REAL TIME CROWD FLOW

Real time interactive simulations to inform decisions in wayfinding design

E.LI,
UNSW, Sydney, Australia
Flim.edenli@gmail.com

Abstract. Real-time simulations enable the freedom of letting the computer create, whilst having the user control the input data. This creates an emphasis on real-time technology as it solves the problem of having the designer devote time to execute multiple roles of animating, calculating and finalizing. This paper explores the potential of such a system specifically within the context of crowd simulations. Dynamics, accuracy and adaptability are key aspects of crowds, and as a resultant, modern approaches are often very computing heavy and tedious. The research aims to create a system that executes a crowd simulation via procedural means within the game engine, Unreal Engine 4, so that it responds to user inputs, such as adding doorways in real time. This investigation ultimately challenges the notion that simulations is just a tool for post-design, but rather, it should be considered throughout the whole production. It encourages new design ideas that are validated via a trial and error habit due to the nature of real time possibilities. Further exploration of real time aspects could lead to newer and faster design methods and systems.

Key words: *Real-Time, Procedural Simulation, Crowd Simulation, Interactive, Dynamic*

1. Introduction: (Research context and motivations)

A truly dynamic simulation, that is in real time (in calculation and rendering) will encourage faster workflows that maintain a high level of detail at no expense of extra effort. It enables the simulation to be run parallel to the design process unlike its current method of restarting at every new iteration. Meaning that at every client request, changes can be instantaneous whilst maintaining a polished look. Just like in video games, there is no practicality, if every time the player decides to do an interaction, it causes the need to re-load up to the point of change. Hence, architects should not have to experience the shortage of a more efficient workflow.

Crowd simulation is a digital representation of people's movement within spaces, often used to prove whether the efficiency of spatial design. It is a collection of humanoids or 'agents' that have specific characteristics assigned to them to best represent human movement. Human behavior is often unpredictable and highly prone to change relative to their contextual surroundings. Having a *real time* simulation is perfect application as the user can constantly test different designs amongst a constantly adapting crowd.

Executed inside Unreal Engine 4, the project is unlike generic simulation programs as game engines prioritize playability/ smoothness over realism, which implies the results to not be taken seriously. But this paper considers those impressions, taking advantage of this 'con' and help realize that game engines are powerful to perform tasks kin to those of 'specialized' programs.

Throughout the paper, it should be noted that the objective of the research is not to solve a problem, but rather to improve on a skill/ task that is already at a high level of production.

Ultimately the journey in developing a real time crowd simulation can be considered just a small peek into the potential of developing real time in the industry.

2. Research aims and objectives

The aim of this research is to investigate procedural simulation techniques to create a dynamic virtual environment that can model scenarios in real-time as well as be interacted by the user. The whole research could be broken down into multiple streams of research on its own, so in order to keep the project succinct, project aims, and goals must be established as guidelines for progress and success.

The real time aspect is the most important element of this research paper. Many crowd simulation programs already exist; they are all a high caliber however; they all lack the essence of real time. In order to set a good standard of real time, the agents and the scene must be instantaneous in the reaction to

the user input. Not only that, but the system must not experience any decrease in render time or any buffering when new elements are introduced.
The next most important aspect is the crowd itself. Its ability to perform human like movements, and to be able to perceive items based off human instincts like sight. Finally, the crowd requires the ability to identify certain elements of the building, this is key in constructing a crowd that is connected to its own surrounding context, which is important for realism.
The final step, if time allows, is to test the system amongst multiple users, and observe how real-time affects their workflow, design decisions and whether they incorporate a trial and error like mindset. There is also curiosity on their mindset on the system, do they trust it? Do they think its applicable in real life? Is it easy to use?

3. Research Question(s)

In What ways can real-time interactive simulations be developed for use in wayfinding design decisions?
This research is both an investigation of the development of a system and the impacts of it on workflow. The biggest question is the notion of real time and its necessity in the progression of programs in the architectural field.
How can procedural simulation techniques enhance interactivity in agent-based virtual design environments?
Why are architectural programs still so stagnant when computing power are increasing at an incredible rate?
What are the effects of using a game engine and classifying it as a simulation tool? Is it reliable? Is it valid? Is it accurate?

4. Methodology

Methodology is the ‘cycle’ of having a problem or question presented, analyzed, followed by theorized solution, execution, interpretation and then repeated (Gabel, 2017). The question at hand is how to create a real time interactive simulation that will help realize the full potential of instantaneous programs and its assist in the architecture field. The execution is in the development of a real time interactive crowd simulation which was proceeded in UnReal Engine 4 (UE4) with the assist and BluePrint(C++). The interpretation is qualitative, it is not a question of ‘does it’ work but rather ‘can it’ work as the full potential of this system requires a much bigger investment of resources available.
Throughout the research, there will be multiple iterations, each one focusing on a specific element to create a successful system.

