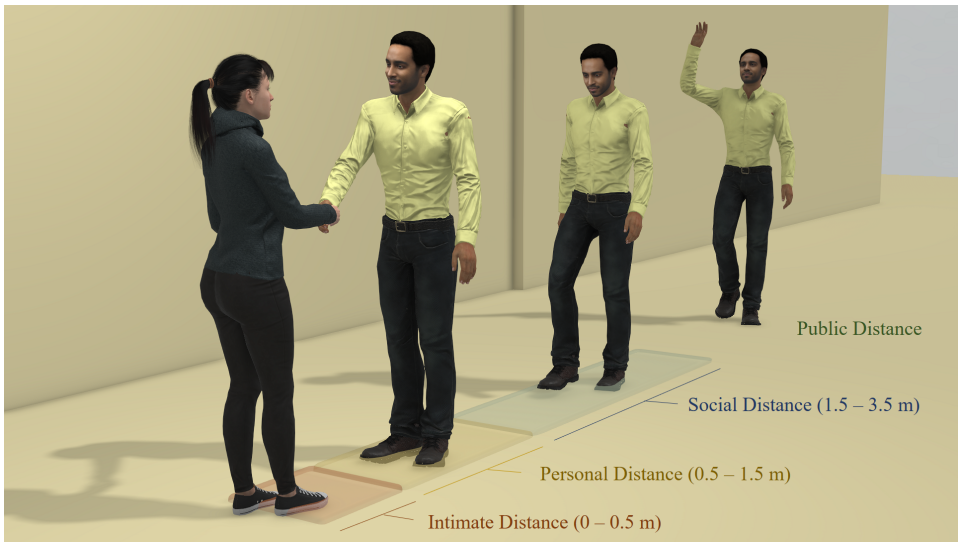

Figure 2.1: Proxemics inspect the four meaningful ranges of interpersonal distance as introduced by Hall and Hall [62]; intimate, personal, social and public. To initiate contact, a sequence of salutations is used to determine the suitable final configuration. Figure from [144].

The study of human behavior in physical and virtual spaces has been a major factor in the development of realistic crowd simulations and social negotiation strategies for *social agents*. By understanding the structure and dynamics of social space, and how it is affected by the presence of other humans, agents can navigate and interact with their environment more effectively. This knowledge allows for more natural and engaging interactions with those around them, as well as a better understanding of the context in which the agents are operating. Research has also helped to identify certain patterns of behavior among humans in social spaces, which can be used to create better algorithms for negotiating co-presence, as well as anticipating and responding to the behavior of others. Additionally, understanding the various motivations and preferences of humans in social spaces can help agents better tailor their behavior to suit the social occasion they are in. All of these elements form a crucial part of developing agents that are capable of occupying and managing social spaces, ultimately allowing them to interact with their surroundings, and other agents, more naturally and effectively. This section provides an overview of the research conducted in the field of human behavior, particularly its relation to the physical social environment.

Concept	Description	Sources
social space	Any space with mutual embodied access	EG
gathering	Collection of two or more people	EG
social situation	Space that adds people immediately to a gathering	EG
social occasion	Reason for a social situation	EG
formation	Tight cluster of coordinated activity within a gathering	AS, AK
dyad	Formation of two people	AS
element	Formation of people with common orientation	AS
F-formation	Formation of people facing each other	AS, AK
o-zone	Innermost zone covered by overlapping orientations	AS, AK
p-zone	Zone occupied by formation participants	AS, AK
r-zone	Outermost zone serving as formation transition area	AS, AK

Table 2.1: Some useful concepts that describe the structure of social space and their sources (EG = Goffman [51]; AS = Scheflen [123]; AK = Kendon [78]). Table adapted from [144].

2.1.1 Social Space

The configuration of the space is both an active and passive factor in the behavior of humans in social spaces. The way space is organized can influence the behavior of the people therein, and conversely, humans can arrange the space to suit their needs according to specific social situations. For this reason, it is important to understand the underlying structure of social space as it is perceived by human beings.

This section provides an overview of the concepts and theories that have been developed to describe the structure of social space and how the body is used to occupy and manage spatial boundaries. Moreover, it discusses the importance of physical presence in social spaces and how it affects the behavior of humans in this context. For a summary see Tables 2.1, 2.2, and 2.3 which have been adapted from [144].

Social Space and Physical Presence

Social space is a term that refers to any environment where people can access one another through their physical presence [144]. All occupants of a social space don't need to be socially engaged with one another, but they must display minimal social awareness and manage their co-presence. To better define the structure of social space, researchers have agreed on a set of terms to describe its various

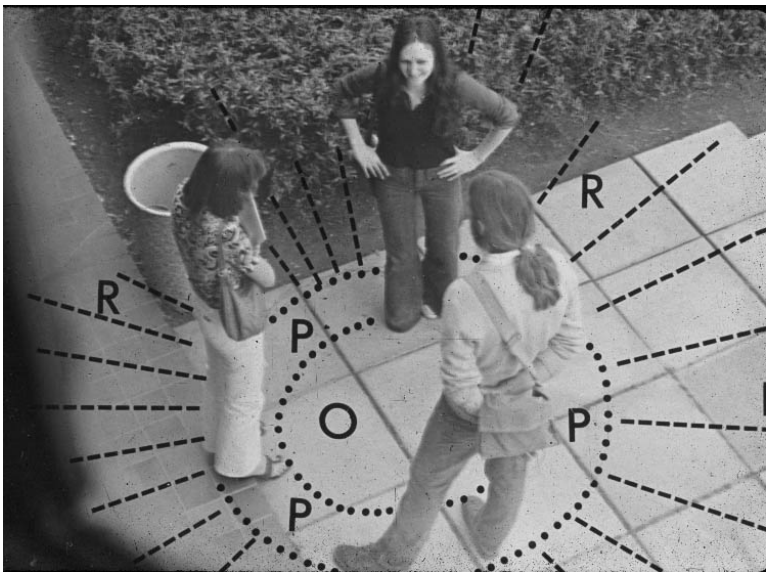

Figure 2.2: An example of a conversational group organized as an F-formation with the main functional spaces indicated. Figure from [79].

elements. *Gatherings* are loosely formed collections of people, while *formations* are tighter clusters where more coordinated activities occur. Formations are made up of individual locations that provide room for the individual's actions, and these locations can be pre-allocated or claimed by participants during a social event. Among the most simple formations are *dyads*, which are composed of two people, and *elements*, which are the ones where participants all face the same direction. Another particular type of formation, proposed by Kendon is the *face formation*, or F-formation, which occurs when participants are facing each other, as shown in Figure 2.2, and usually divides the space so that each participant has equal access to the interaction. The space occupied by and surrounding a formation has been characterized into different zones, arranged in descending order depending on the level of social engagement: o-zone, p-zone, and r-zone [78]. Physical structures, such as furniture, can also influence the formation of gatherings and are sometimes deliberately set up for facilitating social occasions. Overall, social space provides an opportunity for people to interact with one another through their physical presence and can be used to facilitate various types of social engagements.

The physical space occupied by a person in a social setting is an important concept in the study of interpersonal relations and communication. By being present in a social space, a person will occupy a certain physical volume. One useful approximation of this volume is four cubic cubits. The cubit measure was originally defined as the distance from a full-grown person's elbow to the end of their middle finger. While the physical body is bound to its exact location, four different body regions also claim space beyond that location through independent outward orientation. That is, by orienting the head, torso, pelvis, and legs/feet,

Concept	Description	Source
cubit	Elbow to fingertip, approx. 0.5m	AS
region	one $cubit^3$ sized portion of the body	AS
location	Area needed by person for action (approx. 2×2 cubits)	AS
segment	Space claimed by body region orient.	AS
territory	The space both occupied and claimed by a person	AS
intimate distance	$< 1cubit$ away, sharing location	EH
personal distance	Comfortable distance, between 1 to 3 cubits away	EH
social distance	Impersonal interaction distance, between 3 and 7 cubits away	EH
public distance	Safe distance from anyone, over 7 cubits away	EH

Table 2.2: Some useful concepts that describe the spatial structure bodies and their sources (AS = [122, 123]; EH = [62]). Table adapted from [144].

four different segments of space are created and claimed, which can be combined in pairs corresponding to the upper and lower body regions. The space occupied by the person and their claimed segments are termed their territory.

The classification of distances, proposed by [62] and illustrated in Figure 2.1, has resulted in four useful interpersonal distances: Intimate, personal, social, and public. *Intimate distance* is closer than one cubit (0.5m away) thus more or less sharing the same location. *Personal distance* corresponds to the comfortable distance one typically maintains from others, which ranges from 0.5m to 1.5m. *Social distance* is kept with those one intends to conduct relatively impersonal interactions with, ranging from about 1.5m to 3.5m. *Public distance* is anything further than 3.5m and would be considered a relatively safe distance from anyone.

Behaviors in Social Space

Understanding the structure of social space and how the body is used to manage spatial boundaries is crucial to understanding human behavior in social settings. This section provides some examples of phenomena that occur in social space, including the interaction between people, the formation of groups, and how emotions influence behavior. By examining these behaviors, it is possible to gain insight into how individuals interact with one another and how they navigate the complex dynamics of social space.

Interaction. Social norms shape behavior in social spaces. As identified by Goffman, these behaviors can be classified into two main categories: those that belong to unfocused interaction and those that belong to focused interaction.

Unfocused interaction is what occurs when people are merely managing their co-presence in the social space. This involves navigating the space without causing others distress, staying at least a personal distance away, and following established paths such as streams and lanes. People also need to signal an expected level of social awareness and may use *involvement shields* to maintain public distance from other people and prevent others from engaging. In contrast, *focused interaction* involves an extended social engagement or involvement, such as a conversation or watching something together. Several behaviors can be observed in this context, such as pointing towards a clear visual reference and orientating the head towards the interested party with a minimal orientation towards others. Additionally, people participating in focused interaction can establish and maintain membership in formations, and show different levels of commitment through body orientation, exposure, and tactile contact. These behaviors and their interpretation have been implicitly encoded in the social norms of each culture, and understanding these dynamics is key to successful engagement in social situations.

Social Groups. The behavior of individuals within social groups has been the subject of intense study due to its relevance to fields such as sociology and psychology. One of the most notable aspects of social behavior in groups is the tendency for individuals to exhibit coordinated movement or behavior. For example, *herding* is a type of social behavior where individuals in a group align their thoughts or actions through local interaction without centralized coordination [112]. Herding has broad applications, from intellectual trends to mob violence, and is relevant in an increasingly interconnected world where people are part of ever larger social groups. When the herding behavioral pattern is applied to the movements of individuals, it is referred to as *flocking*. Flocking is particularly observed in animals such as bird flocks but, as noted by Belz et al. [14], the basic components of this behavior can be observed in human groups as well. Faria et al. [44] substantiated this claim by showing that individuals near a person crossing a street were inclined to mimic the action and cross the street earlier than those further away. This suggests that humans in group settings utilize social cues from their immediate peers to maintain proximity, resulting in conformity to their neighbors' actions. Finally, *leader-following* behavior refers to the tendency of individuals in a group to follow the actions of a perceived leader. This behavior plays a crucial role in group dynamics, particularly in emergencies where quick decisions and actions are necessary [35]. However, it can also lead to conformity and a lack of critical thinking, potentially causing the group to overlook important information or make poor decisions.

These group dynamics are believed to emerge from the interactions among individuals in the group and are thought to serve a variety of functions, including reducing individual risk, improving group efficiency, and facilitating communication. Thus, they play a crucial role in the behavior of people in social groups.

Social Networks. Another factor that plays an important role in influencing the behavior of people in social spaces is the formation of social networks. Researchers from anthropology and social sciences, such as McPherson et al., highlight the

importance of geographic homophily in shaping various human societies. *Homophily* refers to the tendency of individuals to associate and form connections with those who are similar to them. As a result, people who know each other tend to live and gather in specific spatial locations, increasing their chances of interaction. It is worth noting that people form communities not necessarily to achieve certain goals, but rather because they naturally interact and come into contact with each other regularly. This concept highlights the social and cultural factors that contribute to the formation of human social groups and societies [96].

Researchers can employ *social network analysis* techniques [100] to examine how individuals who share similar characteristics, such as interests, beliefs, and values, form connections and develop social ties with one another. By analyzing the network structure and composition, researchers can identify patterns of homophily and understand how these patterns shape the formation and dynamics of social groups. Additionally, social network analysis can be used to examine the behavior of individuals within these social groups. Researchers can examine how individuals interact with one another, the roles that they play within the group, and how their behavior may be influenced by the characteristics of the social network in which they are embedded.

Emotion contagion. *Emotional contagion* is a process where an individual's emotions and moods can be influenced by those around them [69]. It is a form of mimicry where facial expressions, vocalizations, postures, and movements of others can lead to emotional convergence. Emotional contagion is a complex phenomenon, influenced by psychophysiological, behavioral, and social factors, and can elicit similar or complementary responses. This phenomenon is important because it can produce synchrony or entrainment of attention, emotion, and behavior, which has an adaptive function for social entities. Several basic processes have been proposed to explain emotional contagion. Hatfield et al. [65] examines interactional mimicry and synchrony as a possible mechanism for emotional contagion. The authors suggest that people automatically imitate and coordinate their behavior with those of others during social interactions, leading to the spread of emotions from one person to another. This feedback loop of mimicry and emotional contagion appears to be an unconscious and automatic process. McIntosh et al. [94] suggests that the strength of the contagion may be influenced by the similarity or likability of the source by the target.

Emotional contagion is important in the context of group dynamics because it can have a significant impact on the emotional climate of a group and influence the behavior and attitudes of group members. Thus, understanding how this phenomenon occurs and how it influences the behavior of individuals, can give additional insight into social group dynamics.

The structure of social space, the physical volume of an individual therein, and the behaviors that people engage in to carry out social functions are important concepts that have been extensively studied in social sciences and have been formalized into theoretical frameworks of social intelligence and social cognition.

As seen in Figure 2.1 and 2.2, these concepts are particularly suited to be

Concept	Description	Sources
unfocused interaction	Co-presence without meaningful interaction	[51]
focused interaction	Commitment to social engagement or involvement	[51]
navigating	Moving through space while avoiding collisions	Any
following	Navigating behind another person that picks the path	Any
civil inattention	Minimal eye contact to signal awareness without engagement	[51]
involvement shield	Avoiding social involvement by preventing mutual observation	[51]
approach	Navigating toward another person with intent to engage	[80, 78]
commitment	How interested a person is in interacting with another	[123]
herding	Coordinated decentralized behavior	[112]
flocking	Herding applied to movement	[14]
leader-following	Mimicking actions of a leader	[35]
homophily	Tendency to associate with similar individuals	[96]
Emotional contagion	Tendency of people's emotions to be influenced by those around them	[69, 65, 94]

Table 2.3: Some useful concepts that describe behaviors in social space. Table adapted from [144].

expressed in 2D space. Specifically, it is possible to encode them through two-dimensional social behavioral maps that provide a visual representation of the complex structure of social space, including physical dimensions and social norms that guide behavior. The use of 2D social behavioral maps enables researchers to represent the complexity of social space in a simplified visual form, facilitating the identification of patterns and relationships. For example, a 2D social behavioral map of a park could include the location of benches, playgrounds, and walking paths, as well as the expected behaviors and social norms governing the use of the space. Additionally, individual characteristics of park visitors such as age, gender, and race, can also be included to provide a more comprehensive understanding of the social dynamics at play. A more detailed discussion of this topic will be presented in Chapter 3.

These concepts are important for understanding how people conduct themselves in social environments and have been extensively referenced by those developing algorithms that can simulate the behavior of humans in social spaces – which will be discussed in the next section.

2.2 Computer Simulation

The study of human behavior is essential for creating realistic crowd simulations for socially interactive agents. The concepts of gatherings, formations, proxemics and territoriality have been adapted into algorithms and computational models to analyze and simulate the behavior of humans in social spaces. These algorithms are used in computer simulations to model the structure and dynamics of social space and to identify patterns of behavior among humans in social contexts. By understanding the underlying structure of social space and how it is affected by the presence of other humans, algorithms can be used to generate realistic crowd simulations and social negotiation strategies for virtual agents. Furthermore, research has also been done to understand the motivations and preferences that influence the behavior of humans in social spaces, which can be used to create computational models allowing virtual agents to anticipate and respond to the behavior of others. This has enabled the development of more natural and engaging simulations of crowds of people.

What is Crowd Simulation?

Crowd simulation is the process of simulating the movement and the behavior of a large number of people [136]. The term *social crowd simulation* refers in particular to such simulation based on theories of human social behavior. These theories provide explanations for how human behavior is influenced by the behavior of other humans, as well as by the context in which the behavior takes place.

The Structure of this Section

This section is organized as follows. First, a few applications of crowd simulation are presented to give an overview of the field and highlight the importance of social behavior in crowd simulation. Then, the remainder of the section will provide an overview of the main components of crowd simulation. First, an overview of the main techniques available for *generating diverse crowds* of characters will be presented: heterogeneous crowds foster the creation of more realistic, immersive and believable simulations, promote inclusivity, and provide robust data for research. Then, a review of existing *computational behavior models* will be provided. Those are mathematical or algorithmic formalizations of theories, usually from sociology and psychology, designed to simulate the behavior of agents in a crowd. These models help researchers to better understand group dynamics and behaviors and are used to simulate how agents interact and behave in different situations. After that, the field of *navigation algorithms* will be explained. This includes global pathfinding and local obstacle avoidance, which are two important techniques used in navigation systems. The former is used to find the most efficient route from a starting location to a destination, while the latter dynamically re-plans the route to account for moving obstacles. These two algorithms work together to provide a reliable and safe navigation system for crowd simulation agents. Finally, the section will conclude with a discussion of the *calibration* and the *evaluation* of crowd simulation models. These are two essential steps in ensuring accurate and reliable results. Through the use of real-world data and comparison of outcomes,

Field	Description	Sources
Virtual Heritage	Historians have employed autonomous virtual people to bring the past back to life.	[92, 27, 18]
Entertainment	Crowds are used in video games and movies to fill backgrounds and enhance player mechanics.	[13, 102, 124]
Architecture	Inform architectural design, generate and simulate crowds in virtual cities, and optimize layouts of buildings.	[119, 46, 91]
Safety	Development of better safety protocols and design of safer buildings through emergency simulations.	[67, 107, 98, 68, 60]

Table 2.4: Some fields of application for crowd simulation.

it is possible to identify discrepancies between the model and reality and make adjustments to the parameters to improve the accuracy of the simulation. Ultimately, this process helps to ensure that models reflect the behavior of the crowd, and can be used to identify areas of improvement and suggest changes to the model.

2.2.1 Crowd Simulation Applications

Realistic crowd simulation has become a fundamental research topic in computer science due to its potential useful application in many different fields. This section presents some examples of how autonomous virtual people can be used to recreate historical settings, improve the entertainment industry, facilitate architecture design, and predict safety features, as summarized in Table 2.4.

Virtual Heritage. One example is the field of *virtual heritage*, where historians have employed autonomous virtual people to bring the past back to life. These simulations involve the accurate reconstruction of historical settings in complex 3D environments, and the recreation of the behavior of the people who lived in those settings. Some noteworthy cases of this application can be found in the literature and include: the simulation of ancient Pompeii [92] the reconstruction of a Roman Odeon in Turkey [27], and the reenactment of ancient Mesopotamia [18]. All of these works aim to reenact historical environments by employing crowd simulations. It can be noticed that a lot of resources were spent on implementing agent navigation at the expense of social behavior, which is often left as future work. Upon examining the limitations of these projects, it becomes evident that a framework is required that can alleviate the burden of addressing the complexities of navigation system intricacies, and instead focus on modeling essential social behaviors.

Entertainment. Crowd simulation has also helped the development of the *entertainment industry* concerning video games and movies. Beacco et al. [13] and O'Connor et al. [102] mentioned the fundamental role played by crowds in two very different video game genres. They can be used in sports games such as FIFA as passive entities to fill the background, or they can even be employed to empower the player with new game mechanics. For example, in the cases of Assassin's creed and Hitman, the player can blend inside the crowd and sneak around the environment [13, 102]. In terms of movie production Scott explained how most of the digital human motion in "The Lord of The Rings: The Two Towers" crowd scenes was created with a crowd simulation program called "Massive" [124].

Architecture. Another line of work is one of *architecture* where several projects have shown how crowd simulation can lead to better-informed design. One example presents a framework capable of generating and simulating crowds in virtual cities [119]. With this approach city planners could conveniently experience how people interact with a particular architectural design. Feng et al. developed a novel approach for designing mid-scale layouts optimized for human crowd properties. In their work, they optimize the layout of a shopping mall to accommodate human features such as mobility, accessibility, and coziness [46]. Moreover, Mathew et al. presented a system to generate a procedural environment that produces a desired crowd behavior [91]. This approach enables the definition of goals achieved by automatically altering the environment. All of the aforementioned examples were implemented employing custom software solutions that are often resource-consuming. For this reason, many architectural firms cannot access such solutions and need a convenient way to perform crowd simulation.

Safety. One last example of a field where crowd simulation has had a critical impact is on predicting *safety features*, where much effort has been put into modeling human behavior during emergencies such as evacuation scenarios [67, 107, 98, 68]. Realistic crowd simulation enables researchers and engineers to better understand human behavior during emergencies. By simulating the movements and actions of crowds, such as in the case of building evacuations, researchers can test and validate different safety protocols and evacuation plans, which can save lives in real-world situations. Additionally, crowd simulation can be used to predict the effectiveness of different design features, such as the placement of exits or the width of staircases, in emergencies. This can help architects and engineers design safer buildings and spaces. This interest has evolved into industry-leading software solutions that deal with evacuation safety in various scenarios. One such solution is Exodus which comprises a suite of software packages, tailored to the building, maritime, rail and aircraft environments. Exodus has been employed in several projects featured in the literature [26, 157, 134, 59] and, given the criticality of this field, it is continuously being validated [60].

All these applications would benefit from agents capable of social reasoning, as they make the simulation model more believable. This has sparked a growing interest in creating virtual humans capable of displaying social behavior.

2.2.2 Variety in Crowds

Figure 2.3: Different approaches to introduce variety: (A) Skeletal modification to vary height (B) Using accessories to mask similarities (C) Different walking animation cycles. image adapted from [138]

Generating diverse and variegated virtual characters is crucial for creating a realistic and believable crowd simulation. Real-world crowds are naturally diverse in terms of age, gender, race, and other characteristics, and generating diverse characters in a simulation helps to replicate this. Additionally, a diverse crowd can add a sense of depth, complexity, and context to a scene, and can make the simulation more interesting and engaging for the viewer. Representing a variety of characters in a crowd simulation also helps to avoid perpetuating stereotypes and biases, and instead create a more inclusive and representative simulation.

At the same time, creating heterogeneous characters for crowd simulations can be a challenging task due to a variety of complex, time-consuming, and cost-prohibitive factors. Complexity arises from the need to craft detailed, realistic, and expressive characters. Time-consuming tasks include the creation of multiple designs, textures, and animations. Technical limitations further complicate the process, as advanced techniques and software may be necessary to create a character with a wide range of variations. Additionally, the cost of these resources and services can be prohibitive. Furthermore, memory and storage capabilities may be inadequate to store characters with a wide range of variations in their appearance. Finally, consistency in style and appearance requires a good understanding of design principles and a high level of skill in computer graphics. Ultimately, creating diverse and variegated characters in computer graphics is a complex and difficult endeavor. But, despite the challenges, with the right resources and techniques, it is possible to create characters with a wide range of variations and customizations. Thus, this section discusses some of the approaches available for generating diverse and variegated crowds of virtual characters, such as appearance variety, accessor diversity, and animation customizations (see Figure 2.3), as well as the challenges that remain for the future of this field.

Human Templates

The work of Thalmann et al. provides a good overview of the techniques and tools that can be employed to address these generation challenges [138]. The authors

Technique	Description	Sources
Appearance Variety	Dynamic modification of the character's shape and colors by using a human template skeleton and scaling it.	[89, 135, 36, 55]
Accessory Diversity	Adding accessories (e.g., watches, hats, backpacks) to the human template original mesh to achieve shape variety.	[29]
Animation Customizations	Generating many locomotion and idle cycles, introducing upper-body variations, adding procedural modifications.	[49, 109, 132]

Table 2.5: Techniques to achieve variety in a crowd.

argue about the infeasibility of achieving character variety through traditional means, such as designing and modeling as many virtual characters as there are people in the crowd. They explain that a more reasonable approach is to use *human templates* defined by the characters' skeleton, mesh, and set of textures. These templates can then be instantiated multiple times while using different components dynamically varied by applying color and shape techniques. Moreover, to generate many locomotion animations for each human template, a motion-capture-based locomotion engine can be used, as well as an *inverse kinematic* system to apply upper body variations to the characters. The generated animations can then be applied to meshes that are deformed at runtime. Through this method, large crowds with varied individuals can be generated. Their work is particularly focused on three aspects, summarized in Table 2.5, of character generation: *appearance variety*, *accessory diversity*, and *animation customizations*.

Appearance variety. One way to achieve appearance variety is to dynamically modify the shape of the human template [89, 138]. This involves modifying separately the height of the human body and its shape by using a human template skeleton and scaling it, as shown in Figure 2.3 (A). Naturally, different body parts have different proportions, and the authors propose a method to scale the human template skeleton in a way that preserves them. This is done through a *FatMap*, painted for a given template, which is used to emphasize body areas that store fat. The FatMap values are then used to automatically infer one value for each vertex of the template's mesh. The scaling direction of each vertex is then computed as the weighted normal of the bones influencing it, and the extent to which the body is scaled is defined by a fat scale. This approach allows for mixing various attributes such as body height (short or tall) and size (narrow or wide).

A different method to achieve appearance variety is to modify the color of the various components of the human template, such as the skin, hair, and clothes of the character [135, 36, 55]. Previous work has put forward the idea of dividing a template into several body parts, identified by specific intensities in the alpha channel of the texture. However, when dealing with characters close to the camera, this approach has several drawbacks, such as sharp transitions between body parts.

To overcome these issues, it is possible to employ a method based on segmentation maps. Through its channel intensities, this method allows for each texel to partially belong to several body parts at the same time, thus enabling smoother transitions between body parts [138].

Accessory Diversity. Accessories represent a simple and efficient alternative to costly human template modeling [138]. Accessories represent small meshes that can be easily added to the human template original mesh, ranging from subtle details such as watches and jewelry to larger items like hats and backpacks. By distributing accessories to a crowd of human templates, each instance can be made unique in its shape, as shown in Figure 2.3 (B). These accessories are attached to a skeleton and follow its animation when moving, and can be rendered at the correct position and orientation accordingly to the movements of the character. Depending on the type of accessory, slight modifications of the animation sequences may be required, such as blocking joint movements or applying procedural modifications at runtime. For more complex accessories, such as a cellphone, pre-process modifications may be necessary using an inverse kinematic tool. This approach was employed by Ubisoft to create a large crowd of characters for the game *Assassin's Creed* [29], where a limited number of human models were equipped with different hats to give the impression of a large crowd. Overall, accessories offer an efficient way to vary the shape of a crowd of human templates with limited resources and computational efforts.

Animation Customizations. Another important aspect to consider while striving for variety in a crowd is the animation of the characters, as shown in Figure 2.3 (C). The work of [138] mentions three techniques that allow for variation in the animation of characters: (1) generating a large set of locomotion and idle cycles that are categorized by type; (2) introducing upper-body variations such as having a hand on the hip or in a pocket; and (3) adding procedural modifications to locomotion animations to allow characters to wear complex accessories dissimulating similarities. They also explain how a particular locomotion engine can be used to generate walk and run cycles, which can then be modulated by personification weights, speed, and locomotion weights [49]. Another approach for introducing variation is to blend different keyframe animation clips by interpolating the values of each joint based on a set of parameters. This can be done with animation *blend trees*, as explained in [132]. A more involved approach proposed by Pelechano et al. is to use an *animation planning mediator* (APM) to synthesize animations efficiently for virtual characters in real-time crowd simulation. The APM selects appropriate animation clips, modifies the skeletal configuration of each character to eliminate foot-sliding and restrict upper body torsion, and blends animations using a hardware-accelerated character animation library. By introducing this variety into the mix, virtual humans walking together with different locomotion styles and speeds add to the realism of the simulation.

So far, this section has presented a variety of techniques that can be used to achieve variety in a crowd of virtual humans. The remainder of this section will

discuss several practical tools that can be used to implement these techniques. A summary can be found in Table 2.6.

Tool	Description	Sources
MakeHuman	Open-source 3D character creation suite with customizable features, tools for poses and animation, and export functions.	[2, 19]
Reallusion Character Creator	Powerful 3D character creation tool with tools for customization with a wide library of pre-made characters and add-ons.	[4, 77]
Unreal Engine MetaHuman	Powerful framework for creating photorealistic digital humans for video games, movies and more.	[5, 42, 25]
Unity Multipurpose Avatar (UMA)	Free character creation system for dynamically creating varied characters through adjustable DNA values, custom colors, and a wardrobe of items.	[3, 120]
AI Synthesizers	Machine learning based solutions for creating 3D avatars and characters from 2D images.	[155, 86, 140, 40]

Table 2.6: Practical tools to achieve variety in a crowd.

Static Character Creation Frameworks

Since designing and creating a large number of characters from scratch is a time-consuming and costly endeavor, several frameworks have been developed to expedite the process. Among these, static character creation frameworks are powerful tools that help developers quickly generate virtual humans. These tools provide a wide range of options for customizing the appearance of characters, from body shape and skin tone to hairstyle and facial features. Additionally, they include tools for rigging and animating the characters, making it easy to use them in a variety of settings. The resulting models can then be exported in a variety of standard formats for 3D meshes (FBX, OBJ, etc..) to be used in other applications.

Some examples of this category of frameworks include MakeHuman, Reallusion Character Creator, and Unreal Engine MetaHuman. By using these frameworks, developers can quickly create detailed characters in a fraction of the time. This section will provide an overview of these frameworks and discuss their advantages and disadvantages.

MakeHuman. MakeHuman [2] is an open-source character creation suite developed in Python that allows users to create 3D human characters with a variety of customizable features such as gender, age, ethnicity, muscle, weight, proportions and height. It allows users to edit macro and detail attributes such as skin, hair, eye shape, finger length, clothing, facial expressions, material properties and more.

The suite provides tools for poses, animation cycles, and 21 human measurements that can be adjusted, including height, chest, waist and hip circumference, and neck circumference and height. It uses fuzzy set theory, weighted vertices and a linear combination of rotation and translation morphing to deform the model. MakeHuman's main parameters are divided into groups, and when one parameter is modified, a specific group of modifiers is applied. The MakeHuman project is developed by a community of programmers, artists, academics, and enthusiasts interested in 3D computer models of human beings and allows for exporting to other software for further refinement. One drawback is that the base human meshes produced by MakeHuman are not of a high enough quality to be on par with the meshes created by other frameworks that meet today's standards. For a more detailed description of the framework, refer to [19].

Reallusion Character Creator. Reallusion Character Creator [4] is an advanced 3D character creation software designed to produce realistic 3D characters for use in a variety of digital environments. This software provides users with a comprehensive suite of tools to customize facial features, body type, clothing, and other characteristics to create a unique 3D character. It also includes a library of pre-made 3D characters and props, which can be used to create a scene or environment, as well as a range of physical and emotional expressions which can be used to animate the character and make it appear more realistic. Character Creator offers a selection of add-ons to expedite the character design workflow, provides access to a character library made by professional designers and 3D artists, and can be easily integrated with game engines such as Unreal and Unity. Finally, it includes a character reduction feature that can optimize characters for large crowd simulations through advanced level of detail mesh simplification techniques. Overall, Character Creator is a relatively good option for those looking to design realistic 3D characters for use in digital projects, such as [77], where it can provide a comprehensive set of tools to quickly and easily generate high-quality 3D characters. On the other hand, it is expensive, and it is not open-source, which may be a drawback for some users.

Unreal Engine MetaHuman. MetaHuman Creator [5] is a novel browser-based application developed by Epic Games, the company behind the Unreal Engine, that enables users to quickly generate photorealistic digital humans, fully rigged and complete with hair and clothing [42, 25]. MetaHuman Creator can be used in combination with motion capture and animation technology to create realistic movements for video games, movies, television, and other interactive scenarios. The workflow allows users to choose from a range of preset faces, as well as modify teeth, bones, and other features of the virtual character. It is also possible to blend existing examples in the library, to easily craft novel characters without much effort.

Metahuman strives to bridge the gap between real and virtual characters, by creating lifelike digital characters that can interact in real time. While the discomfort experienced when encountering an artificial or humanoid entity that is almost, but not quite, convincingly human-like, known as the "uncanny valley", is still present, the project is a step in the right direction towards achieving a level of realism that

can be accepted and enjoyed by the consumer.

Dynamic Character Creation Frameworks

Dynamic character creation frameworks are similar to their static counterparts, but they allow for more flexibility in the customization of characters, especially at runtime. This means that the user has access to a character template and can define several parameter ranges for each feature of the character, such as body characteristics as well as clothing, accessories, and colors. These ranges are then queried at runtime to generate a unique crowd of characters, which are different at each iteration. One example of this type of framework is the Unity Multipurpose Avatar (UMA).

Figure 2.4: UMA Character with some of its DNA values.

Unity Multipurpose Avatar. UMA [3] is a free character creation and modification system for Unity which allows creating crowds of virtual characters displaying varied traits and features. The system comes with standard male and female human models, composed of a fully rigged 3D mesh and the relative textures. These models are tied to the respective standard DNA definitions, which define the baseline sizes for each bone group in the 3D mesh, as shown in Figure 2.4. It is then possible to deform the mesh by altering the DNA values, thus creating characters with varied shapes and sizes. Additionally, it is possible to define a custom set of colors that are going to be applied to the skin, hair and eyes. Finally, another way to introduce variety in the UMA crowds is to create a custom collection of clothes called a *wardrobe*. These items are 3D meshes rigged on the same skeleton of the UMA character, and so they adjust to the DNA modifications as well. The crowds can then be dressed with different items from the collection so that, even if two

characters share the same DNA and colors, they will still appear different. It is worth noting that any 3D mesh can be part of the collection. This includes any gadget that one desires to give to the character, and in fact, even hairs are part of the wardrobe - so the characters have many hairstyles.

There are two main ways of working with UMA characters: either through the *Dynamic Character Avatar*, or the *Random Generated Character*. The first method allows instantiating one single character and manually giving them a static set of features (DNA, colors and wardrobe) at design time – which will always be the same across multiple simulations. The second way allows instantiating a population of characters based on a randomizer. The randomizer is a probability distribution of features bound to limited ranges. So, for example, it is possible to specify that the population should be between 1.5m and 2m tall; should have blonde hair 20% of the time; or have 10% probability of wearing glasses. It is then possible to specify how many characters to generate with this randomizer, and they will be instantiated when the simulation starts. For this reason, the population is going to look different every time a new simulation is performed.

UMA has been employed in several projects, such as [120], where it was used to generate a crowd of 3D characters for testing collision avoidance algorithms.

AI-based synthesizing

The advent of machine learning has opened up a range of promising possibilities in computer graphics, including the generation of fully rigged 3D characters from images. This technology offers several benefits in the field of crowd simulation, allowing the creation of a larger, more varied population of 3D characters than would otherwise be possible. One of the main advantages of machine learning-based character generation is the efficiency with which it can be achieved. Through the use of deep learning algorithms, a database of images can be used to create a large population of 3D characters with unique facial features and body types. This eliminates, or at least reduces, the time and effort required to manually construct each character, leading to a much faster and more efficient process. This approach can be both scalable and flexible: once a machine learning model has been trained, it can be used to generate any number of 3D characters, making it easy to create large numbers of characters quickly and efficiently, and it can be adapted to work with different types of images and 3D models, making it easy to create a wide range of characters with different styles.

Although the generation of 3D characters from images through machine learning is a relatively novel field, and many aspects can be improved, it has already been successfully incorporated into several different applications. Some examples of academic research applications include the works of Zhang et al. and Li et al., both offering pipelines for the reconstruction of 3D human avatars and characters from a single image employing Generative Adversarial Networks (GANs) [155, 86]. Moreover, there are several commercial applications of this technology, such as AvatarSDK [40] and Wolf3D [140] that offer a variety of tools for the creation of 3D characters from images.

This section has discussed various approaches to the creation of 3D characters for use in crowd simulation, along with the related practical tools and frameworks. These techniques can be used to introduce appearance variety into the crowd, both at the visual level and at the animation level. However, what ultimately makes crowds believable is how the individuals behave, so the next section will discuss the state of the art of computational models that have been employed to implement human behavior theories in crowd simulation.

2.2.3 Computational Behavior Models

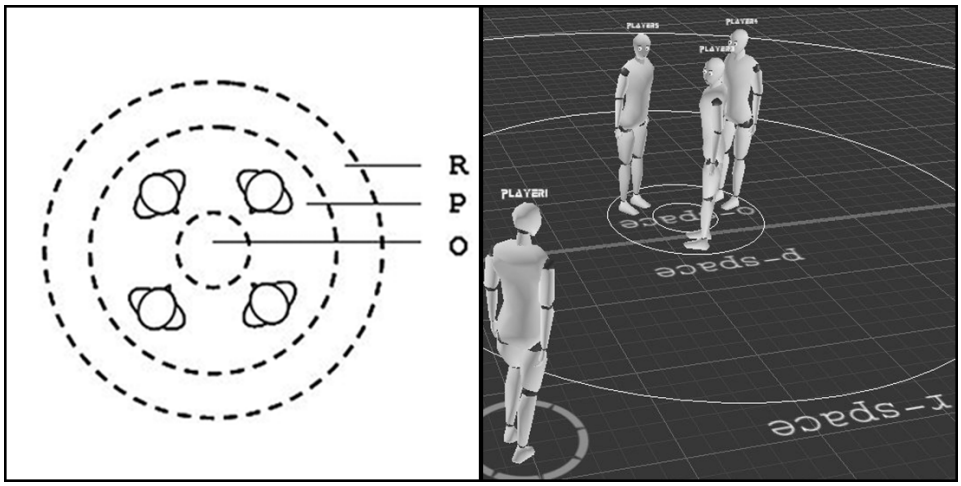

Figure 2.5: Left: main territorial fields of Scheflen's classification. Right: corresponding implementation by Pedica and Vilhjálmsson [105].

Crowd simulation involves using computer-generated agents to realistically simulate the motion of people and other entities in a virtual environment. Recent advancements in this field have enabled the creation of increasingly convincing simulations of human motion in 3D space. This is achieved by extending the underlying motion simulator with new and improved theories of perception and movement. This section discusses the state of the art of computational behavior models for social theories that have been employed in crowd simulation, as well as the challenges that remain for the future of this field. Each model has its theoretical foundation in the social sciences, discussed in section 2.1, which is implemented in a computer simulation model. A summary of the main categories is provided in Table 2.7.

Social Group Models One set of theories studies the formation of groups of individuals and regulates how people deal with aggregate motion. A possible way of modeling *group behavior* is presented by Li and Lin through the use of graph theory and social networks analysis [85]. The authors perform an initial agent-based crowd simulation without considering social factors. Afterward, they

Category	Description	Sources
Social Group Models	Forming groups among individuals and how people manage collective movement.	[85, 12, 101, 76]
Social Emotion Models	Modeling emotion contagion, panic, and hierarchical structures in emergencies.	[90, 151, 83, 45]
Social Activity Models	Models of social crowds have been enhanced by artificial life simulations that capture daily life scenarios.	[129, 70]

Table 2.7: Some useful categories of social theories and corresponding models employed in crowd simulation.

create a social network based on the interactions that happened between agents during the simulation. The resulting social network can later be used to create a crowd simulation accounting for social connections among agents. Pertaining to navigation, Bayazit et al. employed Rule-Based Roadmaps for better group Path Planning [12]. They propose to extend navigation meshes, usually only employed to compute trajectories, with additional group behavior information. This means that whenever a group member reaches a particular position in the road map, the system triggers a specific behavior such as herding, leader-following and so on. Niederberger and Gross proposed a system that provides mechanisms for group behavior and group definitions [101]. With their approach, it is possible to diversify background character behaviors, by extending and combining generic behavior definitions. Pedica and Vilhjálmsson implemented Kendon's social theory of F-formation for agent conversation into the simulation of virtual humans. As shown in Figure 2.5, the authors employed the concept of human territories to model how humans behave in and around group conversations. Group members can invite others into their formation through eye gaze, positioning and orientation. This model was later extended by Karimaghalou et al. with group-leaving and group-revisiting mechanisms based on habituation and boredom theories [76].

Social Emotion Models An important aspect of human behavior is *emotion contagion* which describes how people's emotions are influenced by those around them. For this reason, Mao et al. extended a crowd simulation system by incorporating agents' personality through a five-trait model, often referred to as *Big Five*, and employed an emotion group contagion model based on thermodynamics heat dissipation called ASCRIBE [90]. Faroqi and Mesgari brought this concept to emergencies, modeling the effect of panic during evacuations [45]. Other approaches investigate the impact of groups in emergencies, for example, by equipping agents with the following behavior [151, 125], or by modeling the formation of hierarchical structures [107, 73, 111]. Furthermore, Lee et al. uses machine learning to extract group trajectories from videos in various emotional settings, learn their patterns, and replicate them with virtual agents [83].

Social Activity Models Social crowd models have also been enhanced by the inclusion of daily life scenarios, sometimes referred to as *artificial life*. One example is the pedestrian simulation by Shao and Terzopoulos who replicated New York's Penn Station and modeled the behavior of hundreds of people [129]. The pedestrians were driven by a high-level cognitive control, in charge of deciding destinations and activities such as sitting to rest, watching performers, chatting with friends and queuing up at vending and ticketing machines. This decision mechanism was then coupled with reactive local navigation controls, to ensure smooth traversal of the environment. The result was an impressive showcase of a crowd in a complex environment, but on closer inspection, the behaviors were individualistic and lacked social awareness. Another example is the work of Huang and Terzopoulos focusing on a framework for simulating Doorway Etiquette [70] based on three main factors: (i) *effort*, a person is more likely to hold doors for another that does the same; (ii) *care*, it is more compelling to help people in need; and (iii) *emotion*, based on kindness and rush. These factors are encoded with a Bayesian network, which yields the appropriate door-holding behavior – realized by a locomotion system. This system was shown to produce convincing real-time behaviors for small crowds, of up to 16 people.

In summary, the examples of computational models above can be classified into three main categories: (i) *social group models*, which focus on the formation and behavior of groups in collective movement; (ii) *social emotion models*, which address the influence of emotions, panic, and hierarchical structures, particularly in emergencies; and (iii) *social activity models*, which enhance crowd simulations by incorporating artificial life scenarios and daily life activities. To be run in a simulation, these models need to be implemented in a way that can be executed by the agents. This usually requires agents to be capable of traversing the environment and possibly avoiding or interacting with other agents. The next section will discuss the main techniques for navigation and obstacle avoidance in crowd simulation.

2.2.4 Navigation Algorithms

As a technical layer, on top of which higher-level social models are commonly built, a navigation algorithm is essentially a solution for making an agent successfully traverse an environment, from a starting position to a destination position, without getting stuck. Such algorithms can be classified into *local* and *global* approaches, according to the amount of information that is being processed. Global navigation computes a complete path leading the agent to its destination, but this can be resource expensive because it needs to account for all possible trajectories [6]. On the other hand, local navigation only works with the general direction to the destination and tries to guide the agent moment-to-moment considering its immediate surroundings. In practice, these approaches are often used in combination: global path planning is performed once considering only static obstacles, and then the agent employs local obstacle avoidance to steer around the environment.

Global Path Finding

This section gives an overview of the main techniques available for finding routes between a source point and a destination and is summarized in Table 2.8. For an in-depth review, refer to [6].

Technique	Description	Sources
Uniform Grid	Superimposes a grid over the environment to generate paths via traversing cells.	[153, 64, 104, 128]
Visibility Graphs	Visibility graph approach stores vertices in a graph and computes line of sight to find optimal paths.	[48]
Waypoint Graph	The environment is subdivided into regions and points are connected in a graph, which can be searched to find navigation trajectories.	[148]
Navigation Mesh	Graph of convex polygons that define traversable areas. Agents navigate within polygons directly and use graph search for pathfinding.	[31, 108, 113, 139, 114]
Cellular Automata	Simulates pedestrian movement using a centralized entity that moves agents among cells with strict mechanics.	[16, 21, 17]

Table 2.8: Global pathfinding techniques.

Uniform Grid. One global path planning approach is to superimpose a *grid* over the environment, and then produce paths by traversing the cells [153]. The grid cells can be of several shapes [64, 104, 75] and sizes [128], and this affects the number of adjacent cells. Traversing any two connected cells has a uniform cost, which simplifies pathfinding. The most evident limitation of this approach is the difficulty of matching a grid over the environment. This drawback makes the approach suitable mainly for rather simple topologies.

Visibility Graphs. Another approach is called *visibility graph*, which consists of saving the vertices of the geometry in a graph, and then computing whether there is a line of sight between any two vertices [48]. If such a line exists, agents can navigate between them without encountering any obstacles. The graph can be searched to retrieve an optimal path between a source and a destination. This approach can suffer from memory overhead for complex environments.

Waypoint Graph. A similar approach, the *waypoint graph*, consists of processing the geometry to find the most relevant points in the environment. For example, one could subdivide the environment into regions, voxelize the geometry, and use corner detection to generate waypoints [148]. Then, these points are connected in a

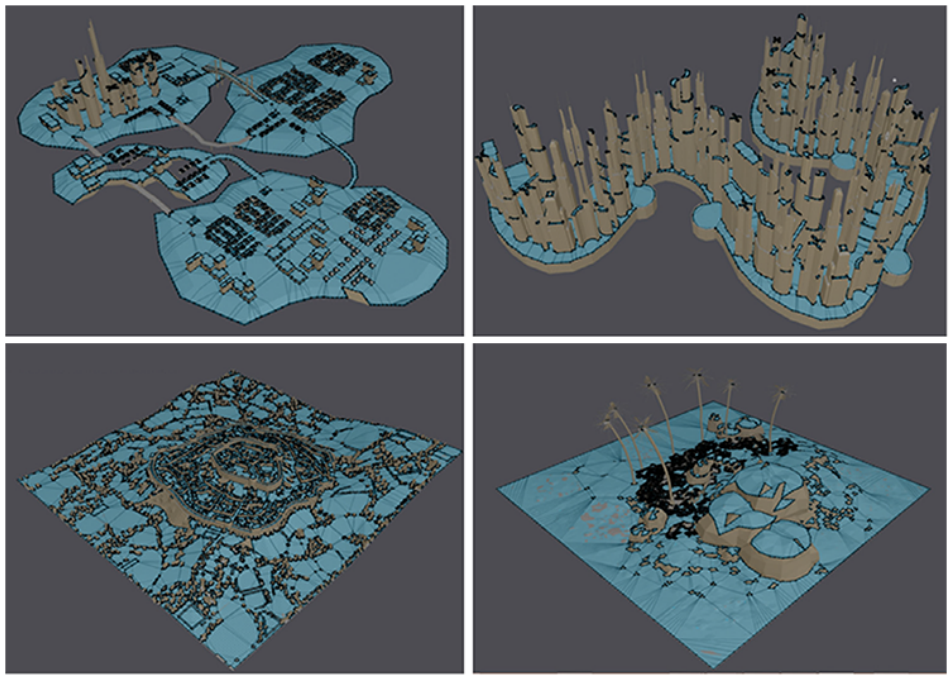

Figure 2.6: Navigation meshes of different environments, as shown in Rahmani and Pelechano [114].

graph, which can be searched to find navigation trajectories. This approach is very flexible and can produce both surface and volumetric motion, but still suffers from time and memory constraints, along with repetitive paths – agents always traverse the same lines.

Navigation Mesh. The most popular approach for global agent navigation is represented by *navigation meshes*. A navigation mesh is a graph of two-dimensional convex polygons that defines which areas of an environment are traversable by agents, as shown in Figure 2.6. Agents can navigate between any two points inside a polygon in a straight line because it is convex and traversable. On the other hand, pathfinding between different polygons in the navigation mesh can be done with one of many graph search algorithms available, such as A* [31, 30]. The concept of navigation mesh dates back to the field of robotics and autonomous navigation, where it was necessary to create a representation of the environment that allowed robots to traverse their surroundings. For this reason, much effort was put towards dividing space into convex polygons that were directly traversable by agents - they were first called meadow maps [10]. It was later adopted by the artificial intelligence community, especially in games that needed agent navigation. The navigation mesh approach evolved in three main directions: (i) improving graph search for path finding [108, 113, 114], (ii) designing techniques for automatic navigation mesh

generation [110, 15], and (iii) supporting dynamic navigation meshes [139].

Cellular Automata. Another approach for simulating pedestrian movement is Cellular Automaton (CA) [16, 21, 17]. While agent-based techniques distribute the decision effort among all the agents, cellular automaton uses a single centralized entity that decides how to move every agent. This method can be compared to a checkerboard where agents are pawns and the CA is the one moving them among cells. Agents have very strict mechanics that allow them to move among cells, and the CA can adopt two kinds of navigation strategies: (i) sequential where agents are moved one at a time, and (ii) parallel where they can be moved all at once. The first strategy is simpler but less realistic, while the second one needs to account for conflicts – two agents cannot occupy the same cell. Since this is a rule-based approach and does not require any advanced perception system from agents, it is less resource intensive compared to the others.

This section provided an overview of the most popular approaches for global navigation. However, they all only account for static environments and do not consider the dynamic nature of other agents and obstacles. The next section will discuss the techniques that were developed to address local navigation and obstacle avoidance.

Local Obstacle Avoidance

This section gives an overview of the existing techniques for preventing agents from colliding with obstacles, with a special focus on pedestrian motion. A summary of the techniques is shown in Table 2.9.

Technique	Description	Sources
Steering behaviors	Set of rules for agents to navigate complex environments using simple forces and heuristics.	[116, 117, 118]
Velocity Obstacles	Avoiding collisions in motion planning by predicting future velocities of other agents.	[47, 142]
Social Force Model	A model for simulating pedestrian movement based on physical and social forces.	[66]
Vision-based Steering	Uses low-resolution cameras and shaders to detect and avoid collisions by adjusting angles and speed.	[103]
Machine Learning	Navigation is modeled as a reinforcement learning problem. Agents are rewarded for reaching a goal, and punished for collisions.	[152, 50]

Table 2.9: Local navigation techniques.

Steering Behaviors. One of the first efforts towards local navigation for realistically moving agents through their surroundings is called *steering behaviors* and is described in the work of Reynolds. In his early work Reynolds proposed a way of simulating the movements of agents, which were named *boids*, similar to schools of fish, flocks of birds, or herds of other animals [116]. The author equipped each boid with three basic components, namely *collision avoidance* of nearby agents, *velocity matching* and *flock centering* to simulate the cohesion of group members. Each boid has a simple perception model which allows it to sense its surroundings, mimicking the imperfect senses of animals. With these simple components, the author was able to quickly compute the velocity vector for each agent and convincingly simulate their movement. This line of work later evolved into avoidance of static and dynamic obstacles. Reynolds describes several approaches to steering agents around obstacles based on *force field* reflection, *curb feeler* deflection, or even *silhouette* following [117]. Finally, Reynolds divided agent locomotion into three layers which go from high to low level: *action selection*, *steering* and *locomotion*. The work focused on the second layer, steering, simplifying the underlying locomotion layer assuming a vehicle-like agent moved by a single velocity vector [118]. Reynolds led the mathematical foundation for computing a velocity vector that simulates how agents move during a plethora of behaviors such as: while seeking, fleeing, avoiding obstacles, following a path or another agent.

Velocity Obstacles. A different technique for agent local navigation is by computing *velocity obstacles*. A velocity obstacle (VO) is the set of all velocities of an agent that will result in a collision with another entity, either static obstacles or other moving agents [47]. One way to traverse the environment using this approach is by first selecting a velocity vector that's heading in the general goal direction. After that, it is possible to adjust the velocity by selecting vectors that are not in the set of the velocity obstacle field to avoid collisions. Many improvements have been proposed to the basic approach such as Reciprocal Velocity Obstacles (RVO) [142], which refines trajectories by considering the mutual responsiveness of agents in collision avoidance. RVO acknowledges that other agents in the simulation are also actively trying to prevent collisions, and it leverages this information to generate more effective and coordinated trajectories through anticipatory behavior.

Social Force Model. A particular technique for simulating the motion of human-like agents based on social factors was first described in the *Social Force Model* [66] by Helbing and Molnár. This approach models agent navigation as if it was under the influence of several competing forces. As such, the final velocity vector is computed as follows: $F_a = F_p + F_{int}$ where F_p is the agent's preferred velocity to reach its destination. F_{int}, on the other hand, is the interaction force consisting of repulsive and attracting factors based on the psychological tendency to keep a social distance between pedestrians, while also avoiding hitting walls, buildings, and other obstacles.

Vision-based Steering. Another approach for local steering relies on *synthetic vision* [103]. This technique consists of equipping each agent with a low-resolution

camera, through which they can see a simplified model of their surrounding. Using shaders, it is possible to analyze the pixels and determine whether they are on a collision course with any obstacles. This computation is based on two factors: an angle α and a time to collision *ttc*. Based on the outcome, the agents re-orient themselves and decelerate, to avoid the collision.

Machine Learning. Finally, in recent times there have been efforts to perform autonomous navigation through *machine learning*. Most approaches have been employed in the context of robot navigation, such as [152] which presented an autonomous navigation system capable of operating in densely populated environments and utilizing information of social groups. Group-Navi GAN incorporates a deep neural network to track social groups and join the flow of a social group to facilitate navigation. A collision avoidance layer ensures navigation safety. The method generates socially compliant behaviors in line with other state-of-the-art methods and is capable of navigating safely in a densely populated area, following the crowd flows, to reach the goal. Some techniques, such as the work of Godoy et al., have been employed for virtual agents. They present the Adaptive Learning for Agent Navigation (ALAN) framework [50], for the decentralized navigation of multiple agents in a crowded space. By leveraging techniques from machine learning and game theory, agents using this framework dynamically adapt their motion depending on local conditions in their current environment. This approach models navigation as a reinforcement learning problem: agents explore an action space made of preferred velocity vectors and are rewarded for reaching their goal, while punished for colliding with obstacles.

