Ministry of Higher Education and Scientific Research

ACADEMIC YEAR 2022/2023

Private International Higher School of Polytechnic

END OF STUDIES PROJECT

In order to obtain the National Diploma in Engineering

Field of study: Software Engineering

Entitled

Integrating the Learned Motion Matching Concept for smart character animation in Unreal engine 5

Internship place:	Author:	Supervisors:
Lanterns Studio	Mr. Halloul tarek	Mr. Ben Saad Montassar Mr. Ben Meriam Oussama

Dedications

I would like to dedicate this report to my family, who have been my constant support throughout my academic journey. Their unwavering love and encouragement have been instrumental in helping me achieve this milestone.

I would also like to thank my professors and mentors, whose guidance and expertise have challenged and inspired me to become a better student and engineer.

Lastly, I dedicate this report to all the individuals who have paved the way for me to pursue higher education, and to those who continue to advocate for equal opportunities for all.

This achievement is not only mine, but also theirs.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Mr. Montassar Ben sâad, for his/her invaluable guidance, support, and expertise throughout my study. His insights and feedback have been instrumental in shaping this report and my overall academic journey. I would also like to thank Lanterns Studio, especially the team at Lanterns Studio, for providing me with the opportunity to complete my internship and gain practical experience in the field of game development.

I am grateful for the support and mentorship provided by the team, which has helped me develop my skills and grow both personally and professionally.

Finally, I would like to extend my gratitude to all the individuals who have supported and encouraged me throughout my academic journey, including my family and friends. Without their unwavering support, this accomplishment would not have been possible.

Table of contents

Table of figures

List of tables

General introduction

Motion matching technology has revolutionized the field of game development, allowing for more natural and fluid character movements in games. This technology uses machine learning algorithms to match pre-recorded animations to the movements of characters in Realtime, creating a more realistic and immersive gaming experience for players.

In recent years, motion matching has gained widespread adoption in the gaming industry, with major developers such as Epic Games, Ubisoft, and EA Sports incorporating it into their games. One of the most notable examples is Epic Games' Fortnite, which uses motion matching to create seamless and dynamic character movements in its game world.

Ubisoft's Assassin's Creed franchise is another excellent example of the use of motion matching in game development. The franchise has been using motion matching since its 2017 release, Assassin's Creed Origins, to create more realistic and fluid animations for its characters.

EA Sports has also adopted motion matching in its sports games, such as FIFA and Madden NFL, to create more realistic player movements and enhance the overall gaming experience.

Moreover, motion matching has the potential to change the way game developers approach character animation, allowing for more creative freedom and more dynamic and engaging game worlds. As motion matching continues to evolve, it is likely to become an even more integral part of the game development process.

In this report, we will explore the innovations in motion matching technology and its impact on the gaming industry. We will examine the benefits of motion matching for game development and discuss the challenges and limitations of this technology. Additionally, we will analyze case studies of games that have successfully implemented motion matching and evaluate the potential for future advancements in this field.

Chapter I
General Introduction

Chapter 1: General Context

1. Introduction:

In this chapter, we will provide an overview of the host company and the project we worked on during the internship. Firstly, we will introduce the company and describe the general environment in which it operates. Next, we will outline the project's objectives and goals, followed by a discussion of the primary issue that the project aimed to address. Additionally, we will detail the methodology used to tackle the problem and highlight our approach to managing the project's progress. Overall, this chapter will provide a comprehensive understanding of the project and the strategies employed to achieve its goals.

2. Hosting Company:

Lanterns Studio is a Tunisian gaming company that was established in 2019 and is headquartered in Tunis. The company specializes in delivering advanced technology solutions that help accelerate its clients' digital transformation, as well as providing extended reality applications and motion capture services. With a team of highly skilled and knowledgeable employees and strategic partnerships with industry leaders, Lanterns Studio is dedicated to delivering cutting-edge solutions that meet the evolving needs of its clients.

Figure 1: Lanterns Studios logo

Lanterns Studio provides a range of innovative solutions across diverse industries, including information technology, computer graphics, branding, and media content. The company offers a comprehensive suite of services, from mobile application development to augmented and virtual reality experiences, and from PC to console video games. With a focus on creativity and technology, Lanterns Studio is committed to providing its clients with

solutions that are tailored to their unique needs and goals, and that push the boundaries of what's possible in the digital landscape.

Company:	Lanterns
Creation Date:	2019
Category:	Startup
Sector:	Gaming
Address:	Rue de energy, Tunis
City:	Tunis
Zip Code:	2035
Country:	Tunisia
Phone:	+216 23586707
Website:	https://lanterns-studios.com/
E-mail:	info@lanterns-studios.com

3. Project presentation:

The purpose of this project is the creation of a crowd system with motion matching which is a technology used in game development to simulate large groups of characters or entities moving in a realistic and coordinated manner. The system works by creating a group of "agents" that represent individual characters or entities within the crowd. Each agent is given its own set of rules and behaviors, such as moving towards a certain destination or avoiding obstacles.

Motion matching is a technique used to create more realistic animations for these agents. It involves capturing a wide range of motion data and then using an algorithm to match the desired movement to the most appropriate motion clip. This allows for more natural and fluid movements that are responsive to the environment and other agents within the crowd.

Combined, these two technologies allow for the creation of large, dynamic crowds that can move and behave realistically in response to the game's environment and events. Crowd

systems with motion matching are particularly useful in games that require realistic and dynamic crowds, such as sports games or open-world games with large cities or events.

4. Study of the existing & Research:

In the past, creating realistic crowd systems in games was a difficult task that required significant development resources. Before the advent of motion matching, game developers typically used pre-animated crowd animations or simple rule-based systems to simulate crowds. These approaches were limited in their ability to create truly dynamic and realistic crowds, as the animations were often repetitive and lacked responsiveness to the game's environment and events. Rule-based systems were also limited in their ability to create natural movement, as they relied on simple rules that did not account for complex behaviors or interactions. As a result, developers often had to resort to various tricks and techniques, such as duplicating models and animations or reducing the number of characters in a crowd, to achieve the desired effect. However, these solutions were often unsatisfactory, and the resulting crowds were often static and lacked the realism and dynamism that modern games require.

4.1 Case of study:

To undertake this ambitious project successfully, it is crucial to comprehend its specific requirements and intricacies. Thus, following the establishment of the project's overarching context, this section is dedicated to examining comparable endeavors that share similar visions.