Iteration 1 – Agent pathing and obstacle avoidance
Iteration 2 – Dynamic scene and interaction

Iteration 3 – Advanced AI perception and reaction
Iteration 4 – Real time texturing and rendering
Iteration 5 – User testing

In more detail, models and contexts will be developed in Rhino3D, a NURBS modeling program that is then exported into UnReal. Inside UE4, the models will be imported into its scene where all the items are centralized into one place. Then BluePrint will be used to apply the 'personalities' or attributes to the scene. The question at hand is how to implement certain instructions and quantifying them into components and scripts that is understandable by the game engine.

5. Background Research/Literature review

Modern culture and technological advances have driven our society into a people who desire results instantaneously, whilst also at relative accuracy and quality. As our capabilities to perform quality real time products increases, so does its potential to be applied to architecture.

5.1. OPTIMIZATIONS

5.1.1 Nivida GPU

Nivida, a leading company in graphical development recently released a new direction of graphic cards that they name RTX cards. In these cards, they commercialized a feature named 'Ray Tracing'. This feature has been around for many years and is considered the 'holy grail' of rendering, as it accurately calculates reflection and lights. Up until now, it has always been a restriction because of computing power (Caulfield. 2018) but, it is important to note Nivida's push for this feature their understanding of the potential of attacking such a computing heavy task in real time, as they recognize it as the future.

5.1.2 Industry Standards

In Architecture, the bridging of the digital to creating a seemingly realistic representation is key, in its ability to prove validity and vision. In the context of crowd simulations, for them to be viable, they must be able to render convincing scenes whilst also maintaining the integrity of the behavior of the agents (Lozano. 2008. Pg1). Crowd simulation's biggest challenge is the large computing resources needs to be devoted to creating a finessed product due to the sheer amount of data and processing needed to represent each agent. Lozano in his article tries to combat this lack with hardware, linking up multiple computers in a server like manner, to produce computing power in a net amount. Whilst this system deemed effective, it is was a very costly experiment that also gave them no advantage besides speed. This paper focuses on the software and the switch to a game engine instead. As

explored before with the addition of Nivida support in Unreal Engine, it allows optimized instantaneous renders and simulations all at the expense of a single device.

5.1.3 Unreal Engine

The choice of using Unreal Engine to host the real time simulation was quite an easy decision that was both logical and based off background research. Firstly, as explored in 5.1.1, Nividia has been developing with Unreal for many years and the software takes advantage of any new release Nividia has. There is access to the PhysX (Nividia) system which a physics engine that is optimized to any Nivida GPU, and it helps aid any form of graphical processing that includes Artificial Intelligence, Animation, Scripting and other forms of processes. The introduction of live Ray Tracing, though not used in the duration of this paper, proves that it is a future proofed product, that is only prone for constant development. The default AI/ agent system is quite simple, but their ways to introduce complicated behaviors, only at the expense and knowledge of the developer. Unreal also released a new feature called Hot Pixel Streaming, which essentially is hosting the whole program on a server, allowing clients to instantly see the project anywhere with an internet connection. This creates even more emphasis on the notion of real time as now, there is instant access anywhere, hence showing the future proofing and advantages of Unreal Engine 4.

5.2 PHILOSOPHY AND METHODOLOGY

5.2.1 Simplification of complexity, to scripts

There are three main segments that define the term ‘procedural’. Algorithmic mainframe (where rules and definitions are created), variable input (where the mainframe receives data), and output (product that is created when the variables run through the definitions), which they work together to create a loop. Relevant to architecture, designers can finally create scenarios that can be updated, nearly instantaneously when a client desires a variation.

Zeeshan Bhatti in his report Procedural Animations of 3D Humanoid Characters Using Trigonometric Expressions uses procedural animation to explore the unique application of procedural animation into the observation and creation of human movements in a digital space. This report has no direct mentions of architecture, but the philosophy and methods of their research still holds as Bhatti explores the familiar idea of ‘real time’ and realism. “Key frame being a traditional approach takes a huge time to render a realistic looking animation” (Bhatti 2016 p.2) expresses the need to use newer methods to rpelace tedious tasks that requires time and resources. He takes a mathematical approach, resulting in his team to firstly deconstruct every human movement, and translating them into formulas. It represents a logical and editable medium that can change the ‘algorithmic mainframe’, in

only a few lines of code. This is extremely efficient for people who need to constantly refine their product, to achieve the optimal outcome. However, with specialization, comes exclusiveness of the system, meaning only skilled developers can understand the mainframe, whilst other users are restricted to the surface level elements. A big application from his research is the logic behind using equations over, lines of code. Rather than using a long loops of 'ifs' and 'when' statements to create a loop script, a simple equation such as " f(pelvis) = sin (T x vM x O) x wB" (Malik 2016 p.2) can serve the exact same purpose but in a much simplified form, supporting the notion that there are multiple ways at reducing elements.