To summarize, many different technical solutions have been used to make agents traverse their surroundings. Research in this area has particularly pushed techniques that balance movement accuracy with the cost of computation. Advancements in navigation techniques have supported crowd simulation research, by making it easier to model more complex and realistic movements. The produced models often rely on a set of parameters that need to be calibrated to produce more realistic results. The next section presents an overview of how different models can be calibrated to produce results that are in line with real-world data.

2.2.5 Calibrating Model Parameters

Every simulation model aims to emulate a specific process. The way these processes unfold is influenced by many factors that may be variable, or even unknown while designing the model. For example, meteorological simulation models need to account for air pressure, wind strength, and many other factors. For this reason, simulation models are often made parametric to emulate various instances of the process, without changing the entire model. Crowd simulation is no exception. As it tries to emulate something as complex as human beings, models need to capture many of their features through a range of parameters. Typical agent parameters in crowd simulations include walking speed and acceleration, turning velocity, and field of view. While all these parameters create an opportunity to represent individual

differences or even different social contexts, they need to be specified accurately. Thus, there is a clear need for a method that finds the best value for every parameter of the simulation, for producing acceptable results. This section presents an overview of the different techniques that have been used to calibrate model parameters and is summarized in Table 2.10.

Technique	Description	Sources
Expert Opinion	Calibrating parameters with expert opinion to produce results close to reference data.	[53]
Search Formulation	Calibration can be modeled as a search problem, using search algorithms to find a solution.	[11]
Simulated Annealing	Simulated Annealing combines exploration and exploitation to overcome local minima in search formulations.	[81, 20, 7, 9]
Evolutionary Algorithms	Find the optimal set of parameters for a model by iteratively evolving a population of candidate solutions.	[24]

Table 2.10: Parameter calibration techniques.

Expert Opinion. The most trivial way of calibrating the parameters of a simulation is to adjust them by comparison. With this approach, it is possible to tune the parameters so that the simulation produces results as close to reference data as possible, based on an expert opinion. Expert opinion can be valuable in situations where data is scarce or unreliable, or where the model is being used to make predictions about a complex system that is difficult to fully understand. In such cases, expert opinion can provide additional insights and knowledge that can be used to improve the model's predictions. To calibrate a model using expert opinion, the first step is to identify the key parameters that need to be adjusted. These parameters will typically be based on the model's input variables, such as demographic data or environmental factors.

Once the key parameters have been identified, experts can be consulted to provide their opinions on the values that should be assigned to these parameters. Expert opinions can be gathered through various methods, such as surveys, interviews, or focus groups and the model's parameters can be adjusted accordingly. This method is best applied on a small number of parameters [53].

Search Formulation. When the number of parameters to tune grows, more sophisticated techniques are needed. One possibility consists of modeling calibration as a search problem: the state space consists of all the possible outputs of the simulation, and to go from one state to the other the system changes parameters and runs the simulation to produce a new output. By defining a distance metric that evaluates simulated against reference data, it is possible to define a goal state

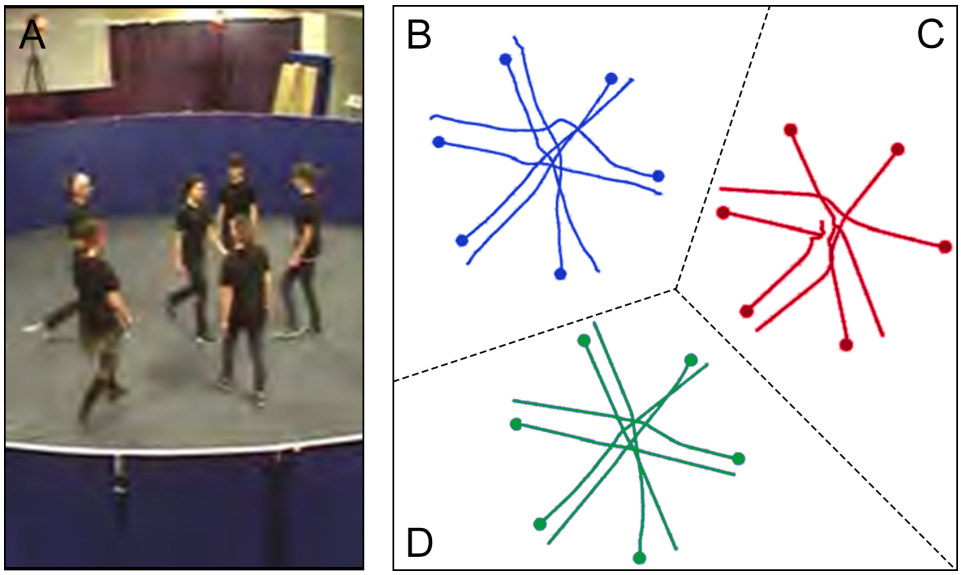

Figure 2.7: Parameter calibration applied to crowd data. (A) real-world scenario, (B) extracted trajectories, (C) *non-calibrated* simulation, (D) *calibrated* simulation. This image is adapted from the work of Wolinski et al. [149].

such that the metric has its global minimum value. It is then clear that this problem can be treated with one of the many search algorithms [11]. Since the state space is very wide, oftentimes greedy search algorithms are used to quickly get a solution. These algorithms only accept actions that increase the estimation function – as such they can get stuck in local minima.

Simulated Annealing. One approach which helps overcome local minima is to combine *exploitation* of the most promising solutions, with *exploration* of the entire problem state space. One such technique is *Simulated Annealing* (SA) [81, 20, 7, 9]. This approach has its roots in metallurgy where to strengthen materials they are heated up and cooled down multiple times, at decreasing temperatures. This procedure allows molecules to re-arrange themselves in better patterns – yielding a stronger metal. When the temperature is very high, molecules are freer to change the configuration, while at lower temperatures they are more constrained by physics. The same concept is applied in SA for optimization problems. Starting with a simulated high temperature the algorithm is willing to accept configurations that are worse than the current one – that way it can explore the state space and find the most promising areas. The temperature is then incrementally lowered so that the algorithm converges to a near-optimal configuration.

Evolutionary Algorithms. One additional approach is represented by *Evolutionary Algorithms*. These can be used to calibrate the parameters of a model by iteratively searching through the parameter space in order to find the optimal set of

parameters. The algorithm begins by generating an initial population of candidate solutions, which are evaluated for fitness using an objective function. Subsequently, a selection operation is performed to select a subset of the population for reproduction. Variation operators such as mutation and crossover are then applied to the selected individuals to generate a new population of candidate solutions. This process is repeated until convergence criteria are met, at which point the best solution found is output. By iteratively evolving the population through this process, evolutionary algorithms can effectively search the parameter space for good fits to the available data.

An example is the work of Jin and Bhanu which discusses how to optimize a navigation algorithm (RVO) so that it gets as close to reality as possible. The authors employ a genetic algorithm to find the set of parameters that makes RVO best resemble the trajectories of a real data set. To do this they need to use a metric that measures the similarity between two trajectories called Edit Distance Metric on Real Sequence (EDR) [24]. The dataset was divided into training and test sets. Applying a genetic algorithm with EDR as a fitness function to data points of the training set, the authors were able to systematically mutate RVO parameters - one at a time with the others fixed - and obtain the best possible set. Experiments conducted on the test set show that this calibration procedure greatly helps to make RVO resemble real-world trajectories.

From this overview of calibration techniques for simulation models, it is clear that many approaches exist. It is worth mentioning that there have been efforts towards unifying many of these approaches. One example is the work of Wolinski et al. that created a framework capable of optimizing the parameters of a crowd simulation using many different techniques [149]. The results are very promising, as the tuned simulation is much closer to reference data compared to the non-calibrated one as shown in Figure 2.7. Nevertheless, calibration is commonly a rather overlooked step in crowd simulation design – even if it can greatly improve the model's accuracy. This is partly caused by the difficulty of choosing the right calibration technique, and the overhead caused by implementing it into a wide variety of underlying systems. There may therefore be an opportunity here to provide a better-defined and more system-independent way to access a range of useful calibration techniques.

Once the parameters of a simulation have been calibrated, it is important to evaluate how the simulation captures the behavior of the real world. To this end, several approaches can be used to evaluate the accuracy of a simulation with respect to real-world data. The next section will discuss some of these approaches, and will also provide an overview of the most common metrics used to evaluate crowd simulation.

2.2.6 Evaluating Simulation Output

With increasing focus on the realism offered by crowd simulation, there is a growing need to validate them against real-world data. Crowd simulation evaluation can be divided into two main approaches which are *Macroscopic* and *Microscopic* evaluation. The first one treats the crowd as a whole entity and estimates metrics such as

mean speed, densities, collision or path length. On the other hand Microscopic evaluation reasons in terms of individual agents and/or trajectories, and estimates the similarity between simulated and reference data points. These techniques are summarized in Table 2.11.

Technique	T	Description	Sources
Edit Distance Metric	S	Calculates the similarity of two trajectories by computing the cost of turning one set of points into the other.	[24]
Entropy Metric	M	Objectively evaluates the similarity between simulated and reference data by measuring the amount of information missing from the simulation.	[58]
Trending Paths	M	Unsupervised clustering is used to learn sets of trending paths from reference data to objectively quantify crowd simulator realism.	[146]
Fundamental Diagram and ANCOVA	L	Macroscopic metrics used to compare pedestrian velocities and crowd densities of simulated motion with real data.	[127, 126, 156, 150]

Table 2.11: Simulation evaluation techniques. The "T" column indicates the type: S for small-scale or Microscopic, M for medium-scale or Mesoscopic, and L for large-scale or Macroscopic.

Entropy Metric. Guy et al. propose a way of objectively evaluating the similarity between simulated motion and reference data [58]. The metric is the *entropy* of the distribution M of errors between crowd states predicted by a simulator $\hat{f}$ and real ones extracted from reference data. The authors compute the Entropy Metric using a two-phase process. Firstly, they estimate the crowd states X from the given validation data. Secondly, for each transition between inferred crowd states, from X_k to X_{k+1}, the distribution of prediction errors $m_k = X_{k+1} - f(\hat{X}_k)$ is computed using a maximum likelihood estimator. The collection of all errors m_k over all timesteps is denoted as M. To compute the entropy of the distribution they use its variance. The entropy of the distribution M measures the amount of information that is missing from the simulator $\hat{f}$ that would be needed to completely model the function f and capture true crowd motion. Figure 2.8 shows how the entropy metric can be used to compare simulated and reference data. In particular, the figure shows a comparison between a rendering of real-world crowd data (a), and stills from three different simulation algorithms applied to the same scenario.

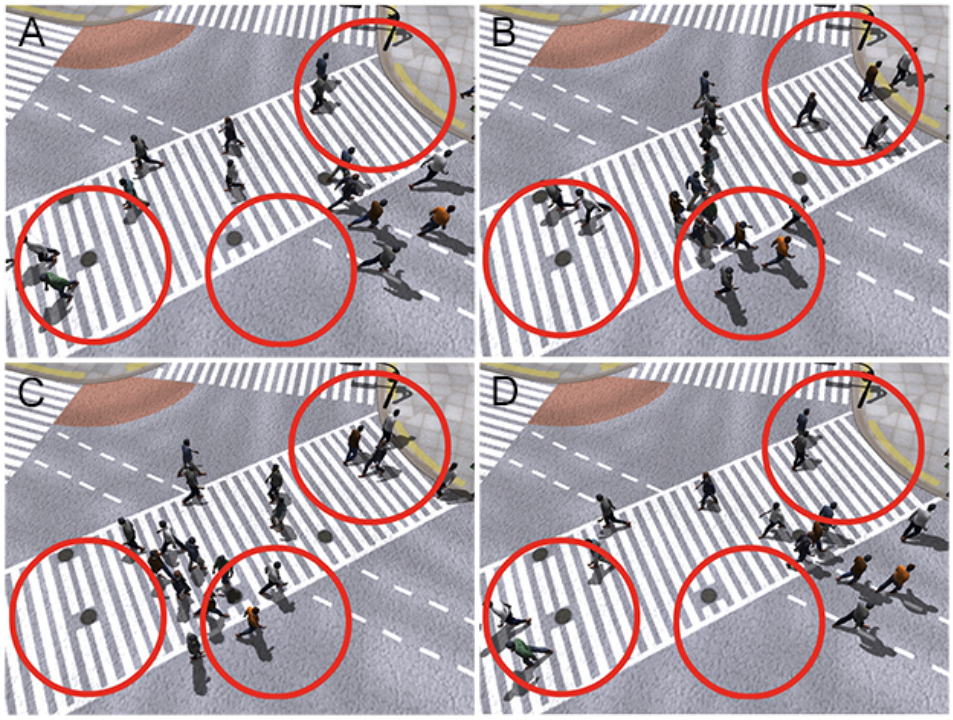

Figure 2.8: Entropy Metric approach for measuring the similarity between simulated and real-world data from [58]. (A) Real-world Data, (B) Entropy 4.7, (C) Entropy 3.8, (D) Entropy 2.7.

Edit Distance Metric. Another approach that has its background in Natural Language Processing (NLP), computes the similarity between two data points employing the Edit Distance on Real Sequence (EDR) metric. This metric is a distance function introduced by [24] and was originally used to compute the similarity between strings of characters. Given strings R and S, the Edit Distance on Real sequence (EDR) between them is the number of insert, delete, or replace operations that are needed to change R into S. In practical terms, navigation trajectories can be thought of as collections of points, instead of characters that make up strings. Every two given points of the trajectories match if their distance is below a certain threshold. With this definition, the cost of turning each point of R into its respective in S can be recursively computed – thus, ultimately, the similarity between the two trajectories can be estimated. According to Chen et al., EDR is a metric that tackles many problems of previous distance functions: it is resilient to noise and outliers, can handle local time shifting, and accounts for lengths of gaps in sub-trajectories.

Trending Paths. One approach that tries to strike a balance between low-level features such as individual trajectories, and global ones such as densities and exit

rates, is presented in the work of Wang et al. [146]. The authors use an unsupervised clustering method to learn a set of *trending paths* from reference data. These trending paths carry more general information regarding how pedestrians traverse the space, compared to single trajectories. It is then possible to objectively quantify the realism of a crowd simulator by comparing trending patterns of simulated trajectories with the ones extracted from reference data.

Fundamental Diagram. One macroscopic metric studies the relation between pedestrian velocities and crowd densities, and is reported in the *Fundamental Diagram* [127, 126, 156, 150]. This diagram asserts that individual pedestrian velocities are inversely proportional to crowd density – crowd velocity linearly decreases as density increases. This relation can be trivially compared with velocities generated by crowd simulators to assess whether it holds for virtual agents. Furthermore, this metric can be more formally validated through the analysis of covariance (ANCOVA) test [97]. This test establishes the degree of similarity between the slopes of the fundamental diagram and the simulated motion. In a similar macroscopic evaluation approach, the work of Cassol et al. [22] strives to assess whether the output of a crowd simulation meets some informed predictions such as (i) counter flow should impact negatively on the evacuation time, (ii) halving available exits should double the evacuation time, and (iii) agents should follow exit routes.

This overview highlights that many different evaluation techniques are available for comparing crowd simulation outputs to real-world data. On the one hand, this plethora of approaches is advantageous because it gives simulation designers a wide choice to pick from when evaluating their models. But on the other hand, if every researcher uses a different evaluation technique, it is hard to fairly compare the results. It might therefore be beneficial for the field to have a common evaluation platform where different simulation models can be compared using the same metric.

This is among the reasons that motivated the effort to create frameworks capable of unifying the various approaches in which crowd simulation is performed. The following section discusses some of these attempts.

2.3 Attempts to Unify

Given the fact that multiple research fields are contributing to the progression of crowd simulation, and that employing a "divide and conquer" strategy can render fruitful outcomes, it is unsurprising that the field is becoming increasingly fragmented. As concluded by Kleinmeier et al., this divergence is likely attributable to the various scientific backgrounds and objectives of the respective research groups. The authors explain that crowd simulation is a multifaceted area of study that has been explored by multiple scientific disciplines [82]. Each field may have its own unique set of procedures and practices for exploring human movement, and at times the same terminology can be used to denote disparate concepts. Nevertheless, the ultimate objective is to ensure the most advantageous outcome, hence the notion of

unification. This section discusses some of the attempts to unify the field of crowd simulation, which are summarized in Table 2.12.

Framework	Description	Sources
OpenSteer	OpenSteer is a library for creating steering behaviors for autonomous agents.	[115]
ADAPT	Extensible platform for animating crowds supporting locomotion, gaze tracking, reaching, and reaction to external forces.	[130]
JuPedSim	C++ simulation framework for simulating and analyzing pedestrian movement with a locomotion system, collision avoidance, route choice, and reporting tools.	[145]
Steer Suite	framework for evaluating steering algorithms with visualization and benchmarking tools.	[131]
Vadere	Framework for unifying crowd simulation, with 7 locomotion models, GUI, and evaluation Python scripts.	[82]
Menge	Crowd simulation framework decoupling the problem into goal selection, plan computation, and plan adaptation components.	[32]

Table 2.12: Crowd simulation frameworks.

2.3.1 Crowd Simulation Frameworks

OpenSteer. OpenSteer [115] is an open-source library of components used to create steering behaviors for autonomous agents in multi-agent simulations. Compatible with Linux, Windows, and Mac OS X, the toolkit consists of abstract mobile agents called vehicles, sample code, and simple steering behaviors which can be combined to produce more complex behavior. OpenSteer also provides an interactive application, OpenSteerDemo, based on a plug-in architecture and written in C++ with OpenGL graphics. This application can be used to visualize, annotate, and debug code, run simulations, and create novel steering behaviors, as well as control simulation time and view and track behavior via camera capabilities. OpenSteerDemo also has various scenarios such as “capture the flag”, “multiple pursuit”, “boids”, and “waypoint following”, and can be used to simulate human pedestrians. However, the framework lacks certain capabilities such as high-level behaviors, global path planning, and events, and has no external specification mechanism. As a result, simulation scenarios must be hard-coded into the plug-in.

ADAPT. ADAPT [130] is an extensible, scalable platform for animating crowds of autonomous virtual characters in sophisticated 3D virtual worlds. It is designed to enable the seamless integration of modular character controllers, such as data-driven

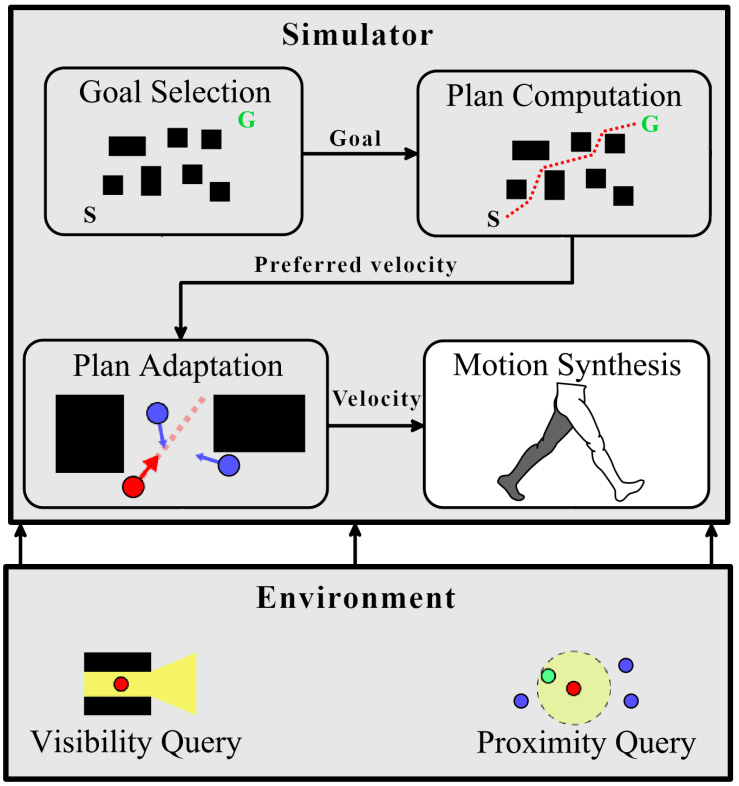

Figure 2.9: This figure, adapted from [32], shows the main components of the Menge crowd simulation framework. The goal selection component decides a destination for each agent. Then, plan computation generates a path to reach the goal. Finally, the plan adaptation component modifies the path in case of obstacles. During this process, the simulator can perceive the environment through visibility and proximity queries.

locomotion, procedural reaching, gesturing and physical reactions, with integrated navigation and event-centric behavior authoring for multi-actor interactions. The platform uses a system for blending arbitrary poses in a user-authorable data flow pipeline, allowing designers to choose between techniques and leverage established systems already produced by the character animation research community. Furthermore, ADAPT provides an interface for path-finding and steering, as well as a comprehensive behavior authoring structure for authoring both individual decision-making and complex interactions between groups of characters.

The system is based on a hierarchy of abstractions called the ADAPT Character Stack, which is split into four main tiers: Behavior, Actor, Body, and Animation. The Behavior layer contains commands comprising multiple sequential calls to the Actor layer and abstracts commands in the Body layer, keeping track of the duration

of a task. The Body layer converts abstract commands into messages sent to the navigation or animation system, and the Animation layer provides the lowest-level external access to the character's animation.

The platform also supports parameterized behavior trees, that allow data to be managed and transmitted within their hierarchical structure, for controlling multiple characters in an environment. These are used to author events that take one or more characters as parameters and temporarily grant exclusive control over those participants. Smart objects are also used to allow characters to interact with the environment and are useful for a wide array of tasks.

ADAPT is capable of performing tasks such as locomotion, gaze tracking, reaching and reaction to external forces and can support approximately 150 agents with full fidelity at interactive frame rates. The platform allows researchers to rapidly and visually iterate on character controller designs, and employ features from other packages to provide the functionality they lack without the need to deeply integrate or reinvent known techniques. At the same time, the framework does not allow convenient model comparisons: the user must manually change the code to switch between different models and compile two different binaries. Overall, ADAPT is best suited for the focused development of the final stages of crowd simulation such as motion synthesis.

JuPedSim. JuPedSim [145] is an open-source, C++ simulation framework that can be used to simulate and analyze the movement of pedestrians in a given geometry. The framework consists of three modules, namely, a simulation module, a visualization module and a reporting module. The simulation module includes two operational-level models (locomotion system and collision avoidance) and three tactical-level models (route choice, and short-term decisions). The reporting tool integrates four measurement methods, which allows for the analysis of densities, velocities, flows and profiles of variables in a given geometry. The visualization module reads simulation results and allows the user to interact with the results in an animation, or to record high-resolution videos. JuPedSim is also supported by a suite of verification and validation tests, offers a wide range of features, and is continually being developed with plans for a graphical user interface, import capabilities for various CAD formats, and connection of the pedestrian simulation with a fire simulator. The simulator is based on the generalized centrifugal-force model and 3D navigation mesh-like algorithm and supports dynamic environments and the concept of pedestrian knowledge and how it moves through the crowd. JuPedSim includes tools for analyzing the results and has an extensive XML-based specification language for flexible specification. However, due to its tightly coupled components, it lacks a plug-in architecture which limits the ability to perform focused development and efficient dissemination of new models.

Steer Suite. SteerSuite [131] is an open-source framework that aims to encourage community participation in the development and evaluation of steering behaviors. It includes a flexible test case format that is designed to be flexible yet easy to use, allowing users to create and validate their test cases; a library to automatically set up initial conditions, as well as more elaborate goal specifications in the test cases;

and a novel steering algorithm called PPR (Plan, Predict and React), which is a novel rule-based pedestrian steering algorithm that combines three aspects into a single steering decision.

The software allows users to visualize the simulation, draw annotations, and step through the simulation manually, making it easier to debug and analyze algorithms in the field of agent steering. SteerSuite also provides benchmark/metrics reports and recordings of the simulations, as well as a library that can read the test cases and automatically set up all initial conditions. Additionally, the benchmarking process is generalized, allowing users to experiment with their benchmark techniques and use metrics to debug or analyze a simulation. Overall, despite SteerSuite being a useful tool for evaluating and debugging agent steering algorithms and encouraging community participation in the field, it does not provide support for high-level behavior, and the XML specification is not extensible, meaning that referencing novel components requires modifications to the core application.

Vadere. *Vadere* [82] is an open-source simulation framework that enables the comparison of implementations of existing and new locomotion models. It offers a lightweight user interface and pre-implemented versions of the most widely used models. The framework offers modern algorithms and data structures, a graphical user interface (GUI), Python scripts to compare simulation outputs, and a command line interface to run multiple simulations. It was designed keeping interdisciplinary collaboration in mind and focusing on usability, thus, Vadere is open-source and allows other researchers to use and extend the code to fit their requirements. The simulation process involves three main steps: creating an input file, running the simulation, and analyzing the simulation results. The GUI has multiple features, such as providing an overview of the scenario files and the corresponding simulation outputs and offering a drawing program to define the topography of the environment. Vadere currently supports seven locomotion models, including the behavioral heuristics model, a simple bio-mechanics model, the gradient navigation model, the optimal steps model, an optimal velocity model, an implementation of Reynolds' steering behaviors, and an implementation of the social force model. The framework uses the Model View Controller (MVC) pattern to separate responsibilities for the three modules: (i) the Model represents the simulation state, (ii) the Controller contains the logic to change objects of the model layer, and (iii) the view visualizes the current simulation state. This, along with the definition of a common interface for defining novel locomotion models, makes it easy to extend the framework with new features. Finally, to ensure the quality of the framework, Vadere is tested using the Java unit testing framework JUnit, and its outputs are compared against 16 scenarios based on real-world experiments. However, Vadere uses a simplified representation of the environment that may not capture all the nuances of real-world environments. For example, it does not support terrain height variations or complex building geometries. Moreover, while the framework focuses on the comparison of models, it does not provide a way to combine different ones.

Menge. Another notable example is a crowd simulation framework called *Menge* [32]. The framework, as shown in Figure 2.9 aims at separating crowd simulation into decoupled sub-problems, whose solutions can then be more easily reused by other members of the community. First, *goal selection* involves determining what each agent wants to achieve based on several factors such as psychology and world knowledge, this is achieved through a Behavioral Finite State Machine (BFSM). Second, *plan computation* means devising a sequence of actions for reaching the chosen goal, by employing several techniques such as Navmesh, road maps, and velocity fields-based approaches. Finally, there is *plan adaptation* which adjusts the previously computed plan to account for dynamic phenomena, based on one of several available pedestrian models. These abstractions allow researchers to focus on a single aspect of the simulation model, delegating the complexity overhead of the remaining components to the framework itself. It then becomes possible to compare simulation models at a more granular level, since any of the components can be specifically matched against its alternative.

Although excelling in the decomposition of the crowd simulation process, and providing integrated bespoke solutions for each subcomponent, Menge does not provide any built-in feature to evaluate simulation outputs. This hinders the ability to perform meaningful comparisons between components. Moreover, the framework lacks support for combining high-level behaviors, which is essential for developing holistic models of human social behavior.

This section has presented an overview of the most relevant crowd simulation frameworks. The result has been summarized in Table 2.13 which shows the strengths and weaknesses of each framework. Although there has been a prolific effort that led to the development of several solutions, these frameworks have some important limitations. Most of them do not offer any built-in solutions to calibrate the model parameters, nor to evaluate simulation outputs. This makes it difficult to perform meaningful comparisons between components. Even more importantly, none of them support the combination of high-level behaviors, which is a required feature of a comprehensive model of human social behavior. For example, comparing or combining navigation mesh-based and steering behavior-based models of group formation is challenging due to different underlying paradigms, parameters, and variables. For a more detailed discussion of this example, refer to Section 2.5. Thus, it is important to develop a framework that addresses these limitations, and that can be used as a basis for future research in the field of crowd simulation.

Despite the proposed solutions, crowd simulation remains a complex endeavor. For this reason, the framework should be accessible enough to be used by researchers with different backgrounds. To this end, there have been several efforts to develop tools that can simplify the authoring of crowd simulations to foster collaboration and the development of new models. The next section presents some of these tools.

Framework	Modular	Extensible	Evaluation	Behavior Combination	Offers visualization	Convenient Authoring
OpenSteer [115]	Limited	Limited	Limited	Limited	Limited	Unsupported
ADAPT [130]	Supported	Limited	Unsupported	Limited	Limited	Supported
JuPedSim [145]	Limited	Limited	Supported	Unsupported	Limited	Limited
Steer Suite [131]	Limited	Limited	Supported	Unsupported	Limited	Limited
Vadere [82]	Supported	Supported	Supported	Unsupported	Limited	Supported
Menge [32]	Supported	Supported	Unsupported	Unsupported	Limited	Limited

Table 2.13: Strengths and weaknesses of the crowd simulation frameworks surveyed from the literature.

2.4 Attempts to Simplify

Authoring crowd simulations is a complex task, as it involves dealing with the multitude of components mentioned before. In particular, they can be classified into (i) high-level behaviors, which describe the agents' desires and their goals; (ii) path planning, which translates the high-level behaviors into a sequence of actions for reaching the desired goal; (iii) local movement, which accounts for the interactions with dynamic parts of the environment and other agents; and (iv) visualization, which introduces variety into crowds both at the static appearance level of the agents and their animations while moving. This classification is similar to the one proposed by Lemonari et al., who thoroughly reviews the authorable components of crowd simulations and explains the most relevant approaches for each of them, along with the tools available in the literature [84].

Given the complexity of the task, several attempts have been made to simplify the authoring process, which are summarized in Table 2.14. These solutions often revolve around graphical user interfaces (GUIs) that allow augmenting the environment with additional information influencing agents' behavior without having to directly deal with the underlying implementation details. This section reviews the most relevant approaches in this direction and argues that (heat) map painting is a particularly promising solution for authoring behaviors in crowd simulations.

Tool	Description	Sources
Crowdbrush	Paints various traits of agents in a crowd simulation, inspired by image manipulation tools.	[141]
Semantically Augmented Nav. Graphs	Tool for annotating navigation graphs with semantic labels to enable more intuitive and varied agent behavior.	[154]
Pedestrian Environment Designer	Using bitmap layers to define entrances, exits, collisions, attractions and avoidance.	[93]
Visual Sketching	Sketching for controlling agent spawning, navigation, barriers, and flow paths.	[54]
Spatial Situations	Situations can be defined through a painting interface, and agents respond with a probability distribution.	[133]
Map Formations	Set up formations by drawing them on a map, generating the necessary waypoints for the agents to follow.	[8, 57]

Table 2.14: Crowd simulation Authoring Tools.

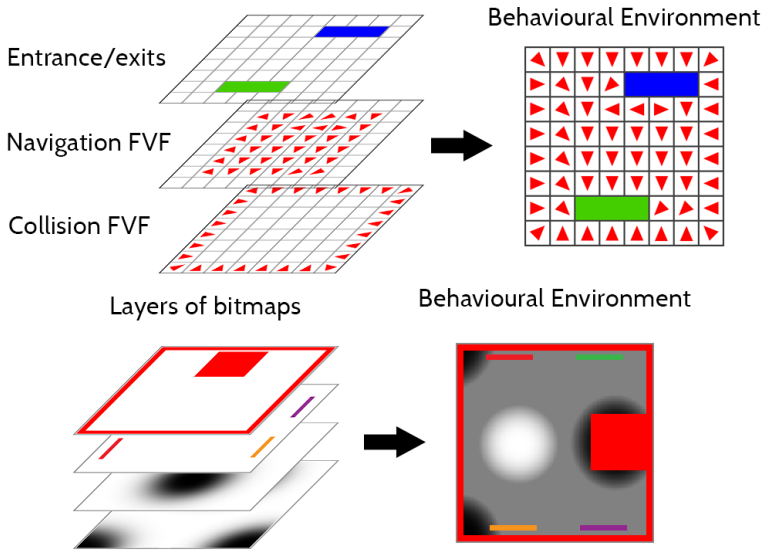

Figure 2.10: Example of how several maps can be painted to affect various aspects of the crowd simulation. In this case, the map defines the agents' spawn and despawn locations, and impacts the way they move and interact with the environment and other agents. Figure from [93].

Crowdbrush. Ulicny et al. [141] employ a brush metaphor to allow users to paint various traits of the agents in a crowd simulation scenario. The authors take inspiration from the image manipulation process, extending changes performed in 2D space to the 3D environment and characters. The system includes several brush functions including the ability to paint the agents' positions and orientations, their appearance and animations, and (to some extent) their behavior – by generating events that affect behavioral rules. This approach fosters simplicity and intuitiveness in the authoring process of simple crowd simulations and enables novice users to easily fill environments with multiple animated agents without having to deal with complex simulation models.

Semantically Augmented Navigation Graph. The work of Yersin et al. [154] focuses on a GUI tool that allows users to annotate a navigation graph with semantical labels (such as *park*, *hotel*, *station*, etc...) so that the goals can be defined more intuitively. Additionally, this results in more varied and realistic behavior, since the agents can choose their destination based on their current location and the available options fitting the desired goal's category, and are not forced to follow a predefined path.

Pedestrian Environment Designer. In the work of McIlveen et al. [93], the authors propose a novel approach for authoring high-level behaviors by offering a map sketching graphical interface, shown in Figure 2.10, allowing users to define

various layers with different functions: (i) the *Entrance/Exit* layer determines where the agents spawn and where they despawn; (ii) the *Collision* layer defines the traversable parts of the environment; (iii) the *Attraction* layer attracts pedestrians to move through the painted areas while moving towards their exit; and (iv) the *Avoidance* layer discourages movement through their specified areas. These layers are bitmaps that can be painted using tools similar to those found in common raster image editing software. After a collection of layers has been produced, a compilation process is used that converts the layers into an XML model input format for FLAME GPU, which then simulates the movement of the agents through force vector fields and the social force model.

Visual Sketching. Gonzalez and Maddock [54] propose a sketching interface that allows users to augment force-field-based navigation systems with additional information which can be used to influence the behavior of the agents. The system allows sketching three different elements: (i) *spawn and exit locations* controlling the instantiation of agents; (ii) *barriers* limiting the movement of the agents; and (iii) *flow paths* which steer the agents towards the desired direction. These various elements are then compiled into a force field map used for navigating the agents through the environment. Similarly, Colas et al. [28] explore a solution for authoring high-level behaviors by allowing users to sketch the desired flow of agents through the environment based on a target object, a guide curve, and a zero area. These three elements are then employed to generate an *interaction field*, which is a vector field that guides the agents' movement.

Spatial Situations. Sung et al. [133] presented a crowd simulation system wherein the behavior of pedestrians is impacted by *situations* associated with the environment. This system allows the user to define these situations by drawing regions on the environment through a painting interface. The responses of the agents to these situations are implemented through the sampling of a probability distribution.

Map Formations. Graphical techniques can be applied to more specific behaviors such as formations. Allen et al., Gu and Deng propose a graphical interface for authoring formations, where the user can draw the desired formation on a map and the system will generate the corresponding waypoints for the agents to follow [8, 57]. This allows the user to set up the desired formation without having to worry about the technical details.

This overview of the existing authoring approaches for crowd simulation has focused on the use of graphical interfaces for authoring agent behaviors. In particular, the use of map painting techniques is a promising solution for augmenting the environment with additional information that can be used to influence the behavior of the agents at various steps of the simulation pipeline.

At the same time, all the approaches presented in this section are limited in their scope, as they focus on low-level aspects of the simulation pipeline, such as the agents' spawn locations and goal destinations and their interactions with obstacles in the environment. While these factors play a significant role, they are more akin

to directing interfaces that enable the user to manipulate the agents' movements similar to a puppeteer, rather than authoring interfaces that allow modeling the desired behavior of the agents.

Moreover, as concluded by Lemonari et al. [84], there is no unified framework for authoring the desired behavior of crowds in simulations due to the complexity of the pipeline, a massive set of parameters to control, and the assumptions of how crowd simulation is achieved. The authors highlight the importance of a more unified framework that can integrate all components and facilitate all levels of the simulation at once – with particular focus being put on high-level behaviors as they influence the rest of the pipeline.

The use of heatmaps proposed in this dissertation, and introduced in Chapter 3, is a natural extension of this approach, as it allows the user to influence agents' behavior by generating heatmaps containing spatial information about the environment that can be used to steer the agents' movement. Additionally, the approach is integrated with state-of-the-art crowd simulation systems: Menge for agent behavior modeling and simulation, and Unity for visualization. This allows leveraging heatmaps at various stages of the simulation pipeline including influencing high-level behaviors with dynamic goal selection; steering the agents' movement through the environment; and comparing the simulation output in the form of heatmaps with the real-world counterpart.

To foster the adoption of this approach, the authoring process must be intuitive and easy to use. For this reason, it is fundamental to create a graphical interface that allows users to easily generate heatmaps, modify and combine them, and use them to influence the behavior of the agents.

2.5 Discussion

Researchers have successfully employed a range of social behavior theories, studied in psychology and sociology, for generating and animating virtual human behavior in crowd simulations. Moreover, several navigation algorithms have been proposed that allow virtual humans to traverse space in increasingly realistic ways. Recent advancements in simulation techniques have produced ways of objectively calibrating parameters, and validating simulation models – thus pushing these simulations closer and closer to reality. Finally, to manage the complexity of crowd simulations, researchers have developed a variety of authoring tools that allow users to easily create and modify simulations. The foundation of this field, therefore, rests on several sub-fields, each contributing an approach and a technical solution.

On the one hand, this "divide and conquer" strategy has been useful in letting researchers focus on isolated and smaller problems, but on the other hand, it has led to fragmentation in the field. Crowd simulation models are developed relying on a specific selection of components from the aforementioned sub-fields. This makes it difficult to compare and even more challenging to combine different models, as they are based on different assumptions, use different techniques, rely on different components, and produce different outputs.

For example, a model of group formation that has been developed by relying on the global path planning from a navigation mesh cannot easily be combined with one

built by extending steering behaviors. The first is built on a navigation mesh which is a system of interconnected polygons that allows a character to move from one polygon to the next. This type of system bases its navigation on a predetermined path and does not allow for any deviation or improvisation. The second, on the other hand, is built on steering behaviors which rely on a character's ability to sense their environment and make decisions based on their perceived obstacles and goals. This type of system allows for more improvisation and flexibility in the character's movements. Combining the two simulations would require combining the two underlying paradigms, which can be difficult because they rely on different sets of parameters and variables. Furthermore, it is also challenging to predict how the characters would interact with each other or the environment when using a combination of the two principles. As a result, it is not easy to combine the two models. Moreover, since each model relies on different sets of parameters and variables, it is difficult to fairly compare the two models' outputs. The differences in results can be attributed to the chosen parameter sets, rather than to the models themselves.

Thus, fragmentation has led to a need for a framework that supports designing and implementing crowd simulation models in a standard way so that the resulting models are comparable and can be combined. This framework should be able to integrate different models and approaches in a unified system, allowing for the efficient development, evaluation, and integration of different crowd simulation models. Furthermore, such a framework should be able to provide a platform for the comparison of different models and techniques, as well as for the integration of existing models into a unified system. This would allow for the development of more complex and realistic crowd simulations, as well as for the easy comparison and combination of different models.

To this end, there have been several attempts to create frameworks that allow for the decomposition of the crowd simulation field into smaller sub-problems, such as Menge [32], which provides a generic architecture for the simulation of pedestrian dynamics. However, these frameworks cannot still combine several high-level social behavior models, which is essential for designing agents with more holistic social intelligence. Moreover, introducing graphical user interfaces and map-based encoding of environmental and behavioral information can enhance user experience and facilitate the design and testing of various scenarios. Ultimately, further research is needed to create a unified system that supports the development of more complex and realistic models, while allowing for easy comparison and integration of different approaches.

Chapter 3

Theoretical Framework

3.1 What Influences Human Navigation

When humans walk around an environment, they are guided by several stimuli. For example, suppose a tourist is exploring a place they have never seen before. In that case, they might prefer the most visible locations or points of interest, and - at the same time - they might want to avoid overcrowded attractions. On the other hand, a resident of a city who is familiar with that particular environment might associate certain places with specific memories or sensations, which can either attract or repulse them depending on whether they are positive or negative.

Several stimuli that influence human navigation are related to the social sphere. One particularly important example is the influence of group dynamics on the way people move around an environment since, most of the time, people walk in groups [97]. Group behavior can influence navigation while walking by causing people to be more likely to follow and stay with the group instead of veering off and taking a different route. For example, if a group of people is walking to their destination together, they may all travel in the same direction and avoid any detours or shortcuts that could be taken. Additionally, they may be more likely to make the same decisions when it comes to crossing the street or waiting for a light to change. An advantage of modeling group behavior in crowd simulation is that it allows for a better understanding of how people interact and behave in larger groups. This can be useful for analyzing crowd dynamics in public spaces, such as public transportation, stadiums, and malls, as well as for studying the impact of policy changes on crowd behavior. Additionally, modeling group behavior can provide insights into how individuals make decisions and interact with each other in different environments.

Thus, each time a human being takes a step, they solve a challenging decision task that involves the complex evaluation of both their mental state and external inputs, as shown in Figure 3.1.

Chapter 2 highlighted how several models have been proposed in the literature to address many of these aspects of human behavior. Particularly, Section 2.5 drew attention to the challenges of combining different human behaviors into a unified

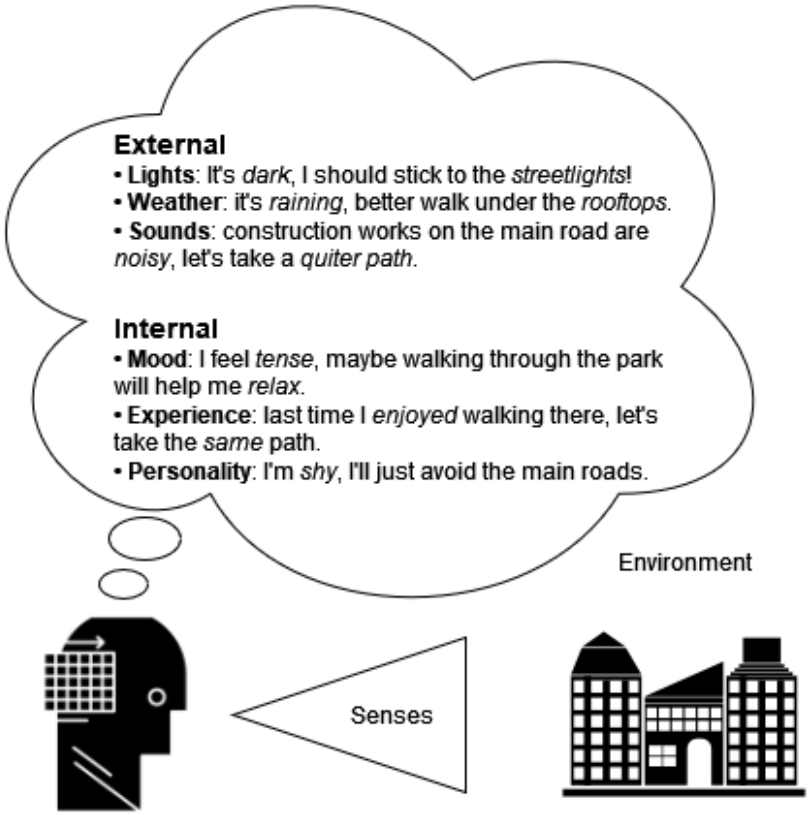

Figure 3.1: Example of internal and external stimuli which can influence the way that people walk around the environment.

model. This chapter will argue that a particular subset of human behavior theories can be modeled with graphical data structures called heatmaps and that those can be blended to create a simulation of human reasoning that considers stimuli from several sources simultaneously.

3.2 Modeling Behaviors with Heatmaps

A heatmap is a data visualization technique showing a phenomenon's magnitude as color in two dimensions. Heatmaps often serve to shade matrices so that larger values yield colors that differ from those associated with smaller weights, as shown in Figure 3.2. Usually, the colors differ in their hue value or intensity (brightness), but they can also depend on a gradient between different colors.

By subdividing the environment in a grid-like fashion, it is possible to create a

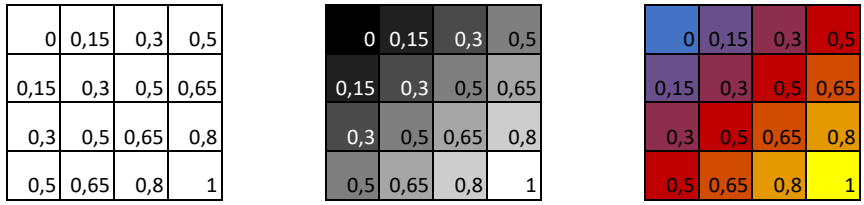

Figure 3.2: (A) Example of a matrix with ascending values from top left to bottom right, (B) the same matrix shaded as a gray-scale heatmap, (C) the matrix shaded according to a gradient going from blue to yellow.

matrix holding arbitrary data in its cells. For example, it is possible to create a matrix whose values represent the height of the buildings in the environment, as shown in Figure 3.4. Then, it is possible to encode the matrix as a heatmap by assigning a color to each value contained in its cells, thus making the visualization more effective than simply reporting the raw values.

Heatmaps are highly effective tools for designers and other decision-makers, as they provide a visual representation of data that can emphasize patterns, trends, and correlations. By presenting data in this accessible format, they can assist in identifying areas that warrant further investigation and offer insights into the rationale behind specific choices. Furthermore, by enabling decision-makers to rapidly and effortlessly discern patterns, trends, and correlations, heatmaps facilitate more evidence-based decisions rather than those solely reliant on intuition [39].

The opposite process is also possible: color intensity or gradients are used to author data mappings, which are then expressed as textures used for various purposes. One notable example is in the field of skeletal animation and rigging [95]. In that context, weight painting is commonly used to associate a character's mesh geometry with the underlying bone structure. In that case, 3D artists use digital brushes to paint each face of the mesh with a particular color, corresponding to a certain weight representing the magnitude of the bone's influence on those faces. The result, illustrated in Figure 3.3, is a heatmap that wraps around the character's mesh and provides an easy-to-interpret visualization. Moreover, this is a convenient way of assigning weights: different bones can share their influence on the same geometry by just combining the heatmaps.

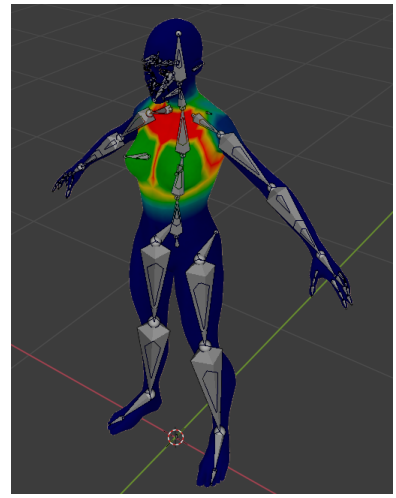

Figure 3.3: Representation of the weight paint process in Blender involves painting each face of the mesh with a color representing a bone's influence.

Often, when the heatmap is superimposed on a geographical location, it is called a *spa-*

tial heatmap. Then, the information held in the heatmap can be used in any model of human behavior. For example, while simulating a tourist visiting a city, it might be helpful to know which building is the tallest - because they might want to go there to have a good view.

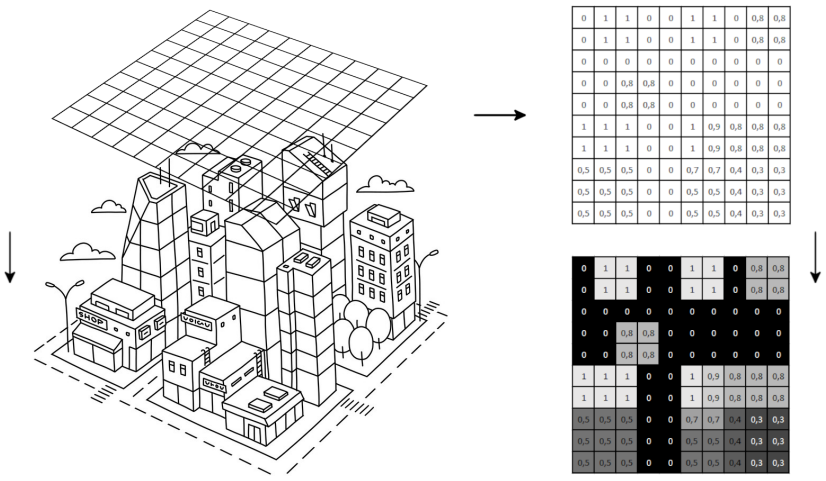

Figure 3.4: Example of how a spatial heatmap can be used to represent features of the environment, such as building height. Here, cells that correspond to tall buildings have higher values, while the ones covering the ground have lower ones. Then they were shaded according to a grayscale gradient going from black to white.

At the same time, spatial information does not only pertain to physical data such as height. Several theories in the literature propose that humans associate all sorts of characterization with the space surrounding them - including social ones. For example, Scheflen [123] proposed the existence of unconscious rules about space governing much of human behavior. And Kendon defined how people arrange themselves in formations when engaging in a conversation. These models, among others, can also be expressed by spatial heatmaps.

3.3 Advantages of Modeling with Heatmaps

There are several advantages of modeling human behavior through heatmaps. First, they help cluster the data needed by each human behavior model by visually representing the geographic distribution of certain activities or behaviors, and offering a common way of encoding arbitrary information used for simulating a particular aspect of human reasoning. This clustering allows for fully defining each behavior within its self-contained layer, holding the data needed for modeling the behavior characterized by a heatmap. Moreover, having a standard format for encoding

behavioral information allows for combing them together in a straightforward and computationally efficient way.

For example, there could be several approaches for modeling of the noise stimulus represented by the heatmap shown in Figure 3.5 (D). In fact, there are several models that could be employed for simulating how sound propagates throughout the environment [88]. In this case, the heatmap approach makes it possible to employ multiple sound propagation models, and cluster the resulting information into a single data structure. Since the clustered sound data is represented by single heatmap, the resulting behavior does not rely on any additional information. Thus, it is possible to assert that the behavior belongs to a self-contained layer, which could be enabled or disabled without impacting other parts of the simulation.

3.3.1 Spreading Data Across Several Heatmaps

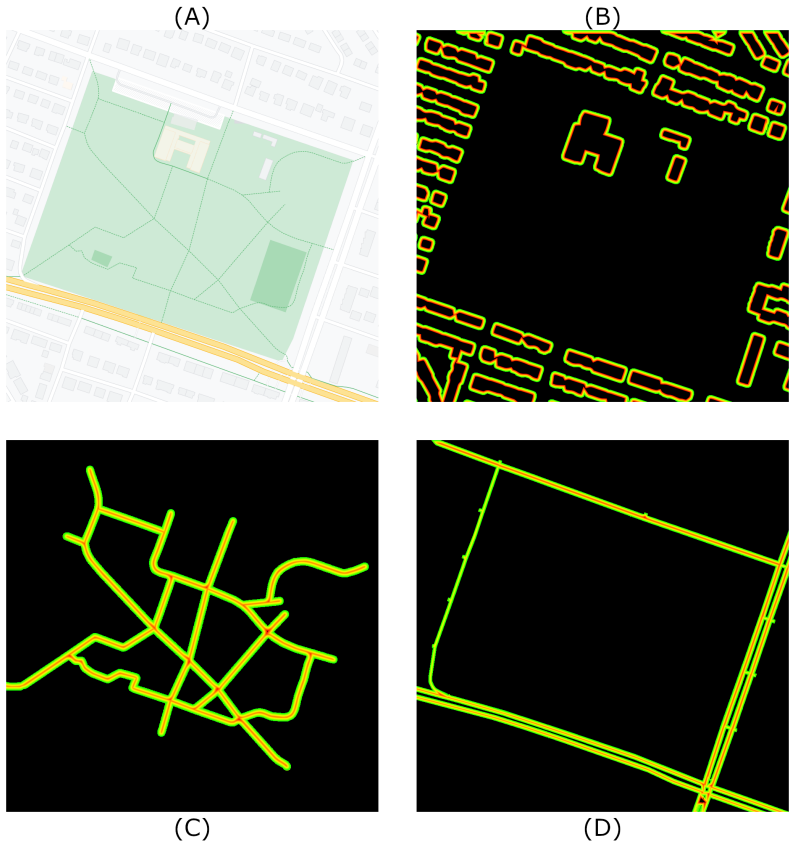

Figure 3.5: (A) Top down view of a city district; (B) Rain shelter heatmap; (C) Park trails heatmap; (D) Main roads heatmap

Spatial heatmaps make it convenient to encode arbitrary information into

separate layers. For instance, Figure 3.1 considers several stimuli that might govern human behavior. Let's focus on only three of them: preferring locations sheltered from rain, prioritizing the trails in the park, and stressing the importance of avoiding the main roads. By applying behavioral rules to a person in a real-world location, it is possible to see how the heatmap encoding would look like. Figure 3.5 (A) shows a top-down view of a city district containing several buildings, streets, and a walkable park in the middle. A possible heatmap highlighting the locations sheltered from the rain is visible in Figure 3.5 (B), where areas surrounding buildings have balconies stopping the rain. Similarly, Figure 3.5 (C), models the stimulus of walking in or near nature, by highlighting park trails, and finally (D) encodes data about the location of the busiest roads.

It is possible to see here how each heatmap encodes only one specific stimulus that influences human behavior, and this comes with several advantages:

- Each behavior is independent of the others since it only relies on data contained in one particular layer. Although this is a simplification of human behavior, it is convenient for modeling purposes.
- Layers can be enabled or disabled: much like layers in image editing programs, it is very convenient to toggle a particular stimulus by modifying the activation status of the layer.
- Influence can be weighted: the behavior associated with each heatmap can have a stronger or weaker impact on the simulation, based on the weight of the associated layer.

3.3.2 Absolute and Relative Heatmaps

Heatmaps can be used to encode both absolute and relative data, depending on the nature of the information they represent. Absolute heatmaps encode data about the entire environment, reflecting fixed attributes or characteristics. For instance, a heatmap displaying the height of buildings in a city would be absolute, as it conveys information about the entire urban landscape that is not dependent on any specific location within it.

Conversely, relative heatmaps encode data concerning a particular position or entity, capturing information that changes based on the context or location. An example of a relative heatmap would be one illustrating the perception of personal space for individuals within a crowd. In this case, the heatmap is relative to each person's position, as the perception of personal space varies depending on the surrounding environment, proximity to others, and other factors that dynamically change as the individual moves.

By utilizing both absolute and relative heatmaps, designers and decision-makers can gain a comprehensive understanding of various aspects of an environment, as well as the dynamic relationships and interactions among its elements. This enables the creation of more accurate and effective models that account for both fixed attributes and situational factors.

