DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
UNIVERSITY MOHAMED KHIDER OF BISKRA
FACULTY OF EXACT SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

**DOCTORAT T H E S I S**

In order to obtain the degree of **LMD Doctorate**
in computer science

**Option : Applied Computer Science**

**Entitled:**

# The Crowd Simulation: Creating Variety Within Crowds

Presented by :

**DRIDI Imane**

Defended on: 04/01/2026, in front of the jury:

| | | | |
|---|---|---|---|
| **Mr. BABAHENINI Mohamed Chaouki** | **Professor** | **University of Biskra** | **President** |
| **Mr. CHERIF Foudil** | **Professor** | **University of Biskra** | **Supervisor** |
| **Mr. MEADI Mohamed Nadjib** | **MCA** | **University of Biskra** | **Examiner** |
| **Mr. KHALDI Belkacem** | **MCA** | **ESI-SBA** | **Examiner** |

Academic year : **2025-2026**

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

[illegible]

## ملخص الأطروحة

في حياتنا اليومية، نتفاعل ونلتقي بالعديد من الأشخاص الذين يقومون بأنشطة مختلفة تندمج بسهولة مع روتين حياتهم اليومي. نلاحظ أن كل فرد له مظهر وشكل فريد، وأنهم يقومون بمهام محددة بأفعال مختلفة. كما نلاحظ أيضًا اختلافات في حركاتهم، بما في ذلك الاختلافات في أنماط المشي والجري، إلى جانب السرعات المتفاوتة التي يتخذونها أثناء قيامهم بحركة معينة. بالإضافة إلى ذلك، نحن قادرون على مراقبة كيفية تنقلهم حول بعضهم البعض وتجنب العقبات التي قد يواجهونها. وفي هذا السياق، نقوم بمحاكاة الحشود، مع التركيز على التحدي المتمثل في تمثيل أوجه التشابه وتحقيق التنوع لتنفيذ المحاكاة بطريقة واقعية ودقيقة.
الهدف هو التركيز على تنوع الحشد مع التأكد من أن كل شخصية لها خصائص فردية. تجمع المنهجية المقترحة بين إستراتجيتين. تركز الإستراتيجية الأولى على تنوع المظهر، بما في ذلك الشكل والإكسسوارات واللون والملمس. يستخدم الأسلوب الثاني أنواعًا مختلفة من الحركات.حيث يسمح بإنشاء حلقة تشمل مجموعة من الحركات مختلفة لكل شخصية، باستخدام كل من وحدة التحكم في الحركة والبرنامج النصي للحركة.
تم استخدام استراتيجيات تجنب الاصطدام بين الشخصيات، مع خوارزمية مقترحة تعتمد على Raycasting، ومبادئ نيوتن، والحفاظ على مقدار الحركة والطاقة الحركية في غياب القوى الخارجية. تنتج هذه الخوارزمية نتائج فعالة لتحديد قيم السرعة النهائية الدقيقة للشخصيات. تمت محاكاة نظام الحشد المقترح في ظروف إضاءة مختلفة لعرض شخصياتنا وخلق الواقعية البصرية. حيث أنه يشتمل على مزيج من كل هذه المصطلحات لتحسين واقعية الحشد من حيث الجودة المرئية.
**الكلمات المفتاحية:** محاكاة الحشود، الجودة البصرية، تنوع الحشود، تنوع المظهر، الحركة،الإضاءة

# Résumé

Dans notre vie quotidienne, nous interagissons et rencontrons de nombreuses personnes qui effectuent diverses activités s'intégrant naturellement dans leur routine quotidienne. Nous constatons que chaque individu a une apparence et une forme uniques, et qu'ils accomplissent des tâches spécifiques avec des actions différentes. Nous remarquons également des variations dans leurs mouvements, y compris des différences dans les modèles de marche et de course, ainsi que des vitesses variables qu'ils choisissent de déplacer. De plus, nous pouvons observer comment ils se déplacent les uns autour des autres et évitent les obstacles qu'ils pourraient rencontrer.

Dans ce contexte, nous simulons une foule, en nous concentrant sur le défi de représenter les similitudes individuelles et d'atteindre la diversité pour exécuter une simulation de manière réaliste et précise. L'objectif est de se concentrer sur la variété de la foule tout en veillant à ce que chaque personnage ait des caractéristiques individuelles. La méthodologie proposée combine deux approches. La première stratégie se concentre sur la variation d'apparence, incluant la forme, les accessoires, la couleur et la texture. La deuxième approche utilise différents types d'animations. L'application d'un cycle de locomotion permet la création de diverses animations pour chaque modèle, en utilisant à la fois le contrôleur d'animateur et le script d'animateur. Des stratégies d'évitement de collision sont utilisées entre les personnages, avec un algorithme proposé basé sur le lancer de rayon, les principes de Newton et la conservation de la quantité de mouvement et de l'énergie cinétique en l'absence de forces externes. Cet algorithme produit des résultats efficaces pour déterminer les valeurs exactes de la vitesse finale des personnages. Le système de foule proposé est simulé dans différentes conditions d'éclairage pour rendre nos personnages et créer un réalisme visuel. Il intègre une combinaison de tous ces termes pour améliorer le réalisme de la foule en termes de qualité visuelle.

**Mots clés :** simulation de foule, qualité visuelle, variété de foule, variation d'apparence, animation, rendu

# Abstract

In our daily lives, we interact and meet with many people performing various activities that effortlessly blend into their daily routines. Each individual has a unique appearance and shape and performs specific tasks with different actions. We notice variations in their movement, including differences in walking and running patterns and varying velocities they select for motion. Additionally, we can observe how they navigate around each other and avoid obstacles they may encounter.

In this context, we simulate a crowd, focusing on the challenge of representing individual similarities and achieving diversity to execute a simulation realistically and accurately.

The objective is to focus on crowd variety while ensuring each character has individual characteristics. The proposed methodology combines two methodologies. The first method focuses on appearance variation, including shape, accessories, color, and texture. The second method uses different types of animations. Applying a locomotion cycle allows for the creation of various animations for each model, using both the animator controller and script. Collision avoidance strategies are used between characters, with a proposed algorithm based on Raycasting, Newton's principles, and the conservation of momentum and kinetic energy in the absence of external forces. This algorithm produces efficient results for determining the exact final velocity values for the characters. The proposed crowd system is simulated under different lighting conditions to render our characters and create visual realism. It includes combining all of these terms to enhance crowd realism in terms of visual quality.

**Keywords:** crowd simulation, visual quality, crowd variety, appearance variety, animation, rendering.

# List of publications

- **International Journal Paper**

- Dridi, I., & Cherif, F. (2025). Enhancing real-time animation: Ensuring distinctiveness in crowd dynamics through physics-based collision avoidance. ITEGAM-JETIA, 11(52), 237-246. https://doi.org/10.5935/jetia.v11i52.1677

- **International Conference Paper**

- Dridi, I., Bouguetitiche, A., & Cherif, F. (2025, January). Individuality within crowd: Distinctive features to ensure a diverse crowd appearance. In Proceedings of the 2025 International Symposium on Innovative Informatics of Biskra (ISNIB) (pp. 1–6). Biskra, Algeria.

- **Posters**

- Dridi, I., & Cherif, F, The Crowd Simulation: Creating Variety Within Crowds, 1st Ph.D Days, JDITA'2018, Biskra, Algeria, January 28-30, 2018.

- Dridi, I & Cherif, F. Diversity in the crowd: distinctive features for improved individuality. In 2024 International School on Virtual Reality and Augmented Reality (IVAR'2024) helds at Biskra University. 3-6 November 2024.

# Acknowledgments

I want to dedicate this dissertation to the following:

•First and foremost, I wish to express my heartfelt gratitude **to God**, my creator. His guidance and encouragement have been my strength during the challenging moments of finishing this dissertation. I am deeply grateful for the infinite love, mercy, and grace.

•My supervisor, **Prof. Foudil Cherif.** Thank you for your invaluable guidance, direction, and pearls of wisdom. More importantly, I sincerely appreciate your patience throughout this process. Your belief in my abilities and constant encouragement pushed me to excel. I am truly grateful for your support.

• I would like to extend my sincere gratitude to **the jury members**, your time dedication, and critical insight have been invaluable in evaluating this work. Thank you for your thoughtful and constructive feedback.

• I would like to express my heartfelt thanks to my dear friends, including Walid N, Meftah Z, Laid S, Imad B, Maha S, Khadija G, Assma H, El-haja B, Assma B, Souhila B, I am deeply thankful for their constant support and unwavering belief in me.

• To my **beloved family**, whose endless love, encouragement, and sacrifices have been my foundation. This achievement is as much yours as it is mine.

EL_hamdullah.

DRIDI Imane

# Dedication

With all my heart, I dedicate this work to those who have been a treasured and irreplaceable part of my heart.

To my **beloved parents**.

This thesis is dedicated to you with all my heart. I am beyond grateful for everything you have done. My words can hardly capture the depth of my appreciation. You have been my constant source of inspiration and joy, offering unwavering support during sadness and frustration. Your guidance has shaped me into someone special, determined, and modest. Thank you for teaching me to believe in myself and to persevere through every challenge. It is an honor to have you as my parents, and I am forever thankful for your endless love and support throughout this journey. I love you both more than words can express, always and forever.

To my **sisters**, **brothers** and **lovely nieces**

This work is dedicated to : Himou, Izma, Akrem, Bijou, Taki, Zizo, Tano , Mazin, Assil, Maram and Nour who have never left my side, and their words of encouragement, Sincere love and push are the source of my effort and success. All of you have been my best cheerleaders.

•I also dedicate this dissertation to my special persons dear to my heart; thanks for supporting me throughout the process and for the unconditional love. I will always appreciate all they have done.

EL_hamdullah.

DRIDI Imane

# Contents

# List of figures

# List of tables

# General Introduction

Daily life involves interacting with various individuals and navigating various environments. Each individual has his habits. This characteristic displays variation in how individuals perform their activities, behavior, and external appearance, such as the forms, colors, and textures of their clothing and accessories .

Simulating pedestrian crowds in computer graphics is an immense challenge due to the complexity of effectively and convincingly modeling human behavior. This challenge emphasizes the necessity of crowd behavior simulation in various disciplines. Applications for pedestrian crowd simulation include public safety, security, entertainment, and transportation systems.

This field's research has received considerable interest because of its ability to provide accurate, practical, and successful results that can be applied in real-world scenarios. Effective crowd simulation models are essential for understanding and evaluating interactions in various scenarios. They improve mobility in densely urban areas, and navigation behavior and perform virtual experiments. These simulations solve significant issues, improve security measures, and ensure efficiency in diverse environments. At the same time, providing valuable insight into human behavior and crowd dynamics.

When presenting individuals of various ages and appearances that interact in a virtual environment, it becomes a complex and difficult challenge to represent them convincingly and successfully to reflect the real world. Accurately modeling fundamental real-life behaviors is highly complex, requiring substantial computational effort and time.

To generate these individuals, several issues must be addressed: how to represent them with diverse appearances, alter their shapes, change the colors and textures of their clothing, and ensure their movement, including avoiding collisions in a particular environme

## Motivations

Crowds consist of individuals in a specific location, forming an essential aspect of human interaction. These individuals engage in customary routines, including commuting to work, dining at restaurants, participating in sporting activities, utilizing various modes of transportation, and going to the mosque to perform prayers (Figure 1).

We observe specific differences in how these individuals perform their activities and variations in their behavior and external appearances, such as shape, the different colors, and the texture of their clothes and accessories. We also discern variations in their mobility, encompassing disparities in their walking and running patterns and the varying speeds at which they choose to move. Furthermore, some individuals can be observed seated in specific locations, not engaged in particular actions. Additionally, we can observe how they navigate around each other and avoid obstacles they may encounter (Figure 2).

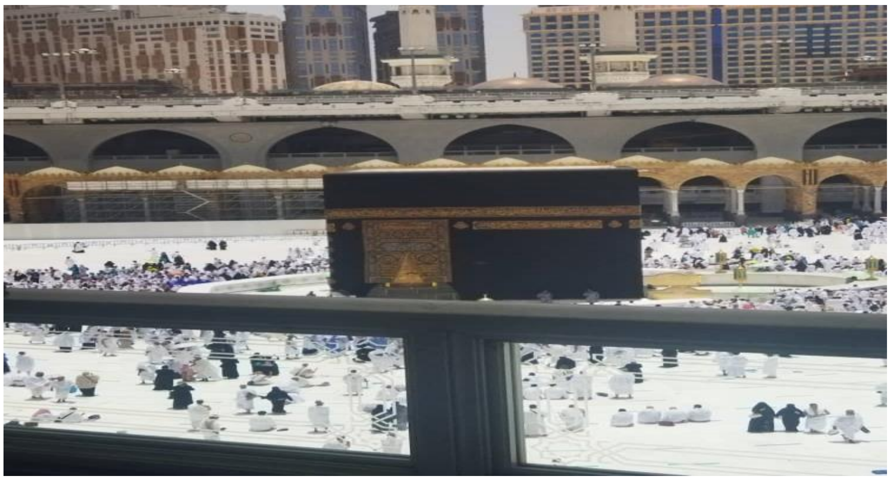


**Figure 1:**A picture taken in 2022 of a group of Muslims performing Umrah, they perform Tawaf around the Kaaba

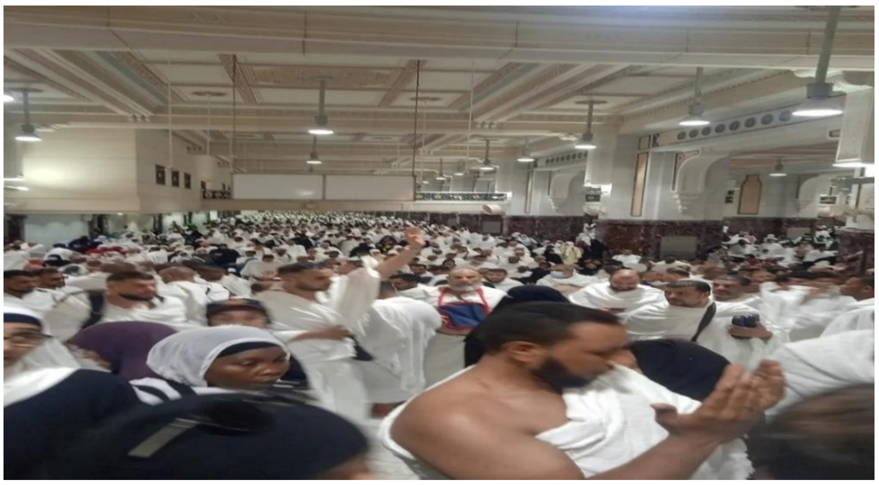


**Figure 2**:A picture of a group of Muslim pilgrims of different ages, genders, and nationalities, performing the Safa and Marwa

Beacco [2] distinguishes between precomputed simulations and real-time applications in computer graphics for crowd simulation. Precomputed simulations, common in the movie industry, focus on generating high-quality virtual crowds, such as those in The Lord of the Rings (Figure 3). In contrast, real-time applications, like video games or virtual reality systems, prioritize interactivity, where computational speed becomes crucial. These applications involve navigation algorithms for individual agents in large virtual environments and require efficient techniques for generating complex scenarios. These scenarios feature animated characters represented by fully articulated 3D figures or similar representations, ensuring realism and performance in dynamic, interactive settings [2].

**Figure 3**:The Lord of the Rings film used MASSIVE software to produce and display massive armies within scenes [2].

## Problem Overview

A virtual crowd is a collection of independent virtual beings that behave similarly to their surrounding entities or have different individual behaviors and personalities. These crowds are composed of many groups of virtual entities. Similar to real-life extras in films, virtual crowds require simulation. Computers enable artists to evaluate the activities of an entire crowd, supervise the movements of subgroups within the crowd, and control the behaviors of each virtual individual in the crowd. Furthermore, it is possible to control the crowd's overall movement, move around obstacles, interact with them, and provide additional animations.

The concept of levels of detail is very useful for effective crowd rendering. When a virtual human moves away from the point of interest, typically the camera, we will gradually replace its detailed mesh with other meshes, more and coarser, then again with alternative representations of more economical virtual humans in calculation time. This allows us to display large crowds of several thousand people.

In this context, each virtual human model serves as a template for instantiating thousands of individuals.

The ability to change the colors and shapes of each instance is important for creating variation and realism in a crowd. Adjusting the colors and textures of clothing and accessories increases the variance in appearance among the created instances. People have different hairstyles, wear glasses, carry bags, use mobile phones, and exhibit other distinguishing characteristics. These characteristics are just as significant as variations in appearance, as their combination contributes to creating virtual humans that are truly unique within the crowd.

Locomotion is one of the primary actions performed by crowds. Thus, variation is an important consideration. To achieve this, a locomotion engine is employed, capable of generating walking and running cycles at varying speeds. This naturally creates a sense of diversity, as individuals do not move at the same pace. Exploring how variety can be effectively introduced during the simulation stage of a crowded engine remains a key area of interest.

Problem-Solving Approaches:

• Appearance Variety:

-Aim to improve visual quality while decreasing the perception of clones.

-Displays the value of each character's unique characteristics.

-Explore the crowd's diversity to create realistic aspects such as shape, texture, and color of clothes and accessories to generate character variation.

• Animation Variety:

-Includes the use of different locomotion cycles for each model.

-Utilizes the animator controller and animator script to generate different motions.

-Uses collision avoidance technique to avoid collisions between characters.

## Objective and Contributions

Our work presents a research focus on addressing a challenge in real-time crowd simulation, specifically the visual duplication of characters. The central objective is to suggest a solution that avoids the perception of duplicated characters in the simulated crowd.

This is achieved by implementing concepts across various virtual human templates. To achieve this overarching objective, we combine two approaches: introducing variety in appearance and diversifying animations with collision avoidance.

The objective of the thesis is to focus on:

- The work concentrates on a real-time crowd simulation problem.
- The primary purpose is to provide a method to avoid the simulation's perception of cloned characters.
- The use of different human models within a single simulation aims to influence the visual quality of the crowd by creating a sense of variation. This approach involves incorporating different models into the simulation.
- The application of a microscopic model in this context includes a specific approach that assesses individual human entities and their behavior inside a virtual environment.
- In the absence of external forces, using the force model based on the Raycasting technique, Newton's equations, and concepts of momentum and kinetic energy conservation refers to implementing a simulation framework. To simulate interactions among individuals in a virtual environment. This framework integrates to determine accurate values for the final velocity of characters. The algorithm efficiently addresses collision avoidance and contributes to the realism of character movements.
- Provides an overall approach based on these physical principles to consider the dynamics and behaviors of entities within the simulated environment.
- A combination of appearance, animation variety, and collision avoidance techniques contributes to rendering characters realistically.
- The proposed crowd system is simulated under various lighting conditions to render the various characters and to improve visual realism.

The primary objective of this study is to improve the realism of crowd simulation by addressing the challenge of visual replication. This is achieved through an approach that considers appearance and animation variation to ensure the representation of a crowd, where each element has individual characteristics. Thus contributing to a more precise and realistic representation of crowds in simulated environments.

## Organization of the thesis

This thesis is focused on major parts:
We begin with the introduction, which outlines the essential points of our work, followed by the problem statement, objectives, contributions, and the organization of this document.
We started with a literature review dedicated to the areas addressed in this thesis. It consists of four chapters:
In the first chapter, we present a state-of-the-art review of crowd simulation, discussing the relevant works in this field by researchers.
We introduce various types of crowd simulations, including the design that describes the concept of the crowd, path planning and virtual human navigation methodologies, the classification model of the crowd, and behavioral factors decomposed into three types: physical aspects, social factors, and psychological ones.
In the second chapter, we explore some approaches in this research domain. We provide a state-of-the-art review of crowd variety. It offers a comprehensive review of crowd diversity, encompassing recent advancements and research focused on appearance variability, animated crowds, and collision avoidance techniques.
In the third chapter, we describe our approach to modeling crowd diversity, detailing the methodologies employed to achieve visual variety within crowds. We elaborate on the different strategies and concepts applied in our implementation, showcasing their effectiveness in enhancing crowd visual dynamics.
The fourth chapter of our work presents our experiments and results. We begin by presenting the gaming engine and programming languages utilized in our implementation, followed by an overview of the architecture and the methods employed to realize the crowd diversity concepts. We then present a thorough analysis of the results and the impact of our applied concepts. We conclude our manuscript with a general conclusion discussing the contributions of our work and the envisioned research perspectives.

# Chapter 1

# Crowd simulation

## 1.1 Introduction

The field of crowd simulation has recently attracted much attention due to its many uses, including planning and organizing pedestrian areas, subway or railroad stations, emergency preparation, and evacuations, as well as educational, training, and entertainment purposes. Various computer simulation models are commonly developed to model human crowds [3].

The movies [4], gaming [5], interactive simulation [6], and safety science [7] industries all make extensive use of virtual crowd simulation techniques. Virtual crowd simulation necessitates a variety of building blocks, including path planning, collision detection, individual rendering and animation, and locomotion [8]. Crowd modeling and simulation have emerged as a critical design issue in various fields, including military simulation, safety engineering, architectural design, and digital entertainment. Crowd simulations have been widely used for real-time tactical military training, including creating civilian behaviors and combat actions in peacekeeping scenarios [9].

Simulated crowd modeling is the main topic of this chapter. This section reviews previous work and discusses the state of the art in crowd simulation. This section aims to provide an overview of the various simulation models; it presents details about each model. In addition, it presents path planning and virtual human navigation methodologies. Then, we indicate the model classifications and behavioral factors.

## 1.2 Related works

Crowd simulation has long been the focus of computer animation studies. Much pertinent research has been done in pedestrian dynamics, path planning, and navigation [10]. Creating behavioral models that consider the actions of actual pedestrians and validating these models against observations of actual pedestrians is the most effective approach to producing realistic crowd simulations

Several research endeavors have attempted to comprehend how almost every species can move around and navigate in space with minimal effort [11, 12, 13]. In particular, humans participate in other cognitive activities, such as social interactions with other group members, while walking in groups, avoiding moving or stationary obstacles, and guiding toward common goals. This theoretical framework [11, 15] suggests that typical patterns that result from these basic behavioral rules should be identifiable [16]. Many approaches and methods have been suggested for crowd simulation and agent interaction. Boids, the previous method, utilized basic velocity rules to produce a net velocity that mixed agent navigation and collision avoidance with flock cohesive behavior [17]. According to [18], macroscopic models are mostly used to calculate traffic simulations and the capacity of large buildings like stadiums and conference centers [19].

Additionally, there are models of Gas-kinetics that use partial differential equations to determine how crowd density and velocity must change over time by comparing crowds to fluid or gas dynamics [20]. Agents use the idea of usefulness to determine their path in route choice models. The optimal approach is determined by attempting to optimize the usefulness of their final destinations [21].

Using Markov chain models, queuing models provide information on how agents move from one network node to another. Typically, these links are doors or portals, and these nodes are rooms. A set of states and transition probabilities are used by Markov chains [22].

Physical force-based approaches direct toward a goal while maintaining a comfortable distance between agents and obstacles by using repulsive and attractive forces [23]. Real forces, including friction, dissipation, fluctuations, and repulsive interaction, are described as "virtual" social forces in social force models. The model accounts for attractive and repulsive forces and simulates crowd behavior using social forces during emergencies [24].

Discrete dynamical systems known as cellular automata (CA; singular: cellular automaton) simulate complex behavior using basic, local rules that move the cells in a lattice. Traffic modeling, biology, physics, and other complex systems have all been explored using CA models [25]. The space is divided into a uniform grid by cellular automata models. According to the modeling system, each agent moves between specific cell positions.

Certain techniques were used to modify the simulation paradigm by applying complex agent representations, such as footsteps [26]. A 2D approximation of an inverted spherical pendulum model of bipedal locomotion can also be used to plan steps.

The "animation-dependent planners method" [27] can be categorized into two primary sets. The first set primarily concerns itself with simulating realistic behaviors related to overall character navigation and does not significantly emphasize animations.

On the other hand, the second set is attentive to the availability of animation clips and conducts planning while considering them. This second set requires a preliminary analysis of the available animation clips to plan paths that adhere to constraints involving the interaction between the character's feet and the ground.

The Based Models method [28] divides the simulation space into several zones. Each zone is simulated for a macroscopic or microscopic model, depending on the application's needs. [29] provided a hybrid method to simulate pedestrian movements in evaluating evacuation problems on a road network, combining macro and micro models. The pedestrian behaviors in the cross-section are simulated using an agent-based Leader-Follower technique.

To simulate both the typical crowd flow in the context and evacuation scenarios, [30] have devised a hybrid modeling approach. This approach combined the macroscopic and microscopic models into a single simulation and simultaneously applied them to different corridor partitions. The crowd flow in the zone simulated using a macroscopic technique is provided, while the behavior of each individual is observed in the zone simulated using a microscopic model.

Sequential Models is a hybrid model based on layers that run macro and micro models for the whole crowd. It applies a micro model to the same crowd to observe individual behaviors after running a macro model to control the crowd's movement pattern [31].

[32] Provides a framework for simulating crowds that incorporates the BDI (Belief, Desire, and Intention) model for planning and decision-making and the RVO (Reciprocal Velocity Obstacle) model for navigating crowded spaces. Low computational costs are available. [33] It involves integrating macroscopic and microscopic models into a single framework. It uses a microscopic model in low-density areas to simulate individual pedestrian behaviors. In other areas of the simulation environment, they use a faster continuum model of pedestrian flow. [2] focuses on creating a multi-domain planning method in crowd simulation, specifically focusing on planning that considers individual footsteps rather than root velocities and positions.

## 1.3 Crowd modeling approaches

Research in real-time crowd simulation presents challenges, especially as convincing results necessitate the effective utilization of processors (CPU and GPU) [34]. To better understand the behavior and movement patterns of actual crowds, crowd simulation has become an effective technique. There are different types of crowd simulation, each with a specific purpose and area of concentration. Some common varieties of crowd simulation include the following:

### 1.3.1 Macroscopic models

The earlier crowd simulation models were macroscopic, according to their inexpensive computational resource requirements. These models can simulate many pedestrians without focusing on individual actions, making them appropriate for developing evacuation simulations and determining safety conditions during evacuations [35].

Most of these models are derived from mathematical models, such as equations that are based on the averaged traits of a group of people. The population is considered a structure in this type of simulation, and a certain number of variables can characterize it. The simulation model attempts to analyze variations and changes in these variables. Macroscopic simulation employs a viewpoint that focuses on the entirety of the system, aiming to replicate the collective dynamics of the system as a whole [36].
Typically, the system is represented at a relatively high scale, which suggests an imprecise simulation. Furthermore, the environment is typically not well represented in these models. The environment is frequently reduced to a container that represents the crowd and through which one or more fluids flow. Macroscopic simulation generally does not consider the interactions between the system's entities. The macroscopic simulation does not take into account in detail the interactions between the entities of the system [36]. This model includes types of representation, such as Route choice models, Markov chain models, Queuing models, and Gas-kinetics models.

### 1.3.2 Microscopic models

The microscopic models concentrate on modeling the local behavior of individual agents and how they interact with other agents within the crowd. These models depend on an individual scale, determining the location of each agent at any given time. Individual behavior and interactions are presented using microscopic models representing pedestrian flow at the individual level. Crowds are often regarded as a collection of independent individuals who can interact with each other, have different characteristics and motivations, and follow a leader or objective. These models are classified according to their method of describing the relationships among the individuals they represent [37].
The microscopic approach involves identifying the interactions between each individual and its neighborhood to influence the individual's behavior. These models differ in their way of considering information from a pedestrian's neighborhood and the individual's reaction to it [35].
These models are classified into various approaches, differing in their agent representations, local movement functions, and the discretization of space and time [2]. They encompass several types of representations, including Rule-Based Models, Social Force Models, Cellular Automata Models, Footstep-Driven Approaches, and Animation-Dependent Planners.

### 1.3.3 Hybrid models

Recently, researchers have been considering hybrid approaches that combine microscopic and macroscopic methods, using their strengths well. Prior research has tried to utilize both models in part or as an integrated whole [37]. The complementarity between the advancements of macroscopic and microscopic models is evident. Some work aims to merge the two types of models in order to take advantage of their respective advances [14].

Hybrid simulation creates a flexible, interactive, high-fidelity simulation on a large scale by combining the best features of both types of crowd modeling [37]. It combines a novel continuum method for pedestrian collective motion with a microscopic model of individual navigation; with commodity hardware, it may be used to simulate the movements and behaviors of very large crowds at almost real-time rates. This method can identify the best modeling model in real-time and in a continuous environment for each area in the environment, considering the simulation context.

The approach of Sequential Models provides a synchronization technique to move the crowd state between the macro and micro models while executing them sequentially. First, the macro model simulates the crowd's movement by running based on the speed-density relationship. The synchronization module enables the transfer of macro results to the microscopic model inside the same simulation time step.

A microscopic model is then applied following the movement patterns and density produced in the initial macroscopic step. Synchronization is essential to integrate the results of both simulation models within a single time step [37].

## 1.4 Path planning and virtual human navigation methodologies

The primary challenges mobile robots consider revolve around navigation and path-planning areas that have been studied for many years. Therefore, numerous techniques were described and implemented for the mobile robot path-planning problem. There are two optimization strategies for mobile robots: nondeterministic or classical and deterministic or heuristic approaches (Figure 1.1) [116].

### 1.4.1 Classical approach

Classical approaches were introduced and established in the 1980s and 1990s. Before the heuristic technique, researchers used traditional methods to address path-planning problems [117]. However, the fundamental disadvantage of this strategy is that it has a high processing cost and fails to adapt to the environment's unpredictability. Therefore, it is rarely used for associated performance [118].

This section includes popular classical approach methods, including the Dijkstra algorithm, artificial potential fields, probabilistic road maps, and cell decompositions.

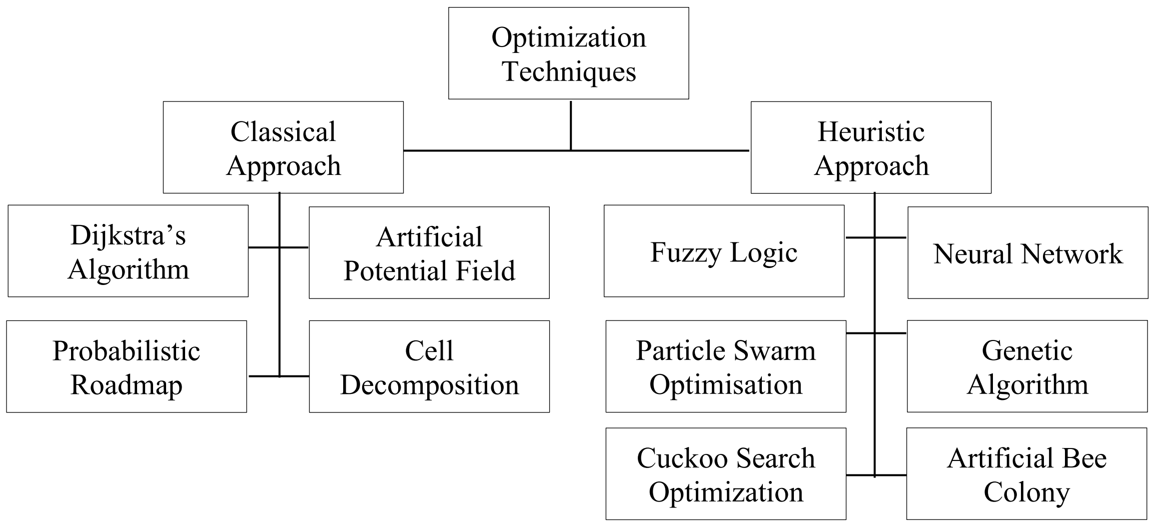


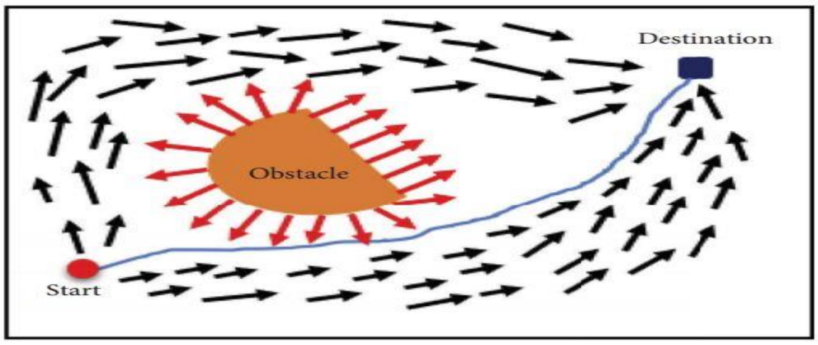
**Figure 1.1:**The collective arrangement of optimization techniques employed by various researchers [116].