5.2.2. Creative Application of procedural ability

A creative application of procedural workflow can be seen in Procedural rhythmic character animation: an interactive Chinese lion dance by Tsai Yen Li. He created a digital lion dance, with the input being musical beats and procedural elements to create parameters for movement control. His report affirms the concepts of Bhatti's report in that they first break down essential components and then represent them through either coding or math. He and his partner recognized the diverse applications of procedural workflows, and instead of using it for data and engineering, they used it for an art form (dancing). Li's biggest deciding factor to use this style of workflow was his discovery that not only was it fast, but it was consistent. Having a mainframe that he added input to, meant that results were always like each other due to them having the same base. It makes logical sense for them, as a lion dance performance requires multiple human that are different but must be within the restraints of human capability. Regarding to architecture, taking the dance moves as metaphor to client changes means that changes can be consistent to its nature, but suited for the change.

6. Case Study

"Scientific research often necessitates the usage of middleware for proof-of concept implementations. In computer graphics, rendering engines are a type of middleware used for such a purpose. For real-time rendering, however, rendering engines often do not provide all functionality that is required, as in real-time, a certain degree of user interactivity aside from graphics is necessary, or can be the center of research." (Wimmer .2008.p. 2)

This project is not aimed at completing a fully functioning product but create a system that will encourage people to see and understand the potential of such a system. Creating a real time simulation from scratch with the base of Unreal Engine requires learning with the consideration of adapting to the systems base nature. This case study explores the process of the creation of the simulation, whilst also pointing out the pro and cons.

6.1. EARLY TESTING AND EXPERIMENTS

6.1.1. Cinema4d

Cinema4d is designed to compute large amounts of data due to its strength at simulating particles and physics. This makes it a strong candidate to host the real time simulation especially since there is the ability to use a plugin named 'X-Particles' which enables the user to code in Python, providing more freedom. However, upon experimenting, it proved itself lacking for the task, due to it not having real time rendering. In other words, regardless of how good the simulation is, there still needs to be long periods of time needed to devote to rendering.

6.1.2 Houdini

Houdini is a procedural animation program that recently released the function to directly line to Unreal Engine 4. They provide the ability to create assets that hold procedural properties, meaning that whatever object the user produces, the same properties will follow. This results in consistency in production, which is great for Unreal Engine, as the user can constantly create new elements that are all consistent with properties. This in application to crowds, implies that no matter the variety of agents produced, they can all possess the same movement style and other attributes. However, Houdini has a steep learning curve due to how in-depth the program is. Even with three weeks devoted to learning the program, it was still incredibly hard to learn enough to produce the desired result for Unreal Engine.

6.2. MODELING AND CONTEXTUALISING

For any simulation to be functional, they first need to be able to adapt to any model or building provided. This first step is crucial in making sure all the elements are compatible and true with the system in Unreal.

6.2.1 Rhino3D

Models and building context were created inside Rhino3d. Rhino is a favoured choice as it accepts a wide variety of file types which is ideal for client deliveries, and the easy user interface allows the designer to be able to quickly optimize the scene. It became apparent that with some programs, they experience problems dealing with complex meshes, and with surfaces that have a null (0) thickness. However, Rhino3d allows the user to quickly fix the scene by finding these object types and reducing mesh definitions and adding thicknesses to surfaces respectively. Not only is Rhino good at import and optimizing, it is ideal for creation of geometry and ultimately offers a diverse

range of file types for export as well, which is perfect for Unreal Engine and cross program communication.