For instance, Figure 3.6 presents two distinct heatmaps side by side. The heatmap on the left is represented with an absolute coordinate system, illustrating

the height of buildings in a city. As it is based on the world's coordinate system, it remains fixed and independent of any particular individual's position within the environment.

In contrast, the heatmap on the right portrays a relative concept—personal space—using a coordinate system that is dependent on each person's position. As individuals move throughout the environment, the heatmap dynamically adjusts to reflect the varying perception of personal space relative to their changing locations.

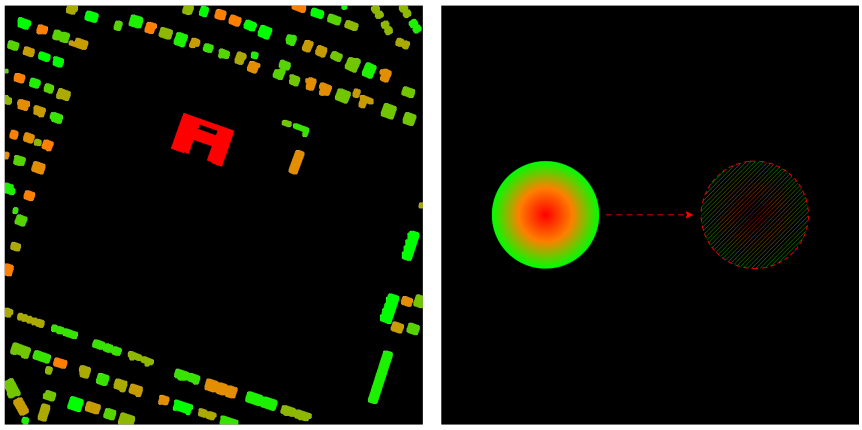

Figure 3.6: Left: An example of an Absolute Heatmap, encoding the height of buildings in a city. Left: An example of a relative heatmap encoding personal space.

3.3.3 Heatmaps with Varying Resolutions

The size of environments represented by heatmaps can differ significantly, and as a result, the computational resources required for larger heatmaps can be substantial. Consequently, heatmaps must support varying resolutions to accommodate different scenarios effectively.

The resolution of a heatmap might depend on several factors, such as the map's size, the data's importance, and the desired level of detail. For instance, when encoding a heatmap for an entire city, it might be necessary to use a lower resolution due to the vastness of the environment. This approach conserves computational resources while still providing an overview of the city's relevant data.

In contrast, a heatmap representing personal space can typically utilize a higher resolution since it covers a smaller area. The increased resolution allows for a more detailed and accurate representation of the personal space, which may be crucial for understanding individual interactions within the environment.

To accommodate these varying requirements, heatmaps should include a scale parameter. This parameter represents the relationship between world units (such as meters) and heatmap units, enabling adjustments to the resolution as needed. By incorporating different resolutions, heatmaps can effectively represent diverse environments and situations, allowing designers and researchers to make informed decisions based on the available data.

Figure 3.7 presents an exaggerated example of two heatmaps with distinct resolutions. The high-resolution heatmap, which encodes personal space, is overlaid on top of the lower-resolution heatmap representing building height. The personal space heatmap requires greater detail due to its smaller size and the importance of capturing subtle variations. In contrast, the building height heatmap covers a larger area and features less variation, so a lower resolution is suitable for this representation.

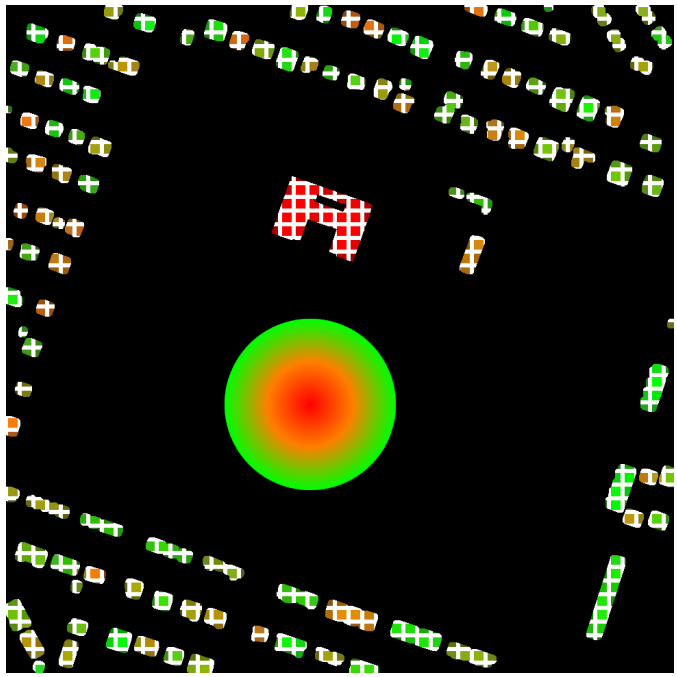

Figure 3.7: An example of heatmaps with varying resolutions. The building height heatmap has a lower resolution than the personal space heatmap.

3.3.4 Blending Heatmaps for Combining Models

Since heatmaps are two-dimensional arrays where each cell contains color data, they can be treated as images, and it is possible to combine them by applying color operations. Several color operations can be applied to heatmaps, among which are addition, subtraction, multiplication, and division. These operations are well-defined in the field of image processing, where they are used to combine images.

By separating the data of each behavior into a single heatmap, or layer, it becomes convenient to combine them in several meaningful ways. For example, the layers of Figure 3.5 represent two attractive stimuli: the rain shelter and the park trails, and one repulsive stimulus: the noisy main roads. Even though each heatmap highlights a distinct aspect of the environment, the first two can be combined through a simple addition, while the last one can then be subtracted from the result.

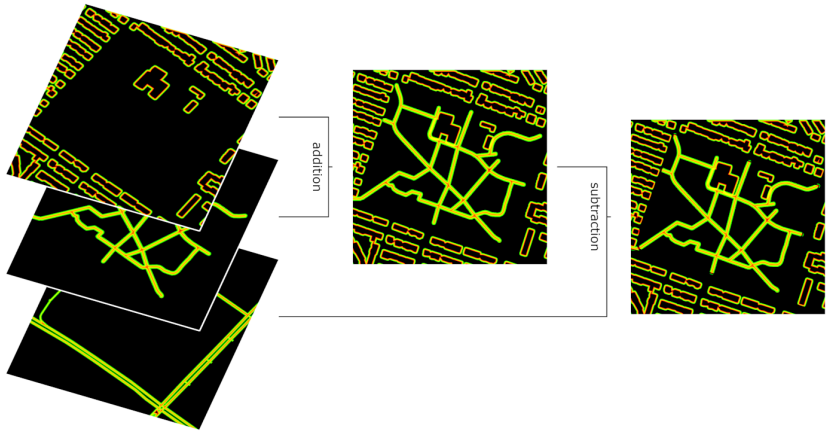

Figure 3.8: Each heatmap encodes a different stimulus that might influence human behavior. The colors of the first top two are summed together, and the third one is subtracted from the result.

That way, the resulting heatmap will highlight the areas that are sheltered from the rain and have trails, while avoiding the main roads, as shown in Figure 3.8.

The layers can be weighted to control the influence of each behavior on the final result. Weights are essential when several stimuli influence a person's behavior, and each has different importance, possibly depending on the context. For example, in an emergency such as a severe storm, the person might be more likely to seek shelter from the rain, and the trails in the park might be less critical. In this case, the weight of each heatmap layer can be dynamically adjusted to reflect the importance of each stimulus. This feature both empowers the authoring process by allowing for more control over the simulation, and it also allows for more realistic simulations since the weights can be based on real-world data.

Finally, modeling an agent whose behavior is based on the resulting heatmap makes it possible to simulate a person who accounts for several stimuli simultaneously. Referring to the example of Figure 3.8, the simulated person would prefer to walk in dry spots of the park while avoiding the main roads.

3.3.5 Combining Heatmaps with Different Sizes

In order to address the issue of combining heatmaps with different sizes, where some cells of the larger heatmap may not overlap with those of the smaller heatmap, it is crucial to define an out-of-bounds strategy. This strategy determines how to handle color operations between cells containing data and those that do not. An example of this strategy is illustrated in Figure 3.9.

A potential solution is to treat empty cells as color neutral for the given operation (e.g., black for addition and subtraction, white for multiplication and division). Another approach involves assigning colors to out-of-bounds cells based on colors

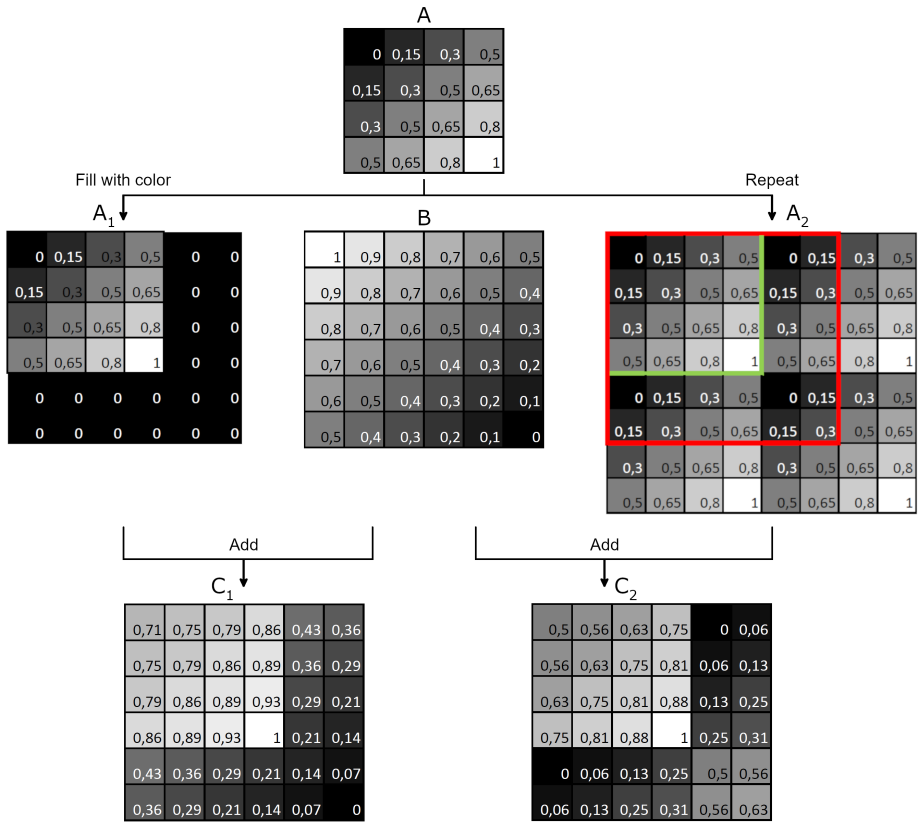

Figure 3.9: In this example two heatmaps A and B are combined using the addition operation. Since they have different sizes, respectively 4x4 and 6x6, it is necessary to define an out-of-bound strategy for the cells that are not covered by both heatmaps. The first strategy is to consider the out-of-bound cells as neutral to the operation (black for addition), which results in A_1. The second strategy is to consider the out-of-bound cells belonging to the heatmap, effectively wrapping the heatmap cells around its borders - shown in A_2 (red). Then, it is possible to carry out the addition operation between the two heatmaps, resulting in C_1 and C_2 depending on the out-of-bound strategy adopted.

inside the heatmap. In this case, methods from computer graphics, such as texture mapping with UV coordinates, can be applied. UV in this context refers to the coordinates in a 2D space (usually a texture map in computer graphics) that are used to map textures or colors onto a 3D object; the "U" coordinate can be understood as the equivalent of "X" in a 3D space, and "V" as equivalent to "Y". When the bounds of the UV space are reached, the texture can either be repeated or clamped. Similarly, with heatmaps, the color of the nearest border cell can be repeated, or the heatmap can wrap around so that cells on the opposite side of the border are

considered adjacent.

3.3.6 Dynamic Heatmaps

Some stimuli that influence human behavior are dynamic and change over time. For example, when modeling an agent which reacts to overcrowding, it is necessary to consider that the number of people in particular locations changes during the day. Additionally, even if the stimulus does not change, the agent's perception of it might change over time. For instance, the overcrowding perception might be more prominent during the night, when it is not expected to see many people in the streets, even when that number is the same or lower than during the day.

For this reason, it is necessary to support dynamic heatmaps, which can change over time. Dynamic heatmaps can be updated through localized operations, which only affect a particular region, or by editing the whole heatmap at once. For example, a crowdedness heatmap can be locally updated by increasing the value of the cells where people are and decreasing the value of the cells where people are spreading out, all in real time.

3.3.7 Heatmaps Performance

Since heatmaps can be encoded into images, creating and modifying them through image processing libraries is possible. These libraries often take advantage of the GPU to accelerate operations, which makes them very efficient. In particular, when considering a heatmap as a texture where each cell is a pixel, instead of processing the heatmap sequentially cell by cell, the GPU can process multiple cells in parallel, which makes the operations much faster. This advantage is especially noticeable when encoding data that can be computed visually, such as the height of buildings, which can be obtained by rendering the environment from a top-down perspective and using the depth buffer to encode the building size in each cell. Then, it becomes possible to create and modify heatmaps in real time, which is very useful for encoding phenomena that continuously change.

3.4 Leveraging Heatmaps for Human Behavior Modeling

Once heatmaps are defined, they can be used to model human behavior. There are several levels of reasoning at which heatmaps can be used, from low-level to high-level. Lower levels concern obstacle avoidance and pathfinding, while higher levels concern goal selection and decision-making. This section describes how heatmaps can be used to model human behavior at each level.

3.4.1 High Level Behavior Modeling

At this stage of human behavior simulation, heatmaps can be used to model high-level goals and decision-making. In particular, they help decide which goals to pursue and how to pursue them. This can be achieved by finding the highest value

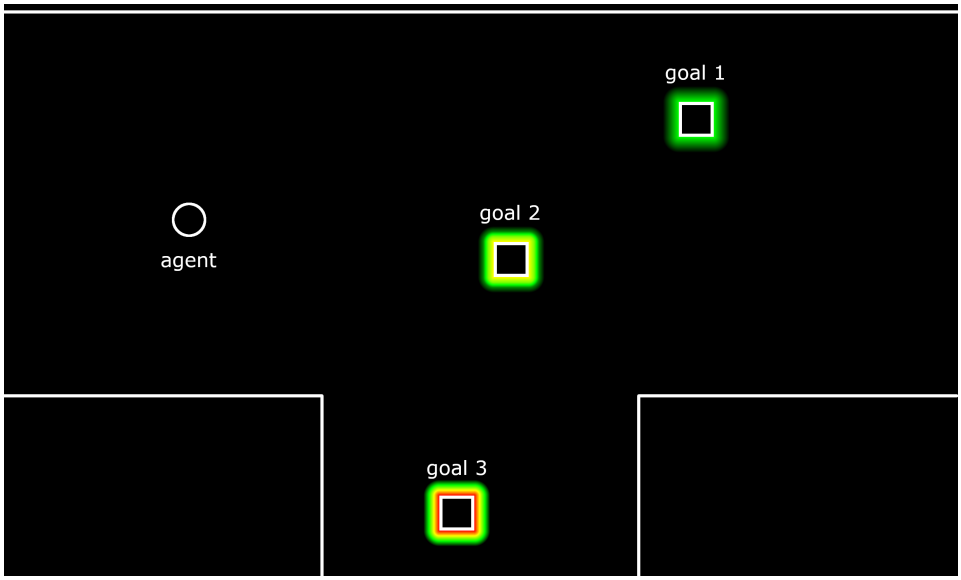

Figure 3.10: Example of a heatmap used to select a goal. Each goal influences the heatmap differently, contributing a specific value to the cells that are close to it, based on its importance. The agent then selects the goal which has the highest value in the heatmap, which in this case is goal 3.

in the heatmap and considering the underlying location as the goal to reach, as shown in Figure 3.10. For example, a person might be looking for a place to eat, and the heatmap can encode the quality of the restaurants in the area. Then, the person can move to the location with the highest value, which is the restaurant with the highest quality. Again, since heatmaps can be combined, it is possible to combine several heatmaps to encode different aspects of the environment. Thus, the goal selection can be based on a combination of several weighted goals. This is very useful for modeling complex behaviors, such as the decision-making process of a person who is driven by multiple goals.

3.4.2 Global Path Planning Adjustments

Heatmaps can also be used for global path planning adjustments. They are particularly useful for modeling the effects of the environment on the agent's perception of the path. Thus, trajectories computed by path planners can be adjusted by leveraging the heatmap. Since there are several global path-planning approaches, and not all of them produce complete trajectories, it is necessary to define how to apply the adjustments. For the algorithms that compute a complete trajectory, the adjustments can be applied by modifying the trajectory itself. For instance, the trajectory can be modified by adding a new waypoint in any of the locations with the highest value sampled at regular intervals, as shown in Figure 3.11. One example of this is the navigation mesh approach, which computes a complete trajectory from

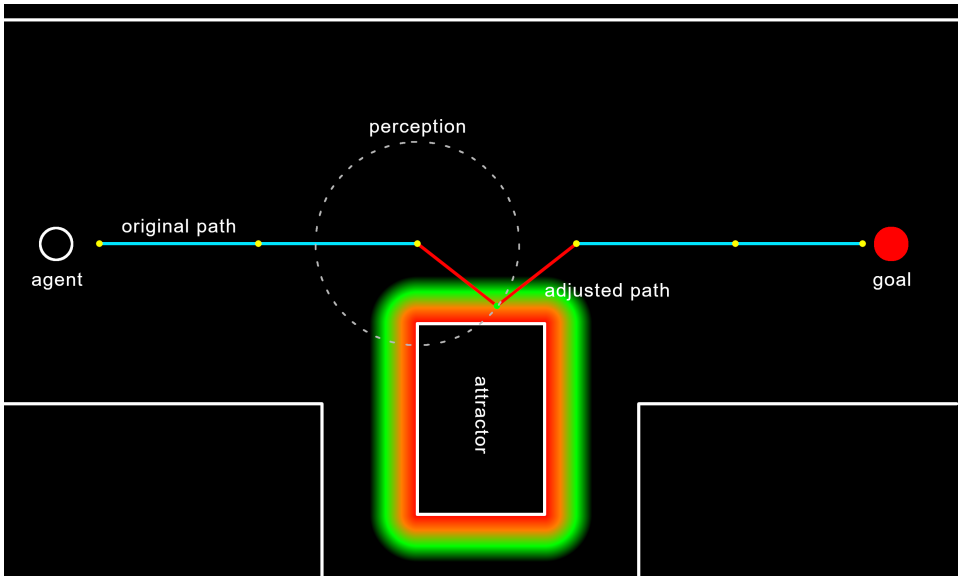

Figure 3.11: Example of a heatmap used to adjust the path of an agent. The agent originally tried to reach the goal following the shortest path, composed of waypoints, returned by the pathfinding algorithm. At each waypoint, the agent checks the heatmap and adjusts its path going towards the direction of the highest value. In this example, the agent perceives an attractor on the third waypoint of the path. The agent then adjusts its trajectory toward the attractor before reaching the goal.

the start to the goal. In this case, it is simply possible to traverse the trajectory, sample the heatmap at regular intervals, and add new waypoints in the locations with the highest values. The result is a trajectory that follows the heatmap as well as the original path, thus guaranteeing that the agent will reach the goal while also taking into account the stimuli encoded in the heatmap.

On the other hand, there are path planning algorithms that do not compute a complete trajectory, but rather follow a path incrementally step by step. In these cases, the agent does not have a complete trajectory of waypoints to follow but is rather instructed to move according to a particular velocity vector, which is dynamically recomputed at each step. This means that the algorithm does not offer, out of the box, a way for knowing where the agent will be in the future. For this reason, simply modifying the velocity vector to follow the heatmap is not enough, because the agent might end up moving in the wrong direction. Thus, it is necessary to define a way to transform the sequence of velocity vectors into a trajectory and then apply the necessary adjustments for following the heatmap. This can be achieved by using a simple integration scheme, such as the Euler method, which computes the trajectory by integrating the velocity vector at each step. Then, the trajectory can be modified by adding new waypoints in the locations with the highest value sampled at regular intervals. This approach can be used with any path planning algorithm that does not compute a complete trajectory, such as the

potential field approach, which computes the velocity vector at each step.

3.4.3 Local Steering

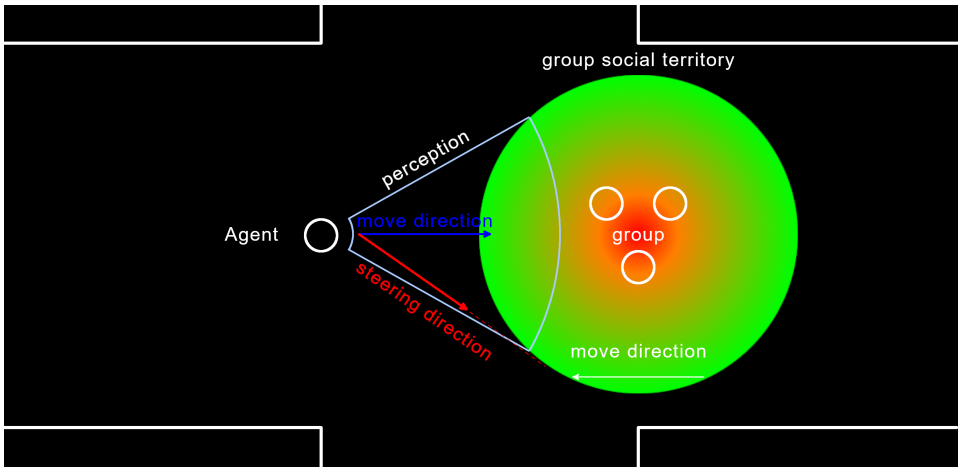

Figure 3.12: Example of local steering using a group social territory heatmap relative to the group's position. An agent is walking when it suddenly notices a group of people moving in the opposite direction. The group claims a region of social territory, encoded in the heatmap and perceived by the agent - who then changes direction to avoid intruding on the group's space.

Another application of heatmaps is local steering, which deals with dynamic stimuli that change over time, and thus cannot be encoded in a static heatmap or accounted for in the global path-planning process. At this level of reactive navigation, the agent aims to reach the goal while avoiding other agents or moving obstacles. Generally, the agent perceives its surroundings, computes the vectors that would lead to a collision, and then chooses a velocity outside this set [142]. In this context, heatmaps can augment the agent's perception of the obstacles in the environment, thus enabling it to react to multiple dynamic obstructions at the same time. The agent can use a dynamic heatmap to steer towards the lowest values, indicating places where there are no or fewer obstacles. This approach can be used to compound the effects of multiple dynamic obstacles by combining the corresponding heatmaps.

For example, when walking in a crowded street, a person might suddenly notice that a group of people is moving in the opposite direction. In that case, they might change direction to avoid intruding on the group's social territory. This stimulus can be modeled by a relative heatmap that encodes the group's social territory, which represents the space claimed by the group of agents. Then, any agent can perceive the group's social territory in their local surroundings through the heatmap and adjust their path accordingly, as shown in Figure 3.12.

3.5 Operators Formalization

Formalizing the syntax of an operator is a crucial aspect of developing a robust and user-friendly framework. By formally defining the syntax of an operator, it is possible to specify exactly how the operator should be used, including any parameters needed to compute the operation. This helps ensure that different framework users can employ the operators correctly and consistently. Formally defining the syntax of an operator also helps to improve the readability of the operations, making it easier for users to understand the meaning of the operations. Additionally, the formalization of operator syntax is an important aspect that enables writing concise and meaningful expressions to indicate complex operations. This section formalizes the operators used to create and modify heatmaps. The operators are defined in terms of the heatmaps' underlying data structure, a 2D grid of cells. Table 3.1 lists the operators used to create and modify heatmaps.

Operation	**Notation**	**Description**	**Complexity**
Add	$A + B$	Sum the value of every cell of A with the ones in B	$O(n)$
Subtract	$A - B$	Subtracts the value of every cell of A with the ones in B	$O(n)$
Multiply	$A \cdot B$	Multiplies the value of every cell of A with the ones in B	$O(n)$
Divide	A/B	Divides the value of every cell of A with the ones in B	$O(n)$
Average	$\mu(A, B)$	Average the value of every cell of A with the ones in B	$O(n)$
Threshold	$\tau(A, t, v1, v2)$	Sets the value of each cell in the heatmap to v1 if it is below a threshold t, v2 otherwise.	$O(n)$
Normalize	$\hat{A}$	Normalizes the heatmap by setting the maximum value to 1.	$O(n)$
Invert	$!A$	Inverts the heatmap by subtracting the value of each cell from 1.	$O(n)$

Table 3.1: Operators for Heatmaps.

The addition operation $A + B$ sums the value of every cell of A with the ones in B. Summing two heatmaps is useful for combining concordant stimuli, either both attractive or repulsive. For example, referring to Figure 3.5, heatmaps A and B are both attractive, representing rain shelters and park trails. In this case, they are summed together, resulting in the heatmap shown in the center of Figure 3.8.

The multiplication operation computes the product of the values of the cells in the two heatmaps. Differently from addition, since the mathematical product has the property that $0 \cdot x = 0$, the result of the operation is a heatmap where the values of the cells are zero if the value of the corresponding cell in either of

the heatmaps is zero. Thus, multiplying two heatmaps is particularly useful for modeling the influence of stimuli that only affect certain parts of the environment. For example, a heatmap that encodes the effect of a light source on its surrounding can be multiplied by a heatmap that encodes the visibility of the environment so that the influence of the light source is only applied to the visible parts of the environment. In this example, the light source heatmap would not need to account for occlusions, which would be encoded in the visibility heatmap, and affect the result.

The subtraction operation $A - B$ subtracts the value of every cell of A from the ones in B. Subtracting two heatmaps is useful for combining discordant stimuli, either one attractive and the other repulsive. For example, referring to Figure 3.5, heatmap C encodes a repulsive stimulus representing the noisy main roads. Then, C is subtracted from the others, resulting in the heatmap shown on the right of Figure 3.8. Similarly, the division operation A/B divides the value of every cell of A by the ones in B and can be used to combine discordant stimuli in a stronger way than subtraction.

The average operation $\mu(A, B)$ computes the mean of the values of the cells in the two heatmaps. This operation is useful when combining stimuli whose agreement is irrelevant, or cannot be determined a priori. For example, referring to Figure 3.8, if it was not known how the rain shelters and the park trails stimuli interact with each other, they could be averaged together resulting in a heatmap that encodes the average effect of both stimuli.

The threshold operation $\tau(A, t, v1, v2)$ sets the value of each cell in the heatmap to v1 if it is below a threshold t, v2 otherwise. This operation is particularly useful for toggling the effect of a stimulus based on the context. For example, a heatmap that encodes the level of emotional distress of a person can be limited to only affect the behavior when the value of the heatmap is above a certain threshold, such as in an emergency.

The normalization operation $\hat{A}$ normalizes the heatmap by setting the maximum value to 1. This operation is useful for ensuring that the values of the cells in the heatmap are in the range $[0, 1]$. This is important because different heatmaps can have different ranges of values, for dissimilar units of measurement, which can affect the result of the operations. For example, a heatmap that encodes a light source can be measured in lumens, while a heatmap that encodes a sound source can be measured in decibels. In this case, the light source heatmap would have a higher range of values than the sound source heatmap, which would affect the result of the operations. Thus, normalizing the heatmaps ensures that the values of the cells are in the same range and that the result of the operations is not affected by the units of measurement.

The inversion operation $!A$ inverts the heatmap by subtracting the value of each cell from 1. This is useful for modeling stimuli that drastically change based on context, without the need to recompute the opposite effect into a new heatmap. For example, a heatmap that encodes the visibility of the environment can be used to model the behavior of a confident person who is attracted by the most visible parts of the environment. The same heatmap can be inverted to model the behavior of a shy person who is repulsed by the most visible parts of the environment, without

the need to recompute the opposite effect into a new heatmap.

3.6 Heatmap Calibration and Evaluation

Current models of human behavior have become increasingly sophisticated to capture the complexity of human reasoning. However, sophisticated models often require many parameters influencing the resulting behavior. Then, it is necessary to calibrate the models' parameters to obtain a realistic and believable simulation. However, first, it is necessary to define metrics for evaluating the simulation's realism and believability. For this reason, parameter calibration and model evaluation are closely related. The calibration process can be seen as a search for the best parameters that maximize the model's evaluation metrics. Thus, the calibration-evaluation process is often iterative: the model is employed in a simulation, the results are evaluated, the parameters are adjusted, and the process is repeated until the convergence criteria are met.

By employing heatmap-based models, the calibration and evaluation process can be improved. The calibration process can be simplified because the model's parameters are encoded in the heatmap and do not depend on the agent architecture. Thus, the calibration process can be reduced to finding the heatmap that best fits the desired behavior.

Existing literature reveals numerous instances where crowd simulations are evaluated through visual comparison with real-world or other simulated data. While providing valuable insights, this method of qualitative, subjective assessment may lack precision and is potentially influenced by individual interpretation.

Within the work of Wolinski et al. [149], images are utilized to depict agent trajectories in the simulation and other models. These visual representations provide a basic level of comparison, offering a qualitative sense of the model's performance. Similarly, Wang et al. [146] employ a method of clustering agent trajectories, striking a balance between microscopic and macroscopic evaluation. While the visual comparison between simulated and real-world data is incorporated, a reliance is seen on custom similarity metrics for more rigorous, quantitative evaluation.

Despite the significant insights these studies offer, a comprehensive quantitative evaluation based on image similarity is notably absent. More traditional metrics, such as path absolute difference, inter-pedestrian distance, and the fundamental diagram, are employed instead. The lack of a structured, quantitative approach to image similarity assessment for crowd simulations is thus a noticeable gap in current evaluation practices.

This gap is highlighted when considering the established value of image similarity metrics in other domains, such as signal processing. Metrics such as the Mean Squared Error (MSE) [147], the Peak signal-to-noise ratio (PSNR) [38], the Structural Similarity Index (SSIM) [37], and the Earth Mover Distance (EMD) [121] are routinely used in these fields to quantitatively assess the quality of an image or signal replication.

The theoretical framework proposed here presents a novel approach: established image similarity metrics are to be leveraged for the quantitative evaluation of crowd

simulations. By transforming the output of simulations into heatmaps, a direct, quantitative comparison with reference data can be achieved, echoing established practices in signal processing. This approach introduces a new dimension to crowd simulation evaluation, offering a systematic, objective measure of how accurately a real-world phenomenon is captured in a simulation.

Heatmaps, as a clustered representation of agent behavior, offer an avenue for comprehensive evaluation of simulation accuracy, going beyond the capabilities of traditional metrics. A balance is struck between high-level evaluation, which focuses on broad metrics like the ratio between crowd density and agent velocities, and low-level evaluation which compares individual simulated agents' trajectories with the real ones. The high-level approach often fails to account for the heterogeneity of human behaviors, while the low-level approach is generally too specific to the exact scenario being simulated.

By recording the activity of real pedestrians and clustering them into a heatmap—a process that can be mirrored with the simulated crowd—heatmaps can be utilized to compare the simulated crowd with the real one. The two heatmaps can then be compared using image similarity metrics such as MSE, PSNR, SSIM, or EMD. This method enables the comparison of various aspects of crowd behavior, from pedestrians' trajectories to velocities, against their real-world counterparts. The results can then be averaged to obtain a single, comprehensive evaluation metric.

In summary, this framework integrates visualization and quantitative evaluation through the use of image similarity metrics, advancing the robustness of simulation quality assessment. This novel approach moves beyond traditional subjective visual assessment towards a more objective and precise evaluation strategy, enhancing understanding of the accuracy and usefulness of crowd simulations.

3.7 Discussion

This chapter has argued that several stimuli can influence the behavior of pedestrians in a crowd. Traditional models of human behavior, which rely on fragmented components of crowd simulation, have been developed over the years to capture the complexity of human reasoning and decision-making at different levels of abstraction. Here, a new theoretical framework for modeling human behavior in crowds is proposed based on heatmaps. The concept of heatmaps in the context of crowd simulation was defined, and a formalization of the operators used to create and modify them was presented. Then, it has been proposed how heatmaps can be leveraged to model the influence of stimuli on the behavior of pedestrians in a crowd at different levels of abstraction: high-level goal selection, global path adjustments, and local steering adaptation. Moreover, the calibration and evaluation of crowd simulation models were discussed, and it was argued that heatmaps help evaluate the simulation's accuracy more comprehensively than other traditional metrics.

Chapter 4

Agora Architecture

The Agora crowd simulation framework is designed to provide a usable, modular, scalable, and versatile solution for simulating and analyzing crowd dynamics in various scenarios. As explained in Section 1.2.1, Agora aims to address the fragmentation and limited interoperability in existing crowd simulation models, which hinders the creation of accurate and realistic simulations of human behaviors and interactions. By offering a unified approach that enables seamless integration and combination of multiple behavior models, Agora facilitates meaningful comparisons and supports the creation of comprehensive and accurate simulations of human behavior in various scenarios and environments.

This chapter presents the architecture of the Agora framework, discussing the design and user considerations, requirements, generic components, and their specific software implementations within the system. The first part of this chapter outlines the design considerations and requirements that guided the development of the framework. By focusing on aspects such as usability, modularity, scalability, and versatility, Agora is designed to provide an effective and user-friendly solution that meets the needs of a wide audience and addresses the constraints of current crowd simulation solutions.

Following the discussion of design considerations, the high-level architecture of the framework is presented, describing the primary generic components and their respective roles and responsibilities. That section emphasizes the importance of a clear separation of concerns, modularity, and maintainability in the design of the framework. The generic components include User Interface, Data Handler, Crowd Generator, Crowd Simulator, Visualizer, Evaluator, and Plugin System.

Next, the software architecture of Agora is further explored, demonstrating how each component corresponds to specific software systems or techniques designed to fulfill its intended function. The backbone of the framework is composed of the integration of Unity 3D and Menge, which collectively address the authoring, simulation, visualization, and evaluation aspects of the crowd simulation process. The software architecture leverages a range of well-established tools and systems to achieve the framework's goals explained in Section 4.1. Throughout this chapter, each component is examined in detail, with its respective implementation presented

and the merits of the chosen architecture argued.

4.1 Requirements and Objectives

This section discusses four key design considerations for the architecture of the Agora framework: usability, modularity, scalability, and versatility. These are summarized in Table 4.1. Each of these considerations directly addresses aspects of the problem statement, which highlights the fragmentation and limited interoperability in existing crowd simulation models.

Usability fosters a convenient authoring process for crowd appearance, environment, and agents' behavior, allowing users to easily create and modify simulations, which addresses the challenge of a lack of user-friendly tools for integrating and comparing models.

Modularity ensures domain independence and extensibility, making the framework applicable to different scenarios and easily integrated with new features. This consideration tackles the problem of closely tied behavior models to their underlying components and technologies.

Scalability includes simulating a large number of agents at interactive framerates while maintaining a balance between the level of detail and performance. This consideration directly addresses the need for more accurate and realistic simulations of human behaviors and interactions.

Versatility allows heatmaps to be applied to various sub-problems of crowd simulation, facilitating meaningful comparisons, behavior combination, reusability, and collaboration. This consideration aims to resolve the fragmentation issue by providing a unified approach to integrating and comparing multiple behavior models.

Objective	Description
Usability	User-friendly interface for configuring crowd appearance, authoring heatmaps and environment elements, and evaluating models.
Modularity	Ensures a flexible and reusable framework through domain-independent design and extensibility, accommodating the changing needs of different domains and facilitating the integration of new features.
Scalability	Balances complexity and performance by managing agent count, behavior variety, and level of detail, ensuring efficient simulations across various scenarios and crowd dynamics.
Versatility	Enables meaningful model comparisons, heatmap integration, reusability, and collaboration, supporting the development of comprehensive behavior models and collective advancements.

Table 4.1: Summary of the requirements and objectives for the crowd simulation framework.

4.1.1 Usability

A *user-friendly* interface for visualizing and authoring crowd appearance, agent behavior, environment, and performing model evaluation is essential for the success of the Agora framework. It increases user engagement, reduces the barriers to entry for less experienced people, and facilitates collaboration among multiple parties. Thus, the Agora framework should satisfy the following requirements:

Authoring Crowd Appearance

It should be possible to author several elements of the crowd's appearance, including the shape, color, accessories, and animations. This allows the creation of a diverse-looking crowd, which is important for the believability of the simulation, as seen in Section 2.2.2.

Authoring Behavioral Heatmaps

The framework should include a user-friendly interface for creating and editing heatmaps that represent human behavior. Moreover, it should be possible to easily combine the heatmaps through color operations, as seen in Table 3.1, to create more complex agent behaviors.

Authoring Environment and Parameters

One should be able to conveniently author several elements of the environment and the related agents' parameters. The former refers to the actual physical appearance of the environment that will host the crowd as well as the number of agents, the

spawning and exit locations, and the obstacles that will be present in the scene. The latter refers to the parameters that relate the agents to the environment, such as the agent's radius and maximum speed/acceleration.

Performing Model Evaluation

The framework should include a user-friendly interface for model evaluation that allows comparing the simulation output with real-world data. It should be possible to easily import location data, such as trajectories, cluster it into a heatmap, and compare it with the corresponding simulation output.

4.1.2 Modularity

To work effectively across various domains and adapt to changing needs, the framework needs to have a *modular* design. This approach involves isolating components, which results in a more flexible and reusable framework. By adopting a modular design, customization becomes more convenient, as extensive changes to the core architecture are not necessary. Additionally, modularity makes it easy to add new features, ensuring that the framework remains relevant to the specific needs of different domains. As such, the following design considerations should be satisfied:

Domain Independent

Heatmaps provide a generic visual representation of how individuals interact with elements in a given environment and are thus a versatile tool for modeling human behavior across multiple domains such as evacuation, public transportation, and large-scale events. The framework should leverage this versatility by providing a generic interface for applying heatmaps to crowd simulation in a domain-independent manner.

Extensible

The framework should be easily customizable to meet the changing needs of different domains and to enable the incorporation of new features and functionalities.

4.1.3 Scalability

The framework should be *scalable* to accommodate varying levels of complexity in simulations. It should be possible to adjust the level of detail of several aspects to strike a balance between complexity and performance.

Number of Agents

The framework should be able to handle an increasing number of agents with a manageable increase in computational complexity, for ensuring that it can be applied to a wide range of scenarios and can efficiently simulate complex crowd behaviors.

Number of Behaviors

The framework should be able to accommodate a growing number of behaviors while maintaining a reasonable level of performance, guaranteeing its applicability across various scenarios and facilitating the efficient simulation of complex social interactions and diverse crowd dynamics.

Level of Detail

The framework should allow the adjustment of the level of detail in various aspects of the simulation, encompassing both graphical elements (such as models and animations) and behavioral components (ranging from simple to complex behaviors).

4.1.4 Versatility

Last but not least, the Agora framework should be *versatile*, promoting consolidation within the crowd simulation field, to address the problem explained in Section 2.5, by fostering the solution of key aspects such as meaningful model comparisons, behavior combination, reusability, and collaboration.

Meaningful Comparisons

The framework should facilitate standardized and straightforward model comparison methods, enabling researchers to conduct meaningful evaluations of their crowd simulation results and fostering coherence and consensus within the field.

Behavior Combination

The framework should support the combination of multiple human social behavior models through heatmap integration, contributing to the development of a comprehensive and holistic model of human social behavior in simulations.

Reusability

The framework should promote the reusability of heatmaps or their generation processes, allowing researchers to seamlessly incorporate these elements into their models or use cases, enhancing efficiency and interoperability.

Collaboration

The framework should encourage collaboration among researchers by streamlining the sharing and integration of behavior models into a repository, enabling the seamless incorporation of diverse behaviors into any simulation, and fostering innovation and collective advancements in the field.

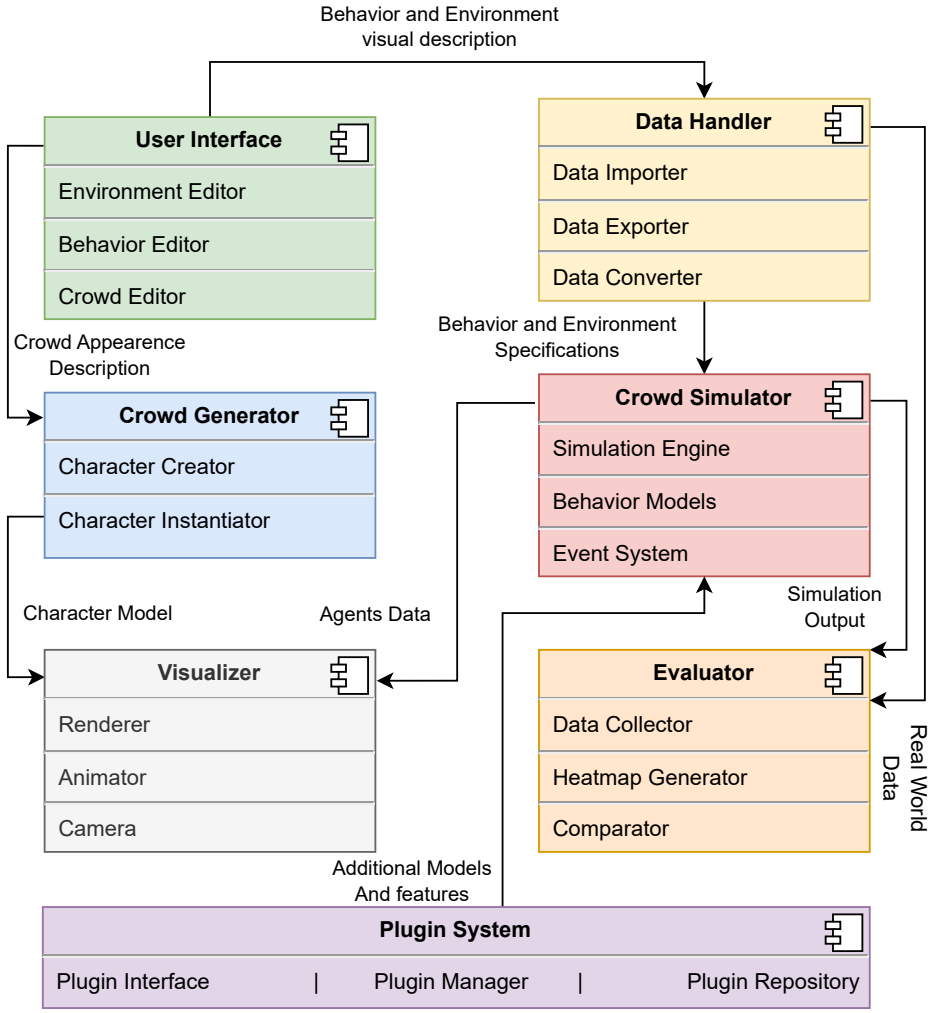

Figure 4.1: This component diagram illustrates the structure of the Agora crowd simulation framework.

4.2 Components Overview

This section presents a high-level description of the crowd simulation framework architecture, focusing on the division of the system into various components and their respective roles and responsibilities. The primary goal of this architecture is to create a system that effectively addresses the challenges associated with crowd simulation (see Chapter 2) while satisfying the considerations explained in the previous Section 4.1.

The proposed framework architecture, shown in Figure 4.1, is divided into seven main components, each of which is further subdivided into specialized subcomponents tailored to address specific aspects of the simulation process. The components include User Interface, Data Handler, Crowd Generator, Crowd Simulator, Visualizer, Evaluator, and Plugin System. The following subsections describe each component in detail, outlining their roles and importance within the architecture, and arguing for the merits of the chosen division.

4.2.1 User Interface

The *User Interface* (UI) component serves as the primary point of interaction between the user and the crowd simulation framework. It is designed to provide a user-friendly and intuitive experience that facilitates the editing of both the environment and agent behavior properties, as well as the management of visualization options. By simplifying these tasks, the UI enhances the user's ability to effectively create, modify, and analyze crowd simulations.

The UI component is responsible for several key functions within the framework, including:

1. **Environment Editing**: The UI allows users to visually and interactively define and modify various aspects of the environment, such as agent spawn locations, obstacles, and boundaries. Moreover, it is capable of visualizing these elements with graphical representations. This seamless interaction enables users to efficiently create and modify simulation scenarios that represent the desired real-world environment.

2. **Behavior Authoring**: The UI facilitates behavior authoring by providing a node-based graphical interface that encapsulates a finite state machine, which has been demonstrated to be a powerful way to model agent behavior by previous research [137]. Each node represents a specific aspect of an agent's reasoning. The UI introduces the innovative concept of heatmap nodes at this stage, which can store behavioral data and offer combination capabilities.

The user interface component's division into distinct functions ensures that each aspect of user interaction is managed effectively. By providing a comprehensive and user-friendly interface, this component greatly enhances the overall *usability* of the crowd simulation framework, satisfying the considerations explained in Section 4.1.1.

4.2.2 Data Handler

The *data handler* component is responsible for managing data import and export within the crowd simulation framework. It ensures seamless data flow and conversion between different formats. This component simplifies user interaction and allows for a more streamlined simulation process by handling data-related tasks, such as generating specification files in the correct format and without errors, which users would otherwise have to do manually. This frees users from the burden of technical details and allows them to focus on the simulation itself.

The primary functions of the data handler are:

1. **Data Import**: Handles the loading and parsing of various input data, such as XML specification files and real-world data. This enables the framework to incorporate user-defined scenarios and behavior models, as well as compare simulation outputs with actual data.

2. **Data Export**: Is responsible for exporting simulation outputs and other relevant data for analysis, visualization, or storage. This ensures that users can easily access and utilize the results of their simulations for further study or evaluation.

3. **Data Conversion**: Facilitates the conversion between different data formats as required by the framework. This includes converting XML files to visual elements for the UI or transforming real-world data into a format suitable for comparison with simulation outputs.

The data handler's organization into specific functions highlights the importance of efficient data management in the framework. As a dedicated component for handling data-related tasks, the data handler ensures smooth data flow and seamless integration between various parts of the framework, thereby increasing its overall *versatility*, as required by the considerations explained in Section 4.1.4.

4.2.3 Crowd Generator

The *crowd generator* component is responsible for generating diverse and realistic-looking crowds of agents based on human templates. By incorporating a comprehensive character creation and modification system, explained in Section 2.2.2, the crowd generator enhances the overall realism and visual appeal of the crowds generated by the framework, making it suitable for a wide range of applications and research purposes.

The crowd generator comprises three main sub-components:

1. **Character Creator**: Is responsible for creating diverse virtual characters with varied traits and features. It offers a system for defining standard human models, composed of fully rigged 3D meshes and corresponding textures, which can be deformed by altering inherent properties to create characters with diverse shapes and sizes.

2. **Character Customizator**: Enables the creation of a collection of clothes and accessories that can be applied to characters. It also allows for the definition of custom textures and color schemes for various character features, such as skin, hair, and eyes, enhancing their visual diversity and individuality.

3. **Character Instantiator**: Offers multiple character creation methods that provide users with various levels of control over the creation process. These methods include manual definition and random generation within defined feature ranges. This versatility allows users to create diverse populations of agents that fit their specific needs.

4. **Level of Detail Manager**: Allows users to control the complexity of the agents' models and animations. This feature helps optimize the performance of the simulation, making it more efficient and allowing for larger crowds.

The separation of the crowd generator into these distinct sub-components highlights the emphasis on producing realistic and visually engaging crowds. By including a dedicated component for generating diverse virtual agents, the architecture maintains a balance between performance and visual quality, thereby enhancing the *scalability* of the framework, as mentioned in Section 4.1.3.

4.2.4 Crowd Simulator

The *crowd simulator* component is at the core of the framework. It is responsible for running the simulation based on the user-defined environment, agent behaviors, and other specified parameters. This component ensures that the simulation accurately and efficiently represents the desired scenarios, taking into account various factors that influence agent behaviors and interactions within the environment.

The crowd simulator is composed of three sub-modules:

1. **Simulation Engine**: Manages the core simulation logic, handling agent updates and interactions between agents and the environment. It processes the input data and computes the resulting agent positions, rotations, velocities, and other relevant parameters based on the specified behavior models and environmental factors.

2. **Behavior Models**: Encapsulates different agent behavior models and provides a modular way to define, modify, and combine them. This enables users to create customized and realistic agent behaviors that accurately represent the desired real-world scenarios.

3. **Event System**: Is responsible for managing events that facilitate communication between the simulator and other components of the framework. This allows for dynamic updates and real-time adjustments during the simulation, ensuring that the system remains responsive and adaptable to changing conditions or user inputs.

By having a dedicated component for running the simulation and managing agent behaviors, the architecture is better equipped to integrate new features and

techniques, ultimately enhancing the framework's overall *versatility*, as mentioned in Section 4.1.4.

4.2.5 Visualizer

The *visualizer* component is responsible for rendering the crowd simulation in a visually accessible and interactive manner. By providing a clear and comprehensible representation of the simulated environment, agents, and behaviors in real time, the visualizer enables users to effectively analyze and evaluate the ongoing dynamics of their simulations.

The visualizer comprises three main parts:

1. **Renderer**: Handles the actual rendering of agents and environments, supporting multiple levels of detail and various graphical representations. The Renderer ensures that the visualization of the simulation is accurate, efficient, and visually appealing, facilitating user understanding and analysis.

2. **Animation System**: Is responsible for managing agent animations and transitions, particularly for more complex 3D meshes. By providing smooth and realistic animations, this sub-component enhances the overall visual quality agents' behaviors.

3. **Camera Controller**: Allows users to interact with and control the camera during visualization, enabling them to navigate the simulated environment and focus on specific areas or agents of interest. This feature contributes to the user's ability to effectively analyze and evaluate the simulation results.

By offering a visually accessible and interactive representation of the simulated environment, agents, and behaviors in real time, the visualizer component effectively addresses the *usability* design consideration discussed in Section 4.1.1.

4.2.6 Evaluator

The *evaluator* component is designed to facilitate the assessment and validation of the crowd simulation outputs by comparing them with real-world data. This component enables users to measure the accuracy and reliability of their simulations, identify areas for improvement, and gain insights into the influence of various factors on agent behaviors.

The evaluator consists of three main elements:

1. **Data Collection**: Records agent positions, agent states, and other relevant data during the simulation. By collecting this data, the evaluator lays the foundation for subsequent heatmap generation and comparison with real-world data.

2. **Heatmap Generator**: Converts the collected simulation data into heatmaps, which provide a graphical representation of agent behaviors and interactions within the environment. These heatmaps serve as the basis for comparison with

real-world data and facilitate the identification of similarities and differences between the simulated and actual scenarios.

3. **Comparison Module**: Compares the heatmaps generated by the simulation with the ones created from real-world data using image similarity metrics. This allows users to assess the accuracy and reliability of their simulations, as well as identify the influence of specific factors, such as overcrowding or group formations, on the simulation outcomes.

The Evaluator component highlights the importance of objectively evaluating simulation models, a factor that is often overlooked in the crowd simulation process. By including a dedicated component for assessing simulation outputs, the architecture fosters continuous improvement, ultimately enhancing the models' performance. As such, this component addresses both *usability* and *versatility*, as discussed in Sections 4.1.1 and 4.1.4.

4.2.7 Plugin System

The *plugin system* component is the part of the crowd simulation framework designed to facilitate the development and integration of custom plugins. It enables users to incorporate new features, algorithms, and techniques, tailoring the framework to their specific requirements. Additionally, it provides a common platform for sharing these extensions, allowing other users to discover and reuse them.

The plugin system comprises three main units:

1. **Plugin Interface**: Defines the standardized interfaces and protocols that custom plugins must adhere to in order to interact with the core framework. The plugin interface ensures a consistent and seamless integration of plugins, allowing them to effectively extend and enhance the functionality of the system.

2. **Plugin Manager**: Is responsible for managing the discovery, loading, and activation of plugins within the framework. This includes handling the registration of plugins, resolving dependencies, and managing their lifecycle. The plugin manager streamlines the integration of custom plugins and ensures their compatibility with the core framework.

3. **Plugin Repository**: Serves as a centralized location for storing and sharing custom plugins, facilitating the distribution and adoption of new features and techniques within the crowd simulation community. By providing a common platform for plugin discovery and access, the plugin repository encourages collaboration and innovation among users and developers.

By providing a structured and flexible approach to plugin development, the plugin system ensures that the framework can evolve with the changing needs of users and advancements in the field of crowd simulation. Moreover, by providing a common platform for sharing these extensions, it allows other users to discover and reuse them, fostering collaboration and innovation within the community. These features satisfy the *modularity* and *versatility* design considerations discussed in Section 4.1.4 and 4.1.2.

Component	Description
User Interface	Enables users to author and configure simulation parameters and agent behaviors.
Data Handler	Handles data input, output, and management for simulation configuration and results.
Crowd Generator	Generates diverse and visually engaging virtual crowds based on human templates.
Crowd Simulator	Executes and manages the crowd simulation and agent behaviors.
Visualizer	Renders the crowd simulation in an accessible and interactive manner.
Evaluator	Assesses simulation performance and compares results with ground truth data.
Plugin System	Integrates custom plugins and extensions to the framework.

Table 4.2: Components of the crowd simulation framework and their brief descriptions.

4.2.8 Summary and Discussion

The proposed framework architecture presents a comprehensive and modular approach to crowd simulation, addressing various aspects of the process from user interaction and data management to visualization and evaluation. The division of the framework into seven main components, each with specialized sub-components, ensures a clear separation of concerns – summarized in Table 4.2. This robust and flexible architecture lays a solid foundation for the development of a crowd simulation framework that can effectively handle diverse scenarios and accommodate the ever-evolving needs of researchers and practitioners in the field of crowd simulation and analysis.

4.3 Agora Software Architecture

In the previous section, a high-level description of the proposed crowd simulation framework's generic components was presented, emphasizing their roles and responsibilities within the system. This section further explores the software architecture of the framework, demonstrating how each generic component corresponds to a specific software system or technique designed to fulfill its intended function, resulting in the definition of the *Agora* crowd simulation framework. For a comparison between the generic and specific components, refer to Figures 4.2 and 4.3.