### 1.4.1.1 Dijkstra's algorithm

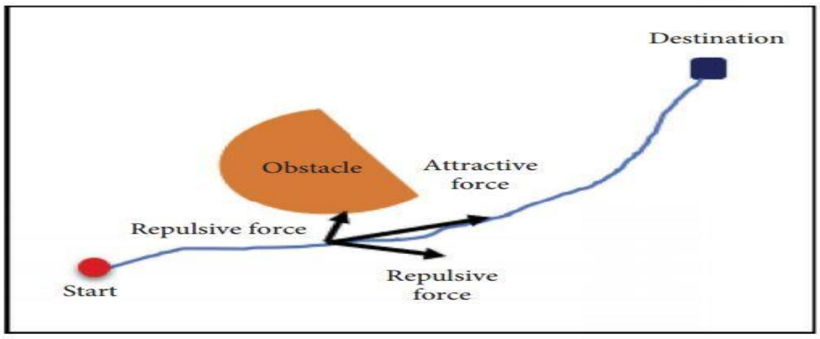
The Dijkstra algorithm is an established approach for determining the shortest path between nodes in a directed network [119], making it a popular choice for path planning. Although effective, it becomes less efficient when the starting point and destination are far apart [120]. This strategy is also employed in algorithms that identify the shortest evacuation route.

### 1.4.1.2 Artificial potential field

[121] proposed the artificial potential field (APF) method for mobile robotics, which relies on the difference between attraction and repulsion forces to improve robot movement through the environment. The APF concept indicates that mobile robot movements run inside the field of forces. The attractive force attracts the robots to the objective position, while the repulsive force keeps them away from each obstacle [122], as illustrated in (Figure 1.2).



(a)



(b)

**Figure 1.2** :(a) and (b) represent a mobile robot's navigation system uses APF [116].

#### 1.4.1.3 Probabilistic roadmap

Kavraki et al [123] established the probabilistic roadmap (PRM) in 1996, with a fundamental objective mind guiding a robot through a static environment. PRM offers multi-query planning [124]. Road maps describe the robot's free space connection configuration, shown on a one-dimensional curve or line. The roadmap has also been called highway strategy [125], withdrawal strategy, and skeletal system [126]. This method creates road maps using the visibility graph and Voronoi graph [125] to find the shortest path from the starting point to the destination. (Figure 1.3) shows both a PRM visibility graph and a Voronoi graph.

In contrast to the visibility graph, the Voronoi graph employs a different approach to generate road maps in robotics and path planning. This graph utilizes Voronoi cells, partitioning the space based on proximity to obstacle edges. The plane is divided into distinct subregions by selecting two neighboring points on the obstacle's periphery as center points. Consequently, the resulting road maps maintain a safe distance from obstacles, ensuring a secure path but longer than those produced by the visibility graph [127].

#### 1.4.1.4 Cell decomposition

The cell decomposition (CD) technique offers a fundamental concept for categorizing open and occupied spaces by obstacles between geometric sections or cells [127, 130]. A cell is a succession of simple components formed by dividing open space [127]. The purpose of CD is to minimize the search area by using a cell-based representation. In addition, the goal is to generate a series of collision-free cells from the starting point to the target point [33] (Figure 1.4).

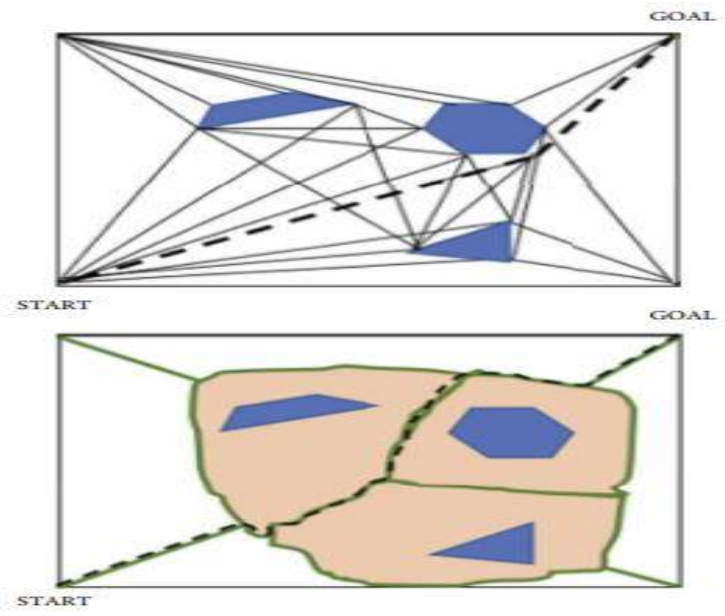


**Figure 1.3**:Visibility graph (up); Voronoi graph (down) [116].

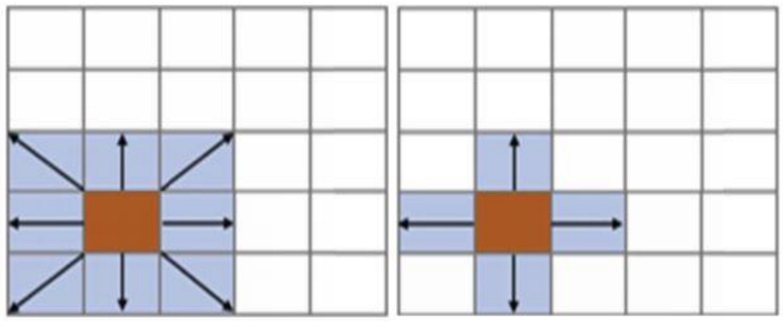


**Figure 1.4:** Approximate CD [116].

### 1.4.2 Heuristic approach

A heuristic approach has been designed to optimize solving complex problems [128]. This technique has shown significant effectiveness and is now widely employed in autonomous navigation systems, contributing to enhanced decision-making and operational efficiency in various applications [129].

#### 1.4.2.1 Fuzzy logic

Fuzzy logic (FL) is a method for convincing an individual's intellect. FL is a unified estimation (linguistic) method for determining unknown facts with uncertain rules [131]. Lotf A. Zadeh first proposed the FL system in 1965 [132]. FL is a formal plan that represents and executes human experts' heuristic intelligence and observation-based methods [126], [133]. (Figure 1.5) shows an example of a primary FL controller utilized in [125]. Hex Moor [134] developed the application of fuzzy logic principles to navigate robots and navigate around obstacles.

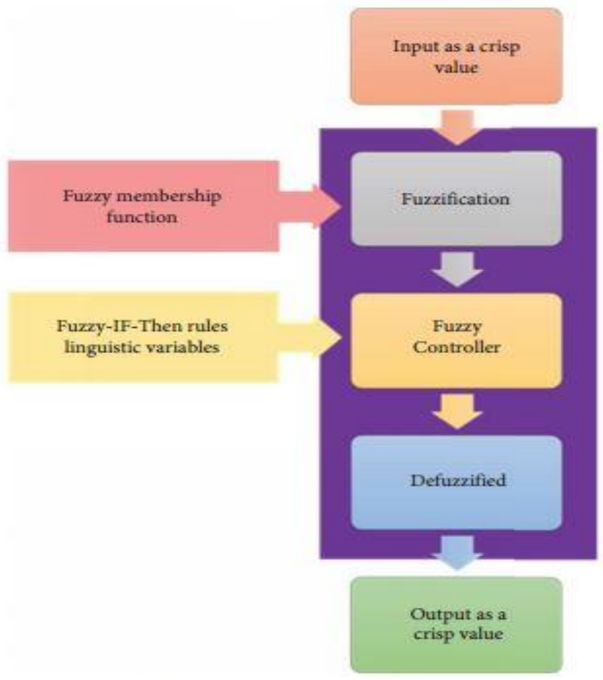


**Figure 1.5:** Basic controller for FL. [116].

### 1.4.2.2 Neural network

A neural network (NN) is an intelligent system inspired by the neural perception system. It was first designed by [135] to plan mobile robot routes. It reproduces the neurons that exist in the human brain [136]. NN has applications in various fields, including discovery, search optimization, pattern recognition, image processing, mobile robot path planning, signal processing, and more. NN comprises various plain and highly interconnected processing components that move data to external inputs [125]. (Figure 1.6) illustrates the input, objective, and output of NN path planning.

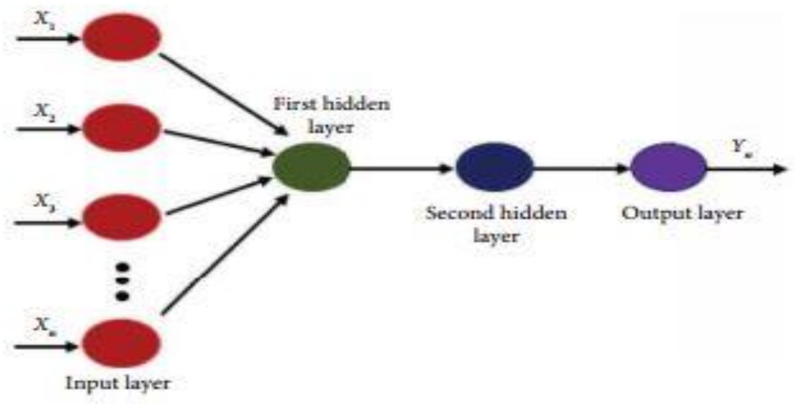


**Figure 1.6**:The architecture of neural networks [116].

### 1.4.2.3 Particle swarm optimization

Particle swarm optimization (PSO) is a path-planning technique developed by Kennedy and Eberhart in 1967[137]. The nature-based heuristics approach is simulated by modeling the social behavior of groups of birds, fish, or flocks of creatures to find food, adjust their environment, and deal with predators [138]. These simulations label each population member as a particle, representing a potential solution to the problem.

PSO is similar to a Genetic Algorithm, beginning with a randomly generated population [139]. Unlike GA, each solution is assigned a random velocity. Particles represent feasible solutions that move inside the problem space. (Figure 1.7) illustrates how the placement and velocity of PSOs in the search region determine their basic notion.

The PSO is utilized for mobile robot path planning in various fields, including humanoid robots [140], industrial robots [141], wheeled robots [142], flying robots [143], and underwater navigation [144] in complicated 3D environments. PSO, Zhang, and Li [145] optimized path time for mobile robots in dynamic environments.

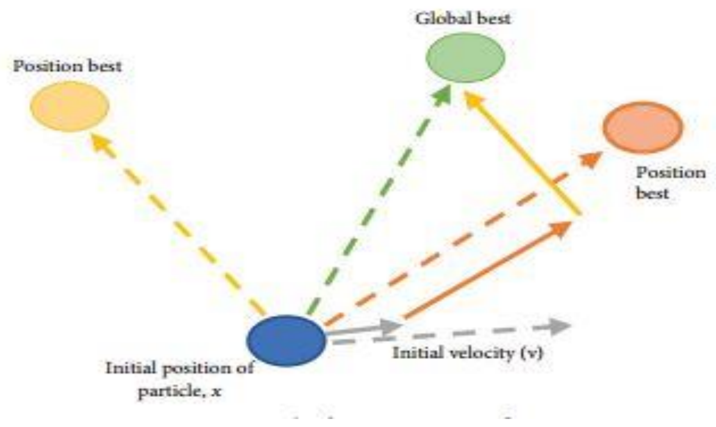


**Figure 1.7**:The fundamental concept of PSO [116].

### 1.4.2.4 Genetic algorithm

Bremermann [146] proposed the genetic algorithm (GA) in 1958, which optimizes natural genetics and selections. This fundamental concept was inspired by the GA's selection of the concept of reproduction of fitness. The fitness value represents all members of the population. The most important members of the population will survive, while the weakest will be abandoned.

According to their fitness value, surviving individuals allow genes to be presented to the next generation using bio-inspired operators such as crossover, mutation, and selection. This randomized structure information was combined to construct a search algorithm that generated solutions to the presented problems, allowing suitable paths to be developed. The population can be described as a binary string arrangement (related to chromosomes) [116]. Bio-inspired operators must maintain population variety. By preventing early convergence, the chromosomal population varies from one another.

The GA is applied in a variety of mobile robot path planning problems, including navigating a humanoid robot [147], an underwater robot navigation challenge in 3D path planning [148], and an aerial robot [149], [150].

#### 1.4.2.5 Cuckoo search algorithm

Yang and Deb [151] identified the cuckoo search algorithm (CSA) as a path-planning approach 2009. This strategy simulates cuckoos' swarm intelligence or inefficient behavior when putting eggs in other host nests. CSA is a recently developed heuristic technique in mobile robot path planning; therefore, its application is limited.

#### 1.4.2.6 Artificial bee colony

In 2005, Karaboga [152] invented the artificial bee colony (ABC) technology, a swarm-based intelligent foraging behavior of honeybees in search of food. Safari and Mahjoob [153] utilize the Bee Colony Algorithm for real-time path planning in mobile robots, focusing on optimizing navigation and decision-making. Its potential application in Mixed Reality (MR) environments enhances efficiency and adaptability for robots navigating dynamic or static hybrid spaces requiring effective path planning and navigation solutions. The author [154] suggested employing MR inside navigation in a static environment to find the best real-time path.

### 1.4.3 Synthesis

Although the heuristic method was created to solve the limitations of the traditional approach, it still has benefits and disadvantages. Fuzzy logic may reduce reliance on environmental information while maintaining high flexibility and performance. However, because experts typically establish fuzzy rules, robots cannot learn and have limited flexibility. While neural networks can manage tasks that linear programs cannot, they demand significant processing time. While Particle Swarm Optimization is simple to develop and requires fewer parameters, it struggles with parameter control. Genetic algorithms support optimization with multiple objectives and stability.

An extensive computing process causes it. Cuckoo Search is relatively easy to construct as it only requires one parameter. However, it can only address continuing difficulties. The initial solution limits the search space in a bee colony [116].

## 1.5 System crowd design

### 1.5.1 What’s a crowd?

The question is, what exactly defines a crowd simulation model? Also, how do we define a crowd? A crowd is a collection of individuals, creatures, or other entities that move, interact, and exist in the exact location. The size of the obtaining varies according to the individual problem being addressed. For example, the aim is to provide scenarios involving as few as one agent or hundreds to thousands of agents. A crowd is essentially a collection of individuals known as agents. An individual agent represents one of the individuals in the crowd.

Agents can move, engage in various activities, and exhibit different interactions or responses to simulation events. Another crucial component of the model is the environment, which should be represented using a 3D mesh or abstracted through a graph structure. Lastly, the third essential element that must be considered in our model is time and its division into discrete intervals [2]. (Figure 1.8) illustrates this.

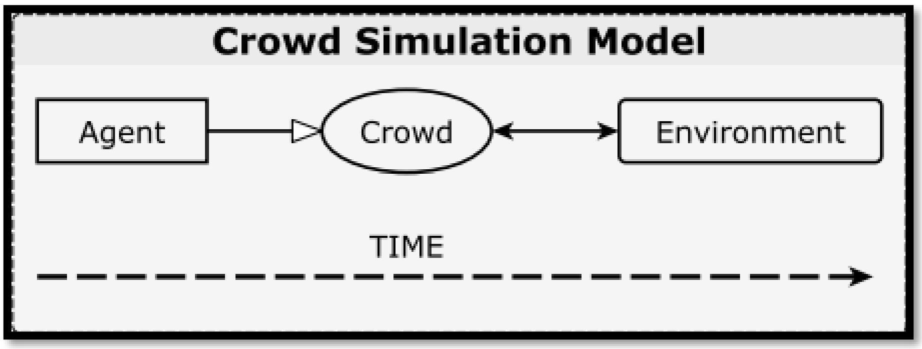


**Figure 1.8**:A model of crowd simulation consists of a group of agents and the environment that they inhabit in. Then, the simulation runs continuously [2].

In nature, human crowds are an interesting social phenomenon. A crowd exhibits a highly constructive force and a well-organized structure in certain scenarios. In some scenarios, though, a crowd rejects cultural norms and turns into a pack of egotistical animals. A crowd consists of more than a group of individuals. Others can influence individual behavior in the group for various physiological, psychological, and social reasons [9].

Individuals often notice that they behave very differently than alone when they are in a crowd. That means that an individual might be pressured to act in a way that the general public considers appropriate in some cases. As such, crowd dynamics can be extremely complex. Analysis models or entirely mathematical methods are typically insufficient for accurately describing crowd dynamics.

Over the years, various pedestrian simulation models have been created in several fields, such as computer graphics, robotics, and evacuation dynamics. These can be divided into two main types: microscopic and macroscopic approaches. Microscopic models consider individual pedestrians' behavior decisions and interactions with others in the crowd. In contrast, macroscopic models concentrate on the system as a whole (flow characteristics rather than individual pedestrians) [25].

Several aspects of a crowd may also be included in crowd models. The external characteristics of a crowd, such as appearance, individual poses or movement patterns, and coordinated positions, are the subject of some studies; the social behaviors of a crowd are the subject of other work that focuses on how they change over time [9].

### 1.5.2 Model Classifications

The existing models of crowd phenomena can be divided into three groups based on the size and time scale of the crowd. These categories include models of long-term crowd phenomena, short-term phenomena of large crowds, and short-term phenomena of medium- to small-sized crowds.

(Figure 1.9) illustrates how each category relates to a region in the 2D space, which is determined by the crowd size and the time scale. A simulated crowd's size and time scale are directly related to the features and modeling techniques of the crowd model [9].

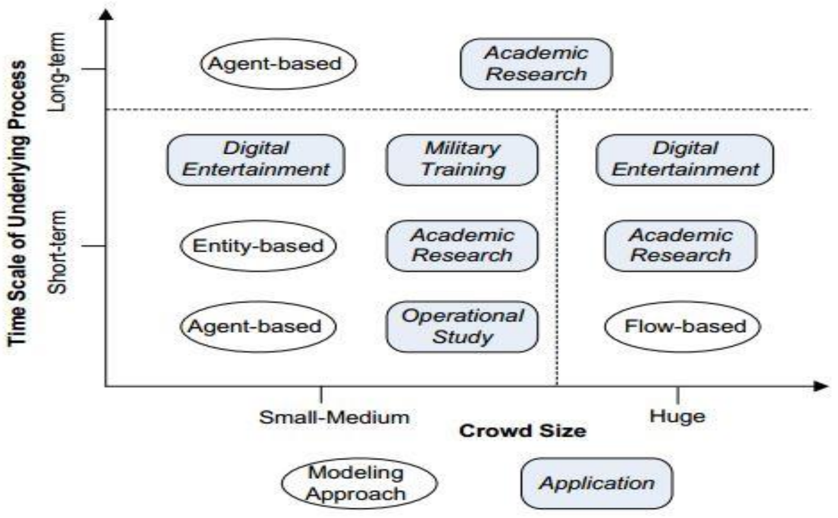


**Figure 1.9**:Classification of crowd model [9].

Several approaches to crowd modeling may replicate a real crowd's features, conditions, and actions at varying levels of detail. The goal of a crowd simulation system is to limit the phenomena's application and determine how long it takes to occur.

The crowd's intangible social and psychological aspects are typically the emphasis of a crowd simulation model of a long-term event, as opposed to its physical attributes. Generally, studies in this category are appropriate for high-level operational studies or social science academic research, such as those examining the formation and spread of extreme ideologies within a crowd. The physical properties of crowds, particularly their placements and movements, are typically described via simulation models of short crowd occurrences. In addition to academic studies, Military training, digital entertainment, and daily operational studies have all extensively used these simulations [9].

### 1.5.3 Behavioral factors

Crowd dynamics are defined and influenced by several elements, including individual positions and speed of movement, diverse social connections, and emotional states. Depending on their goals and design requirements, various crowd models may account for different behavioral characteristics. We classify these components into three major categories: physical aspects, which are tangible; social factors, which are intangible; and psychological ones, which are intangible.

Physical factors are those of an individual visible from the outside, such as their position, movement patterns, and appearance. A crowd model that only considers these concrete aspects typically looks at the potential effects that other individuals within the crowd may have on an individual's movement.

Physical elements are typically considered fundamental and low-level decision-making inputs in most crowd models [9]. A wide variety of social factors, including culture, social norms, family ties, and leadership, influence behavior. Typically, social theories and social studies observations serve as the foundation for the computer models that consider these elements. Many studies demonstrate that psychological elements like emotion have an impact when they affect human decision-making. According to the complex nature of the process, perspectives on how these factors affect human behavior remain varied.

## 1.6 Conclusion

This chapter classifies the various existing works and development efforts related to the simulation of crowds. Additionally, it aims to explain various approaches for modeling pedestrian behavior in simulations that improve real-time crowd behavior simulation and intuitive navigation control, including macroscopic, microscopic, and hybrid models. Following this, we delve into two optimization strategies for moving entities: nondeterministic or classical and deterministic or heuristic approaches. Traditional methods addressing path planning problems include Dijkstra's Algorithm, Artificial Potential Field, Probabilistic Roadmap, and Cell Decomposition. On the other hand, heuristic strategies designed to solve complex problem resolution, encompass Fuzzy Logic, Neural Networks, Particle Swarm Optimization, Genetic Algorithms, Cuckoo Search Algorithms, and Artificial Bee Colonies. Additionally, the chapter addresses behavior factors in various physical, psychological, and social domains when simulating crowds.

# Chapter 2

# Research studies focusing on the diversity within crowds

## 2.1 Introduction

The amount of diversity incorporated into the 3D models used to visualize the virtual human population significantly impacts the perception of crowds [38].

This chapter focuses on the concepts used to ensure variation within the crowd. Firstly, we start by the modeling of appearance variation within crowds. This part delves into a comprehensive review of prior research, exploring the current state of the art in appearance variation. This segment's primary objective is to provide a comprehensive overview of the diverse forms of appearance variation, such as shape, clothing, color, and texture variety.

Secondly, we delve into studies examining animation diversity within crowds. We present a review of relevant work to provide a complete overview of several techniques to improve animation variation. We investigate strategies used in motion synthesis. Additionally, collision avoidance within crowds takes center stage in this chapter. We offer a concise overview of related work and present approaches to collision avoidance concepts. Then, we present the concept of collision detection to avoid collisions between objects and categorize them into two systems. Finally, we offer a synthesis of these related works.

## 2.2 Appearance variety

Modeling appearance diversity within crowds is crucial for crowd simulation. This aspect is significant for developing realistic and immersive simulations. We delve into a comprehensive review of prior research, exploring the current state of the art in appearance variation.

This segment's primary objective is to provide a comprehensive overview of the diverse forms of appearance variation, such as shape, clothing, color, and texture variety.

### 2.2.1 Studies exploring appearance variation

This section presents previous research works that ensure a realistic crowd variety in different aspects such as form, height, color, and texture. The study in [39] presents a variety of colors and textures. It offers a way to create as many color variations as possible from a single texture.

Magnenat-Thalmann et al. [55] categorized the methodologies employed in modeling virtual individuals into three primary groups: creative, reconstructive, and interpolated. The creative approach involves artists generating geometric models, such as anatomically based models. The reconstructive category focuses on constructing the 3D geometry of virtual humans by capturing existing shapes from 3D scanners, images, and video sequences. In the category of interpolated modeling, new geometric models are created by combining example models with the interpolation scheme. When addressing crowds, the initial creative technique becomes impractical, and the second reconstruction method is unsuitable for large crowds due to cost and inflexibility. The preferred approach is interpolation, which offers greater efficiency. [56] involves separately adjusting the height and shape of the human body for large crowds. Scaling the skeleton of a human template changes its height. The space of height scaling for a specific human template skeleton can be specified (Figure 2.1).

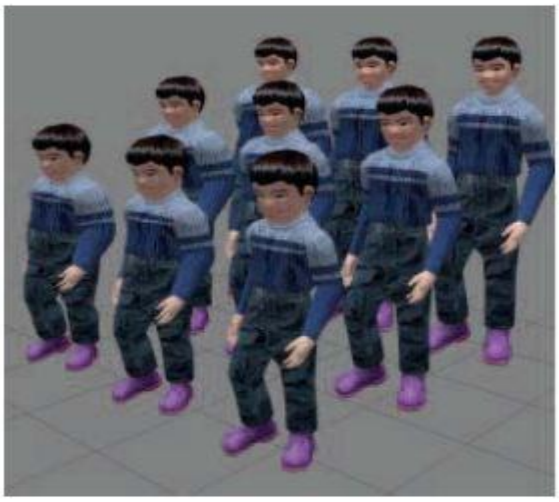


**Figure 2.1**:Height diversity [56].

The variation of body shapes is essential point to get convincing results. The authors in [40] aim to improve the clothes variety of models. They use a set of different human meshes and change their appearance using different "outfits." These outfits define a set of mesh skin and clothing colors, each associated with a specific material ID. The findings in [41] highlight that color variation can mask cloned models, random orientation, and motion. The authors created an alpha map, which coded each region with a unique greyscale value. Then, they manually generated 32 unique outfits for each model with different hair, skin, clothing, and shoe colors.

Previous works use different methods to realize the variety of colors and textures. One of the first examples is presented as the template texture's alpha channel, so each character's body part affects the color and encodes each region with a unique greyscale value [58]. In [63], the authors improve the application of the PCA (Principal Component Analysis) method to identify the appearance information structure relative to a shape. The aligned maps and the texture warps generate the Eigen textures and the Eigen warps that encode the appearance variations.

The variation of body shapes is an essential point to get convincing results. For example, [42] presents a method from a corpus of scans of the 3D range. The authors provide shape variation based on identity- and pose-dependent, allowing for the creation of different virtual individuals with realistic body deformations. The work of [43] presents a parametric editing system for a dressed human body using a Shape Completion and Animation of People (SCAPE) model. This revised SCAPE approach separately represents variations in body shape and poses, using semantic parameters, providing intuitive control, and simultaneously allowing real-time computation. They demonstrate that this system is adequate for 3D human bodies in various poses, with various body shapes and clothing.

To simulate a real-world crowd, [38] created a set of crowd templates in various sizes and forms. They represent a data-driven technique for creating a crowd with diverse body shapes. They base their model development and distribution approaches on body measurement and demographic data from the anthropometric database Civilian American and European Surface Anthropometry Resource (CAESAR) (Figure 2.2).

[44] Examined the attraction and realism of female body shapes developed as morphs between realistic and stylized versions using concepts for design from notable computer animation studios (Figure 2.3). The study indicated that characters with in-between morphs were the most appealing. Approximately 33% of these morphs received the highest scores for realism and appeal, while 66% were evaluated as likewise appealing but less realistic.

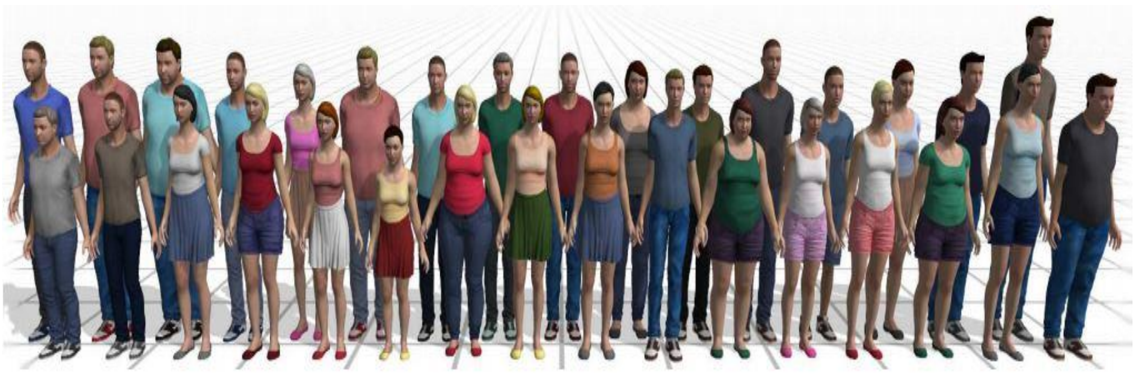


**Figure 2.2**:A collection of models generated with data-driven shape selection and creation method [38].

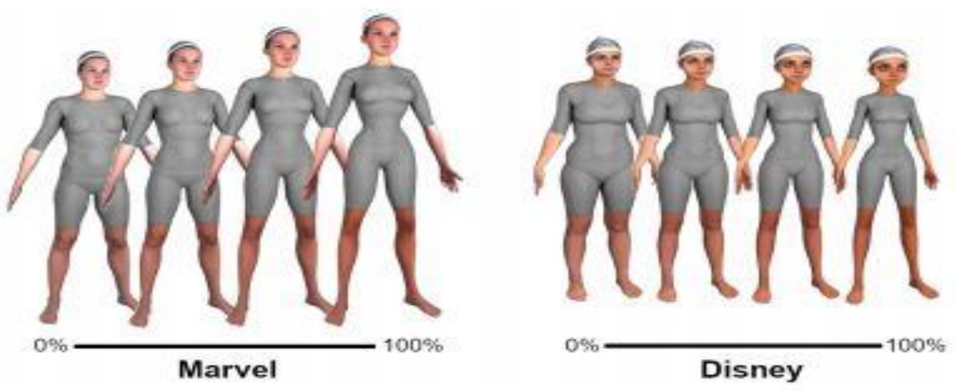


**Figure 2.3**:Transitions from realistic to specific stylizations of virtual characters, sampled at 0%, 33%, 66%, and 100% [44].

Participants also found it challenging to identify color clones within a scene [41], and introducing variations solely in the colors and textures of characters' upper garments proved to be as effective as comprehensive variation [57] (refer to Figure 2.4).

In real-time crowd applications, two widely adopted techniques for introducing diversity are hardware-accelerated color modulation per body part [58], [40], [59] and texture variation [60], [61]. (Figure 2.5) illustrates a typical pipeline, while (Figure 2.6) showcases a real-time rendering of a typical crowd employing similar techniques.

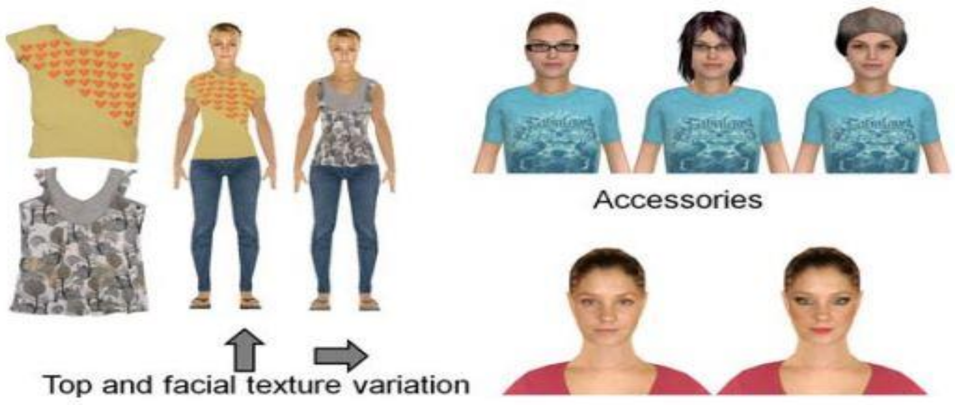


**Figure 2.4**:Clones can be disguised utilizing texture changes in top garments and faces. Accessories, while more expensive, significantly improve the perceived variety of a crowd [57].