4.1.1 Animation Clips:

movement and actions in characters or objects within a game or animation. The classification of animation clips encompasses three primary types: cycles, actions, and transitions. The categorization is not solely based on the nature of the action itself but also takes into account the initiation and conclusion of the animation.

Cycles:

They refer to animations that possess the ability to repeat endlessly, exemplified by the continuous cycling of a character's walking or running motion.

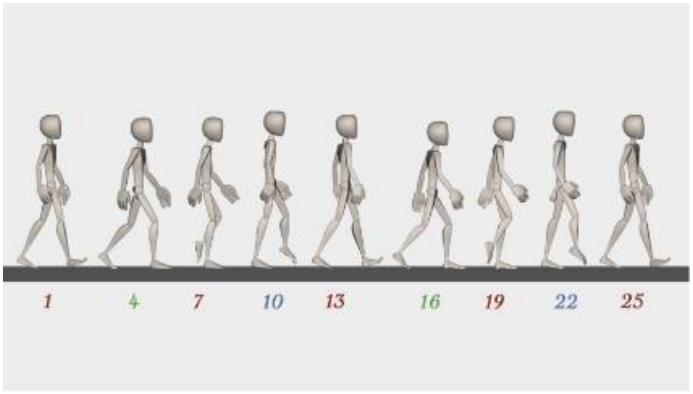

Figure 2: walk cycle animation example.

Actions:

They refer to animations that have a distinct initiation and conclusion, such as the attack animation of a character.

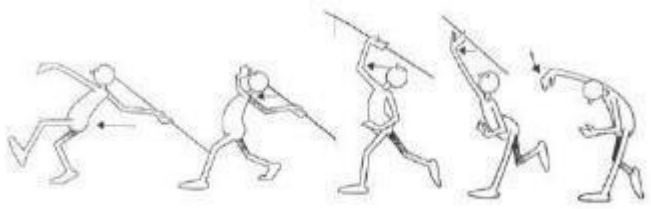

Figure 3: Attack action example.

Transitions:

Transitional animations, also known as blend animations, bridge the gap between two distinct animations, facilitating a smooth and seamless transition from one to another. For example, these animations are commonly employed when a character transitions from their idle animation to their running animation.

Two primary methods exist for generating animation clips:

- **Keyframe Animation Clips:**

The animation is defined through a collection of clips that outline a sequence of keyframes, with each keyframe denoting a specific moment and location of an object or character. Subsequently, the computer interpolates the intermediate frames, bridging the gaps between keyframes to produce a seamless animation.

- **Motion Capture Animation Clips:**

These motion clips utilize data obtained from real-life motion capture sessions to animate characters and objects within the game. This approach entails capturing the movements of actors outfitted with motion capture suits or markers and subsequently transferring that data onto a digital character model. The resultant animation clip achieves an exceptional level of realism, accurately capturing the intricate details and subtleties inherent in human motion.

4.2 Skeletal rig:

In the domain of computer graphics and animation, it is common practice to employ a skeletal framework composed of interconnected joints, as illustrated in Figure 4. These joints serve as counterparts to the articulated components of the human body. Each vertex of the mesh is connected to one or more joints, and any motion performed by a joint, leads to a corresponding displacement of the associated vertices.

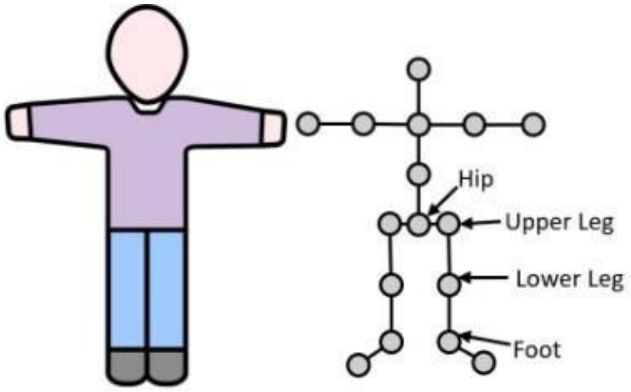

Figure 4: Representation of a humanoid skeleton

Note: Spheres represents joints.

4.3 State Machines:

State machines are a variant of behavior nets, encompassing a collection of states or actions, transitions, and an entry point. The states within the state machine denote specific activities, while the transitions outline the process of transitioning between these activities. Traffic lights serve as an illustrative instance of a state machine, where circles represent states, arrows denote transitions, and specific conditions trigger these transitions. In the realm of computer animation, states and transitions can be interpreted as animation clips and gameplay conditions, respectively.

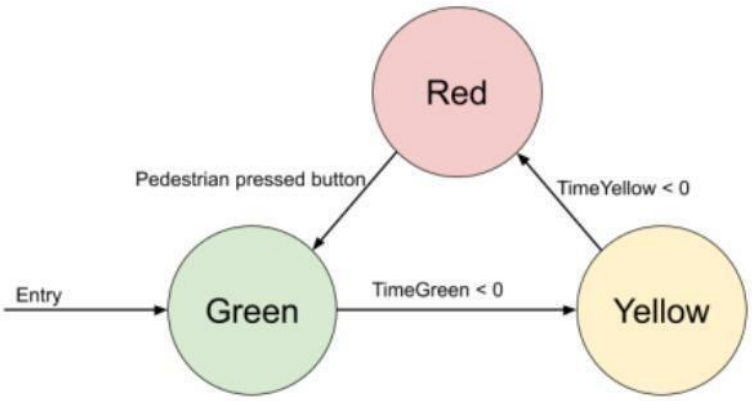

Figure 5: Traffic light state machine

4.4 Problematics:

The creation of a perfect animation system for NPCs where the solution provides with as much detailed movements as possible as well as visually attractive was truly difficult to maintain and provide. It is difficult to create such system with **state machines** alone, it implicates adding multiple animation record and using too much **blend trees** at the same time with a single character animation (Which uses an amount of memory and calculations needed to achieve an acceptable result which is incredibly large) and results with non-organized projects and sometimes messy animation structures, that's what traditional animation technique is handling.