6.2.2 Collisions and optimisation

The first stage in developing a 'stage' for the characters to simulate on is defining the model imported from Rhino3d for the engine to recognise it as a building. In Figure 1, it demonstrates the automatic assignment of a 'bounding box' around the scene, meaning that it doesn't allow direct collision, but only around it. When creating very detailed definitions of collisions, it comes with the consequence that some paths are hard or cannot be accessed. For example, staircases are registered as 'walls' rather than walkable pathways, or if there is a small misalignment of levels, it becomes untraversable. Correct collision parameters are very important because when a wall isn't registered well, then the crowd would always walk into it, as they cannot 'see' the collision or the wall in front. In Figure 1, the green represents walkable paths whilst the cut-out areas are inaccessible

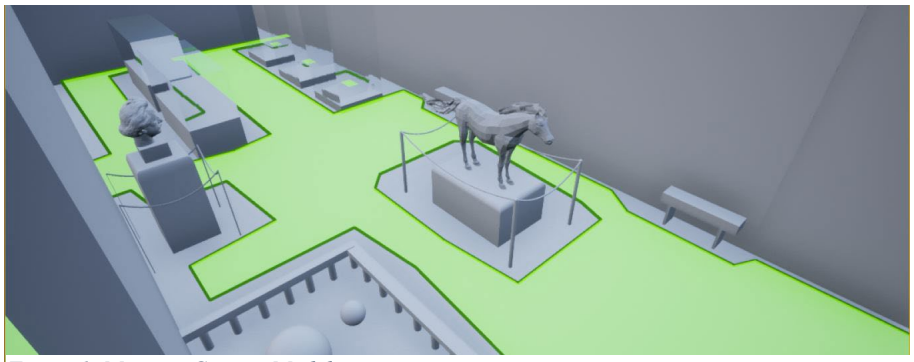

Figure 1. Museum Context Model

6.3 BLUEPRINT

The backbone for both Unreal Engine and this research project is the node-based scripting function called BluePrint. It can be assigned to the level in general and or assigned to each individual agent that is represented by 'class' like layers.

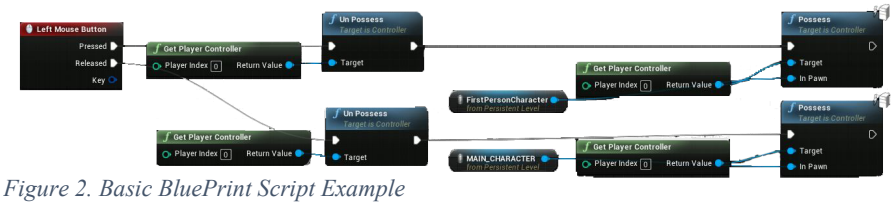

Figure 2. Basic BluePrint Script Example

6.3.1 Agent Crowds

To create a crowd, first, there needs to be individuals. BluePrint provides the ability to manage multiple ‘classes’ of people all at once, allowing diversity in mass. Referencing to Bhatti’s paper, it was key that chaos and movement can be summarised and quantified. In our case, instead of maths, we used scripting in Blueprint, keeping workflow efficient and manageable.

6.3.2 Spawning System

People in real life do not spawn from thin air, hence the introduction of agents must be smart and feel natural. Extra rooms are located outside of the users view, creating the illusion the agents are spawned behind doorways. With the use of BluePrint, spawn rate and quantity can be controlled and modified easily. This is useful for clients who have a specific number of people they want simulated in an environment. Due to the nature of procedural node-based scripts, this can be done in one alteration to the values, and the script will still run perfectly fine with no delay.

6.3.3 Collision and Crowding

The default method of navigation is defined by Unreal as ‘AInavigate’ and it is a good system that allows the agents to traverse towards a certain location. However, once there were multiple agents, it was observed upon crowding the agents would stop moving due to them being unable to untangle themselves within their logical parameters. A straight path to a location, is now blocked by multiple other agents who do not know how to react to each other as they all have their own instructions. Upon research, there was another method of movement named ‘AIdetour’, and this method allows the agents to evade any objects in their line of path, including other agents. Whilst this solves our initial problem of the stagnating movement in crowds, it also introduced other problem. Movement became forced and even when there is meant to be points of crowding/ friction, the ai would find a way that is inhumanely possible to navigate around each other. A counter to this problem was to integrate both the default assets of Unreal, mixed in with the user defined scripts in BluePrint. BP offers the freedom in targeting specific controls and attributes that the default settings offer, allowing them to be altered and specified in scripting.

6.3.4 Behaviour Tree

Behaviour Tree or ‘BT’ is the system that regulates the usage of BluePrint which is particularly ideal in profiling characters. Rather than repeatedly executing the same script for each character, the program can now trigger

events based of ‘true’ and ‘false’ situations (Booleans). In Figure 3 the graph represents the hierarchy of the BT. Each node under ‘sequence’ are individual scripts that are executable based on the event defined. In this particular script, it tells the agent to, receive target point, wait and then walk to point. When the agent fails to execute a certain branch, then it will execute into another branch (to the right), allowing two layers of decisions which greatly simulates human decision making. An application of this was to fix the crowding problem mentioned in 6.3.3 above. The moment an agent found itself to be stuck due to crowding, they would execute another task, untangling the situation as they all have new directions to.