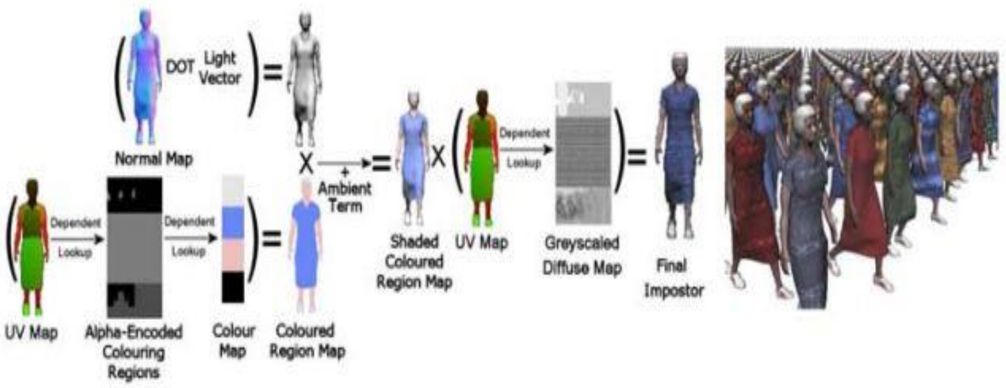


**Figure 2.5**:A typical rendering pipeline that enables variation (left) and an example of adding variation to one character using eight different diffuse textures (right) [40] [61].

[45] report that the variation of the textures of the faces, for example, beards and makeup, were equally as effective as expensive calculating techniques that changed the shape of the face. Another solution focusing on the face as a factor is presented in [46]; this solution avoids the problem of uniform faces in the crowd by a technique for automatic or semi-automatic generation of human faces, using a template matching segmentation on face models in the composition process for virtual actors. It creates the desired variety of 3D human faces. The study in [47] only focuses on the crowd simulation rendering aspect, which generally requires three criteria: to know the quantity, the quality, and the diversity of characters. Numerous studies have established that variations in facial texture contribute to a considerably enhanced level of attractiveness [48], [49].

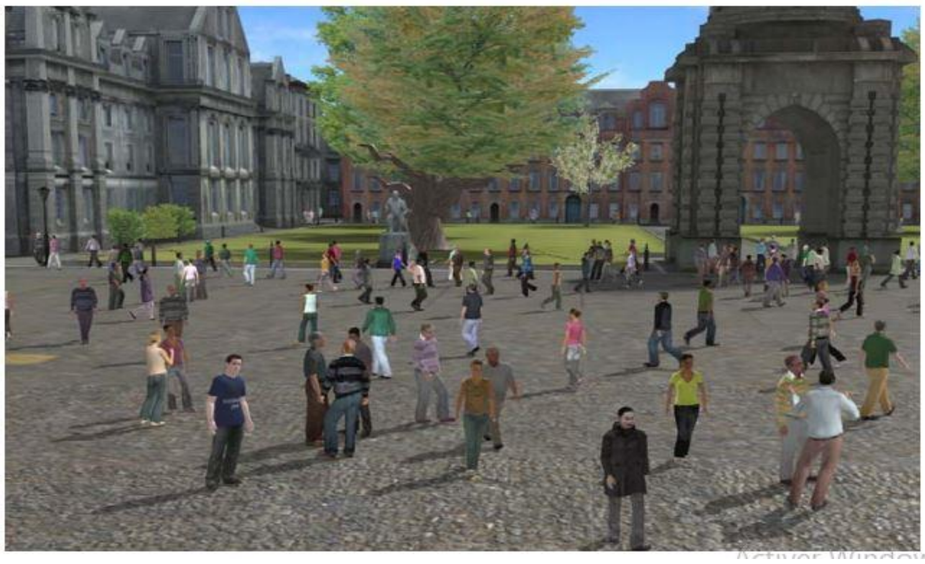


**Figure 2.6:**A crowd scene generated using our Metropolis crowd system using color and texture diversity to create a varied crowd [45].

[50] Created textures from images of women of various ages and tested these textures on a single female virtual character in the cosmetics area. The results indicated that the images with perfect skin were perceived as younger and more desirable than the representations with significant variances in color in the skin.

According to [51], the crowd rendering system preprocesses a source character by applying the fine-to-coarse progressive mesh simplification algorithm to every source character. Vertex normal vectors were utilized to achieve appropriate shading effects for the crowd and related data, such as textures, UV coordinates (2D texture axes), skeletons, skinning weights, and animations. The authors rebuild the meshes of source characters based on the fatten pieces of texture UV sets and organize the source characters and instances into buffer objects and textures on the GPU. Then, they perform runtime tasks on the GPU through five parallel processing components. Their approach is integrated seamlessly with continuous LOD (Level of detail) and View-Frustum Culling techniques.

Impostors are helpful for color modification; however, when viewed closely, the characters appear identical [75], [76], [77], [40], [78]. Texture variation solves this issue by combining additional textures with the character's textures to create effects like make-up, facial hair, wrinkles, eye sockets, face spots, hair coloring, and clothing patterns [57], [79].

Authors in [52] study the number of identical models or clones in the crowd without the viewer noticing them. They present 34 videos using Unity 3D. Each has 10 seconds of time frames with 40 models walking in place. An online survey was set up to validate the hypothesis by asking participants if they found any clones in the video and how difficult it was to find them. Their responses indicate that determining the clone that was acquired or spread is simple.

The study in [51] represents a real-time crowd-rendering system. It arranges the different types of character data on the GPU, like vertices, triangles, and vertex normal, and then performs runtime tasks on the GPU.

## 2.3 Animation crowd

The movement of characters in a virtual environment can be studied through crowd simulation. Most research on crowd modeling and simulation focuses on how a crowd moves. Managing agent motion in virtual environments necessitates precise control over navigation. Crowd modeling and simulation is a rapidly evolving field, so the research developments and achievements of different communities tend to be diverse and deal with issues from different perspectives [9]. Each human has to do his/her action to get a convincing variety of animation visually because if they take the same animation, the results become unrealistic [41]. Simulating and realistically visualizing large crowds in virtual environments is a costly task. Realistic simulations require characters to be animated and to look different from each other [71].

### 2.3.1 Research on animation within crowds

Motion is important to sense the variety of the crowd because the physical appearance of virtual characters cannot vary enough. As a result, many researchers have proposed methods for animating. For instance, [72] focuses on reducing memory requirements to generate and animate various characters of the crowd. There are two steps: The first is to reduce segment and label a set of virtual character data into simpler body parts. The second step includes a method for integrating installation and peels information into the shared texture space between the new characters. Rig and skinning information is included in global textures via the UV parameterization that every created character shares, requiring only one set of rig and skinning textures for each appropriate gender or age variant (male, female, child) to animate an entire crowd [72] (Figure 2.7).

The study in [2] proposes a few techniques for improving the following areas: It works on a multi-domain planning approach and planning using steps instead of just the speeds and positions of the roots in crowd simulation.

The animation focuses on a frame to eliminate the artifacts of sliding of the feet and on the synthesis of the movements of the characters to follow the track. It also provides a novel rendering method based on joint impostors in validation experiments.

[71] Reduce memory usage and optimize the rendering stage. They use two complementary structures; the initial structure is a skeleton with octrees connected with each limb, which is used to compute the amount of detail of geometry and animation. The second structure is a scene tiling used to select the level of detail for the character's geometry, animation, and behavior. On top of this tiling, a quad-tree is generated and used for additional rendering optimization, allowing us to merge geometry from multiple characters in parts of the scene far from the camera. This tiling can also be a balancing mechanism, making collision avoidance calculations more efficient [71]. Impostors present a significant memory challenge because they require much memory to compute and pre-store each animation key frame and viewpoint. Impostor's memory footprint has been successfully reduced due to the efforts of [73] and [74]. Regardless of these developments, the introduction of visual variation, which is required to ensure character individuality, causes an unexpected increase in memory utilization. This is due to the use of visual variety methods, which involve changing the character's color, textures, shape, and movements, requiring a larger memory footprint even for modified impostors.

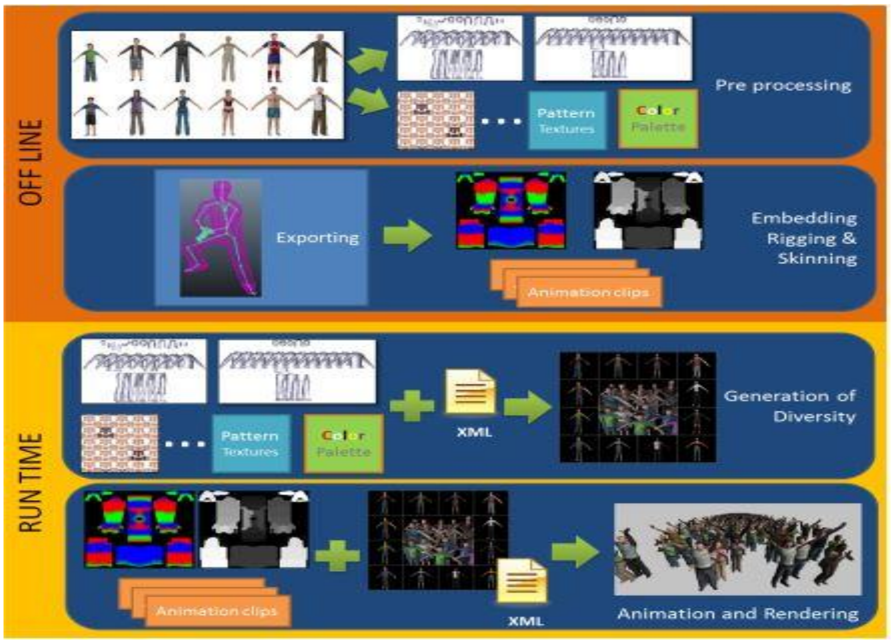


**Figure 2.7**:Overview of the Method. Pre-processing and runtime phases are included in our method. Offline asset organization, reduction, segmentation, and rig and skinning encoding in texture space are all done. The resulting assets are used to generate diversity, animation, and rendering during runtime [72].

[81] Describe an improved impostor technique in which each character is represented as textured boxes that store color and depth information. The boxes are animated using rigid transformations, and the shape is reconstructed via relief mapping. The same authors show how to render animated characters efficiently using optimized per-joint impostors sampled from discrete view directions [74]. Instead of selecting a single image for the entire character from a series of pre-defined postures, their characters are animated by applying rotations directly to the impostor's joints.
[41] It discovered that character duplicates were difficult to detect due to animation variability. Especially when a duplicate was associated with random motions or when motions were out of step. The difficulties of animation variation include establishing the mechanism by which a character will move (rigging), reducing or eliminating the visual artifacts produced by joint motion (skinning), and animation reuse. Furthermore, to reduce creating time, they must be executed automatically.
[80] Describe a multi-resolution point sample rendering technique for keyframe animations in their study. This method takes keyframe animations of triangle meshes as input and creates a hierarchy of point samples and triangles that display the scene's various resolutions. This approach uses a randomized sampling step and a stratification step to compute point sample sets with the least oversampling. The algorithm results in a rendering time independent of the geometric model's complexity.
[94] provides a probabilistic low-dimensional creation of shape poses using Gaussian Process Dynamical Models. New motion variants are created by sampling trajectories using the Monte Carlo Markov Chain. It synthesizes an infinite number of variants of any of the input movements and any blended version without requiring costly non-linear interpolation of input movement variations in the mesh domain. The output variations are statistically similar to the input movements but differ possibly in poses and timings. The goal is to generate infinite variations for each blended motion (Figure 2.8) and (Figure 2.9).
The resulting mesh matches the training samples in realism, facilitating 4D model dataset augmentation. To produce a diverse heterogeneous crowd, [53] proposes a multi-agent reinforcement learning-based approach and provides varied input settings to learn generalized crowd steering and present a variety of crowds.

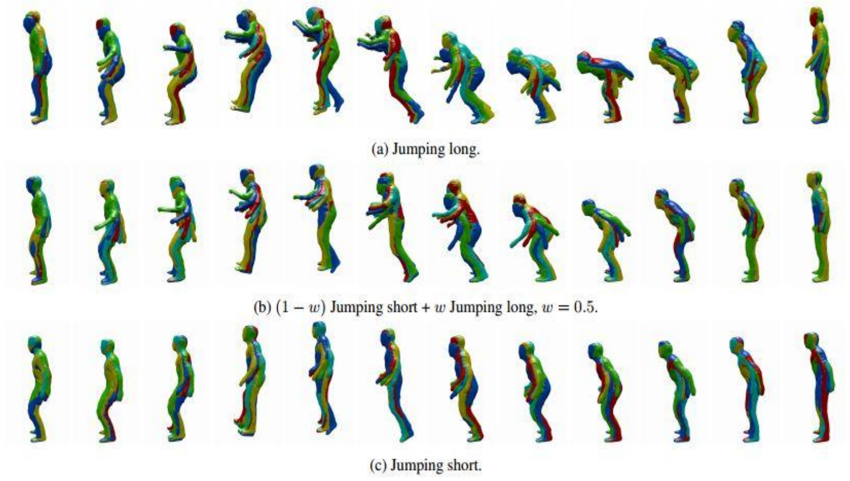


**Figure 2.8**:Jumping variations overlapping, each with a different color [94].

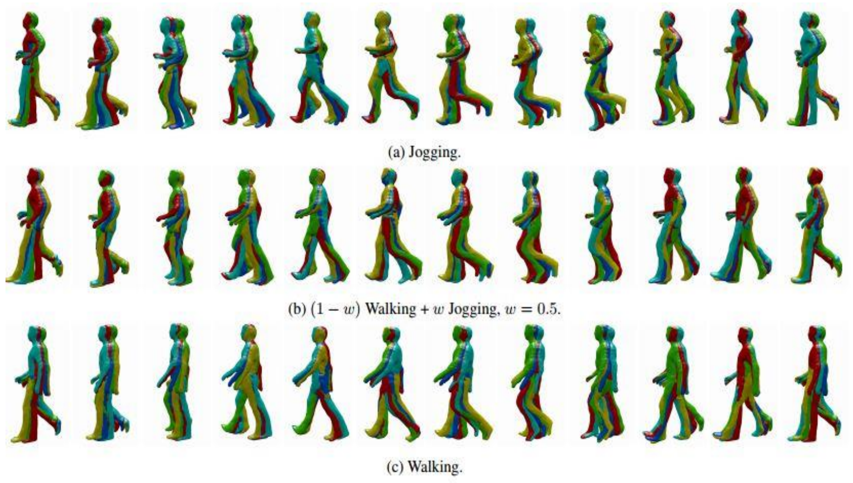


**Figure 2.9:** Locomotion variations overlapping, each with a different color [94].

### 2.3.2 Animation synthesis

Many studies have been performed in motion synthesis, particularly in the context of walking locomotion. The methods used can be divided into procedural techniques, physics-driven approaches, and example-driven procedures [2].

#### 2.3.2.1 Physics-based techniques

Physics-based algorithms provide animations that capture realistic forces on joints by focusing on dynamics and physical property principles [84], [85]. One significant disadvantage is the limited level of control over the animation, as well as the processing expenses involved. Because of the complex calculations, they are unsuitable for real-time applications where swift responsiveness is a critical requirement [2].

#### 2.3.2.2 Procedural techniques

Procedural techniques include creating motion from the base and applying scientific and biomechanical concepts. While these methods enable a high level of control, they often fall short in perceived realism [2].
[82] proposed a biomechanical-based human walking model that uses knowledge of body structure, joint mechanics, and motion dynamics to create a realistic representation of human locomotion. Despite the exact control over animation, the results appear unnatural. Notably, procedural techniques may impact specific types of movements, such as running [83]. Notably, procedural techniques may have impact on specific types of movements, such as running [83].

#### 2.3.2.3 Example-based techniques

Example-based techniques in motion synthesis involve applying pre-existing motions to produce a locomotion sequence. These motions are typically taken from motion capture clips to improve the natural quality of the results.

The process involves concatenating clips to form novel sequences or employing blending and parametrization to fuse or merge them, ultimately synthesizing entirely new motion clips [2].

- **Transitioned motion fusion**

Motion concatenation is effectively creating motion clips with transitional features between them. This approach produces highly realistic motions because no part of the sequence is created or entirely novel [2].

An alternate method called motion patches is used to synthesize interactive motions between different characters [86]. This method uses deformable motion patches that are covered spatially and temporally. Each motion patch is a collection of motion fragments encapsulating character interactions (see Figure 2.10).

Considering the complexity of these interactions, the resulting motions provide an efficient simulation of virtual characters interacting with one another in a specific way, making them unsuitable for traditional crowd simulation modules [2].

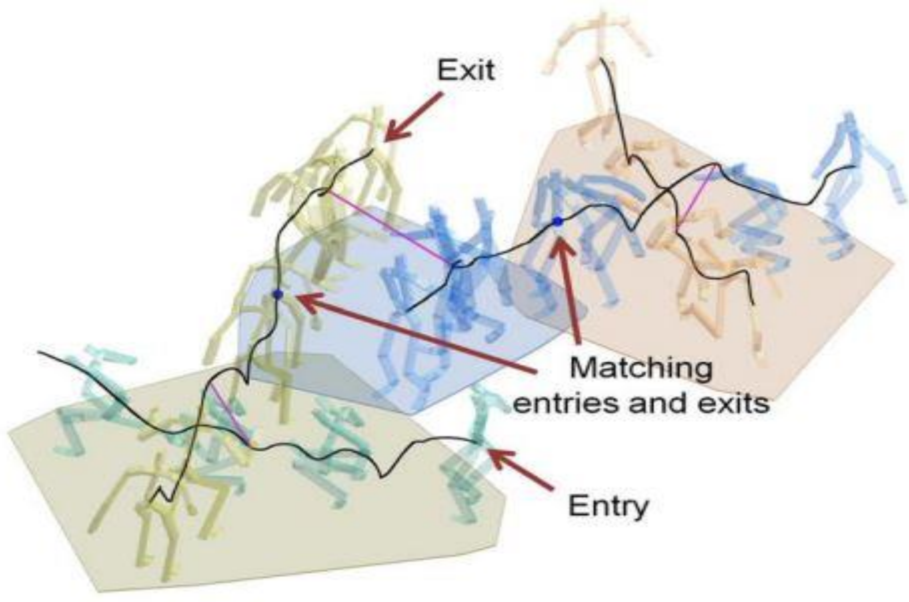


**Figure 2.10**:Synchronizing the corresponding entry and exit locations of deformable patches is required while connecting them. As depicted by [86], each patch encapsulates its motion paths projected onto the ground as convex polygons.

- **Parameterized motion synthesis**

Techniques such as motion parametrization and motion mixing are used in dynamic motion interpolation. These algorithms build novel sequences based on specific parameters, such as an end effector position, by interpolating between several pre-existing motions [2]. [87] For example, proposed parametric motion graphs allow a higher level of control than alternative paradigms (see Figure 2.11). The survey released by [88] provides a comprehensive review of motion combining and interpolation approaches.

- **Retrieving movements**

There are two primary methods in the process of incorporating motion clips. The first involves manual creation by skilled artists using software like 3D Studio Max [89] or Maya [90]. While this approach offers accessibility and flexibility, it is time-consuming and highly reliant on the artist's expertise.

On the other hand, motion capture systems provide an alternative. Although they may be less accessible, the resulting clips are generally more realistic and natural than those created manually [2].

Motion capture systems can range from expensive, high-quality results, such as those used in movies (see Figure 2.12) or high-budget video games, to low-cost solutions with bad results, such as those using the Microsoft Kinect camera (see Figure 2.13). These systems can also be classified into those that use markers to obtain actor positions and those that do not, using computer vision techniques or depth cameras. It is also worth noting that motions recorded straight from a motion capture system have much noise that needs to be cleaned out [91], [92]. Fortunately, the web contains some excellent databases of motion capture recordings that are freely accessible to the public. For example, consider the Carnegie Mellon University database [93], which has thousands of motions that categories have accurately classified.

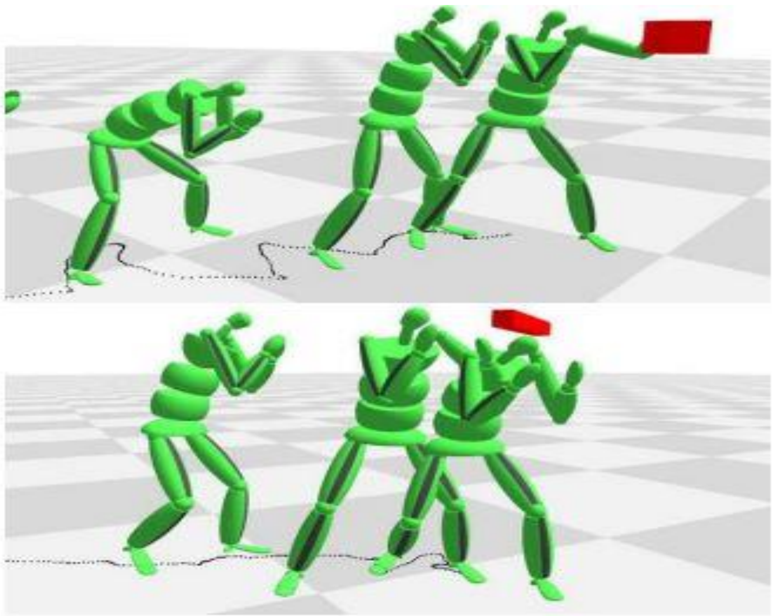


**Figure 2.11**:An interactively boxing character employing parametric motion graphs, executing a user-directed punch in the top image and seamlessly ducking below a user-specified height in the bottom image [87].

## 2.4 Collision avoidance

Collision detection is an important aspect of game creation because it allows game components to interact with one another in a realistic and evident way. Collision detection in 2D games is often performed by looking for overlaps between the bounding boxes or forms of game objects. However, as games have become more complicated, improved algorithms and strategies have been developed [95].

When we entirely focus on crowds, we run into challenges. For example, collision avoidance among many individuals in the exact location necessitates different resolving methods to avoid collisions between characters. Collision avoidance is a key aspect to consider if we want to simulate the crowd system. Using collision avoidance will increase the realism of the crowd system while also making it more interesting [96].

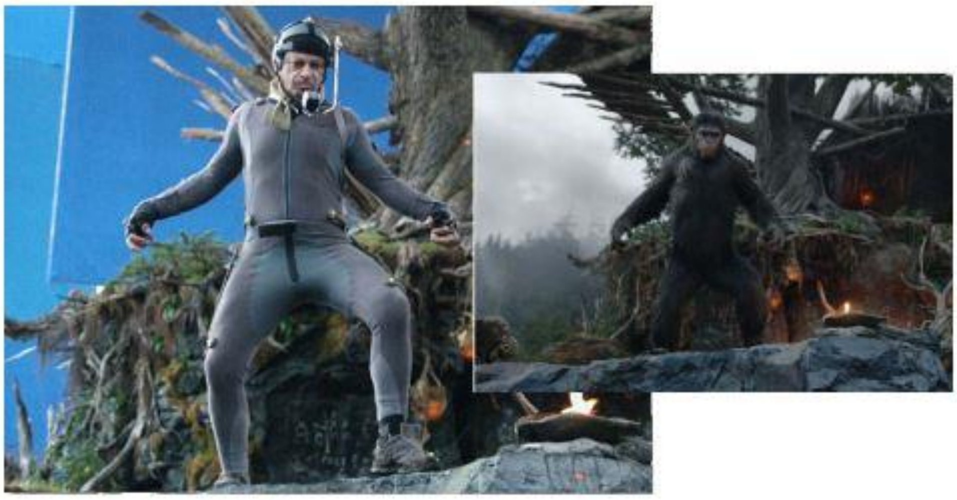


**Figure 2.12:** Andy Serkis as Caesar in the 20th , 2014 Fox film "The Dawn of the Planet of the Apes." The actor's performance is captured by Weta Digital's complicated motion capture technologies, which involve numerous cameras and markers (left), and then transferred transferred to its digital character (right) [2].

**Figure 2.13**:Microsoft Kinect 2.0 is an inexpensive depth camera. One of its uses could be markless motion capture [2].

### 2.4.1 Research on collision detection within crowds

Collision detection is an essential feature in game development that determines when two or more objects in the game world collide or intersect. It is essential for creating realistic and immersive game experiences. If collision detection is unsuccessful, the sense of realism and immersion in gaming decreases [95]. Numerous techniques have been suggested to attain collision avoidance behavior in crowd simulation. Craig Reynolds first suggested flocking as a crowd behavior in crowd simulation [17] and introduced a combination of cohesion, separation, and alignment. A collision avoidance mechanism and separation behavior are provided. After that, he created collision avoidance, obstacle avoidance, and path following for steering behavior [97].

Helbing created Social Force [23], which simulates the pedestrian boundary between individuals in crowd simulation. According to this study [96], collision avoidance in crowd simulation mainly focuses on agent-based designs that focus on factors such as the agent's position, distance, direction, and velocity.

Steering strategies consider the locomotion rules of an animation system. Paris and Donikian [112] present an approach in which the animation module can potentially inform steering that an action is not feasibleMusse, Thalmann [99], Shao, and Terzopoulos [113] solve higher-level pedestrian challenges; their navigation modules provide a range of navigation behaviors that apply directly to animations the character may generate. Van Basten and Egges [114] examine the difficulties of integrating navigation with animation, offering concepts to decrease such disparities. [115] Presents a method for the problem of generating crowd motions with shape preference based on a user-defined goal. The approach consists of two steps:

First, generate a rough path for the desired shape template and then use fuzzy rules to guide the crowd's movement. The method generates a path for the shape template based on the goal configuration and exacting shape requirements, which may not be completely collision-free. The crowd will be guided along the path using a template with several fuzzy rules. The crowd motion should follow a reasonable route and maintain the desired shape to the greatest extent possible (Figure 2.14) and (Figure 2.15).

To ensure safe navigation in a marine environment with nearby objects, [103] proposed a local collision avoidance algorithm for uncrewed surface vehicles (USVs), which is based on a steering strategy that corresponds to the International Regulations for Preventing Collisions at Sea (COLREGs). The algorithm in this study is divided into three sections: collision risk assessment, steering occasion determination, and navigation waypoint update.

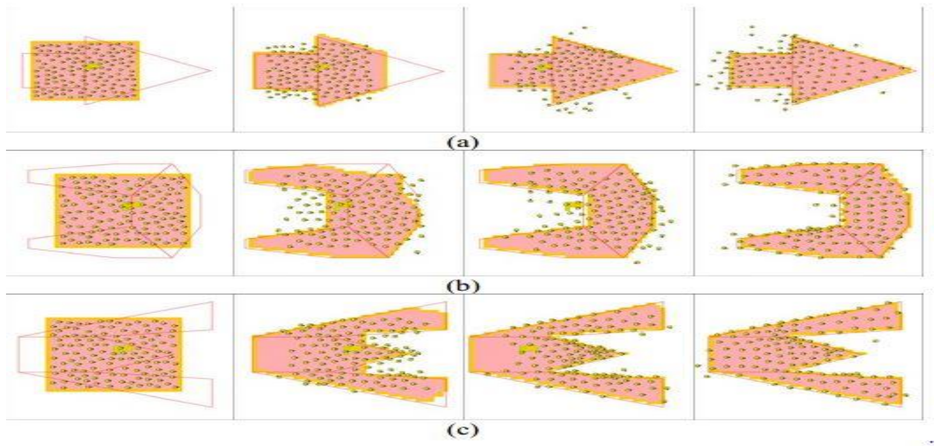


**Figure 2.14:**Examples of crowd motions that correspond to a specified shape [115].

The closest point of approach method is used to estimate a collision risk based on an analytically reasonable angle range. The steering occasion determination gives a safe distance interval from an obstacle by determining a supplementary steering angular velocity.

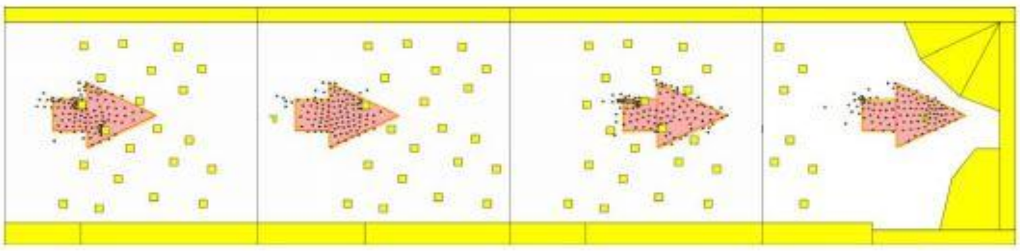


**Figure 2.15:**An example of crowd movement through obstacles in an arrow-shaped structure [115].

The navigation waypoint update provides a temporary waypoint to navigate the USV while considering COLREG passing, direct, and crossing regulations. Finite-state machines are used to solve these three parts. [104] used a rule-based Bayesian Network technique to model the collision risk of ships and marine buildings.

Although the rule-based approach has clear logic, high visibility, and stability, it results in irregular ship behavior due to the state-cutting condition. It is easy to have conflicts between the causing conditions of a behavior, resulting in system failures. Additionally, there are obstacles in the processing of complex working situations as well as the development of algorithm performance [105] [106].

Many researchers have developed soft computing approaches to solve these difficulties, such as genetic algorithms [107], velocity Obstacle [108], fuzzy logic [109], geometric calculation [110], and model predictive control [111]. Crowd simulation collision avoidance strategies may include short-term avoidance [98] and multi-resolution techniques [99]. Reciprocal velocity Obstacles (RVO) and Hybrid RVO [100] are approaches to collision avoidance behavior that can be implemented in real time for significant individual contexts. Motion Oriented Bounding Box (MOBB) [101] is another crowd simulation technique based on the bounding volume technique. The authors in [156] created a field of vision for each character to demonstrate a hybrid collision detection method. Their three-stage collision avoidance behavior approach begins with constructing perception, producing a bounding volume representing the agent's range of vision and mobility. The handling of collisions comes next, followed by the collision response, dependent on perceived velocity.

### 2.4.2 Approaches of collision avoidance concepts

Computer games and training simulations are among the rapidly growing computer uses nowadays. Real-time simulations, such as crowd simulation, are becoming more necessary due to these concerns [96]. Collision avoidance is a key consideration in crowd simulation. Using collision avoidance instead of collision detection has no extra costs.

Detecting and resolving collisions in crowds is challenging and varies by size. Many researchers used the collision avoidance strategy to represent flocks, herds, and large crowds. Flocking is a technique developed for modeling crowds of animals or humans [17]. Follow three rules of thumb: steer away from neighbors and obstacles, steer towards related entities' centers, steer at an average velocity with the group, and maintain direction with neighbors.

In crowd simulation, there are two types of obstacles: static and dynamic objects. Static items, such as buildings, furniture, and trees, are immobile, while dynamic objects, such as people, animals, and cars, are moving. Avoiding collisions between multiple people in the same location necessitates various strategies than avoiding collisions between two people. In Crowd Simulation, Humans avoid obstacles in their environment in a complicated way. Sometimes, they wait for others to move first; sometimes, they move aside while still moving. Collision avoidance is a fundamental problem in crowd simulation [96].

Crowd simulation without collision avoidance reduces realism. A world without collision avoidance would be terrifying, resembling a ghost city where individuals and obstacles pass directly through each other. It would be a confusing and disappointing situation. Collision avoidance must be implemented to provide a realistic and efficient crowd simulation. Previous studies have used a variety of collision avoidance strategies. For example, consider the long-term avoidance predicts and prevents collisions. Short-term avoidance is used when long-term avoidance fails to prevent a collision. The objective is to control thousands of individuals in real-time. It is a low-level agent-based method [98].

long-term avoidance produces less believable results than agent-based approaches since they fail to account for the individuality of each pedestrian. However, they have significantly reduced computing costs. [98] presents a hybrid architecture for real-time crowd motion planning that remains realistic and scalable. Using a navigation graph, it divides the environment into sections of varying interest. In regions of great interest, it uses a potential field-based strategy. Because it only uses it locally, this method can plan motion for many more groups and smaller grid cells than with an algorithm purely based on it. In other regions, motion planning is guided by a navigation graph and short-term collision avoidance algorithms.

[99] presents an approach to crowd behavior that considers the interaction between groups of individuals and the resulting emergent behavior. It treats individuals as autonomous virtual beings that change their parameters based on their environment and provides a multiresolution collision approach designed for crowd modeling.

Crowd simulation has proven effective in movies like The Chronicle of Narnia and The Lord of the Rings series. The game features a highly realistic crowd simulation that enhances realism. Without it, players could encounter unrealistic scenarios, reducing their interest level. A well-designed crowd system is essential to prevent collisions and maintain a dynamic, exciting experience [96].

#### 2.4.2.1 Collision detection

Collision detection in current game production is a complicated process that necessitates an excellent knowledge of the physical features of objects in the game environment and the computational algorithms used to simulate their behavior. There are numerous ways to detect collisions, each with benefits and disadvantages. [95].