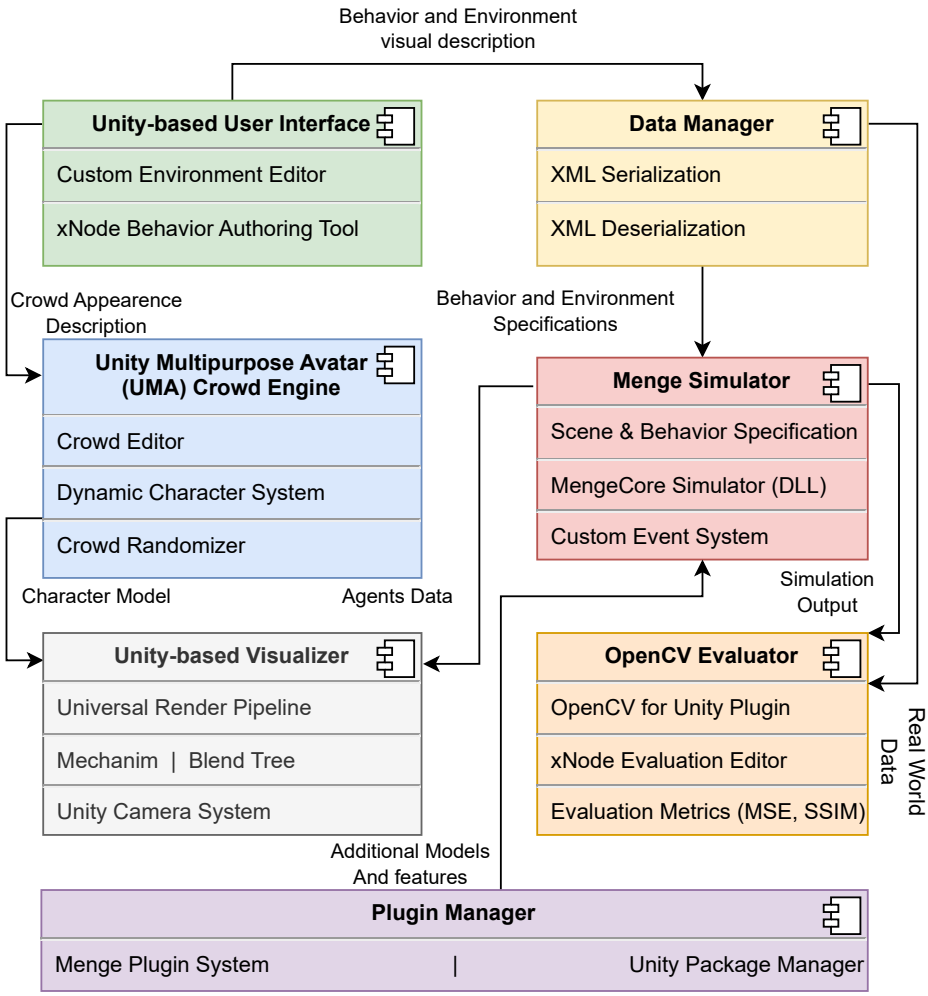

Figure 4.2: This component diagram illustrates the software architecture of the Agora framework. It is based on Figure 4.1, where each generic component has been matched with a specific software system.

The overall backbone of Agora is composed of the integration between Unity 3D and Menge, which collectively address the authoring, simulation, visualization, and evaluation aspects of the crowd simulation process. Unity 3D provides a set of tools for creating, editing and rendering complex environments and characters, while the Menge crowd simulator offers a flexible platform for modeling and simulating agent behaviors in various scenarios. The combination of these two systems results in a comprehensive framework capable of addressing the challenges of the crowd simulation field described in Section 2.5.

The software architecture incorporates a range of components, each designed to perform specific functions within Agora. Overall, the user interface for designing scene specifications is developed within *Unity's custom editors*, while the authoring tool for agent behavior is based on *xNode*. The Data Handler leverages *.NET serialization*, while the crowd generator employs the *Unity Multipurpose Avatar system*. *Menge*, extended to integrate heatmaps, serves as the crowd simulator component, while Unity's *universal render pipeline*, *Mecanim*, and *Cinemachine* handle visualization and interaction. Lastly, the evaluator component relies on *OpenCV* to compare the generated heatmaps with ground truth data, and the plugin system utilizes both Menge's *plugin* architecture and Unity's *package Manager* for extensibility.

The choice of this particular configuration of components provides several advantages in achieving Agora's goals. By leveraging the strengths of both Unity 3D and Menge, Agora can efficiently model and simulate agent behaviors while offering visualization capabilities. Additionally, the use of well-established tools and systems ensures a degree of reliability and maintainability, facilitating future enhancements and modifications. Furthermore, the integration of heatmaps as a core aspect of Agora allows for more holistic representations of agent behaviors, contributing to the overall realism of the simulations.

In summary, the software architecture of the proposed crowd simulation framework aims to address the challenges associated with crowd simulation, as discussed in Section 2.5, offering a comprehensive and extensible solution that takes advantage of the combined capabilities of Unity 3D and Menge crowd simulator.

4.3.1 Unity-based User Interface

The Unity-based *User Interface* (UI) component, developed within the Unity 3D game engine, serves as the primary point of interaction between the user and the crowd simulation framework. By utilizing Unity Custom Editors and the Unity xNode editor, the UI component provides a user-friendly and intuitive experience that streamlines the process of editing both the environment and agent behavior properties. This ultimately enhances the user's ability to effectively create, modify, and analyze crowd simulations.

The Unity-based UI component is responsible for several key functions within Agora, including:

1. **Environment Editor**: Using Unity Custom Editors, the UI allows users to define and modify various aspects of the environment, such as agent spawn locations, obstacles, and boundaries. By visually editing the parameters

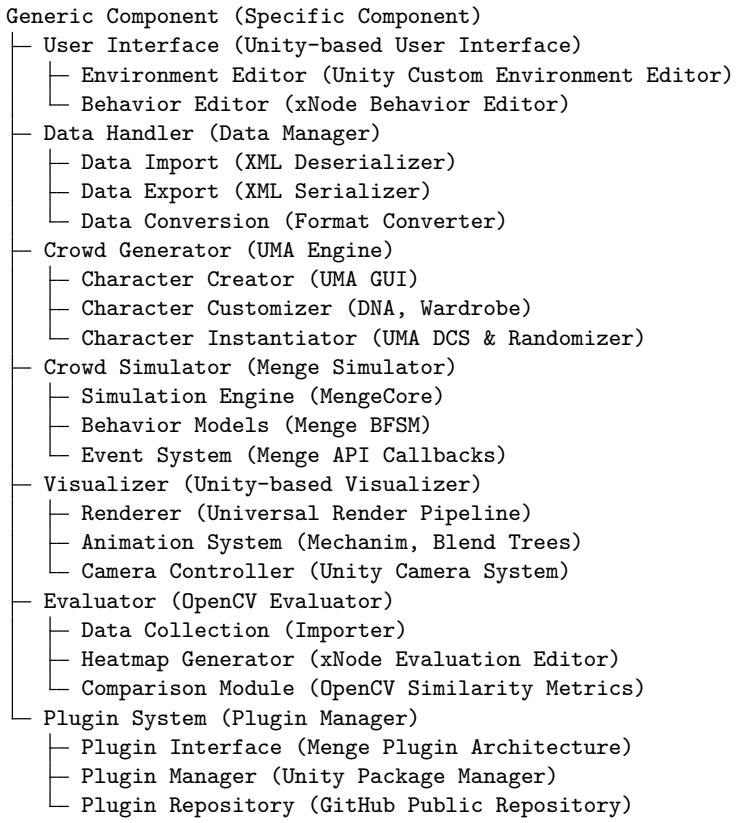

Figure 4.3: Generic vs specific components of the crowd simulation framework.

related to the scene specification and offering a convenient GUI, the UI ensures that the simulation environment accurately represents the desired real-world scenario.

2. **Behavior Authoring**: The Unity xNode editor, a visual scripting tool for the Unity game engine, facilitates behavior authoring through a custom-made node-based graphical interface. Each node represents a specific aspect of an agent's reasoning, corresponding to components of Menge's Behavioral Finite State Machine. Additional custom nodes integrate with the heatmap plugin extension, enabling users to easily leverage heatmaps for influencing agent behaviors. The integration between xNode and Menge BFSM allows for dynamic modification of agent behavior at runtime. For example, the heatmap goal selector can be notified when an agent changes its state and feed the BFSM a new goal based on the relevant heatmap.

The Unity-based user interface component effectively implements the features outlined in the user interface Section 4.2.1 by providing environment editing and behavior authoring functionality through the use of Unity custom editors and the xNode authoring tool. This implementation simplifies the interaction between the user and the crowd simulation framework, aligning with the goals of the generic UI component and enhancing the overall user experience.

4.3.2 Data Manager

The *data manager* component is responsible for the efficient handling of data input and output within the crowd simulation framework, facilitating seamless data exchange between the user, the environment, and agent behaviors. By leveraging the .NET serialization capabilities, specifically the IXmlSerializable interface, the data manager ensures that the simulation data is stored, retrieved, and transformed accurately and reliably.

The data manager component fulfills several essential roles within Agora, including:

1. **Data Serialization**: Utilizing the IXmlSerializable interface, the data manager allows for the customization of object serialization to and from XML. This enables Agora to effectively store and manage simulation data in a structured and human-readable format. By defining custom XML serialization methods, the data manager ensures that objects are serialized consistently and accurately, simplifying the process of saving and loading simulation scenarios.

2. **Data Deserialization**: Similarly, the IXmlSerializable interface also empowers the data manager to deserialize XML data back into objects within the crowd simulation framework. This process is crucial for restoring previously saved simulation scenarios and enabling users to modify and analyze them further. Custom deserialization methods ensure that the data is reconstructed with precision, preserving the integrity of the simulation data.

3. **Data Exchange**: The data manager component also facilitates data exchange between different elements of the framework, such as the Unity-based user interface and the Menge crowd simulator. By managing the transfer of data between these systems, the data manager helps to maintain synchronization and consistency throughout the simulation process.

The data manager addresses the requirements for efficient data management within the Agora framework. By leveraging .NET serialization capabilities, the data manager successfully fulfills the primary functions of data serialization, deserialization, and exchange, aligning with the goals of the generic data handler component noted in Section 4.2.2.

4.3.3 Unity Multipurpose Avatar Engine

The *UMA Engine* is responsible for the creation and customization of dynamic and diverse character models within the crowd simulation framework. By integrating the UMA package, Agora gains access to a modular and extensible system that allows for a wide range of character appearance, clothing, and accessory options, which ultimately enhances the realism and visual variety of the agents in the simulation.

UMA serves several key functions within the framework, including:

1. **Character Generation and Customization**: UMA enables the generation of customizable 3D character models, providing users with the ability to create diverse and realistic agents that better represent real-world populations. UMA's DNA system, inspired by the concept of human DNA, allows users to adjust various aspects of a character's appearance, such as skin color, facial features, and body proportions. This customization extends to clothing and accessories, with numerous options that can be combined and tweaked to achieve the desired visual representation of agents.

2. **Agent Instantiation**: In addition to creating single characters with predefined appearances using the Dynamic Character System (DCS), UMA features a randomizer that lets users define ranges for the DNA values and probabilities for accessories and colors. This powerful tool allows for the dynamic generation of populations of agents, which will probabilistically exhibit the specified traits, resulting in varied and realistic crowd simulations.

3. **Performance Optimization**: To maintain a balance between visual fidelity and performance, UMA incorporates several optimization techniques, such as texture and mesh atlasing. These techniques help minimize the rendering overhead associated with displaying large numbers of unique characters in a crowd simulation, enabling users to run simulations with large crowds without sacrificing visual quality or simulation speed.

The integration of UMA into Agora significantly enhances the system's capabilities in generating visually diverse and realistic agents. By offering a flexible and extensible solution for character generation and customization, instantiation with both predefined and probabilistic appearances, and optimization, UMA offers an effective implementation of the requirements of Section 4.2.3.

4.3.4 Menge Simulator

The crowd simulator is built upon *Menge*[1], a flexible and modular framework for simulating crowds and developing innovative pedestrian models. Menge was developed by Curtis et al. at the University of North Carolina - Chapel Hill. It decomposes crowd simulation into computational components which address various aspects of crowd simulation such as goal selection, global path planning, and local collision avoidance. In this work, Menge has been integrated more closely with Unity and extended to incorporate the heatmap paradigm for influencing agent behavior. This enhanced version is designed to interact seamlessly with other software components of Agora, including authoring through the User Interface, obtaining input and producing outputs with the Data Manager, and displaying 3D animated agents via the Unity Visualizer.

The three main subcomponents of the Menge simulator are as follows:

1. **Scene Specification**: The scene specification delineates the elements of the simulation, encompassing static obstacles, elevation objects, spatial query mechanisms, and agent populations. By incorporating a graphical user interface as an extension of the Unity editor to author scenes and parameters, the process of setting up simulations can be streamlined, thereby reducing the time required to create realistic and dynamic environments tailored to specific use cases.

2. **Behavior Specification**: The behavior specification ascertains how the simulation's state evolves. Menge employs the concept of a Finite State Machine (FSM) to encode behaviors, which govern agents' goals, strategies, and characteristics. In the present use case, the heatmap paradigm is utilized to influence agent behavior, connecting it to the xNode component of the user interface. This approach permits more intuitive and efficient control over agent actions and reactions within the simulated environment.

3. **Event System**: The simulation pipeline has been extended to encompass arbitrary callbacks that can be set by an external entity, such as the Simulation Manager in Unity. This extension enables Menge to perceive and respond to external data, adjusting agent behaviors in real time accordingly.

In summary, the integration of Menge with Unity and the incorporation of the heatmap paradigm provide significant advantages. This combination enables users to easily create realistic simulations and explore various pedestrian models and techniques. Additionally, it promotes a more interactive and responsive simulation environment, facilitated by the event system, which ultimately expedites the advancement of crowd simulation methods. This effective implementation successfully addresses the requirements of Section 4.2.4.

4.3.5 Unity-based Visualizer

The Unity-based *Visualizer* component, leveraging the Unity 3D game engine, is responsible for the real-time visualization of the crowd simulation environment and

[1] http://gamma.cs.unc.edu/Menge/

agents. By utilizing advanced rendering and animation techniques, such as the Universal Render Pipeline (URP), Mecanim, as well as a versatile Camera System called Cinemachine, the Visualizer component significantly enhances the realism and user experience of the crowd simulation framework.

The Unity-based Visualizer component comprises the following key elements:

1. **Universal Render Pipeline**: provides advanced graphics features and performance optimization tools for rendering virtual environments. In the context of the Agora framework, URP is used to visualize both the environment where the simulation takes place and the agents that are part of the crowd simulation. Menge handles the backend simulation, continuously streaming output data such as agent positions, orientations, and velocities. These data are then visualized by the Unity-based Visualizer in the form of optimized 3D human meshes, generated by the Crowd Generator.

2. **Mecanim**: The Unity Animation System, Mecanim, is a toolkit for crafting animations, which can be employed to introduce diversity to pedestrian movements in crowd simulation. In this context, *blend trees* enable developers to blend various animations, including those from external services like Mixamo, based on an agent's state, producing seamless transitions that enhance the realism of the visualization. In the Agora framework, this system is used to animate the 3D human models rendered by the URP. The agent velocities streamed by Menge during the simulation are used as input to a blend tree that appropriately interpolates Mixamo animations to match speed and direction, resulting in realistic and diverse pedestrian movements.

3. **Camera System**: Unity's Camera System, captures and displays scenes from different perspectives, offering features for adjusting position, field of view, and projection. *Cinemachine* supports multiple camera views and can create custom effects using scripting or post-processing. In the Agora framework, this component is employed to explore and observe the simulation as it unfolds, providing users with the ability to gain unique insights into the behavior of the simulated agents and the overall performance of the crowd simulation.

In summary, the Unity-based visualizer effectively implements the requirements described in Section 4.2.5. By leveraging advanced rendering and animation techniques, such as the Universal Render Pipeline, Mecanim, as well as the versatile Cinemachine camera system, this system ensures a visually accessible and interactive real-time representation of the simulated environment, agents, and behaviors. This comprehensive visualization enhances users' ability to effectively analyze and evaluate the ongoing dynamics of their simulations.

4.3.6 OpenCV Evaluator

The *Evaluator* component is designed to objectively assess the realism of the crowd simulation model by comparing the generated heatmaps with their ground truth counterparts. It achieves this by employing OpenCV for Unity and a custom

node-based xNode editor window, which allows users to effectively leverage heatmap nodes and perform evaluations seamlessly.

1. **Evaluation Window**: The Evaluator component is designed with a custom node-based xNode editor window, similar to the behavior editor, which makes it convenient for users to perform evaluations. This window allows users to effectively leverage heatmap nodes, combine them (in cases where there are multiple outputs), and employ comparator nodes to obtain a similarity metric between the generated heatmap and ground truth data.

2. **OpenCV for Unity**: Performs the comparison of heatmaps in the crowd simulation framework. OpenCV is a computer vision library that can be integrated into the Unity game engine through plugins. It provides a wide range of functionalities, including comparing images using objective similarity metrics such as Mean Squared Error (MSE), or the Structural Similarity Index Measure (SSIM). In the context of the Agora framework, this is employed to compare the heatmaps generated by the crowd simulation with ground truth data, enabling an objective evaluation of the simulation's performance and realism.

The novel integration of OpenCV with the xNode user interface not only emphasizes the importance of evaluating simulation models but also enhances the evaluation process by offering a range of features. The Agora OpenCV Evaluator is capable of converting agent trajectories into heatmaps, visually representing agent behaviors, and providing a convenient, node-based GUI for user interaction. By offering the ability to compare heatmaps and generate similarity scores, the OpenCV Evaluator effectively implements the requirements described in Section 4.2.6.

4.3.7 Plugin Manager

The *plugin manager* component of the Agora framework is designed to offer extension capabilities by leveraging both Menge's plugin architecture and the Unity Package Manager. This approach provides a flexible platform for developers to extend the functionality of the crowd simulation framework, share agent behaviors, and introduce new features.

The plugin manager is composed of two elements with different functions:

1. **Menge Plugin System**: Menge's plugin architecture allows developers to extend the functionality of the crowd simulation framework by adding new plugins. These plugins can be used to customize or add new simulation behaviors or data, such as new agent types or steering behaviors. For instance, in the case of Agora, the Menge plugin system enabled the compartmentalized implementation of the heatmap plugin, offering additional capabilities for the crowd simulator.

2. **Unity Package Manager**: The Unity Package Manager is a tool that manages packages in Unity projects, allowing developers to import, update, remove,

and create custom packages. This functionality enables the distribution of the entire Agora framework as a Unity package, making it easily accessible and usable for users. By utilizing the Unity Package Manager, Agora can be easily shared and extended by the community, fostering collaboration and innovation.

By leveraging the strengths of both Menge's plugin architecture and the Unity package manager, the Agora framework's plugin manager provides an extensible platform for sharing crowd simulation components. This approach encourages the development of new features, agent behaviors, and tools, ultimately contributing to the ongoing improvement and expansion of Agora in the broader community. These features satisfy the requirements described in Section 4.2.7.

4.3.8 Summary and Discussion

In summary, the Agora framework's software architecture employs a combination of well-established and specialized software systems, summarized in Table 4.3, such as the Unity-based user interface, the UMA engine, the Menge simulator, and the OpenCV evaluator. These systems were carefully chosen, integrated, built, and extended to provide a cohesive, intuitive, and efficient experience for users. This allows Agora to be an effective solution capable of addressing various requirements in crowd simulation and agent behavior modeling.

Component	Description
Unity-based User Interface	Visual editor and node-based scripting tool for simulation configuration and agent behavior, with heatmap plugin integration.
Data Manager	Handles I/O and management of simulation data using .NET serialization and IXmlSerializable interface for XML data formatting.
UMA Engine	Generates diverse virtual crowds using UMA, providing a modular and extensible architecture for character customization.
Menge Simulator	Simulates crowd behavior with several algorithms offered by Menge extended with the Heatmap Plugin for influencing agent behavior.
Unity-based Visualizer	Uses Unity's URP for rendering, Mecanim for animation, and the Camera System for visualization.
OpenCV Evaluator	Uses OpenCV for Unity similarity metrics to compare simulation heatmaps against real-world data, objectively evaluating the simulation.
Plugin Manager	Enables customization and extension of the framework through Menge's plugin architecture and Unity Package Manager.

Table 4.3: Specific components for the crowd simulation framework and their brief descriptions.

Chapter 5

Agora Implementation

This Chapter presents a comprehensive overview of the implementation of the Agora crowd simulation framework, detailing the development and integration of its various components. Some elements of the software architecture were extended from existing solutions, while others were built from scratch, all coming together to form a cohesive crowd simulation framework. Section 5.1 discusses the integration of Menge and Unity, showcasing how these frameworks were combined to create crowd simulations. Section 5.2 focuses on the Unity Multipurpose Avatar, detailing the creation, animation, and instantiation of agents within the crowd. Section 5.3 explores scene authoring and the process of designing realistic environments for simulations. In Section 5.4, the focus is on behavior authoring, discussing the tools and techniques used to create and manage agent behaviors within the framework. Section 5.5 presents the Menge heatmap plugin, explaining its implementation and various functionalities that enable the use of heatmaps within the simulation. Finally, Section 5.6 discusses the OpenCV Evaluator, describing the process of converting positional data to heatmaps, comparing heatmaps, and providing a graphical user interface for evaluation purposes.

5.1 Menge-Unity Integration

This section discusses the integration of Menge and Unity, two of the primary components of the Agora crowd simulation framework. It covers the integration of Menge with Unity, the adaptation of a native plugin loader, and the extension of the Menge C-API. These features contribute to a more robust and efficient simulation framework.

5.1.1 Menge so Far

Menge, a powerful, cross-platform, modular framework for crowd simulation [32], serves as a foundational component for the Agora framework. Being a versatile and extensible platform, Menge offers a solid basis for research and development in various aspects of crowd simulation, such as behavioral modeling, global path

planning, and local collision avoidance. Its plug-in architecture enables researchers to focus on specific aspects of crowd simulation while relying on built-in functionality for other components.

Menge is written in C++ and compiled into a Dynamic Link Library (DLL), a library of functions that can be loaded and executed at runtime, enhancing portability and enabling code reuse across different applications. Menge exposes a C API, making it possible to use the simulator with other programming languages, such as C#.

A C# wrapper project, MengeCS, produces a DLL that can be included in C# projects, providing access to the Menge simulation core. This simple wrapper of the Menge C-API allows for the initialization of the Menge simulator, stepping through simulations, and querying minimum agent state.

There is already a rudimentary integration between Menge and Unity 3D, another backbone component of the Agora framework. The Unity project provides an initial structure and organization for integrating the two systems. This integration primarily includes a *SimController* script capable of running a simulation with Menge and visualizing a basic output where agents are displayed as 3D mesh cylinders. This initial integration lays the groundwork for the development of a more comprehensive and user-friendly crowd simulation framework in Agora.

Menge, being a C++ DLL, requires integration with Unity, which runs on C#. To better understand the challenges and solutions in achieving this integration, it's essential to discuss the differences between managed and unmanaged code.

Native code, also known as unmanaged code, is compiled directly into machine language, which is executed by the computer's hardware. It provides better performance but lacks features such as automatic memory management and type safety, which are characteristic of managed code. Managed code runs on a virtual machine—in the case of .NET, it's the Common Language Runtime (CLR) with Just-In-Time (JIT) compilation. Managed code offers benefits such as memory management, type safety, and platform independence.

Interoperability options between managed and unmanaged code include Platform Invocation Services (P/Invoke) and C++/CLI. P/Invoke enables managed code to call unmanaged functions in dynamic link libraries. This method is simple and offers high performance but is limited to specific function signatures and requires careful memory management. C++/CLI, on the other hand, is a language extension that allows seamless integration between C++ and .NET languages. It provides a more natural and flexible interface between the two but may introduce overhead and additional complexity.

Menge relies on P/Invoke for its interoperability, enabling C# code in Unity to call functions in the Menge C++ DLL. This approach requires careful consideration of memory management and function signatures, ensuring compatibility between the two systems.

Unity, in dealing with native code, provides facilities for integrating unmanaged code through plugins, allowing developers to utilize the performance benefits of native code while still leveraging the features and ease of use offered by the Unity environment. This integration between Menge and Unity lays the foundation for the Agora framework and paves the way for further development of a comprehensive

and user-friendly crowd simulation system.

The remainder of this section will focus on discussing a selection of challenges and solutions encountered during the process of integrating Menge and Unity.

5.1.2 Native Plugins in Unity

The Problem. By default, native plugins in Unity do not unload, leading to garbage instance retention. This occurs when native class destructors are not called, resulting in retained references to garbage instances. This retention can cause memory leaks, resource mismanagement, and even memory access violations, potentially leading to undefined behavior, crashes, or data corruption in the application. Implementing a system that enables the loading and unloading of native plugins at runtime, like the Agora framework's native plugin loader, is crucial to mitigate these issues and enhance the efficiency and stability of the development process.

The Solution. In the implementation of the Agora framework, the native plugin loader[1] system was adapted to address the challenges associated with working with native plugins in Unity. This solution enables the loading and unloading of native plugins, such as DLL files, at runtime, streamlining the integration of native code (e.g., Menge C++ libraries) with managed code (C#) in Unity.

By utilizing custom attributes to designate C# classes as native plugin wrappers and static fields as native plugin functions, the NativePluginLoader class efficiently handles the loading and binding of native libraries and their functions. Crucially, it manages the unloading of native plugins when required.

This approach resolves the issue of Unity not unloading native plugins by facilitating their explicit unloading and reloading during runtime. Consequently, developers can modify and update native plugins without restarting the Unity editor, enhancing the development process's efficiency.

Moreover, this system ensures that consecutive simulation runs remain independent and consistent, as native plugins can be unloaded and reloaded as needed. This eliminates resource conflicts from previous runs, preventing memory leaks, crashes, or other unexpected behaviors.

The native plugin loader in the Agora framework dynamically loads Menge's DLL and its dependencies when the play button is pressed in Unity, leveraging the *Awake*[2] Unity event. Awake is a built-in Unity event that is called when a script is initialized, allowing the simulator to efficiently integrate Menge for simulation purposes.

Once Unity exits play mode, the *OnDestroy*[3] event of the Simulator is triggered, unloading all native plugins to ensure a fresh state for the next simulation run. OnDestroy is another built-in Unity event that is called when a script is being destroyed or when the application is closing. This process significantly improves the stability and consistency of consecutive simulation runs, making the development process more efficient and predictable when working with native plugins in Unity.

[1] https://github.com/forrestthewoods/fts_unity_native_plugin_reloader
[2] https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html
[3] https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnDestroy.html

5.1.3 Menge C-API Extension

Significant improvements have been made to the Menge C-API, enhancing its capabilities and facilitating more complex interactions with the underlying simulation framework. This section highlights two key advancements: the addition of functions for greater interaction with the behavioral finite state machine, and the implementation of event callbacks to inform external entities of crucial transitions. These extensions enable more sophisticated use cases, such as heatmap-based goal selection, and foster improved interoperability between Menge and other systems like Unity, ultimately resulting in a more powerful and flexible simulation framework.

Additional Funcitons. The original Menge API provided basic functionality for simulator initialization and querying agent states. The simulator could be initialized by supplying initialization files, including scene and behavior specifications. Additionally, the simulation could be started and basic agent states, such as positions and velocities, could be queried. However, access to the behavioral finite state machine governing the agents' behavior was not granted.

To overcome this limitation, the Menge API was extended to allow greater interaction with the behavioral FSM. For instance, functions were implemented to obtain an agent's goal and retrieve the goal selection mechanism of the agent state. This extension proved particularly useful for a plugin feature that facilitates goal selection based on a heatmap.

With the heatmap-based goal selection plugin, the agent state can be retrieved and the goal selection mechanism checked to determine if it is managed by an external entity. If so, a new goal can be assigned to the agent based on the heatmap. By extending the Menge API, more complex interactions with agent states have been enabled, enhancing the overall capabilities of the system.

External Agent Generators. Menge, as a configuration-centric framework, initializes all its elements by parsing configuration files for the scene and behavior at the beginning of the simulation. This is true for agents as well, which are defined in the configuration file and instantiated only during the initialization phase. However, this approach is unrealistic and limiting in terms of dynamically spawning agents at runtime.

To overcome this limitation, a new method for agent instantiation was implemented, which involved saving the relevant instances of agent initializers and profile selectors (required for creating new agents) in the simulator. Previously, these instances were discarded by default after the initialization, as they were not used for spawning new agents during the simulation.

Next, a new base class of persistent agent generators was created, which would be maintained in memory instead of being discarded after initialization. A derived class, called "external" agent generator, was introduced to spawn agents on demand when prompted by an external entity (such as a Unity-based simulator).

Lastly, these new functions were exposed through the C-API, enabling the spawning of agents programmatically at runtime using `C#` function calls. This new implementation allows for a more flexible agent generation during the simulation.

Event Callbacks. The Menge simulation pipeline was adjusted to include event callbacks, addressing the limitations in the original implementation where external entities could not be informed about certain transitions, such as changes within the behavioral finite state machine. During the simulation process, Menge executes a series of functions, one of which involves advancing the behavioral finite state machine. This step encompasses evaluating the current state's transitions to ascertain if a new state should be entered based on pre-defined conditions.

In the context of this use case, it is essential to determine if the BFSM's new state includes a heatmap goal selector. If so, Unity will supply a new goal based on a heatmap, which is elaborated further in the Heatmap Goal Selection Section. To facilitate this functionality, event callbacks were implemented in the simulation pipeline.

The event callback was devised by defining an AgentChangedStateCallback function, which is invoked when the BFSM is on the verge of transitioning into a new state. This function informs any subscribers of the impending transition, providing the agent ID and the name of the new state. Through this focused modification, external entities can now respond appropriately to state transitions, enhancing the overall capabilities and interoperability of the Menge simulation framework, while maintaining a streamlined implementation.

5.2 Unity Multipurpose Avatar

This section discusses the integration of the Unity Multipurpose Avatar (UMA) framework to generate realistic and diverse virtual crowds. It covers the creation of a fitting population using custom DNA values and wardrobe, the animation of agents, and their instantiation in the simulation environment. These adaptations contribute to a more believable and dynamic simulation.

5.2.1 Creating a Fitting Population

In the UMA engine, two fundamental human templates for male and female characters are provided. The DNA comprises values for various body parts, with some being cumulative, such as height, while others are specific, like arm length. Employing randomizers, ranges of DNA values can be defined to establish a diverse population of agents. With a curation process carried out in this work, a selected set of values has been established, along with appropriate colors for skin, hair, and eyes, ensuring the generation of a realistic and diverse population in appearance, particularly fitting for the urban environment of case study two (Section 6.2). UMA incorporates the concept of a wardrobe, which encompasses the collection of meshes that can be attached to the basic agent template. This primarily includes clothing but may also feature hairstyles and other accessories that enhance the simulation's realism and make agents appear more lifelike. For example, smartphones can be utilized during idle cycles, allowing agents to appear busy texting or calling instead of merely standing still.

A key challenge is ensuring that these additional meshes adapt based on the deformations imposed by the DNA values on the human template. Specifically,

clothing must stretch or shrink to accommodate the varying sizes of agents. Furthermore, these meshes must animate correctly, following the agent's body as they move.

The default UMA wardrobe is limited and does not provide sufficient clothing and accessories to create a realistic population suitable for common environments, especially in colder climates such as Iceland. To address this, additional clothing and accessories have been designed and incorporated into the UMA wardrobe, showcased in Figure 5.1. This results in a more varied population that blends seamlessly into the environment.

To achieve this, 3D meshes were adapted from online marketplaces such as Sketchfab[4]. This process entailed sourcing suitable 3D meshes for clothing and importing them into Blender, along with the basic UMA male or female model, depending on the mesh being adapted. The UMA template would be positioned in an A-pose, with legs and arms spread out. Utilizing the tools provided by Blender, the clothing mesh was modified to fit the UMA model. Once a satisfactory result was obtained, the clothing mesh was weight painted to ensure it moved correctly in accordance with the UMA model during animation. Finally, the refined product was imported into Unity, and a custom recipe was developed to enable the clothing to be equipped by UMA agents and added to the UMA global library.

Figure 5.1: A diverse UMA crowd showcasing custom DNA values and wardrobe, illustrating the varied appearances of agents fitting a northern european urban environment.

[4]https://sketchfab.com/

5.2.2 Animating the Agents

UMA enables the use of a Unity animation controller to animate agents and facilitate their movement. The animation controller functions as a state machine, featuring states and transitions that govern the agents' actions. The custom animator created primarily introduces variety in the agents' idle cycles, ensuring natural appearances while standing, and facilitates believable movement throughout the environment.

The animator consists of several states, including idle and locomotion, both of which are regulated by blend trees that seamlessly blend relevant animations. The idle state encompasses multiple animation poses, and as time elapses while an agent remains stationary, the blend tree interpolates the animations based on time as a parameter, causing the agent to change its pose. This creates the impression of a varied idle stance rather than repetitive motion. To prevent synchronization among multiple agents' stance changes, the regulating parameter begins with a random value for each agent, resulting in distinct and dynamic pose alterations over time.

The locomotion state's blend tree, shown in Figure 5.2, manages the agents' animations during movement. It contains ten different substates, determined by the direction and speed of the movement. For instance, if an agent moves forward quickly, the "sprint forward" animation is played, whereas a rapid backward movement triggers the "run backward" animation, with interpolations for variations in speed.

Notably, these animations do not influence the agents' positions, which are instead controlled by Menge. These animations, known as *in-place animations*, only affect the visual appearance of the agents while they navigate the environment. During each timestep of the simulation, the Unity SimulationController script acquires the instantaneous velocity of each agent from the MengeSimulator and assigns it to the animator, which then visually represents the moving agent appropriately.

The blend tree's versatility allows for the addition of more parameters to further enhance the variety of animations. In this work, the age of the character was used, allowing the blend tree to impact both idle poses and walking animations. Consequently, this creates a more varied and realistic representation of agent behaviors in the simulation.

5.2.3 Instantiating the Agents

When the simulation starts and the Menge simulator is initialized using the scene specification file containing the initial positions of the agents, the Simulation-Controller script in Unity employs the agent spawner (UMA random avatar) to instantiate the correct number of agents in their designated positions. Subsequently, the random population definition is utilized to generate a diverse crowd of agents, ensuring a varied and dynamic simulation population.

5.3 Scene Authoring

Menge utilizes scene specification files to define the elements of the simulation, including static obstacles, elevation objects, spatial query mechanisms, and agent populations. These files are written in XML format, which can render the authoring

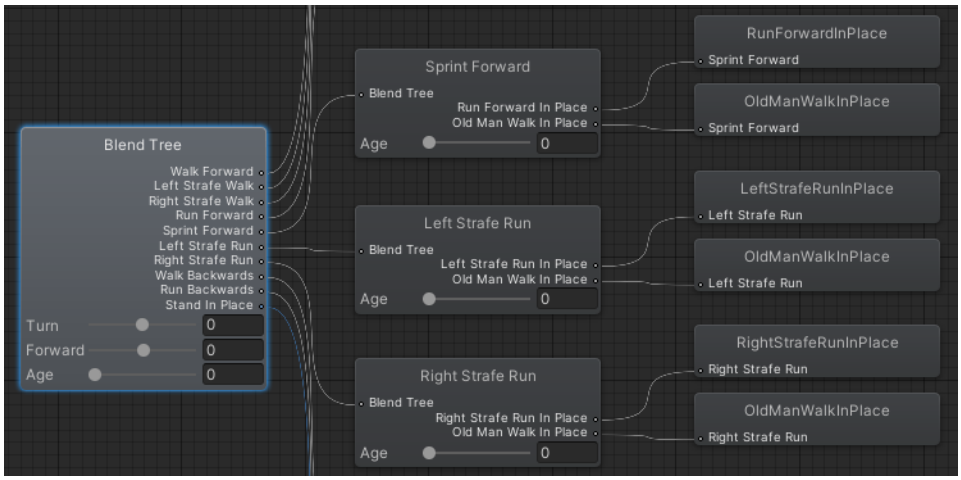

Figure 5.2: Part of the animation blend tree that animates the UMA virtual agents. It interpolates between the various animations based on the agent's movement speed and direction.

process cumbersome and prone to errors. Syntax errors or poorly formatted XML files may not be parsed by the simulator, resulting in a nonfunctional simulation. Additionally, determining the agents' spawn locations in the simulation can be challenging since the agent generator coordinates are not visually represented. Consequently, the authoring process can be inconvenient, highlighting the need for a graphical user interface that simplifies this process.

The scene authoring tool, shown in Figure 5.3 is a custom Unity Editor window designed to serve several functions. It enables the selection of a Menge scene specification XML file, which could be an existing file containing a simulation's specifications or an empty file. In the former case, the file will be parsed into the appropriate classes; otherwise, it will simply serve as a reference for saving the authored file. Parsing the file required partially recreating Menge's scene specification class structure in C#. The appropriate .NET attributes were then used to manage the serialization process[5], allowing the conversion of XML scene specification tags into corresponding C# objects.

The GUI facilitates a more intuitive scene parameter authoring process for the simulation, eliminating the need to write raw XML. Furthermore, it visually represents some graphical elements of the scene within Unity Editor's 3D space, enabling users to interactively modify them. For instance, agent generators can be dragged around the environment rather than manually editing the XY positions by inputting values. Once the user has completed authoring the scene specification file, they can save it, and the editor will serialize the content back into XML format. This approach greatly enhances the authoring process's robustness, engagement, and convenience.

[5]https://learn.microsoft.com/en-us/dotnet/standard/serialization/controlling-xml-serialization-using-attributes

Figure 5.3: Example of a Menge pedestrian simulation in an urban environment running inside the Agora visual framework. Superimposed are the navmesh and the interactive scene specification GUI.

5.4 Behavior Authoring

Menge employs behavioral finite state machines to determine the behavior of agents within the simulation. This is achieved through the use of XML in a behavior specification file. However, the process presents several challenges that impede its effectiveness. Directly writing the behavior specification in XML can be counterintuitive, making it difficult for users to grasp the intricate logic and structure of the behavioral state machines. The complexity of the behavioral state machines encompasses multiple aspects, such as agent goals, states that include goal selection, global pathfinding, obstacle avoidance, and transitions with intricate conditions.

Due to the nature of XML, understanding and authoring the interactions among these multifaceted elements can be cumbersome, and it is easy to lose track of the relationships between different components of the behavioral state machines. In addition to the challenges in visualizing the overall structure and connections between the elements, the manual authoring process itself can be prone to errors, such as syntax issues or improper formatting. Such errors may result in a malfunctioning simulation, necessitating extensive troubleshooting to identify and resolve the problem.

These challenges emphasize the need for a more user-friendly approach to

authoring agent behaviors, one that simplifies the process and allows for a clearer understanding of the interactions between the various components of the behavioral state machines. This can be achieved through a graphical user interface that facilitates a more intuitive authoring experience, minimizes the risk of errors, and streamlines the process of creating and modifying behavioral state machines, shown in Figure 5.4.

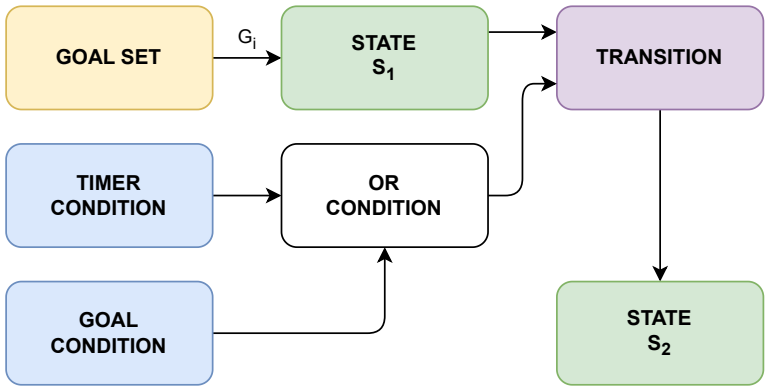

Figure 5.4: A simple visual representation of a Menge BFSM, which includes an initial state where agents head toward a specific goal. The transition conditions specify that once agents reach the goal or after a certain amount of time has elapsed, they will shift into another state where they come to a halt.

5.4.1 xNode Behavior Authoring Tool

xNode is an open-source Unity plugin that offers a flexible framework for developing custom node-based systems within the Unity editor. This plugin enables the creation of intuitive graphical interfaces, such as visual scripting tools or behavior trees, using a node-based approach with customizable nodes connected to define relationships or data flow.

In Menge, the behavioral finite state machine forms the core of agent behaviors. Each state in the BFSM determines the agent's goal, and the methods for pursuing that goal, and can even impact the agent's fundamental characteristics, simulating changes in mood or thought. Transitions between states direct alterations in agent behavior.

Utilizing xNode for visualizing and authoring BFSMs in Unity offers numerous benefits. The node-based approach provides a clear visual representation of the BFSM structure, simplifying the understanding and management of complex agent behaviors. Custom nodes can be created to meet the specific needs of a BFSM, facilitating a more efficient development process. Furthermore, the graphical interface allows for more convenient creation and modification of BFSMs compared to traditional text-based methods like XML.

In xNode, nodes serve as the fundamental building blocks, while ports enable communication between these nodes. Each element of the BFSM has been en-

capsulated within an xNode node, allowing them to be instantiated within the graph. This implementation simplifies the process of creating and modifying BFSM elements, ensuring a more intuitive and user-friendly experience when designing agent behaviors within the Unity environment, as shown in Figure 5.5.

During the development process, custom editor windows were created for each of the nodes to provide additional functionality in a user-friendly manner. In some cases, this necessitated a reevaluation of Menge's class hierarchy and deviation from the XML structure to create a more intuitive GUI.

For instance, in Menge's original XML hierarchy, goals are defined within a parent *GoalSet* tag. A direct translation of this structure would result in a *GoalSet* xNode containing a list of *Goal* nodes. However, implementing this approach would lead to a cumbersome GUI, as supporting various features of the *Goal* node within a nested window inside the *GoalSet* editor would be challenging.

To address this issue, custom editor windows were developed for each of the goals and the goal set separately, connecting them using node ports. This approach enhances the user experience by streamlining the process of configuring goals and goal sets, making it more visually appealing and easy to manage.

These user experience design considerations were carefully observed and implemented throughout the entire xNode GUI, ensuring an efficient and intuitive workflow for authoring agent behaviors.

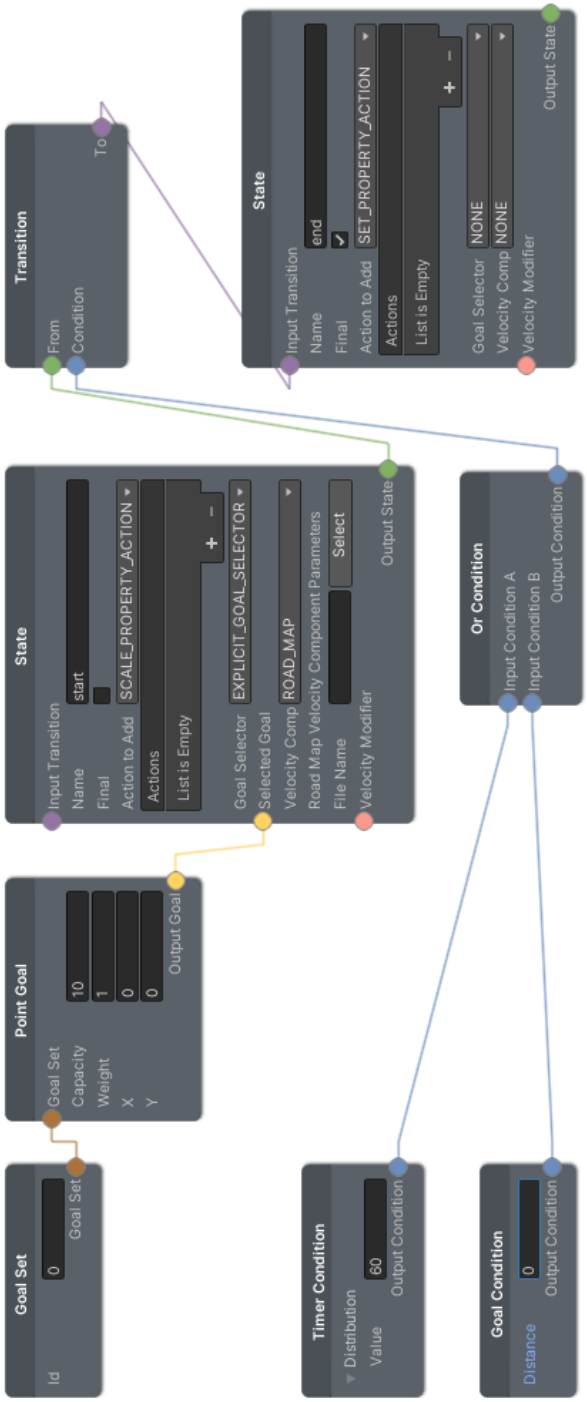

Figure 5.5: The xNode behavior authoring tool in Unity for the BFSM graph of Figure 5.4. The distinction between the class hierarchy of *GoalSet* and *Goal* can be observed.

5.4.2 xNode Heatmap Nodes

Besides the standard Menge BFSM elements implemented as xNode nodes, additional nodes have been developed to manage the new heatmap-related features. A notable custom node facilitates the incorporation of heatmaps to influence agent behaviors within the Menge plugin. This supports the selection of a Texture2D representing a heatmap and displays a preview of the image. As a result, it becomes possible to encode a variety of data into heatmaps for guiding the agents.

The true value of this node emerges when it is used in conjunction with another custom node, known as combiners. Combiners are designed to execute a range of color operations, summarized in Table 3.1, enabling the merging of multiple heatmaps. This functionality permits the individual authoring of various stimuli that affect human behaviors, which can subsequently be integrated to achieve a more comprehensive modeling of the underlying behavioral patterns.

The combiners function efficiently by employing Unity shader graphs, which allow to visually design and construct shader programs. By taking advantage of Unity shader graphs, the combiners can execute color operations and heatmap blending directly on the GPU, which considerably enhances performance compared to carrying out these operations on the CPU.

The application of shader graphs for combiners presents two primary benefits. Firstly, the performance improvement arises from shaders operating on the GPU, allowing for parallel processing capabilities and significantly alleviating the computational burden on the CPU. Secondly, the visual programming aspect of shader graphs simplifies the development of new functions or operations, as they can be readily created and incorporated into a new node. This method refines the development process while ensuring optimal performance.

Building on this foundation, the output generated by the combiners can be utilized in a wide array of heatmap-related features to influence agent behavior. For instance, the resulting combined heatmap can play a significant role in goal selection. By analyzing the heatmap, agents can identify the most appealing locations within the environment, thereby driving their movement toward these areas. This facilitates a more realistic and nuanced representation of how individuals navigate their surroundings based on various stimuli.

Moreover, the heatmap data can be employed in local obstacle avoidance mechanisms. In this context, agents can adapt their steering behavior according to the information provided by the heatmap. This approach enables agents to dynamically respond to the environment and make informed decisions in real-time, further enhancing the accuracy and believability of their behavior.

In summary, the integration of combiners and heatmap-related nodes within the xNode behavior authoring tool offers a powerful and flexible means for incorporating diverse stimuli into agent behavior modeling. By enabling the combination of multiple heatmaps and the application of the resulting data to various aspects of agent behavior, such as goal selection and local obstacle avoidance, a more comprehensive and realistic representation of human behavior can be achieved within the Menge heatmap plugin, as discussed in the next section.

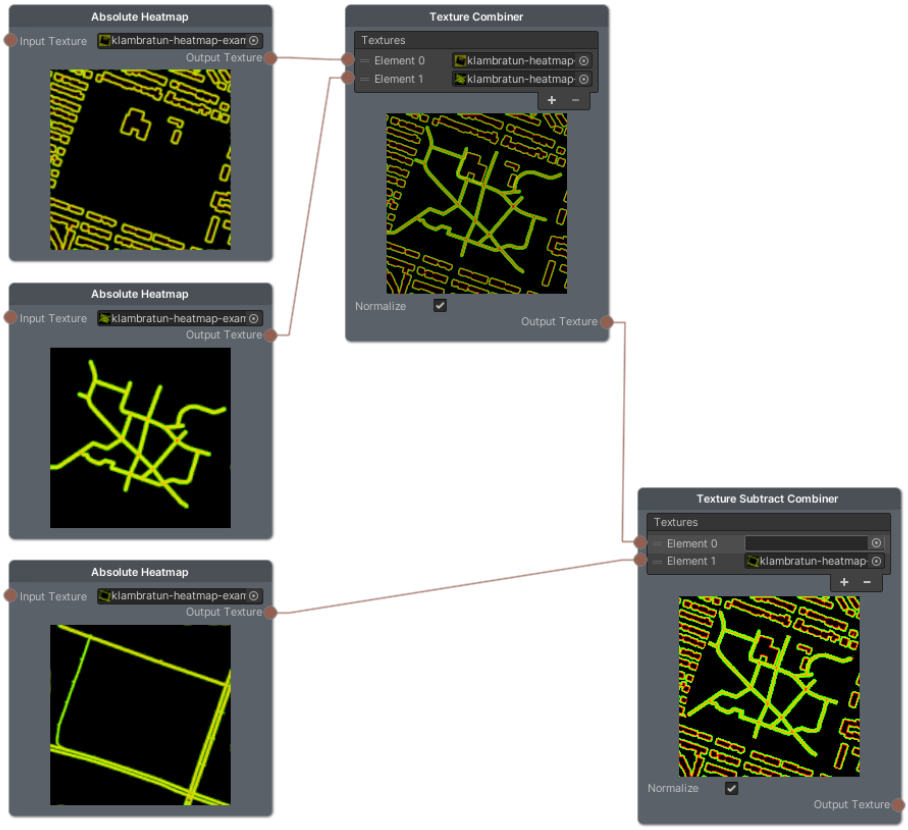

Figure 5.6: The xNode Editor for Interacting with Menge's Heatmap Plugin Using Images. The authored stimuli are the ones discussed in Figure 3.8.

5.5 Menge Heatmap Plugin

This Section discusses the implementation of the Menge heatmap plugin, which enables the integration of heatmaps into the Menge simulation framework. The plugin encompasses a heatmap class, a heatmap goal selector, a heatmap velocity modifier, and a heatmap transition which are discussed in detail in the following subsections.

5.5.1 Heatmap Implementation

The new heatmap plugin is built upon a dedicated class designed to manage the heatmap data and offer diverse functionalities for interacting with it. This class is a crucial component in the overall Agora implementation, allowing for the seamless integration of heatmaps into the simulation framework.

One of the primary features of this class is the capability to handle conversions between world coordinates and pixel coordinates. This is achieved by considering the scale and offset factors to ensure proper alignment of the heatmap data with the world environment in which the agents operate. Such functionality is essential for the accurate representation and utilization of heatmap data in the simulation.

Additionally, the class encompasses methods that allow users to query color values at specific positions within the heatmap. These color values can be used to influence agent behavior based on the heatmap data. A higher-level functionality is also provided, enabling the retrieval of color values directly from specified world coordinates. This further streamlines the integration of heatmap data into the simulation.

The implementation also includes a function responsible for loading heatmap data from a file, returning a managed pointer to an instance of the class. This ensures that the heatmap data is easily accessible and manageable within the simulation environment.

5.5.2 Heatmap Goal Selector

The Heatmap Goal Selector is a crucial component in the Agora crowd simulation framework, responsible for determining agents' goals based on spatial heatmaps. Heatmaps represent the desirability of different locations in the environment, with color values assigned to each point on the map, reflecting the attractiveness of that location. This section discusses two approaches to goal selection using heatmaps: the Menge-based implementation and an alternative Unity-based implementation. Both methods focus on efficiently selecting goals for agents by considering various factors, including the agent's position, orientation, and perception of the environment. This comprehensive goal-selection process enhances the overall crowd simulation experience by providing more realistic and dynamic agent behavior.

Menge-Based Implementation (Static). The HeatmapGoalSelector is designed to effectively select goals for agents in the Menge crowd simulation framework by leveraging a heatmap. The heatmap represents the desirability of different

locations in the environment, with each point on the map being assigned a color value that corresponds to the location's attractiveness.

This custom goal selector uses a heuristic function to choose the best goal for each agent. The heuristic function combines various factors, such as the heatmap's color information, the angle between the agent's current orientation and the direction to the potential goal, and the distance from the agent to the potential goal. The use of a heuristic function allows for a flexible approach to goal selection that can be tailored to various scenarios and agent behaviors.

One example of a heuristic function involves the agent's field of perception, as shown in Algorithm 1. The field of perception is determined by the agent's position, orientation, perception angle, and perception range. Within this field, a set of points are sampled to identify potential goal locations. The selector then evaluates each sampled point based on the following factors:

1. **Color information from the heatmap**: The color value at each sampled point reflects the desirability of that location. Higher color values indicate more desirable locations, while lower values signify less desirable ones.

2. **Angle to the point**: The angle between the agent's current orientation and the direction to the sampled point is considered. Ideally, agents should select goals that require minimal turning, so points with smaller angles are favored.

3. **Distance to the point**: The distance from the agent's position to the sampled point is also taken into account. Goals that are closer to the agent are generally preferred, as they require less time and effort to reach.

By combining these factors, the heuristic function computes a score for each sampled point, ultimately choosing the point with the highest score as the agent's next goal.

Unity-Based Implementation (Static/Dynamic). A key feature of the SimulationManager is the implementation of the AgentChangedStateCallback function, designed to respond to events triggered when agents alter their states during the Menge simulation.

The AgentChangedStateCallback function acts as an event handler that assigns a new goal to an agent under specific conditions, as described in Algorithm 2. When an agent's state changes, the function is called with the agent's ID and the new state name as arguments. The script verifies if the new state possesses an external unity-side goal selector, which signifies a custom goal selection strategy managed by Unity. In the absence of such a selector, the function returns without assigning a new goal to the agent.

If an external goal selector is present, the SimulationManager searches the mengeBFSMSceneGraph, a representation of the behavior finite state machine employed by Menge to model agent behavior, for a matching state node. The HeatmapGoalSelector is subsequently retrieved, using a texture or heatmap to establish the agent's new goal based on its current position and orientation.