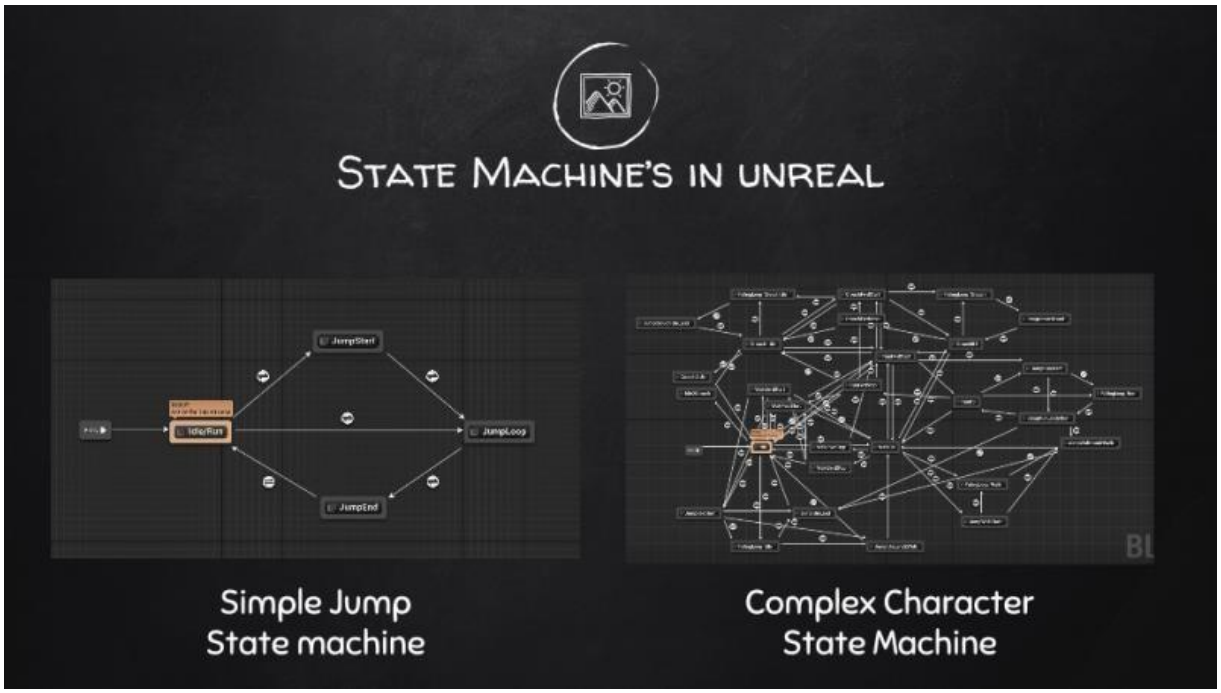

Figure 6: Complex state machines

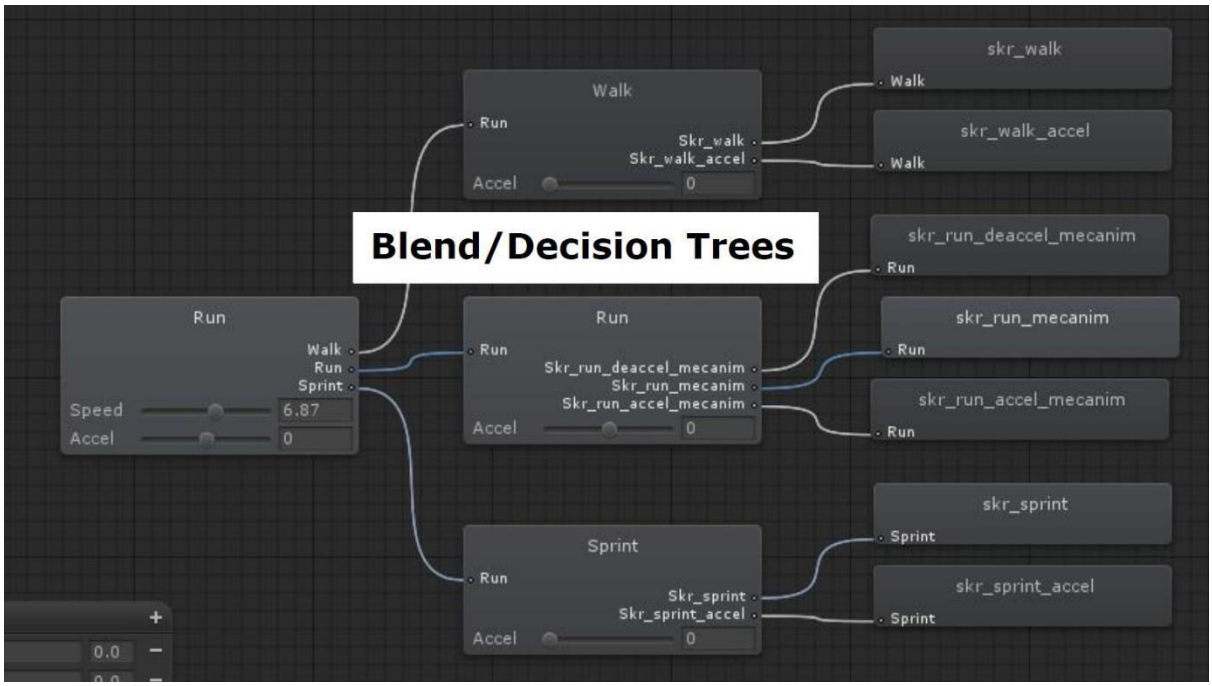

Figure 7: Complex Blend trees

So, we need to find a solution that include a way to deal with loops and animation transitions in a uniform way to create a precise gameplay and believable animations.