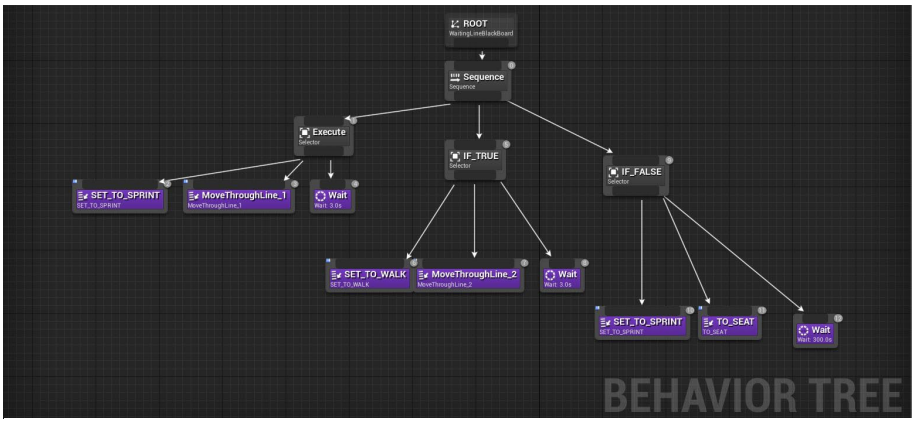

Figure 3. Behaviour Tree for Movement

6.3.5 EQS system

EQS or Environment Query System is a new experimental feature that accompanies the Behaviour Tree system. It is classified under as Unreal’s ‘Artificial Intelligence System’ and it helps receives data from the scenario and context around the agent. In other words, relative to where the actor is in the scene, it can react to its relative surroundings. The human senses, whether it be hearing, smell or the more importantly, sight, are all realistic elements that causes people to perceive their bearings (Dutra.2014.pg1). Hence by using EQS, it increases the dynamics of the system, adding more layers to the realism of the crowds. An implementation of EQS in the research project is vision perception, where the agent can only react or execute functions only if they have a direct line of sight to their target. In Figure 4, the system returns a green sphere when the value is true (that the agent can’t see the user) however when it does see the user, then it will return purple, meaning it will execute a different function. In this specific case, it shoots the command for the agent to go towards the user, and if user is not visible, then keep walking

around randomly. In terms of the crowd simulation, this could be useful for simulating wayfinding and signages, where agents can only react if they have a clear view of signage, proving the design is efficient. It also fills in lesser considered scenarios, such as if in a dense crowd situation, are people of shorter height able to navigate successfully around spaces when there is obstruction. Other uses of the EQS includes the ability to virtually map out the space around the agents, or the ability to hear or even the ability to choose the best route based off distance, the possibilities are plentiful.

Figure 4. EQS Trace example

6.4 SIMULATION USAGE

6.4.1 Camera Work

Camera is a very important yet generally overlooked aspect of simulations. A well-placed camera can both narrate the features whilst concealing faulty productions. The default camera for UnReal engine is 3rd person, where, positioned right behind the user character as shown in Figure 5(Left). The most ideal angle is birds' eye, to provide the ability to view the crowds as if patterns from an aerial down shot as portrayed in Figure 5(Right). However, the problem arises when roofs and high walls interfere with the clear line of sight down. The problem was solved when the obstructing objects were referenced in BluePrint, causing the visibility to change whenever the camera is recognised in top view. The addition of a new camera changed the whole simulation and its usage. The user now has visual access to all the rooms and all the agents regardless of where the player is, which is perfect for observing big crowds that access multiple rooms simultaneously.

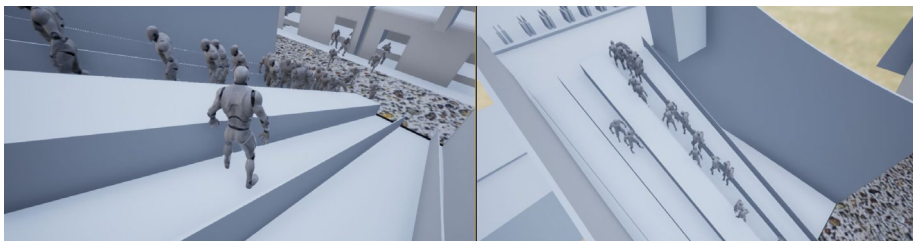

Figure 5. Camera 3rd person (Left) Birds eye (Right)