Collision detection is an important technique for avoiding collisions between objects. Collision detection can represent an object's distance, direction, and velocity to determine the estimated location of an object over a certain number of times, accordingly predicting intersection and executing collision avoidance action [102]. The analysis of when two objects collide or make contact with each other is an essential aspect of computer games, implying that the interaction between objects is a fundamental element of computer games.

In computer games, correctly detecting interactions is important because these interactions determine the gameplay experience. In an action game, for example, the bullet hits a player, or the hero cannot pass through the maze wall. In basic terms, collision detection aims to identify whether the images in two dimensions of two or more objects overlap one other by an algorithm. To specify further, the challenge is slightly more complex: detecting whether one object contains a pixel that overlaps with a pixel of another object [95]. A critical decision regarding the collision detection system or model must be made at some point through the game development process. The choice is not always straightforward and uncomplicated. There are several various types of games where interactions can be highly complex, and not all issues are always evident [95].

However, the model used is essential as it significantly impacts both development time and the gaming experience itself. Collision detection systems can be categorized into two types:

- Pixel-based collision detection: checks the overlap of the pixels of the images corresponding to the colliding objects. It is capable of detecting a precise, real-world collision.
- Bounding object-based collision detection: involves determining the intersection of objects using enclosing shapes (such as boxes, circles, and polygons) rather than individual pixels. In general, the method does not provide exact collision detection.

Depending on the complexity of the object's texture, pixel-based collision detection can be computationally intensive and complex. Consequently, game developers often aim to enclose the moving elements in simpler objects and apply collision detection to these whenever possible. Circles or boxes are commonly chosen due to their simplicity. The collision testing, rotation, and translation calculations using these shapes are far less computationally expensive than methods like bounding polygons or pixel-level tests. While these shapes may not perfectly match the object's proper form, they serve a practical purpose in game development [95].

## 2.5 Synthesis of related literature

| **Author(s)/ discipline** | **Approach /Model** | **Contributions** | **Advantages /Disadvantages** |
|---|---|---|---|
| -McDonnell et al [2] (2008). -Computer Graphics -Human-Computer Interaction | -Experiments assessed human sensitivity to cloned appearances, varying color, orientation, and animations. | -Variations in color enhance variety. -Optimizing crowd rendering | -Improves the perception of crowd variety -Results from experiments lack realism. -Focuses on visual diversity and excludes interaction. |
| -Boukhayma, A et al. [3] (2016). -Computer Graphics. | -Developed a method to generate Eigen appearance maps for representing dynamic shapes. -Used 3D reconstruction for temporal shapes. | -Principal Component Analysis (PCA) was applied to identify the structural relationships between appearance and shape. | -Offer dynamic shape more efficiently -Limited performance in shape changes. -High computational cost for extensive, complex data. |
| -Shi, Y. et al [4] (2017). -Computer Graphics -Virtual Reality -Crowd Simulation. | -Data-driven approach generates realistic, distinct body shapes. | -Technique generates perception based on real-world measurements. | -Generated diverse body shapes, - Focused on body shapes, excluding dynamic interaction. -Limited generalizability for large-scale complex crowd simulations |
| -Johansson, D and Ravantti, R [5] (2020). -Computer Graphics. | -Experiment in which viewers are asked to perceive and evaluate simulated crowds containing cloned characters. | -Examined cloned character's impact on realism in virtual crowds. | -Demonstrated how to recognize clones, -It involved only 14 participants, which limits the generalizability of results. -Focus only on clones' distribution, neglecting other factors like /clothing variations. |

| | | | |
|---|---|---|---|
| -Reuben, F et al [44] (2016).<br>-Computer graphics.<br>-Computer vision<br>-Human-computer interaction. | -Explored creating appealing female avatars through 3D body scans. | -A proposed framework for creating appealing avatars with stylization techniques.<br>-Explored balancing realistic body representations and enhanced stylization. | -Provided a framework for appealing female avatars, balancing realism and stylization.<br>-The study focused on female avatars, limiting generalizability to other avatars.<br>-Perception of attractiveness varies, challenging universal appeal without cultural sensitivity testing. |
| -Ruiz, S. et al [72] (2013).<br>-Computer graphics.<br>-Animation. | -Proposed a memory-efficient technique for simulating diverse crowds, using animation parameterization, compression, and resource sharing to reduce computational load. | -Reduced memory for efficient large-scale crowd simulations.<br><br>-Enable scalable crowd simulations for real-time, large-scale virtual environments. | -Reduced memory for real-time, diverse crowd simulation efficiently.<br>-Optimized memory preserves diverse, realistic, engaging crowd animations<br>-Reducing memory usage may limit individual behavior complexity and interaction variety through animation parameterization and instancing<br>-Compression techniques reduce memory but may introduce artifacts, impacting animation quality and crowd smoothness. |
| -Boukhayma, A., and Boyer, E [94] (2017).<br>-Computer vision.<br>-Motion capture. | -Proposed a method to synthesize controllable variations in motion capture data, preserving realism with statistical modeling. | -Generated realistic motion variations, enhancing efficiency and flexibility without extra capture sessions.<br><br>-Provided a tool for motion capture, enabling flexibility and reducing variations capture needs. | -Enable diverse motion variations, saving time and resources in animation production.<br>-Provided control for customizing motion and simulations.<br>-Parameterizing motion capture and controlling variations can be complex.<br><br>-Less effective for complex movements. |
| -Wang, D, et al [103] (2021).<br>-Marine robotics.<br>-Control systems engineering. | -Proposed a COLREGs-compliant collision avoidance algorithm for USVs, featuring dynamic obstacle detection, steering strategies, and optimized path adjustments for safe, efficient navigation. | -Presented an innovative collision avoidance algorithm for uncrewed aerial vehicles (USVs), following the International Regulations for Preventing | -Algorithm ensures COLREGs compliance, reducing collisions.<br>-Managed dynamic scenarios, enhancing USV effectiveness.<br>-COLREGs integration may increase computational demands, limiting performance. |

| | | | |
|---|---|---|---|
| | | Collisions at Sea (COLREGs).<br><br>-Ensuring secure maritime navigation.<br><br>-Provided a real-time steering framework that accounts for the dynamic behavior of other vessels.<br><br>-Increased the operational safety and navigation efficiency of USVs. | -Environmental conditions limit the algorithm's real-world applicability. (weather, wave interference). |
| -Shaobo, W et al[108] (2020).<br>-Marine robotics<br>-Autonomous systems. | -It adapted the traditional velocity obstacle concept for autonomous ships, using real-time sensor data to calculate safe velocity space while incorporating decision-making algorithms for collision avoidance. | -Modified velocity obstacle method improves autonomous ship navigation.<br><br>-The decision-making system ensures collision avoidance and balances safety and efficiency. | -Modified VO method enhances collision avoidance in autonomous ships.<br>-Ensuring safe autonomous navigation in dynamic conditions.<br>-Modified VO algorithm may increase computational complexity<br>-Requires further testing, especially in unpredictable or extreme conditions like poor weather. |

**Table2.1**: Analysis of specific related works .

This section synthesizes significant results from previous studies on crowd simulation, focusing on how to represent crowds in ways that reflect the characteristics of humans, preserve their appearance, and model their movement to ensure realistic and practical simulations. Researchers have extensively investigated methods to enhance crowd differentiation and recognition, mainly through appearance and motion. For instance, [2] demonstrated that variations in color and animation enhance diversity, while [3] proposed a method for generating Eigen appearance maps to represent dynamic shapes. Additionally, [4] employed a data-driven approach to create realistic and distinct body shapes, and [44] explored 3D body scans to develop visually appealing female avatars. These findings demonstrate the importance of appearance diversity.

Furthermore, [94] introduced a method to synthesize controllable variations in motion capture data, maintaining realism through statistical modeling. Similarly, [103] presented an innovative collision avoidance algorithm for uncrewed surface vehicles (USVs) following COLREGs, and [108] improved autonomous ship navigation by modifying the velocity obstacle method. These studies emphasize the significance of animation diversity.

However, despite these contributions, it is evident that each study focuses on a distinct component. The integration of appearance diversity, motion variety, collision avoidance, and approaching methods into the same system remains to be realized. It emphasizes the necessity for a comprehensive approach that combines these elements. Accordingly, our study aims to build results by simulating crowds with diverse appearances and shapes, incorporating accessories to enhance individuality, and applying varied movements at different speeds. Additionally, the Raycasting method, physical rules, including Newton's laws, will be implemented using to ensure effective collision avoidance and realistic interactions, enabling the simulation to achieve a higher degree of realism and offer significant insight into developing realistic environments.

## 2.6 Conclusion

This chapter examines different works and efforts related to the diversity of appearances in crowds. It includes a combination of shape, clothing, accessories, color, and texture. We also focus on various works and efforts associated with crowd animation. We introduce the concept of animation synthesis and categorize existing methods into three categories: procedural techniques, physics-driven approaches, and example-driven procedures. Subsequently, we determine three distinct processes: transitioned motion fusion, parameterized motion synthesis, and retrieving movements. In addition, we present various works and efforts related to collision avoidance and introduce the concept of collision detection. Then, we determine its two categories of systems: pixel-based collision detection and bounding object-based collision detection. Finally, we delve into a discussion of the related works, indicating their methods, contributions, advantages, and disadvantages.

# Chapter 3

# Modelling crowd variety

## 3.1 Introduction

Simulating several crowds is essential for real-time applications like games, animation movies, and navigation systems. These applications can include features and details that improve the appearance of the virtual environment and its components while also increasing the system's realism. This chapter focuses on enhancing diversity within crowd simulations by exploring various concepts and techniques. It is essential in crowd simulations to include diverse appearances among crowd characters; otherwise, the simulation may appear unreal. We aim to produce convincing results that demonstrate various forms of appearance difference, such as clothing, color, and texture diversity, in order to differentiate between them and eliminate the vision of visual replications within the crowd while taking into account diverse human genders, such as women, men, young children, and an older adult. Each individual is attired in distinct clothing, ensuring that every model possesses a unique color and texture map encompassing variations in hair, skin, body, and accessories. Then, we present our crowd animation approach to provide a different range of motions by building locomotion cycles for each model to make various animations in our scene.

Additionally, we detail a method to prevent collisions with our characters. The proposed algorithm is grounded in Newton's laws, the laws of momentum, and the conservation of kinetic energy without external forces. This approach yields reliable results for determining accurate values of the characters' final velocities.

## 3.2 Modeling virtual human variation

The simulation of the crowd has to contain different characteristics that can be used to present a real crowd rendering in the virtual scene. The aim is to merge the items mentioned in (Figure 3.1) to get the desired result. Diversity must be applied to create a crowd to realize each character's individuality.

We present the diversity of our characters through a combination of the following concepts:

- Appearance variety, featuring hair mesh variation, shape and height, clothing, accessories, colors and textures [155].
- Animation variety, allowing specific motions for each character while ensuring collision detection and avoidance.

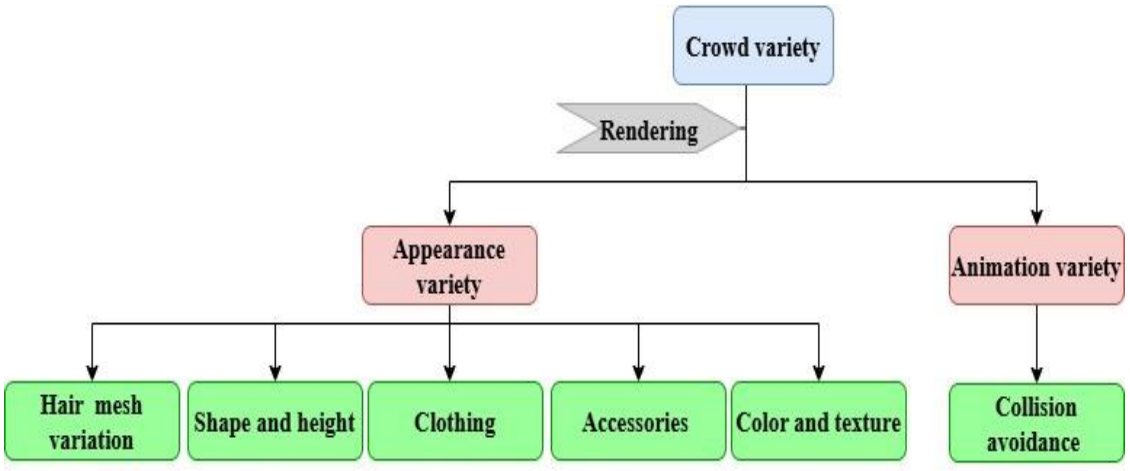
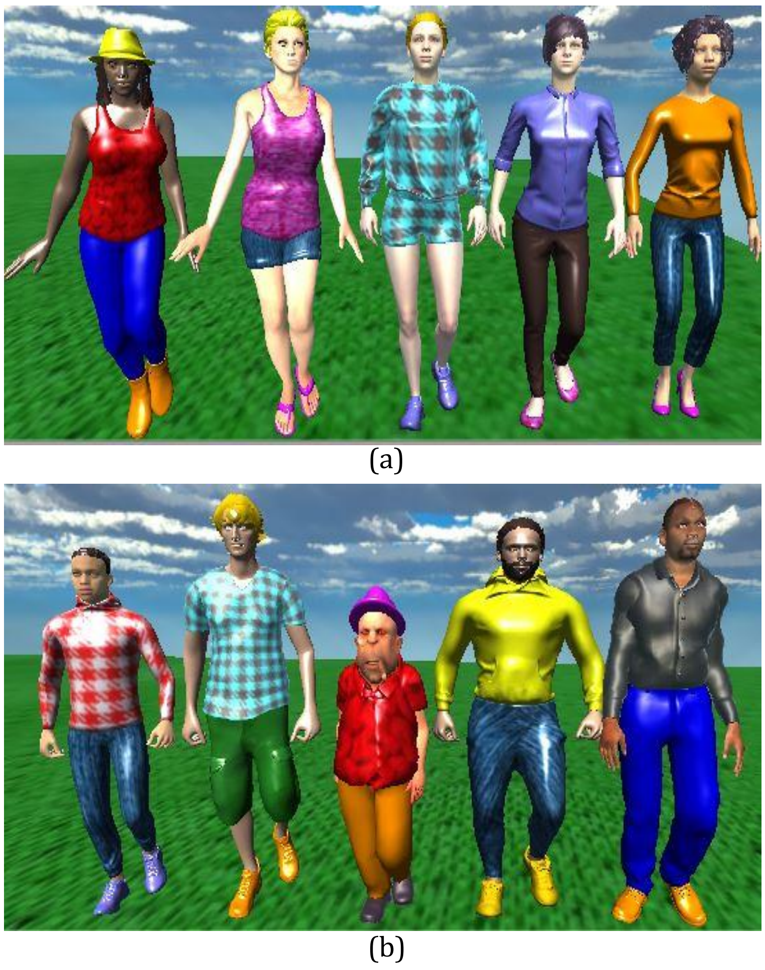


**Figure 3.1**:Modeling virtual human variation.

The proposed approach involved various types of virtual humans. We utilized ten women, five men, two young children, and an older adult from the Mixamo site database [54]. Each model presents a trait to ensure a variance in clothes, hair, beard, skin, shoes, and accessories aligning with various textures, images, and colors. Moreover, variations in body shapes and heights are implemented further to enhance the realism and uniqueness of each character. Then, we perform different types of animations on these models to get the variety of animations and enrich their movements within our environment. Additionally, collision avoidance algorithms are employed between the different individuals in the environment to increase the crowd system's realism while ensuring that interactions between individuals within the crowd are realistic.

## 3.3 Appearance variety

This section presents the visual characteristics of virtual characters and technical aspects that influence character appearance design. We need to investigate how various forms of appearance variation, including shape, clothing, color, and texture, impact the discernibility of visual clones within crowds, considering diverse human genders such as women, men, young children, and older adults (Figure 3.2). Each individual is attired in distinct clothing, ensuring every model possesses a unique texture map encompassing variations in clothes, skin, and accessories. This approach allows us to manipulate these terms, aiming to effectively distinguish individuals from one another in simulated crowd scenarios mirroring real-world situations [155].

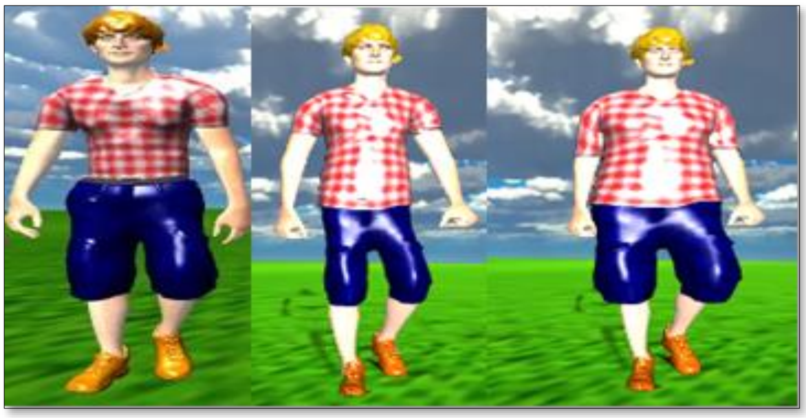

(a)

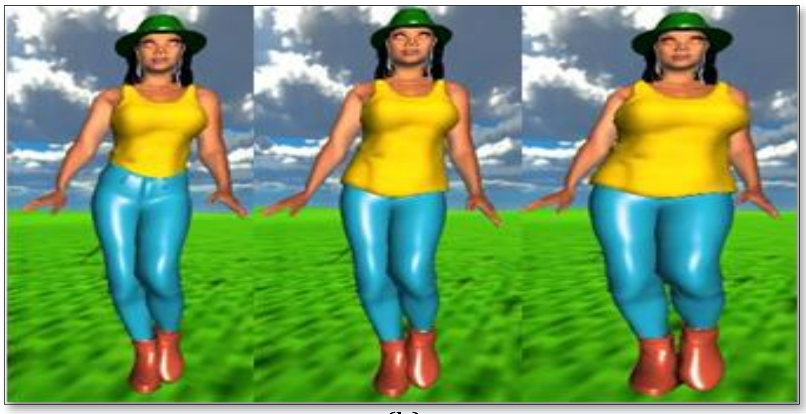
(b)

**Figure 3.2:**Examples with different templates of the crowd:(a) five women, (b) a young child, an older adult, and three men.

### 3.3.1 Shape, clothing, and deformed accessories

To improve the variety of the shape and height of our models, we change the scale skeleton by specifying a range of height and shape values. This adjustment is applied to the x, y, and z axes, allowing us to create several skeletons from a single template.
We can also use a way deformer to increase the variety of our models and apply various effects to the components in our scene. It is helpful for objects and characters. To modify the various components of the model's body, we add deformers by using a set of script mesh deformers written in C#. We apply it on the mesh of clothes and shape and modify the shape of accessories by using the skinned mesh deformer tool (Figure 3.3) (Figure 3.4). Its features include meshing deformation and modification; using a set of scripts affects how the components of our virtual human change during animation. It identifies the model structures and achieves the transformation by attaching a skinned mesh to a script and assigning it to the associated mesh. Additionally, it is used to rig the skins of virtual humans so that they can access the mesh and the skeleton.


(a)


(b)

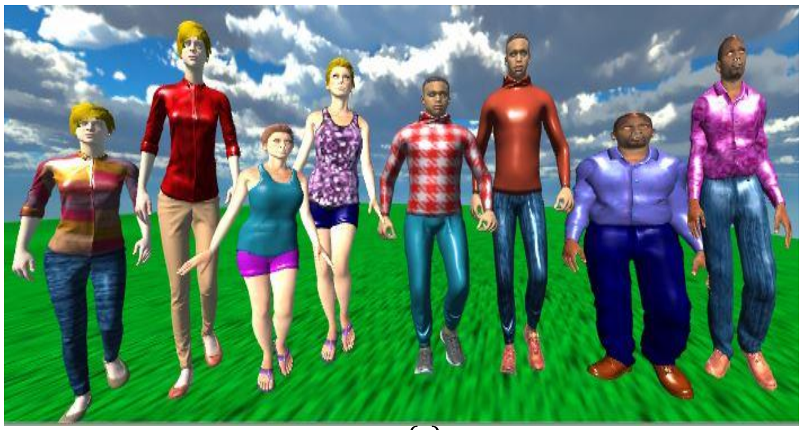

(c)

**Figure 3.3:**(a),(b) and (c) represent the deformed shape, height, and clothing mesh by scaling the skeleton and applying the skinned mesh deformer.

### 3.3.2 Colors and textures

The most effective approach for avoiding character duplication in a crowd is to use variations in color and texture. These components are essential in distinguishing identical characters within a scene, proving that diverse colors and textures successfully separate human character models that would otherwise appear identical.
To improve the variation's color and texture of character and render our crowd, we need to use the human template, which contains a skeleton consisting of joints, a set of meshes, and a set of texture images to vary its appearances like clothes, skin, hair, beard, shoes, hat, and eyes [155].

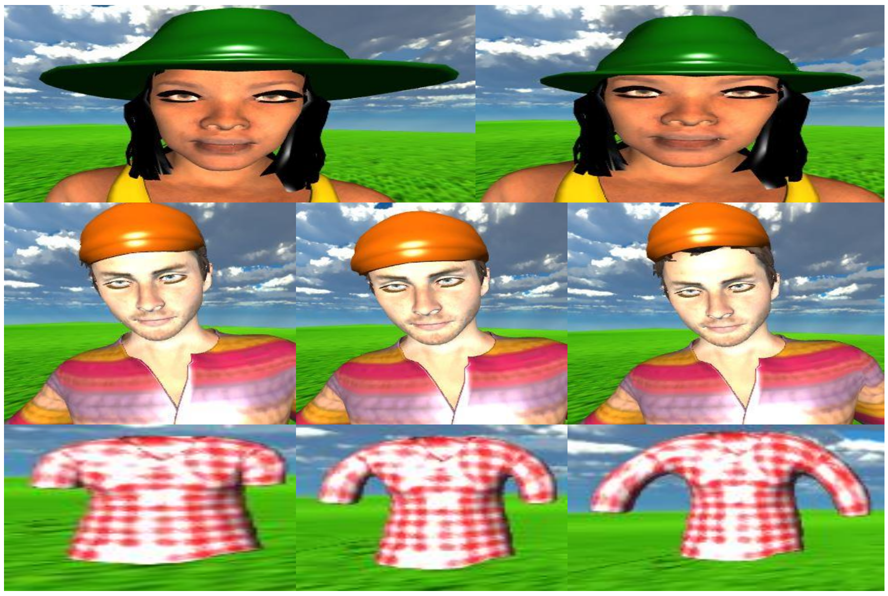
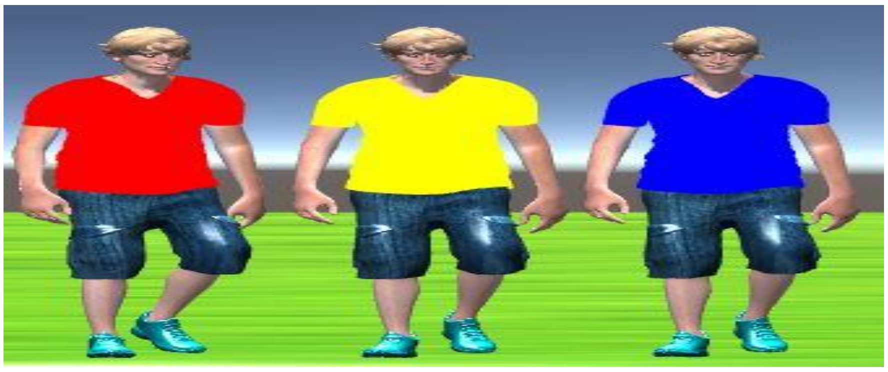


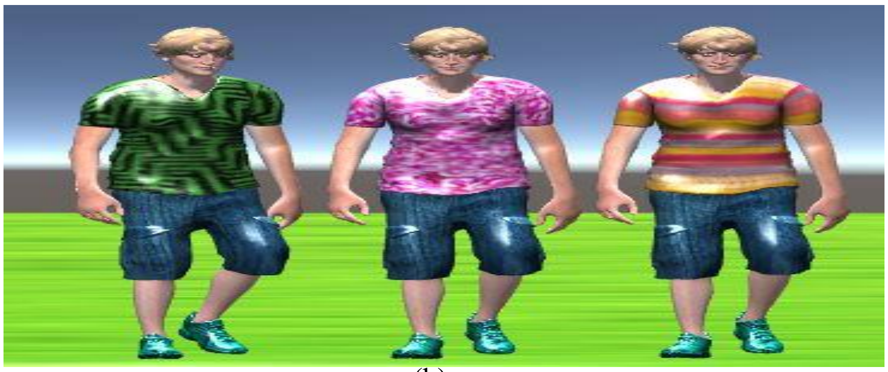
**Figure 3.4 :**Editing clothes and accessories by skinned mesh deformer tool.

To improve this approach, we start by loading our virtual humans into Unity3D, a game engine that supports cross-platform development, and then concentrate on changing essential features of their appearance.

We will discuss how our algorithm ensures diverse colors and textures. First, it concentrates on how to change the color and texture of the parts of our character. Each part's texture is replaced at runtime based on the number of loaded elements. Within our script, we have implemented classes C#. Each class affected a model to enhance the diversity of colors and textures for our models (Figure 3.5).



(a)



(b)

**Figure 3.5:**(a)(b) Example of various colors and textures outfits in the same template.

We declared a list corresponding to saving our game objects with their different parts representing essential elements like hair, eyes, beard, skin, shoes, hats, shirts, and pants. These parts variables link to the respective renderer component to allow access to the materials associated with each part.

To establish that textures and colors changed randomly, we have developed a function that dynamically alters the character's appearance. Various textures2D are stored in a designated folder, enabling us to load and apply them randomly to different material components of our character according to its path resources [155].

Additionally, we have developed a function to store these Texture2D assets in a dedicated array. This setup allows us to reference and apply specific textures using their index during runtime. Doing so allows us to efficiently manage and dynamically assign textures to different parts of the character as execution proceeds. The textures for each section are updated based on the corresponding index in the array that holds the selected assets [155].

This approach lets us easily swap textures during runtime, personalizing our character's appearance. We have also implemented a reasonable range with appropriate values to express colors of specific parts such as hair, eyes (Figure 3.6) and beard [155].

It ensures that a range of colors is applied to specific material components of our models. It is accomplished through a dynamic coloring process that assigns distinct hues to each element, enhancing visual diversity and offering players more customization options within the scene [155].

When our model is executed, we observe different models within our open environment, each distinguished by unique combinations of colors and textures. This diversity in appearance is achieved through our dynamic application of textures and colors to various parts of the models, showcasing a variety of visual characteristics that enhance the richness and realism of our environment.

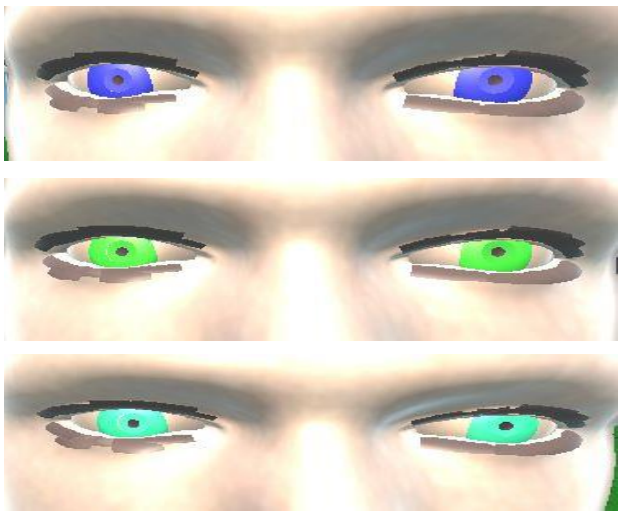


**Figure 3.6**:Variation iris color of eyes in same template

Next, we replicate each model on our list to instantiate them and create other characters that differ from the first characters in appearance. Each character is assigned to a determined location within the boundaries of our area, distinct from the positions of other characters. It ensures that our crowd is varied and realistically distributed around the area, improving our model's overall immersion and visual appeal (Figure 3.7).
The Raycasting technique calculates intersections between characters by projecting from the current character's position onto others. It ensures that each time we duplicate our models and increase their number, every character occupies a unique location within our environment by dynamically adjusting their positions based on these intersections.

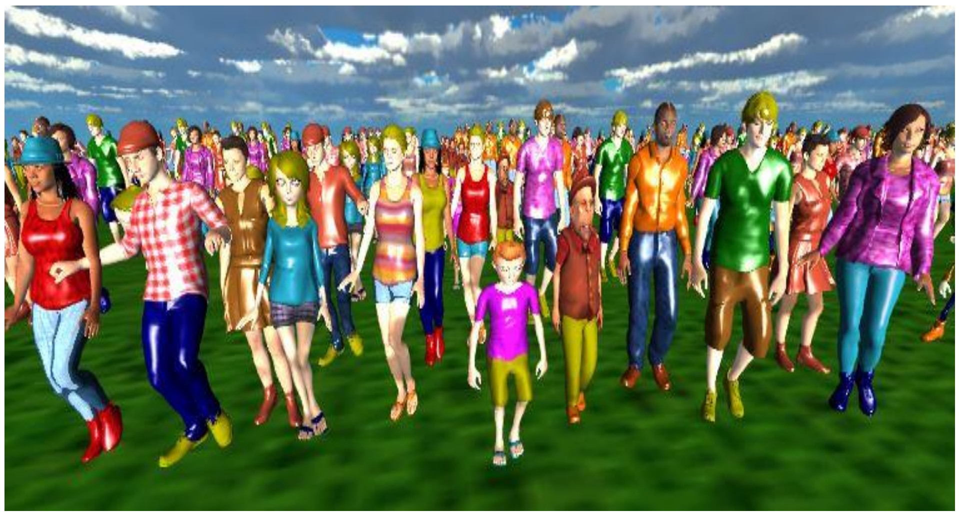


**Figure 3.7**:Enhance the diversity of colors and textures of specific parts of our different models and forming our crowd appearance variety.

In addition, we use scripting shaders by applying different basic illumination models such as diffuse reflection, specular reflection, and ambient light to give reasonably good results.
It is widely used in various graphics systems and provides a simple and efficient approach to determining surface intensity.

**Algorithm 1: Color and texture variety**

```
Input:  list of different templates of characters,
        different parts of characters (pants, shoes, eyes, beard, shirt,
        hat, hair, and skin),
        array of textures_images, range_colors.
Output: Different textures (pants, shoes, shirt, skin and hat).
        Different colors (beard, eyes, and hair).
-For each character do
    - Duplicate model.
    - Give a random position.
    - Avoid intersections.
    - Give a random different texture for specific parts
    - Give a random different color for specific parts
-End for
```

This algorithm uses the following denomination:
**GetComponent<Renderer>():** Access the Renderer component of a character to read or manipulate the character's material.
**Material.SetTexture:** Use SetTexture to change the texture in runtime.
**Material.SetColor:** Use SetColor to change the color of the material.
**GameObject.FindWithTag:** This function returns the first character it finds that matches the provided tag.
**Transform.GetChild(int_index).gameObject**: Returns the transformed child at the specified index. If the transform has no children or the index argument surpasses the number of children, an error will be produced.
**Instantiate (Object original, Vector3 position, Quaternion rotation)**: Clones and returns the original object. This function copies an element selected. Cloning a character, then we change its position and rotation.
**List_name.Add(Element):** Adds an object to the end of the list.

### 3.3.3 Rendering models under lighting conditions

To realistically represent our models and achieve high visual realism in our crowd environments, we must simulate them under various lighting conditions. This part of the simulation evaluates the effects of varying illumination.
We begin by applying the Simple Diffuser. In this case, the light reflects approximately equal in all directions in our scene. So, applying this type of illumination ensures a uniform illumination that spreads from any viewpoint in our scene when viewing our models.

The Diffuse Reflection is expressed mathematically as:

$$L^{diff} = 1/\pi . P^{*} L^{Light}.(n.l) \tag{3.1}$$

Where $L^{Light}$ is the light from a light source, $L^{Light}.(n.l)$ is the illumination at the surface with the normal vector $n$ of the surface and the illumination vector of light source $l$. $P^{*} L^{Light}.(n.l)$ is the overall reflected light, $1/\pi.P^{*} L^{Light}.(n.l)$ is the reflected light toward the viewer [68] (Figure 3.8).