4.5 Solution presentation:

To develop a crowd system with motion matching for our game, we conducted a thorough study of the existing technologies and techniques in the field. We analyzed various crowd simulation and motion capture techniques used in popular games and studied the strengths and limitations of each approach. We also reviewed academic literature and research papers to gain a deeper understanding of the underlying principles and algorithms involved in crowd simulation and motion matching. Through this comprehensive study, we were able to identify the most effective techniques and technologies for our project, and we incorporated them into the development process to ensure the creation of a realistic and dynamic crowd system. Motion matching is a revolutionary technology in game development that solves many of the problems associated with traditional animation techniques. One of the primary issues with traditional animation is that it can be time-consuming and expensive to create large numbers of unique animations for every possible scenario. **Motion matching** solves this problem by using a database of motion data and an algorithm to seamlessly blend motion data together to create smooth and natural movements that are responsive to the game's environment and events. Additionally, traditional animation techniques can result in animations that look stiff and

unnatural when used in complex situations such as crowds. Motion matching addresses this issue by allowing for more realistic and dynamic animations that can adapt to changing situations in real-time. Overall, motion matching allows game developers to create more immersive and realistic worlds that are responsive to player actions, while also reducing the development time and costs associated with traditional animation techniques. The work process of this technology is presented in simple steps down below:

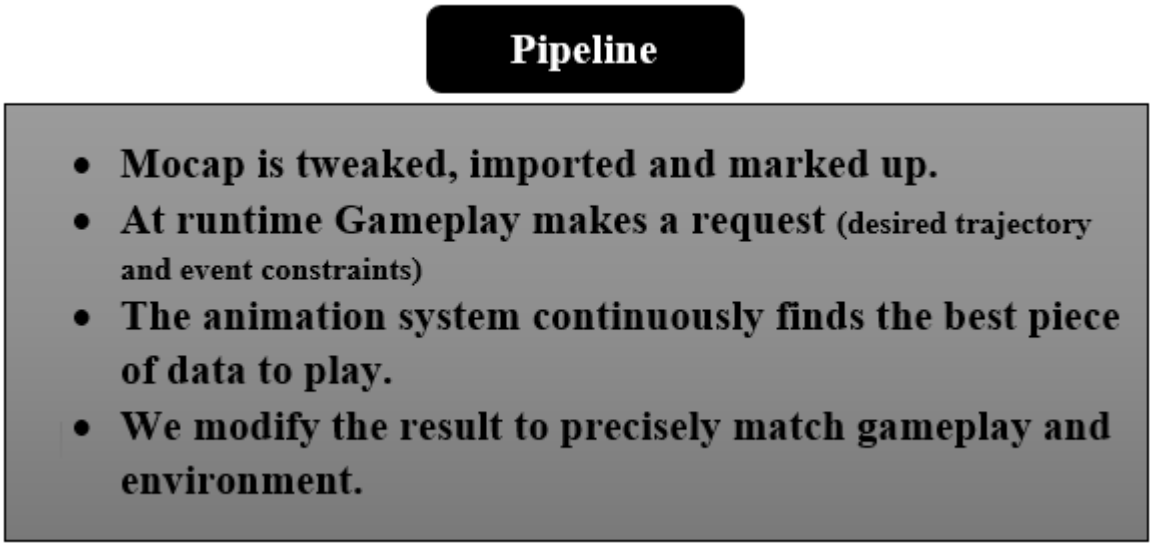

Figure 8: Motion Matching Pipeline process

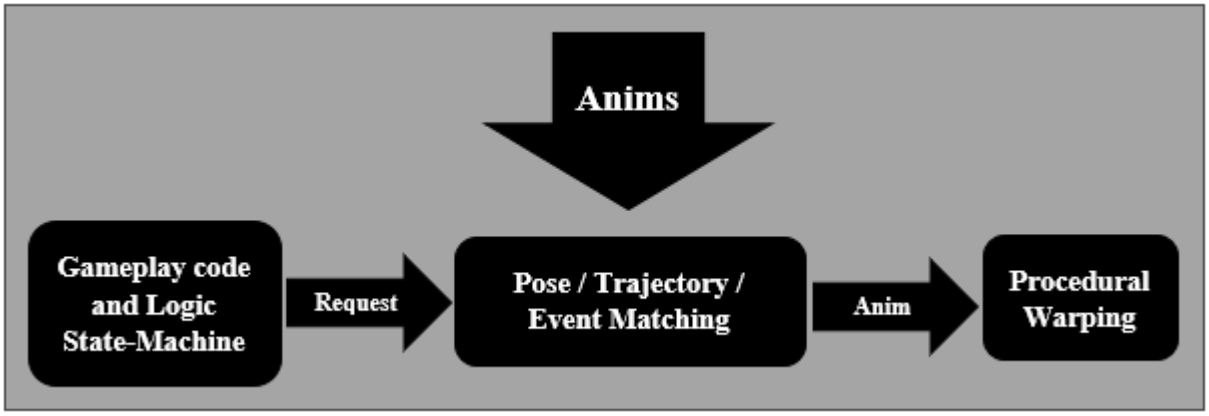

Figure 9: Motion Matching Pipeline in a simple Diagram

This technology focusses on the manipulation of different animations throw the **motion graph** and that, throw an unstructured list of animations, can give us the ability to make chose the key frames needed to perfectionate our animations and limit the use of excessive animations tracks throw playing specific animation key frames throw the motion matching Database. More details are provided in the scientific paper named "**Motion Graph**" written by Lucas Kovar,

Mickael Gleicher and Frédéric Pighin. It explains more details about the creation of realistic and controllable motions.

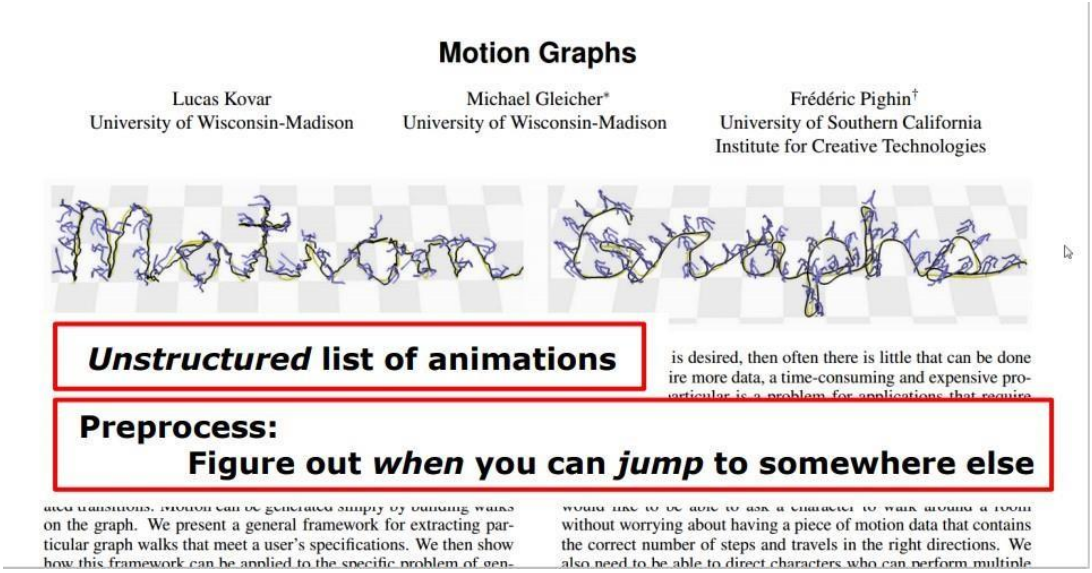

Figure 10: Motion Graph

The transition between animations is done after a smart selection of key frames depending on the position of the 3D character in the scene and when reaching a transition point, decide where to go from a list of possibilities guided by a path on the graph and gives us our desired animation.