Within Unity authoring framework, the HeatmapGoalSelector node, shown in Figure 5.7, offers an alternative method for assigning goals to agents based on a

Input: Agent's position p, orientation θ, perception angle α, perception range r, heatmap H
Output: Selected goal g

foreach *angle in* $[-\frac{\alpha}{2}, \frac{\alpha}{2}]$ **do**
  $sampleDir \leftarrow \theta.rotate(angle)$
  for $sampleDist \in [0.01, r]$ **do**
    $samplePoint \leftarrow p + sampleDir \times sampleDist$
    $color \leftarrow H(samplePoint)$
    $score \leftarrow$ computeScore$(color, angle)$
    $sampledPointsScores.insert(\{score, samplePoint\})$
  end
end
$pointWithHighestScore \leftarrow$ getHighestScorePoint$(sampledPointsScores)$
$newGoal \leftarrow$ createNewGoal$(pointWithHighestScore)$
return *newGoal*

Algorithm 1: Goal Selection Process Based on Agent's Perception and Heatmap Information.

heatmap, extending the Menge-based implementation's capabilities. This custom approach delivers a dynamic and efficient goal selection process that seamlessly integrates with the SimulationManager, taking into account both the agent's position and their perception of the environment.

Upon locating the HeatmapGoalSelector, the SimulationManager calls its getGoal method to compute the new goal, considering the agent's position in the 2D heatmap coordinates. The getGoal method employs a custom shader and a compute shader to generate a resulting heatmap, reflecting the agent's perception of the environment. This resulting heatmap is then used to identify the most desirable location for the agent.

Once the new goal is determined, the SimulationManager adjusts its position back to the 3D scene coordinates by incorporating an offset. Subsequently, the Menge library's setStatePointGoalForAgent method is invoked to assign the new goal to the agent, completing the integrated goal selection process.

In essence, the HeatmapGoalSelector and the SimulationManager work in tandem to provide a cohesive and unified goal-selection process. This process enables dynamic goal assignment based on the agent's perception of the environment and its position within it. The Unity-based implementation of the HeatmapGoalSelector thus provides a valuable alternative to the Menge-based approach for goal assignment using heatmaps, enriching the overall crowd simulation experience.

Differences between the approaches. Both the Menge-based and the Unity-based Heatmap Goal Selector implementations offer unique advantages in the context of crowd simulation. The Menge-based implementation does not require Unity, making it more accessible and easier to integrate into various other simulation frameworks. However, its main limitation is the static nature of the heatmap, which

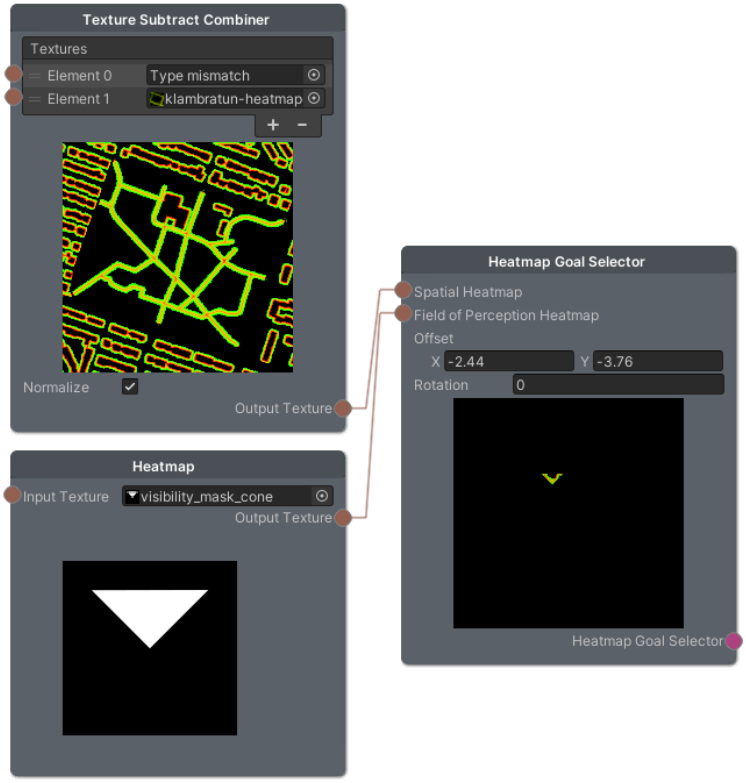

Figure 5.7: HeatmapGoalSelector node combining a spatial heatmap, and a cone-shaped perception field using a custom shader that adjusts tiling and offset for accurate agent perception. The resulting composed heatmap is a portion of the original, reflecting the agent's environment and location for dynamic goal selection.

Input: Agent ID, New State Name
Output: Assigned Goal
function AgentChangedStateCallback(agentID, newStateName)
if *newState has external goal selector* **then**
    Find matching state node in mengeBFSMSceneGraph
    Retrieve HeatmapGoalSelector
    Compute agent's position in 2D heatmap coordinates
    newGoal ← getGoal(agentPosition)
    Adjust new goal's position to 3D scene coordinates
    Invoke setStatePointGoalForAgent to assign new goal to agent
end
function getGoal(agentPosition)
    Create temporary render texture
    Blit heatmap and perception field textures using custom shader
        Calculate UV ratio and apply agent position as offset
    Feed resulting texture to compute shader
    Calculate highest value pixel coordinates
    Translate pixel coordinates to world coordinates
    Instantiate new PointGoal
    return *new PointGoal*

Algorithm 2: Heatmap-based goal selection process.

is defined at the start of the simulation and remains unchanged throughout.

Conversely, the Unity-based implementation, while requiring Unity integration, offers a more powerful and flexible approach to goal selection. It allows for dynamic changes to the heatmap during the simulation, providing a more realistic and adaptable representation of the environment. Additionally, the Unity implementation benefits from increased performance, utilizing shaders and shader graphs to efficiently process the heatmap data and accurately determine agents' goals.

Ultimately, the choice between the Menge-based implementation and the Unity-based approach depends on the specific requirements and constraints of the crowd simulation project. Both methods contribute valuable tools for effective goal selection, enhancing the overall realism and dynamism of agent behavior within the simulation.

5.5.3 Heatmap Velocity Modifier

The new Menge *HeatmapVelocityModifier* builds upon the previously described heatmap functions to effectively guide agents through the environment, as described in Algorithm 3.

The heatmap class allows projecting agents from world space to heatmap space and extracting relevant information, such as color values, for the agents' positions. The heatmap modifier uses these functions to sample colors from the heatmap at various points within the agents' vision field. By doing so, it can determine the attractiveness or repulsiveness of each sampled location based on the heatmap data.

In the context of Menge's simulation pipeline, the heatmap modifier comes into play after a goal has been selected and a global plan has been devised to reach that goal position. The velocity modifier operates on the premise that, while the global plan provides a general direction for agents to follow, the heatmap modifier serves as a supplementary guide to influence the agents' movement decisions. This additional guidance steers agents around the environment based on the heatmap's hot and cold zones, promoting more informed and context-aware navigation.

The adaptPrefVelocity function is responsible for incorporating the heatmap data into the agent's preferred velocity. It first checks if the agent has a current subgoal based on the heatmap; if not, the algorithm samples colors from the heatmap within the agent's vision field, calculates scores for these samples, and selects the highest-scoring point as the new subgoal. Once a subgoal has been established, the function adjusts the agent's preferred velocity to move toward the subgoal. This process allows the agent to smoothly navigate the environment with the influence of the heatmap data.

By integrating the heatmap into Menge's existing velocity modifier, the code enhances the agents' decision-making process within Menge's simulation pipeline, allowing them to better respond to the environment's hot and cold zones as they navigate towards their goals.

```
Data: Agent, Heatmap
Result: Adapted PrefVelocity
if Agent has no SubGoal then
    SampleColors ← SampleHeatmapColors(Agent, Heatmap);
    SampleScores ← CalculateSampleScores(SampleColors);
    SubGoal, SubGoalScore ← SelectHighestScoringPoint(SampleScores);
    if SubGoalScore > Threshold then
        Agent.HasSubGoal ← True;
    end
else
    DistanceToSubGoal ← CalculateDistance(Agent.Position, SubGoal);
    if DistanceToSubGoal < MinDistanceToSubGoal then
        Agent.HasSubGoal ← False;
    end
end
if Agent.HasSubGoal then
    AdjustedDirection ← SubGoal - Agent.Position;
    Agent.PrefVelocity ← AdjustedDirection.normalized() *
     Agent.PrefSpeed;
else
end
```

Algorithm 3: Adapt Preferred Velocity based on Heatmap

5.5.4 Heatmap Transition

The ColorCondition class is an essential component of the Agora simulation framework, designed to detect specific conditions based on agents' positions and their interactions with a heatmap. This versatile class can be used for various purposes, such as representing "personal zones" around agents, where specific actions or behaviors occur once another agent enters the zone.

In conjunction with the Behavior Finite State Machine (BFSM), the ColorCondition class enables state transitions and adaptation of an agent's behavior based on their interactions with other agents in their personal zones. These zones can be marked with unique RGB colors on the heatmap, allowing the ColorCondition class to detect when agents enter or leave these zones. For example, agents can change their speed, direction, or goal when another agent enters their personal zone, resulting in more intelligent and context-aware behaviors that better mimic real-world interactions.

The class works by storing a reference to the heatmap information and the desired RGB color for the condition. Its main method, conditionMet, checks whether the color condition is met by iterating through all the agents in the simulation, excluding the agent for which the condition is being checked. It calculates the relative position between the agent and the other agents, projects it onto the heatmap, and obtains the corresponding RGB color values. If any of the calculated RGB colors match the specified condition RGB color, the method returns true, signaling that the condition is met.

By detecting the corresponding RGB color values on the heatmap and allowing the BFSM to transition between states when the specified color conditions are met, the ColorCondition class offers a powerful tool for simulating complex agent interactions and behaviors.

```
Function conditionMet(agent, conditionColor)
    foreach testAgent in simulation do
        if testAgent.id ≠ agent.id then
            relativePosition ← agent.position - testAgent.position
            mapColor ← SampleHeatmapColors(relativePosition)
            if mapColor == conditionColor then
                return true
            end
        end
    end
    return false
```

Algorithm 4: Checks whether the condition is met by projecting agent positions onto a heatmap and comparing the obtained colors to the specified condition color.

5.6 OpenCV Evaluator

The OpenCV Evaluator is an essential component of the Agora crowd simulation framework, designed to assess the performance and accuracy of the simulation models by comparing them to real-world data. The core concept behind the evaluator is to generate heatmaps based on arbitrary data from the simulation, create corresponding heatmaps using real-world data, and then compute a similarity score by comparing the two heatmaps using image similarity metrics. A higher similarity score indicates that the simulation model closely replicates the real-world phenomenon.

The implementation of the evaluator consists of three main components: importing real-world data into the framework and generating heatmaps from it, producing comparable heatmaps from the simulation, and calculating similarity metrics between the heatmaps. To facilitate the computation of various image similarity metrics, OpenCV for Unity is utilized. Furthermore, an evaluation GUI based on xNode is implemented to streamline the evaluation process.

5.6.1 Positional Data to Heatmap

This section explains the process of generating heatmaps from both real-world and simulated data using the Agora framework. The purpose of creating these heatmaps is to evaluate and compare the accuracy of crowd simulation models. By generating two distinct heatmaps, one for real-world positional data and the other for simulated data, the effectiveness of the simulation in reproducing actual movement patterns observed in real-world scenarios can be assessed.

Real World Data. The Agora framework includes an importer component within its evaluation module, which is designed to handle the conversion of real-world spatial data, recorded in arbitrary ways, into a suitable format for generating heatmaps. The importer reads a JSON file containing spatial locations with latitude and longitude, parses the data into the appropriate format, and creates a spline representation for each data point. A spline is a smooth curve that passes through or near a set of control points, which makes it a convenient way to visualize and potentially modify spatial data. Moreover, the data points can include additional information, such as orientations and timestamps, which can be utilized to create more detailed and informative heatmaps for evaluation.

The Agora framework's evaluation module includes a component that uses spline representations of real-world paths to generate heatmaps. The component first verifies if a position from the spline lies within the map boundaries. If the position is inside the map, it increases the corresponding heatmap location's value by a constant amount. After processing all positions, the heatmap is normalized to ensure that the values range from 0 to 1.

Using splines in this context offers several advantages, particularly when working with sparse data. For example, if the data being analyzed consists of GPS locations sampled at relatively long intervals, the splines can help interpolate and substantiate this data. This interpolation leads to a more comprehensive representation of

the real-world paths, providing a better foundation for generating heatmaps and evaluating the accuracy of the crowd simulation models.

Simulated Data. The evaluator module in the simulation framework incorporates a method for extracting relevant data and turning it into a heatmap using Unity Events. The simulation manager, a singleton, is responsible for keeping track of spawned and despawned agents. When a new agent is spawned, a listener can be added to it, which is triggered whenever the agent performs an action, such as moving. This allows for updating the respective heatmap cell accordingly.

By simulating a scenario over a determined period, it is possible to generate a heatmap containing the relevant data. The process involves registering a callback function that gets triggered whenever an agent moves. As the simulation progresses and agents move around the environment, the heatmap is updated to reflect the agents' movements and locations.

5.6.2 Heatmaps Comparison

In this section, the process of comparing heatmaps generated from real-world and simulated data to evaluate the effectiveness of crowd simulation models is discussed. The similarity between the two heatmaps serves as an indicator of how well the simulation captures real-world movement patterns. A high degree of similarity suggests that the simulation model accurately represents the observed real-world phenomenon, while a low similarity indicates that the model may not be performing well in replicating actual movement patterns.

Numerous methods are available for evaluating image similarity. In this study, the following methods are chosen for implementation: Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Earth Mover's Distance (EMD). Utilizing OpenCV as a tool for heatmap comparison is advantageous for several reasons. First, OpenCV provides bespoke implementations for some of the chosen metrics, which streamlines the comparison process. Second, even for those metrics that OpenCV does not directly implement, the library simplifies matrix operations, making it easier to develop custom implementations.

By employing these image similarity metrics and leveraging the capabilities of OpenCV, heatmaps generated from real-world and simulated data can be effectively compared. This comparison enables the determination of the accuracy of the crowd simulation models and assesses their ability to reproduce real-world movement patterns.

Mean Squared Error. The Mean Squared Error (MSE) metric is a widely used quantitative measure for comparing two images, in this case, heatmaps. The metric calculates the average squared differences between the corresponding pixel values of the two images. A smaller MSE value signifies that the two images are more similar, whereas a larger MSE value indicates greater dissimilarity between the images.

In the context of heatmaps, the MSE is computed by first checking the dimensions of the input images. If the dimensions are different, a warning is displayed, and an error value is returned. For images with matching dimensions, the process continues

by converting the Texture2D format of the heatmaps into OpenCV Mat objects, the way that OpenCV stores 2D matrices. The absolute difference between the corresponding pixel values of the two Mat objects is then computed and squared. The mean of the squared differences is calculated, resulting in the MSE value.

The raw MSE value is normalized to make it more interpretable. The normalization process involves dividing the raw MSE value by the maximum possible pixel value and multiplying by 100. The result is a similarity score ranging from 0 to 100, with higher scores indicating greater similarity between the two heatmaps.

Peak Signal-to-Noise Ratio. The Peak Signal-to-Noise Ratio (PSNR) is a widely used image quality metric, especially when evaluating the quality of images or videos that have undergone lossy compression. PSNR is expressed in decibels (dB) and describes the relationship between the maximum possible power of a signal and the power of the corrupting noise affecting the fidelity of its representation.

To compute the PSNR, one needs to calculate the Mean Squared Error (MSE) between two input images and the maximum possible pixel value (e.g., 255 for 8-bit images). The formula for PSNR is given by:

$$PSNR = 10 \cdot \log_{10} \left(\frac{R^2}{MSE} \right) \tag{5.1}$$

In this formula, R represents the maximum integer value of the image depth, and MSE is the mean squared error between the two images (which should have the same type and size).

When comparing heatmaps, the PSNR metric is employed to assess how well a simulated heatmap matches a real-world heatmap. A higher PSNR value indicates a better match, as the differences between the two images are smaller. On the other hand, lower PSNR values suggest a larger discrepancy between the simulated and real-world heatmaps, implying that the simulation may not accurately capture the real-world phenomenon. OpenCV has a built-in function for calculating the PSNR between two images, which is used in the evaluator.

Structural Similarity Index. The Structural Similarity Index Measure (SSIM) is a method for predicting the perceived quality of digital images and videos. It measures the similarity between two images, with values ranging from -1 (no similarity) to 1 (perfect similarity). SSIM is a perception-based model that takes into account image degradation as a perceived change in structural information, luminance masking, and contrast masking. Unlike the MSE and PSNR, which focus on pixel-level differences, SSIM captures the structural, luminance, and contrast information of the images, making it better suited for evaluating the similarity between heatmaps, especially when the heatmaps have local and global structures or patterns that are relevant for the comparison.

The SSIM calculation is based on comparing local patterns of pixel intensities between two images, using sliding windows. The SSIM index is defined as:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{5.2}$$

where x and y are windows of the images being compared, μ_x and μ_y are the average pixel intensities of the two windows, σ_x^2 and σ_y^2 are the variances of the pixel intensities in each window, σ_{xy} is the covariance of pixel intensities between the two windows, and C_1 and C_2 are constants to prevent division by zero.

SSIM is widely used for various applications such as image compression, image restoration, and pattern recognition. However, it is computationally expensive due to the need for comparing local patterns using sliding windows.

Earth Mover's Distance. The Earth Mover's Distance (EMD) is a measure of dissimilarity between two probability distributions. It can be used to compare various types of data, including images, such as heatmaps. In the context of comparing heatmaps, EMD measures the "minimal work" required to transform the spatial distribution of intensity values in one heatmap to match the other, where "work" refers to the product of the mass (or intensity value) being moved and the distance it is moved. Compared to other metrics that focus on pixel-by-pixel differences, EMD provides a more intuitive and meaningful measure of dissimilarity between two heatmaps, as it accounts for the spatial distribution of the intensity values. This makes EMD more suitable for cases where the relative spatial distribution of intensity values is of greater importance than individual pixel differences, such as when comparing heatmaps generated from the simulation with those encoding real-world data. EMD is computed as follows:

$$EMD(P,Q) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j}d_{i,j}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j}} \tag{5.3}$$

Where P and Q are the two distributions being compared, m and n are the number of points in each distribution, $f_{i,j}$ represents the flow between point i in distribution P and point j in distribution Q, and $d_{i,j}$ denotes the distance between the two points.

EMD can effectively assess the similarity between two heatmaps by quantifying the least amount of effort needed to change the distribution of intensity values in one heatmap to match the other. To calculate the Earth Mover's Distance between two images, they must be represented as signatures, which are matrices containing point coordinates and associated weights. The EMD then measures the minimal cost of transporting mass (intensity values) from one signature to another, given a certain distance metric.

OpenCV offers an implementation of EMD that can be used to compare heatmaps or other types of images, however, the computation of EMD can be computationally expensive, particularly for large distributions.

5.6.3 Evaluator GUI

In order to make the evaluation process more accessible and user-friendly, an xNode GUI has been developed. This graphical user interface is designed to facilitate the comparison of heatmaps through a node-based approach, featuring three main types of nodes:

1. **Heatmap nodes**: These nodes serve as the fundamental input for the evaluation process, allowing users to select and input the heatmaps they want to compare. Heatmap nodes provide an easy way to manage and organize the data used for evaluation.

2. **Combiner nodes**: Combiner nodes are particularly useful when dealing with layered data. In scenarios where the evaluation involves comparing the effects of multiple stimuli on pedestrian movement or other phenomena, you may have one heatmap for each stimulus. Combiner nodes allow you to merge these individual heatmaps, creating a clustered result that can be compared against the aggregated real-world data.

3. **Evaluator nodes**: There is one evaluator node for each of the evaluation metrics discussed earlier. These nodes let users perform the actual evaluation by comparing the input heatmaps. Each evaluator node features two inputs for the heatmaps being compared and an additional input for an optional mask. The mask allows users to focus the evaluation on the relevant portions of the input heatmaps. A button initiates the evaluation process, and the output results are displayed on a label and could be plugged into other nodes. Where possible, the results are normalized and color-coded to provide a clear visual representation of the similarity between the heatmaps.

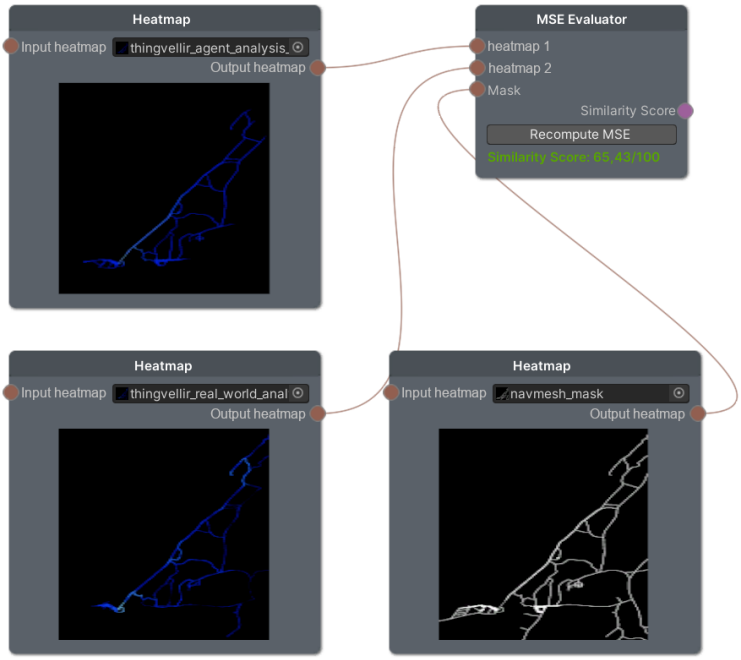

Figure 5.8: The xNode evaluator GUI, showcasing four nodes in action. Two heatmap nodes represent the real-world data of walking paths and the simulated paths, respectively. The third heatmap is a mask that defines the relevant parts of the inputs that are going to be compared. The heatmaps are inputted into an MSE evaluator node, which calculates a matching score of approximately 65%.

Chapter 6

Case Studies

This chapter presents two case studies designed to assess the various features of the Agora framework and demonstrate its capabilities in simulating crowd behavior in different real-world scenarios. These case studies serve as practical examples, illustrating the usability, versatility, and effectiveness of the framework in supporting the development and evaluation of crowd simulation models.

The first case study, set in Þingvellir National Park, investigates the impact of visibility on pedestrian navigation while focusing on assessing the convenience and support provided by the Agora framework in the crowd simulation modeling process. Additionally, this case study explores the effectiveness of the evaluation function, specifically the node-based image similarity evaluator. The second case study, centered on an urban environment, examines the framework's capabilities in combining agent behaviors through the integration of guiding heatmaps, specifically by combining the thermal and density comfort theories. Both case studies aim to showcase the potential improvements offered by the Agora framework in the field of crowd simulation and establish its value as a versatile and reliable tool for researchers and practitioners.

6.1 First Case Study: Þingvellir

This case study aims to examine the usability and versatility of the Agora framework, by simulating tourists' behavior in a national park in Iceland, Þingvellir. The primary objectives of this case study are to investigate the impact of visibility on pedestrian navigation, to evaluate the accuracy of the simulation model by comparing the simulated results with real-world data, and to assess the convenience and improvement provided by the Agora framework in the simulation process.

In this case study, the idea that visibility might affect the way people navigate the environment is explored, with individuals potentially being more attracted to visible locations. This concept serves as the basis for developing a simulation model using an assortment of components tailored to address specific aspects of the simulation process. The case study aims to showcase the advantages and potential improvements offered by the Agora framework in the field of crowd simulation by

applying it to a real-world scenario and comparing the results with those obtained through traditional methods.

The relevance of this case study to the Agora framework lies in its role as a benchmark to measure the framework's capabilities compared to commonly adopted solutions in the field of crowd simulation. As argued in Section 2.5, researchers and practitioners often resort to a combination of isolated technologies to develop crowd simulation models, relying on a specific selection of components from various sub-fields. To encourage the adoption of the Agora framework, it must demonstrate its ability to support the development of crowd simulations at least as effectively as these existing solutions, if not better.

This case study serves as a crucial test for the evaluator component of the Agora framework. By comparing heatmaps produced by simulations with those derived from actual real-world data, the quality of the simulation model can be evaluated using image similarity metrics. This process not only enables the quantification of the simulation model's accuracy but also demonstrates the effectiveness and utility of Agora's evaluation mechanism in facilitating the creation of realistic simulations.

By applying the framework to a real-world scenario and comparing the results with those obtained through traditional methods, this case study aims to showcase the advantages and potential improvements offered by the Agora framework in the field of crowd simulation.

6.1.1 Unity Native Simulation

This section describes the creation of an ad-hoc crowd simulation model in Unity to study the impact of spatial visibility on agent navigation within Þingvellir national park. The process involved acquiring map data and developing software to analyze visibility and generate heatmaps, which were used to model agent behavior in the simulation environment using Unity scripts and tools. Notably, this work took place prior to the development of the Agora framework, and the approach employed can be considered an early precursor to the general heatmap approach discussed in the theoretical framework. The resulting simulation model, shown in Figure 6.1, provides a means to investigate the influence of visibility on agent movement.

The process involved the following steps:

1. **Environment Modeling:** In order to create a realistic simulation environment, map data of Þingvellir national park was acquired using OpenMapBox for Unity. This data provided the graphic environment that hosted the agent simulation and served as the basis for the visibility analysis. The map data represented the various walkable zones of the park, including the trails. A navigation mesh was then created from the walking paths to facilitate agent navigation within the environment.

2. **Visibility Analysis:** With the navigation mesh in place, the next step was to analyze the spatial visibility within the environment. To achieve this, supporting software was developed for managing heatmaps in relation to the environment, which included subdividing it into an overlayed grid with corresponding size, offset, and other parameters. Visibility analysis from the

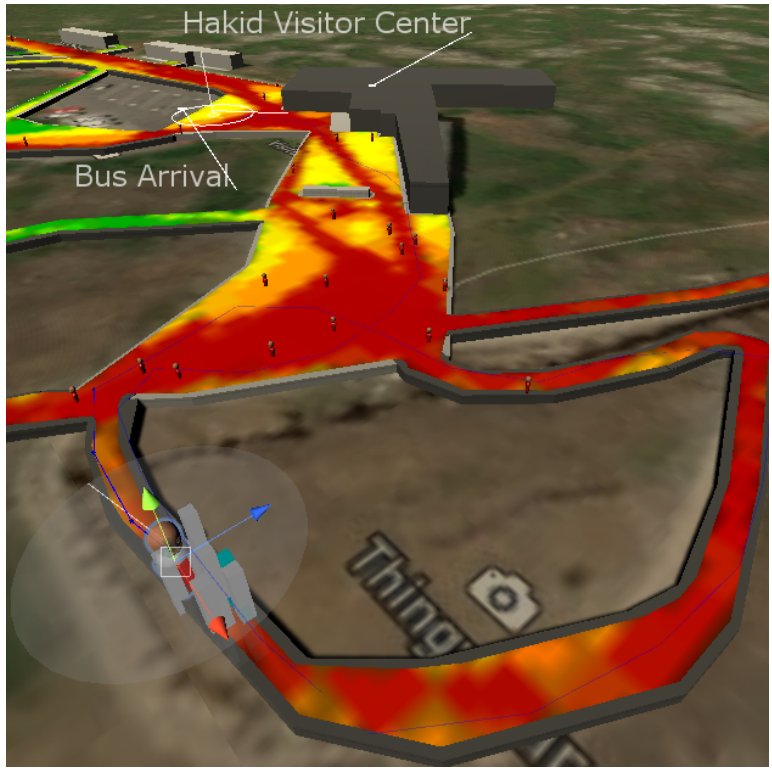

Figure 6.1: Unity native simulation for the first case study, featuring the Þingvellir environment generated with MapBox. A visibility heatmap, with brighter colors indicating higher visibility areas, is overlaid on the NavMesh, guiding agent behavior through a C# script that drives navigation based on a heuristic.

space syntax theory [99] was then applied to the navigation mesh, computing a heatmap that encoded the space based on its visibility. Pixels corresponding to less visible locations were assigned a darker color, while visible spots were brighter. This was achieved using raycasts that were iteratively computed from within the navigation mesh, looking for unobstructed lines of sight. The analysis essentially calculated how many grid cells were directly visible from each other grid cell. This information was then used to model agent behavior in response to the visibility of their surroundings.

3. **Agent Behavior Authoring:** The agent behavior was modeled using a Unity C# script designed to control the movement of agents based on the visibility heatmap. The script periodically sampled the surrounding environment by generating random points within the agent's perceptual range, testing their line of sight, and determining if they are on the navigation mesh. It then scored these points based on factors such as visibility, angle, and novelty, selecting the most appealing point as the agent's destination. The agents were attracted to more visible locations, and their movement decisions were influenced by the spatial visibility of the environment.

4. **Simulation Scenario and Execution:** The final step involved setting up the simulation scenario and executing the simulation. Agent spawners were implemented and controlled using spawning curves, which determined the number of agents spawned at specific times during the simulation. Two main spawners were placed strategically in the environment to represent tourists arriving by bus (tours) and those arriving by car (private). These different arrival methods influenced the agent behaviors, with tour agents having less time and needing to return to their spawn location before being despawned, while private agents could linger for longer periods. The agents were visually represented by simplistic human models, consisting of a floating torso and head.

6.1.2 Field Study

By collecting information on tourist behaviors and their experiences of overcrowding and isolation at Þingvellir national park (shown in Figure 6.2), the main goal of the field study was to better understand how these factors influenced their overall experience at the park. For more information refer to the published paper [106] written in collaboration with other members of the Center for Analysis and Design of Intelligent Agents (CADIA). The field study also served as a valuable source of real-world data to assess the simulation model developed for the park.

The study was conducted in the following stages:

1. **Device Building:** A physical device, named Rök, was designed, which consisted of an Arduino board with a GPS tracker, a memory stick for data storage, and two buttons to be pressed when the user felt overcrowded or isolated. The components were enclosed in a 3D printed exterior.

2. **Data Gathering:** Data collection in Þingvellir was carried out based on a strict onboarding protocol. This included approaching potential participants, providing an overview of the study, obtaining consent, having them complete an anonymous demographic questionnaire, explaining the device, and collecting the device at the end of their visit, followed by an exit interview. The study was conducted over nine days between July and August 2019, covering the hours from 10:00 to 16:30, and managed to onboard 66 participants.

3. **Analysis:** The collected data was processed, analyzed, and visualized. Outlier detection and removal were performed during the processing step. The primary analysis, conducted using Python Pandas' linear regression, revealed a strong correlation between overcrowding and the number of visitors at the park, suggesting a relationship between the two factors. Visualization in Google Earth enabled the creation of informative graphics about the most overcrowded park areas and the walking trajectories of the participants shown in Figure 6.3.

Figure 6.2: A photo of Þingvellir National Park taken during the field study. On the left, it is possible to see Almannagjá, the gorge at the edge of the North American tectonic plate.

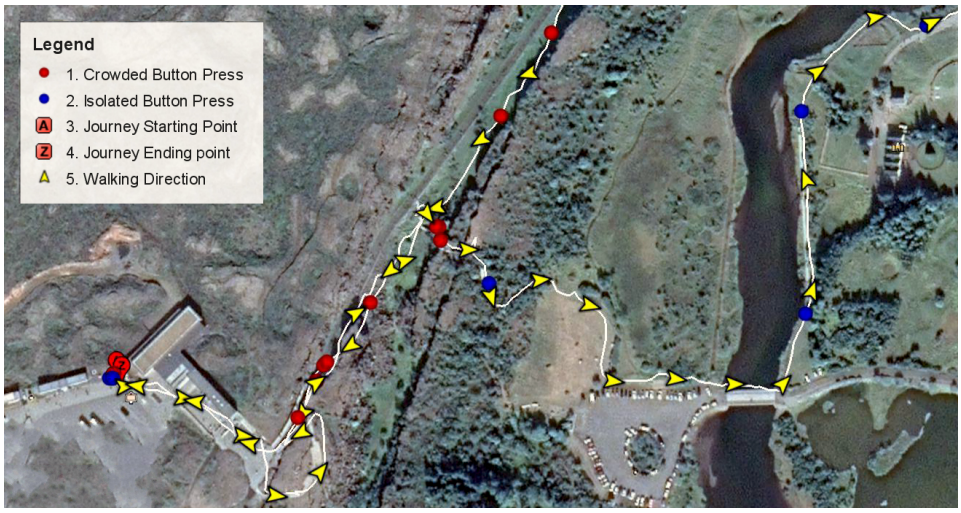

Figure 6.3: Example of data gathered by the device from a single participant. The data was processed and visualized in Google Earth, showing the participant's walking trajectory and the number of times they felt overcrowded or isolated.

6.1.3 Agora Simulation

The Þingvellir simulation was recreated using the Agora framework once it became available to assess its ability to support the development of practical crowd simulations. The implementation process can be divided into four main components:

1. **Environment Modeling**: The environment was set up similarly to the Unity native simulation using Mapbox for geometry. However, an additional step was required to convert the Unity navmesh into Menge's custom format. A script was implemented to access the underlying triangulated geometry of the mesh and export it as an OBJ file. The file was then converted using a Python utility. The mesh required conditioning to fix issues with the underlying geometry. This process was performed in Blender and involved filling holes, beautifying faces, and merging vertices by distance.

2. **Visibility Analysis**: Visibility analysis was conducted in a manner similar to the Unity native simulation, using iterative raycasts within the navigation mesh to identify unobstructed lines of sight.

3. **Behavior Authoring**: Agent behavior was authored using the xNode tool by creating a behavioral finite state machine that leveraged the generated heatmap to influence navigation, shown in Figure 6.5. The BFSM consisted of two states: a "visibility" state, in which agents perceive their surroundings and select a goal based on the visibility heatmap, and a "random goal" state, in which agents navigate towards a random point of interest within the Þingvellir area. These states were connected by transitions that occurred at regular intervals or whenever agents reached their goals, resulting in a wandering

behavior that accounted for both visibility and points of interest. The visibility state used a heatmap goal selector with two inputs: the Þingvellir visibility heatmap and the perception field, which limited the agents' perception field and allowed them to partially observe their surroundings.

4. **Simulation Scenario and Execution**: The simulation scenario was created using the scene specification Unity custom editor, which enabled the specification of agent spawners similar to those in the native simulation. Agent profiles with varying parameters could be visually specified.

The resulting working simulation, shown in Figure 6.4, demonstrated the capabilities of the Agora framework in modeling and simulating crowd behavior in the Þingvellir National Park practical scenario. Moreover, certain aspects of the implementation process, particularly behavior authoring, were greatly improved. For a more comprehensive analysis, please refer to the discussion in Section 6.1.5.

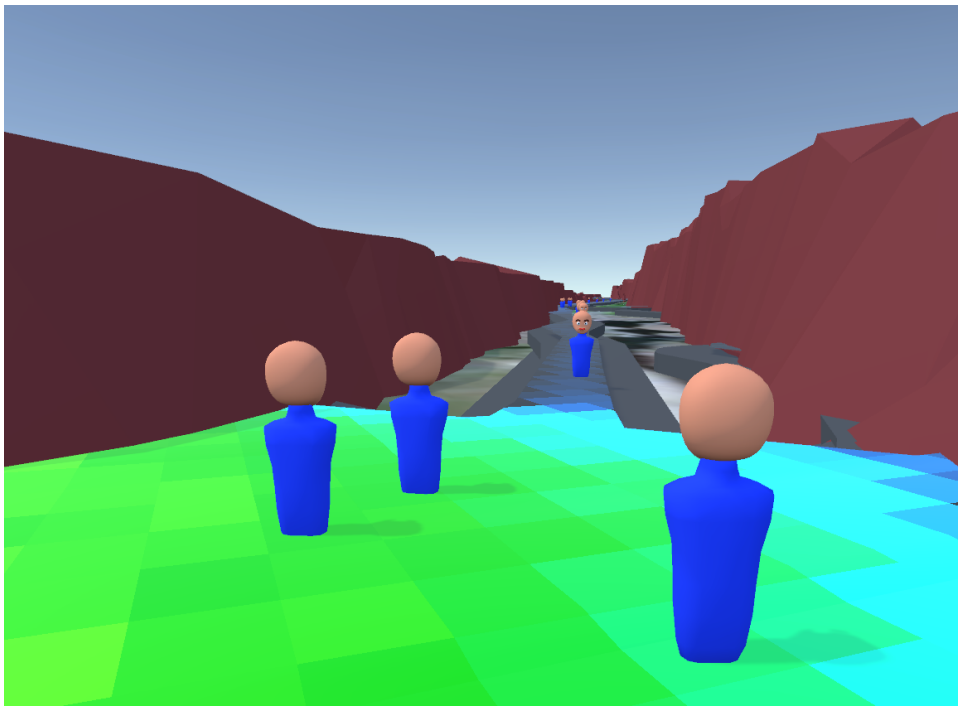

Figure 6.4: Agora simulation of the first case study. The agents' behavior is managed by Menge using a BFSM, which leverages the visibility heatmap to influence navigation. The agents steer towards Almannagjá, the gorge within Þingvellir National Park which marks the edge of the North American tectonic plate.

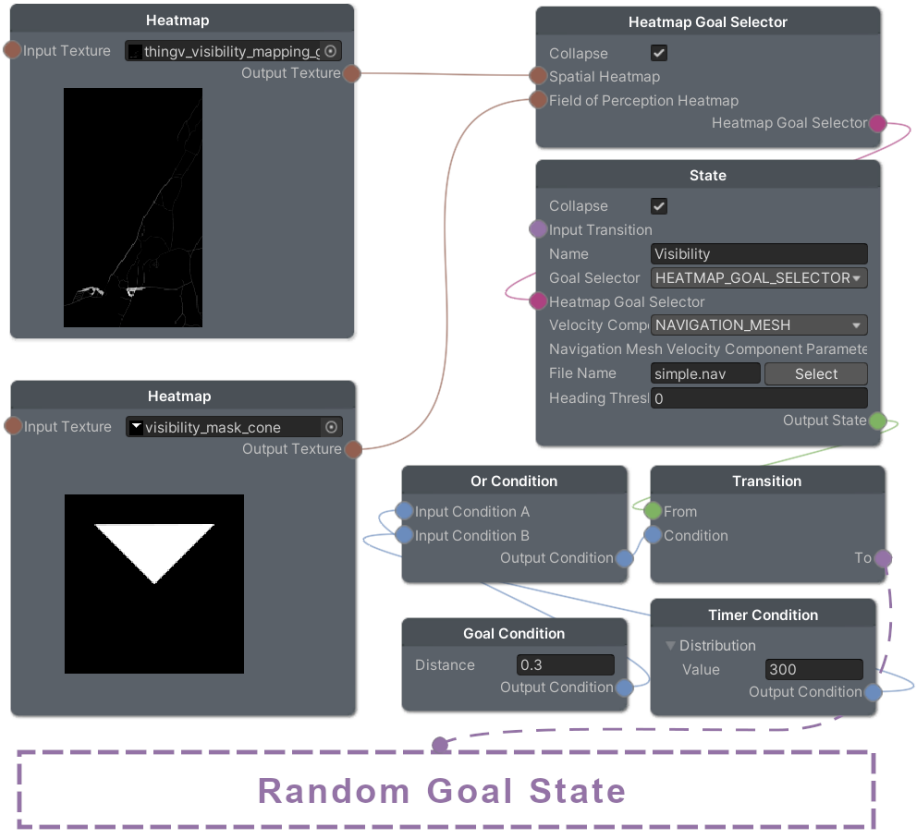

Figure 6.5: Simplified view of the BFSM used in the Agora framework implementation for the Þingvellir scenario, showcasing the state where agents perceive their surroundings and select goals based on the visibility heatmap. The BFSM is alternating between visibility and random goal selection.

6.1.4 Output Evaluation

In order to evaluate the accuracy of the crowd simulation models developed using both the Unity native implementation and the Agora framework, the heatmaps generated by the simulations were compared with the real-world data heatmap using the Agora evaluation function. This comparison aimed to assess the ability of each simulation model to replicate the patterns observed in the real-world data, and to test the evaluation functionality of the Agora framework.

Three heatmap comparisons were conducted, each utilizing metrics mentioned in Section 5.6 (Mean Squared Error, Structural Similarity Index, Earth Mover's Distance, and Peak Signal-to-Noise Ratio):

1. **Real World Data Heatmap and Agora Heatmap:** assess the accuracy of the simulation model created using the Agora framework in a real-world scenario.

2. **Real World Data Heatmap and Native Simulation Heatmap:** evaluate the accuracy of the traditionally developed simulation model, serving as a baseline to determine the success of the Agora framework simulation.

3. **Agora Heatmap and Native Simulation Heatmap:** This comparison was carried out to identify the differences and potential improvements offered by the Agora framework, highlighting a successful recreation of the original simulation.

Initially, the comparison results reported suspiciously high accuracy, mainly because most of the resulting heatmaps were predominantly black. The non-traversable space is much larger than the traversable space, and that portion can never change between simulation and the real world. Therefore, it was essential to focus the comparison on the relevant part of the environment. To achieve this, the evaluations were repeated with a mask as supported by the Agora evaluation nodes. The mask was a black and white image with white pixels covering only the traversable parts of the environment. These traversable areas were extracted based on the navigation mesh, which accounted for roads and trails.

The results, shown in Table 6.1, show that the Agora framework and the native simulation exhibit comparable performance in replicating the real-world data across all metrics. Higher values are better for the Mean Squared Error (MSE) and the Structural Similarity Index (SSIM), both being inverted and normalized. In these metrics, Agora performs slightly better, especially with masked data in MSE where it's nearly 15% better. The values around 50% for MSE and near 80% for SSIM suggest that both simulations are reproducing more than half of the real-world data in a very similar manner. The Peak Signal-to-Noise Ratio (PSNR) gives higher values for better similarity, and we see all values for both methods above 24 dB, indicating a good level of fidelity in the simulations. The Earth Mover's Distance (EMD) is a measure of the distance needed to change one distribution to another. Lower values here are better, and we see both simulations presenting low EMD values, particularly with Agora for masked data. In conclusion, while Agora presents slight advantages in MSE and EMD, both simulations exhibit solid performance

	Metric	RW vs. NH	RW vs. AH	AH vs. NH
No Mask	MSE (%)	98.12	98.76	98.3
	PSNR (dB)	24.71	27.93	24.97
	SSIM (%)	99.01	99.11	99.16
	EMD (pixels)	5.7	10.03	4.47
Masked	MSE (%)	49.85	65.43	54.15
	PSNR (dB)	27.06	26.68	27.45
	SSIM (%)	78.85	79.68	80.39
	EMD (pixels)	11.26	9.8	5.79

Table 6.1: Pairwise heatmap evaluation in the Þingvellir scenario, comparing real-world data (RW), Unity native simulation (NH), and Agora framework implementation (AH) using four metrics for both unmasked and masked heatmaps. MSE and SSIM are inverted and normalized, with higher values indicating better similarity. PSNR is measured in decibels, with higher values representing better similarity, while EMD is a distance metric, with lower values indicating better similarity between the compared heatmaps.

in replicating real-world data. This implies the Agora framework is a good tool for crowd simulations, providing similar quality to a native Unity simulation but potentially more efficiently.

6.1.5 Summary of Results and Discussion

This case study involved conducting a field study in Þingvellir National Park to gather data on tourist movement patterns, which was then used to develop a Unity native simulation and an Agora framework-based simulation. The performance and quality of the two simulations were found to be comparable, with the Agora framework capable of managing 3000 agents at the peak of the scenario. A comparison of the two simulations allowed for an assessment of their capabilities and limitations.

The Agora framework's capacity to accurately emulate the native simulation, while also introducing a layer of convenience and efficiency, was a highlight of the study. One of the standout features of this framework is its capacity to visually author behavioral finite state machines that guide agent behavior. This provided a more straightforward method for modeling agent behavior, utilizing a graphical user interface to facilitate the integration of the visibility heatmap into the goal selection component of a state node. The framework's usability was particularly noticeable in its capacity to enable the iterative and interactive authoring of different simulation elements, without the necessity to alter the underlying programming code. The native implementation, in contrast, depended on scripting to define the behavioral model. Every required change in this model implied a corresponding adjustment to the code.

The BFSM in Agora provides a clear separation between the foundational

programming and the specification of behavior and scenario, which was entirely authored through the graphical user interfaces. This separation substantially reduced the complexity associated with altering behavioral models and scenarios and resulted in a more efficient workflow. The convenience extended to being able to dynamically integrate a heatmap, like the visibility heatmap, simply by dragging it into the BFSM, quickly affecting the goal selection process and nudging agents towards more visible areas of the environment.

However, as expected, there is a tradeoff between convenience and expressive power. The native simulation, which relies on a general-purpose programming language, provides greater flexibility, allowing for a virtually limitless range of possible operations. This flexibility enables the implementation of more complex and customized solutions, albeit at the cost of increased complexity.

The native implementation guided agent movement using a heuristic accounting for multiple factors, including memory, which influenced goal novelty. This aspect is not yet supported by the Agora framework, which led to the introduction of a new state where agents aim for random points of interest in the park, encouraging exploration. While the lack of memory is a limitation, the Agora framework allowed for the decomposition of the complex heuristic into its fundamental components, enabling the analysis of each component's effect on agent behavior.

Furthermore, the Agora implementation was found to be less dependent on the underlying systems, offering greater modularity. It is possible to switch components, such as the navmesh for a different global pathfinding algorithm, without affecting the overall functionality. This modularity is advantageous as it facilitates easier experimentation with different components, fostering innovation and encouraging more in-depth analysis of crowd simulation models.

This case study made practical use of the output evaluation functionality of the Agora framework, highlighting its effectiveness and convenience in assessing the quality of the simulation model. The user-friendly GUI facilitated a seamless comparison of heatmaps generated from the simulation with those encoded from real-world data. The image similarity metrics provided an efficient and convenient way to gauge the accuracy of the simulation model at a glance, offering valuable insights into the model's performance. This evaluation functionality streamlines the process of refining simulation models, making it easier for researchers and practitioners to improve their simulations and better understand the underlying phenomena.

Alongside the objective evaluation provided by image similarity metrics, a more subjective assessment of the crowd simulations developed with Unity and the Agora framework also supports their validity in reproducing real-world trajectories.

In the visual examination, both simulations successfully replicated certain emergent behaviors observed in the real-world scenario. Notably, the simulated agents exhibited a strong inclination to navigate towards the viewpoint near the visitor center, a location offering a comprehensive view of the entire park. This movement pattern mirrors observed behavior in the real-world data, indicating that the simulations effectively capture key spatial attractors within the environment.

Additionally, agents within the simulations displayed a propensity to explore their surroundings. This behavior aligns with the known tendencies of real-world

park visitors and could be seen as the agents moved through various points of interest within the park.

In conclusion, the alignment of simulated agent behavior with real-world patterns provides further support for the accuracy and effectiveness of both the native Unity simulation and the Agora framework in replicating real-world crowd dynamics. Thus, in addition to the quantitatively robust performance on image similarity metrics, the simulations demonstrate qualitative success in portraying realistic crowd behaviors within the park setting.

6.2 Second Case Study: Urban Environment

The second case study focuses on examining the capabilities of the Agora framework in relation to combining agent behaviors through the integration of guiding heatmaps. The primary objective of this case study is to evaluate the effectiveness of this behavior combination method within the context of the Agora framework, specifically by combining the thermal and density comfort theories in a simulation of pedestrians in an urban environment.

A secondary objective is to assess the perceptual impact of the resulting behavior in a visually immersive simulation. This simulation diverges from the first case study by implementing high-quality meshes for both the environment and the agents, creating a more realistic and engaging visual experience. This heightened level of realism provides an enhanced opportunity to measure how the combined behaviors exhibited by the agents are interpreted and perceived by observers.

In this case study, rather than developing a new, arbitrary set of theories influencing human behavior that may yield subjective results, the decision was made to base the investigation on existing literature. Consequently, the case study re-implements an existing thermal and density comfort behavior model from Chen et al. [23], showcasing the feasibility of combining the underlying theories within the Agora framework, employing the heatmap paradigm for an urban pedestrian simulation.

The relevance of this case study to the Agora framework is twofold. First, it serves as an opportunity to validate the framework's ability to accurately and effectively combine agent behaviors using heatmaps, as outlined in the theoretical framework and architecture/implementation chapters. Second, it emphasizes the practical applicability of the Agora framework, as it demonstrates its capacity to incorporate established behavior theories, such as thermal and density comfort, from the literature. By showcasing the framework's potential to combine and implement these theories in an urban pedestrian simulation, this case study aims to further establish the Agora framework as a valuable and versatile tool in the field of crowd simulation.

6.2.1 Urban Environment

The urban environment of the second case study is built upon the work of Hafsteinsson [61], which examined the influence of compact urban area design on people's psychological well-being as part of the Cities that Sustain Us (CiSuUs)

research project [87]. The project was grounded in Restorative Environmental Design (RED), which explored the relationship between architectural design and psychological restorative capabilities. Hafsteinsson focused on the automatic procedural generation of high-detail residential street scapes, resulting in a framework that enabled the creation of realistic cities for users to explore using Virtual Reality (VR) equipment.

The framework utilized pixel maps (Figure 6.6 top) to describe the environment, with different layers for features such as roads, buildings, decorations, green areas, building colors, and other details. The generator filled the environment using various tiles (Figure 6.6 bottom), including building tiles, road tiles, park tiles, and dirtbed tiles. An example of an urban environment generated by the framework is shown in Figure 6.7.

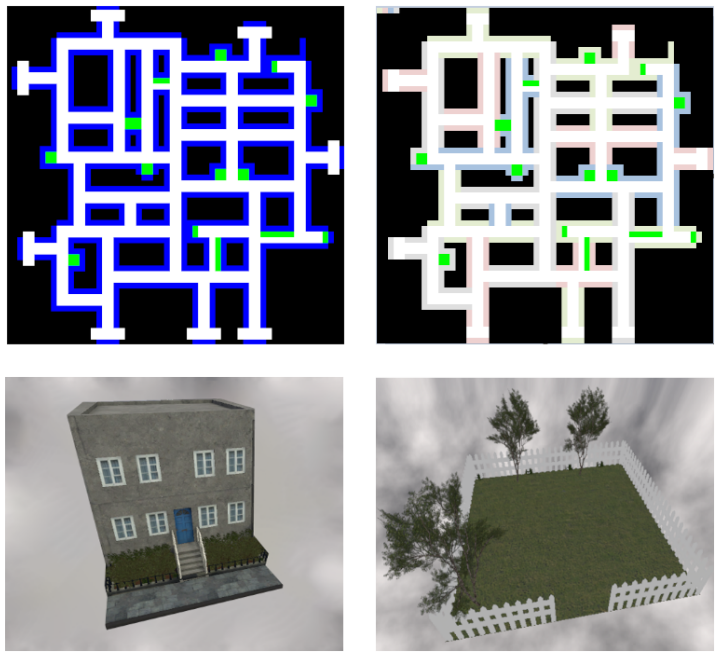

Figure 6.6: (top) Pixel maps describing features of the environment such as buildings and parks. (bottom) Tiles employed to fill the environment. The image was adapted from [61].

The framework also included an interface that supported questionnaires, which could be used while a person experienced the environment in virtual reality. This, in conjunction with physiological measurements, facilitated the evaluation of the restorative capabilities of specific urban settings. By modifying pixel maps, researchers could efficiently generate and adjust environments to test the impact of different aspects on users. Further information about this research can be found in [71].

Figure 6.7: An overview of a possible urban environment generated with Hafsteinsson's framework [61]. The image is from the same paper.

6.2.2 Behavior Theories

In this case study, the simulation focuses on people's reactions to varying levels of **thermal comfort** and **crowd density**, based on the research conducted by Chen et al. [23]. The authors argue that human behavior is influenced by both their thermal comfort and the density of others in their environment. As a result, people may take appropriate actions to increase their comfort, such as relocating to a less crowded area or adjusting their clothing.

6.2.3 Native Implementation

To model the thermal and density comfort, the authors developed a framework, shown in Figure 6.8, using an *agent map* consisting of a uniform grid corresponding to the 3D simulation environment. Each cell in the grid contains information about the agents located in it. These agents, acting as moving heat sources, contribute to a heat transfer model that calculates air temperature alongside other heat sources or sinks. This *temperature map* and other environmental factors, such as humidity and air speed, are used to determine each agent's thermal comfort level. The agent map is also utilized to compute a density comfort level based on agent locations, which is then combined with the thermal comfort map to derive the *overall comfort* of each agent.

When agents experience discomfort, they react by either changing their clothing (affecting individual thermal comfort) or changing their location, based on local thermal context and nearby agents. A crowd simulator is employed to move the agents when they need to relocate, and their new positions serve as inputs to update

the heat transfer model.

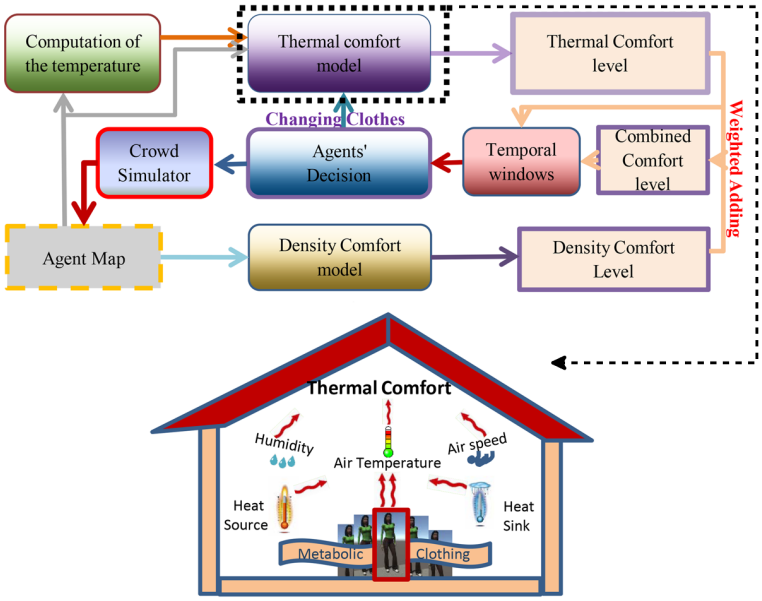

Figure 6.8: The overview of the thermal comfort framework developed by Chen et al. [23]. The image was adapted from the same paper.