We can also use Simple Specular to create bright specular highlights on our model so our models will appear brighter from specific angles. This reflection focuses on the light reflected by an object's surface in an aspect that depends on the angles of the incoming light vector and the viewer's perspective. It is written as:

$$L^{Spec} = P^{white*} L^{Light}.(n.l).(r.v)^{m} \tag{3.2}$$

Where $L^{Light}$ is the light from a light source, $L^{Light}.(n.l)$ is the illumination at the surface. $P^{white*} L^{Light}.(n.l)$ represents the overall reflected light. $P^{white*} L^{Light}.(n.l).(r.v)^{m}$ is the reflected light towards the viewer, $m$ is the exponent governing the size of the high light area, and $r$ is the reflection vector of light direction $l$ concerning normal $n$ [68] (Figure 3.9).

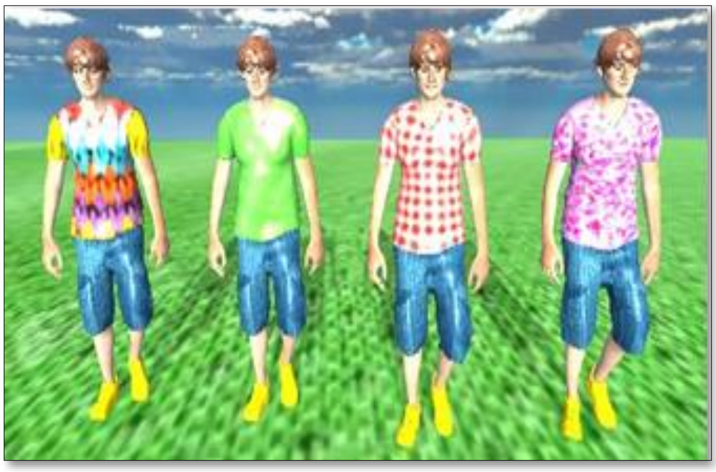


**Figure 3.8:**Applying a simple diffuse model on virtual humans.

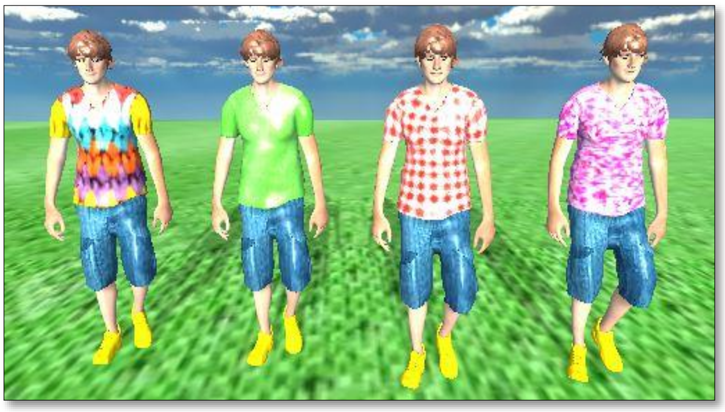


**Figure 3.9**:Applying a simple specular lighting model on virtual humans.

Another type of the basic illumination model is Ambient Light or background light like sunlight; this produces a uniform level of light, which indicates the lighting is constant from all directions. It is expressed mathematically as:

$$L^{Amb} = 1/\pi . P^* \, L^{indirect} \tag{3.3}$$

Where $L^{indirect}$ is the indirect illumination at the surface, $P^* \, L^{indirect}$ is the overall reflected light, and $1/\pi.P^* \, L^{indirect}$ is the light toward the viewer [68] (Figure 3.10). These shading models are used in computer graphics to obtain a realistic light rendering in the environment. For this study, we use the Phong model (Figure 3.11), a local illumination model applied to our characters to calculate the color of a point on a surface. It consists of a combination of three components: ambient lighting, diffuse lighting, and specular lighting. The Phong Shading Algorithm determines the normal at each polygon vertex. It includes the linear interpolation of the vertex normal over the surface polygon. The illumination model applies along each scan line to calculate the intensity of each surface point [69].

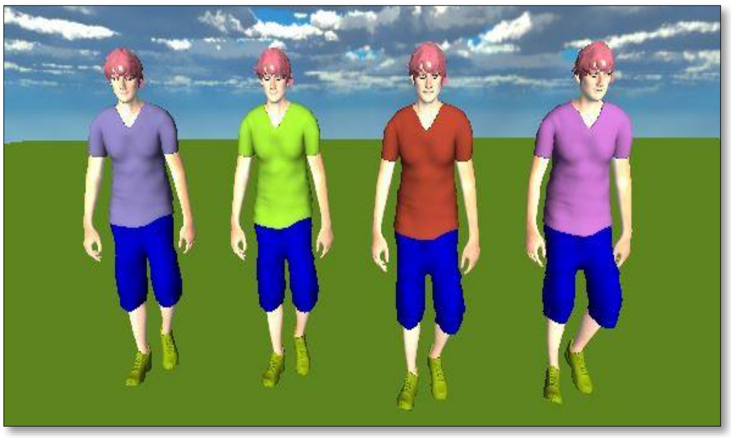


**Figure 3.10**: Simple ambient lighting model on the virtual humans.

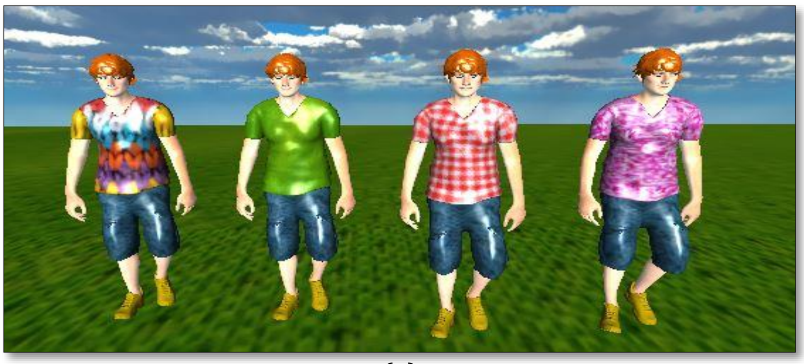
The combination of specular, diffuse, and ambient components forms the model of Phong. Typically, the Phong model is written as a function consisting of three terms, namely the ambient, diffuse, and specular reflection terms:

$$L^{phong} = \frac{1}{\pi}.P * L^{indirect} + \sum_i L_i^{light}.(n.l_i) * \left(\frac{1}{\pi}.P + P^{white}.(r_i.v)^m\right)) \tag{3.4}$$

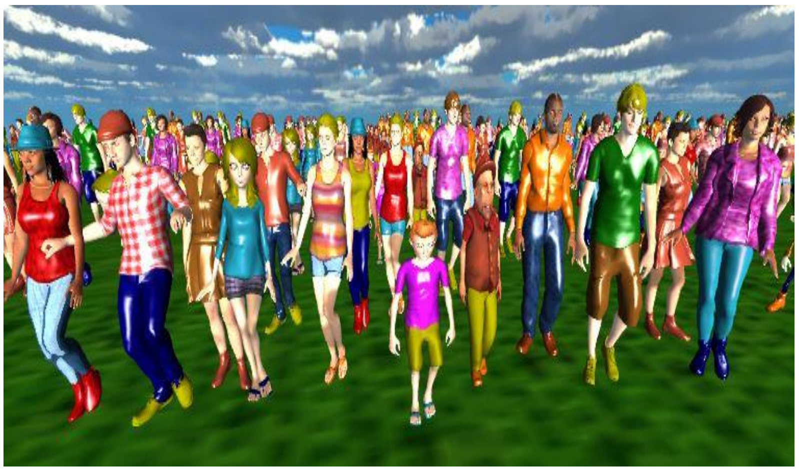
Where $\sum_i L_i^{light}$ is the multiple light sources and $(n.l_i)$ is the angle between the surface normal *n*, and light source *l* direction influence the surface brightness [68].

The Phong reflection model's ambient component provides an appropriate intensity level without direct illumination in our scene, preventing the models from appearing completely dark. The ambient light is constant over the surface. The diffuse component depends on the concept that light impacts our models from any direction, is reflected uniformly in all directions, and adds the object's ultimate illumination.

The specular component gives our models a shining appearance with highlights that reflect the viewer's viewpoint and light source position. This model illustrates the importance of realistic light rendering through its components.



(a)



(b)

**Figure 3.11:**(a) Applying the Phong model on the same virtual human with different textures and colors of clothes. (b) Applying the Phong model to different templates of our virtual humans with different textures and clothing colors.

### 3.3.4 Hair mesh variation

Real-time crowds are developing an interest in gaming because of improved levels of detail (LOD) techniques. Characters within crowd systems in games have detailed skinned meshes for their bodies and hair, improving virtual environments' realism and immersion [155].

Hair is essential in defining one's identity, offering significant insights into age, appearance, cultural heritage, and personality. Researchers are concentrating on perfecting hair modeling in computer graphics and virtual reality. We apply many hair mesh versions for our characters to provide rich appearance characteristics in our environment and show a diverse range of hairstyles in various colors. To achieve real-time variation in a character's hair mesh, we employ an array containing multiple hair meshes that can dynamically attach to the skinned head mesh of our model, ensuring each looks distinct and unique. These hair meshes are sequentially swapped during runtime, adding variety to the model's appearance [155].

Furthermore, we adjust the mesh's placement on the head and modify its orientation, shape, and color, allowing for further customization and realism in virtual representations (Figure 3.12).

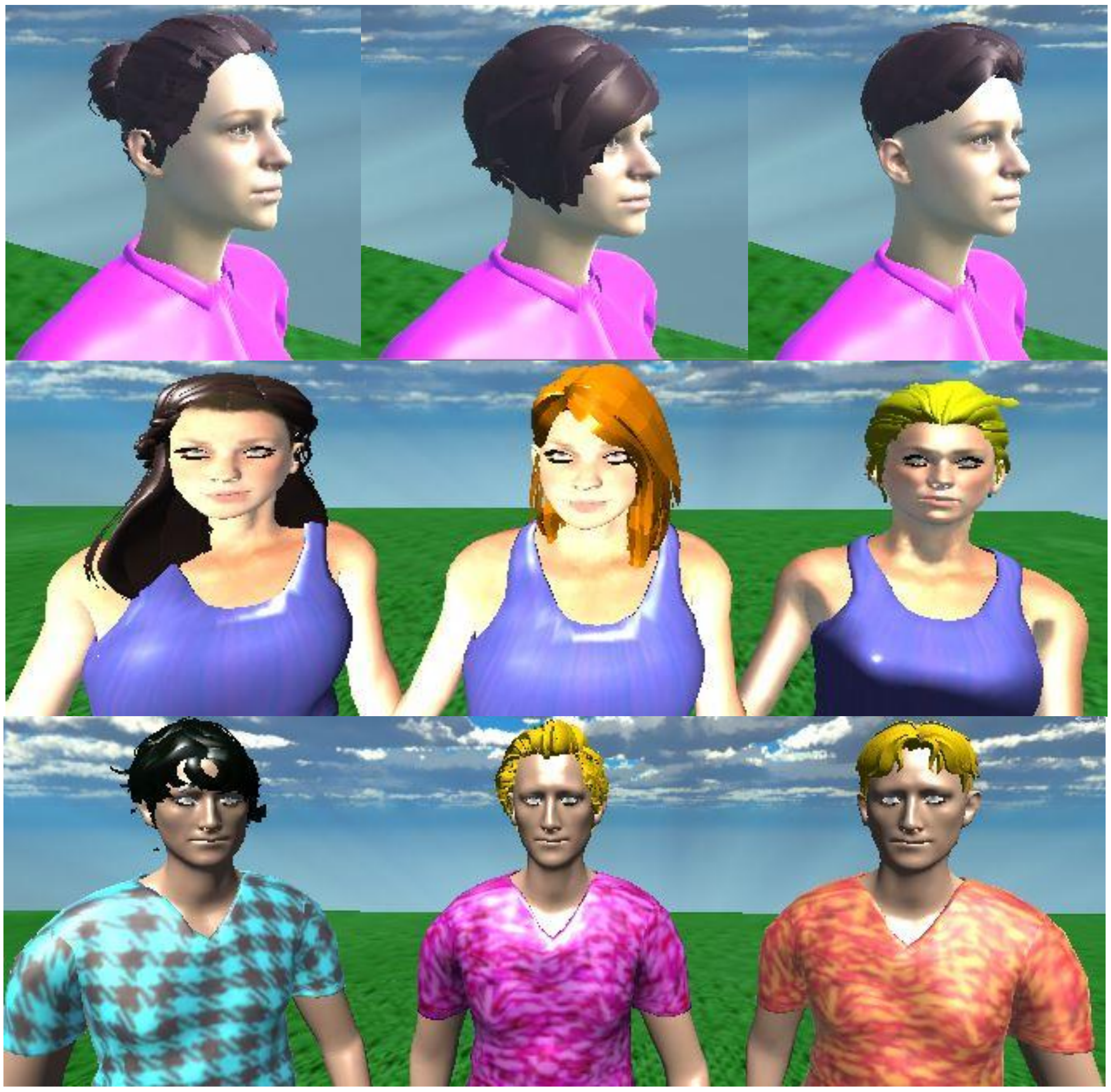


**Figure 3.12**:Modify hair meshes having different colors and shapes in the same model.

### 3.3.5 Adding different accessories

To provide variety to a character's appearance, we use accessories, which are individual meshes linked to each character, to increase their individuality. We aim to ensure that each character appears effortlessly and has adequate visual quality. Developing unique characters in a crowd involves variances in their accessories. However, as the number of polygons utilized for making these accessories increases, so does the rendering cost for crowds [155].

Adding accessories to crowds considerably enhances the individual human template process. Accessories are specific mesh parts that can be added to the original human models to increase the overall appearance diversity across genders and age models. Watches, rings, glasses, hats, phones, and bags contribute significantly to this diversity [155].

We can produce more visually appealing scenarios by providing a crowd with diverse accessories from multiple human templates. These accessories are smoothly and precisely attached to a skeleton, ensuring they are positioned correctly based on their type. A hat, for example, is usually attached to the head, whereas a watch is attached to the wrist [155].

During the initialization of the crowd simulation, each character is assigned a selection of accessories, which are displayed dynamically at runtime. This approach enhances visual variety without significantly increasing computational overhead [155].

Accessories may not require any specific modifications to the animation of the virtual human. Each accessory is represented as a simple mesh and must only be correctly positioned and oriented on the corresponding part of our virtual human. For example, items like phones, watches, bracelets, and rings would be attached to the hand joint, while hats and glasses would be attached to the head joint [155].

Once attached, we can modify their shape, color, and texture to enhance visual variety. These accessories synchronize with the head and hands as the person moves, maintaining realistic motion and appearance throughout the simulation (Figure 3.13) [155].

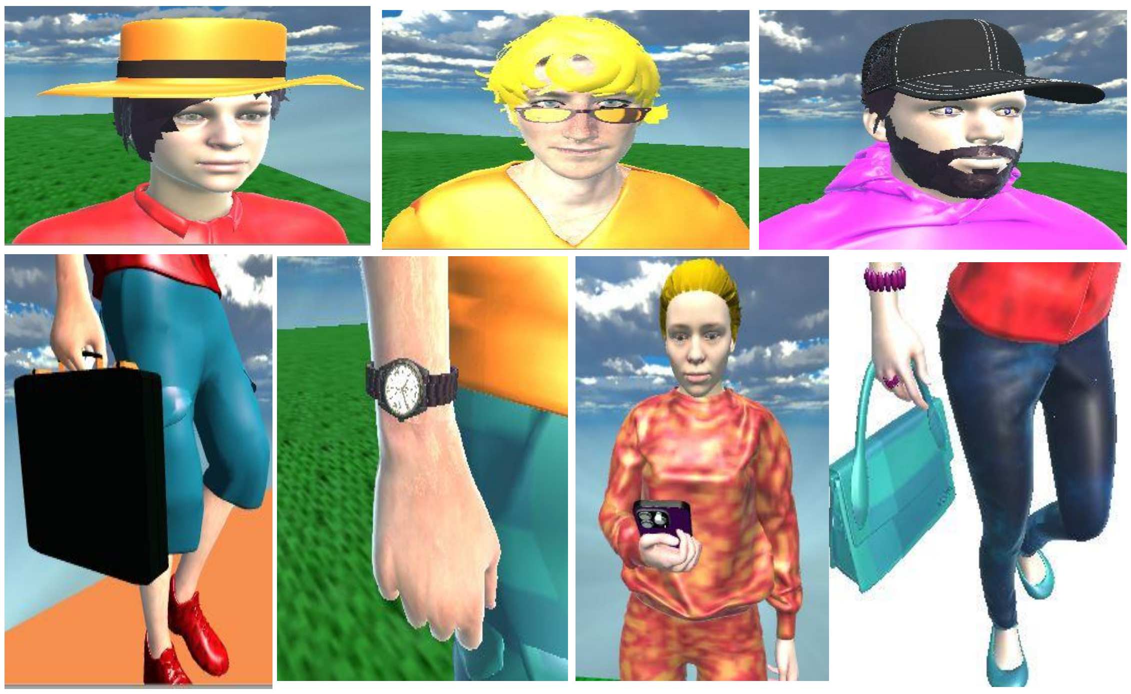


**Figure 3.13**:Various colored accessories are added and attached to different joints or parts of our models.

**Algorithm 2: Various accessories of characters**

```
Input:   list of different templates of characters,
         different accessories (bags, watch, glasses, hats, rings, bracelet
         phone)
Output: varied accessories
         different color accessories
         different shape accessories
- Find specific accessory.
- Find character.
- Makes the game object character the parent of the game object accessory.
- Placed accessory.
- Oriented accessory.
- Edit shape accessory.
- Renderer component to access the material.
- Apply a random color to the accessory.
```

## 3.4 Crowd animation approach

Real-time crowd simulation is a method for creating the movement of many animated characters in a virtual environment. The crowd moves in different directions as the animated characters interact with each other in different ways according to their behavior or as they stroll through the environment while avoiding each other or avoiding certain obstacles in front of them. All of these actions contribute to the final collective behavior of the crowd, which must be performed in real-time.

Our crowd animation approach used in this work aims to generate diverse types of motions. We make a locomotion cycle to produce different animations for each model (Figure 3.14). We create an Animator Controller and use animator scripting to achieve diverse and specific movements for our models. The Animator Controller allows us to manage and blend multiple animations seamlessly. By scripting with the Animator component, we can trigger different animations based on specific conditions or user interactions, adding life and personality to each model. This approach ensures that our models move realistically and dynamically [1].

Our characters can perform various actions like walking, running, jumping, turning, or having idle cycles (Figure 3.15). To achieve this, we set up a collection of animation clips. Once we apply scripting with our Animator component to our model, these animations are configured to play based on different conditions or player input. This setup ensures that our characters move realistically and perform actions that bring them to life within the scene or our environment .

It implements the various animations defined for our models, each with distinct characteristics and controllable motions.

Additionally, we can blend and combine these animations seamlessly using Blend Trees to introduce even more variety.

To organize the Animator Controllers effectively, we employ Sub-State Machine hierarchies. This approach allows us to define locomotion cycles within individual Sub-State Machines, ensuring that each layer of the Animator Controller can access and utilize these defined animations. This structured organization helps manage and integrate complex animation behaviors while maintaining clarity and flexibility within our animation system. Animator Scripting is used to bring life to our model's animations by adding C# scripting, so we create scripting lines in our work and apply them to our models to get a realistic result with various motions. One important aspect is dynamically generating velocity values to control the speed of our animation cycles. These velocity values provide how transitions between animations occur. For instance, the Animator Controller can smoothly transition between walking, running, jumping, or idle states based on these velocity values. The Animator manages these transitions based on the velocity values we set, ensuring a smooth flow between different animations. This flexibility allows us to create various motions and behaviors for our characters, enhancing realism and immersion in our environment [1].

We will explain how our algorithm assures a diverse animation cycle for various types of virtual humans. First, we positioned our models in random positions and directions. Each model provides a specific type of walking or running with precise speed values. In addition, it can jump at a certain height. These steps generate locomotion cycles that produce different motions for each model. It achieves a good result that improves visual realism.

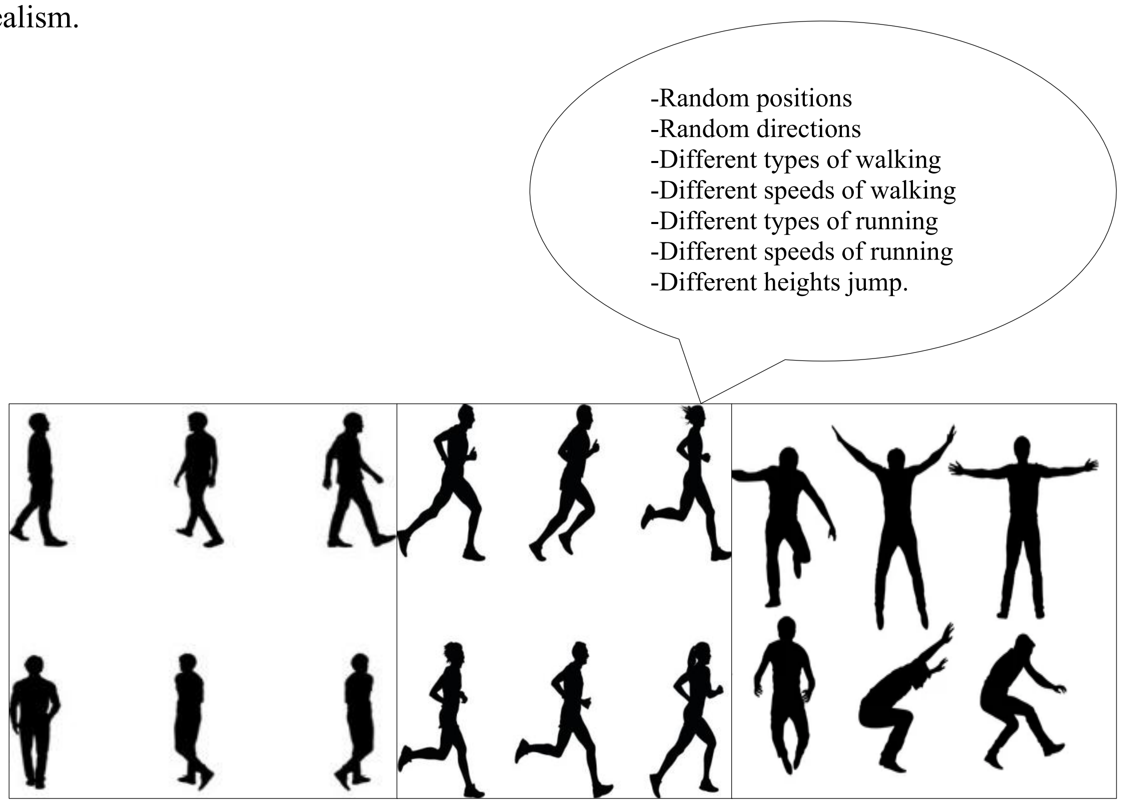

-Random positions
-Random directions
-Different types of walking
-Different speeds of walking
-Different types of running
-Different speeds of running
-Different heights jump.

**Figure 3.14**:Using different animation styles for various models at varying speeds.

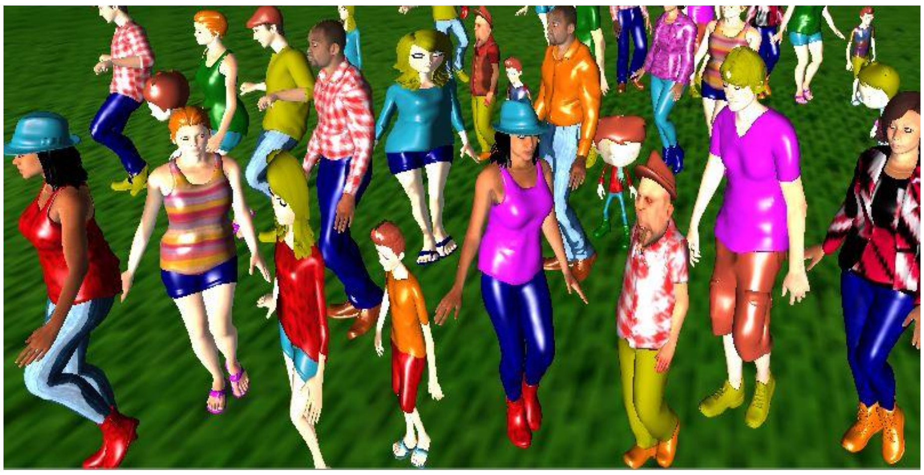


**Figure 3.15**:Applying different cycles of locomotion to each model to get animations variety.

**Algorithm 3: Various animations of characters**

```
Input:   walk velocity, running velocity, jump velocity, direction, jump
          height, gravity, Character controller, characters.
Output: different cycle animations.
          different speed animations.
If (Character controller touches the ground during the last frame) then
    Switch (option)
        - Case 1:  get walk type with walk velocity.
        - Case 2:  get turn type.
        - Case 3: get the run type with run velocity.
        - Case 4: get idle type.
        - Case 5: get jump type with jump velocity and jump height.
End if
```

We will examine a strategy demonstrating how our crowd may distinguish its animation from others. It involves animating our characters with diverse animations and directing their movement with different speed values along a specified path using basic C# scripting. Within our open environment, we designate each character's starting point and final destination. It ensures that each character transforms his position and follows a path sequence of points to reach his goal effectively [1].

**Algorithm 4: Character animation with paths**

```
Input:   character, movement, velocity, position list of points.
Output: a path to follow.
- Determinate the initial position of the path
- Determinate the destination position of the path
- Determinate the movement.
- Determinate the velocity
- Determinate the path
- Following the path
```

## 3.5 Crowd collision detection approach

We provide a detailed description of our method to enhance and achieve convincing results in the realism of collision avoidance within crowds. Collision avoidance is a problem that needs to be focused on in crowd simulation between the different individuals in the environment. We must solve this problem and avoid unrealistic results to ensure an efficient crowd system. Many researchers propose several approaches to realize collision avoidance behavior in crowd simulation [1].

The algorithm proposed in this study is based on using a technique called Raycasting for collision detection and possibly for determining visibility or field of view:

- **Raycasting Basics**: It estimates the intersection between our characters during their movements from the projection of the current character's position to the other character. It involves casting a ray (a straight line) from a specific point (often a character's position) in a particular direction (e.g., the direction of movement or field of view). The ray continues until it intersects with an object in the scene or reaches a maximum distance.
- **Collision Detection:** In character movement, Raycasting can detect collisions between the character and other objects in the environment. For example, if a character is moving forward, we can cast a ray in the direction of movement to check for any obstacles (like walls or other characters).
- **Field of View Representation:** Raycasting can also determine what is visible from a character's perspective. By casting rays in multiple directions within a specified field of view angle, we can check which objects or entities are within that field and thus visible to the character. We then determine the mass and initial speed of the type of movement that will be applied to our character (WalkSpeed, RunSpeed).
- **Collided characters:** The raycasting method determines if our characters have collided after a certain number of time steps. This stage involves calculating the distance between them, determining the masses and velocities of each character, and selecting the faster one.
- **Newton's Laws and the concepts of momentum and kinetic energy conservation rules**: We used them to represent the dynamics of motion phenomena, calculate the final velocities for each character, and change the direction. The conservation of momentum and kinetic energy rules are valid before and after an elastic collision without external net forces.

$$p_1 + p_2 = p'_1 + p'_2 = cte \tag{3.5}$$

$$K_1 + K_2 = K'_1 + K'_2 = cte \tag{3.6}$$

| Scenarios | Various mass and velocity cases. | The velocities at the time of collision. | Final velocities after collision detection. |
|---|---|---|---|
| Scenario1 | M >> m<br>And<br>$V_M > V_m$ | $v'_M = \dfrac{v_M (M-m)}{M+m}$ | $v'_M = v_M$<br>$v'_m = 2\, v_M$ |
| Scenario2 | M = m<br>And<br>$V_M > V_m$ | $v'_m = \dfrac{2M * v_M}{M+m}$ | $v'_M = 0$<br>$v'_m = v_M$ |
| Scenario3 | M>m<br>And<br>$V_M < V_m$ | $v'_M = v_M$<br>$v'_m = v_m$ | $v'_M = v_M$<br>$v'_m = v_m$ |
| Scenario4 | M=m<br>And<br>$V_M = V_m$ | | $v'_M = v_M$<br>$v'_m = v_m$ |

**Table3.1**: Determination of the final velocity of the characters after the sensing collision by the raycasting technique, newton's laws, the concepts of momentum and conservation of kinetic energy.

Where, $p_1$, $K_1$, $p_2$, $K_2$, $p'_1$, $K'_1$, $p'_2$ and $K'_2$ are the momentum and kinetic energy of body 1 and body 2 before and after the collision, respectively [158].

When our two characters collided at two different velocities, $v_M$ and $v_m$ and two different masses, $M$ for the heavier object and $m$ for the lesser one. Along with the sum of the two momenta, the sum of the two kinetic energies is conserved in elastic collisions.

The two conservation equations are written as follows:

$$\tfrac{1}{2}Mv_M^2 + \tfrac{1}{2}mv_m^2 = \tfrac{1}{2}Mv_M'^2 + \tfrac{1}{2}mv_m'^2 \tag{3.7}$$

$$Mv_M + mv_m = Mv'_M + mv'_m \tag{3.8}$$

Where the final velocities of both objects $v'_M$, and $v'_m$ are given. The sums of the velocities before and after the collision for each object are equal when the two equations are rearranged with the velocities of one object on one side of Eq. (3.7) and Eq. (3.8) and those of the other on the opposite side, then dividing the rearranged Eq. (3.7) with the rearranged Eq. (3.8) [159].

$$v_M + v'_M = v_m + v'_m \tag{3.9}$$

The ultimate velocities following the collision are determined by solving the set of linear Eq. (3.8) and Eq. (3.9):

$$v'_M = \frac{(M-m)v_M + 2mv_m}{M+m} \tag{3.10}$$

$$v'_m = \frac{(m-M)v_M + 2Mv_M}{M+m} \tag{3.11}$$

Depending on the values of m and M for our colliding characters in Eq. (3.8) and Eq. (3.9), we disregard the smallest value and get the final velocities $v'_M$and $v'_m$ (Table 3.1).

We derive a valid method for collisions from Newton's Laws, the laws of momentum, and kinetic energy conservation. It provides a way to determine the final velocities after detection collision using the Raycasting technique without external net forces, according to each character's initial masse and velocity (Figure 3.16) [1].

In the second scenario, when the masses are equal, the final velocity $v'_M$ of this character will be reduced to a small value for some seconds. At this time, the second character will change his/her direction and take the velocity of the first character to move. After steps of seconds, both of them will take their first velocity [1].

To ensure that we examine all possible conditional cases that could occur between characters, we have added two additional cases based on the same previous data. In these two situations, the end velocity used to predict the occurrence of a collision is the same as the initial velocity of the two characters before the collision [1].