6.4.2 User Interface (UI)

As a simulation that is both user and client focused, it is important that the user interface must be easy to interpret. The controls act like a game where the user walks around using the WASD keys, unlike programs where they have assigned combinations to be able to pan, rotate and zoom. The game like nature of the system allows all three of those movements to be simplified into one seamless navigation from a character. Spawning characters are done through button presses and the camera is based off mouse movement. There were attempts at being able to spawn walls and variables based on mouse location, but it was not successful within the time frame of testing. Further improvements, that could be implemented in the future, is an inventory system. Where the user can simply press 'TAB' on the keyboard and all the available elements like walls or columns can be displayed and dragged into the scene. This will allow the client to instantly see the options and place them accurately, with no time wastage.

6.5 FINAL PRODUCT

6.5.1 Functions and usability

Now that the mainframe of the system has been created, upon receiving a building context/object file, the simulation quickly can be set up in around twenty minutes. The only set up required for the developer, is making spawn points, assigning pre-made scripts to agents, texturing and importing other pre-made assets. After that, the system will be automatically rendered and able to simulate the crowds whilst also interactable by the user, and most importantly, all this is in real time. The user can walk around the scene in third person, experiencing the building design from the perspective of a pedestrian, granting the ability to identify problems from a simulated human height. They can introduce new variables such as doorways, walls, columns, hazards to alter the course of the crowds, who can only react upon the requirements of the EQS system. Currently, due to the EQS system, this means the agents can only react upon having a clear view of the new variable. The player can also change camera view whenever, to clearly observe the patterns from a higher angle, optimising the ability to observe and interact.

6.5.2 Testing Scene

Whilst the simulation can be performed with virtually any building, in consideration for this paper and its purpose to prove the benefits of real time, a specific scene was created from scratch. This scene is composed of an imitation of a train station, airport, art gallery and a large blank room. The train station is to prove the system works with dealing with mass crowds that

spawns from multiple entrances (train doors) and its purpose is to validate that game engines can process large amount of data without stuttering. This was successful, and the crowd was observed to navigate the corridors in a human like manner. The second section was an airport which purpose was to simulate a multi-process scenario. In airports, there is a process of check in, luggage sends, bag check, security scans and finally boarding. This multi-level setup is ideal in testing the randomness of the agents, giving them the ability to decide which 'booth' to go, creating an effective test of cross pathing and dispersion based off different waypoints. The art museum is to display random human movement from the different interactions to artworks. Some like to move in closer to see the painting, and some like to walk side to side to observe clay works. This unpredictable type of movement is ideal in creating a challenge for the ai to both replicate realistically and to solve navigation. Finally, the blank room is simply for the user to have fun and build walls in an open space that controls the crowds freely.

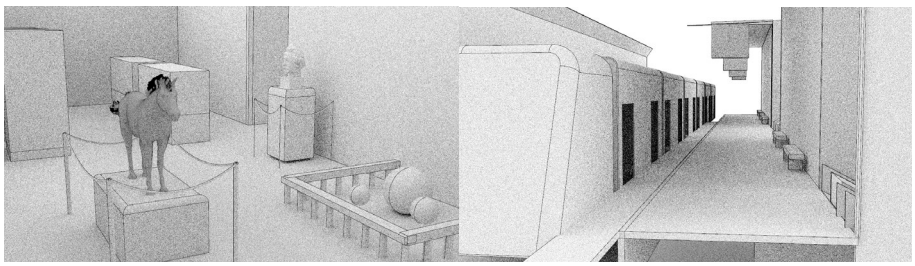

Figure 6. Modelled Scene (Untextured), Gallery (Left), Train Station (right)

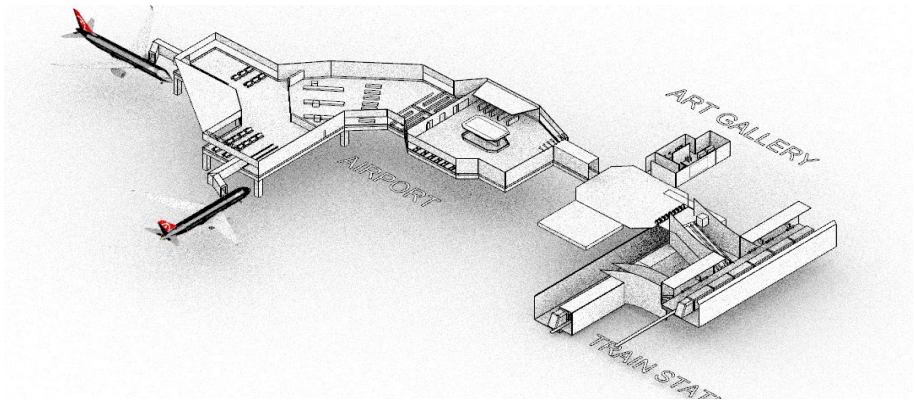

Figure 7. Whole Scene (Untextured)