The Thermal Comfort Model

The thermal comfort model employed in the simulation uses the Predicted Percentage of Dissatisfied (PPD) index, shown in Equation 6.1. This measure is defined by the ISO standard [1] to quantify the proportion of people likely to feel thermally dissatisfied within a given environment:

$$PPD_t = 100 - 95 \cdot e^{-0.03353 \cdot PMV^4 - 0.2179 \cdot PMV^2}. \tag{6.1}$$

This index is derived from the Predicted Mean Vote (PMV) model, which estimates the average thermal sensation of a large group of people exposed to the same environmental conditions [43]. The PMV model considers several factors and is computed as shown in Equation 6.2:

$$PMV = (0.303 \cdot e^{-0.036 \cdot M} + 0.028) \cdot \left\{ \begin{array}{l} (M - W) - 3.05 \times 10^{-3} \cdot [5733 - 6.99 \cdot (M - W) - p_a] \\ -0.42 \cdot [(M - W) - 58.15] - 1.7 \times 10^{-5} \cdot M \cdot (5867 - p_a) \\ -0.0014 \cdot M \cdot (34 - t_a) - 3.96 \times 10^{-8} \cdot f_{cl} \\ \cdot [(t_{cl} + 273)^4 - (t_r + 273)^4] - f_{cl} \cdot h_c \cdot (t_{cl} - t_a) \end{array} \right\} \tag{6.2}$$

Where:

M: metabolic rate (W/m^2)
W: external work (W/m^2)
$\mathbf{t_a}$: air temperature (°C)
$\mathbf{t_r}$: mean radiant temperature (°C)
$\mathbf{f_{cl}}$: clothing area factor (dimensionless)
$\mathbf{t_{cl}}$: clothing surface temperature (°C)
$\mathbf{h_c}$: convective heat transfer coefficient (W/m$^2 \cdot$ °C)
$\mathbf{p_a}$: partial water vapor pressure (Pa)

In the thermal model of the original paper [23], the metabolic rate and the external work depend on the activity being performed and so are user-provided parameters, while the remaining variables are computed as explained below. A complete list of the parameters used in the model is provided in Table 6.2.

Computing the Thermal Comfort Model

($\mathbf{t_a}$) Air Temperature. Is determined using a simplified heat transfer model, which approximates the complex thermal interactions in the environment by considering factors such as heat generation, heat consumption, and thermal conductivity:

$$C\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left[k(x,y)\frac{\partial T}{\partial x}\right] + \frac{\partial}{\partial y}\left[k(x,y)\frac{\partial T}{\partial y}\right] + Q(x,y,t) \tag{6.3}$$

Where:

T: Air temperature (℃)
C: Air volumetric heat capacity ($J/(m^3 \cdot$ ℃))
k: Thermal conductivity ($W/(m \cdot$ ℃))
Q: Rate of heat generation/consumption (W/m^3)

Equation 6.3 models the air temperature by considering the heat generated by various sources and the heat consumed by sinks, along with the thermal conductivity of the surrounding media. Each agent is considered a source with constant heat generation Q_0, contributing to the term $Q(x, y, t)$ in the equation. Moreover, a spatially varying version of thermal conductivity is adopted to approximate the

effects of walls and other thermally insulating objects within the environment, requiring smaller values for $k(x, y)$.

The objective is to determine the temperature value for each location within the discretized grid of the environment, computed for consecutive time steps. To achieve this, a simple explicit Euler solution method (with boundary conditions) is employed, which relies on forward finite differences in time and central differences in spatial coordinates for approximating the solution to the equation.

($\mathbf{t_r}$) Mean Radiant Temperature. Is a measure of the combined effect of radiant heat transfer between an individual and the surrounding environment. It is computed by starting from a base value tr_0, accounting for the environmental factors, and then considering agents that are in close proximity (within intimate or personal distances), applying the formula in Equation 6.4 to capture the influence of these nearby agents on the radiant heat transfer.

$$t_r = t_{r0} + \gamma \sum_i \frac{\epsilon + |\cos \theta_i|}{d_i^2} \tag{6.4}$$

Where:

$\mathbf{t_{r0}}$: Environmental mean radiant temperature (℃)
$\mathbf{d_i}$: distance between agents
$\theta_\mathbf{i}$: Angle to the i-th intimate distance agent
ϵ: Minimum t_r gain for a given distance when $\theta = 90°$
γ: Weight for controlling the temperature increase

($\mathbf{f_{cl}}$) Clothing Area Factor. Adjusts the effective surface area of the clothing based on its insulation value (I_{cl}), as calculated using Equation 6.5.

$$f_{cl} = \begin{cases} 1.00 + 1.290 \cdot I_{cl}, & \text{if } I_{cl} \leq 0.078 \\ 1.05 + 0.645 \cdot I_{cl}, & \text{if } I_{cl} > 0.078 \end{cases} \tag{6.5}$$

($\mathbf{t_{cl}}$) Clothing Surface Temperature. The clothing surface temperature represents the temperature at the outer surface of the clothing, which is computed using an equation relating the environment with the clothing, as shown in Equation 6.6.

$$\begin{aligned} t_{cl} &= 35.7 - 0.028 \cdot (M - W) - I_{cl} \\ &\cdot \left\{ \begin{array}{l} 3.96 \times 10^{-8} \cdot f_{cl} \cdot [(t_{cl} + 273)^4 - (t_r + 273)^4] \\ + f_{cl} \cdot h_c \cdot (t_{cl} - t_a) \end{array} \right\} \end{aligned} \tag{6.6}$$

Where:

$\mathbf{I_{cl}}$: Clothing insulation (m^2K/W) (1 clo = 0.155 $\mathrm{m}^2 \cdot K/W$)

(h_c) Convective Heat Transfer Coefficient. Describes the heat transfer between the clothing surface and the surrounding air, which is influenced by the difference in temperature and the air velocity (v_a) and is determined by Equation 6.7.

$$h_c = \begin{cases} 2.38 \cdot |t_{cl} - t_a|^{0.25}, & \text{if } h_c > 12.1\sqrt{v_a} \\ 12.1\sqrt{v_a}, & \text{if } h_c \leq 12.1\sqrt{v_a} \end{cases} \tag{6.7}$$

(p_a) Partial Water Vapor Pressure. Is a measure of the concentration of water vapor in the air, determined by taking into account the air temperature and relative humidity using Equation 6.8.

$$p_a = 10^{h_a} \cdot e^{\frac{16.6536 - 4030.183}{t_a + 235}} \tag{6.8}$$

Where:

h_a: Air humidity (*percentage*)

By combining all these values, and using them in Equation in 6.1, it is possible to compute the PPD, which represents the thermal comfort level of the environment.

The Density Comfort Model

A simple model is used to estimate the local discomfort with respect to the local density of the crowd. Equation 6.9 computes the density discomfort by considering the number of agents in both intimate and personal spaces described by Hall et al. [63].

$$PPD_d = 100 \cdot \frac{n_i + \beta \cdot n_p}{M_i + \beta \cdot M_p} \tag{6.9}$$

Where:

n_i: Number of agents in the intimate space
n_p: Number of agents in the personal space
M_i: Maximum number of agents in the intimate space
M_p: Maximum number of agents in the personal space
β: Decay factor for the personal space

The Combined Comfort Model

To generate the combined PPD_c, a simple linear weighted average is used.

$$PPD_c = \alpha \cdot PPD_t + (1 - \alpha) \cdot PPD_d \tag{6.10}$$

Where:

α: Blend factor ($0 \leq \alpha \leq 1$)
PPD_t: Thermal PPD (from Equation 6.1)
PPD_d: Density PPD (from Equation 6.9)

6.2.4 Agora Implementation

The primary objective of the study was to evaluate the Agora framework with a focus on the quality of the combined behavior that results from integrating different factors using the heatmap blending technique. By replicating the implementation of the original paper, a solid foundation in the literature was established, ensuring that the approach was grounded in well-established theories rather than arbitrary concepts.

In this context, the heatmap paradigm was applied to the thermal and density comfort models, generating the corresponding heatmaps to represent the discomfort levels in both aspects. This allowed the xNode BFSM heatmap combiners in the Agora framework to effectively blend the thermal and density heatmaps, producing a combined behavior that was influenced by both comfort models.

As previously mentioned, the scene consisted of an urban environment created with the framework from [61], resembling the one depicted in Figure 6.7. To employ the heatmap approach, the environment was divided into a uniform grid of 500x500 world units, as illustrated in Figure 6.9. The size of each cell could be easily modified to strike a balance between the granularity of the heatmap and computational performance, considering that some heatmaps required calculations for each cell.

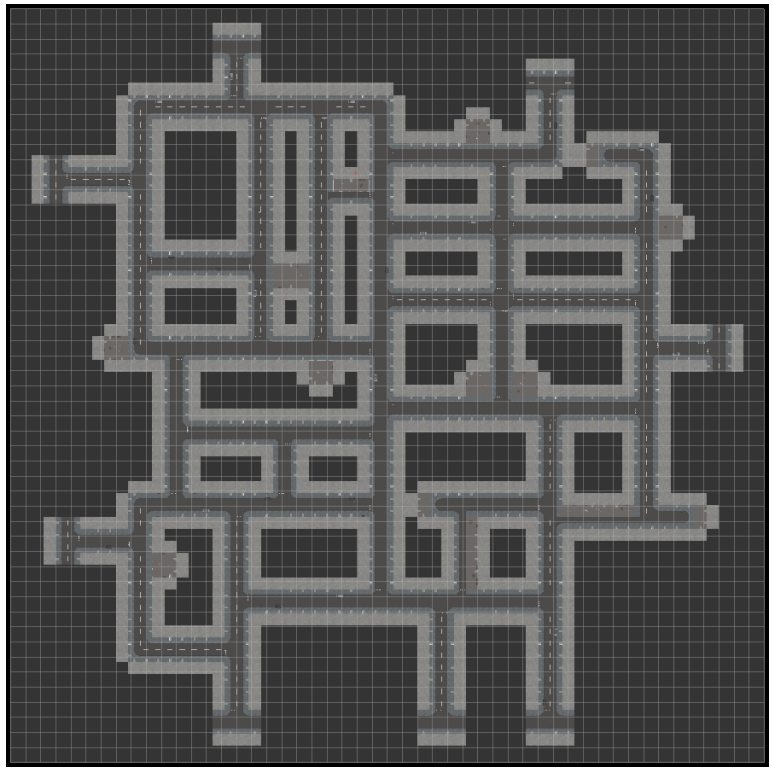

Figure 6.9: The top-down isometric view of the urban environment of the second case study, with the heatmap grid overlaid.

The following sections will give an overview of the heatmaps generated for the thermal and density comfort models, as well as the combined heatmap generated by the Agora framework.

How to interpret the heatmaps:

Size: Each heatmap has a size of 500 x 500 world units, with a cell size of 1 world unit. In Unity 1 world unit corresponds to 1 meter. The timestep (where applicable) is 0.1 seconds.

Color: To avoid confusion, every heatmap has been shaded with the same grayscale gradient linearly ranging from black to white, with black representing the lowest value and white representing the highest value.

Boundaries: Areas marked with diagonal lines represent "void" areas, which are effectively considered "outside the environment". Though these cells also have values, and in some cases even influence the surrounding locations, diagonal lines have been used (in this dissertation) for visualization purposes to provide a clearer graphical context of the "empty space" within the simulation. This distinction helps to emphasize the difference between the active areas of the environment and the empty spaces beyond its boundaries.

Parameters: The complete list of parameter values used for generating the heatmaps is provided in Table 6.2.

Thermal Comfort Heatmaps

This Section presents an overview of the various heatmaps involved in the simulation of thermal comfort. These heatmaps encompass thermal conductivity, thermal generation, air temperature, thermal comfort, and density comfort maps. The thermal conductivity and thermal generation maps are employed for calculating the air temperature map, which in turn serves as an input for determining thermal comfort.

Thermal Conductivity Heatmap. The thermal conductivity heatmap represents the distribution of thermal conductivity throughout various regions of the environment, primarily accounting for two materials: air and concrete. Lower values were assigned to areas occupied by buildings to simulate their insulating properties. In order to generate the heatmap, a downward box cast was performed for each cell in the grid. A box cast[1] is a Unity physics function that projects a box-shaped volume along a specified direction, checking for collisions with other objects in the environment. When a box cast intersected a collider associated with a building game object, the cell was assigned a lower value, indicating insulation. In cases where no intersection occurred, the cell was considered "open air" and had higher thermal conductivity.

[1] https://docs.unity3d.com/ScriptReference/Physics.BoxCast.html

	Name	Unit	Original	Agora
a	thermal conductivity	W/m·℃	N/A	0.001-0.024
b	heat rate	W/m^2	81-1630	155
	heat radius	meters	0.2	3
c	initial air temperature	℃	21-25	20
	volumetric heat capacity	J/m^3·℃	N/A	1206
d	metabolic rate	W/m^2	1.9-3.4	2.4
	clothing insulation	m^2·℃/W	0.4-1.2	1.2
	air velocity	m/s	0.1-0.5	0.2
	relative humidity	%	40	30
	env. mean radiant temperature	℃	25	23
	min. radiant temperature	℃	0.5	0.1
	temperature increase weight	units	0.1	0.5
e	personal space importance	units	0.2	0.5
	max. intimate	agents	6	4
	max. personal	agents	12	12

Table 6.2: Comparison of the models' parameter values between Agora and the original implementation [23]. The parameters are grouped by their pertinence to various parts of the comfort model: (a) thermal conductivity, (b) thermal generation (c) air temperature, (d) thermal comfort, and (e) density comfort.

Figure 6.10 shows the resulting heatmap. Since the difference between the values is very significant, the open-air areas are shaded in white, while the buildings are shaded in black.

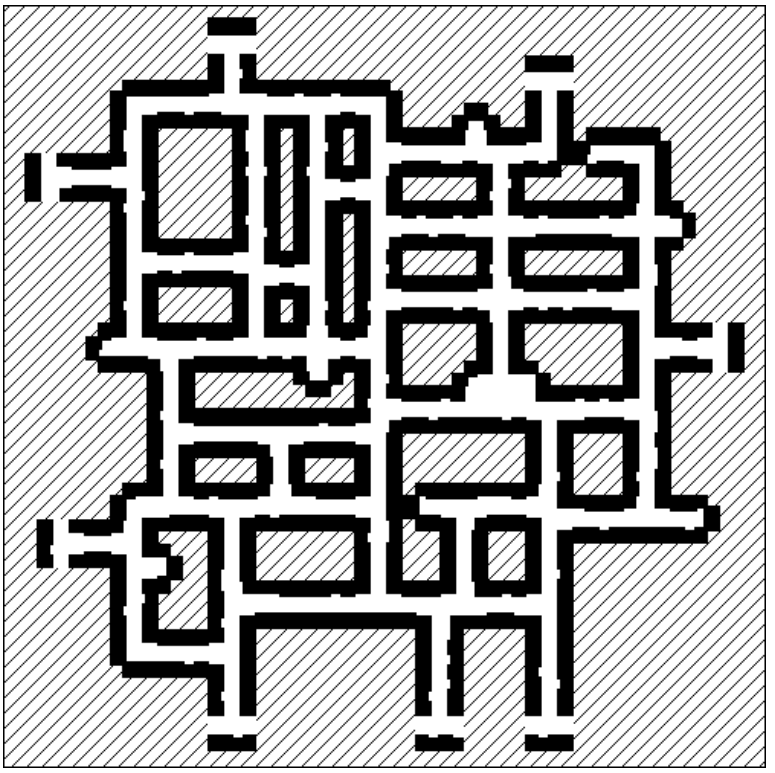

Figure 6.10: The thermal conductivity heatmap for the urban environment used in the second case study.

Thermal Generation Heatmap. The thermal generation heat map provides an instantaneous representation of the heat generation or consumption rate (W/m^3) caused by heat nodes (sources or sinks) in the environment. In the simulation, each game object that represents a heat node has an attached script containing data regarding the heat generation rate and the affected radius. If the heat generation rate is positive, the node acts as a heat source, whereas a negative rate indicates a heat sink. For every heat node $hn(x, y)$ in the simulation, where (x, y) denotes the position of the heat node, the thermal generation heatmap adds the heat generation rate value to all cells surrounding the (x, y) center within the specified radius. Figure 6.11 displays a sample thermal generation heatmap for the simulation, with the only heat nodes (sources) being the agents moving throughout the environment[2]. The color variation is due to overlapping heat source radii, which increase the local

[2]This is a simplification but additional sources can easily be added.

heat generation.

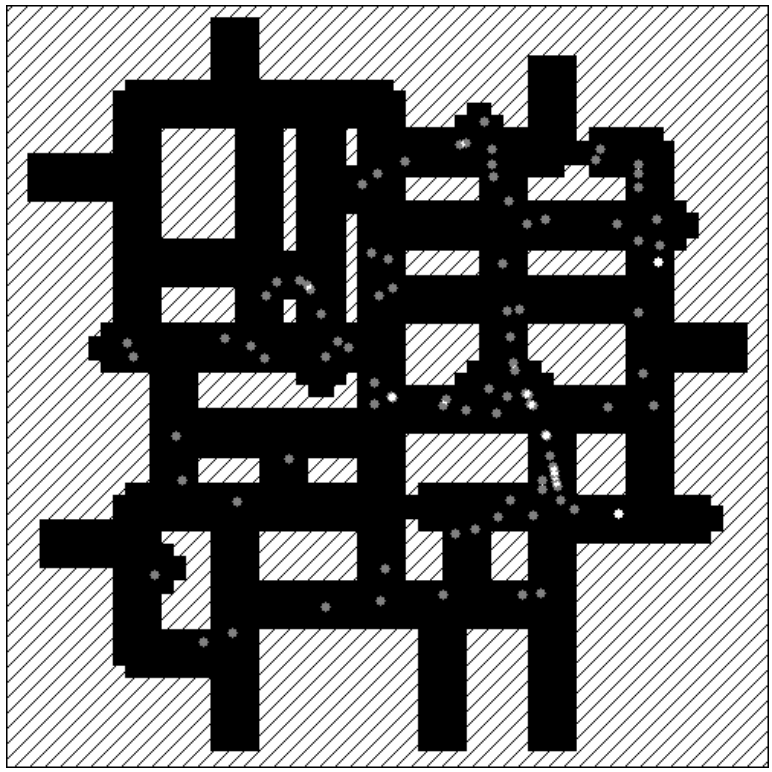

Figure 6.11: The thermal generation heatmap for the urban environment used in the second case study.

Air Temperature Heatmap. The air temperature map is simulated using a 2D diffusion model formalized in Equation 6.3. This equation is a partial differential equation (PDE) that models how the temperature changes over time based on the spatial distribution of temperature and thermal conductivity. PDEs involve derivatives with respect to multiple variables and necessitate specialized techniques for finding solutions.

As described in Section 6.2.3, a simple explicit Euler method is employed to numerically solve the PDE, approximating the solution by discretizing the spatial and temporal domains. Forward finite differences for time and central differences for spatial coordinates are employed in this method. In this context, forward finite differences refer to estimating the rate of change in time by considering only the future time step, while central differences involve estimating spatial derivatives based on the average of the nearby points in the discretized space.

For instance, the first-order partial derivative of temperature with respect to x ($\partial T/\partial x$) is approximated by computing the difference in temperature values at $x-1$

and $x+1$, and then dividing it by the distance between these points. This approach is applied to all spatial derivatives in the equation to estimate their values.

Once the spatial derivatives have been approximated, the temperature change at each grid point can be determined. The explicit Euler method then updates the temperature by adding the calculated temperature change to the current temperature value, multiplied by the time step. This process is iterated over time to simulate the diffusion of heat in the environment.

Algorithm 5 shows the central steps of the computation for applying the numerical solution to the PDE. For the sake of conciseness, only the necessary parts of the numerical solution are computed, while some intermediate computations are skipped. For example, the central difference approximation $\Delta T_x^{Right} \leftarrow \frac{T(x+2,y)-T(x,y)}{2h}$ uses a +2 index because it is required for approximating the outer derivative, the inner derivative approximation is skipped.

Apply boundary conditions:
for *each boundary point* **do**
  set boundary temperature
end
Loop through the interior grid points:
for *each interior point (x, y)* **do**
  Approximate temperature derivatives:
  $\Delta T_x^{Right} \leftarrow \frac{T(x+2,y)-T(x,y)}{2h}$
  (similar calculations for Left, Up, and Down)
  Calculate $k \cdot \Delta T$:
  $k\Delta T_x^{Right} \leftarrow k(x+1,y) \cdot \Delta T_x^{Right}$
  (similar calculations for Left, Up, and Down)
  Approximate diffusion derivatives for x and y:
  $\frac{\partial}{\partial x}(k\partial T/\partial x) \leftarrow \frac{k\Delta T_x^{Right}-k\Delta T_x^{Left}}{2h}$
  $\frac{\partial}{\partial y}(k\partial T/\partial y) \leftarrow \frac{k\Delta T_y^{Up}-k\Delta T_y^{Down}}{2h}$
  Update the temperature:
  $T_{new}(x,y) \leftarrow T(x,y) + \Delta t \cdot \frac{(\frac{\partial}{\partial x}(k\partial T/\partial x)+\frac{\partial}{\partial y}(k\partial T/\partial y)+Q)}{C_v}$
end

Algorithm 5: Numerical Solution of the 2D Heat Diffusion Equation. Variables: h - space delta; T - temperature; k - thermal conductivity; Q - heat generation; C_v - volumetric heat capacity; Δt - time step.

A heatmap of the air temperature, as produced by the algorithm, is displayed in Figure 6.12. Given that the temperature range is not fixed, a min-max normalization approach has been employed in the heatmap representation. Consequently, darker areas in the heatmap correspond to locations with the lowest temperature, while brighter areas indicate locations with the highest temperature. Throughout the simulation, the actual temperature range spanned from 20 to 30 ℃.

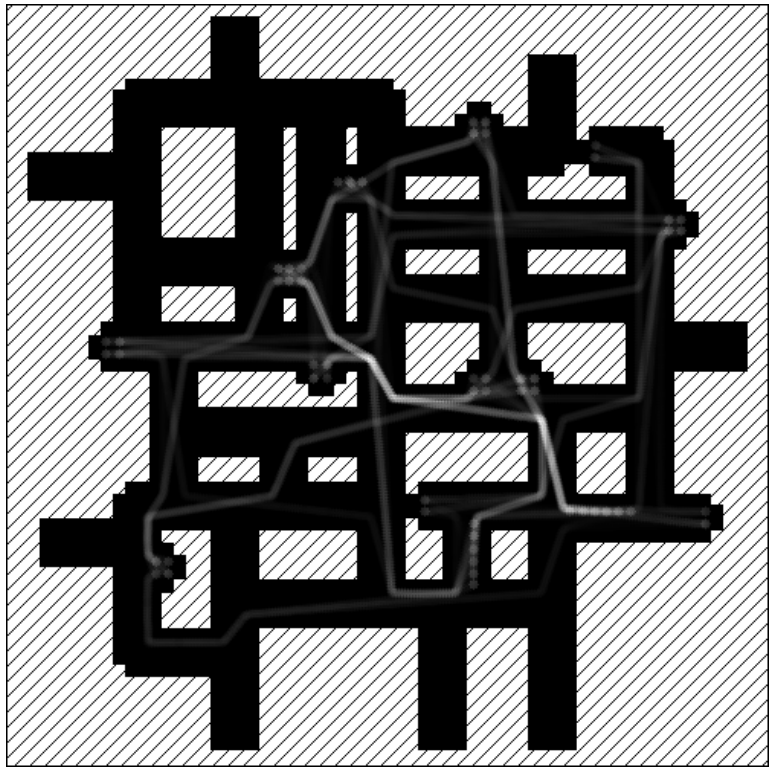

Figure 6.12: The air temperature heatmap for the urban environment used in the second case study.

Thermal Comfort Heatmap. The thermal generation heatmap and the air temperature heatmap are the two spatial maps required for computing the thermal comfort model. Once these maps are generated, Equation 6.2 can be applied to compute the predicted mean vote, which represents the thermal sensation and typically ranges from -3 (cold) to 3 (hot). To further generalize this index, the predicted percentage of dissatisfied agents in the thermal environment can be calculated using Equation 6.1, yielding values that range from 0 (all agents are satisfied) to 100 (all agents are dissatisfied). The PPD heatmap is illustrated in Figure 6.13, with darker areas denoting locations in the environment where most agents are thermally satisfied, and brighter areas indicating locations that are dissatisfactory due to being either too cold or too hot.

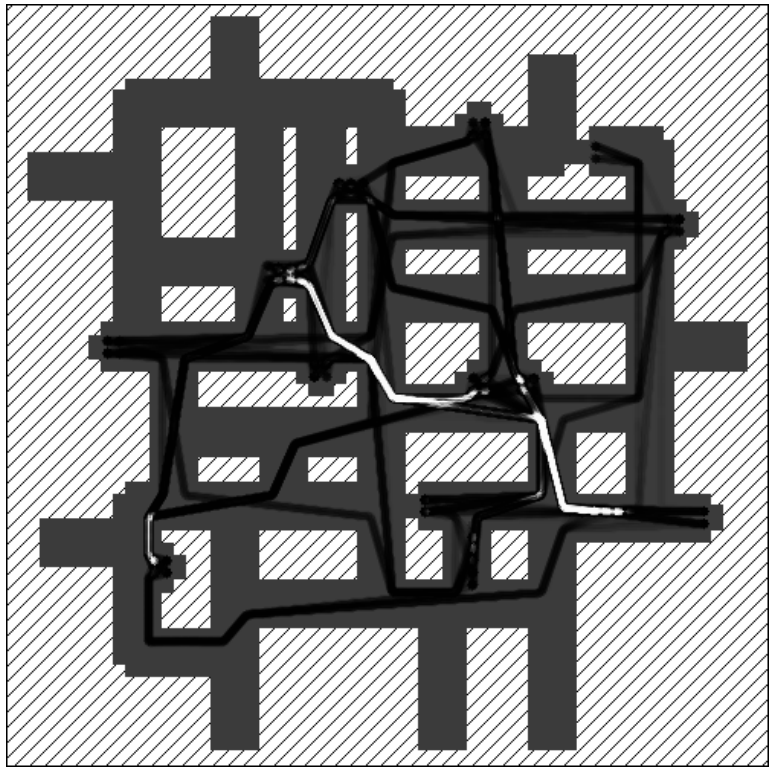

Figure 6.13: The thermal comfort (PPD) heatmap for the urban environment used in the second case study.

Density Comfort Heatmap

The density comfort heatmap evaluates the comfort levels related to the spatial distribution of agents within the environment. It is computed as described in Section 6.2.3 using Equation 6.9. Figure 6.14 displays a possible heatmap resulting from this computation. The most densely populated locations correspond to park

tiles designated as goals for the agents, leading to frequent gatherings and bright spots on the heatmap, which signify density discomfort. In contrast, the majority of the locations where agents walk exhibit less brightness, indicating lower density discomfort. Nevertheless, some congested roads are still clearly visible as areas with higher brightness, denoting locations where agents walk in close proximity to one another, thus experiencing greater density discomfort.

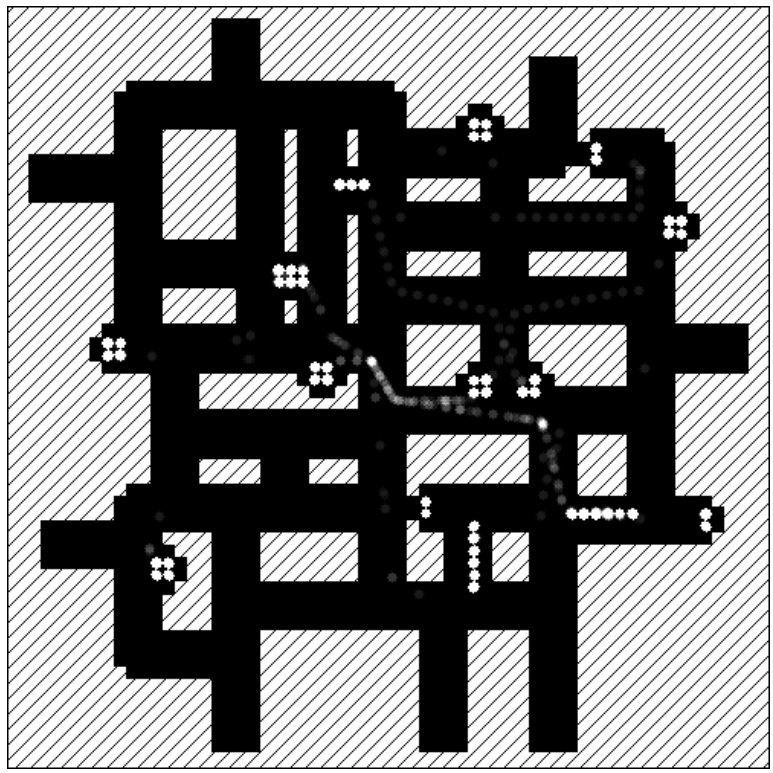

Figure 6.14: The density comfort heatmap for the urban environment used in the second case study.

Agent Behavior

In the original model, the agent behavior is influenced by the combined comfort level, which is a weighted average between the thermal comfort and the density comfort levels, as described by Equation 6.10. The α parameter adjusts the significance of the density stimulus compared to the thermal stimulus. In the Agora framework, this is achieved through color operations. Figure 6.15 illustrates an example of the simulation running in the Agora framework, where the agents are moving around the environment to balance their thermal and density comfort levels.

The density and thermal comfort heatmaps are calculated for every cell in the discretized environment at each time step, and subsequently, they are employed

Figure 6.15: Example of the second case study simulation running in the Agora framework. The comfort heatmap is overlayed on the ground.

as dynamic heatmap nodes in the behavioral finite state machine, as shown in Figure 6.16. This figure illustrates a portion of the BFSM responsible for regulating agent behavior. From left to right, two dynamic heatmap nodes encode the density comfort and thermal comfort levels of each location within the environment. As these heatmaps are single-channel textures, the values they contain are represented by the red channel, resulting in a red and black color scheme. Since PPD_t and PPD_d indicate the level of discomfort, the heatmap undergoes an inversion operation to convey the level of comfort. Following this, the two inverted PPD values are combined using a Heatmap Average Combiner node, which computes the weighted average of multiple textures based on a set of weights. This operation generates a combined comfort level PPD_c equivalent to the one determined in the original model. Lastly, the resulting heatmap is utilized by a heatmap goal selector, which applies a mask to perceive a portion of the surrounding environment and selects a goal based on the most attractive location within that area. This goal is then incorporated into the agent state, effectively influencing their behavior.

Agora Addition: Shadow-seeking behavior

To further show the adaptability of the Agora framework, a unique shadow-seeking behavior was incorporated, aligned with the existing thermal comfort model. This involved generating a new dynamic heatmap, informed by the changing position of the sun within the scene, effectively capturing the environmental shadows.

This was accomplished by utilizing an orthographic camera to render a top-down perspective of the urban environment. The camera was set to cull all but the shadow cast by objects within the environment, resulting in a heatmap. As shown in Figure

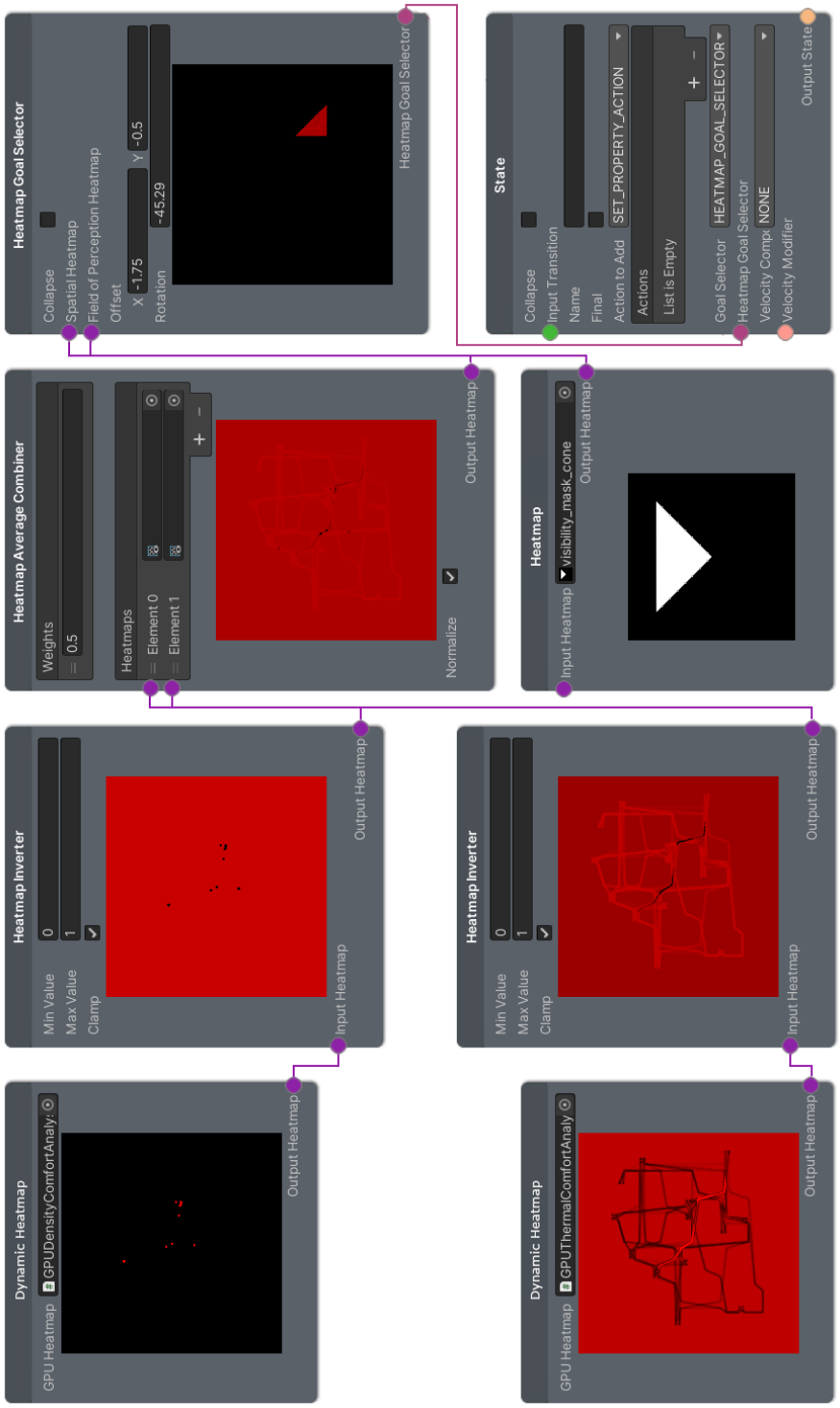

Figure 6.16: Portion of the BFSM modeling density and thermal comfort as dynamic (GPU) heatmaps influencing agent behavior. The stimuli are combined using a weighted average node.

6.17, this heatmap dynamically encodes regions of the environment into shaded areas (low values) and sun-exposed spaces (high values).

Subsequently, the heatmap was inverted and incorporated into the Behavioral Finite State Machine. This intuitive process underscores not only the simplicity of adding novel features to the simulation but also the convenience with which the original model can be expanded to incorporate a myriad of other stimuli. It highlights the Agora framework's capacity for flexible and intuitive behavior modeling, solidifying its position as a versatile tool for complex crowd simulation tasks.

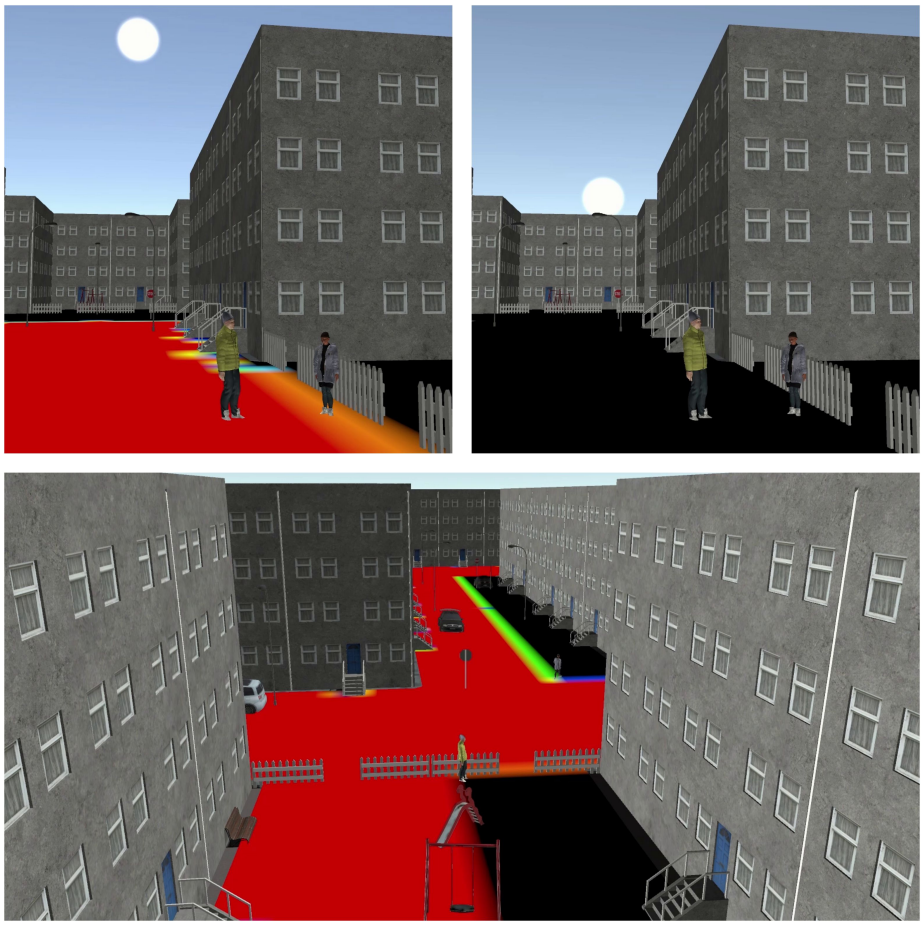

Figure 6.17: Shadow-seeking behavior conveniently added to the Agora simulation. The associated heatmap visually represents the impact of the sun's shifting position on the distribution of light and shadows within the environment. In response to these changes, the simulation's agents actively seek out shadowy areas to avoid exposure to the sun and high-temperature zones.

6.2.5 Summary of Results and Discussion

The second case study aimed to evaluate the performance of the heatmap-based behavioral combination mechanism within the Agora framework and to demonstrate the framework's ability to integrate established simulation models. To achieve this, the work of Chen et al. [23] was replicated and an implementation using the Agora framework was provided.

The original study suggested that individuals exhibit behavioral responses to thermal and density stimuli by attempting to maintain a balance of comfort between the two factors. The authors of the original study modeled this phenomenon by calculating the thermal comfort level (PPD_t) and density comfort level (PPD_d), and conducted agent simulations in two distinct scenarios. The resulting simulations exhibited plausible agent behaviors under varying density and thermal conditions. For instance, agents were observed to move into areas with higher occupant density to enhance thermal balance or to spontaneously distance themselves from a group of people in order to increase comfort levels.

In replicating the original work within the Agora framework, a significant shift in the paradigm was achieved. This new approach introduced the utilization of heatmaps for influencing agent behavior, offering a powerful and intuitive method for representing complex environmental stimuli. Moreover, the framework facilitated color-based combinations, which allowed for the seamless integration of multiple behaviors, providing a more comprehensive understanding of agent interactions with their environment.

The Agora framework also introduced convenient nodes that can be easily plugged together in the BFSM, streamlining the process of authoring behaviors. The suite of features provided by the Agora framework in support of the crowd simulation process significantly reduced the time and resources required for implementation, making the entire process more efficient and user-friendly. This approach further enhances the flexibility and adaptability of the framework, enabling the efficient creation and testing of a wide range of behavioral scenarios.

By incorporating these features, the Agora framework not only successfully replicated the original model but also showcased its potential for contributing to the field of agent-based simulations. This is illustrated by the effective integration of heatmaps representing underlying stimuli, which facilitates the blending of multiple behavior theories within the framework.

In comparing the Agora framework with the original work, it is essential to highlight both the limitations and the improvements introduced. One notable limitation is that the simulated agent responses to discomfort in the implemented model are simpler than in the original work. While agents in the original implementation can change clothes to regulate their temperature, agents in the Agora framework can only navigate to locations that offer greater comfort levels. Despite this limitation, the Agora framework introduces a range of advancements that enhance performance, versatility, and ease of use.

The original work suggests that the model's computational complexity scales linearly with respect to the discretized grid dimensions, implying that it would not impose a heavy performance burden on the overall simulation. Nevertheless, it was observed that in the Agora framework, simulating large environments or

using very small delta times could significantly hinder the model's performance. To address this issue, the thermal comfort model processing was re-implemented using Unity compute shaders[3], enabling parallel computation on the GPU. As a result, the simulation runs at a frame rate between 30-120 FPS, greatly improving performance.

Since the original work uses the ADAPT platform [130] as a crowd simulator, it is fair to assume that the proposed framework is characterized by the same strengths and limitations. For instance, it lacks compatibility with custom navigation mechanisms and doesn't offer any support for external configuration, making it difficult to compare models and explore novel functionality. Nevertheless, it's worth reiterating that the authors claim that the crowd simulator component can be easily changed. These limitations are discussed more in-depth in Section 2.3.1 and in [32].

In contrast, the Agora framework, based on Menge, addresses these limitations by offering native support for various pathfinding techniques and obstacle avoidance algorithms. This makes it easy to assess their impact on the simulation and to design different scenarios using Menge's custom specification files and Agora's convenient authoring GUIs. Consequently, the Agora framework allows for more convenient comparison and result dissemination.

Furthermore, Agora's heatmap-centered approach enables the easy integration of additional behavioral stimuli. For instance, a heatmap representing the influence of light and shadows on air temperature can be easily generated and incorporated into the node combination pipeline, enhancing the thermal comfort computation. Similarly, the visibility mapping computation method from case study one (Section 6.1) can be repurposed to guide agents toward more visible locations. This feature expands the range of possible behavioral combinations and offers a more powerful tool for agent-based simulations and behavioral analysis.

In conclusion, while there are some limitations when comparing the Agora framework to the original work, the advancements introduced by Agora in performance, versatility, and ease of use provide a more powerful and flexible platform for agent-based simulations and behavioral studies.

[3]https://docs.unity3d.com/Manual/class-ComputeShader.html

Chapter 7

Discussion

7.1 Main Results Summary

This research contributes to the field of crowd simulation by addressing critical limitations, developing a comprehensive theoretical framework, designing the software architecture, and implementing Agora, which has been assessed through two case studies. The key outcomes of this work are as follows:

1. **Literature Review**: This dissertation presents a review of the primary driving fields that have propelled innovation in the crowd simulation domain. Furthermore, it reveals the issue of fragmentation and characterizes it through a novel formalization, emphasizing the need for a framework that facilitates the combination of behavioral models to advance the field.

2. **Theoretical Framework**: Stemming from the literature review, a shift towards a novel stimulus-centric approach to crowd simulation is proposed. An innovative theoretical framework is designed that utilizes the heatmap paradigm for modeling stimuli that influence human reasoning and, consequently, agent behavior. The advantages and limitations of this new paradigm are explored, and the necessary rules and operations for working with this paradigm are defined: color operations for combining heatmaps; and three different ways of leveraging heatmaps for influencing agent behavior (goal selection, trajectory adjustment, and local steering). Finally, the theoretical foundation of techniques for leveraging heatmaps for calibrating model parameters and evaluating simulation output is proposed.

3. **Software Architecture**: The theoretical framework is formalized into a software architecture adhering to sensible architectural principles to ensure that the resulting framework can support the modeling of crowd simulations in several scenarios. As a result, the main critical components for the separation of concerns within the framework are defined.

4. **Agora Implementation**: Based on the aforementioned requirements, suitable selections of practical components are made, identifying ideal choices for

each of the architectural critical components. These practical components are extended and integrated, resulting in the implementation of the Agora framework. The overall backbone of Agora comprises the integration between Unity 3D and Menge, which collectively address the authoring, simulation, visualization, and evaluation aspects of the crowd simulation process. Unity 3D provides a set of tools for creating, editing, and rendering complex environments and characters, while the Menge crowd simulator offers a flexible platform for modeling and simulating agent behaviors in various scenarios. The combination of these two systems results in a comprehensive framework capable of addressing the challenges of crowd simulation.

5. **Case Studies**: The Agora framework's capability of supporting crowd simulations modeling and adhering to the previously mentioned architectural principles is tested with two case studies. Specifically, the first case study highlights the potential for objective evaluation of crowd simulations by comparing aggregate simulated agent trajectories with their real-world counterparts. In contrast, the second case study demonstrates plausible agent behaviors under varied density and thermal contexts combined in a holistic model. This underscores the capability of the software architecture and implementation to support the development of crowd simulation.

7.2 Answers to Research Questions

The focus of this research has been on addressing the fragmentation and limited interoperability of existing crowd simulation models. Although the field of crowd simulation has seen significant advancements, a unified approach that enables the seamless integration and combination of multiple behavior models is still lacking. This fragmentation poses challenges for creating more accurate and realistic simulations of human behaviors and complicates the fair comparison and evaluation of different models. Furthermore, the closely tied nature of behavior models to their underlying components and technologies, such as global pathfinding and local obstacle avoidance exacerbates the problem. The primary objective has been to develop a framework that can effectively combine diverse crowd simulation models, facilitate meaningful comparisons, and support the creation of more comprehensive and accurate simulations of human behavior in various scenarios and environments.

To address these challenges, Agora, a novel framework that leverages heatmaps for modeling agent behaviors, has been developed and evaluated through two comprehensive case studies. In light of the gathered results, the following discussion will address the research questions that guided this study, providing insights into the strengths and limitations of the proposed framework and its implications for the field of crowd simulation.

R.Q. I: What are the potential theoretical and technical challenges and limitations involved when attempting to integrate fragmented behavior models in crowd simulations?

Theoretical Challenges. The theoretical challenges concerning the combination of several behavior models can be partly attributed to the predominant rule-centric approach in behavior modeling, as opposed to a stimulus-centric one. In the rule-centric approach, behaviors are typically defined by a set of rules or algorithms that dictate agent responses in specific situations. One of the core issues in combining rules is that they are fundamentally semantical constructs, making it difficult to establish a clear and universally applicable method for combining them, especially when they are contradictory.

However, it is important to acknowledge that rule-based approaches have their own merits, such as a potentially higher expressive power, as rules can capture more nuanced and context-specific behaviors. Nevertheless, this expressiveness may come at the cost of reduced flexibility and modularity when attempting to integrate different behavior models.

On the other hand, a stimulus-centric approach focuses on the factors that influence the behaviors, rather than the specific rules governing the responses. With this perspective, stimuli can be more easily combined or modified using well-defined operations, such as color operations for heatmap-encoded stimuli. This enables a more flexible and modular framework for behavior modeling, allowing for the seamless integration of diverse behavior models in crowd simulations.

As shown by the two case studies (Chapter 6), by shifting the focus to the underlying stimuli that drive agent behavior, it becomes possible to create a more unified and versatile approach for integrating diverse behavior models in crowd simulations. This ultimately addresses the limitations posed by the rule-centric perspective while still acknowledging its strengths in capturing nuanced and context-specific behaviors.

Technical Challenges. The technical challenges in integrating fragmented behavior models in crowd simulations arise from the absence of a unified abstraction interface for implementing and combining behaviors. Each model is typically implemented directly on top of a specific stack of underlying components, such as the navigation system, and may require unique sets of data or expect the data to be provided in a particular format. These assumptions make it difficult to support multiple models in the same implementation without investing significant time and effort in adapting and integrating the models or the underlying systems.

Section (2.3) explained how some of these challenges have been partially addressed by crowd simulation frameworks that abstract and generalize the lower-level aspects of agent navigation within the environment. However, this remains an open challenge for higher-level behaviors, which are still not generalized and are closely tied to the underlying stacks of components. The native implementation of case study one exemplified these challenges in integrating higher-level behaviors (Sections 6.2.3 and 6.1.5).

By contrast, modeling behavior based on a central abstraction, such as the heatmap paradigm proposed in this research, allows for establishing a more unified and flexible way of combining diverse behavior models. This central abstraction can help overcome the limitations imposed by the tight coupling of higher-level behaviors to specific components and technologies, allowing for more seamless

integration and combination of multiple behavior models in crowd simulations. As demonstrated in the case studies, the Agora framework effectively utilizes the heatmap-based abstraction to address the challenges associated with integrating fragmented behavior models in crowd simulations.

R.Q. II: What paradigm can be employed to enable the seamless integration and combination of multiple behavior models in crowd simulations?

One paradigm that can be employed to foster the combination of behaviors is the heatmap paradigm. A heatmap is a graphical representation of spatial data that uses color intensity to indicate the magnitude of a variable, enabling the visualization and analysis of complex patterns, relationships, or behaviors within an environment. By adopting the heatmap paradigm, the focus shifts from a rule-centric to a stimulus-centric approach, which is beneficial for combining multiple behaviors into a more holistic model.

Key advantages of the heatmap paradigm contribute to its suitability for addressing the challenge of combining behavior models. First, heatmaps provide a unified method for encoding data related to various aspects of human behavior, allowing for easier integration of diverse models. Moreover, each behavior model is defined within a self-contained layer, encapsulating the necessary data for modeling, which can be enabled or disabled without impacting other simulation components.

The consistent format of behavioral information in heatmaps allows for the efficient combination of different heatmaps using color operations. This approach not only simplifies the integration process but also fosters adaptability by accommodating multiple models for different aspects of behavior and consolidating the resulting information into a single data structure. Furthermore, the graphical nature of heatmaps enables efficient processing through parallel GPU computing, enhancing simulation performance.

Applying the heatmap paradigm to both case studies (Chapter 6) demonstrates its potential for addressing the challenge of combining fragmented behavior models. By adopting this paradigm, the challenges associated with fragmented behavior models can be tackled, paving the way for more comprehensive and accurate simulations of human behavior in crowd simulations.

R.Q. III: Given a suitable paradigm for integrating multiple behaviors, what are the critical components and architectural principles that should be considered when designing and implementing a system that leverages this paradigm?

Architectural Principles. As argued in Section 4.1, the following key architectural principles should be considered:

1. **Usability**: A user-friendly interface is crucial to ensure accessibility for users with varying levels of experience, reduce barriers to entry, and facilitate collaboration among multiple parties.

2. **Modularity**: Implementing a modular design is essential for adapting the framework to different domains and scenarios, enabling customization, and facilitating the addition of new features and functionalities.

3. **Scalability**: The framework must accommodate varying levels of complexity in simulations by adjusting the level of detail in various aspects, from graphical elements to behavioral components, ensuring applicability across diverse scenarios and efficient simulation of complex crowd behaviors.

4. **Versatility**: The system should foster meaningful model comparisons, behavior combination, reusability, and collaboration, contributing to coherence and consensus within the field, and promoting innovation and collective advancements.

Critical Components. Based on the architectural principles, the following critical components should be supported:

1. **User Interface**: Enables efficient editing and management of environment, agent behavior, and visualization options, enhancing usability.

2. **Data Handler**: Manages seamless data flow and conversion, contributing to overall versatility.

3. **Crowd Generator**: Produces diverse, realistic crowds, addressing scalability.

4. **Crowd Simulator**: Runs simulations based on user-defined parameters, enhancing framework versatility.

5. **Visualizer**: Renders simulations in a visually accessible manner, addressing usability.

6. **Evaluator**: Assesses and validates simulation outputs, fostering continuous improvement and addressing usability and versatility.

7. **Plugin System**: Facilitates development and sharing of custom plugins, promoting modularity and versatility.

These components provide a comprehensive approach to crowd simulation, ensuring a clear separation of concerns and laying a solid foundation for a robust and adaptable framework to meet diverse needs in crowd simulation and analysis. Importantly, the architecture has proved successful in supporting the combination of crowd simulation models in the two case studies presented in this dissertation. This demonstrates the effectiveness of the proposed principles and components in achieving a flexible, adaptable, and robust crowd simulation framework.

R.Q. IV: How can the integration of multiple behavior models contribute to improved realism and accuracy in representing human behaviors and interactions in crowd simulations?

Human behavior is multifaceted, and no single model can capture its complexity. As seen in the literature (Section 2.5), the divide-and-conquer approach adopted by the field of crowd simulation has allowed researchers to focus on more manageable issues and contained behaviors, resulting in significant advancements. However, real human beings account for many different factors in their reasoning and behavioral processes, highlighting the need for integrating multiple behavior models.

The integration of multiple behavior models leads to a more holistic and realistic portrayal of human behavior in crowd simulations. By leveraging the strengths of different models, simulations can benefit from a comprehensive understanding of human behavior and its various influencing factors. Combining multiple models can also help reduce inherent limitations and biases that individual models may possess, leading to a more balanced and accurate representation of human behavior.

For instance, in the case study two (Section 6.2) involving the combination of two comfort models, emergent behaviors were observed that closely resembled those described in the original work [23]. Agents demonstrated plausible behaviors under varied density and thermal contexts, adjusting their movements to improve thermal balance or seek density comfort. Such outcomes would not have been possible if only one of the stimuli were modeled. As demonstrated in the referenced section, Agora generalizes this combination of stimuli, representing an advancement over the current state of the art.

R.Q. V: How can objective evaluation methods be devised to assess the realism of various crowd simulation models, while maintaining a balance between evaluating individual behaviors at a local level and their wider interactions in diverse scenarios and environments?

Evaluation methods can be classified into two categories. Microscopic methods focus on individual agents by comparing single variables, such as individual trajectories. Conversely, macroscopic methods are based on completely aggregate data, such as the overall crowd density/velocity ratio, which is compared to the fundamental diagram.

Heatmap-based evaluation techniques offer a balanced approach that addresses the challenges of both microscopic and macroscopic evaluation. By providing a graphical representation of spatial data, heatmaps indicate the *aggregated* magnitude of a variable. Thus, comparing simulation-generated heatmaps with their real-world counterparts enables an indirect balance between microscopic and macroscopic evaluation. This approach shares similarities with the work on trending paths [146], which clusters trajectory data to produce aggregate trajectories for more robust and meaningful comparisons.

Although the trending paths method employs more advanced techniques, such as Bayesian inference for learning patterns, heatmaps offer similar benefits with a simpler approach. Furthermore, they facilitate the use of well-established image similarity metrics, which can effectively assess the difference between simulated and real-world data. These metrics help to strike a balance between micro and macroscopic evaluation by adjusting parameters, such as the sliding window size in the case of the Structural Similarity Index (SSIM). A smaller window size focuses

on local differences, capturing fine-grained details and discrepancies, while a larger window size considers broader structural similarities and is less sensitive to local noise or small variations.