**Algorithm 5: Collision detection**

```
Input:   initial masse, direction, initial velocity, ray range, characters,
         character controller, positions, and orientations.
Output:  final velocity,
         final direction and rotation.
         distance
          If (collision occurs) then
           - Calculate distance.
           If( $V_M>V_m$ and $M > m$)   then ( $v'_M = v_M$ and $v'_m = 2\,v_M$)
             Else if( $V_M>V_m$ and $M =m$) then ( $v'_M = 0$ and $v'_m = v_M$)
               Else if( $V_m>V_M$ and $m>M$) then ( $v'_m = v_m$ and $v'_M = 2\,v_m$)
                 Else if( $V_m>V_M$ and $m=M$) then ( $v'_m = 0$ and $v'_M = v_m$)
                   Else if( $V_M<V_m$ and $M>m$) then ( $v'_m = v_m$ and $v'_M = v_M$)
                     Else if( $V_m<V_M$ and $m>M$) then ( $v'_m = v_m$ and $v'_M = v_M$)
                       Else if( $V_m=V_M$ and $m=M$) then ( $v'_m = v_m$ and $v'_M = v_M$)
             End if
          -Change direction.
          -Change rotation.
         End if
```

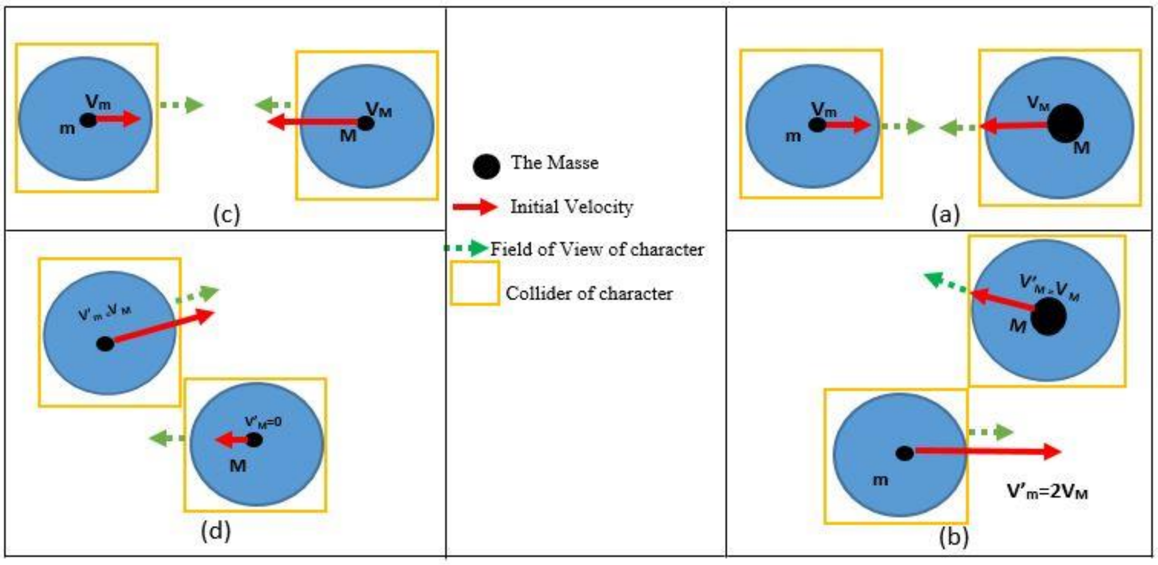


**Figure 3.16**:Examples of two cases of collision

(a) Perception with no collision of the first Scenario. (b) Final velocities are determined according to the initial different velocities and masses of characters by Newton's Laws, as well as the concepts of momentum and kinetic energy conservation after detection collision by the Raycasting technique. (c) Perception with no collision of the second Scenario. (d) Final velocities are determined according to the different initial velocities and equal masses of characters by Newton's Laws, as well as the concepts of momentum and kinetic energy conservation after detection collision by the Raycasting technique.

## 3.6 Conclusion

This chapter explores various methods that include multiple aspects to appeal to a diverse crowd. First, we ensure visual diversity among crowd characters to avoid the perception of unrealism. It includes investigating the impact of appearance variations such as shape, clothing, color, and texture on distinguishing individuals within crowds. The study encompasses diverse human genders and age groups, ensuring each character is uniquely attired with distinct textures and colors covering variations in hair, skin, and accessories. In addition to visual diversity, the chapter details a crowd animation approach aimed at enriching realism through a wide range of motions. It includes creating locomotion cycles specific to each character model, allowing for various movements during the simulation. Such diversity in motion contributes to a more dynamic and effective representation of crowds in virtual environments. Furthermore, the chapter introduces an algorithm utilizing Raycasting for collision prevention among characters. This algorithm is grounded in fundamental physics principles, including Newton's laws, momentum conservation, and the conservation of kinetic energy.

By applying these principles, the algorithm accurately computes the final velocities of characters involved in potential collisions, ensuring realistic and reliable outcomes in crowd simulations. Finally, the chapter integrates visual diversity, dynamic motion, and physics-based collision avoidance to enhance the realism and believability of crowd simulations, making them more immersive and compelling for various applications, including virtual environments and interactive simulations.

# Chapter 4

# Experiments and results

## 4.1 Introduction

The model presented in this chapter demonstrates our contribution to this thesis. The results provide a performance that effectively includes crowd variation inside the system while providing various features.

This chapter includes a comprehensive review of the suggested model, including the experiments and results acquired during the thesis. We start establishing the context for Unity 3D and C# programming, followed by an overview of the system architecture and an extensive description of our implementation process. A number of concepts are provided to demonstrate the feasibility of crowd variation, focusing on their importance in improving the system's flexibility. We then validate the model by examining results aimed at increasing the significance of our crowd system. We examine the combinations utilized to provide appearance and animation diversity, determining the model's overall performance. Finally, we will present a conclusion in the following part.

## 4.2 Unity 3D and C# programming context

Unity, an effective gaming engine, is becoming increasingly popular among game creators. One of the main reasons for its popularity is Unity's ability to bring together developers with varied skill sets who collaborate on projects in an efficient and flexible process. Unity contributes to game production by providing a user-friendly interface, a flexible scripting framework, and an extensive library of materials. It is a powerful game creation and 3D development tool, allowing users to interact with 3D design models within the Unity environment.

When producing games in Unity, creators use C#, a popular and influential programming language designed expressly for Unity 3D development [67]. Unity Game Engine creates video games for web applications, desktop platforms, consoles, and mobile devices. Unity is the world's most popular game engine for development. Unity Technologies developed Unity 3D. C# is utilized because of its excellent connections to Unity, flexibility in implementation, and a strong community that offers sufficient tools and help for developers working on immersive projects [70].

## 4.3 System overview

Our objective is to enhance the functionality of our crowd variety by incorporating and manipulating various elements (Figure 4.1). The process begins with presenting scene data, including human models equipped with textures, skeletons, and meshes. Subsequently, we set up the environment, positioning virtual humans and a directional light to ensure accurate color perception based on photons' energy distribution. This initial step in rendering involves assigning different positions and orientations to our models, thereby achieving a more visually diverse array of instances of the same model

To further enhance visual quality, first, we save each model in a list; then, we can duplicate them and modify their colors and textures. We manipulate the virtual human's color and texture by editing each model part's material. We create arrays that save textures according to the type of parts of models to apply them randomly, such as pants, shirts, shoes, and skin. We assign a material using a shader, colors, textures, and property values. The shader file dictates the color of each pixel, modifies the character's material color, and introduces a texture with a color value to enhance the overall effect. Utilizing the shader to blend pixels, representing the character's surface color determines how pixels are rendered.

Depending on our investigation of color effects, we used focused color changes to increase the variety in our models, particularly in hair, hats, eyes, beards, and bags. This method increased their visual impact and used effective value combinations to create a more impressive overall appearance.
This approach involved pixel mixing with an Alpha source to control transparency levels. We developed a method for obtaining RGB colors from textures and colors, resulting in a transparency range from 0.0 to 1.0. We then calculated the Alpha values within this range to achieve optimal color rendering for each piece, resulting in precise and colorful visual results.
Furthermore, we modify the shapes and heights of our models, including accessories that are carefully attached to precise joints for smooth integration. Each item is treated with effective color and texture modifications to reflect the overall style. This approach ensures that each modification enhances the model's design while preserving functionality and visual accuracy. After determining these terms, we render the models using basic illumination shaders and Phong models.

To infuse life into these models, we incorporate diverse movement types through various animations, forming a locomotion cycle applied simultaneously to each model via a C# script. We describe our collision avoidance method utilizing Raycasting, grounded in Newton's laws, momentum laws, and kinetic energy conservation. This algorithm yields accurate character final velocities [1].
In order to optimize the Frames Per Second (FPS) speed in our scene, we apply Occlusion Culling. This technique operates on the principle of removing objects hidden behind others. Consequently, only the characters remain visible within the camera's field of view, and other objects in our environment are not rendered. It enhances our scene's overall performance and visual experience.

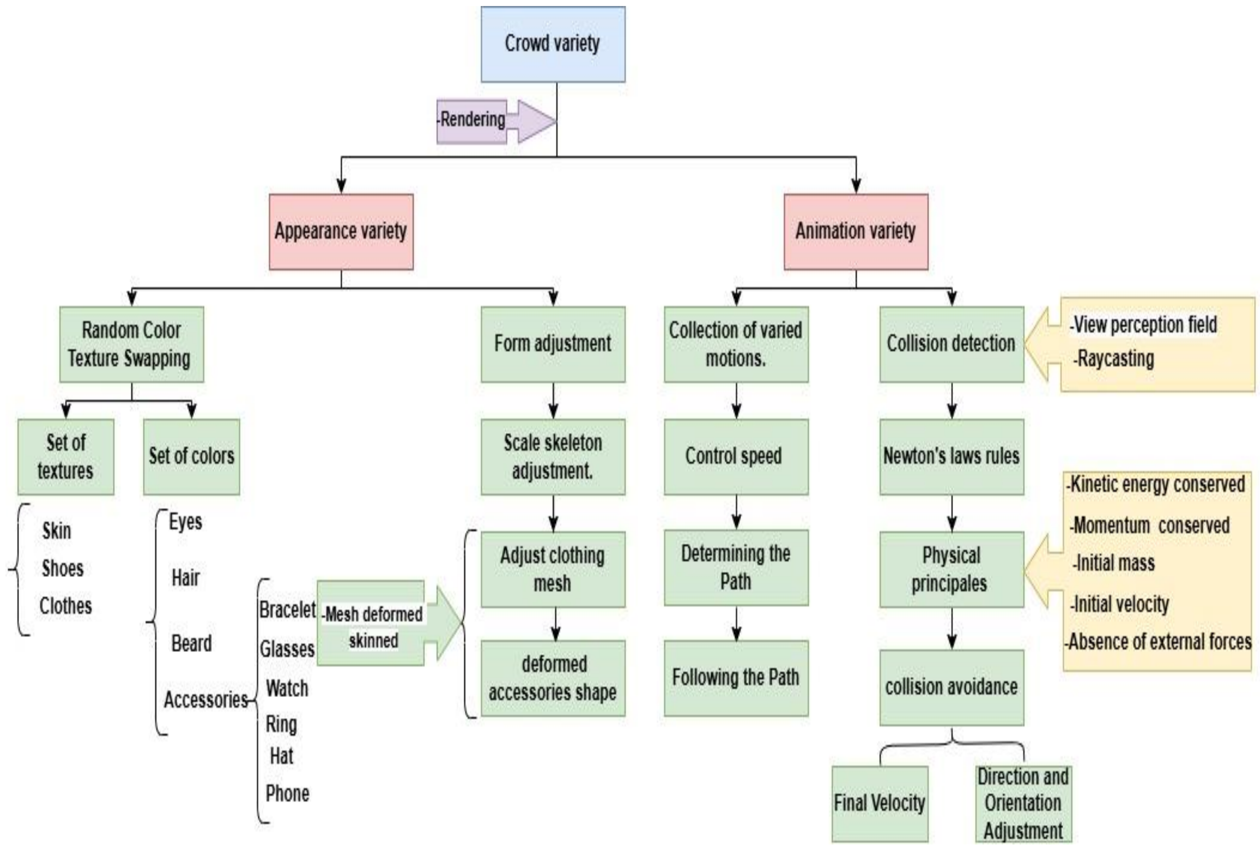


**Figure 4.1**:Different stages of system overview.

## 4.4 Crowd system results and discussion

The simulation of our virtual humans was executed on a Windows 10 computer equipped with an Intel i5 core operating at 2.70 GHz, an AMD Radeon graphics card, and 16 GB of memory. Our pedestrian system comprises shader files for rendering models and C# code files, facilitating the simulation of virtual humans within Unity 3D version 2018.3.2f1 (64-bit). We implemented and tested the approaches discussed here, resulting in a crowd system that features diverse virtual characters [155].

This system showcases various appearance, shape, animation, color, texture, clothing, accessories, and rendering techniques, ensuring higher-quality results.

### 4.4.1 Crowd density and appearance variation in an open environment

This system allows us to simulate crowds in real time within different environments. We start with open environments, which allow us to showcase dynamic scenarios where our simulations feature many characters, each uniquely styled with randomly selected colors and textures in model lighting (Figure 4.2). In our testing phase, we conducted extensive experiments to analyze how varying the number of virtual humans and the diversity of textures and colors impact different aspects of our characters. We precisely calculated each scenario's frames per second performance to determine system efficiency.

Our results obtained an explicit association: as the number of characters increased, the complexity increased, introduced by the number of displayed characters and the variety of textures and colors applied to each. However, the frames per second (Frames per second) decreased. The function of the FPS values is calculated for each execution in scenarios according to the number of characters and the addition of textures and colors. The performance of a display device is measured in frames per second (FPS). It is a number that displays and demonstrates the program's or game's strength and efficiency, and it indicates the display operations on the screen and the screen update operations per second for sequential images.

Our studies indicated significant changes in FPS as we added varied combinations of textures and colors to our virtual characters. Specifically, scenarios with greater texture diversity and color variation proved to have lower FPS values due to increased computational effort. This correlation emphasizes improving resource allocation and rendering processes to ensure consistent real-time performance. Due to the complexity of our scene, which depends on the mesh of the virtual humans, the geometry of each character, colors, and image textures apply. It scales O(n) and reflects the recursive nature of the instantiate method. It is an excellent method for effortlessly creating character models within our sceneries. This flexibility allows us to clone characters with precise direction, position, and rotation specifications. Our system can generate a visually diverse crowd features instances of each character type. It demands significant memory allocation for storage but yields visually compelling results, enhancing details like skin, eyes, hair, and clothing to build a more realistic and interesting virtual environment (Figure 4.3).

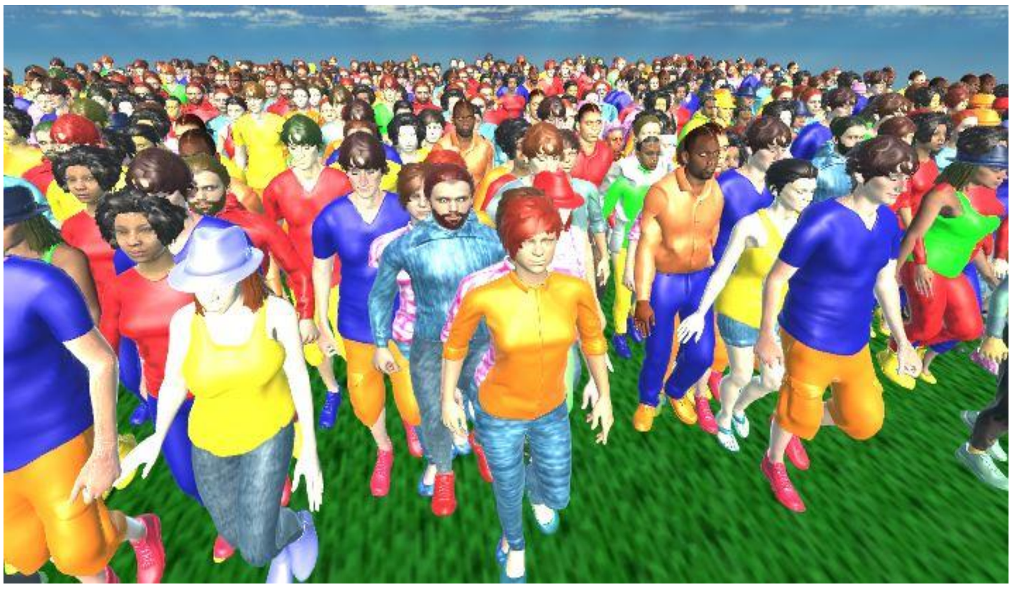


**Figure 4.2**:Crowd simulation with varied color and textures in an open environment.

| Different scenarios | Number of characters | Number of different Textures applied | Number of different colors applied | Frames per second(ms) |
|---|---|---|---|---|
| Scenario1 | 20 | 84 | 34 | 59 |
| Scenario2 | 70 | 294 | 119 | 43 |
| Scenario3 | 150 | 630 | 255 | 22,9 |
| Scenario4 | 310 | 1302 | 527 | 12 |
| Scenario5 | 800 | 3360 | 1360 | 10 |

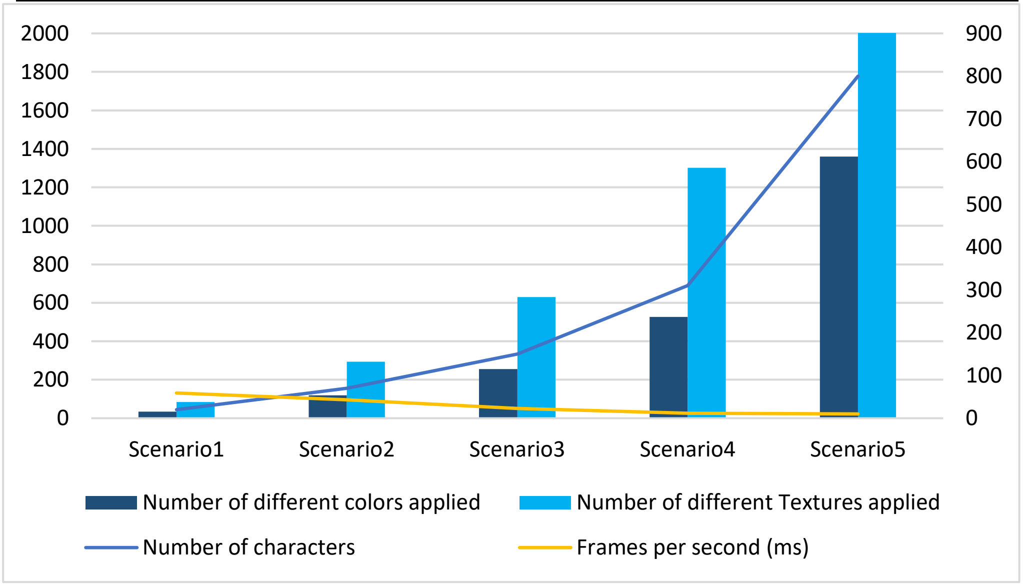


**Figure 4.3:**The performance evaluation of our approach in five different scenarios included investigating the impact of different textures and colors on changing FPS values.

To the scalability of our simulation, we varied the size of simulated crowds by adjusting their shape and height parameters. Each simulation run produced varying results regarding frame rate stability, particularly when scaling our character models using random ranges.

For instance, the frame rate was measured at 60 (ms) for 20 characters, 63 (ms) for 80 characters, and 62 (ms) for 160 characters. Additionally, we observed an increase in the number of polygons, reflecting the growing complexity in the scenes as indicated by our crowds' vertex and triangle counts.
This increase in mesh data significantly impacts scene complexity. It emphasizes the diversity of characteristics in the heights and shapes of virtual individuals, preventing us from seeing the similarities in their physical appearances. It demands more significant memory storage capabilities (see Figure 4.4).

| Different scenarios | Number of characters | Number of Triangles(K) | Number of Vertex(K) | Frames per second (ms) |
|---|---|---|---|---|
| Scenario1 | 20 | 97.6 | 78.6 | 60 |
| Scenario2 | 40 | 146 | 97.2 | 58 |
| Scenario3 | 80 | 226.4 | 147.5 | 63 |
| Scenario4 | 160 | 343 | 234 | 62 |

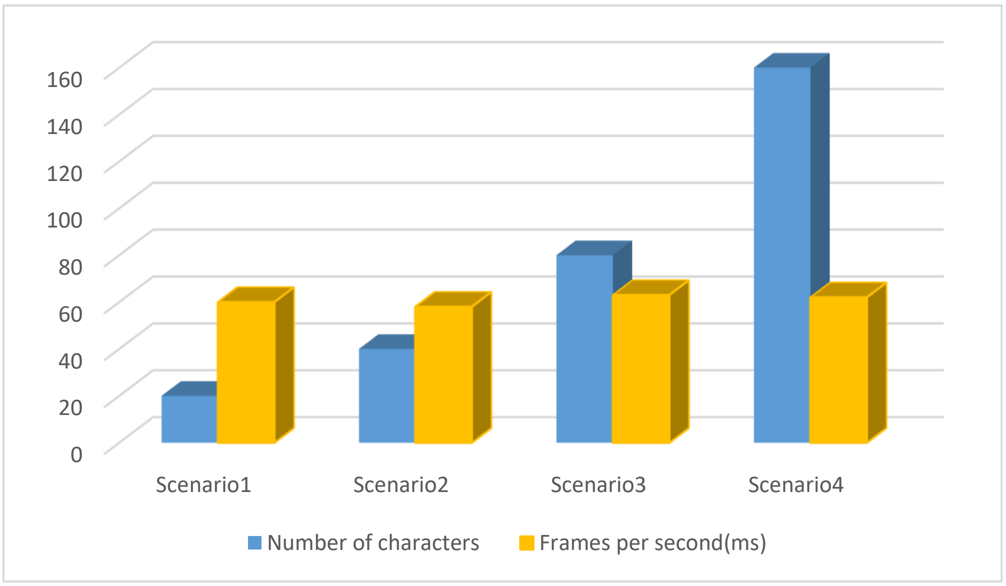


**Figure 4.4**:The performance evaluation of our approach in four different scenarios included investigating the impact of different shapes and heights on changing FPS values.

(Figure 4.5) illustrates the variation in the number of accessories each character can wear, ranging from 0 to 10, such as watch, glasses, phones, bags, rings, bracelets, and hats, each with different colors and textures. Our tests focused on how these accessories impact frame rates. Across the four scenarios depicted in (Figure 4.6), we observed a decrease in frames per second as the number of characters increased. This effect proved effective in achieving results comparable to more computationally expensive methods that changed character shapes and used a variety of textures, colors, and meshes. This effect was comparable to more computationally intensive methods that modified character appearances using a variety of textures, colors, and models. Maintaining individuality among crowd members is important, especially when their similarities are precise [155].

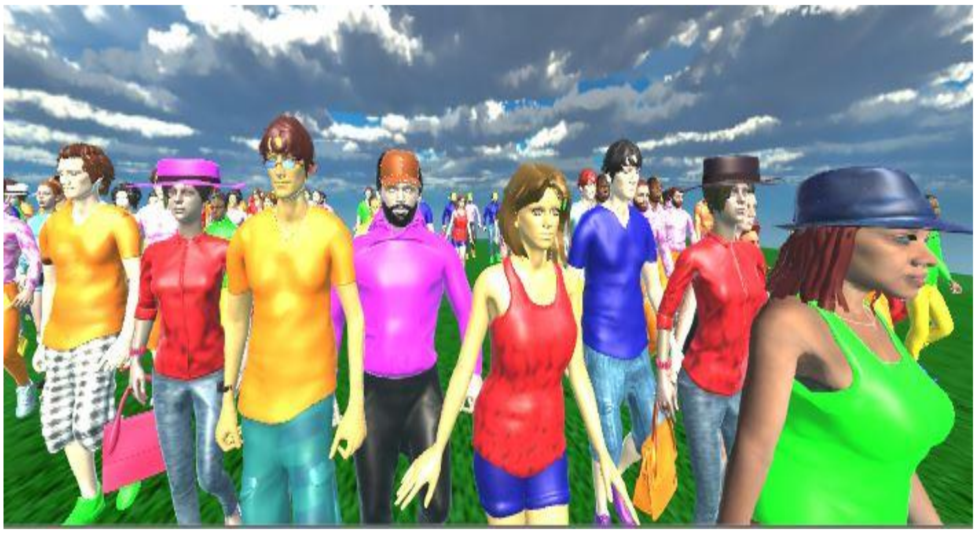


**Figure 4.5**:Crowd simulation with varied hair, accessories colored and textures in an open environment

| Different scenarios | Number of characters | Number of accessories Applied to characters | Frames per second(ms) |
|---|---|---|---|
| Scenario1 | 30 | 20 | 23 |
| Scenario2 | 60 | 40 | 14.6 |
| Scenario3 | 120 | 80 | 10 |
| Scenario4 | 240 | 160 | 10 |

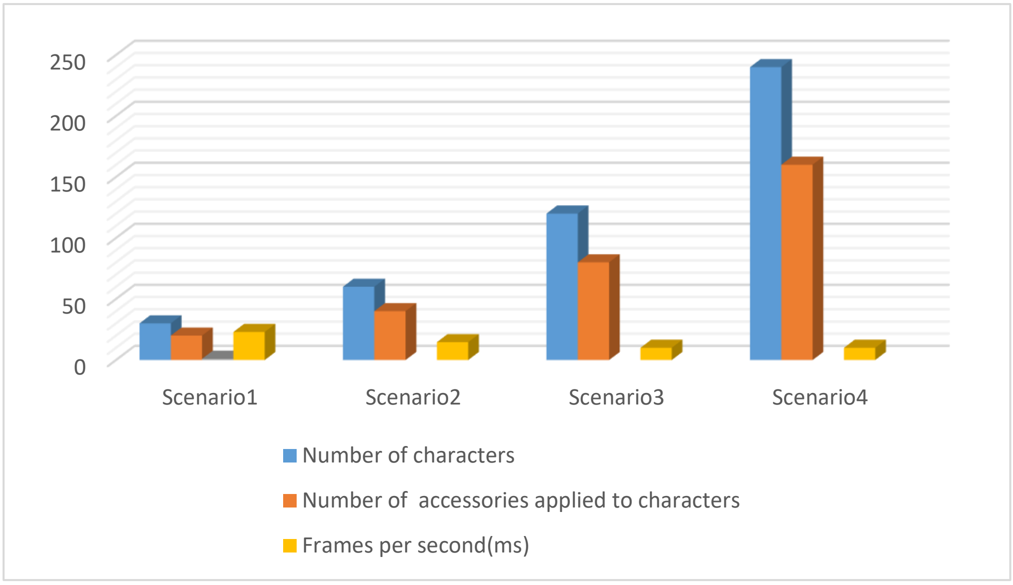


**Figure 4.6:**The performance evaluation of our approach in four different scenarios included investigating the impact of adding various accessories and hair followed by changing FPS values.

### 4.4.2 Crowd density and animation variation in an open environment

Our approach introduces diverse movements by positioning the crowd in an open environment, allowing for dynamic and varied interactions. (see Figure 4.7), each template model exhibits distinctive gaits and actions while ensuring no collisions. These range from idle state, jumping, walking, strolling with a briefcase, gracefully cat walking, the unique gait of an older adult, walking with happiness, performing a rhythmic hip-hop dance, walking while texting on a phone, to running movement. Each model can conduct certain kinds of movement and can vary the movement he does along a predetermined path. The animation is defined by a specified speed of 1 to 50 (m/s). These speed values are determined to ensure the results are realistic and visually appealing, improving the overall sense of movement within the parameters set. So, each person varies in their own, which reflects their unique characteristics and activities (see Figure 4.8) [1].

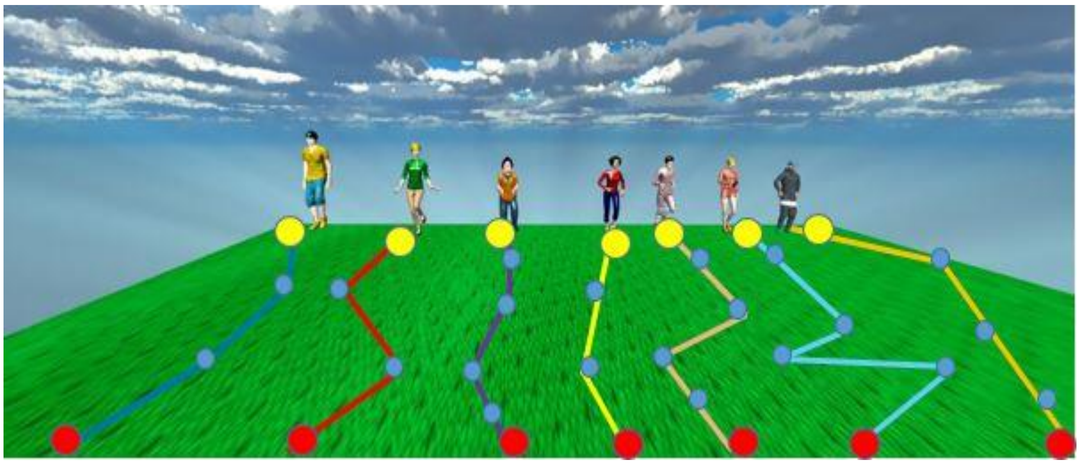


**Figure 4.7**:Animation crowd without collision. The start position represented by (yellow) circle, the destination position shown in (green) circle. The circles represented with (bleu) circles displayed our different paths to follow by characters.

In another approach, we handle character animations in scenarios involving collisions among them (Figure 4.9). Each character is animated with locomotion of different movements and different speeds, enabling interactive collisions where they can collide with each other. We use collision detection algorithms based on Newton's Laws to handle these collisions effectively, emphasizing momentum and kinetic energy conservation principles.

This method accurately computes final velocities after collisions using raycasting techniques, assuming no external forces are involved, and considers each character's initial mass and velocity [1].

| Different scenarios | Number of characters | Number of animation types applied | Range of movements speed applied | Frames per second(ms) |
|---|---|---|---|---|
| Scenario1 | 7 | 11 | [1 m/s,50 m/s] | 60 |
| Scenario2 | 14 | 22 | [1 m/s,50 m/s] | 59.1 |
| Scenario3 | 21 | 33 | [1 m/s,50 m/s] | 58.7 |
| Scenario4 | 63 | 99 | [1 m/s,50 m/s] | 55 |

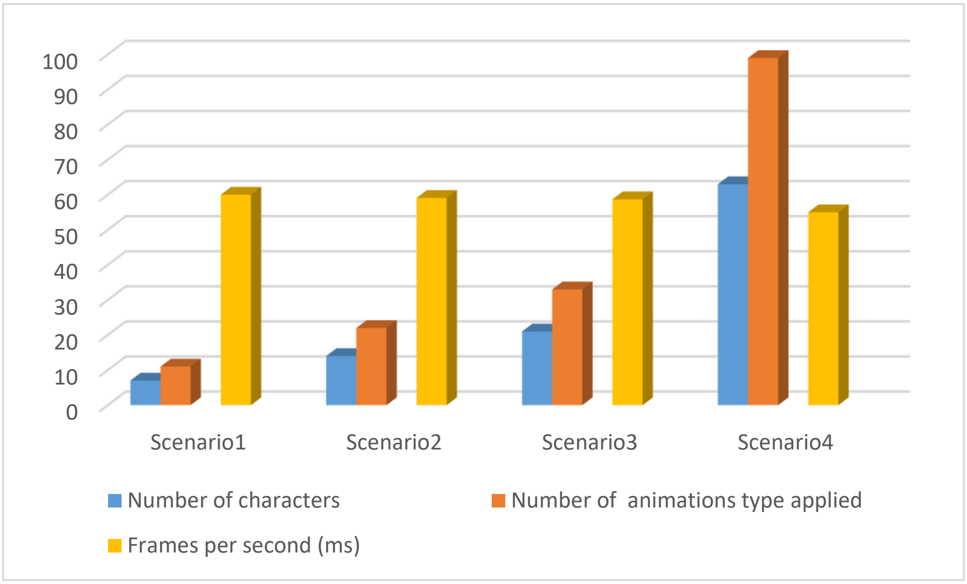
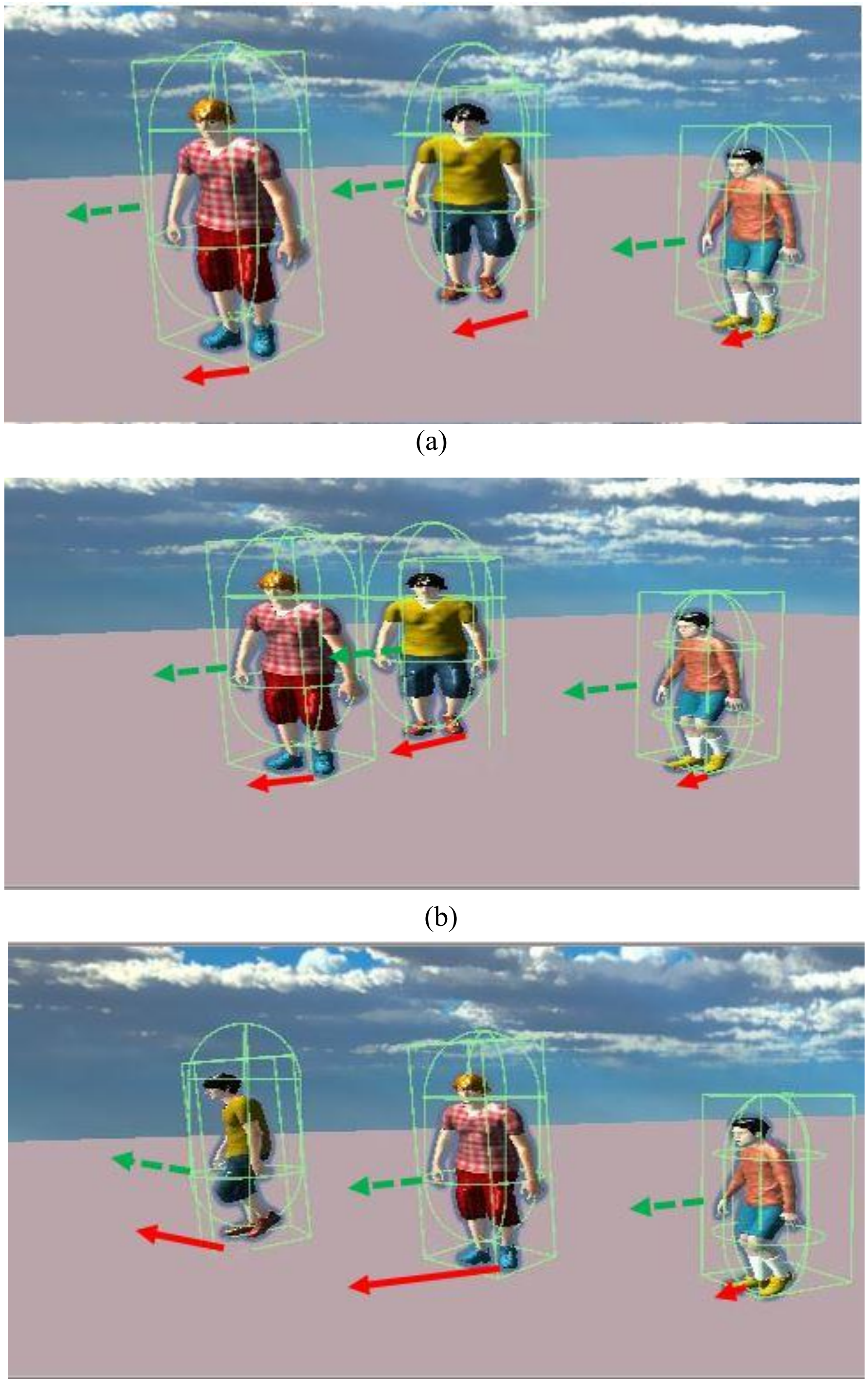


**Figure 4.8**:The performance evaluation of our approach in four different scenarios included investigating the impact of adding various types of animations with different speed values on changing FPS values.

An occlusion culling algorithm performed on the static model could help speed up the rendering. (Figure 4.10) illustrates the computed results of FPS graphics before and after occlusion culling implementation.  It is available in four cases according to the number of virtual characters. The data is presented in four distinct cases, each corresponding to varying numbers of virtual characters. The application of occlusion culling resulted in an observable increase in FPS speed across four distinct scenarios, showcasing an enhancement in our project's runtime performance. When using occlusion culling, Unity divides the scene into cells and generates data describing each cell's geometry and the visibility relationships between adjacent cells. Unity then optimizes by merging cells whenever possible, reducing the final data's size. Occlusion culling, as a feature, functions by turning off the rendering of objects that are not within the camera's view due to occlusion by other objects.



(a)

(b)

(c)

**Figure 4.9**:Scenario1 of crowd animation with collision detection. (a) Perception with collision: The (green) arrow represented the field of view of character, the velocity before collision shown in (red) arrow. (b) The detection of collision between two characters collided. (c) After the detection of collision by raycasting technique. (red) arrow represented the final velocities according to the initial different velocities and masses of characters by Newton's Laws, the concepts of momentum and kinetic energy conservation after collision detection.

Occlusion culling is essential for improving efficiency, particularly in complex scenarios; therefore, during runtime, the virtual environment's data scene is loaded into memory, and occlusion culling is used to identify objects in the camera's view. This initial processing reduces the number of objects treated, which improves overall performance. Eliminating certain items in the rendering pipeline allows us to focus resources on rendering what is visible to the viewer.

| Number of characters | FPS graphics | FPS graphics after applying occlusion culling |
|---|---|---|
| 11 | 92 | 124 |
| 55 | 58 | 81.2 |
| 220 | 13.2 | 24.6 |
| 1000 | 3 | 4.2 |

**Table 4.1**:The change in FPS values of each run before and after the application of occlusion culling with a total of 1000 characters.

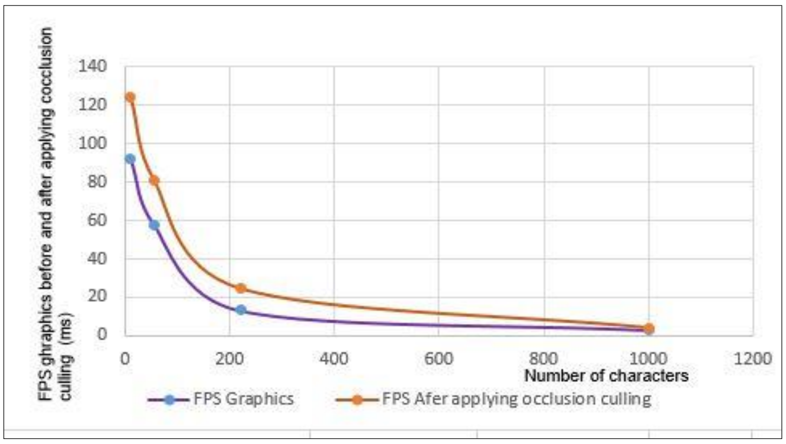


**Figure 4.10:**The change of FPS graphics before and after applying occlusion culling for different numbers of characters.

## 4.5 Analysis and validations of results

### 4.5.1 The performance evaluation of our appearance variation approach by exploring the maximum combinations of elements

To confirm the accuracy of our results, we determine the combinations of the total number of elements used to enhance the variety in appearance, selecting a specific number of elements at a time. The results below include all possible situations in our virtual environment when all components that ensure variations in appearance, height, and form are applied, where ***H*** represents the number of all characters used in our model, ***A*** indicates the number of all accessories used for our characters, ***P*** is the number of all textures used for pants, ***R*** represents the number of all textures used for shirts, ***B*** refers to the total number of textures used for bodies, ***N*** represents the total number of hair types used for characters, ***M*** is the total number of hair colors, ***F*** is the total number of form values, and ***G*** refers to the total number of height values [155].

***Max_Combina_app*** is the value of each model's maximum feasible combination terms that contribute to our appearance variety.

$$H * A * P * R * B * N * M * F * G = Max\_Combina\_app \qquad (4.1)$$

- The maximum feasible men's combination terms that contribute to appearance variety:

$$(1 * 5 * 4 * 5 * 4 * 4 * 7 * 8 * 8) * 5 = (716800) * 5 = 3584000.$$

- The maximum feasible women's combination terms that contribute to appearance variety:

$$(1 * 7 * 4 * 5 * 4 * 4 * 7 * 8 * 8) * 10 = (1003520) * 10 = 10035200.$$

- The maximum feasible young children's combination terms that contribute to appearance variety:

$$(1 * 5 * 4 * 5 * 4 * 4 * 7 * 8 * 8) * 2 = (716800) * 2 = 1433600.$$

- The maximum feasible older adult’s combination terms that contribute to appearance variety:

$$(1 * 5 * 4 * 5 * 4 * 8 * 8) * 1 = (25600) * 1 = 25600.$$

| Number of characters in different scenarios | Visual appearance diversity considering maximum combination values |
|---|---|
| Scenario1 ,H=18 | 15078400 |
| Scenario2 ,H=72 | 60313600 |
| Scenario3 ,H=144 | 120627200 |
| Scenario4 ,H=720 | 603136000 |

**Table 4.2**:The maximum combinations values that enhance visual appearance diversity of different templates on different scenarios.

We realized that our system could generate an extensive selection of character appearances by combining a variety of essential factors such as the type and number of characters, the number of accessories, the textures of pants, shirts, and skin, the kind and color of hair, the height, and the shape of models. The number of feasible appearance combinations increases according to the number of characters in diverse scenarios. For example, employing 1080 characters can produce approximately 904 million distinct combinations in our system.

This exponential expansion dramatically increases the diversity of our character appearances. To further this diversity, we may enhance the number of accessories and add different textures to pants, shirts, bodies, and hairstyles. Furthermore, introducing differences in shape and height will increase visual diversity, resulting in more believable and varied character visuals.

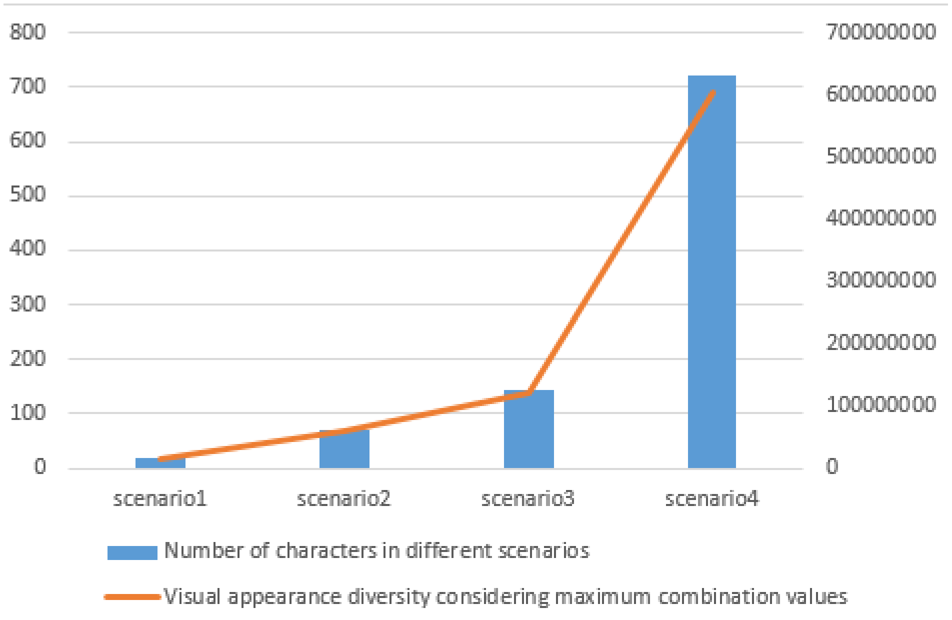


**Figure 4. 11:** Exploring the maximum feasible combinations of elements in our diverse appearances according to the number of characters involved.

## 4.5.2 Evaluation of the performance of possible combinations of our approach for changing appearance using a particular number of element sets for different characters.

This process represents the number of distinct ways to improve the appearance variety by choosing terms at a time. We must determine the number of possible combinations for all terms across various scenarios.

$$C_{H}^{h}.C_{A}^{a}.C_{P}^{p}.C_{R}^{r}.C_{B}^{b}.C_{N}^{n}.C_{M}^{m}.C_{F}^{f}.C_{G}^{g} = \text{Possible\_App\_Var} \quad (4.2)$$

To start, we assign specific labels to each element related to an account:

- ***H*** indicates the total number of characters, while ***h*** indicates the number of characters considered.
- ***A*** represents the total number of accessories, with ***a*** specifying the number of accessories used by the model.
- ***P*** stands for the total number of textures for pants, and ***p*** signifies the number of textures for pants assigned to the model.
- ***R*** refers to the total number of textures for shirts, whereas ***r*** indicates the number of textures for shirts assigned to the model.
- ***B*** is the total number of textures for bodies, with ***b*** representing the number of textures for bodies assigned to the model.

- ***N*** represents the total number of hair types, and ***n*** specifies the number of hair types assigned to the model.
- ***M*** represents the total number of hair colors, and ***m*** specifies the color number of hair assigned to the model.
- ***F*** indicates the total number of form values, and ***f*** specifies the form value assigned to the model
- ***G*** refers to the total total height values, and ***g*** specifies the height value assigned to the model.
- ***Possible_App_Var*** indicates the number of possible combinations of these different terms that contribute to our appearance variety.

Next, we determine the number of possible combinations in our scene according to the various terms defining our models' appearance variety. So we will define our variables as follows: ***H =5*** for the total number of both men and women characters and ***h=1*** for the characters being considered; ***A=5*** for the total number of men's accessories ***A=7*** for the total number of women's accessories and the values of the variable ***a*** represent the number of possible accessories that can used by the model; ***P=4*** for the total number of pants textures and ***p=1*** for the pants textures assigned to the model; ***R =5*** for the total number of shirt textures and ***r=1*** for the shirt textures assigned to the model; ***B=4*** for the total number of body textures and ***b=1*** for the body textures assigned to the model; ***N =4*** for the total number of hair types and ***n=1*** for the hair types assigned to the model; ***M =7*** for the total number of hair colors and ***m=1*** for the color number assigned to the model; ***F=8*** for the total number of form values and ***f=1*** for the value of form assigned to the model; ***G =8*** for the total number of height values and ***g=1*** for the value of height assigned to the model.

Using these variables, we derive the following formula:

- Possible appearance combinations of five men:

$(C_5^1.(C_5^1 + C_5^2 + C_5^3 + C_5^4 + C_5^5).C_4^1.C_5^1.C_4^1.C_4^1\ .C_7^1.C_8^1.C_8^1) *$
$5 =(5*(5+10+10+5+1)\ *4*5*4*4*7*8*8)\ *5$
$= 111104000.$

- Possible appearance combinations of ten women:

$(C_{10}^1.(C_7^1 + C_7^2 + C_7^3 + C_7^4 + C_7^5 + C_7^6 +$
$C_7^7).C_4^1.C_5^1.C_4^1.C_4^1\ .C_7^1.C_8^1.C_8^1)*10=(10*(7+21+35+35+21+7+1)$
$*4*5*4*4*7*8*8)\ *10) \qquad = 1820672000.$

- Possible appearance combinations of two young children:

$(C_2^1.(C_5^1 + C_5^2 + C_5^3 + C_5^4 + C_5^5).C_4^1.C_5^1.C_4^1.C_4^1\ .C_7^1.C_8^1.C_8^1) *$
$2 =(2*(5+10+10+5+1)\ *4*5*4*4*7*8*8)\ *2$
$= 17776640.$

- Possible appearance combinations of older adult:

$$(C_1^1.(C_5^1 + C_5^2 + C_5^3 + C_5^4 + C_5^5).C_4^1.C_5^1.C_4^1.C_8^1.C_8^1) * 1 = (1*(5+10+10+5+1)*4*5*4*8*8)*1 = 158720.$$

After examining the possible modifications for the appearance and shape of our different characters, we obtained many possibilities for creating diverse appearance characters. Our model has further opportunities for diversity when we add different genders of characters and enhance the number of textures of clothes and skin, as well as the amount of hair and color styles. This strategy allows us to include more variation and individuality in our character models.

### 4.5.3 The performance evaluation of our animation variation approach by exploring the maximum combinations of elements

To ensure the precision of our animation results, we identify combinations of elements that increase diversity. We start by selecting some aspects like movement type and the speed affected for each character. This process calculates the various ways to select these factors at the time. We must identify the maximum combinations for these elements across multiple scenarios, where ***H*** represents the number of all characters used in our model, ***T*** is the maximum number of animation types utilized by each character, and ***V*** indicates the number of the maximum speed values assigned to each character [1].
***Max_Combinations_Anim*** is the value of the maximum feasible combination terms of each model that contribute to our animation variety.

$$H * T * V = \text{Max\_Combinations\_Anim} \tag{4.3}$$

- The maximum feasible models combination terms that contribute animation variety:

$$18 * 10 * 50 = 9000.$$

| Number of characters in different scenarios | Visual animation diversity considering maximum combination values |
|---|---|
| Scenario1 ,H=18 | 9000 |
| Scenario2 ,H=72 | 36000 |
| Scenario3 ,H=144 | 72000 |
| Scenario4 ,H=720 | 360000 |

**Table 4.3**:The maximum feasible combinations values that enhance visual animation diversity of different templates on different scenarios.

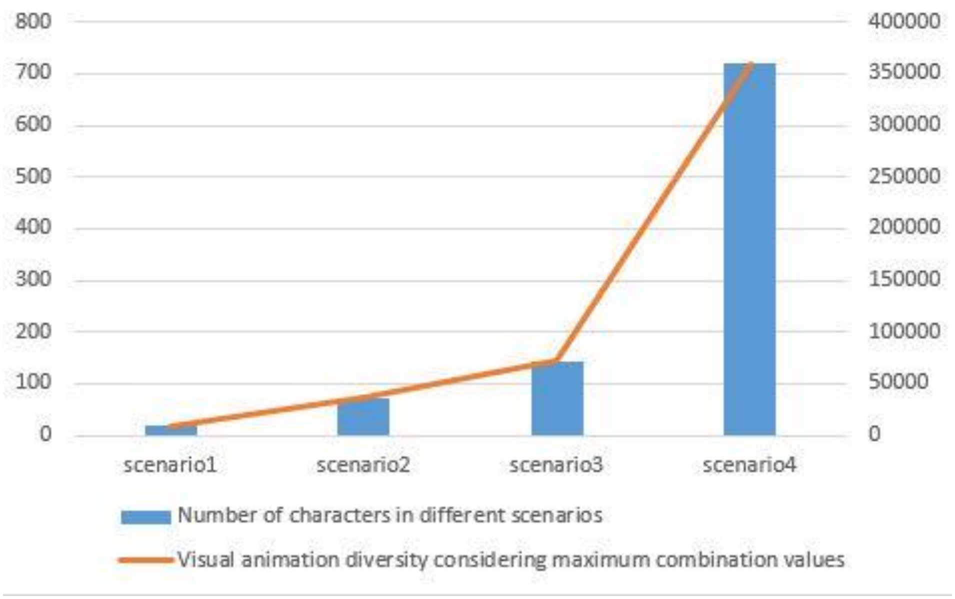


**Figure 4.12:** Exploring the maximum feasible combinations of elements in our diverse animations according to the number of characters involved

We noticed that our system may produce an immense variety of motion types by combining essential factors, such as the number of characters and types of movements available at various speeds. As the number of characters in different settings increases, the variety of possible motion combinations also expands. For example, using 1080 characters, we may generate around 540000 distinct combinations in our system. This expansion significantly increases the variety of our characters' movements. We can add more character types and other different motion styles to further this diversity, resulting in more visually dynamic and accurate animations. It significantly enhances our capacity to provide various animations for crowd scenes [1].

### 4.5.4 Evaluation of the performance of possible combinations of our approach for changing animations using a particular number of element sets for different characters.

The following process describes the different concepts for increasing animation variation by selecting elements simultaneously. We have to determine the number of possible term possibilities across various contexts.

$$\mathrm{C}_{\mathrm{H}}^{\mathrm{h}}.\,\mathrm{C}_{T}^{\mathrm{t}}.\,\mathrm{C}_{\mathrm{V}}^{\mathrm{v}}. = \mathrm{Possible\_Animation\_Variety} \tag{4.4}$$

To begin, we assign precise labels to each item relating to an account:

- ***H*** indicates the total number of characters, while ***h*** indicates the number of considered characters.
- ***T*** represents the total types of animation utilized by our models, with ***t*** specifying the type of animation used by the model.
- ***V*** refers to the height value of speed animation for our characters, and ***v*** signifies the speed value assigned to the model during its animation.

- ***Possible_Animation_Variety*** indicates the number of possible combinations of these different terms that contribute to our animation variety.

Next, we determine the number of combinations in our scene according to the various terms defining the animation variety for our ten distinct models. So we will define our variables as follows: ***H*** **=10** for the total number of characters and ***h=1*** for the characters being considered; ***T=10*** for the total number of types of animation utilized by our models and ***t=1, t=4*** for specifying the number of types of animation used by the model during its animation; ***V=50*** represent the height value of the speed animation for our characters and ***v=1*** signifies the number of speed value assigned to the model. Using these variables, we derive the following formula:

- Possible combinations of our different characters:

$$(C_{10}^{1}.(C_{10}^{1}+C_{10}^{4}).(C_{50}^{1}))*18=(10*(10+210)*(50))*18$$
$$=1980000.$$

### 4.5.5 The performance evaluation of our appearance and animation variation approach by exploring the combinations of elements sets

(Refer to Figure 4.14), we utilize different complex environments, like cities, Haram mosque area, and forests. These places are populated with a variety of templates for virtual humans. The crowd within this environment interacts and moves within our scene, presenting a more realistic portrayal. This realism is emphasized due to the elements that form the appearance and shape of the characters in the environment. The virtual humans move in a different complex environment modeled with several polygons. Each character is articulated with a skeleton and fully skinned and animated with a specific type of animation to detect the happening of collision. We tested this factor because it might help increase variety. So, we were not just evaluating appearance and ignoring the motion.
motion.

To confirm the accuracy of our results, we determine the combinations of the total number of elements used to enhance the variety of appearance and animation crowd. We determine the number of all elements. The results below include all feasible situations in our virtual environment when all components that ensure variations in appearance, height, form, and animation are applied:

$$H*A*P*R*B*N*M*F*G*T*V=\text{Max\_Comb} \qquad (4.5)$$

- The maximum feasible men's combination terms that contribute appearance and animation variety:

$$(1*5*4*5*4*4*7*8*8*10*50)*5=(358400000)*5$$
$$=1792000000.$$

- The maximum feasible women's combination terms that contribute appearance and animation variety:

$$(1*7*4*5*4*4*7*8*8*10*50)*10 = (501760000)*10 = 5017600000.$$

- The maximum feasible young children's combination terms that contribute appearance and animation variety:

$$(1*5*4*5*4*4*7*8*8*10*50)*2 = (358400000)*2 = 716800000.$$

- The maximum feasible older adult's combination terms that contribute appearance and animation variety:

$$(1*5*4*5*4*8*8*10*50)*1 = (12800000)*1 = 12800000.$$

| Number of characters in different scenarios | Visual appearance and animation diversity considering maximum combination values |
|---|---|
| Scenario1 ,H=18 | 7539200000 |
| Scenario2 ,H=72 | 30156800000 |
| Scenario3 ,H=144 | 60313600000 |
| Scenario4 ,H=720 | 301568000000 |

**Table 4.4:**The feasible combinations values that enhance visual appearance and animation diversity of different templates on different scenarios.

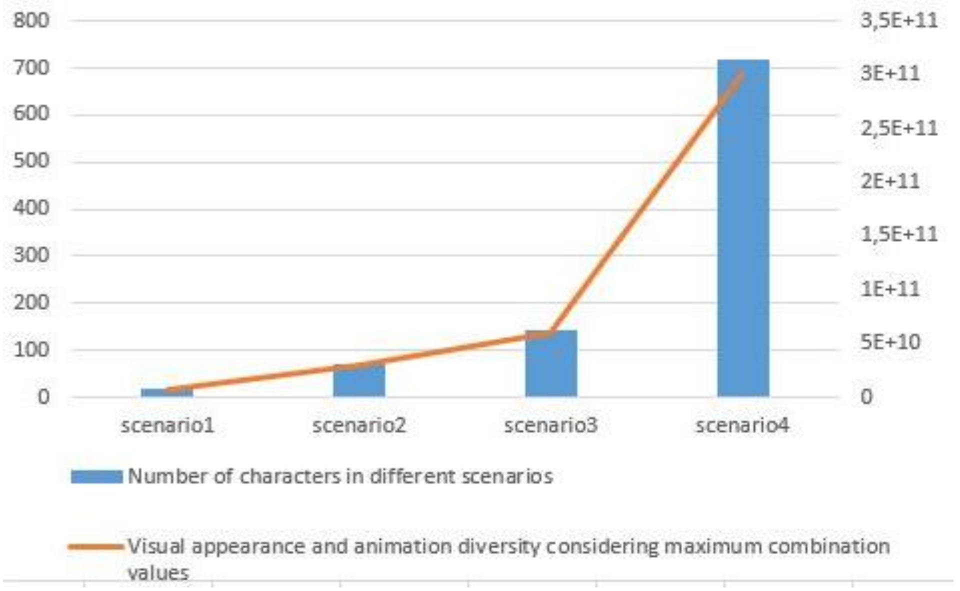
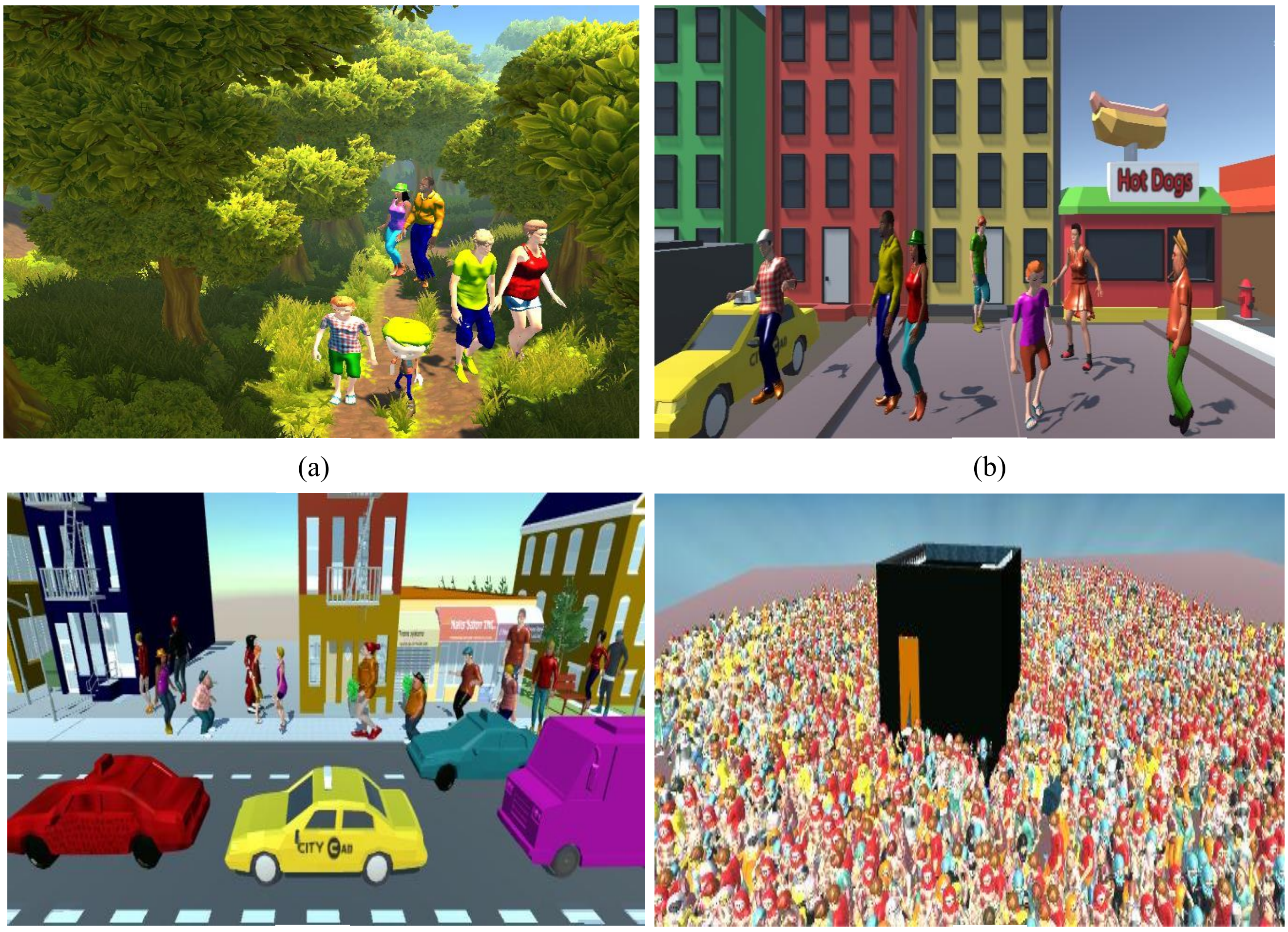


**Figure 4.13**:Exploring the maximum combinations of elements in our diverse appearance and animation according to the number of characters involved



(a) (b)

(c) (d)

**Figure 4. 14**:(a), (b), (c) and (d) represent examples of the feasible combinations that enhance visual appearance and animations diversity of different templates on different complexes environments.

According to the table below, four different environments depend on the type, number of characters in each environment, and the types of character movement. For example, in scenario (d), during the walking around the Kaaba, movements will be limited to walking and turning. It should be noted that walking types change between a lady, a man, a young child, and an older adult.

In addition, we eliminated the consideration of hair type and color for men and women in this situation and the use of certain accessories, such as a phone, which should not be utilized in this context. Because of the variances in environments regarding the number of characters, types of movements, and amount of accessories, the calculations for time in seconds were different after their simulation. For example, the first environment (a) duration was 61.2 seconds with only 10 characters, whereas the time in the fourth environment (d) was 10.1 seconds with 1000 characters. It is caused by the complexity of the environment, which is generated by the significant number of components included in each environment. We also timed 58.7 seconds in an environment with 400 characters and 57.3 seconds in an environment with 800 characters. As the number of components improves, the time value decreases, resulting in more variation in appearance and movement.

| Types of Scenar-ios | Visual appearance diversity considering feasible combination values | Variety of visual appearance with feasible combination results | Visual animation diversity considering feasible combination values | Variety of visual animations with feasible combination results | Total appearance and animation diversity with feasible combinations | Frames per second (ms) |
|---|---|---|---|---|---|---|
| (a) | H=10, A=10, P=4, R=5, B=4, N=4 M=7, F=8, G=8 | 14336000 | T=10 V=10 | 100 | 1433600000 | 61.2 |
| (b) | H=400, A=10, P=4 R=5, B=4, N=4, M=7, F=8, G=8 | 573440000 | T=10 V=10 | 100 | 57344000000 | 58.7 |
| (c) | H=800, A=10, P=4, R=5, B=4, N=4, M=7, F=8, G=8 | 1146880000 | T=10 V=10 | 100 | 114688000000 | 57.3 |
| (d) | H=1000, A=7, P=4, R=5, B=4 F=8, G=8 | 35840000 | T=5 V=5 | 25 | 896000000 | 10.1 |

**Table 4.5**:Perform the frames per second in various environments, considering the combinations of components that impact appearance and animation based on the number of characters involved.

## 4.6 Conclusion

In this section, we start to discuss Unity 3D in the context of our research. We present the conceptual foundation of Unity 3D and the programming language C#. Next, we give an overview of our system, explaining how we combine and improve different elements in a system. We next describe the results we obtained in calculating the amount of the maximum and possible appearance and animation variants across different scenarios and discuss their implementation.

This system facilitates real-time crowd simulation, incorporating diverse approaches across appearance, shape, animation, color, texture, clothing, accessories, and rendering in the same system.

# General Conclusion

Computer graphics, Computer graphics, movies, and video games present exciting challenges for scientists and designers. One such issue is the virtual crowd simulation, requiring distinct virtual beings with different characteristics and personalities. This thesis integrates appearance, animation, and collision avoidance into simulations. Our contribution is summary as follows:

The first contribution is to form a crowd with different characteristics and confirm that there are no similarities between its components. It is important to perceive several characters who appear and interact in the same area. However, it is important to improve that each one differs from the other in terms of appearance, hair, skin, accessories, and clothes.

Secondly, it is essential to provide a crowd that moves and animates in a distinct manner from others. We will conclude that each moves uniquely, at varying speeds and in specific styles, reflecting their routines, personal styles, and objectives.
The third contribution focuses on implementing collision avoidance within crowds by applying raycasting, Newton's laws, and the principles of momentum and kinetic energy conservation. It ensures that interactions between pedestrians occupying the same space are realistically prevented, maintaining consistency with real-world physical behavior and ensuring collisions and results are consistent with the actual reality.

The strategies examined in this thesis have been applied in various contexts, including urban areas, open spaces, street settings, and pilgrimage sites around the Kaabah.
Our approaches improved the realism and visual quality of our crowd simulation system. These practical and adaptable techniques make them appropriate for creating believable virtual crowds. It emphasizes significant developments in computer graphics, with feasible uses in gaming, virtual reality, and animation movies.
The fourth contribution focuses on providing the accuracy of our appearance and animation results by identifying the maximum and possible combinations of parts that increase diversity.

We determined specific elements such as the number of characters, accessories, clothes and skin textures, hair types and colors, character sizes and heights, movement styles, and speeds for each character. We obtained convincing results that demonstrate our reason for contributing to crowd diversity.

We have outlined several areas for future enhancement. First, exploring crowd behavior algorithms could significantly improve the realism of our system by using behavioral rules for decision-making.
Another key improvement involves refining how characters animate and avoid obstacles, ensuring that their positions, velocities, and motion properties account for factors like age and exceptional cases like people with disabilities.
Expanding our system to support more accessories attached to different model joints would enhance its versatility. These accessories should be compatible with various animations and adjust to movement speed and type to achieve more realistic and practical outcomes.

We intend to classify characters and provide characteristics that affect their appearance based on behavior, kind, and age. For example, facial characteristics and physical movements can be personalized, and certain accessories can be assigned based on the character's behavior and actions. For example, an elderly individual will have different behavior, movement, and facial characteristics than a young one, and women will differ from men. We aim to apply reinforcement learning for character animation and obstacle avoidance, enabling adaptive crowd navigation across diverse scenarios. It will facilitate effective autonomous navigation for multiple agents in more complex environments.

# References

[1] Dridi, I., & Cherif, F. (2025). Enhancing real-time animation: Ensuring distinctiveness in crowd dynamics through physics-based collision avoidance. ITEGAM-JETIA, 11(52), 237-246. https://doi.org/10.5935/jetia.v11i52.1677

[2] Beacco, A. (2014). Simulation, animation and rendering of crowds in real-time (Master's thesis, Universitat Politècnica de Catalunya). https://upcommons.upc.edu/handle/2117/95561.

[3] Cherif, F. (2006). Animation comportementale : Simulation de foules d'humains virtuels (Thèse de Doctorat d'État en Informatique). Université Mohamed Khider – Biskra, Algérie.

[4] RVO2 Library. (n.d.). RVO2: Optimal reciprocal collision avoidance. http://gamma.cs.unc.edu/RVO2/. Accessed June 2025.

[5] Reynolds, C. W. (n.d.). *Boids: Background and update*. http://www.red3d.com/cwr/boids/. Accessed May 2025.

[6] van den Berg, J., Guy, S. J., Lin, M., & Manocha, D. (2009). Reciprocal n-body collision avoidance. In Proceedings of the Symposium on Robotics Research.

[7] Ren, C., Yang, C., & Jin, S. (2009). Agent-based modeling and simulation on emergency evacuation. In J. Zhou (Ed.), Complex sciences (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 5, pp. 1451–1461). Springer. https://doi.org

[8] Ozcan, C. Y., & Haciomeroglu, M. (2015). A path-based multi-agent navigation model. The Visual Computer, 31(6–8), 863–872.

[9] Zhou, S., Chen, D., Cai, W., Luo, L., Low, M. Y. H., Tian, F., … Hamilton, B. D. (2010). Crowd modeling and simulation technologies. ACM Transactions on Modeling and Computer Simulation, 20(4), 1–35.

[10] Thalmann, D., & Musse, S. R. (2012). Crowd simulation. Springer.

[11] Gibson, J. J. (1958). Visually controlled locomotion and visual orientation in animals. British Journal of Psychology, 49(3), 182–194.

[12] Schöner, G., & Dose, M. (1992). A dynamical systems approach to task-level system integration used to plan and control autonomous vehicle motion. Robotics and Autonomous

Systems, 10(4), 253–267.

[13] Duchon, A. P., & Warren, W. H., Jr. (2002). A visual equalization strategy for locomotor control: Of honeybees, robots, and humans. Psychological Science, 13(3), 272–278.

[14] Xiong, M., Tang, S., & Zhao, D. (2013). A hybrid model for simulating crowd evacuation. New Generation Computing, 31(3), 211–235.

[15] Warren, W. H., & Fajen, B. R. (2008). Behavioral dynamics of visually guided locomotion. In A. Fuchs & V. K. Jirsa (Eds.), Coordination: Neural, behavioral and social dynamics (pp. 45–75). Springer.

[16] Bonneaud, S., Rio, K., Chevaillier, P., & Warren, W. H. (2012). Accounting for patterns of collective behavior in crowd locomotor dynamics for realistic simulations. In Z. Pan et al. (Eds.), Transactions on edutainment VII (pp. 1–11). Springer.

[17] Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87) (pp. 25–34).

[18] Sewall, J., Wilkie, D., Merrell, P., & Lin, M. C. (2010). Continuum traffic simulation. Computer Graphics Forum, 29(2), 439–448.

[19] Narain, R., H., A., Curtis, S., & Lin, M. C. (2009). Aggregate dynamics for dense crowd simulation. In ACM SIGGRAPH Asia 2009 Papers (pp. 1–8).

[20] Henderson, L. F. (1971). The statistics of crowd fluids. Nature, 229(5284), 381–383.

[21] Hoogendoorn, S. P., & Bovy, P. H. (2005). Pedestrian travel behavior modeling. Networks and Spatial Economics, 5(2), 193–216.

[22] Løvås, G. G. (1994). Modeling and simulation of pedestrian traffic flow. Transportation Research Part B: Methodological, 28(6), 429–443.

[23] Helbing, D., & Molnár, P. (1995). Social force model for pedestrian dynamics. Physical Review E, 51(5), 4282–4286.

[24] Helbing, D., Farkas, I., & Vicsek, T. (2000). Simulating dynamical features of escape panic. Nature, 407, 487–490.

[25] Palechano, N., Badler, N. I., & Allbeck, J. (2022). Virtual crowds: Methods, simulation, and control. Springer Nature.

[26] Felis, M., & Mombaur, K. (2012). Using optimal control methods to generate human walking motions. In Motion in Games (pp. 197–207). Springer.

[27] Singh, S., Kapadia, M., Reinman, G., & Faloutsos, P. (2011). Footstep navigation for dynamic crowds. Computer Animation and Virtual Worlds, 22(2–3), 151–158.

[28] Yersin, B., Maïm, J., Morini, F., & Thalmann, D. (2008). Real-time crowd motion planning: Scalable avoidance and group behavior. The Visual Computer, 24(10), 859-870.

[29] Anh, N. T. N., Daniel, Z. J., Du, N. H., Drogoul, A., & An, V. D. (2012). A hybrid macro–micro pedestrians evacuation model to speed up simulation in road networks. In Advanced agent technology: AAMAS 2011 workshops (pp. 371–383). Springer.

[30] Xiong, M., Lees, M., Cai, W., Zhou, S., & Low, M. Y. H. (2010). Hybrid modelling of crowd simulation. Procedia Computer Science, 1(1), 57–65.

[31] Xiong, M., Cai, W., Zhou, S., Low, M. Y. H., Tian, F., Chen, D., … Hamilton, B. D. (2009). A case study of multi-resolution modeling for crowd simulation. In Proceedings of the 2009 Spring Simulation Multiconference (pp. 1–8).

[32] Sudkhot, P., Wong, K., & Sombattheera, C. (2023). Collision avoidance and path planning in crowd simulation. ICIC Express Letters, 17(1), 13–19.

[33] Rabiaa, C., & Foudil, C. (2016). Toward a hybrid approach for crowd simulation. International Journal of Advanced Computer Science and Applications, 7(1), 1–8.

[34] Thalmann, D., O'Sullivan, C., Yersin, B., Maïm, J., & McDonnell, R. (2007). EG 2007 course on populating virtual environments with crowds. In Eurographics tutorials (pp. 23–123).

[35] Lemercier, S. (2012). Simulation du comportement de suivi dans une foule de piétons : Expérience, analyse et modélisation (Doctoral dissertation, Université de Rennes 1).

[36] Demange, J. (2012). Un modèle d’environnement pour la simulation multiniveau .Master’s thesis.. Université de Technologie de Belfort-Montbéliard, Belfort, France.

[37] Chighoub, R. (2016). Simulation en temps réel de planification d'une foule d'humains virtuels . Magister’s thesis. Université Mohamed Khider — Biskra, Algérie.

[38] Shi, Y., Ondřej, J., Wang, H., & O'Sullivan, C. (2017). Shape up! Perception-based body shape variation for data-driven crowds. In Proceedings of the IEEE Virtual Humans and Crowds for Immersive Environments Conference (pp. 1–7). IEEE.

[39] Mudge, M., Ryan, N. S., & Scopigno, R. (2005). VAST 2005: 6th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage. Eurographics Association.

[40] Dobbyn, S., Hamill, J., O’Connor, K., & O’Sullivan, C. (2005). Geopostors: A real-time geometry/impostor crowd rendering system. In Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games (SI3D ’05) (pp. 95–102).

[41] McDonnell, R., Larkin, M., Dobbyn, S., Collins, S., & O’Sullivan, C. (2008). Clone attack! Perception of crowd variety. In ACM SIGGRAPH 2008 Papers. https://doi.org/10.1145/1399504.1360625

[42] Allen, B., Curless, B., Popović, Z., & Hertzmann, A. (2006). Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis. In Proceedings of the Computer Animation Conference (pp. 147–156).

[43] Chen, Y., Cheng, Z.-Q., & Martin, R. R. (2015). Parametric editing of clothed 3D avatars. The Visual Computer, 32(11), 1405–1414. https://doi.org/10.1007/s00371-015-1120-0

[44] Fleming, R., Mohler, B. J., Romero, J., Black, M. J., & Breidt, M. (2016). Appealing female avatars from 3D body scans: Perceptual effects of stylization. In Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP).

[45] O'Sullivan, C. (2009). Variety is the spice of (virtual) life. In Proceedings of the International Workshop on Motion in Games (pp. 84–93). Springer.

[46] Chalás, I., Urbanová, P., Juřík, V., Ferková, Z., Jandová, M., Sochor, J., & Kozlíková, B. (2017). Generating various composite human faces from real 3D facial images. The Visual Computer, 33(4), 443–458. https://doi.org/10.1007/s00371-016-1277-1.

[47] Lister, W., Laycock, R. G., & Day, A. M. (2009). Geometric diversity for crowds on the GPU. In Proceedings of the 17th International Conference on Computer Graphics, Visualization and Computer Vision (WSCG 2009) (pp. 113–119).

[48] Benson, P., & Perrett, D. (1992). Face to face with the perfect image. New Scientist, 133, 18–19.

[49] Little, A. C., & Hancock, P. J. (2002). The role of masculinity and distinctiveness in judgments of human male facial attractiveness. British Journal of Psychology, 93(4), 451–464.

[50] Fink, B., Grammer, K., & Matts, P. J. (2006). Visible skin color distribution plays a role in the perception of age, attractiveness, and health in female faces. Evolution and Human Behavior, 27(6), 433–442.

[51] Dong, Y., & Peng, C. (2019). Real-time large crowd rendering with efficient character and instance management on GPU. International Journal of Computer Games Technology, 2019.

[52] Johansson, D., & Ravantti, R. (2020). Viewer perception of clones within a simulated crowd. Proceedings of the International Conference on Computer Graphics and Visualization.

[53] Hu, K., Haworth, B., Berseth, G., Pavlovic, V., Faloutsos, P., & Kapadia, M. (2021). Heterogeneous crowd simulation using parametric reinforcement learning. IEEE Transactions on Visualization and Computer Graphics, 27(12), 4888–4902.

[54] Mixamo. (n.d.). Mixamo: 3D character creation and animation. Adobe. from https://www.mixamo.com. Accessed June 2025.

[55] Magnenat-Thalmann, N., Seo, H., & Cordier, F. (2004). Automatic modeling of virtual humans and body clothing. Journal of Computer Science and Technology, 19(5), 575–584.

[56] Thalmann, D., Grillon, H., Maïm, J., & Yersin, B. (2009). Challenges in crowd simulation.

In 2009 International Conference on CyberWorlds (pp. 1–12). IEEE.

[57] McDonnell, R., Larkin, M., Hernández, B., Rudomin, I., & O'Sullivan, C. (2009). Eye-catching crowds: Saliency based selective variation. ACM Transactions on Graphics, 28(3), 1–10.

[58] De Heras Ciechomski, P., Schertenleib, S., Maïm, J., & Thalmann, D. (2005). Reviving the Roman Odeon of Aphrodisias: Dynamic animation and variety control of crowds in virtual heritage. In Proceedings of the 2005 Virtual Reality, Archaeology and Intelligent Cultural Heritage (VAST 2005).

[59] Tecchia, F., Loscos, C., & Chrysanthou, Y. (2002). Visualizing crowds in real-time. Computer Graphics Forum, 21(4), 753–765.

[60] De Heras Ciechomski, P., Schertenleib, S., Maïm, J., Maupu, D., & Thalmann, D. (2005). Real-time shader rendering for crowds in virtual heritage. In Proceedings of VAST (Vol. 5, pp. 1–8).

[61] Dobbyn, S., McDonnell, R., Kavan, L., Collins, S., & O'Sullivan, C. (2006). Clothing the masses: Real-time clothed crowds with variation. In Eurographics Short Presentations (pp. 103–106).

[62] Jonathan, M. (2009). Generating, animating, and rendering varied individuals for real-time crowds .Master’s thesis. Stanford University, Stanford, CA, USA.

[63] Boukhayma, A., Tsiminaki, V., Franco, J. S., & Boyer, E. (2016). Eigen appearance maps of dynamic shapes. In Computer Vision – ECCV 2016: 14th European Conference, Part I (pp. 230–245). Springer International Publishing.

[64] Xie, J. (2012). Research on key technologies based on Unity3D game engine. In Proceedings of the 2012 7th International Conference on Computer Science and Education (ICCSE 2012) (pp. 695–699). https://doi.org/10.1109/ICCSE.2012.6295169

[65] Sanchez, J., & Canton, M. (2003). Lighting and shading. In The PC Graphics Handbook. from https://doi.org/10.1201/9780203010532.ch6. Accessed April 2025.

[66] Zell, E., & McDonnell, R. (2019). Perception of virtual characters. ACM Transactions on Graphics, July. https://doi.org/10.1145/3305366.3328101

[67] Jangra, S., Singh, G., Mantri, A., Angra, S., & Sharma, B. (2023, July). Interactivity development using Unity 3D software and C# programming. In Proceedings of the 2023 14th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1–6). IEEE.

[68] M.T. (2020). Computer graphics shading. University of Freiburg – Computer Science Department. From https://cg.informatik.unifreiburg.de/course_notes/graphics_02_shading.pdf. Accessed May 2025

[69] Foley, J. D., & Van Dam, A. (1996). Illumination models and shading. In Fundamentals of interactive computer graphics (pp. 1–36).

[70] Koulaxidis, G., & Xinogalos, S. (2022). Improving mobile game performance with basic optimization techniques in Unity. Modelling, 3(2), 201–223.

[71] Toledo, L., De Gyves, O., & Rudomín, I. (2014). Hierarchical level of detail for varied animated crowds. The Visual Computer, 30(8), 949–961.

[72] Ruiz, S., Hernández, B., Alvarado, A., & Rudomín, I. (2013). Reducing memory requirements for diverse animated crowds. In Proceedings of Motion on Games (pp. 77–86).

[73] Kavan, L., Dobbyn, S., Collins, S., Žára, J., & O'Sullivan, C. (2008). Polypostors: 2D polygonal impostors for 3D crowds. In Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games (pp. 149–155).

[74] Beacco, A., Andújar, C., Pelechano, N., & Spanlang, B. (2012). Efficient rendering of animated characters through optimized per-joint impostors. Computer Animation and Virtual Worlds, 23(1), 33–47.

[75] Aubel, A., Boulic, R., & Thalmann, D. (2000). Real-time display of virtual humans: Levels of detail and impostors. IEEE Transactions on Circuits and Systems for Video Technology, 10(2), 207–217.

[76] Tecchia, F., & Chrysanthou, Y. (2000). Real-time rendering of densely populated urban environments. In Proceedings of the Eurographics Workshop on Rendering Techniques 2000 (pp. 83–88).

[77] Tecchia, F., Loscos, C., & Chrysanthou, Y. (2002). Image-based crowd rendering. IEEE Computer Graphics and Applications, 22(2), 36–43.

[78] Millan, E., & Rudomin, I. (2006). Impostors and pseudo-instancing for GPU crowd rendering. In Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (pp. 49–55).

[79] Maïm, J., Yersin, B., & Thalmann, D. (2009). Unique character instances for crowds. IEEE Computer Graphics and Applications, 29(6), 82–90.

[80] Wand, M., & Straßer, W. (2002). Multi-resolution rendering of complex animated scenes. Computer Graphics Forum, 21(3), 483–491.

[81] Beacco, A., Spanlang, B., Andújar, C., & Pelechano, N. (2010). Output-sensitive rendering of detailed animated characters for crowd simulation. In CEIG Spanish Conference on Computer Graphics (p. 235).

[82] Boulic, R., Thalmann, N. M., & Thalmann, D. (1990). A global human walking model with

real-time kinematic personification. The Visual Computer, 6(6), 344–358.

[83] Bruderlin, A., & Calvert, T. W. (1989). Goal-directed, dynamic animation of human walking. In Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques (pp. 233–242).

[84] Faloutsos, P., Van de Panne, M., & Terzopoulos, D. (2001). Composable controllers for physics-based character animation. In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (pp. 251–260).

[85] Popović, Z., & Witkin, A. (1999). Physically based motion transformation. In Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (pp. 11–20).

[86] Kim, M., Hwang, Y., Hyun, K., & Lee, J. (2012). Tiling motion patches. In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (pp. 117–126).

[87] Heck, R., & Gleicher, M. (2007). Parametric motion graphs. In Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games (pp. 129–136).

[88] Feng, A., Huang, Y., Kallmann, M., & Shapiro, A. (2012). An analysis of motion blending techniques. In Motion in Games: 5th International Conference, MIG 2012, Rennes, France, November 15–17, 2012, Proceedings (pp. 232–243). Springer.

[89] Autodesk. (n.d.). 3D Studio Max. from https://usa.autodesk.com/3ds-max/. Accessed May 2025.

[90] Autodesk. (n.d.). Maya. from https://usa.autodesk.com/maya/. Accessed June 2025.

[91] Ikemoto, L., Arikan, O., & Forsyth, D. (2006). Knowing when to put your foot down. In Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games (pp. 49–53).

[92] Kovar, L., Gleicher, M., & Schreiner, J. (2002). Footskate cleanup for motion capture. In Proceedings of SIGGRAPH 2002.

[93] Carnegie Mellon University. (n.d.). CMU Graphics Lab motion capture database. from http://mocap.cs.cmu.edu/. Accessed June 2025.

[94] Boukhayma, A., & Boyer, E. (2017). Controllable variation synthesis for surface motion capture. In 2017 International Conference on 3D Vision (3DV) (pp. 309–317). IEEE.

[95] Mileff, P. (2023). Collision detection in 2D games. Production Systems and Information Engineering, 11(3), 10–26.

[96] Abdullah, N. A. B. M., Bade, A. B., & Kari, S. (2009). A review of collision avoidance techniques for crowd simulation. In 2009 International Conference on Information and Multimedia Technology (pp. 388–392). IEEE.

[97] Reynolds, C. W. (1999). Steering behaviors for autonomous characters. In Proceedings of the Game Developers Conference (Vol. 1999, pp. 763–782).

[98] Morini, F., Yersin, B., Maïm, J., & Thalmann, D. (2007). Real-time scalable motion planning for crowds. In 2007 International Conference on Cyberworlds (CW'07) (pp. 144–151). IEEE.

[99] Musse, S. R., & Thalmann, D. (1997). A model of human crowd behavior: Group inter-relationship and collision detection analysis. In Computer Animation and Simulation '97: Proceedings of the Eurographics Workshop (pp. 39–51). Springer Vienna.

[100] Golas, A., Narain, R., & Lin, M. (2013). Hybrid long-range collision avoidance for crowd simulation. In Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (pp. 29–36).

[101] Sung, M., Chenney, S., & Gleicher, M. (2005). Detecting collisions in graph-driven motion synthesis. University of Wisconsin–Madison, Department of Computer Sciences.

[102] Abdullasim, N., Sulaiman, H. A., & Ahmad, I. (2020). Collision detection techniques in crowd simulations. In Proceedings of Mechanical Engineering Research Day 2020 (pp. 174–175).

[103] Wang, D., Zhang, J., Jin, J., & Mao, X. (2021). Local collision avoidance algorithm for an unmanned surface vehicle based on steering maneuver considering COLREGs. IEEE Access, 9, 49233–49248.

[104] Yu, Q., Liu, K., Yang, Z., Wang, H., & Yang, Z. (2021). Geometrical risk evaluation of collisions between ships and offshore installations using rule-based Bayesian reasoning. Reliability Engineering & System Safety, 210, 107474. https://doi.org/10.1016/j.ress.2021.107474

[105] Tam, C., Bucknall, R., & Greig, A. (2009). Review of collision avoidance and path planning methods for ships in close-range encounters. Journal of Navigation, 62(3), 455–476. https://doi.org/10.1017/S0373463308005134

[106] Wang, C., Wang, N., Xie, G., & Su, S. F. (2022). Survey on collision-avoidance navigation of maritime autonomous surface ships. In Offshore robotics (pp. 1–33). Springer. https://doi.org/10.1007/978-981-16-2078-2_1

[107] Tsou, M. C., Kao, S. L., & Su, C. M. (2010). Decision support from genetic algorithms for ship collision avoidance route planning and alerts. Journal of Navigation, 63(1), 167–182. https://doi.org/10.1017/S037346330999021X

[108] Shaobo, W., Yingjun, Z., & Lianbo, L. (2020). A collision avoidance decision-making system for autonomous ships based on a modified velocity obstacle method. Ocean

Engineering, 215, 107910.

[109] Fiskin, R., Atik, O., Kisi, H., Nasibov, E., & Johansen, T. A. (2021). Fuzzy domain and meta-heuristic algorithm-based collision avoidance control for ships: Experimental validation in virtual and real environments. Ocean Engineering, 220, 108502. https://doi.org/10.1016/j.oceaneng.2020.108502

[110] Ding, Z., Zhang, X., Wang, C., Li, Q., An, L., Ding, Z., et al. (2021). Intelligent collision avoidance decision-making method for unmanned ships based on driving practice. Chinese Journal of Ship Research, 16(1), 31. https://doi.org/10.19693/j.issn.1673-3185.01781

[111] Yuan, W., & Gao, P. (2022). Model predictive control-based collision avoidance for autonomous surface vehicles in congested inland waters. Mathematical Problems in Engineering, 2022, 7584489. https://doi.org/10.1155/2022/7584489

[112] Paris, S., & Donikian, S. (2009). Activity-driven populace: A cognitive approach to crowd simulation. IEEE Computer Graphics and Applications, 29(4), 34–43.

[113] Shao, W., & Terzopoulos, D. (2005). Autonomous pedestrians. In Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (pp. 19–28).

[114] van Basten, B. J., & Egges, A. (2009). Path abstraction for combined navigation and animation. In International Workshop on Motion in Games (pp. 182–193). Springer.

[115] Chang, J. Y., & Li, T. Y. (2008). Simulating virtual crowd with fuzzy logics and motion planning for shape template. In 2008 IEEE Conference on Cybernetics and Intelligent Systems (pp. 131–136). IEEE.

[116] Rafai, A. N. A., Adzhar, N., & Jaini, N. I. (2022). A review on path planning and obstacle avoidance algorithms for autonomous mobile robots. Journal of Robotics, 2022, Article ID 2022.

[117] Güneri, Ö. İ., & Durmuş, B. (2020). Classical and heuristic algorithms used in solving the shortest path problem and time complexity. Konuralp Journal of Mathematics, 8(2), 279–283.

[118] Patle, B. K., Pandey, A., Parhi, D. R. K., & Jagadeesh, A. J. D. T. (2019). A review: On path planning strategies for navigation of mobile robots. Defence Technology, 15(4), 582–606.

[119] Chelsea, S., & Kelly, C. (2012). Fuzzy logic unmanned aerial vehicle motion planning. Advances in Fuzzy Systems, 2012, Article ID 989051.

[120] Gonzalez, R., Kloetzer, M., & Mahulea, C. (2017). Comparative study of trajectories resulted from cell decomposition path planning approaches. In 2017 21st International Conference on System Theory, Control and Computing (ICSTCC) (pp. 49–54). IEEE.

[121] Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In Proceedings of the 1985 IEEE International Conference on Robotics and Automation (Vol. 2, pp. 500–505). IEEE.

[122] Abiyev, R., Ibrahim, D., & Erin, B. (2010). Navigation of mobile robots in the presence of obstacles. Advances in Engineering Software, 41(10–11), 1179–1186

[123] Kavraki, L. E., Svestka, P., Latombe, J. C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12(4), 566–580.

[124] Ab Wahab, M. N., Nefti-Meziani, S., & Atyabi, A. (2020). A comparative review on mobile robot path planning: Classical or meta-heuristic methods? Annual Reviews in Control, 50, 233–252.

[125] Patle, B. K., Pandey, A., Parhi, D. R. K., & Jagadeesh, A. J. D. T. (2019). A review: On path planning strategies for navigation of mobile robot. Defence Technology, 15(4), 582-606.

[126] Gilmartin, M. J. (2005). Introduction to autonomous mobile robots [Book review of Roland Siegwart & Illah R. Nourbakhsh]. Robotica, 23(2), 271–272.

[127] Prakash, K. V., Anto, V. R., & Gnanaprakash, M. (2015). Study on mobile robot path planning: A review. International Journal of Applied Engineering Research, 10(57).

[128] Rachmawati, D., & Gustin, L. (2020). Analysis of Dijkstra's algorithm and A* algorithm in shortest path problem. Journal of Physics: Conference Series, 1566(1), 012061.

[129] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269–271.

[130] M. J. Gilmartin. (2005). "Introduction to autonomous mobile robots roland siegwart and illah R. Nourbakhsh," Robotica, MIT Press, vol. 23, no. 2, pp. 271-272.

[131] Tzafestas, S. G. (2018). Mobile robot control and navigation: A global overview. Journal of Intelligent and Robotic Systems, 91(1), 35–58.

[132] Zadeh, L. A. (1965). Fuzzy sets. Information and Control, 8(3), 338–353.

[133] Muhammad, A., Ali, M. A., & Shanono, I. H. (2020). Path planning methods for mobile robots: A systematic and bibliometric review. ELEKTRIKA, 19(3), 14–34.

[134] Vachtsevanos, G., & Hexmoor, H. (1986). A fuzzy logic approach to robotic path planning with obstacle avoidance. In 1986 25th IEEE Conference on Decision and Control (Vol. 25, No. 1, pp. 1262–1264). IEEE.

[135] Zacksenhouse, M., de Figueiredo, R., & Johnson, D. (1988, December). A neural

network architecture for cue-based motion planning. In Proceedings of the 27th IEEE Conference on Decision and Control (pp. 324–327). Austin, TX, USA: IEEE.

[136] Adham, A., & David, M. (2013). Review of classical and heuristic based navigation and path planning approaches. International Journal of Advancements in Computing Technology, 5(14), 1–15.

[137] Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks IV. Perth, WA, Australia.

[138] Ellips, M., & Davoud, S. (2007). Classic and heuristic approach in robot motion planning: A chronological review. International Journal of Mechanical & Mechatronics Engineering, 1(5), 228–233.

[139] Rubio, F., Valero, F., & Llopis-Albert, C. (2019). A review of mobile robots: Concepts, methods, theoretical framework. International Journal of Advanced Robotic Systems, 16(2), 1–22. https://doi.org/10.1177/172988141983959

[140] Kumar, P., Pandey, K., Sahu, C., Chhotray, A., & Parhi, D. R. K. (2017, November). A hybridized RA-APSO approach for humanoid navigation. In Proceedings of the 2017 Nirma University International Conference on Engineering (NUiCONE). Ahmedabad, India.

[141] Gao, M., Ding, P., & Yang, Y. (2015, September). Time-optimal trajectories planning of industrial robots based on particle swarm optimization. In Proceedings of the 2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC). Qinhuangdao, China.

[142] Castillo, O., Martinez-Marroquin, R., Melin, P., Valdez, F., et al. (2012). Comparative study of bio-inspired algorithms applied to the optimization of Type-1 and Type-2 fuzzy controllers for an autonomous mobile robot. Information Sciences, 192, 19–38. https://doi.org/10.1016/j.ins.2011.12.034

[143] Rendón, M. A., & Martins, F. F. (2017). Path following control tuning for an autonomous unmanned quadrotor using particle swarm optimization. IFAC-PapersOnLine, 50(1), 325–330. https://doi.org/10.1016/j.ifacol.2017.03.103

[144] He, B., Ying, L., Zhang, S., et al. (2015). Autonomous navigation based on unscented-Fast SLAM using particle swarm optimization for autonomous underwater vehicles. Measurement, 71, 89–101.

[145] Zhang, Q., & Li, S. (2007, April). A global path planning approach based on particle swarm optimization for a mobile robot. In Proceedings of the 7th WSEAS International Conference on Robotics, Control & Manufacturing Technology. Stevens Point, WI, USA.

[146] Bremermann, H. (1958). The evolutionary of intelligence: The nervous system as a model of its environment. Seattle, WA: Mathematics, Washington.

[147] Kumar, A., Kumar, P. B., & Parhi, D. R. (2018). Intelligent navigation of humanoids in cluttered environments using regression analysis and genetic algorithm. Arabian Journal for Science and Engineering, 43(12), 7655–7678. https://doi.org/10.1007/s13369-018-3372-7

[148] JiaWang, C., Zhu, H., Zhang, L., & Sun, Y. (2018). Research on fuzzy control of path tracking for underwater vehicles based on genetic algorithm optimization. Ocean Engineering, 156, 217–223. https://doi.org/10.1016/j.oceaneng.2018.03.041

[149] Roberge, V., Tarbouchi, M., & Labonte, G. (2018). Fast genetic algorithm path planner for fixed-wing military UAV using GPU. IEEE Transactions on Aerospace and Electronic Systems, 54(5), 2105–2117. https://doi.org/10.1109/TAES.2018.2813298

[150] Roberge, V., & Tarbouchi, M. (2018). Massively parallel hybrid algorithm on embedded graphics processing unit for unmanned aerial vehicle path planning. International Journal of Digital Signals and Smart Systems, 2(1), 68–93.

[151] Yang, X., & Deb, S. (2009, December). Cuckoo search via Levy flights. In Nature and Biologically Inspired Computing (NaBIC 2009) (pp. 210–214). Coimbatore, India: IEEE.

[152] Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Erciyes University, Engineering Faculty, Computer Engineering Department.

[153] Safari, M., & Mahjoob, M. (2009, September). Bee colony algorithm for real-time optimal path planning of mobile robots. In Proceedings of the ICSCCW: Computing with Words and Perceptions in System Analysis, Decision and Control (pp. 1–4). Famagusta, North Cyprus.

[154] Contreras-Cruz, M. A., Ayala-Ramirez, V., & Hernandez-Belmonte, U. H. (2015). Mobile robot path planning using artificial bee colony and evolutionary programming. Applied Soft Computing, 30, 319–328. https://doi.org/10.1016/j.asoc.2015.01.010

[155] Dridi, I., Bouguetitiche, A., & Cherif, F. (2025, January). Individuality within crowd: Distinctive features to ensure a diverse crowd appearance. In Proceedings of the 2025 International Symposium on Innovative Informatics of Biskra (ISNIB) (pp. 1–6). Biskra, Algeria.

[156] Abdullasim, N., Suaib, N. M., Bade, A., Pan, Z., & Yuan, Q. (2014). Dynamic field of view as collision detection for autonomous multi-agent systems. International Journal of Interactive Digital Media, 2(1), 35–40.

[157] Bettini, A. (2016). A course in classical physics 1: Mechanics (pp. 280–289). Springer

International Publishing, Switzerland.

[158] Hernández, A. G. Y., & Alberú, M. del P. S. (2021). Modeling the interaction of an elastic collision between two objects. Journal of Physics: Conference Series, 1929(1), 012016. https://doi.org/10.1088/1742-6596/1929/1/012016 .

[159] Čepič, M. (2019). Elastic collisions of smooth spherical objects: Finding final velocities in four simple steps. American Journal of Physics, 87(3), 200–207. https://doi.org/10.1119/1.5089753.