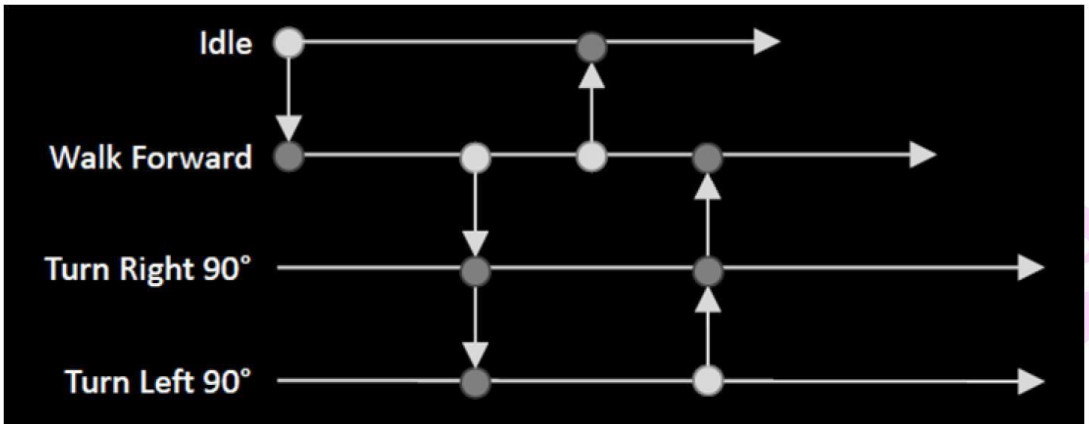

Figure 11: Example Graph of Transition Points

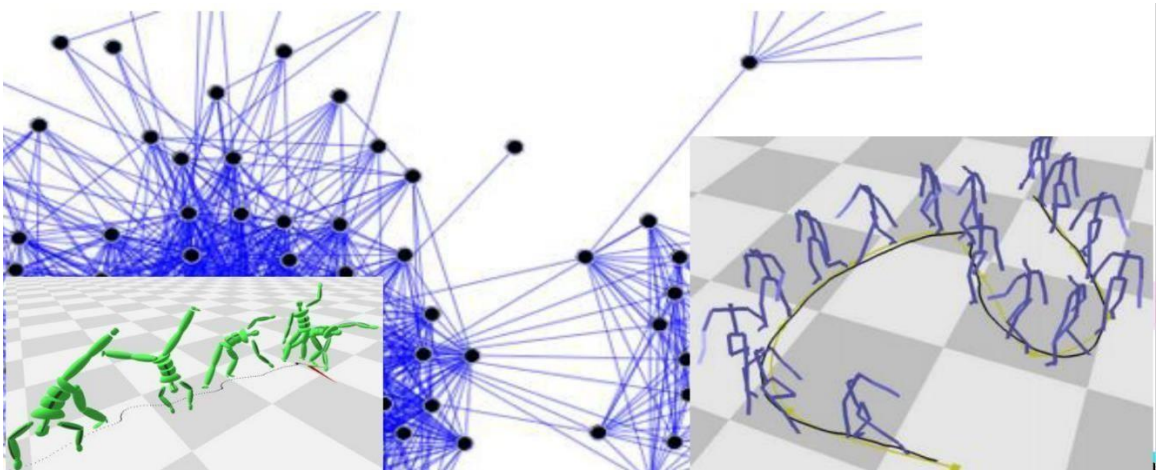

Figure 12: Example Graph of animation paths

The animations are played after calculations and trajectory predictions during runtime. The process of achieving this involves gathering samples over a specific time period, without considering the character's current momentum. Let's consider a scenario where the character maintains a constant linear velocity of 1 unit per sample. Starting at position 5 and moving at a speed of 10 units per sample, we can examine the resulting trajectory over 4 samples:

Sample 1:

Position = Starting position + Velocity * Time = 5 + 0 * 1 = 5

... the resulting trajectory over 4 samples would be 5, 15, 25 and 35.

To account for momentum, the sampling process will involve the use of the linear velocity and a smoothing function like an exponential decay function for acceleration and deceleration.

Inverse transforming trajectories:

To produce the desired character path, the trajectory is initially formulated in world space. However, it must be transformed back into the animation or global space, which denotes the character's initial position and orientation. This transformation is achieved by multiplying each point of the trajectory by the inverse of the animation transform.

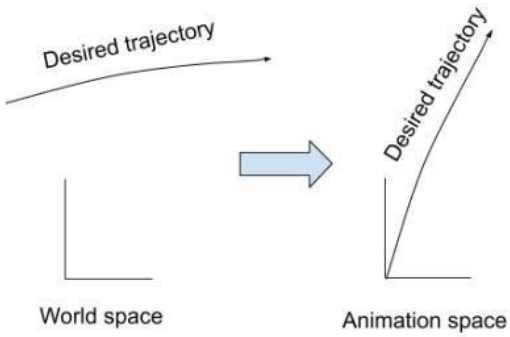

Figure 13: Transforming the desired trajectory.

Trajectory curves matching:

When evaluating the correspondence between a desired trajectory and the set of previously stored trajectories, the procedure entails computing the distances between each sample point of the desired trajectory and its corresponding point on the cached trajectories. These distances are then aggregated to establish a final cost, serving as a measure of the level of alignment between the desired trajectory and the cached ones. A lower cost value indicates a closer resemblance between the desired trajectory and the cached trajectories. The trajectory with the minimum cost is regarded as the optimal match, exhibiting a high degree of similarity to the desired trajectory.