Additionally, the heatmap approach is stimulus-centric, with each behavioral stimulus encapsulated in its layer. This enables the evaluation of both individual and combined stimuli against real-world data, providing insights into their contributions to the overall behavior. This is exemplified by the application of the implemented evaluator in case study one (Section 6.1.4), where the simulated agents' trajectories were projected onto a heatmap and compared to the real-world counterpart. This method allowed for the comparison of aggregate agent trajectories rather than individual ones, leading to a more comprehensive evaluation of the crowd simulation model.

7.3 Advantages and Limitations

This section presents an overview of the key advantages and limitations of the Agora framework, highlighting its potential impact on the field of crowd simulation and outlining areas for further improvement. The advantages that position Agora as a powerful and versatile solution are discussed first, followed by an examination of its limitations, which serve as a foundation for refining and expanding the framework in future work.

7.3.1 Advantages

The Agora framework addresses key challenges of crowd simulation and fulfills the requirements and user considerations identified during the design of the architecture. With its novel heatmap-based approach, Agora supports behavior modeling and combination. Among its key advantages are behavior encapsulation, allowing for flexibility in examining isolated effects, and improved model comparison through image similarity metrics.

Agora streamlines the customization of navigation components and crowd appearance, adapting to specific project requirements. By leveraging parallel processing on the GPU, the framework delivers enhanced performance, scalability, and usability.

Promoting collaboration and knowledge exchange among researchers, Agora facilitates sharing findings and models, driving the development of sophisticated and accurate crowd simulation models. These strengths position Agora as a powerful and versatile solution for a wide range of applications in the field.

This section provides a detailed examination of these key strengths, illustrating Agora's potential to advance the state of the art in crowd simulation.

Paradigm Shift. In situations where modeling the underlying reasoning behind behaviors is challenging or not easily quantifiable, the paradigm shift from directly modeling behavioral rules to producing heatmaps can be particularly beneficial. By focusing on more tangible and measurable stimuli, researchers can represent the factors that influence agent behavior more straightforwardly. The case studies

presented in Sections 6.1 and 6.2 exemplify this approach, with successful modeling of visibility, thermal, and density comfort levels using the heatmap paradigm.

The heatmap paradigm serves as the foundation for representing and combining various stimuli in a more quantifiable and well-defined manner. By focusing on external, quantifiable stimuli rather than the complex reasoning driving behaviors, researchers can create simulations that better reflect the intricacies of human social behavior in crowd simulations. This approach provides a more tangible representation of stimuli, making it easier to work with and combine in a versatile manner, as opposed to combining behaviors that are implemented through programming and encompass complex rules, as argued in the literature analysis of Section 2.5. The heatmap approach thus lays the groundwork for a more generally applicable and versatile combination, addressing the main objective of Versatility (Section 4.1.4).

This new paradigm further facilitates the adoption of data-driven methods in modeling crowd behavior. By utilizing machine learning techniques within the Agora framework, it could be possible to automatically generate specific heatmaps that are capable of eliciting certain emergent behaviors as seen in real-life scenarios.

Heatmap Potential and Advancements. Heatmaps have shown their considerable value as an innovative tool in facilitating the design of stimulus-centered crowd simulation models. Their implementation enables the encoding of numerous stimuli that drive human behavior into distinct layers. These layers can be blended to construct a comprehensive representation of human decision-making processes.

This layered encapsulation brings two significant advantages: the capacity for independent behavioral analysis and the ease of behavioral component sharing. The implementation of heatmaps allows for the isolation and examination of individual behavioral components without interfering with other aspects of the simulation. It offers the flexibility to adjust the impact of specific stimuli by weighting their influence or by entirely removing them from the behavior finite state machine.

Additionally, the heatmap approach enhances the accessibility and exchange of behavioral components within the research community. This can be achieved by sharing static heatmaps or the methodology behind their creation. As graphical representations, heatmaps are computationally efficient, enabling the parallel processing of dynamic and evolving stimuli on the GPU, thereby not impairing the simulation's performance. This improvement upon static map authoring offers an advanced mechanism for behavior modeling, supporting a broad range of existing works in the literature (further explored in section 7.4). Notably, the second case study highlighted the efficiency of this paradigm in integrating additional features into pre-existing models, as shown by the seamless incorporation of shadow-seeking behavior into the thermal comfort model of Section 6.2.4.

Looking toward the future, the heatmap approach paves the way for a comprehensive library of behavioral models. If researchers adopt this stimulus-centric modeling technique, it will allow for a wide variety of behavioral models that can be seamlessly integrated into a BFSM and combined. Such an approach would not only facilitate the creation of more holistic models of human behavior but also significantly propel the state of the art in crowd simulation modeling.

Ease of Authoring Behavior. As shown by the first case study in Section 6.1.3, the Agora framework simplifies the process of authoring behaviors for crowd simulations compared to implementing native simulations from scratch. Agora provides tailored solutions to address various challenges that arise in crowd simulation, particularly in the area of behavior authoring, which is in line with the Usability requirement (Section 4.1.1).

This claim was substantiated by the authoring of the visibility behavior. Using Agora, it was straightforward to employ the heatmap goal selection on the visibility stimulus to enable agents to perceive their surroundings and choose goals leading to the most visible locations, all done using a visual programming paradigm. This focus on a more accessible and user-friendly interface for creating and editing heatmaps builds upon the attempts to simplify behavioral authoring discussed in the background section (Section 2.4). By taking the visual map-based approach one step further, Agora effectively pushes the state of the art in terms of simplifying the process of authoring complex behaviors for crowd simulations.

In contrast, achieving the same result in the native simulation proved to be more challenging, requiring custom programming, which involved creating a custom heuristic for evaluating the maps and projecting goals onto Unity's navigation mesh. By offering a more accessible and streamlined process, Agora allows researchers to focus on the nuances of human social behavior rather than grappling with the complexities of native implementation.

Behavior Encapsulation. The independence of heatmap-based behaviors offers numerous practical advantages in crowd simulations, as argued in Section 3.3.1. In this approach, stimuli computations rely only on the data contained in their corresponding heatmaps and can be easily incorporated into the behavioral finite state machine. This stands in contrast to native implementations, where the classical programming paradigm can lead to tight coupling and interlocking of behaviors.

The benefits of this encapsulation were evident in the first case study. As shown in Section 6.1.5, the visibility behavior became intertwined with the custom-made heuristic that also used memory of previously visited locations and other factors to influence overall behavior. The heatmap approach, on the other hand, enabled the creation of a self-contained, visibility-based goal selection behavior. This paradigm encourages researchers to confront the shortcomings of a particular behavior theory and highlights that modeling just one behavior may not be sufficient for capturing human reasoning. At the same time, it prevents the supplementation of a specific stimulus with unrelated stimuli.

Additionally, encapsulation provides the flexibility to toggle layers on and off, allowing for the examination of their isolated effects on the simulation. As could be observed in Figure 6.16 of the second case study, it would be straightforward to disable one comfort level to study the influence of the other. Likewise, it would be possible to adjust the weights of each heatmap and observe the resulting behavior changes. Although this could have been done in the original implementation by modifying parameters controlling the weighted average of the comfort levels, the Agora framework generalizes this process, enabling the independent toggling of behaviors regardless of how they are combined.

As explained in Section 2.3, previous frameworks have achieved this encapsulation for other lower-level components of crowd simulation such as path planning and adaptation, but not yet for the behaviors. Agora pushes the state of the art as it lays the foundation for behaviors to be treated as independent "puzzle pieces" that can be fit together without problems - effectively achieving the modular design proposed in Section 4.1.2.

Model Comparison. The first case study demonstrated that the Agora framework facilitates the assessment of the simulation output's closeness to the real-world phenomenon being modeled. As highlighted in the literature review in Section 2.2.6, model evaluation is sometimes overlooked in the field of crowd simulation. One reason for this tendency is the difficulty of performing model evaluation.

Agora simplifies this process by offering a convenient method for importing real-world data and converting it into heatmaps, as well as an intuitive graphical user interface for comparing these heatmaps with those generated by the simulation. This comparison utilizes well-established image similarity metrics, as explained in the reference provided (Figure 5.8).

By proposing the application of image similarity metrics to heatmaps built from simulation output, Agora pushes the state of the art with respect to model evaluation. This novel application strikes a good balance between low-level and high-level evaluation, enabling more effective evaluation methods. Furthermore, since each behavioral stimulus is encapsulated in a single node, it becomes much easier to assess the contribution of each stimulus to the overall similarity between the simulation and the real-world data.

This streamlined process for model comparison not only enhances the versatility of the Agora framework (Section 4.1.4), but also allows researchers to better understand the quality of their simulations and refine them to more closely represent real-world human social behavior in crowd simulations. By providing versatile and innovative evaluation methods, Agora fosters a more comprehensive understanding of crowd dynamics, ultimately leading to more accurate and reliable simulations.

Replaceable Components. The Agora framework simplifies the process of changing the underlying navigation components, such as pathfinding and obstacle avoidance. As argued in the literature review (Section 2.5), this is typically a challenging task that requires implementing or integrating specific algorithms and adapting existing behaviors to account for the unique features of each component.

Since Agora is based on Menge, it allows for conveniently selecting the global pathfinding and local obstacle avoidance mechanisms. To modify global pathfinding, users only need to create the relevant environment representation (e.g., a waypoint graph) and select it as the "velocity component" in the corresponding BFSM state input. Changing local obstacle avoidance is even more straightforward; users can simply select a different pedestrian model in the Menge simulator GameObject within the scene.

This versatility in navigation components enables researchers and developers to experiment with different approaches and optimize their crowd simulations more

efficiently, without the need for extensive reimplementation or adjustments to existing behaviors.

Versatile Crowd Generation. As demonstrated in the case studies, the ease of customizing crowd appearance was beneficial in terms of adapting the simulation to suit the specific requirements and goals of each scenario. Agora allows for easy variation of crowd appearance to address challenges of variety and to adapt to the required level of detail. Variety is addressed by UMA, which simplifies the creation and instantiation of different-looking agents. The level of detail is managed by the GUI, which lets users choose whether to use more realistic UMA models (rigged and animated) or more simplistic static models, enhancing scalability (Section 4.1.3).

For example, case study one didn't require detailed human models, as the focus was on the simulation output trajectories for evaluation. In this instance, it was possible to use simplistic characters to improve performance. Conversely, case study two was more focused on the visual behavioral responses, making more detailed (and animated) models more fitting. Creating the crowd was straightforward, involving the definition of UMA population parameters such as DNA, colors, and wardrobe (see Section 5.2).

The overall agent management pipeline, including crowd definition and spawning, is seamlessly integrated into Agora. Once the population features have been defined, it is possible to create "scenarios" for dynamically spawning agents at runtime, which are then managed by Menge and visualized in the renderer. This versatile crowd generation capability streamlines the simulation design process and allows for better adaptation to specific project requirements.

By integrating UMA into the framework, Agora effectively addresses a common gap in crowd simulation frameworks, which often do not incorporate techniques for generating diverse agents. Agora bridges this gap by applying techniques from the crowd variety field (Section 2.2.2) to the framework domain (Section 2.3), demonstrating the potential for enhanced adaptability and realism in crowd simulations.

Enhanced Performance. As demonstrated in case study two, the heatmap approach is an optimal venue for leveraging parallel processing in the GPU (see Section 6.2.5). The encapsulation offered by heatmaps simplifies the process of parallelizing computations, as synchronization issues between different data sources are reduced. By utilizing compute shaders to generate heatmaps in parallel on the GPU, a significant performance improvement was observed, with the simulation frame rate increasing from less than 1 FPS to over 30 FPS.

This increase in performance is crucial for leveraging dynamic heatmaps that are updated at runtime and can be visualized in real time within the xNode BFSM GUI (see Figure 6.16). Agora's heatmap paradigm not only simplifies the representation of behavioral stimuli but also facilitates parallel processing, ultimately leading to more efficient and responsive crowd simulations.

The enhanced performance offered by Agora addresses both scalability and usability concerns (Sections 4.1.3 and 4.1.1). It enables the simulation of a larger number of agents or more complex behavior while maintaining acceptable frame

rates. In addition, the use of dynamic heatmaps in the xNode BFSM GUI would not have been possible if they were being sequentially generated on the CPU. The GPU parallelism in Agora opens up this possibility, providing a more interactive and user-friendly experience.

The dynamic heatmap approach in Agora represents an innovative step forward in the realm of crowd simulation authoring. While existing methods, as discussed in Section 2.4, utilize static maps to author agent behavior, Agora introduces the use of dynamic heatmaps, that reflect a constantly changing world. This novel approach, made possible by the GPU-based implementation, enables Agora to design and implement more flexible and adaptable crowd simulation models.

Facilitating Collaboration and Dissemination. Agora streamlines the process of sharing findings, models, and extensions among researchers. This is partly attributable to the encapsulation provided by heatmaps, which are independent of other data sources, and partly because of Menge's modular design, which allows for easy integration of additional components.

With Agora, heatmaps and heatmap generation processes can be effortlessly extracted and integrated into another simulation model, even if it utilizes a different combination of subcomponents. This flexibility encourages collaboration between researchers, as they can readily adapt and build upon each other's work, fostering the exchange of knowledge and the development of more sophisticated and accurate crowd simulation models.

7.3.2 Limitations

While the Agora framework demonstrates a step forward in crowd simulation, it is essential to acknowledge its limitations. These limitations include challenges in representing sparse phenomena, capturing the full range of social behaviors, and implementing certain aspects of the theoretical framework. Furthermore, the focus on movement may exclude more diverse agent behaviors. Moreover, the integration with Menge and the user interface can be improved for enhanced usability and stability. Recognizing these limitations provides a foundation for refining Agora and expanding its capabilities in future work.

Challenges with Sparse Phenomena. Heatmaps excel at modeling dense phenomena, such as scalar values in the spatial domain. However, they can struggle to represent sparse phenomena effectively.

For example, the heatmaps of case study one were representing the visibility of the walkable areas of Þingvellir. Since the majority of the environment was non-walkable (sparse data), the heatmap was mostly empty. This is an issue for the heatmap paradigm because it still needs to allocate memory for the entire heatmap and possibly perform many unnecessary operations on empty cells. While it is possible to contain this issue by employing masks that define the relevant heatmap cells, the representation can still become unwieldy and computationally inefficient.

In such situations, alternative modeling approaches that efficiently handle sparse data might be more appropriate, as they can better capture the features of the

environment and agent interactions.

Simplistic Attraction Model. While the heatmap paradigm provides innovative ways to influence agent behavior through goal selection, velocity modification, and transitions, it's ultimately based on a simplistic attraction and repulsion model. Similar to the social force model [66], agents are influenced by attractive and repulsive forces, which can be seen as a reinterpretation of the existing paradigm.

For instance, in the second case study (Section 6.2), heatmaps encoded discomfort levels, with higher values indicating increased discomfort. Goal selection based on the highest pixel was not directly applicable, as it led agents towards greater discomfort. This was addressed by using the inversion operation, which computed the most comfortable locations. However, this simplistic approach may not be suitable for all applications.

Furthermore, the model can only "nudge" agents into performing appropriate actions based on heatmaps, but it doesn't guarantee expected behavior. This represents a shift from a "puppeteer-like" paradigm, where authors have more control over behaviors, to an agent-centric approach, where emergent behaviors result from stimuli combinations. This might not be suitable for all purposes. In certain contexts, such as video games or movies, granular control over agents' actions is more important than flexibility, as reliable behaviors are required.

Lack of Calibration. Section 2.2.5 highlighted the importance of calibrating the model's parameters. However, this aspect was not fully implemented, leaving Agora without an integrated way to determine the extent to which the chosen parameters influence the quality of a particular model.

Nevertheless, the evaluation metric used to compare simulation output with real-world data plays a significant role in the calibration process. Image similarity metrics, as implemented in Section 5.6.2, address this aspect. Additionally, the advantages of using heatmaps for the calibration process were discussed in Section 3.6.

While the calibration was not fully implemented, this dissertation lays the groundwork for building techniques that leverage heatmaps for parameter calibration in the future.

Unimplemented Heatmap Trajectory Modifier. The heatmap trajectory modifier proposed in Section 3.4.2 was not implemented in the framework. This component was intended to act as an additional adaptation step between global path computation and local obstacle avoidance, considering heatmaps in the process.

A key challenge in implementing this feature was that not all velocity components in Menge return a complete trajectory as output. For instance, in the vector field velocity component, the preferred velocity is defined for each agent based on their position relative to a uniformly discretized 2D grid, resulting in an instantaneous velocity. Since a complete trajectory is neither computed nor stored within the pipeline, an extra step would be needed to convert these instantaneous velocities into complete trajectories, which could then be adapted from source to destination.

Despite the lack of implementation, the theoretical foundation for this mechanism has been established in this dissertation, contributing to the overall understanding of how such a modifier could work in future research.

Heatmap Usability. The usability of heatmaps within the framework and the associated graphical user interfaces could benefit from improvements to better support user experience. One area of concern is the relationship between world units and pixel units, which can be adjusted using a scale parameter. While this approach is functional, it may prove unintuitive and cumbersome. For instance, when defining perception masks in the heatmap goal selector, taking this distance into account is crucial to avoid unintended consequences on agent perception.

Additionally, the "out-of-bound" strategy, as described in Section 3.3.5, relies on Unity's texture wrap mode[1]. This connection may be overlooked, posing potential challenges to the heatmap combination process.

The framework's relative heatmap was designed to define agent-centric "zones" that interact with the surrounding environment, simulating the impact of local stimuli on overall behavior, such as proxemics with intimate, personal, and social zones. Although the current implementation using offsets and centering achieves its intended purpose, there is room for improvement, particularly in GUI design and user-friendliness.

Thus, the framework would greatly benefit from refining these features and enhancing the overall usability and convenience for new users.

Movement Focus. The Agora framework primarily addresses the movement aspect of crowd simulation, modeling the impact of various stimuli on agents' behaviors and simulating their responses as they navigate the environment. This focus on movement represents only one dimension of the complex range of behavioral responses that individuals can exhibit in response to internal reasoning. Although more intricate behaviors can still be simulated behind the scenes to some extent through the use of BFSM states, effectively representing and rendering these behaviors remains a challenge. Developing a generalized approach that can be applied to a wide variety of scenarios is particularly difficult, given the nearly limitless set of actions real people can perform. However, the possibility of applying the heatmap paradigm to other behaviors has not been fully explored yet.

Limited Exploration of Direct Social Stimuli. While Sections 2.1 and 2.2.3 discussed human behaviors more directly related to social aspects, such as social groups, social activities, and social emotions, the conducted case studies primarily focused on physically grounded stimuli like visibility, thermal comfort, and density. Although these stimuli do influence human social behavior – as Griffit and Veitch [56] reported, various positive mood aspects decrease in denser crowds while negative aspects increase – the current scope of the studies does not fully address direct social stimuli such as group behavior.

[1] https://docs.unity3d.com/ScriptReference/TextureWrapMode.html

The heatmap paradigm shows promise in supporting more direct social stimuli, but further research is needed to validate this claim. Conducting a heatmap-based crowd simulation model and case study or user study that directly incorporates social behavior would be a valuable contribution to future work.

Menge Limited Integration. While Menge is a powerful crowd simulation system, it has its own set of limitations, exacerbated by the integration with Unity.

Menge's strict handling of exceptions, particularly fatal exceptions, can lead to stability issues when integrating it with Unity through P/Invoke (see Section 5.1.1). For instance, when using the navigation mesh velocity component, if an agent requests to compute a path outside the navigation mesh, Menge throws a fatal exception that cannot be caught by Unity, causing the Unity editor to crash. Although this issue was mitigated by modifying Menge's source code to prevent throwing such exceptions, it would be more robust to handle them appropriately.

Menge's design is primarily configuration-centric, relying on XML files for scene and behavior setup. This approach leaves little room for runtime intervention and dynamic adjustments. While the custom event callback system and dynamic agent spawning mechanisms implemented in this work provide more flexibility, the ability to create and manage various simulation components programmatically remains limited.

Menge uses its format for defining spatial query data structures, necessitating conversions between various formats. Although utilities exist to simplify this process, it can be cumbersome and complex. For instance, using Unity's navigation mesh with Agora requires multiple steps, including exporting the underlying data structure with a custom script, conditioning the mesh in a 3D modeling application, and converting the file to a Menge-compatible format. Streamlining this process and integrating it directly into the Agora framework would significantly improve usability and user experience.

7.4 Relevance of Agora to Works of the Literature

This section assesses the compatibility of the Agora framework with various works in the literature in the field of crowd simulation. The evaluation process involves examining the features and limitations of Agora and considering how they may affect the authoring of each crowd simulation behavioral model.

To facilitate this evaluation, two tables are provided. The first Table 7.1 outlines the features and limitations of Agora, accompanied by brief descriptions. Each feature (F1-F5) is presented as a potential advantage that could be compatible with a given work, while each limitation (L1-L5) is identified as a potential constraint that may impede a given work.

The second Table 7.2 applies these criteria to different works presented in the literature review of Section 2.2.3. Each work is assessed for each feature and limitation of Agora, receiving a compatibility rating of "bad", "neutral", or "good". This approach avoids numeric scoring, instead offering a more open and nuanced measure of compatibility. The ratings across features and limitations give a broad view of how well Agora might work with each work.

This methodology offers a clear and systematic approach to assess the applicability and relevance of Agora to other works in the crowd simulation domain. Its purpose is to enhance understanding of the unique capabilities and challenges of the Agora framework and to determine how they may influence different research contexts. The analysis aims to identify opportunities for utilizing Agora in novel and impactful ways, as well as to highlight areas for future development and improvement of the framework.

Label	Title	Description
F1	Crowd Variety & Visualization	To what extent would work X be compatible with the crowd authoring tools and visualization?
F2	Heatmap Authoring	To what extent would work X be compatible with the heatmap authoring?
F3	Behavior Combination	To what extent would work X be compatible with the behavior combination approach?
F4	Component Independence	To what extent would work X be compatible with the offered component independence?
F5	Image Similarity Evaluator	To what extent would work X be compatible with the image similarity evaluator?
L1	Sparse Phenomena & Memory	To what extent does work X rely on sparse phenomena or memory?
L2	Simplistic Attraction Model	To what extent would the simplistic attraction model hinder work X?
L3	Lack of Calibration	To what extent would the lack of calibration hinder work X?
L4	Movement Focus	To what extent does work X rely on actions other than movement?
L5	No Granular Control	To what extent would the absence of granular control hinder work X?

Table 7.1: This table presents an explanation of the labels used for features (F1-F5) and limitations (L1-L5) in the evaluation of the Agora framework. It provides the basis for the assessment in the subsequent Table 7.2.

Ref.	Description	F1	F2	F3	F4	F5	L1	L2	L3	L4	L5	Score
[85]	Group behavior affected by relationships modeled as a social network.	g	b	g	n	n	b	n	g	g	g	**n**
[76]	Joining and leaving groups based on boredom and habituation.	g	g	n	g	g	g	g	g	n	g	**g**
[90]	Personality traits and emotions modeled as heat transfer function.	g	n	g	g	g	g	g	n	g	g	**g**
[45]	Scripted behavioral responses to different emotional states of agents.	g	g	n	g	g	g	n	g	g	n	**g**
[129]	Artificial life agents guided by internal motifs perform several actions in the surrounding environment.	g	n	n	n	b	b	b	n	b	b	**b**
[70]	Agents decide whether or not to hold the door based on internal reasoning.	b	n	n	n	b	g	b	n	b	b	**b**

Table 7.2: Evaluation of various crowd simulation works from the literature review of Section 2.2.3 against the features (F1-F5) and limitations (L1-L5) of the Agora framework. Each work is assessed with a compatibility rating of [b]ad, [n]eutral, or [g]ood.

Social Group Models. In the work of Li and Lin [85], the crowd variety and visualization features of Agora could improve the representation of agents in this work, currently depicted as circles. However, mapping social networks onto heatmaps could prove challenging. The work does employ multiple forces, suggesting that Agora's behavior combination could be beneficial. As for component independence and the image similarity evaluator, these features could be adapted, as the work is based on steering behaviors and lacks an existing evaluation method. Regarding limitations, the work's dependence on possibly sparse social networks and prior information could clash with Agora's approach to sparse phenomena and memory. The complexity of social network connections may also present a hurdle to Agora's simplistic attraction model. Nevertheless, the work's focus on movement aligns well with Agora's emphasis on the same, and the lack of need for granular control in the work means Agora's lack of it is not an issue. Also, the work's limited parameters might make Agora's lack of calibration a non-problem. In summary, despite some challenges, this work shows areas of potential alignment with Agora, especially regarding crowd visualization, behavior combination, and focus on movement.

The work from Karimaghalou et al. [76] makes good use of spatial data and employs SmartBody, an open-source character animation platform, which aligns well with Agora's focus on crowd variety, visualization, and heatmap authoring. Although the work primarily addresses a single stimulus, limiting the use of Agora's behavior combination feature, its modular crowd simulation approach resonates with the component independence offered by Agora. Also, the work's existing heatmap of the simulation could effectively complement Agora's image similarity evaluator. Regarding Agora's limitations, this work is not significantly affected. It does not rely heavily on sparse phenomena or memory-based behaviors, and its foundation on group attraction and repulsion align with Agora's simplistic attraction model. With a minimalistic design involving a small number of parameters, the lack of calibration in Agora wouldn't pose a substantial problem. Although movement is a key focus in this work, it also includes elements of group behavior, balancing Agora's emphasis on movement. Importantly, the work does not necessitate granular control, aligning with one of Agora's limitations. Overall, this assessment indicates a promising potential for the integration of this work with the Agora framework, given its inherent alignment with several key features and its minimal hindrance from Agora's limitations.

Social Emotion Models. The work of Mao et al. [90] influences agents' behaviors based on their personality traits and emotion. Regarding Agora's features, the visualization tools could enhance the existing agent representation, which currently uses cylinders but is already compatible with video rendering. While the transformation of personality and emotion data into the heatmap paradigm would require extra effort, the work's use of a dissipation model, known to work well in Agora, suggests feasibility. The multiple behavioral aspects at work - personality, emotion, intra-group/extra-group interaction, and third-party influences - indicate a strong alignment with Agora's ability to combine behaviors. This, along with the modular nature of the work, suggests that it could effectively integrate with Agora's feature of component independence. Further, the work's existing practice of comparing

evacuation times with other models points to a successful adaptation of Agora's image similarity evaluator. In terms of limitations, the work aligns with Agora by not relying on sparse phenomena or memory. Its use of a threshold-based attraction adjustment for path planning is compatible with Agora's simplistic attraction model. The focus on movement in the work aligns well with Agora's similar focus, and the absence of granular control in the work fits with Agora's identical limitation. However, calibration might be a point of contention, given the lack of calibration in the work and the presence of several parameters. In conclusion, this work presents a promising opportunity for the application of the Agora framework. Some areas, such as heatmap authoring and calibration, may require careful consideration and adaptation.

Agora's feature set appears to adapt well to the work of Faroqi and Mesgari [45]. The 2D visualization currently used could be significantly enhanced by Agora's crowd variety and visualization tools. Furthermore, the relatively simplistic emotional state model employed in this work aligns well with Agora's heatmap authoring capabilities. This work focuses on a single stimulus, which neither benefits from nor conflicts with Agora's behavior combination feature. The model, which is built with NetLogo, is compliant with Agora's emphasis on component independence. Finally, the straightforward nature of extracting trajectories for comparison with the original work makes the image similarity evaluator of Agora an appropriate fit. The limitations of Agora also line up reasonably well with this work. The absence of sparse phenomena or memory matches Agora's limitations. The model's attraction mechanism, though somewhat scripted, operates on simple underlying rules, suggesting possible compatibility with Agora's simplistic attraction model. The lack of calibration in the model, coupled with a small number of parameters, indicates a good alignment with Agora's limitations. The work's exclusive focus on movement is a plus, aligning seamlessly with Agora's movement focus. However, the presence of security agents directing others' movements might pose a challenge to Agora's lack of granular control. Overall, this work presents a promising opportunity for the application of the Agora framework, although some aspects, such as the scripted attraction model and granular control, would require adaptation.

Social Activities. The work from Shao and Terzopoulos [129] which undertakes an artificial life simulation at Pennsylvania train station poses considerable challenges for integration with the Agora framework. The current use of DI-Guy for models and animations aligns well with Agora's capabilities. However, its complex behavior model is not entirely suited for Agora's heatmap authoring, and combining various tasks and stimuli might be challenging. While there's an overlap in the modular approach between Agora and the study, certain modules like thirst and curiosity levels aren't natively supported in Agora. Furthermore, the evaluation methodology, based on permutations of agent actions, doesn't fit Agora's image similarity evaluator. The study's use of memory and its intricate tasks pose challenges to Agora's simplistic attraction model. No calibration is performed in the study, but with possibly many parameters, issues could arise with Agora's lack of calibration. Also, agents in the study perform various actions beyond movement, differing from Agora's focus. Finally, the study's requirement for granular control isn't supported in Agora.

In conclusion, despite some possible adaptations, the overall compatibility between Agora and this work is limited due to Agora's restrictions and the study's task complexity.

Likewise, the work of Huang and Terzopoulos [70] presents several challenges for integration with the Agora framework. The work explores the behavior of virtual agents in the context of door etiquette, where their decisions to hold the door for others are influenced by several stimuli. It relies on procedural animations, while Agora utilizes keyframe animations. Moreover, encoding the agents' internal reasoning into heatmaps could be complex. While both frameworks make use of modular components, differences in handling animations and locomotion might pose some obstacles. Additionally, Agora's image similarity evaluator may not be suitable for assessing the sequence of actions required in the door etiquette behavior. Regarding limitations, the lack of sparse phenomena or memory in the work aligns with Agora's capabilities. However, the nuanced door etiquette scenario may test the boundaries of Agora's simplistic attraction model. Furthermore, the study's focus on non-movement actions and the possible need for granular control do not align well with Agora's preferences. Given these considerations, the overall compatibility of the study with Agora is low.

This analysis, although limited in scope, provides valuable insights into the potential applications and limitations of the Agora framework in various research contexts. Agora is well-suited for studies focusing on crowd behavior influenced by stimuli, such as those involving group dynamics or social emotions. These can be effectively encoded into heatmaps, allowing the agents' actions to be influenced through Agora's attraction model in coordination with the underlying navigation module.

However, the analysis also highlights some challenges Agora might face in more nuanced social scenarios, such as artificial life simulations or specific social etiquettes. These scenarios often involve more intricate actions beyond movement, which is Agora's primary focus. Moreover, they often embody complex rules that may not be easily interpreted or represented with heatmaps, potentially requiring a greater level of control and precision.

In summary, while Agora exhibits significant potential in certain areas of crowd simulation research, the present analysis underscores the importance of further development and fine-tuning to enhance its adaptability in more complex and nuanced scenarios.

Chapter 8

Future Work

The potential of the Agora framework extends beyond the current implementation and successful case studies. This chapter outlines a roadmap for evaluating the framework's scalability, modularity, usability, and versatility. The subsequent sections discuss the potential addition of social behaviors to enrich crowd simulations, the initiation of perceptual studies to validate their realism and effectiveness, and possible extensions to the theoretical and implementation aspects of the framework. Lastly, improvements to the integration with the Menge Crowd Simulation Framework are addressed, focusing on enhancing user experience and utility. The collective aim is to guide further development and ensure the framework's continued evolution and applicability across various domains.

8.1 Roadmap for Principles Assessment

Agora framework's potential for further refinement and research is vast, despite extensive testing and successful implementation in two case studies. The first case study served as a testbed for the evaluation technique, showcasing its effectiveness by comparing the simulation's heatmap output with real-world data from a field study. The second case study exhibited the framework's ability to integrate multiple behavior models, simulating individuals balancing thermal comfort and crowd density. Although these achievements are significant, numerous unexplored scenarios, modalities, and contexts remain. Consequently, this section suggests a roadmap of tests for assessing the framework's features, objectives, and architectural principles.

1. **Scalability Assessment:** This test aims to understand how well the system performs under high load conditions and handles an increasing number of complex behaviors. For example, a simulation could be designed to mimic a large public event, like a music festival, with thousands of virtual attendees interacting simultaneously. The performance metrics evaluated could include frame rate, response time to user input, and the overall realism of the crowd dynamics.

2. **Modularity Evaluation:** This examination focuses on the Agora framework's modular design and the ease of integrating various components. A possible test scenario could involve exploring crowd-generation solutions alternatives to Unity Multipurpose Avatar, such as MakeHuman or Character Creator. This would provide insights into how smoothly different solutions can be incorporated and how they perform. Another aspect to consider is the interchangeability of navigation algorithms. Investigations could involve interchanging various navigation techniques supported by Menge such as waypoint graphs or uniform grids to evaluate their implementation and performance in the given framework. To further probe the system's extensibility, an additional feature could be introduced via the plugin system. This could involve implementing an enhancement to the heatmap authoring tool, such as integrating a painting extension.

3. **Usability Testing:** These tests aim to evaluate the ease of use and the user experience of the Agora framework. An initial exploratory phase could involve users from various domains, such as urban planning or event management, creating crowd simulations and providing feedback. This could transition into a formal user study, asking participants to undertake specific tasks and assessing metrics such as user satisfaction, time taken, and simulation accuracy. Lastly, expert-led heuristic evaluations could be conducted. These evaluations would involve experts systematically inspecting the interface, potentially creating a crowd simulation of a complex scenario such as an emergency evacuation, and assessing the system against established usability principles.

4. **Versatility Exploration:** This investigation seeks to evaluate the flexibility of the Agora framework across various application domains and behavior theories. Each domain—urban planning, safety, architecture, and entertainment—has distinct requirements. For instance, urban planners and safety experts might prioritize interpretability and accuracy over real-time performance, while the entertainment industry emphasizes visually convincing simulations that operate smoothly in real-time. Architects may value the capability to evaluate unique crowd behaviors in custom-shaped environments. Additionally, the framework's adaptability to different behavior theories should be explored. For instance, heatmaps, currently used to represent spatial distribution, could also model gaze behavior in a setting like an art gallery, indicating the focus of attention of AI agents toward different artworks. Comparing the simulated gaze heatmap with real visitor behavior data could provide insightful evaluation metrics, demonstrating the framework's versatility.

The recommendations proposed in this roadmap aim to offer a systematic, practical approach to understanding and enhancing the Agora framework's core principles: scalability, modularity, usability, and versatility. By conducting these investigations and implementing the findings, it is envisioned that the Agora framework will demonstrate improved performance across a wider range of scenarios and become increasingly accessible to users from various domains. Furthermore, these assessments will aid in identifying opportunities to extend the framework's

capabilities beyond its current remit, potentially paving the way for innovations in the broader field of crowd simulation and analysis.

8.2 Adding Social Behaviors

Addressing the limitation related to the lack of more direct social behaviors within the Agora framework could involve integrating additional behavioral models and examining their interactions with the existing heatmap-based approach. One potential direction for a new case study might include the introduction of grouping behavior in the simulation.

Using the heatmap paradigm in Agora, this can be accomplished by assigning each agent to a group and creating a dynamic heatmap that emphasizes the locations of other group members for each agent. By incorporating this heatmap into the plan adaptation pipeline and combining it with other heatmaps through color operations, locations occupied by other group members become more attractive for the virtual agent, prompting them to move toward their companions. However, it is crucial to acknowledge that this method may not suffice in preserving group formations, as agent behavior is influenced by various stimuli and their perception is limited, possibly preventing them from detecting companions who are too distant.

8.3 Perceptual Studies

To better comprehend the impact of more direct social behaviors on agent reception, a suggested perceptual study design was created but not executed (due in part to covid-19). The experiment was intended to investigate the influence of virtual humans on participants' experience in the urban environment of Section 6.2.1. Employing virtual reality technology, participants would be fully immersed in the simulated environment, allowing them to get a first-hand experience of being in the city with the virtual agents.

The rationale for this study is threefold: first, it would evaluate the quality of the social urban environment created within the Agora framework, built on top of the work of Hafsteinsson [61]. Second, it would assess the effectiveness of the implemented behaviors, such as thermal and density comfort theories, in conjunction with more direct social behaviors like grouping. Finally, it would measure the overall perception and interaction of the agents, encompassing their appearance, movement, and responsiveness.

8.4 Extensions to the Theoretical Framework

A more advanced selection model for heatmap attractiveness can be designed to better account for zones of high value and deliver a more realistic representation of people's preferences when choosing the most attractive location. This can be achieved by implementing a clustering approach that identifies areas of contiguous high-value pixels and considers the centroid of these clusters as the most appealing location.

One potential clustering method to achieve this is the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm [41], which groups together spatially close pixels based on a density criterion. Using DBSCAN, areas of contiguous high-value pixels can be identified as clusters, and the centroids of these clusters can be calculated by averaging the coordinates of the pixels within each cluster.

For example, Figure 8.1 illustrates a magnified section of the air temperature map in case study two, emphasizing a specific spot. Using the current mechanism for selecting the attractiveness of the heatmap, a "single selection" process would identify the left heatmap cell as the most appealing location. However, there are other cells with the same value in the vicinity, which could be equally suitable choices. By employing the DBSCAN clustering approach, these promising locations can be grouped, and the centroid of the cluster can be returned as the most attractive location.

The selection process can be refined further by considering additional factors, such as the size or area of the clusters. By prioritizing clusters with larger areas, the model can account for the fact that people tend to prefer areas where a desirable phenomenon is more widespread. In this case, the model would return the centroid of the largest cluster as the most attractive location.

8.5 Extensions to the Implementation

Heatmap-based Model Calibration. A promising direction for future research is the adaptation of existing calibration techniques to the heatmap paradigm and their integration into the Agora framework. The calibration process typically involves defining a distance metric to compare the output data from a simulation model with reference data, which can be sourced from real-world observations or other validated models. Through iterative parameter variation and comparison of the simulation output with reference data using the chosen metric, model parameters can be calibrated until convergence is achieved based on a predefined threshold.

The approach proposed by Wolinski et al. [149] serves as a compelling starting point, as it applies the calibration process using a variety of distance metrics and optimization approaches. Building upon this work, a novel calibration method that leverages the advantages of encoding model parameters as heatmaps and using image similarity metrics can be developed. Specifically, it would be possible to employ the image similarity metrics previously defined (Section 5.6.2) in the evaluation component as the distance metrics, rather than those employed in the cited work.

A major advantage of this approach is the potential for utilizing GPU parallelization, which can significantly accelerate the calibration process. This is especially important when dealing with large-scale simulations and complex models that require substantial computational resources. By combining heatmap-based parameter encoding, image similarity metrics, and GPU parallelization, the calibration process can be enhanced, and the overall performance of crowd simulation models can be improved.

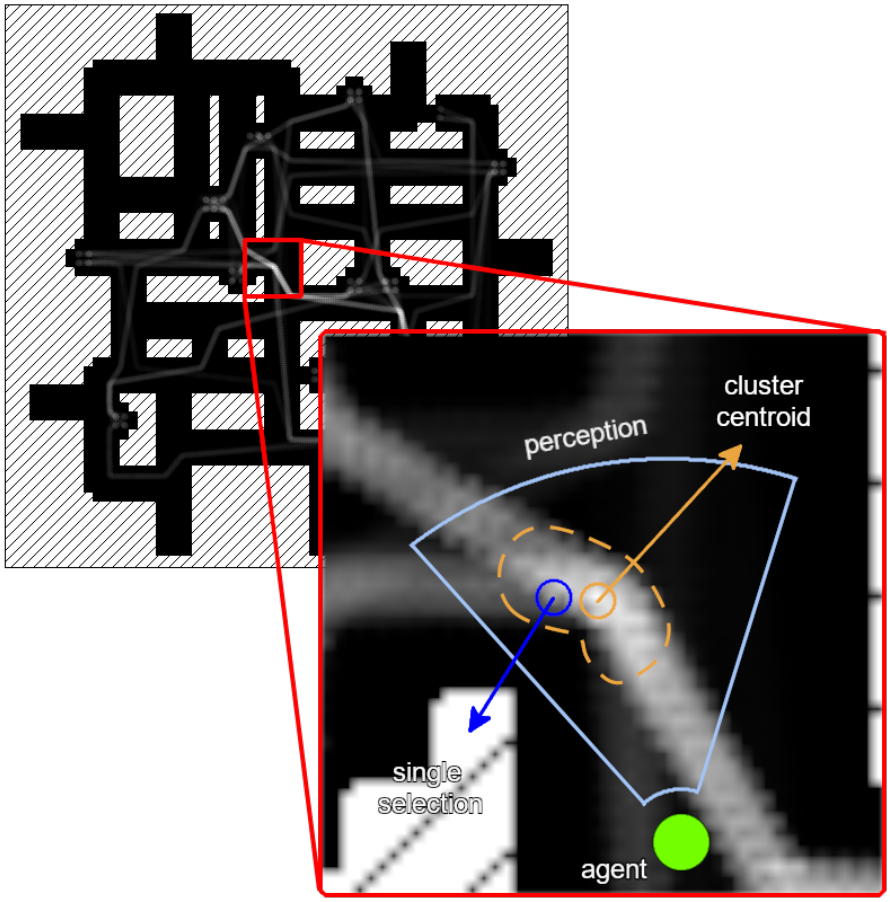

Figure 8.1: Comparison of single selection (blue) and clustering approach (orange) in a magnified air temperature map from case study two.

Heatmap-based Trajectory Adaptation. A possible approach to address the challenge of heatmap-based trajectory adaptation (see Section 7.3.2) involves creating an interface that extracts complete trajectories from the velocity components. For components that already store a complete trajectory, such as the navmesh, this process would be straightforward and involve simply returning the existing trajectory. For other components, iteratively simulating agent movement according to the velocity component function and creating a trajectory from the resulting waypoints could be a viable solution.

For example, in the case of the velocity field component, which only offers an instantaneous velocity at each step of the simulation and indirectly accounts for the global environment, the trajectory of an agent can be calculated by integrating the velocity vectors over time, as shown in Figure 8.2. By generalizing this approach

through an interface that can be applied to any velocity component, modifying the trajectories based on heatmaps becomes more feasible. This would involve iterating through the waypoints and adjusting them according to the heatmap information, as previously explained.

Implementing this heatmap trajectory adaptation approach could enhance the ability of the Agora framework to account for global influences and provide more accurate, realistic agent-based simulations.

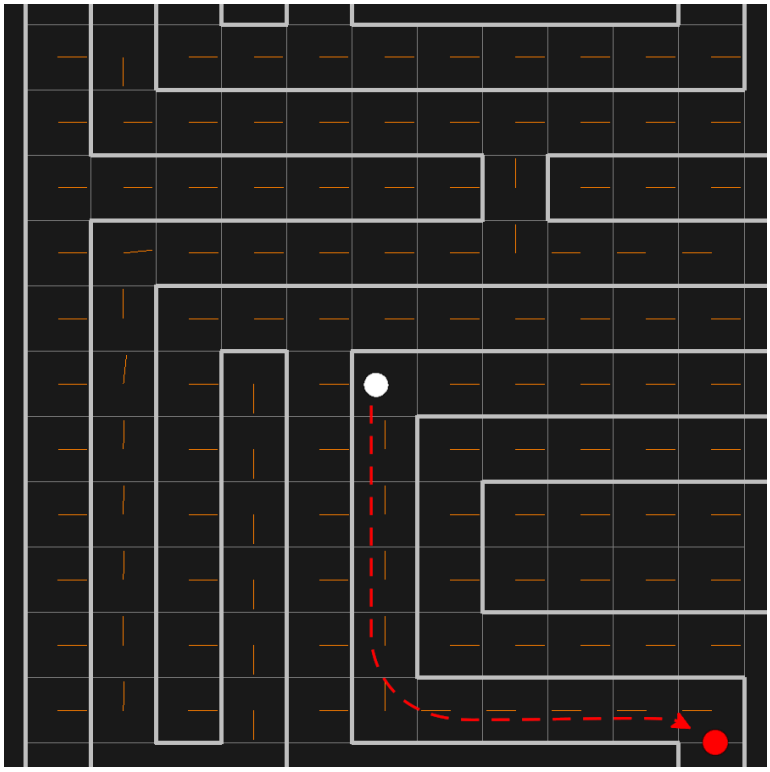

Figure 8.2: Example of Menge vector field velocity component. The orange bars are the instantaneous velocity vectors for each grid cell. It is possible to calculate the trajectory of an agent by integrating the velocity vectors over time.

Agent State Visualization. In order to expand the scope of Agora and further improve its capabilities in simulating complex and nuanced agent behaviors, a potential approach could involve leveraging Unity's rendering and visualization capabilities to build upon the existing groundwork laid by Menge's BFSM states. This would allow for a more comprehensive visualization of agent behaviors and actions, broadening the scope beyond navigation.

One possible way to achieve this is by taking inspiration from the icon-based visualization used in one of the example scenarios in the original Menge paper [32]. A generalized icon system could be developed within Agora and Unity to

represent actions and internal states more effectively, making it easier to visualize agent behaviors across various scenarios.

For a more advanced solution, Agora could integrate a behavior realizer such as SmartBody [74] to handle the visualization of agent behaviors. SmartBody is designed around the Behavior Markup Language (BML) standard [143], which enables the definition of how behaviors are realized and visualized. As a realizer, SmartBody consumes these BML specifications and acts out the behaviors, providing a more generalized approach to representing complex agent interactions.

SmartBody supports a wide range of BML behaviors, including body postures, full or partial body animations, gaze, head motions, facial expressions, speech, event or progress notifications, and the interruptions of prior behaviors. By integrating a behavior realizer like SmartBody within Agora, the resulting simulation would become more engaging, detailed, and better suited for close exploration by users.

Implementing such visualization enhancements in Agora would provide a more accurate and immersive representation of crowd behavior, broadening its capabilities beyond navigation and allowing for a deeper understanding of complex and nuanced agent interactions.

8.6 Improving Menge Integration

Exception Handling. To enhance Agora's stability, it would be beneficial to develop an exception handling mechanism that bridges the gap between unmanaged code (Menge's C++ DLL) and managed code (Unity's C#/.NET). This would involve a thorough examination and modification of Menge's codebase to ensure that the C-API interface effectively catches the underlying exceptions raised within the simulator.

One possible approach for achieving this is to implement a comprehensive error handling system that returns error codes alongside any operation results. These error codes would be checked by the managed code side, allowing Unity to recognize when an exception has been thrown in the Menge simulator. This would enable Unity to handle exceptions gracefully, without crashing the editor, and ultimately lead to a more robust and stable Agora framework.

While this process may be challenging due to the size of Menge's codebase, investing the time and effort into refining the exception-handling mechanism between Menge and Unity would significantly improve Agora's stability and reliability, making it more user-friendly and better suited for a variety of applications.

Event System. In order to foster a tighter integration between Menge and Unity within the Agora framework, it would be advantageous to develop a more powerful event system. The current approach relies on a single callback defined in the simulator wrapper instance in C# and executed during the simulation step inside the native DLL in Menge. Although this is functional for certain purposes, such as when an agent changes state, there is room for improvement in the way events are handled.

A potential approach would be to design a more flexible simulation pipeline in Menge, featuring a series of callback functions placed at key points of the simulation

process. This would enable the system to respond to various events occurring within the Menge simulator. In order to achieve this, the C-API could be extended to facilitate the registration of arbitrary callbacks that would be executed at the designated points throughout the simulation. This would make the simulation process between Menge and Unity more cohesive and streamlined.

Programmatic Modeling. An intriguing direction for making Agora more interactive would be to further extend Menge to enable programmatic creation and editing of simulation components during runtime. As previously mentioned, Menge is primarily a configuration-centric framework, with all elements being instantiated at initialization time through the parsing of configuration files for the scene and behavior. This can be limiting, as there may be cases where new elements are required during the unfolding of the simulation.

An example of this constraint is the requirement to define agents and goals at the design stage, which may not be suitable for dynamic simulations. A more practical approach would be to enable the dynamic creation of these simulation elements during runtime. This dissertation has made progress in addressing this issue for agents and goals, significantly enhancing Menge's capabilities.

Building upon these improvements, it would be beneficial to further extend the framework to allow for dynamic instantiation and modification of other simulation components, such as elements of the BFSM. This could involve programmatically creating new states in response to events within the simulation or editing existing transitions based on changes in the environment. To achieve this, Menge's original codebase would need to be extended similarly to the modifications made for agents and goals. Additionally, relevant functions would need to be exposed through the C-API, enabling them to be called from external sources, such as the Agora Unity side.

By enabling programmatic interaction with the simulation during runtime, Agora would become a more flexible and adaptable framework, better suited for a variety of dynamic and evolving scenarios.

Data-Driven Crowd Simulation. The potential use of the Agora framework, and the heatmap paradigm in particular, to support data-driven behavioral modeling in crowd simulations presents a promising future research direction. Consider a researcher aiming to investigate or replicate the underlying stimuli that may influence a specific real-world scenario. It could be possible to collect relevant data, such as pedestrian trajectories, and encode these into a heatmap for each individual. This information could serve as the foundational input for a data-driven approach.

One proposed application of this approach would be integrating a machine learning module into Agora. For instance, a neural network could be trained to automatically generate the guiding heatmaps that inform agent behavior. This would establish a dynamic, iterative learning loop that operates as follows:

1. The neural network produces a series of guiding heatmaps that dictate agent behavior.

2. A simulation is conducted using these heatmaps, running for a predetermined number of simulation steps.

3. The simulation output, converted into a heatmap format, is compared to the heatmap representation of the actual real-world phenomenon. This comparison is facilitated by Agora's image similarity evaluator.

4. The process is repeated, with the neural network updating its heatmap generation based on feedback from the image similarity evaluation. This iterative learning process continues until the output heatmaps converge with the real-world heatmaps.

Regarding the machine learning aspect, this approach could benefit from the use of reinforcement learning algorithms. These algorithms are especially adept at managing situations where decisions made at one point in time influence future states, which aligns well with crowd movement dynamics. For instance, the heatmap generation process could be framed as a policy learning problem, with the reward being the similarity between the simulated and real-world heatmap.

A realistic implementation could involve using a convolutional neural network due to its effectiveness in handling image data. The CNN could be trained to generate heatmaps from a combination of the initial state of the crowd and the current set of guiding heatmaps. This approach would necessitate a sufficiently large dataset of crowd movements and behaviors, as well as computational resources for training the neural network.

The integration of machine learning with the Agora framework could facilitate more sophisticated, data-driven modeling of crowd behavior, enhancing the realism and applicability of crowd simulations in future research.

Assets Conditioning Tools. To enhance the usability of the Agora framework, it would be advantageous to integrate the necessary authoring tools for creating assets required by Menge's velocity components. Many of these components, which deal with global path planning (such as roadmap, navmesh, and velocity field), rely on an underlying representation of the environment in a custom Menge format. Providing utilities within Agora that streamline the conversion process between 3rd party formats and Menge formats would greatly improve the user experience.

Some of these utilities have already been developed by Sean Curtis, the creator of Menge, and are available in a repository[1]. These tools are written in Python, so to incorporate them into Agora, they would need to be converted to C# or integrated into Unity through another method. Additionally, creating user-friendly graphical interfaces for these utilities would make them more accessible and easier to use.

By integrating authoring utilities within Agora, the framework would become more convenient and efficient for users, ultimately simplifying the process of creating assets and setting up simulations.

[1] https://github.com/curds01/MengeUtils/

Dissemination. To effectively disseminate Agora and promote its use in the field of crowd simulation, it is crucial to make the framework accessible and user-friendly for a wider audience. This can be achieved through a combination of appropriate packaging, distribution, documentation, and conducting a usability study.

First, Agora should be packaged in a convenient manner that takes into account its various components (Unity Assets, Menge) and dependencies (UMA). By creating an organized package structure, users will find it easier to navigate and integrate the framework into their projects.

Second, distributing Agora through a platform such as GitHub would enable users to easily access and download the framework. By making Agora available as a GitHub repository, users can add it to any Unity project as a custom git URL through the package manager. This streamlined distribution process would further encourage the adoption of the framework.

Comprehensive documentation and demo tutorials are essential in guiding users through the process of integrating Agora and building with it. Clear instructions and examples can help users understand the framework's capabilities and how to utilize them effectively in their projects.

In addition to these dissemination strategies, conducting a usability study would be a valuable step in further enhancing the Agora framework. By gathering feedback from users who have tried or integrated the framework into their projects, developers can gain insights into its strengths, areas for improvement, and potential new features. A usability study would involve a diverse group of participants from various backgrounds and expertise levels, ensuring a wide range of perspectives on the framework's performance and user experience.

This feedback can then be used to inform future development of the framework, addressing any limitations and refining the overall user experience.

Chapter 9

Conclusion

9.1 Supported Claims

Human behavior is influenced by a multitude of spatial stimuli that can be effectively represented using heatmaps. This paradigm enables the combination of various stimuli to create a more comprehensive behavioral model, accounting for the numerous factors at play. The resulting model output, in the form of heatmaps, can be objectively compared with real-world counterparts utilizing well-defined image similarity metrics for evaluation.

The Agora framework showcases its strength by facilitating the development of simulation models that achieve comparable quality to those implemented from scratch, while significantly simplifying the process. The features provided by the Agora framework streamline the development of simulation models, making them more accessible and efficient without sacrificing quality. This balance between ease of use and maintained quality is a key advantage of the Agora framework.

The Agora framework serves as a robust foundation to address the numerous challenges present within the field of crowd simulation. Agora's modular software architecture and encapsulation of behaviors through heatmaps ultimately facilitate the seamless sharing of behavior models within the research community. This should promote collaboration and accelerate advancements in the field, driving progress toward more realistic and nuanced simulations.

9.2 Contributions

This thesis has made contributions to various fields of study, as outlined below:

Crowd Simulation: The research introduces a novel approach for modeling, combining, and evaluating agent behaviors in crowd simulations. Centered around the heatmap paradigm, the approach enables the encoding and encapsulation of individual stimuli that affect human behavior into distinct layers. These layers

can then be combined to create more holistic models, and the heatmaps can be compared for model evaluation. This innovative method offers a new perspective on behavior modeling and combination in crowd simulations.

Systems Engineering: The thesis presents a robust framework that adheres to well-established software architecture principles. The framework effectively decomposes the various challenges of crowd simulation and addresses each component separately while maintaining a cohesive integration to support the entire simulation pipeline. Additionally, the implementation of the heatmap interface allows for leveraging the parallel computation capabilities of GPUs, enhancing performance and scalability.