6.5.3 User response

This section wasn't extensively tested amongst many users especially not in a controlled experiment context. Hence its observations can only be taken as assumptions rather than validated evidence-based observations. Users who interacted with the system managed to easily pick up the controls and explored the program with no trouble. They were all entertained in their exploration of the scene through a 3rd person character and was intrigued when they could play with the crowd directly. Not many of them doubted the validity of the crowd simulation, but it was interesting to observe, that before being explained the goal of the program, most of them treated it like a game. Their approach was very much like *The Sims©,* a game about simulation, but most players approach it with an entertaining perspective, trapping the characters and messing around with them. None of the users expressed frustration with the flow of the system, unlike how other programs would often experience computing lag when imputing large amounts of data. Ultimately, user experience returned positive, and most of them enjoyed using the system whilst most agreed they understood the potential.

7. Discussion

The goal of the research paper at its core, was not to create a fully polished product that could replace current programs, but rather an investigation into the benefits and application of real time elements in crowd simulations. In that philosophy, the research was successful due the ability to create a real time crowd simulation in Unreal. The user could effectively and easily import their context building, create a crowd, let them simulate their navigation through the buildings and most importantly, the user could change the design and the crowd would instantly react with no consequence of processing and time.

The project lead to the realization that human movement behavior may seem chaotic at first due to combination of multiple variables such as 'where to go', 'what is in my way 'or 'where's the fastest route'. As a resultant, the need for simplification and ability to change was key in making this system work. Hosting the research in a game engine was the perfect match, as the scripting allowed for simplification of chaos, representing movement with just components and nodes, whilst also letting them execute variables all at once. Furthermore, the focus on real time began to really show its colors when it introduced the freedom of input at any given time, rather than anticipated only in the beginning. Without real time, wayfinding flow simulations would need to be restarted every time a new input is introduced, which is tedious and unpractical for experimental use.

Throughout the research and creation of the simulation system, there was a heavy emphasis and consideration of the benefits of such an interactive system on design decision psychology. Jodie Goodman in one of his psychological articles highlights the implication of the ability to learn based

off error detection and correction skills and how it ultimately leads to longer lasting (and valid) knowledge. With this notion as an anchor point, even though the project wasn't fully achieved nor tested amongst multiple people, the few participants who tried the early stage of the product all expressed interest in the product and wanted to explore the what the crowd could do via its altered surroundings. Although they approached it more like a game (due to the nature of a game engine) the purpose was still fulfilled. The interface was easily understood, exploration and experimentation of outcomes were encouraged, and ultimately, they all showed interest when they could see the crowds move according to what they did to the surroundings and constantly wanted to interact with it.

Overall, the project worked a lot better in some areas whilst expectedly lacked in others. It paved and revealed its potential in the future if more time and testing were available. There were many wanted but unexplored ventures in this paper, such as Virtual Reality (VR) and Pixel Streaming. UnReal Engine 4 is very well optimized for VR, initially the research was planned to implement HTC VIVE, a headset that is also paired with controllers that allow users to both view the environment and interact with it without a keyboard or mouse. The benefit of VR is to literally set the user in the shoes of the agents. Letting him or her see what the agents would see, giving the ability to identify problems based off the agent's point of view. This includes sight of signage, anxiety from personal space being overcrowded, or even just the unsightly feeling of seeing so many people in one space. On the other hand, Pixel Streaming is also a new experimental feature of UnReal, and it provides the ability for the scene to be hosted on servers. This results in the accessibility of the program virtually anywhere with a strong internet connection. Not only that, but clients could run the full program even on outdated hardware, because everything is hosted on the server and is essentially just streaming data on par to that of a video. This is again, an extremely nice touch to the idea of 'real time' as it is instantaneous access to the program, anytime, anywhere.

Ultimately, however, in its most perfect form, the system could change the notion that simulations are an 'add on' to building design, something that is only performed after the building is digitally produced. But instead, it could become a dualistic design method, where the designer could integrate both simulation and construction, where the simulation would run completely through the whole creation process, informing design changes in real time and validate those choices.

8. Conclusion

How can the element of Real Time impact and shape the user experience, time investment and the design itself when working with such a highly chaotic study such as crowd simulations?