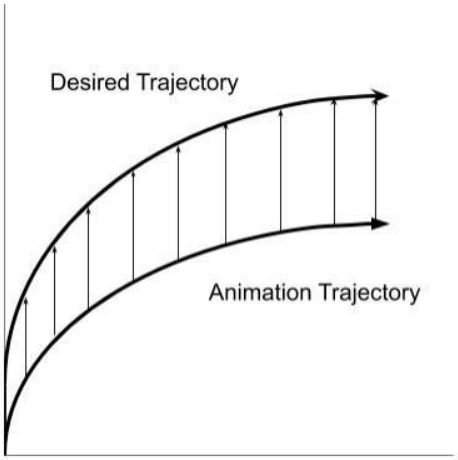

Figure 14: Matching trajectories.

The selection of an animation pose that closely matches the desired trajectory is accomplished through trajectory-matching. However, this approach may not always yield realistic human locomotion. In order to achieve top-notch animation quality, the emphasis shifts towards selecting animations with a comparable pose to the one currently being played, prioritizing the overall excellence of the animation over immediate responsiveness to user input.

Pose matching:

A motion library or database is created, consisting of a collection of reference poses that represent various actions or movements available to the character. During runtime, the current pose of the character or object in motion is captured and compared against the reference poses stored in the motion library, aiming to identify the most suitable match. To optimize performance and achieve desired outcomes, it is crucial to limit the pose matching process to the relevant joints specific to a particular type of movement (e.g., feet for biped locomotion, as suggested by Michael Buttner). Each joint's position and orientation are defined within its local frame, preserving the information in relation to its parent as local space.

Figure 15: Feet joints

Pose comparison and Forward kinematics:

Typically, bone transformations are stored in local space, indicating their position relative to their parent bone. However, relying solely on this information for matching purposes is inadequate since bones in identical positions can exhibit different transformations influenced by their parent's transformation. To overcome this limitation, it is essential to represent bone transformations in global space. This is achieved through the accumulation of transformations from child bones to their respective parents, extending up to the root bone, commonly referred to as Forward Kinematics. By adopting this approach, a comprehensive understanding of bone transformations in relation to the entire skeleton can be obtained.

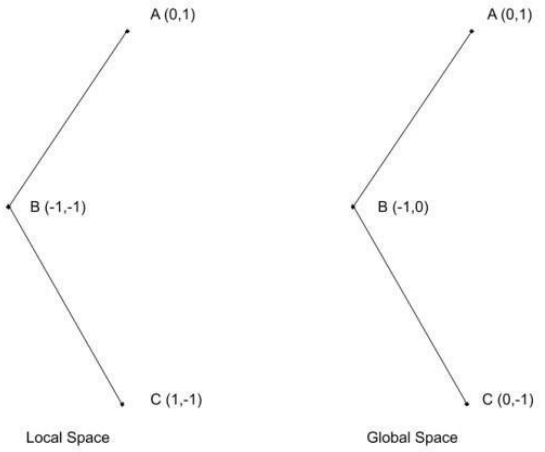

Figure 16: Feet joints between local and Global space

At each animation frame, the bone transformations will be performed and subsequently stored in the animation Metadata container. This container, which represents a subset of animations, is stored in a compressed format optimized for cache efficiency and includes extracted trajectory data. Within the Compute-Cost function, the matching process entails evaluating the disparity between the present global position of the bone and all the previously cached positions. This evaluation determines the level of alignment between the current bone and the stored positions. The resulting disparity is subsequently combined with the cost derived from trajectory matching calculations.

The preliminary findings demonstrate notable enhancements, manifesting in more realistic movement that aligns with the intended trajectory. However, a slight jittering effect is noticeable. This phenomenon arises from the animation system's disregard for bone velocity, resulting in selected animation poses that may unexpectedly oppose the desired direction, despite being the most cost-effective choices in terms of position and future trajectory. To tackle this concern, it is imperative to incorporate bone velocity within the animation system as a necessary consideration.

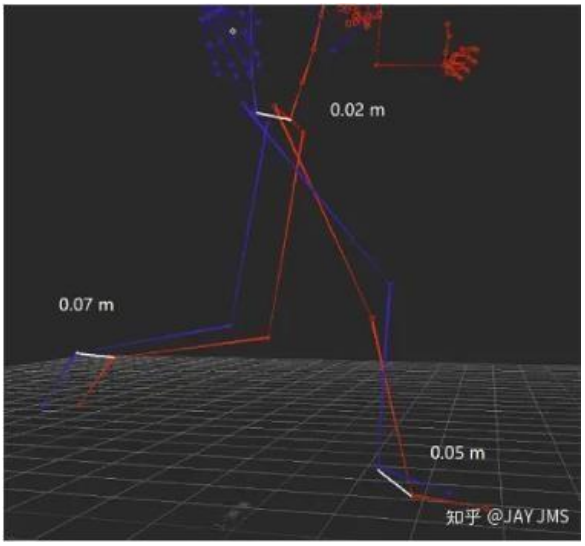

Figure 17: Pose matching and comparison.

Velocity matching:

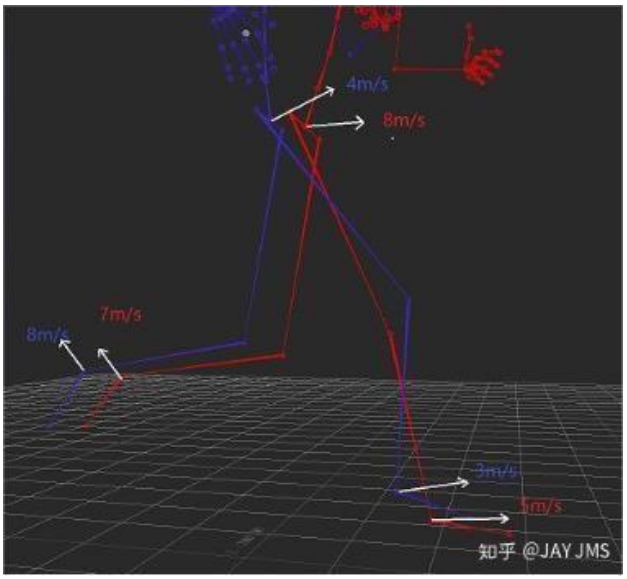

Figure 18: Velocity matching and comparison.