Human-Computer Interface: This research contributes to the field of human-computer interface by introducing an intuitive way of visualizing and authoring agent behavior through a visual programming-based approach. The node-based user interface facilitates the editing and visualization of behavioral finite state machines, simplifying the process of leveraging the implemented systems to influence agent behavior. This method further refines the heatmap concept, enabling the dynamic simulation of heatmaps encoding behavioral stimuli, which can be directly integrated into an agent's decision-making process. Furthermore, the thesis presents an innovative device for field studies that collects walking trajectories and additional dyadic data, demonstrated in the Þingvellir field study.

Human Behavior Modeling: The research presents a novel model of human behavior based on the attraction to the most visible locations in an environment. It introduces a method for performing space-syntax visibility analysis of 3D environments in Unity and modeling agent behavior influenced by the most visible locations in their surroundings. This model can be used to simulate tourists walking in popular destinations. The simulation output was compared against real-world data, demonstrating the effectiveness of the model and providing valuable insights into human behavior modeling.

9.3 Limitations and Challenges

Not a Universal Solution. While the proposed approach of utilizing heatmaps to nudge agents towards appropriate actions offers several advantages, it also presents certain limitations that may render it unsuitable for specific applications. This method represents a shift from the "puppeteer" paradigm, which grants authors more control over agent behaviors, to a more stimulus-centric approach where emergent behaviors arise from various stimuli combinations.

However, in contexts where granular control over agents' actions is crucial, the flexibility provided by this approach may be outweighed by the need for consistent and reliable behaviors. Consequently, the stimulus-centric approach may not be the ideal choice for every scenario, as it may not always guarantee the desired outcomes.

Moreover, while heatmaps excel at modeling dense phenomena, such as scalar values in the spatial domain, they may struggle to represent sparse phenomena. This limitation can hinder the method's effectiveness in scenarios that involve sparse data. Therefore, alternative modeling approaches that efficiently handle such data may be more appropriate in these situations.

Only one Dimension of Human Behavior. The Agora framework primarily emphasizes the movement aspect of crowd simulation, effectively modeling the influence of diverse stimuli on agents' behaviors and simulating their responses as they navigate the environment. Nevertheless, this focus on movement captures merely a single dimension of the complex array of behavioral responses that individuals can exhibit based on internal reasoning and external stimuli. Consequently, the framework does not sufficiently address the full spectrum of human behavior, particularly in more intricate and nuanced situations. Developing a generalized approach applicable to a wide range of scenarios is inherently challenging due to the vast set of actions real people can perform, and remains an open problem in the field of crowd simulation.

9.4 A Step Towards the Protopia

This thesis has presented a novel approach to crowd simulation, enabling the combination of human behavior models through the innovative use of heatmaps. The central aspect of this approach, the heatmap paradigm, shifts the focus from a rule-centric formulation to a stimulus-centric one. Heatmaps effectively encode the various stimuli that factor into human reasoning processes and can be employed to influence human behavior. This transition from behavioral rules to heatmap-encoded stimuli allows for the effective integration of different driving factors of human behavior, enabling the creation of more comprehensive models. Furthermore, the use of heatmaps also provides a means for objectively evaluating these models, ensuring their accuracy and reliability.

Although this research has demonstrated the effectiveness of the proposed approach through an easy-to-use and highly performant software solution and two case studies, it merely scratched the surface of the potential applications and implications of this new paradigm for modeling human behavior. Exploring different ways to utilize heatmaps for influencing human behavior, applying the approach to diverse scenarios, and creating more expressive agents that can take advantage of the powerful underlying behavioral system all present a range of exciting future research directions.

Bibliography

[1] Ergonomics of the thermal environment — Instruments for measuring physical quantities. Standard, International Organization for Standardization, Geneva, November 1998. 133

[2] MakeHuman, 2000. URL http://www.makehumancommunity.org/. 21

[3] Unity Multipurpose Avatar - GitHub, 2013. URL https://github.com/umasteeringgroup/UMA. 21, 23

[4] Reallusion Character Creator, 2015. URL https://www.reallusion.com/character-creator/default.html. 21, 22

[5] MetaHuman Creator: Fast, High-Fidelity Digital Humans in Unreal Engine, 2021. 21, 22

[6] Zeyad Abd Algfoor, Mohd Shahrizal Sunar, and Hoshang Kolivand. A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games. *International Journal of Computer Games Technology*, 2015:1–11, 2015. ISSN 1687-7047, 1687-7055. https://doi.org/10.1155/2015/736138. URL http://www.hindawi.com/journals/ijcgt/2015/736138/. 27, 28

[7] Mohamed A. Ahmed and Talal M. Alkhamis. Simulation-based optimization using simulated annealing with ranking and selection. *Computers & Operations Research*, 29(4):387–402, April 2002. ISSN 03050548. https://doi.org/10.1016/S0305-0548(00)00073-3. URL https://linkinghub.elsevier.com/retrieve/pii/S0305054800000733. 33, 34

[8] Thomas Allen, Aleksandar Parvanov, Sam Knight, and Steve Maddock. Using Sketching to Control Heterogeneous Groups. *Computer Graphics and Visual Computing (CGVC)*, page 5 pages, 2015. https://doi.org/10.2312/CGVC.20151249. URL https://diglib.eg.org/handle/10.2312/cgvc20151249. Artwork Size: 5 pages ISBN: 9783905674941 Publisher: The Eurographics Association. 45, 47

[9] Mahmoud H Alrefaei and Sigrún Andradóttir. A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management science*, 45(5):748–764, 1999. Publisher: INFORMS. 33, 34

[10] Ronald C. Arkin. Path Planning For A Vision-Based Autonomous Robot. In *Mobile Robots I*, volume 0727, pages 240–250. International Society for Optics and Photonics, February 1987. https://doi.org/10.1117/12.937802. URL https://www.spiedigitallibrary.org/conference-proceedings-of-spie/0727/0000/Path-Planning-For-A-Vision-Based-Autonomous-Robot/10.1117/12.937802.short. 29

[11] Farhad Azadivar. SIMULATION OPTIMIZATION METHODOLOGIES. *Proceedings of the 31st conference on Winter simulation: Simulation a bridge to the future-Volume 1*, page 8, 1999. 33, 34

[12] O Burchan Bayazit, Jyh-Ming Lien, and Nancy M Amato. Better Group Behaviors in Complex Environments using Global Roadmaps. page 11, 2002. 26

[13] A. Beacco, N. Pelechano, and C. Andújar. A Survey of Real-Time Crowd Rendering, December 2016. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12774. 16, 17

[14] Michael Belz, Lennart W. Pyritz, and Margarete Boos. Spontaneous flocking in human groups. *Behavioural Processes*, 92:6–14, January 2013. ISSN 03766357. https://doi.org/10.1016/j.beproc.2012.09.004. URL https://linkinghub.elsevier.com/retrieve/pii/S0376635712001921. 12, 14

[15] Glen Berseth, Mubbasir Kapadia, and Petros Faloutsos. ACCLMesh: curvature-based navigation mesh generation. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, MIG '15, pages 97–102, Paris, France, November 2015. Association for Computing Machinery. ISBN 978-1-4503-3991-9. https://doi.org/10.1145/2822013.2822043. URL https://doi.org/10.1145/2822013.2822043. 30

[16] Victor J Blue and Jeffrey L Adler. Emergent fundamental pedestrian flows from cellular automata microsimulation. *Transportation Research Record*, 1644(1):29–36, 1998. Publisher: SAGE Publications Sage CA: Los Angeles, CA. 28, 30

[17] Victor J. Blue and Jeffrey L. Adler. Cellular Automata Microsimulation of Bidirectional Pedestrian Flows. *Transportation Research Record*, 1678(1):135–141, January 1999. ISSN 0361-1981. https://doi.org/10.3141/1678-17. URL https://doi.org/10.3141/1678-17. Publisher: SAGE Publications Inc. 28, 30

[18] Anton Bogdanovych, Juan Antonio Rodríguez, Simeon Simoff, A. Cohen, and Carles Sierra. Developing Virtual Heritage Applications as Normative Multiagent Systems. In Marie-Pierre Gleizes and Jorge J. Gomez-Sanz, editors, *Agent-Oriented Software Engineering X*, volume 6038, pages 140–154, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-19207-4. https://doi.org/10.1007/978-3-642-19208-1_10. URL http://link.springer.com/10.1007/978-3-642-19208-1_10. 16

[19] Leyde Briceno and Gunther Paul. MakeHuman: A Review of the Modelling Framework. In Sebastiano Bagnara, Riccardo Tartaglia, Sara Albolino, Thomas Alexander, and Yushi Fujita, editors, *Proceedings of the 20th Congress of the International Ergonomics Association (IEA 2018)*, volume 822, pages 224–232. Springer International Publishing, Cham, 2019. ISBN 978-3-319-96076-0 978-3-319-96077-7. https://doi.org/10.1007/978-3-319-96077-7_23. URL http://link.springer.com/10.1007/978-3-319-96077-7_23. Series Title: Advances in Intelligent Systems and Computing. 21, 22

[20] A.A. Bulgak and J.L. Sanders. Integrating a modified simulated annealing algorithm with the simulation of a manufacturing system to optimize buffer sizes in automatic assembly systems. In *1988 Winter Simulation Conference Proceedings*, pages 684–690, December 1988. https://doi.org/10.1109/WSC.1988.716241. ISSN: null. 33, 34

[21] C Burstedde, K Klauck, A Schadschneider, and J Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295(3):507–525, June 2001. ISSN 0378-4371. https://doi.org/10.1016/S0378-4371(01)00141-8. URL http://www.sciencedirect.com/science/article/pii/S0378437101001418. 28, 30

[22] Vinicius Jurinic Cassol, Jovani Oliveira Brasil, Amyr B. Fortes Neto, Adriana Braun, and Soraia Raupp Musse. An Approach to Validate Crowd Simulation Software: A Case Study on CrowdSim. In *2015 14th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 192–203, November 2015. https://doi.org/10.1109/SBGames.2015.11. ISSN: 2159-6662. 38

[23] L. Chen, C. R. Jung, S. R. Musse, M. Moneimne, C. Wang, R. Fruchter, V. Bazjanac, G. Chen, and N. I. Badler. Crowd Simulation Incorporating Thermal Environments and Responsive Behaviors. *Presence: Teleoperators and Virtual Environments*, 26(4):436–452, November 2017. ISSN 1531-3263. https://doi.org/10.1162/PRES_a_00308. URL https://direct.mit.edu/pvar/article/26/4/436/92674/Crowd-Simulation-Incorporating-Thermal. 130, 132, 133, 134, 139, 149, 156

[24] Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 491–502, Baltimore, Maryland, June 2005. Association for Computing Machinery. ISBN 978-1-59593-060-6. https://doi.org/10.1145/1066157.1066213. URL https://doi.org/10.1145/1066157.1066213. 33, 35, 36, 37

[25] Aleena Chia. The metaverse, but not the way you think: game engines and automation beyond game development. *Critical Studies in Me-*

dia Communication, 39(3):191–200, May 2022. ISSN 1529-5036, 1479-5809. https://doi.org/10.1080/15295036.2022.2080850. URL https://www.tandfonline.com/doi/full/10.1080/15295036.2022.2080850. 21, 22

[26] N. Chooramun, P.J. Lawrence, and E.R. Galea. Urban Scale Evacuation Simulation Using buildingEXODUS. In Richard D. Peacock, Erica D. Kuligowski, and Jason D. Averill, editors, *Pedestrian and Evacuation Dynamics*, pages 449–458. Springer US, Boston, MA, 2016. ISBN 978-1-4419-9724-1. https://doi.org/10.1007/978-1-4419-9725-8_40. URL http://link.springer.com/10.1007/978-1-4419-9725-8_40. 17

[27] Pablo de Heras Ciechomski, Sébastien Schertenleib, Jonathan Maïm, and Daniel Thalmann. Reviving the Roman Odeon of Aphrodisias: Dynamic Animation and Variety Control of Crowds in Virtual Heritage. Technical report, EPFL, VRLab, CH-1015, Lausanne, Switzerland, 2005. 16

[28] A. Colas, W. van Toll, K. Zibrek, L. Hoyet, A.-H. Olivier, and J. Pettré. Interaction Fields: Intuitive Sketch-based Steering Behaviors for Crowd Simulation. *Computer Graphics Forum*, 41(2):521–534, 2022. ISSN 1467-8659. https://doi.org/10.1111/cgf.14491. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14491. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14491. 47

[29] Francois Cournoyer. Massive Crowd on Assassin's Creed Unity: AI Recycling, 2015. URL https://www.gdcvault.com/play/1022141/Massive-Crowd-on-Assassin-s. 19, 20

[30] Xiao Cui and Hao Shi. A*-based pathfinding in modern computer games. *International Journal of Computer Science and Network Security*, 11(1):125–130, 2011. Publisher: International Journal of Computer Science and Network Security (IJCSNS). 29

[31] Xiao Cui and Hao Shi. An Overview of Pathfinding in Navigation Mesh. *IJCSNS*, 12(12):4, 2012. 28, 29

[32] Sean Curtis, Andrew Best, and Dinesh Manocha. Menge: A Modular Framework for Simulating Crowd Movement. *Collective Dynamics*, 1:1–40, March 2016. ISSN 2366-8539. https://doi.org/10.17815/CD.2016.1. URL https://collective-dynamics.eu/index.php/cod/article/view/A1. 39, 40, 43, 44, 49, 86, 91, 150, 178

[33] Michelangelo Diamanti and Hannes Högni Vilhjálmsson. Social Crowd Simulation: The Challenge of Fragmentation. In *2021 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 145–149, November 2021. https://doi.org/10.1109/AIVR52153.2021.00034. 5

[34] Michelangelo Diamanti and Hannes Högni Vilhjálmsson. Extending the menge crowd simulation framework: visual authoring in unity. In *Proceedings of the 22nd ACM International Conference on Intelligent Virtual Agents*,

pages 1–3, Faro Portugal, September 2022. ACM. ISBN 978-1-4503-9248-8. https://doi.org/10.1145/3514197.3549698. URL https://dl.acm.org/doi/10.1145/3514197.3549698. 5

[35] Ning Ding and Chang Sun. Experimental study of leader-and-follower behaviours during emergency evacuation. *Fire Safety Journal*, 117:103189, October 2020. ISSN 03797112. https://doi.org/10.1016/j.firesaf.2020.103189. URL https://linkinghub.elsevier.com/retrieve/pii/S0379711219305521. 12, 14

[36] Simon Dobbyn, John Hamill, Keith O'Conor, and Carol O'Sullivan. Geopostors: A Real-Time Geometry / Impostor Crowd Rendering System. 19

[37] Richard Dosselmann and Xue Dong Yang. A comprehensive assessment of the structural similarity index. *Signal, Image and Video Processing*, 5 (1):81–91, March 2011. ISSN 1863-1703, 1863-1711. https://doi.org/10.1007/s11760-009-0144-1. URL http://link.springer.com/10.1007/s11760-009-0144-1. 67

[38] Johannes Erfurt, Christian R. Helmrich, Sebastian Bosse, Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. A Study of the Perceptually Weighted Peak Signal-To-Noise Ratio (WPSNR) for Image Compression. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2339–2343, September 2019. https://doi.org/10.1109/ICIP.2019.8803307. ISSN: 2381-8549. 67

[39] Michael A. Erskine, Mohammed Khojah, and Alex E. McDaniel. Location selection using heat maps: Relative advantage, task-technology fit, and decision-making performance. *Computers in Human Behavior*, 101: 151–162, December 2019. ISSN 07475632. https://doi.org/10.1016/j.chb.2019.07.014. URL https://linkinghub.elsevier.com/retrieve/pii/S0747563219302572. 53

[40] Victor Erukhimov. Avatarsdk- Create avatar for Metaverse, 2017. URL https://avatarsdk.com/. 21, 24

[41] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. Density-based spatial clustering of applications with noise. In *Int. Conf. knowledge discovery and data mining*, volume 240, 1996. Issue: 6. 176

[42] Zhixin Fang, Libai Cai, and Gang Wang. MetaHuman Creator The starting point of the metaverse. In *2021 International Symposium on Computer Technology and Information Science (ISCTIS)*, pages 154–157, June 2021. https://doi.org/10.1109/ISCTIS51085.2021.00040. 21, 22

[43] Poul O Fanger and others. Thermal comfort. Analysis and applications in environmental engineering. *Thermal comfort. Analysis and applications in environmental engineering.*, 1970. Publisher: Copenhagen: Danish Technical Press. 133

[44] Jolyon J. Faria, Stefan Krause, and Jens Krause. Collective behavior in road crossing pedestrians: the role of social information. *Behavioral Ecology*, 21(6):1236–1242, 2010. ISSN 1465-7279, 1045-2249. https://doi.org/10.1093/beheco/arq141. URL https://academic.oup.com/beheco/article-lookup/doi/10.1093/beheco/arq141. 12

[45] H. Faroqi and M.-S. Mesgari. AGENT-BASED CROWD SIMULATION CONSIDERING EMOTION CONTAGION FOR EMERGENCY EVACUATION PROBLEM. In *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XL-1-W5, pages 193–196. Copernicus GmbH, December 2015. https://doi.org/https://doi.org/10.5194/isprsarchives-XL-1-W5-193-2015. URL https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XL-1-W5/193/2015/. 26, 168, 170

[46] Tian Feng, Lap-Fai Yu, Sai-Kit Yeung, KangKang Yin, and Kun Zhou. Crowd-driven mid-scale layout design. *ACM Transactions on Graphics*, 35(4):1–14, July 2016. ISSN 07300301. https://doi.org/10.1145/2897824.2925894. URL http://dl.acm.org/citation.cfm?doid=2897824.2925894. 16, 17

[47] Paolo Fiorini and Zvi Shiller. Motion Planning in Dynamic Environments Using Velocity Obstacles. *The International Journal of Robotics Research*, 17 (7):760–772, July 1998. ISSN 0278-3649, 1741-3176. https://doi.org/10.1177/027836499801700706. URL http://journals.sagepub.com/doi/10.1177/027836499801700706. 30, 31

[48] Subir K. Ghosh and Partha P. Goswami. Unsolved problems in visibility graphs of points, segments, and polygons. *ACM Computing Surveys*, 46(2):1–29, November 2013. ISSN 0360-0300, 1557-7341. https://doi.org/10.1145/2543581.2543589. URL https://dl.acm.org/doi/10.1145/2543581.2543589. 28

[49] P. Glardon, R. Boulic, and D. Thalmann. PCA-based walking engine using motion capture data. In *Proceedings Computer Graphics International, 2004.*, pages 292–298, June 2004. https://doi.org/10.1109/CGI.2004.1309224. ISSN: 1530-1052. 19, 20

[50] Julio Godoy, Ioannis Karamouzas, Stephen J Guy, and Maria Gini. Adaptive Learning for Multi-Agent Navigation. *International Conference on Autonomous Agents and Multiagent Systems*, page 9, 2015. 30, 32

[51] Erving Goffman. *Behavior in Public Places; Notes on the Social Organization of Gatherings*. The Free Press, New York, NY, 1963. ISBN 978-0-02-911940-2. 9, 14

[52] Erving Goffman. *Behavior in public places*. Simon and Schuster, 2008. 11

[53] David Goldsman and Barry L Nelson. Ranking, selection and multiple comparisons in computer simulation. In *Proceedings of Winter Simulation Conference*, pages 192–199. IEEE, 1994. 33

[54] Luis Rene Montana Gonzalez and Steve Maddock. Sketching for Real-time Control of Crowd Simulations. *Computer Graphics and Visual Computing (CGVC)*, page 8 pages, 2017. https://doi.org/10.2312/CGVC.20171282. URL https://diglib.eg.org/handle/10.2312/cgvc20171282. Artwork Size: 8 pages ISBN: 9783038680505 Publisher: The Eurographics Association. 45, 47

[55] David Gosselin, Pedro V Sander, and Jason L Mitchell. Drawing a crowd. *ShaderX3 (CHARLES RIVER MEDIA)*, pages 505–517, 2005. Publisher: Citeseer. 19

[56] William Griffit and Russell Veitch. Hot and crowded: Influence of population density and temperature on interpersonal affective behavior. *Journal of Personality and Social Psychology*, 17(1):92–98, January 1971. ISSN 0022-3514. https://doi.org/10.1037/h0030458. URL https://search.ebscohost.com/login.aspx?direct=true&db=pdh&AN=1971-09845-001&site=ehost-live. Publisher: American Psychological Association. 164

[57] Qin Gu and Zhigang Deng. Generating Freestyle Group Formations in Agent-Based Crowd Simulations. *IEEE Computer Graphics and Applications*, 33 (1):20–31, January 2013. ISSN 1558-1756. https://doi.org/10.1109/MCG.2011.87. 45, 47

[58] Stephen J. Guy, Jur van den Berg, Wenxi Liu, Rynson Lau, Ming C. Lin, and Dinesh Manocha. A statistical similarity measure for aggregate crowd dynamics. *ACM Transactions on Graphics*, 31(6):1, November 2012. ISSN 07300301. https://doi.org/10.1145/2366145.2366209. URL http://dl.acm.org/citation.cfm?doid=2366145.2366209. 36, 37

[59] S Gwynne, E. R Galea, P. J Lawrence, and L Filippidis. Modelling occupant interaction with fire conditions using the buildingEXODUS evacuation model. *Fire Safety Journal*, 36(4):327–357, June 2001. ISSN 0379-7112. https://doi.org/10.1016/S0379-7112(00)00060-6. URL http://www.sciencedirect.com/science/article/pii/S0379711200000606. 17

[60] S. Gwynne, E.R. Galea, M. Owen, P.J. Lawrence, and L. Filippidis. A systematic comparison of buildingEXODUS predictions with experimental data from the Stapelfeldt trials and the Milburn House evacuation. *Applied Mathematical Modelling*, 29(9):818–851, September 2005. ISSN 0307904X. https://doi.org/10.1016/j.apm.2004.11.005. URL https://linkinghub.elsevier.com/retrieve/pii/S0307904X04001696. 16, 17

[61] Hörður Már Hafsteinsson. *Procedural city generator for research in the field of restorative environmental design.* Thesis, Reykjavík University, Reykjavík, June 2019. Accepted: 2019-06-12T13:46:02Z. 130, 131, 132, 137, 175

[62] Edmund T Hall and Edward T Hall. *The hidden dimension*, volume 609. Anchor, 1966. 8, 11

[63] Edward T Hall, Ray L Birdwhistell, Bernhard Bock, Paul Bohannan, A Richard Diebold Jr, Marshall Durbin, Munro S Edmonson, JL Fischer, Dell Hymes, Solon T Kimball, and others. Proxemics [and comments and replies]. *Current anthropology*, 9(2/3):83–108, 1968. 136

[64] Daniel Harabor and Adi Botea. Breaking Path Symmetries on 4-Connected Grid Maps. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 6(1), October 2010. ISSN 2334-0924. URL https://ojs.aaai.org/index.php/AIIDE/article/view/12393. Number: 1. 28

[65] Elaine Hatfield, John T Cacioppo, and Richard L Rapson. Emotional contagion. *Current directions in psychological science*, 2(3):96–100, 1993. Publisher: Sage Publications Sage CA: Los Angeles, CA. 13, 14

[66] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, May 1995. https://doi.org/10.1103/PhysRevE.51.4282. URL https://link.aps.org/doi/10.1103/PhysRevE.51.4282. 30, 31, 163

[67] Dirk Helbing, Illes J Farkas, Peter Molnar, and Tamas Vicsek. Simulation of Pedestrian Crowds in Normal and Evacuation Situations. *Pedestrian and evacuation dynamics*, 21(2):21–58, 2002. 16, 17

[68] Dirk Helbing, Lubos Buzna, Anders Johansson, and Torsten Werner. Self-Organized Pedestrian Crowd Dynamics: Experiments, Simulations, and Design Solutions. *Transportation Science*, 39(1):1–24, February 2005. ISSN 0041-1655, 1526-5447. https://doi.org/10.1287/trsc.1040.0108. URL http://pubsonline.informs.org/doi/abs/10.1287/trsc.1040.0108. 16, 17

[69] Michael A. Hogg and R. Scott Tindale, editors. *Group processes*. Blackwell handbook of social psychology. Blackwell Publishers, Malden, MA, 2001. ISBN 978-0-631-20865-5. 13, 14

[70] Wenjia Huang and Demetri Terzopoulos. Door and Doorway Etiquette for Virtual Humans. *IEEE Transactions on Visualization and Computer Graphics*, 26(3):1502–1517, March 2020. ISSN 1941-0506. https://doi.org/10.1109/TVCG.2018.2874050. Conference Name: IEEE Transactions on Visualization and Computer Graphics. 26, 27, 168, 171

[71] Pall Jakob Lindal. Future Cities: Using virtual technology to design restorative residential neighborhoods. In *International Conference on Environmental Psychology*, Plymouth, UK, 2019. 131

[72] Zhixing Jin and Bir Bhanu. Optimizing crowd simulation based on real video data. In *2013 IEEE International Conference on Image Processing*, pages 3186–3190, September 2013. https://doi.org/10.1109/ICIP.2013.6738656. ISSN: 2381-8549. 35

[73] Susi Juniastuti, Moch Fachri, Supeno Mardi Susiki Nugroho, and Mochammad Hariadi. Crowd navigation using leader-follower algorithm based Reciprocal Velocity Obstacles. In *2016 International Symposium on Electronics and Smart Devices (ISESD)*, pages 148–152, November 2016. https://doi.org/10.1109/ISESD.2016.7886709. ISSN: null. 26

[74] Marcelo Kallmann and Stacy Marsella. Hierarchical Motion Controllers for Real-Time Autonomous Virtual Humans. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Themis Panayiotopoulos, Jonathan Gratch, Ruth Aylett, Daniel Ballin, Patrick Olivier, and Thomas Rist, editors, *Intelligent Virtual Agents*, volume 3661, pages 253–265. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-28738-4 978-3-540-28739-1. https://doi.org/10.1007/11550617_22. URL http://link.springer.com/10.1007/11550617_22. Series Title: Lecture Notes in Computer Science. 179

[75] Mubbasir Kapadia, Kai Ninomiya, Alexander Shoulson, Francisco Garcia, and Norman Badler. Constraint-Aware Navigation in Dynamic Environments. In *Proceedings of Motion on Games - MIG '13*, pages 111–120, Dublin 2, Ireland, 2013. ACM Press. ISBN 978-1-4503-2546-2. https://doi.org/10.1145/2522628.2522654. URL http://dl.acm.org/citation.cfm?doid=2522628.2522654. 28

[76] Nahid Karimaghalou, Ulysses Bernardet, and Steve DiPaola. A model for social spatial behavior in virtual characters: A model for social spatial behavior in virtual characters. *Computer Animation and Virtual Worlds*, 25(3-4):505–517, May 2014. ISSN 15464261. https://doi.org/10.1002/cav.1600. URL http://doi.wiley.com/10.1002/cav.1600. 26, 168, 169

[77] Effie Karuzaki, Nikolaos Partarakis, Nikolaos Patsiouras, Emmanouil Zidianakis, Antonios Katzourakis, Antreas Pattakos, Danae Kaplanidi, Evangelia Baka, Nedjma Cadi, Nadia Magnenat-Thalmann, Chris Ringas, Eleana Tasiopoulou, and Xenophon Zabulis. Realistic Virtual Humans for Cultural Heritage Applications. *Heritage*, 4(4):4148–4171, November 2021. ISSN 2571-9408. https://doi.org/10.3390/heritage4040228. URL https://www.mdpi.com/2571-9408/4/4/228. 21, 22

[78] Adam Kendon. *Conducting interaction: Patterns of behavior in focused encounters*, volume 7. CUP Archive, 1990. 9, 10, 14, 54

[79] Adam Kendon. Spacing and Orientation in Co-present Interaction. In Anna Esposito, Nick Campbell, Carl Vogel, Amir Hussain, and Anton Nijholt, editors, *Development of Multimodal Interfaces: Active Listening and Synchrony*, volume 5967, pages 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-12396-2 978-3-642-12397-9. https://doi.org/10.1007/978-3-642-12397-9_1. URL http://link.springer.

com/10.1007/978-3-642-12397-9_1. Series Title: Lecture Notes in Computer Science. 10

[80] Adam Kendon, Thomas A. Sebeok, and Jean Umiker-Sebeok, editors. *Nonverbal Communication, Interaction, and Gesture*. Approaches to Semiotics. Mouton Publishers, The Hague, 1981. ISBN 90-279-3489-4. 14

[81] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, May 1983. ISSN 0036-8075, 1095-9203. https://doi.org/10.1126/science.220.4598.671. URL https://science.sciencemag.org/content/220/4598/671. Publisher: American Association for the Advancement of Science Section: Articles. 33, 34

[82] Benedikt Kleinmeier, Benedikt Zönnchen, Marion Gödel, and Gerta Köster. Vadere: An open-source simulation framework to promote interdisciplinary understanding, July 2019. URL http://arxiv.org/abs/1907.09520. arXiv:1907.09520 [cs]. 38, 39, 42, 44

[83] Kang Hoon Lee, Myung Geol Choi, Qyoun Hong, and Jehee Lee. Group behavior from video: a data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 109–118, San Diego, California, August 2007. Eurographics Association. ISBN 978-1-59593-624-0. 26

[84] Marilena Lemonari, Rafael Blanco, Panayiotis Charalambous, Nuria Pelechano, Marios Avraamides, Julien Pettré, and Yiorgos Chrysanthou. Authoring Virtual Crowds: A Survey. *Computer Graphics Forum*, 41 (2):677–701, 2022. ISSN 1467-8659. https://doi.org/10.1111/cgf.14506. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14506. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14506. 45, 48

[85] Cheng-Te Li and Shou-De Lin. Social flocks: a crowd simulation framework for social network generation, community detection, and collective behavior modeling. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 765–768, San Diego, California, USA, August 2011. Association for Computing Machinery. ISBN 978-1-4503-0813-7. https://doi.org/10.1145/2020408.2020531. URL https://doi.org/10.1145/2020408.2020531. 25, 26, 168, 169

[86] Zhong Li, Lele Chen, Celong Liu, Fuyao Zhang, Zekun Li, Yu Gao, Yuanzhou Ha, Chenliang Xu, Shuxue Quan, and Yi Xu. Animated 3D human avatars from a single image with GAN-based texture inference. *Computers & Graphics*, 95:81–91, April 2021. ISSN 00978493. https://doi.org/10.1016/j.cag.2021.01.002. URL https://linkinghub.elsevier.com/retrieve/pii/S0097849321000029. 21, 24

[87] Pall Jackob Lindal, H. Miri, K. R. Johannsdottir, T. Hartig, and Hannes Högni Vilhjalmsson. Cities that sustain us: Using virtual reality to test the restorative

potential of future urban environments. In *11th Biennial Conference on Environmental Psychology*, Groningen, The Netherlands, 2015. 131

[88] Shiguang Liu and Dinesh Manocha. Sound Synthesis, Propagation, and Rendering: A Survey, May 2021. URL `http://arxiv.org/abs/2011.05538`. arXiv:2011.05538 [cs]. 55

[89] Nadia Magnenat-Thalmann, Hyewon Seo, and Frederic Cordier. Automatic modeling of virtual humans and body clothing. *Journal of Computer Science and Technology*, 19(5):575–584, September 2004. ISSN 1000-9000, 1860-4749. `https://doi.org/10.1007/BF02945583`. URL `http://link.springer.com/10.1007/BF02945583`. 19

[90] Yan Mao, Shanwen Yang, Zuning Li, and Yongjian Li. Personality trait and group emotion contagion based crowd simulation for emergency evacuation. *Multimedia Tools and Applications*, May 2018. ISSN 1573-7721. `https://doi.org/10.1007/s11042-018-6069-3`. URL `https://doi.org/10.1007/s11042-018-6069-3`. 26, 168, 169

[91] C. D. Tharindu Mathew, Paulo R. Knob, Soraia Raupp Musse, and Daniel G. Aliaga. Urban Walkability Design Using Virtual Population Simulation. *Computer Graphics Forum*, 38(1):455–469, February 2019. ISSN 0167-7055, 1467-8659. `https://doi.org/10.1111/cgf.13585`. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13585`. 16, 17

[92] Jonathan Maïm, Simon Haegler, Barbara Yersin, Pascal Mueller, Daniel Thalmann, and Luc Van Gool. Populating Ancient Pompeii with Crowds of Virtual Romans. In *Proceedings of the 8th International Symposium on Virtual Reality, Archeology and Cultural Heritage-VAST*, page 8, 2007. 16

[93] James McIlveen, Steve Maddock, Peter Heywood, and Paul Richmond. PED: Pedestrian Environment Designer. *Computer Graphics and Visual Computing (CGVC)*, page 8 pages, 2016. ISSN -. `https://doi.org/10.2312/CGVC.20161304`. URL `https://diglib.eg.org/handle/10.2312/cgvc20161304`. Artwork Size: 8 pages ISBN: 9783038680222 Publisher: The Eurographics Association. 45, 46

[94] Daniel N McIntosh, Daniel Druckman, and Robert B Zajonc. Socially induced affect. *Learning, remembering, believing: Enhancing human performance*, pages 251–276, 1994. 13, 14

[95] Tim McLaughlin, Larry Cutler, and David Coleman. Character rigging, deformations, and simulations in film and game production. In *ACM SIGGRAPH 2011 Courses on - SIGGRAPH '11*, pages 1–18, Vancouver, British Columbia, Canada, 2011. ACM Press. ISBN 978-1-4503-0967-7. `https://doi.org/10.1145/2037636.2037641`. URL `http://dl.acm.org/citation.cfm?doid=2037636.2037641`. 53

[96] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 27

(1):415–444, August 2001. ISSN 0360-0572, 1545-2115. https://doi.org/10.1146/annurev.soc.27.1.415. URL https://www.annualreviews.org/doi/10.1146/annurev.soc.27.1.415. 12, 13, 14

[97] Mehdi Moussaïd, Niriaska Perozo, Simon Garnier, Dirk Helbing, and Guy Theraulaz. The Walking Behaviour of Pedestrian Social Groups and Its Impact on Crowd Dynamics. *PLoS ONE*, 5(4):e10047, April 2010. ISSN 1932-6203. https://doi.org/10.1371/journal.pone.0010047. URL https://dx.plos.org/10.1371/journal.pone.0010047. 38, 51

[98] Mehdi Moussaïd, Mubbasir Kapadia, Tyler Thrash, Robert W. Sumner, Markus Gross, Dirk Helbing, and Christoph Hölscher. Crowd behaviour during high-stress evacuations in an immersive virtual environment. *Journal of The Royal Society Interface*, 13(122):20160414, September 2016. ISSN 1742-5689, 1742-5662. https://doi.org/10.1098/rsif.2016.0414. URL http://rsif.royalsocietypublishing.org/lookup/doi/10.1098/rsif.2016.0414. 16, 17

[99] A. Natapov and D. Fisher-Gewirtzman. Visibility of urban activities and pedestrian routes: An experiment in a virtual environment. *Computers, Environment and Urban Systems*, 58:60–70, July 2016. ISSN 01989715. https://doi.org/10.1016/j.compenvurbsys.2016.03.007. URL https://linkinghub.elsevier.com/retrieve/pii/S0198971516300370. 122

[100] M. E. J. Newman. The Structure and Function of Complex Networks. *SIAM Review*, 45(2):167–256, January 2003. ISSN 0036-1445, 1095-7200. https://doi.org/10.1137/S003614450342480. URL http://epubs.siam.org/doi/10.1137/S003614450342480. 13

[101] C. Niederberger and M. Gross. Hierarchical and Heterogenous Reactive Agents for Real-Time Applications. *Computer Graphics Forum*, 22(3):323–331, September 2003. ISSN 0167-7055, 1467-8659. https://doi.org/10.1111/1467-8659.00679. URL http://doi.wiley.com/10.1111/1467-8659.00679. 26

[102] Stuart O'Connor, Fotis Liarokapis, and Christopher Peters. An initial study to assess the perceived realism of agent crowd behaviour in a virtual city. In *2013 5th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*, pages 1–8, Poole, September 2013. IEEE. ISBN 978-1-4799-0965-0. https://doi.org/10.1109/VS-GAMES.2013.6624220. URL http://ieeexplore.ieee.org/document/6624220/. 16, 17

[103] Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, and Stéphane Donikian. A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics (TOG)*, 29(4):123:1–123:9, July 2010. ISSN 0730-0301. https://doi.org/10.1145/1778765.1778860. URL https://doi.org/10.1145/1778765.1778860. 30, 31

[104] Mohd Fauzi Othman, Masoud Samadi, and Mehran Halimi Asl. Simulation of Dynamic Path Planning for Real-Time Vision-Base Robots. In Khairuddin

Omar, Md Jan Nordin, Prahlad Vadakkepat, Anton Satria Prabuwono, Siti Norul Huda Sheikh Abdullah, Jacky Baltes, Shamsudin Mohd Amin, Wan Zuha Wan Hassan, and Mohammad Faidzul Nasrudin, editors, *Intelligent Robotics Systems: Inspiring the NEXT*, volume 376, pages 1–10. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-40408-5 978-3-642-40409-2. https://doi.org/10.1007/978-3-642-40409-2_1. URL http://link.springer.com/10.1007/978-3-642-40409-2_1. Series Title: Communications in Computer and Information Science. 28

[105] Claudio Pedica and Hannes Vilhjálmsson. Social Perception and Steering for Online Avatars. In Helmut Prendinger, James Lester, and Mitsuru Ishizuka, editors, *Intelligent Virtual Agents*, volume 5208, pages 104–116. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-85482-1. https://doi.org/10.1007/978-3-540-85483-8_11. URL http://link.springer.com/10.1007/978-3-540-85483-8_11. 25, 26

[106] Claudio Pedica, Michelangelo Diamanti, and Hannes Högni Vilhjálmsson. Assessing the Disturbance from Overcrowding in Outdoor Nature Experiences. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, number 58, pages 1–8. Association for Computing Machinery, New York, NY, USA, May 2021. ISBN 978-1-4503-8095-9. URL https://doi.org/10.1145/3411763.3443439. 5, 122

[107] N. Pelechano and N. I. Badler. Modeling Crowd and Trained Leader Behavior during Building Evacuation. *IEEE Computer Graphics and Applications*, 26 (6):80–86, November 2006. ISSN 0272-1716. https://doi.org/10.1109/MCG.2006.133. 16, 17, 26

[108] Nuria Pelechano and Carlos Fuentes. Hierarchical path-finding for Navigation Meshes (HNA∗). *Computers & Graphics*, 59:68–78, October 2016. ISSN 0097-8493. https://doi.org/10.1016/j.cag.2016.05.023. URL http://www.sciencedirect.com/science/article/pii/S0097849316300668. 28, 29

[109] Nuria Pelechano, Bernhard Spanlang, and Alejandro Beacco. Avatar Locomotion in Crowd Simulation. *International Journal of Virtual Reality*, 10(1): 13–19, January 2011. ISSN 1081-1451. https://doi.org/10.20870/IJVR.2011.10.1.2796. URL https://ijvr.eu/article/view/2796. 19, 20

[110] Sebastian Pütz, Thomas Wiemann, Jochen Sprickerhof, and Joachim Hertzberg. 3D Navigation Mesh Generation for Path Planning in Uneven Terrain. *IFAC-PapersOnLine*, 49(15):212–217, January 2016. ISSN 2405-8963. https://doi.org/10.1016/j.ifacol.2016.07.734. URL http://www.sciencedirect.com/science/article/pii/S2405896316310102. 30

[111] Fasheng Qiu and Xiaolin Hu. Modeling group structures in pedestrian crowd simulation. *Simulation Modelling Practice and Theory*, 18(2):190–205, February 2010. ISSN 1569190X. https://doi.org/10.1016/j.simpat.2009.10.005. URL https://linkinghub.elsevier.com/retrieve/pii/S1569190X09001555. 26

[112] Ramsey M. Raafat, Nick Chater, and Chris Frith. Herding in humans. *Trends in Cognitive Sciences*, 13(10):420–428, October 2009. ISSN 13646613. https://doi.org/10.1016/j.tics.2009.08.002. URL https://linkinghub.elsevier.com/retrieve/pii/S1364661309001703. 12, 14

[113] Vahid Rahmani and Nuria Pelechano. Improvements to hierarchical pathfinding for navigation meshes. In *Proceedings of the Tenth International Conference on Motion in Games*, MIG '17, pages 1–6, Barcelona, Spain, November 2017. Association for Computing Machinery. ISBN 978-1-4503-5541-4. https://doi.org/10.1145/3136457.3136465. URL https://doi.org/10.1145/3136457.3136465. 28, 29

[114] Vahid Rahmani and Nuria Pelechano. Multi-agent parallel hierarchical path finding in navigation meshes (MA-HNA*). *Computers & Graphics*, 86:1–14, February 2020. ISSN 0097-8493. https://doi.org/10.1016/j.cag.2019.10.006. 28, 29

[115] Craig W Reynolds. OpenSteer. URL https://opensteer.sourceforge.net/doc.html. 39, 44

[116] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 25–34, New York, NY, USA, August 1987. Association for Computing Machinery. ISBN 978-0-89791-227-3. https://doi.org/10.1145/37401.37406. URL https://doi.org/10.1145/37401.37406. 30, 31

[117] Craig W Reynolds. Not bumping into things. *Computer Graphics*, Notes on 'obstacle avoidance'for course on physically-based modeling at SIGGRAPH (88):G1, 1988. 30, 31

[118] Craig W. Reynolds. Steering Behaviors For Autonomous Characters. *Game developers conference*, 1999:14, 1999. 30, 31

[119] Otger Rogla, Nuria Pelechano, and Gustavo A. Patow. Procedural Crowd Generation for Semantically Augmented Virtual Cities. *Computers & Graphics*, 99:83–99, 2018. 16, 17

[120] Francisco Arturo Rojas and Hyun Seung Yang. Minimizing Collision among Social Groups in Wide-Open Spaces. In *2014 International Conference on Cyberworlds*, pages 77–84, October 2014. https://doi.org/10.1109/CW.2014.19. 21, 24

[121] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. 67

[122] Albert E. Scheflen. Micro-territories in human interaction. In Adam Kendon, editor, *Organization of Behavior in Face-to-Face Interaction*, World Anthropology, pages 159–174. Mouton Publishers, 1975. ISBN 978-3-11-090764-3. 11

[123] Albert E. Scheflen. *Human Territories: how we behave in space and time*. Prentice-Hall, New York, NY, USA, 1976. ISBN 1-59593-364-6. 9, 11, 14, 54

[124] Remington Scott. Sparking life: notes on the performance capture sessions for the Lord of the Rings: the Two Towers. *ACM SIGGRAPH Computer Graphics*, 37(4):17–21, 2003. 16, 17

[125] Michael Seitz, Gerta Köster, and Alexander Pfaffinger. Pedestrian Group Behavior in a Cellular Automaton. In Ulrich Weidmann, Uwe Kirsch, and Michael Schreckenberg, editors, *Pedestrian and Evacuation Dynamics 2012*, pages 807–814. Springer International Publishing, Cham, 2014. ISBN 978-3-319-02446-2. https://doi.org/10.1007/978-3-319-02447-9_67. URL http://link.springer.com/10.1007/978-3-319-02447-9_67. 26

[126] A. Seyfried, M. Boltes, J. Kähler, W. Klingsch, A. Portz, T. Rupprecht, A. Schadschneider, B. Steffen, and A. Winkens. Enhanced empirical data for the fundamental diagram and the flow through bottlenecks. *arXiv:0810.1945 [physics]*, October 2008. URL http://arxiv.org/abs/0810.1945. arXiv: 0810.1945. 36, 38

[127] Armin Seyfried, Bernhard Steffen, Wolfram Klingsch, and Maik Boltes. The Fundamental Diagram of Pedestrian Movement Revisited. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(10):P10002–P10002, October 2005. ISSN 1742-5468. https://doi.org/10.1088/1742-5468/2005/10/P10002. URL http://arxiv.org/abs/physics/0506170. arXiv: physics/0506170. 36, 38

[128] Wei Shao and Demetri Terzopoulos. Environmental Modeling for Autonomous Virtual Pedestrians. pages 2005–01–2699, June 2005. https://doi.org/10.4271/2005-01-2699. URL https://www.sae.org/content/2005-01-2699/. 28

[129] Wei Shao and Demetri Terzopoulos. Autonomous pedestrians. *Graphical Models*, 69(5-6):246–274, September 2007. ISSN 15240703. https://doi.org/10.1016/j.gmod.2007.09.001. URL https://linkinghub.elsevier.com/retrieve/pii/S1524070307000252. 26, 27, 168, 170

[130] Alexander Shoulson, Nathan Marshak, Mubbasir Kapadia, and Norman I. Badler. ADAPT: The Agent Developmentand Prototyping Testbed. *IEEE Transactions on Visualization and Computer Graphics*, 20(7):1035–1047, July 2014. ISSN 1941-0506. https://doi.org/10.1109/TVCG.2013.251. Conference Name: IEEE Transactions on Visualization and Computer Graphics. 39, 44, 150

[131] Shawn Singh, Mubbasir Kapadia, Petros Faloutsos, and Glenn Reinman. An Open Framework for Developing, Evaluating, and Sharing Steering Algorithms. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar,

Moshe Y. Vardi, Gerhard Weikum, Arjan Egges, Roland Geraerts, and Mark Overmars, editors, *Motion in Games*, volume 5884, pages 158–169. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-10346-9 978-3-642-10347-6. https://doi.org/10.1007/978-3-642-10347-6_15. URL http://link.springer.com/10.1007/978-3-642-10347-6_15. Series Title: Lecture Notes in Computer Science. 39, 41, 44

[132] Qiyun Sun, Wanggen Wan, and Xiaoqing Yu. The simulation of building escape system based on Unity3D. In *2016 International Conference on Audio, Language and Image Processing (ICALIP)*, pages 156–160, July 2016. https://doi.org/10.1109/ICALIP.2016.7846656. 19, 20

[133] Mankyu Sung, Michael Gleicher, and Stephen Chenney. Scalable behaviors for crowd simulation. *Computer Graphics Forum*, 23(3):519–528, September 2004. ISSN 0167-7055, 1467-8659. https://doi.org/10.1111/j.1467-8659.2004.00783.x. URL http://doi.wiley.com/10.1111/j.1467-8659.2004.00783.x. 45, 47

[134] Noor Aqilah A. Tajedi, Nur Sabahiah A. Sukor, Mohd Ashraf M. Ismail, and Shahrul A. Shamsudin. Verifying the buildingEXODUS through an emergency response procedure (ERP) exercise at an underground intervention shaft. *AIP Conference Proceedings*, 1892(1):060002, October 2017. ISSN 0094-243X. https://doi.org/10.1063/1.5005713. URL https://aip.scitation.org/doi/abs/10.1063/1.5005713. Publisher: American Institute of Physics. 17

[135] Franco Tecchia, Celine Loscos, and Yiorgos Chrysanthou. Visualizing Crowds in Real-Time. *Computer Graphics Forum*, 21(4):753–765, 2002. ISSN 1467-8659. https://doi.org/10.1111/1467-8659.00633. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00633. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00633. 19

[136] Daniel Thalmann. Crowd simulation. *Wiley Encyclopedia of Computer Science and Engineering*, 2007. 15

[137] Daniel Thalmann and Soraia Raupp Musse. *Crowd simulation*. Springer, London ; New York, 2nd ed edition, 2013. ISBN 978-1-4471-4449-6. OCLC: ocn805017772. 75

[138] Daniel Thalmann, Helena Grillon, Jonathan Maim, and Barbara Yersin. Challenges in Crowd Simulation. In *2009 International Conference on CyberWorlds*, pages 1–12, Bradford, West Yorkshire, United Kingdom, 2009. IEEE. ISBN 978-1-4244-4864-7. https://doi.org/10.1109/CW.2009.23. URL http://ieeexplore.ieee.org/document/5279720/. 18, 19, 20

[139] Wouter G. van Toll, Atlas F. Cook, and Roland Geraerts. A navigation mesh for dynamic environments. *Computer Animation and Virtual Worlds*, 23(6):535–546, 2012. ISSN 1546-427X. https://doi.org/10.1002/cav.1468. 28, 30

[140] Timmu Tõke. Wolf3D – Personal 3D Avatar Creator For Games, Mobile Apps, VR/AR, 2014. URL `https://wolf3d.io/`. 21, 24

[141] Branislav Ulicny, Pablo de Heras Ciechomski, and Daniel Thalmann. Crowdbrush: Interactive Authoring of Real-time Crowd Scenes. 2004. 45, 46

[142] Jur van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal Velocity Obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 1928–1935, May 2008. `https://doi.org/10.1109/ROBOT.2008.4543489`. ISSN: 1050-4729. 30, 31, 64

[143] Hannes Vilhjálmsson, Nathan Cantelmo, Justine Cassell, Nicolas E. Chafai, Michael Kipp, Stefan Kopp, Maurizio Mancini, Stacy Marsella, Andrew N. Marshall, Catherine Pelachaud, Zsofi Ruttkay, Kristinn R. Thórisson, Herwin van Welbergen, and Rick J. van der Werf. The Behavior Markup Language: Recent Developments and Challenges. In Catherine Pelachaud, Jean-Claude Martin, Elisabeth André, Gérard Chollet, Kostas Karpouzis, and Danielle Pelé, editors, *Intelligent Virtual Agents*, volume 4722, pages 99–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-74996-7 978-3-540-74997-4. `https://doi.org/10.1007/978-3-540-74997-4_10`. URL `http://link.springer.com/10.1007/978-3-540-74997-4_10`. Series Title: Lecture Notes in Computer Science. 179

[144] Hannes Högni Vilhjálmsson. Interaction in Social Space. In Birgit Lugrin, Catherine Pelachaud, and David Traum, editors, *The Handbook on Socially Interactive Agents*, pages 3–44. ACM, New York, NY, USA, 1 edition, October 2022. ISBN 978-1-4503-9896-1. `https://doi.org/10.1145/3563659.3563662`. URL `https://dl.acm.org/doi/10.1145/3563659.3563662`. 8, 9, 11, 14

[145] Armel Ulrich Kemloh Wagoum, Mohcine Chraibi, Jun Zhang, and Gregor Lammel. JuPedSim: an open framework for simulating and analyzing the dynamics of pedestrians. 39, 41, 44

[146] He Wang, Jan Ondřej, and Carol O'Sullivan. Trending Paths: A New Semantic-Level Metric for Comparing Simulated and Real Crowd Data. *IEEE Transactions on Visualization and Computer Graphics*, 23(5):1454–1464, May 2017. ISSN 2160-9306. `https://doi.org/10.1109/TVCG.2016.2642963`. 36, 38, 67, 156

[147] Zhou Wang and Alan C. Bovik. Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures. *IEEE Signal Processing Magazine*, 26 (1):98–117, January 2009. ISSN 1558-0792. `https://doi.org/10.1109/MSP.2008.930649`. Conference Name: IEEE Signal Processing Magazine. 67

[148] Nicholas Mario Wardhana, Henry Johan, and Hock Soon Seah. Enhanced waypoint graph for surface and volumetric path planning in virtual worlds. *The Visual Computer*, 29(10):1051–1062, October 2013. ISSN 0178-2789, 1432-2315. `https://doi.org/10.1007/s00371-013-0837-x`. URL `http://link.springer.com/10.1007/s00371-013-0837-x`. 28

[149] D. Wolinski, S. J. Guy, A.-H. Olivier, M. Lin, D. Manocha, and J. Pettré. Parameter estimation and comparative evaluation of crowd simulations. *Computer Graphics Forum*, 33(2):303–312, 2014. ISSN 1467-8659. https://doi.org/10.1111/cgf.12328. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12328. 34, 35, 67, 176

[150] Yao Xiao, Ziyou Gao, Rui Jiang, Xingang Li, Yunchao Qu, and Qingxia Huang. Investigation of pedestrian dynamics in circle antipode experiments: Analysis and model evaluation with macroscopic indexes. *Transportation Research Part C: Emerging Technologies*, 103:174–193, June 2019. ISSN 0968090X. https://doi.org/10.1016/j.trc.2019.04.007. URL https://linkinghub.elsevier.com/retrieve/pii/S0968090X18312610. 36, 38

[151] Jiang Xueling. Simulation Model of Pedestrian Evacuation in High-Rise Building: Considering Group Behaviors and Real-Time Fire. *International Journal of Smart Home*, 9(2):81–92, February 2015. ISSN 19754094, 19754094. https://doi.org/10.14257/ijsh.2015.9.2.07. URL http://gvpress.com/journals/IJSH/vol9_no2/7.pdf. 26

[152] Xinjie Yao, Ji Zhang, and Jean Oh. Following Social Groups: Socially Compliant Autonomous Navigation in Dense Crowds, November 2019. URL http://arxiv.org/abs/1911.12063. arXiv:1911.12063 [cs]. 30, 32

[153] Peter Yap. Grid-Based Path-Finding. In Robin Cohen and Bruce Spencer, editors, *Advances in Artificial Intelligence*, Lecture Notes in Computer Science, pages 44–55, Berlin, Heidelberg, 2002. Springer. ISBN 978-3-540-47922-2. https://doi.org/10.1007/3-540-47922-8_4. 28

[154] Barbara Yersin, Jonathan Maım, Pablo de Heras Ciechomski, Sebastien Schertenleib, and Daniel Thalmann. Steering a Virtual Crowd Based on a Semantically Augmented Navigation Graph. 2005. 45, 46

[155] Jianfeng Zhang, Zihang Jiang, Dingdong Yang, Hongyi Xu, Yichun Shi, Guoxian Song, Zhongcong Xu, Xinchao Wang, and Jiashi Feng. AvatarGen: A 3D Generative Model for Animatable Human Avatars, November 2022. URL http://arxiv.org/abs/2211.14589. arXiv:2211.14589 [cs]. 21, 24

[156] Jun Zhang, Wolfram Klingsch, Andreas Schadschneider, and Armin Seyfried. Transitions in pedestrian fundamental diagrams of straight corridors and T-junctions. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(06):P06004, June 2011. ISSN 1742-5468. https://doi.org/10.1088/1742-5468/2011/06/P06004. URL http://arxiv.org/abs/1102.4766. arXiv: 1102.4766. 36, 38

[157] D. L. Zhao, Q. Zhang, P. C. Liu, M. J. Sun, X. Q. Wang, X. L. Zhang, and X. Wang. Simulation on Occupant Evacuation at a Public Site Based on SMARTFIRE and Building EXODUS. *Studies in Engineering and Technology*, 5(1):15–24, August 2017. ISSN 2330-2046. https://doi.org/10.11114/set.v5i1.2627. URL http://redfame.com/journal/index.php/set/article/view/2627. Number: 1. 17