To cut through the mass data and chaos, first and foremost requires speed and responsiveness. If the program is slow, then it will decrease user morale and increase frustration. A game engine solves this problem instantly, giving the user instant response upon input, simply due to the power of the nature of Unreal Engine. Not only that, but now, users are encouraged to constantly make trials and errors, uplifting a sense of entertainment whilst also serving the exact purpose of conventional programs. Whilst currently, the project's crowd simulation is nowhere near as realistic as other programs such as Mass Motion, what was achieved in such a short amount of time amplifies the potential of the system and proves that technology is not the issue now, but rather the user himself.
So, while technology is already at an incredible level, and will continue to increase, why are we still limiting ourselves into a world of stagnation and not delve more into real time?

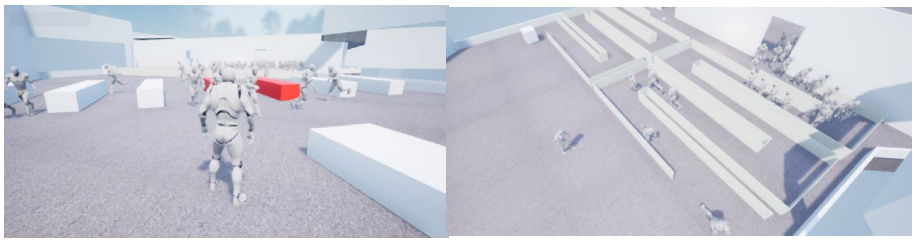

Figure 7. Final Scene, 3rd Person (Left), Birds eye (Right)

Acknowledgements

This project was made possible through the support of UNSW built environment, BIM Consulting and Architectus.

University mentions Nicole Gartner, Hank Frausler, Yannis Zavoleas, Cristina Ramos, Nariddh Khean and friendly peers

Big thanks to Ali Siddiqui from BIM Consulting for help in the direction and supervision of this project.

References

Wimmer, M.: 2008, Penta G- A Game Engine for Real-Time Rendering Research, Computergraphik und Algorithmen, Vienna, 1-30

Zhou, S. and Cai, W. and Chen, D. and Luo, L.: 2019, Crowd Modelling and Simulation Technologies, ACM Transactions on Modelling and Computer Simulation, 1-6

Dutra, B.T and Priem, G. and Calvalcante, B.B and Creto, V.: 2014, Synthetic Vision-Based Crowd Simulation, Reactive vs Reactive Planning approaches ,1-4.

Ullrich, T. and Schinko, C. and Fellner, D.W.: 2010, Procedural Modelling in Theory and Practice, 18th International Conference In Central Europe on Computer Graphics, Visulization and Computer Vision, EuroGraphics, 1-20

Lozano, M. and Morillo, P. and Orduna, J.M. and Cavero, G.V.: 2008, A new System Architecture for Crowd Simulations, Journal of Network and Computer Applications, Elsevier, Spain, 1-5

Macedo, J.: 2004, International Journal of Production Research **42**(17), 3565-3588. Unified Structural- Procedural Approach for Designing Integrated Manufacturing System

Caulfield, B.: 2018, "What's the Difference Between Ray Tracing and Rasterization". Available from: Nividia < https://blogs.nvidia.com/blog/2018/03/19/whats-difference between-ray-tracing-rasterization/>(accessed 26 November 2019).

Haines, E. and Moller, T.A.: 2019, "An Introduction To Real-time Ray Tracing". Available from: Apress < https://www.apress.com/br/blog/all-blog-posts/an-introduction-to-real-time-ray-tracing/16559492/> (accessed 25 November 2019).

Bhatti, Z. and Abid, H.: 2016, Procedural Animation of 3D Humanoid Characters Using trigonometric Expressoins, Bahria University Journal of Information and Communication Technologies, Pakistan, 1-7

Barron, J. and Sorge, B. and Davis, T.: 2001, Real Time Procedural Animation of Trees, Clemson, 1-10

Fewings, R.: 2001, Wayfinding and Airport Terminal Design **54**(2), Journal Of Navigation, 177-184

Raman, D. and Crawford, E. and Wu, Y.: 2013,Wayfinding in the Rail Environment: Technology and Behaviour Review, CRC for Rail Innovation, Australia, 2-13

Weiss, A.: 2013, Museum Signage Design and Implementation, Graphic Communication, California, 1-4

Hughes, K.:2015, "Museum and Gallery Wayfiunding: Tips for Signage, Maps and Apps" Available from: The Guardian < https://www.theguardian.com/culture-professionals-network/2015/aug/25/museum-gallery-wayfinding-tips-signage-maps-apps/> (accessed 24 November 2019)