Motion Warping:

Motion warping is a fundamental technique in computer animation that plays a vital role in creating realistic and visually appealing character movements. It involves the manipulation of existing motion data to achieve desired motion variations while preserving the overall naturalness and coherence of the animation. This research paper explores various motion warping techniques used in character animation and their applications in the gaming and film industries. The objective is to analyze the advancements, challenges, and potential future

directions in motion warping research, with a focus on enhancing realism, efficiency, and artistic control in character animation.

Character animation requires the generation of lifelike and believable movements, which can be a challenging task for animators. Motion warping techniques offer a promising solution by allowing animators to modify existing motion data to meet specific artistic requirements. By manipulating motion at various levels of granularity, such as joint positions, orientations, and timing, motion warping enables the creation of unique character behaviors, expressive gestures, and dynamic interactions. This research investigates several motion warping techniques employed in character animation. These include traditional approaches such as time warping, where temporal aspects of motion are altered, and space warping, where spatial transformations are applied to the motion data. Additionally, we explore advanced methods such as pose warping, which focuses on modifying joint angles to achieve specific poses or exaggerate movements, and procedural warping, which leverages mathematical functions or procedural algorithms to generate complex motion variations.

Conclusion:

Motion matching techniques have demonstrated remarkable advantages in real-time animation synthesis. By matching the input motion with a vast database of motion samples, motion matching achieves seamless and contextually appropriate animations. It excels in capturing the intricate nuances of character movements and seamlessly transitioning between different motion states. The hierarchical and adaptive approaches further enhance the scalability and adaptability of motion matching, enabling the synthesis of high-quality animations in dynamic and interactive scenarios.

5. Project Objective:

Our project endeavors, for making professional and more advanced games, to:

- Implement the Motion Matching system solution in a player and an NPC character.
- Implement the Motion Warping technology solution in both player and NPC character.

- Test the NPC with both the motion matching and the motion warping technologies in a crowd simulation.

6. Working Methodology:

In order to ensure the reliability of software, it is crucial to establish a well-defined work methodology and a comprehensive monitoring process prior to initiating a computer project. This approach outlines a systematic procedure aimed at formalizing the early stages of system development, with the goal of enhancing the alignment between the development process and the specific requirements of the client.

6.1 Methodology choice:

In software engineering, various development methodologies are employed to guide the development process. These methodologies, such as Waterfall, Agile, and Scrum, differ in their approach and principles. The Waterfall methodology follows a linear sequential approach, where each phase is completed before moving on to the next. However, this rigid structure can hinder adaptability to changing requirements. On the other hand, Agile methodologies, including Scrum, emphasize flexibility, collaboration, and iterative development. Scrum stands out as a highly effective methodology for complex projects. It promotes self-organizing teams, regular feedback loops, and short development cycles known as sprints. This iterative nature enables continuous improvement, increased stakeholder involvement, and early delivery of valuable software increments. By embracing empirical control and allowing for incremental and adaptive development, Scrum minimizes risks, enhances transparency, and ensures a higher degree of customer satisfaction. Therefore, in the realm of software engineering, the Scrum methodology emerges as a preferred approach due to its ability to address the challenges of modern development projects.

6.2 Scrum presentation:

Agile Scrum is an iterative and incremental project management framework that promotes flexibility, collaboration, and continuous improvement in software development. It emerged as a response to the challenges faced by traditional project management approaches, which often struggled to adapt to rapidly changing requirements and lacked effective communication and transparency. At its core, Agile Scrum follows the principles of the Agile Manifesto, emphasizing individuals and interactions, working software, customer collaboration, and

responding to change. It embraces an iterative and time-boxed approach, dividing the project into small, manageable units called sprints. Each sprint typically lasts two to four weeks and focuses on delivering a tangible and potentially shippable product increment.

The Scrum framework consists of three primary roles: the Product Owner, the Scrum Master, and the Development Team.

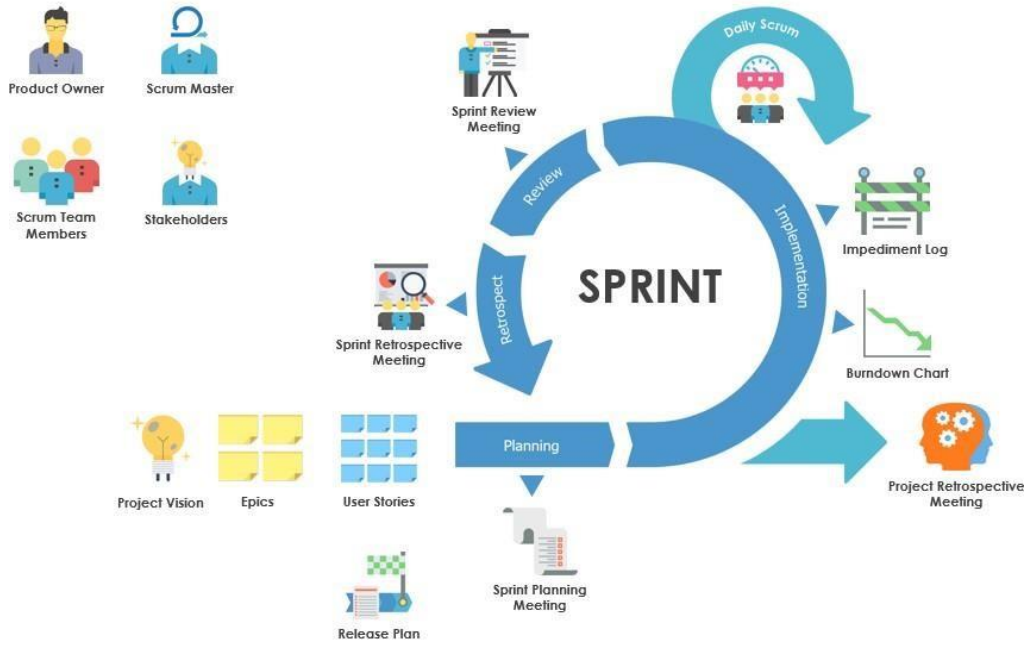

Figure 19: Scrum Life Cycle

The Product Owner represents the stakeholders and ensures that the product backlog, a prioritized list of features and requirements, is well-defined and reflects the customer's needs. The Scrum Master facilitates the Scrum process, removes any impediments that hinder the team's progress, and fosters a productive and collaborative environment. The Development Team, consisting of cross-functional members, is responsible for delivering the product increment at the end of each sprint. Key practices within Agile Scrum include daily stand-up meetings, where team members share progress, discuss challenges, and plan the day's work; sprint planning, where the team selects the backlog items to be worked on in the upcoming sprint; and sprint review and retrospective, where the team demonstrates the completed work to stakeholders and reflects on the sprint to identify areas for improvement.

One of the significant benefits of Agile Scrum is its emphasis on customer collaboration and early and frequent delivery of working software. This iterative approach allows for continuous feedback, enabling the team to adapt and adjust based on customer needs and

changing requirements. Additionally, the transparency and regular communication within Scrum foster a shared understanding and alignment among team members and stakeholders.

6.3 Scrum Roles:

SCRUM encompasses three primary roles, which will be detailed in the table presented subsequently:

- **Product Owner:**

The Product Owner serves as the designated representative of the client within a Scrum project. They act as the primary point of contact for the Scrum Master and team members, assuming responsibility for articulating the product's requirements and crafting the specifications. Collaborating with functional experts, the Product Owner may receive assistance in defining the specifications. Additionally, they play a crucial role in establishing and prioritizing user stories for each sprint.

- **Scrum Master:**

He fulfills the role of overseeing the method's execution and ensuring its objectives are met. While not assuming the position of a Project Leader, he assumes the responsibility of eliminating any potential hurdles that could impede the team's progress and hinder the project's advancement throughout the multiple sprints.

- **Development Team:**

These individuals are responsible for attaining the sprint goals and delivering a functional product by the end of the iteration. They encompass a range of roles, including developers, architects, and functional testers. The team maintains direct communication with the client.

7. Requirements definition:

7.1 Actors' identification:

Within our project, we have recognized a specific set of tasks linked to an individual user known as the developer or animator, who possesses the ability to:

- **Integrate the motion symphony system in any project.**
- **Add motion matching to the main player character in any level.**
- **Add motion matching to any created AI character within any level.**

- **Add any animation into any motion dataset or existing ones.**
- **Tag any animation sequence or keyframes used for the motion data set.**

7.2 Functional needs:

The system's functional requirements encompass its core functionality, which primarily originates from developers who act as users. The objective of this application is to enable users to:

1. **Integrate the motion matching symphony system in any unreal project.**
2. **Add the motion matching trajectory functions in the player character logic using both the player's blueprint file and the animation Blueprint file.**
3. **Add the motion matching trajectory functions in the AI character logic using both the player's blueprint file and the animation Blueprint file.**
4. **Add any animation in the motion dataset or modifying it to the similar animations in the dataset applied to the selected character in the level.**
5. **Tag selected animation frames within every single animation in the dataset and chose from either "Do_not_use"," Cost_multiplier" or "Tag_trait" which is a custom tag created to specify a specific use to play the animation frames selected.**

7.3 Nonfunctional needs:

In addition to the fundamental prerequisites, the project must also fulfill specific criteria related to performance and design specifications, encompassing:

Performance:

The foremost requirement for the system is its efficiency, which entails meeting all user requirements optimally through its functionalities.

Maintainability:

As an integral component of this project, it is imperative that the system possesses the quality of extensibility.

Understandability:

- Design, architecture, and code that are straightforward and accessible for comprehension and learning.

- The capacity for other developers and animators to implement changes swiftly and efficiently.

Efficiency:

- Satisfactory promptness in addressing user interactions. ○ Error Recovery: The duration required to restore or redo work following an error.
- Supporting the unreal project versions from unreal engine 4 to 5.

8. Project planning:

Training:

Upon the commencement of the internship, we embarked on an intensive training program focused on the utilization of the Unreal Engine 4 game development engine and the C++ language. This training was facilitated through tutorials available on the official website "learn.unrealengine.com" as well as YouTube resources.

Conception/Design:

Throughout the duration of the internship, we contemplated undertaking the design phase. As a result, we initiated an exploration of the UML language, a fundamental design language, utilizing the "Draw.io" software as our tool of choice.

Development:

The development phase commenced with a slight delay, as previously mentioned in the self-study section. During this phase, we employed the Unreal Engine 5 to seamlessly integrate and refine the new system, incorporating all its intricate aspects and adjustments. This comprehensive implementation aimed to optimize its utility for future production purposes, facilitating the work of fellow developers and animators.

Realization of the report:

We commenced the report writing process in early May and successfully completed it within the initial days of June.

Appliance of the system:

We dedicated a period of over two and a half months to the implementation and advancement of our modified system.

9. Conclusion:

In this chapter, we have provided a concise overview of the project at hand, outlining the identified problem and proposing a solution to address the current situation.

Chapter II
Conceptual Study

Chapter II: Conceptual Study

1.Introduction:

The most intriguing phase of the development process is the design phase, where we had the opportunity to define the system's structure and behavior, enabling us to address the various challenges encountered during development. Initially, we present the architectural design of our solution, followed by the depiction of several diagrams including global and specific Use case diagrams, Activity diagram, Class diagram, and Sequence diagram. These diagrams provide a comprehensive overview and detailed visualization of the system, aiding in the decision-making process and the identification of effective solutions for development hurdles.

2.Modeling and Design Methodology:

Software development methodologies are approaches that guide the process of creating software applications. These methodologies provide structure, principles, and guidelines for managing the various stages of software development. One important aspect of software development is the use of modeling languages, which facilitate the representation and communication of software designs and concepts. Modeling languages, such as Unified Modeling Language (UML) and Business Process Model and Notation (BPMN).

It enables software developers to visually depict system architectures, workflows, data structures, and interactions between components. UML, for instance, offers a standardized set of diagrams, including class diagrams, sequence diagrams, and use case diagrams, which aid in designing and documenting software systems. Modeling languages play a crucial role in enhancing collaboration, reducing ambiguity, and fostering a shared understanding among stakeholders throughout the software development lifecycle. By leveraging modeling languages in conjunction with suitable software development methodologies, teams can effectively plan, design, and implement software solutions while maintaining clarity and consistency in their development processes. We opted for UML as the modeling language for constructing our system.

Figure 20: UML logo

2.1 Use case Diagram: