**FACULTY OF ENGINEERING, COMPUTING AND THE ENVIRONMENT**

**School of *Computer Science & Mathematics***

**BSc DEGREE**

**IN**

***Computer Games Programming***

**PROJECT REPORT**

**Name:** Maciej Baranowski

**ID Number:** K2120853

**Project Title:** DJ Simulator

*https://github.com/LightBlazeMC/dj-simulator*

**Date:** 24/April/2025

**Supervisor:** Andreas Hoppe

KU London Logo

# Plagiarism Declaration

The following declaration should be signed and dated and inserted directly after the title page of your report:

## Declaration

I have read and understood the University regulations on plagiarism and I understand the meaning of the word *plagiarism*. I declare that this report is entirely my own work. Any other sources are duly acknowledged and referenced according to the requirements of the School of Computer Science and Mathematics. All verbatim citations are indicated by double quotation marks (“…”).  Neither in part nor in its entirety have I made use of another student’s work and pretended that it is my own.  I have not asked anybody to contribute to this project in the form of code, text or drawings.  I did not allow and will not allow anyone to copy my work with the intention of presenting it as their own work.

**Date:** 24/April/2025

**Signature:**



# Table of contents

[Plagiarism Declaration 2](#_Toc196408330)

[Declaration 2](#_Toc196408331)

[Table of contents 3](#_Toc196408332)

[Introduction 4](#_Toc196408333)

[Aims and objectives 4](#_Toc196408334)

[Personal developmental goals 4](#_Toc196408335)

[Summary of upcoming analysis 4](#_Toc196408336)

[State of the Art Review (Literature Review) 5](#_Toc196408337)

[Niche review 5](#_Toc196408338)

[Beat Saber VR 6](#_Toc196408339)

[Vinyl Reality 7](#_Toc196408340)

[GTA Online – Nightclub Missions 7](#_Toc196408341)

[Product analysis review 8](#_Toc196408342)

[Tools and Technologies 8](#_Toc196408343)

[Game engine review 8](#_Toc196408344)

[Secondary features review 9](#_Toc196408345)

[Entertainment review 10](#_Toc196408346)

[Development cycle review 10](#_Toc196408347)

[Conclusion of Literature review 10](#_Toc196408348)

[Analysis, requirements specification and development methodology 10](#_Toc196408349)

[Game engine analysis 10](#_Toc196408350)

[Secondary features analysis 11](#_Toc196408351)

[Project specific functional & non-functional requirements 12](#_Toc196408352)

[Requirements prioritisation 12](#_Toc196408353)

[SWOT analysis 12](#_Toc196408354)

[Risk and Contingency planning 13](#_Toc196408355)

[Development methodology analysis 13](#_Toc196408356)

[Conceptual Game Design 14](#_Toc196408357)

[Product overview 14](#_Toc196408358)

[Game outline 15](#_Toc196408359)

[Gameplay 16](#_Toc196408360)

[Level design 17](#_Toc196408361)

[Character design 18](#_Toc196408362)

[Graphical User Interface design 18](#_Toc196408363)

[Meeting requirements via design 19](#_Toc196408364)

[Technical Game Design 20](#_Toc196408365)

[Implementation 20](#_Toc196408366)

[Testing and Evaluation 20](#_Toc196408367)

[Legal, Social, Security and Ethical Issues 20](#_Toc196408368)

[Critical Review and Conclusion 20](#_Toc196408369)

[Technical Appendices 20](#_Toc196408370)

[References 20](#_Toc196408371)

# Introduction

DJ Simulator is a strategic simulation game made in Unity 2023 LTS. The game has been developed mainly on macOS, however anticipated platform deployment is both Windows and macOS. The game is set in a low-budget nightclub, where the player plays as the DJ who must entertain the crowd with good music choices.

## Aims and objectives

The primary aim of the game is to perform excellent DJ sets that score plenty of points within a 5-minute time limit. In order to gain points, the player must ensure that they are selecting tracks that are of a genre that is the most popular within the crowd.

The main game objective is to score the most points within the 5-minute time limit. In order to score the most points, the player must first identify the most popular genre within the crowd. Once identified, the player must play a track of that genre in order to start accumulating points. The crowd will react accordingly. The player must continue to monitor the crowd, due to the fact that the crowd will change in numbers. As the popular genre changes within the crowd, the player must choose new music that fits accordingly. Once the 5-minute timer elapses, the game will notify the player of their score. Not selecting appropriate genres, or not changing the music based on the crowd’s changes in favourite genres will not award many points.

## Personal developmental goals

When embarking on this project, I had 3 main goals in mind:

* To create a convincing simulation of tasks & experiences of a typical beginner DJ
* To realistically simulate crowd behaviours that can be observed within nightclubs
* To simulate a low-budget nightclub environment that feels ‘alive’ and responsive to the player’s actions.

Being a DJ myself, I intended to create a game that is able to simulate and showcase the typical workflow and tasks that a beginner DJ in a low-budget venue would experience. This is an experience that many DJs can relate to, and in order to deliver on a genuine representation of this experience, I set the 3 main goals above as my developmental goals since I believe that achieving these well will result in a realistic depiction of the intended experience.

## Summary of upcoming analysis

This report will delve into an analysis of how this project was developed and formed to achieve the personal developmental goals outlined above. It is important to analyse both the infancy and late development stages of the project thoroughly.

The early planning stages will be covered in the form of a Literature Review, where sources of inspiration will be explored. The information collected from this research will then be thoroughly analysed and translated into a development methodology and requirements specification, where a clear developmental plan can be established.

At this point, clear requirements and goals will be apparent, therefore conceptual and technical game design can be discussed. A deep insight into the reasoning of how the game is designed and why those design choices have been made is crucial, due to the many intentional mechanics that are implemented in order to create an immersive simulation. Many of those design choices are implemented in unique and abstract way, therefore a detailed analysis of the technical implementation is also necessary.

Since this game contains a diverse simulated system, thorough testing is necessary. The report will detail how this has been achieved to ensure that the game’s simulation systems are robust, and as many simulation cases as possible have been tested. Legal and moral issues will also be considered here.

Finally, the report will conclude with a critical evaluation and conclusion of the information and research conducted during the development of this project. This will be followed by technical appendices and references.

# State of the Art Review (Literature Review)

## Niche review

DJ Simulator aims to depict the humble beginnings of a new DJ. Most DJs, when starting their career, will perform in low-budget, run-down venues [7], therefore it is important that this project depicts the gritty atmosphere well. There are certain industry standards within the DJ world, such as the hardware, which often tends to be Pioneer CDJs or Technics SL-1200s [8]. It is important that the game represents those expected industry standards faithfully in order not to alienate the player – which is highly likely to be (or aspiring to be) a DJ.



*Figure 1 – A typical Pioneer DJ setup.*

Not many gaming products exist within the niche of the DJing world. There does not seem to be a strong link between video games and DJing. Despite that, several gaming products and generic niche conventions have been found and analysed. While no singular product has satisfied the entirety of the developmental objectives of this project, the following products each contain features that have proven insightful for the development of this project. Identifying those, and adapting them for the needs and goals of this project will help create a cohesive DJing experience. Beat Saber VR has been selected as an example of reactive environment design, Vinyl Reality has been selected as an example of realistic DJ hardware simulation, and finally GTA Online has been selected as an example of a realistic crowd simulation system.

## Beat Saber VR



*Figure 2 – In-game screen shot of Beat Saber VR.*

Beat Saber VR is a music rhythm game where the player must break blocks with their sword in beat with the music. This game features heavily adaptable environments, which change based on the player’s actions and song choice. Visual effects may appear or morph when the player triggers an event system [1]. This means that the game adapts visually based on what they player is doing and how well the player is doing.

This level of visual feedback is a great device for signalling important gameplay messages to the player, and creating an immersive, engaging and ‘breathing’ in-game world [2]. Based on this observation, it has been decided that in aid of improving environmental immersion and player feedback, audio reactive lighting will be implemented into the project.

Audio reactive lighting is a very typical element of most conventional nightclubs [3], therefore including it in the game is a great way of combining visual feedback, environmental immersion and generic convention within the DJ niche; most DJs would have had experience performing in a venue with reactive lighting.

## Vinyl Reality



*Figure 3 - In-game screen shot of Vinyl Reality (via Steam Store page).*

Vinyl Reality is a game that deeply focuses on track selection, and the performance of said track. It is a deep emulation of a vinyl-based DJ system. Being developed by professional DJs [4], this product provides an excellent representation of typical tasks for a performing DJ. Most of the expected DJ system mechanics are thoroughly and realistically portrayed within this product.

The implementation of allowing the player to drag-and-drop music [5] on the turntable is an excellent way of giving the player control over the in-game world. This product is a strong source of inspiration for the simulation of the interactable in-game systems within the project. It allows for realistic interfacing and control over the in-game world, while retaining it within a format that is very synonymous within the DJ niche.

The turntable depicted within Vinyl Reality is the Technics SL-1200. It is a turntable that is a staple within the world of DJing, and has been around since the 1970’s, making it a crucial piece of hardware ad heritage within the DJ world [6]. More importantly, the selection of such widely recognised hardware means that most DJs can easily interact with the in-game depiction of the turntable, since it is highly likely that they have used the real-life counterpart. Using this renowned system reduces the risk of confusion or alienation of the player, by utilising a familiar layout and system.

## GTA Online – Nightclub Missions



*Figure 4 – A crowd and a DJ inside the Music Locker nightclub in GTA Online.*

The nightclub missions form a relatively small element within Grand Theft Auto Online’s expansive and busy world. They are however a perfect example of excellent crowd simulation within a nightclub environment. There is a multitude of NPCs that have unique behavioural patterns [9], and enhance the mise-en-scene of the nightclub drastically.

While in GTA Online the crowd is mostly utilised for an aesthetic purpose, within DJ Simulator the crowd will provide visual aid for the player. The player should be able to identify popular music genres by looking at the crowd.

A realistic implementation of crowd simulation within the nightclub can greatly benefit the project by highly enhancing the environment (and its realism) and also as a means of providing visual feedback to the player on their performance and areas of improvement.

## Product analysis review

To summarise, based on this analysis, utilising the reactive environmental systems of Beat Saber, the realistic crowd simulation systems from GTA Online nightclubs, along with the detailed, recognisable and robust hardware simulation systems of Vinyl Reality, is highly beneficial for creating a lively and detailed simulation of a DJ setup in a nightclub. Combining these systems creates a synergy between functional, immersive, ‘breathing’ environments and recognisable and realistic DJ hardware. These sources of inspiration provide an insight into the ingredients required for a robust, practical and non-alienating simulation of a DJ's environment within a nightclub.

# Tools and Technologies

## Game engine review

There are several essential features that the game engine of choice must provide for this project:

* High performance – large crowds and real-time visual effects can be computationally expensive. Therefore, a well optimised engine is needed that will not suffer from performance issues when simulating detailed scenes.
* Powerful scripting – there are many features that will need to be implemented that require complex logic to work. The engine must provide useful tools and libraries in order to make this process easier.
* Stability – since this project is not utilising any novel/evolving concepts (such as generative AI), an engine version with the greatest stability is of highest priority.
* Versatile animation system – in order to achieve realistic crowd simulation, an engine that can handle and calculate animations well is important.
* Environmental sensing – due to the large number of stimuli within the scene, the engine must be able to identify and act accordingly to player and environmental triggers.

Based on this list of requirements, two prospective engines have been considered: Unity 2023 LTS and Unreal Engine 5. Both of these engines satisfy the majority of these requirements much better than other game engines such as Godot or GameMaker, which either lack the necessary functionality, or are not as capable with it.

Unity is known for excellent performance, even with complex scenes. Therefore, scenes with large crowds should not cause performance issues for this engine. C# is a very powerful scripting language and is the one used by Unity. C#, along with MonoBehaviour [10] provide extremely powerful scripting opportunities. The 2023 LTS version of Unity provides all necessary features while being very stable. No preview (or more recent) version is needed here, and the project will benefit from using the very stable LTS branch. Unity’s Animator is very powerful and will allow for complex NPC simulations. Unity’s colliders and map constructing systems work very well with events and triggers.

On the other hand, Unreal Engine provides high fidelity graphics at ease [11], albeit with a large performance cost. The usage of C++, which is a very powerful language, along with the versatile and practical Blueprint system offers huge scripting capabilities. Unreal Engine offers no LTS solution, and crashes (especially on macOS) are very frequent. Unreal Engine provides a very powerful [12] suite of animation and simulation tools. They are very detailed, however overkill for this application. Unreal Engine’s environment sensing is excellent, especially when paired with the Blueprint system.

## Secondary features review

Due to this project being a DJ Simulator game, audio analysis algorithms are very important here. Unity offers the GetSpectrumData function, which provides the block of audio frequencies (spectrum data) of the AudioSource that is currently playing [13]. This is essential for the reactive crowd and lighting that is envisioned within this game.

A complex AI pathfinding, such as Dijkstra's Algorithm may not be necessary for DJ Simulator. A random crowd distribution model can instead be opted for since it provides realistic and convincing randomised NPC behaviours. Furthermore, the traversal of the NPCs is not important in this game as much as the actual portrayal of them on the dancefloor.

To simulate NPC behaviours, an animation state machine can be opted for. This is because the animation state machine system within Unity allows for various animation triggers, based on Boolean states, and also enables smooth transitions [14]. These features provide smooth and convincing crowd NPC behaviour modelling.

## Entertainment review

It is important that DJ simulator is a fun game. According to Raph Koster in his book “A Theory Of Fun For Game Design”, a fun game can be defined as one that is “a source of enjoyment” [15]. He states that “fun is the act of mastering a problem mentally” [15]. The problem in this scenario being the goal of entertaining the crowd with the right music. Based on this, it can be concluded that DJ Simulator is a fun game due to the fact that it provides the player “a source of enjoyment” [15] by allowing the player to “master a problem mentally” [15].

## Development cycle review

For a successful and efficient development cycle, it is critical that the project follows a development methodology. For this project, there are two methodologies that could be considered: Agile and Waterfall. Waterfall is “particularly useful for large, complex projects with very specific and unchanging requirements” [16], whereas Agile “works well for projects that prize learning and are seeking or refining their product” [16].

## Conclusion of Literature review

Based on this literature review, the project now has information based on the niche market of DJing and also a technological and developmental overview of potential possibilities and opportunities that can be used for the development of DJ Simulator.

# Analysis, requirements specification and development methodology

Having considered the research conducted within the literature review, it is now essential to streamline this information into segments that will be advantageous for this project. Having multiple approaches for many tasks means that now the best approach must be considered and selected.

## Game engine analysis

Selecting the appropriate game engine is crucial. While both engines provide excellent opportunities for developing a good DJ simulation game, the engine of choice is Unity 2023 LTS. Based on the analysis conducted within the literature review, a table weighing the pros and cons of each engine has been created. Using a table easily illustrates the greater advantages of Unity over Unreal Engine for this application.

|  |  |  |
| --- | --- | --- |
|  | **Unity 2023 LTS** | **Unreal Engine 5** |
| **Performance** | Due to lesser (unnecessary for this use case) graphical fidelity, this engine provides better performance | The high-fidelity graphics are very computationally heavy. |
| **Scripting** | C# and MonoBehaviour – provides all necessary functionality for this project. The scripts compile quickly on play. | C++ and Blueprints – provides plenty of additional functionality. However, it is not needed for this project. C# in-engine compilation is not straightforward. |
| **Stability** | LTS version offered – provides excellent stability and no unexpected updates. | No LTS version. Very unstable on macOS. |
| **Animation/simulation systems** | Robust suite of animation tools. Everything needed for a realistic crowd/NPC simulation. | Overwhelming suite of tools. Many of which are not needed for this application. |
| **Environmental sensing** | Great interaction between colliders, triggers and MonoBehaviour scripts. | Great implementation of Blueprints. |

Looking at the data in this table, it is can be observed that Unity 2023 LTS should be the engine of choice. It fulfils all requirements, without much extra bloat. Being an LTS release, it should not cause down-time or issues related to unexpected behaviours or updates.

## Secondary features analysis

Unity provides the GetSpectrumData function [14]. This in-engine function provides a solution for reactive lighting. Other than this, either manual keyframing of waveforms would be necessary, or a highly inaccurate estimated down-beat calculation based on the track’s BPM (BPM/60). Using manual keyframing is extremely time consuming and means changes in music will not be updated within the engine automatically. Using the BPM down-beat calculation is an extremely flawed solution, since it assumes the drum track is of a constant, never-ending 4/4-time signature duration, and estimates the music’s downbeat based on the BPM/60 calculation, intending to estimate exact time markers of down-beats. Not much music exists composed in this monotonic way. Furthermore, due to audio latency, this calculation would quickly become desynchronised. Therefore, GetSpectrumData is the best choice, allowing for automated real-time audio waveform data processing, with drag and drop analysis based on file importing. This method, while difficult to initially develop, ensures that all critical music analysis can be done without much more development efforts, and it is done extremely accurately based on each individual music file’s waveform data.

Since Unity provides a robust Animator state machine [12], this is the best solution to use. It is built into the engine, and allows for Boolean conditional scripting, that calculates what animation the NPC should perform based on Boolean conditions. External animation rigging could have been used via Mixamo or After Effects, however that would be a tedious manual process, would not integrate within Unity’s scripting and would not be directly controllable by the engine. Therefore, no external tools are necessary beyond the Animator state machine provided by Unity.

Due to the game revolving around music, and specifically track selection by genre, it is critical that the player is provided with a diverse library of music. For legal purposes (which will be later discussed), the music must be copyright free. Some of the music can be composed by me, using my strong experience with music production. However, producing a good quality takes time, therefore original content can be combined with the multitude of royalty-free music libraries accessible via YouTube, SoundCloud and other sources.

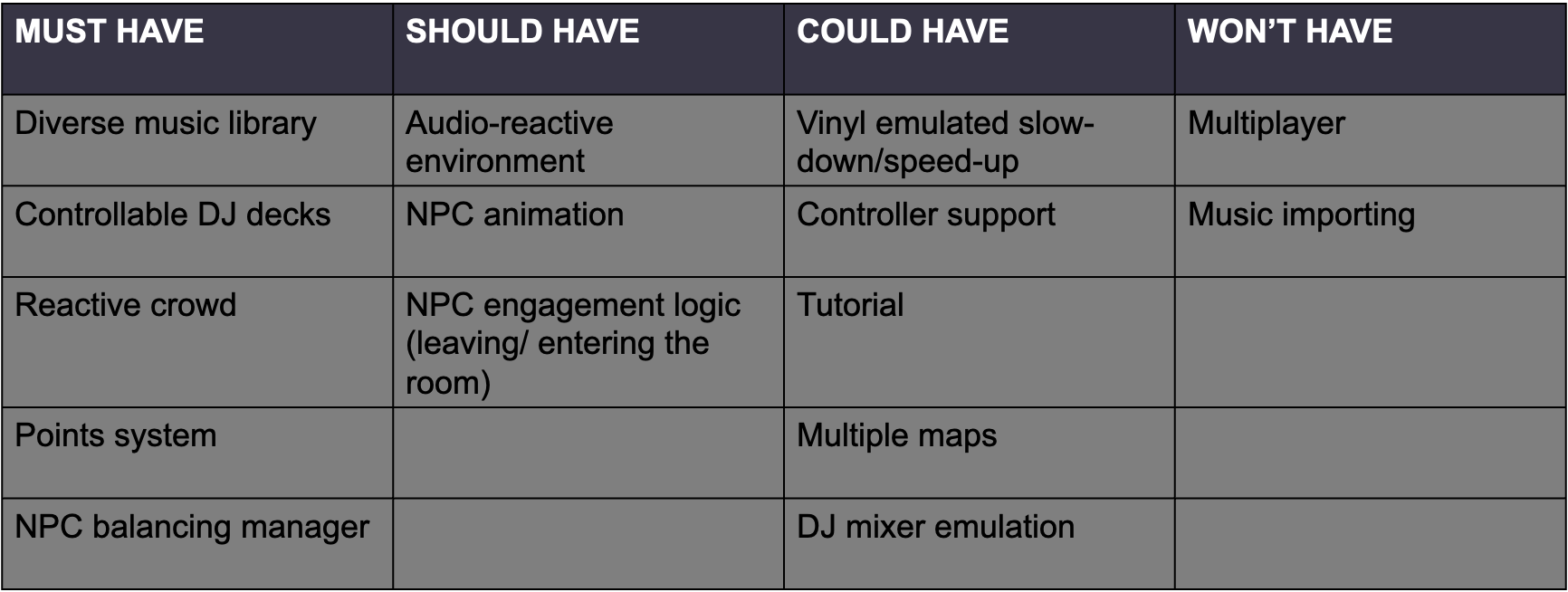
So that the game can be fun, it must allow the player to “master a problem mentally” [15]. This problem can be presented via a timed points system. This provides the player with a time limit to achieve the best possible performance. This mechanic is a good choice since it closely models the way a real-life DJ has to “master [the] problem mentally” [15]. It is a great visual indicator for the player to see how well they are performing. Combining the fun factor and effective visual feedback of this mechanic makes it a great choice for this project, while invoking a sense of urgency and providing the player with a clear target, purpose and objective.

## Project specific functional & non-functional requirements

* Based on the conducted research and analysis, the following functional and non-functional requirements have been identified:
* Powerful, high-performance game engine
* Realistic and reactive crowd behavioural simulation animator & state machine
* Reactive environmental lighting elements
* Real-time music analysis algorithm
* Diverse, multi-genre, royalty-free music library.
* Simulated DJ hardware in accordance with industry standards
* Game logic that assesses the player’s performance and returns feedback via points
* A convincing & immersive map
* Straight forward visual cues on preferred crowd music genres

## Requirements prioritisation

Not all of these requirements will be able to be fully met. The MoSCoW system is a great way to quickly organise priorities of requirements. Based on the analysis of this project, it is important to focus on core gameplay and simulation features. The game must have a purpose, and a means of executing that purpose. Using a MoSCoW analysis table allows the development process to focus on critical features first, and then implement less important features later. This means that the final product will not be missing any of its core features.



## SWOT analysis

Conducting a SWOT analysis greatly helps ensuring there is no unexpected issues arising or any other factors that can result in project downtime. Considering how complex of a system this project is, avoiding downtime and development-halting issues is a must.

|  |  |  |  |
| --- | --- | --- | --- |
| **Strengths** | **Weaknesses** | **Opportunities** | **Threats** |
| * Utilising version control. * Knowledge of niche. | * Time constraints * Lack of knowledge | * Documentation * Tutorials | * Software issues – engine crashes, bugs. * Lack of implementation. * Time limitations. * Data loss |

## Risk and Contingency planning

Based on the SWOT analysis above, several risks and threats have been identified. The project’s success is heavily dependent on the resolving and mitigation of these issues, to ensure no developmental downtime occurs.

Using a legacy Long Term Support (LTS) branch of Unity has been an intentional decision in order to mitigate software issues. When beginning this project, Unity 6 Preview has been available for use. However, upon testing and exploration, it has been found that it offers no real advantages for the requirements of this project. Instead, it contains many undocumented issues and is much less stable, causing frequent crashes. Using a 2023 LTS version of Unity is an excellent balance, where it is recent enough to still be supported, and mature enough to have rich documentation and bug/crash fixing patches.

A major threat for this project has been a lack of implementation. This means that certain requirements outlined for DJ Simulator require solutions and implementations that are not well documented and do not provide much support. Real-time audio analysis is an example of a requirement that may suffer from the threat of lack of implementation. There are not many resources available that cover advanced levels of audio analysis within Unity. However, using my extensive experience in audio sciences and theory, I have been able to adapt formulae from real-life audio applications into DJ Simulator.

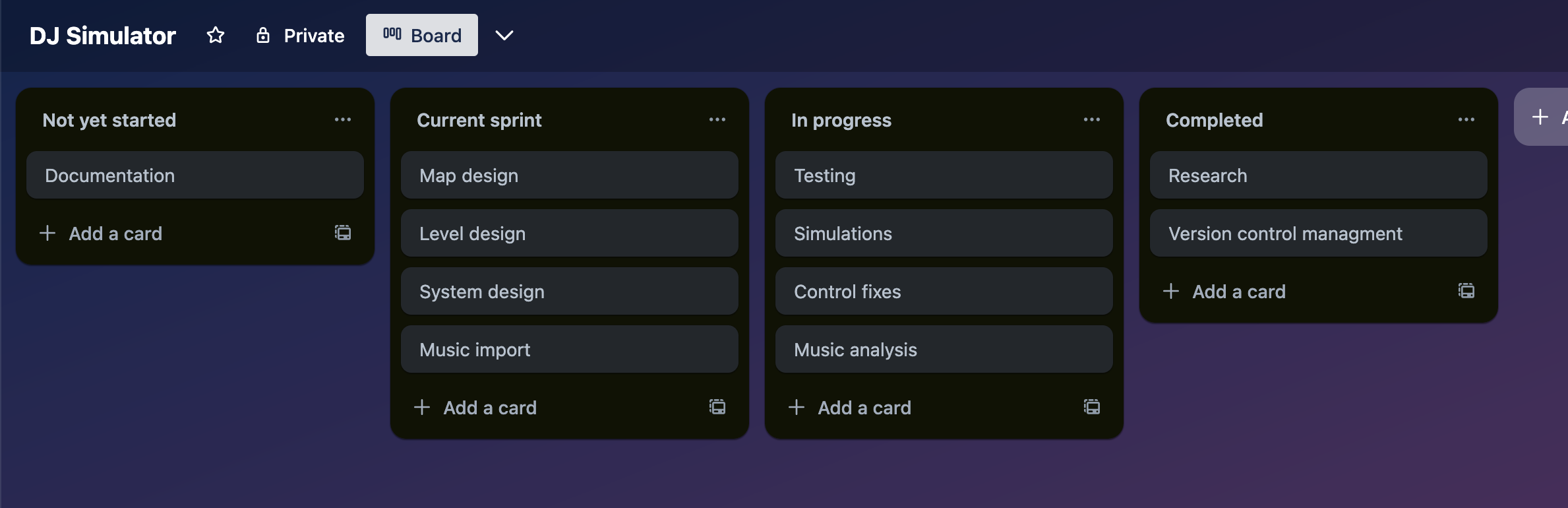
A very frequent threat for large-scale projects is the lack of time, or poor time management. This can cause serious problems to the quality of the final product. In order to mitigate this, the project has been begun early. Furthermore, using a high-end MacBook Pro laptop means that development can be made anywhere, thus providing much more opportunities to work on the project.

When working within the digital realm, data loss is always a threat. It can cause severe down-time and loss of progress. In order to mitigate this, GitHub version control has been utilised alongside a backup on two independent SSD locations. This means that should any data loss occur; a backup will be readily available.

## Development methodology analysis

As discussed earlier, two development methodologies have been considered for this project: Waterfall and Agile. It is important to consider which methodology allows for the most efficient usage of development time. Agile is the ideal development methodology for this project. This is because of the nature of this project being feedback heavy and iterative cycles being essential on this scope of work. Due to the frequent interim demos, drop-in sessions and showcases, following an Agile methodology, such as the Scrum framework is beneficial since it enables continuous improvement and encourages rapid and flexible response to change [17]. Considering the experimental and research nature of this project, going for a static approach like Waterfall would result in time being wasted, and features not being implemented based on feedback. There are many situations where the static nature of the Waterfall methodology would result in missed opportunities in improvement and feedback [18].

One of the leading Agile-based frameworks that is favoured by game developers [18] is the Scrum framework. While this is a team-based framework, it can be adapted for solo development [19] by adapting the roles to be fulfilled by one person. This modified solo Scrum methodology can then be planned out in a project management platform such as Microsoft Project, Monday or Trello. In this case, Trello has been used due to its easy and free online application. Microsoft Project is not available on macOS, and on Windows requires extensive setup. Monday does not have a free package that is useful for this project.



*Figure 5 – an extract of the solo Scrum Agile project plan via Trello.*

Now that the project contains a clear Scrum vision mapped out on Trello, the iterative development can begin. Feedback can be collected and improvements can be made. When research breakthroughs are made, Scrums can be organised, and the findings can then be added into the future development cycles, and then implemented into the project.

# Conceptual Game Design

## Product overview

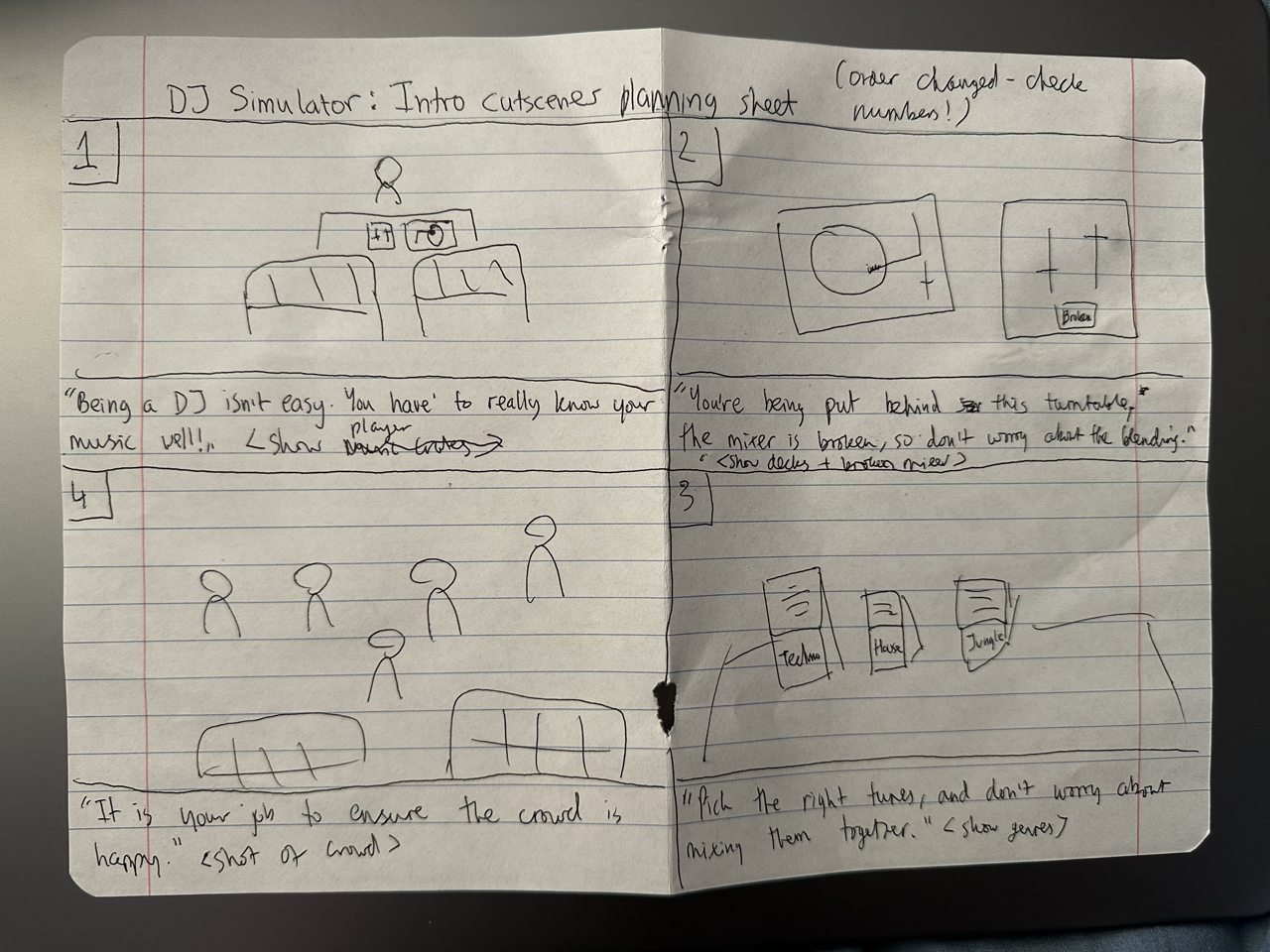
The project is titled DJ Simulator as a homage to the vast array of simulation games, such as Microsoft Flight Simulator or Goat Simulator. The target age is PEGI 7. No themes of violence, gore or foul language are expected to be present in the game. The reason for a rating of ages 7 or higher is due to the depiction of nightlife themes. Being able to keep the age rating low enables the product to be accessible to a wider audience. Being a PEGI 7 game gives young kids/teenagers who are too young to experience nightlife and venue DJing a chance to immerse themselves into the DJing niche. There is no necessity for any adult content, and steps were taken in order to keep it this way, such as a lack of profanity within music, the avoidance of suggestive portrayal of in-game characters, and the omittance of any alcohol or drug related stereotypes that are frequently associated with nightclubs and nightlife. Omitting these controversial topics of nightlife has been done in such a way that does not hinder the immersion of the game, yet enables the age rating to be kept as low as possible.

## Game outline



*Figure 6 – a simple diagram explaining how to succeed at the game.*

The player is spawned into the club, and the club is then populated with a randomly generated crowd, consisting of colour-coded NPCs that enjoy House, Techno or Jungle music. The player then has 5 minutes to play music that aligns with the genre that will satisfy the crowd the most.



*Figure 7 – storyboard planning paper.*

Due to the near-sandbox simulator qualities of the game, no major story has been implemented into the game. A simple, 4 frame cutscene plays at the start of the game in order to provide context and purpose to the player. Other than that, it is assumed that the player already exhibits a level of interest into the DJing niche, and no further gripping story enticement is necessary. After the brief context-setting cutscene introduction, the player is thrown straight into the game. Considering how simple the story is, a simple storyboard has been designed on paper in a 4-panel format, since this illustrated the vision in the most direct and detailed way possible.

## Gameplay



*Figure 8 – early concept diagram.*

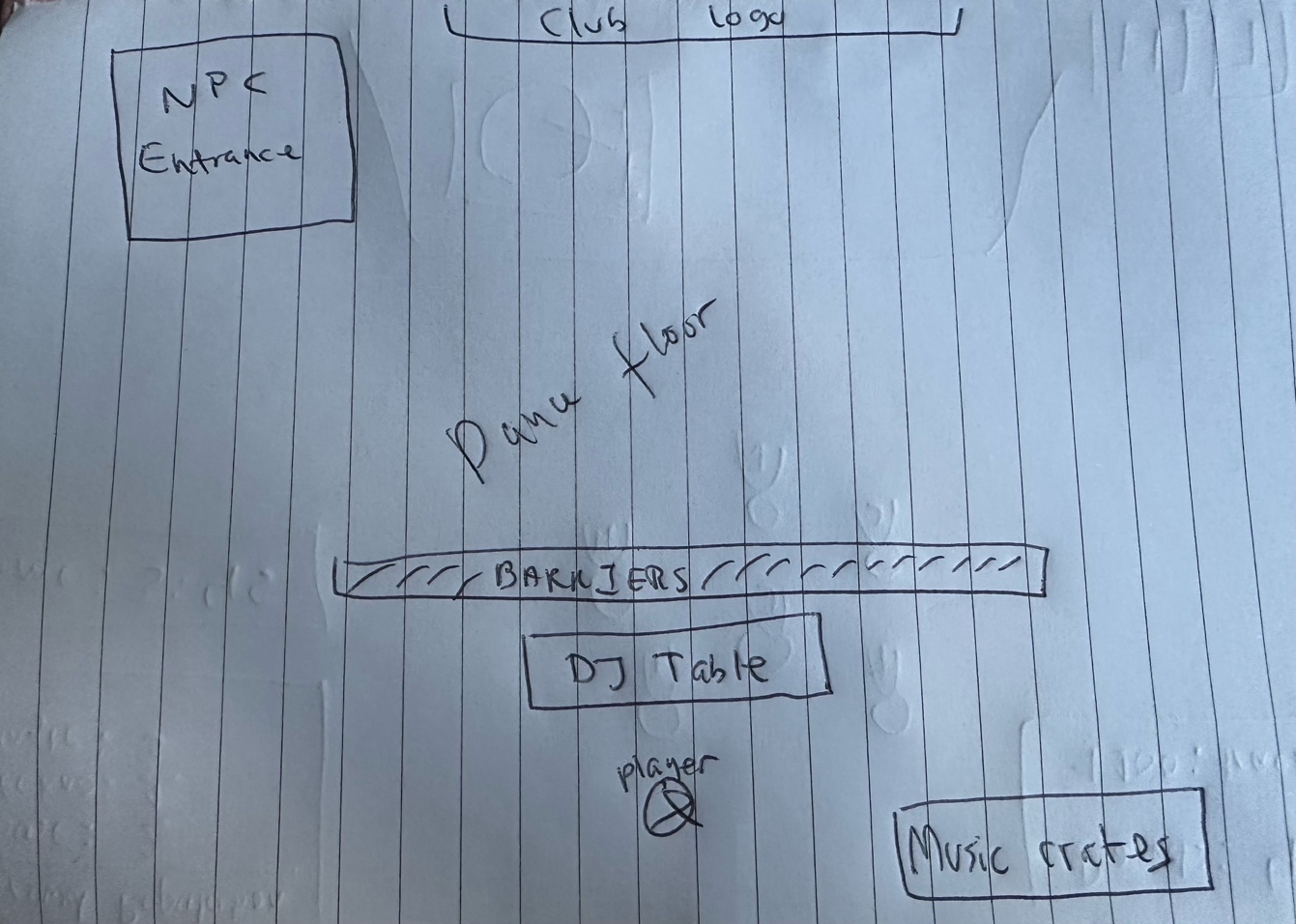
The player’s view throughout the game will be of the DJ decks and of the crowd. The crowd will be colour coded based on the music that they like. The tracks that are being loaded onto the decks will also be colour coded based on their genre. The aim of the game is to score as many points as possible during a timed DJ set. Points are awarded by playing music that the crowd enjoys (see *Figure 6.*).

There is only one, time-limited round. However, based on feedback, a few different play modes have been implemented. The main game mode (which will be mostly the focus within this report) shows a tally of the current crowd population, as well as the score list. The hard game mode is more realistic since it removes the tally of the crowd population. This is more realistic with real-life DJing scenarios, where the DJ would not have such a tool, and would be required to analyse the crowd manually. Finally, the free-play mode removes the time limitation, so that the player can practise, explore and play without competitive game objective constraints.

When spawned in, the player must analyse the crowd and then select music from one of the three music genre crates. Once selected, the player must then load the track onto the deck, play it, and observe the crowd’s reaction and the feedback provided by the points list. Occasionally, to encourage adaptability and creativity, the crowd will reset. The player must then repeat the process. When the timer ends, the player will be provided with their score, showcasing how well the player managed to impress the crowd.

The game contains no enemies, unless the timer can be considered one. There are no hazards or power-ups. These features are normally reserved for action/adventure games, and most simulator games do not require such features. For example, Microsoft’s hugely successful Flight Simulator series, which can be considered a benchmark for simulation games, does contain any of these features for the sake of realism.

## Level design



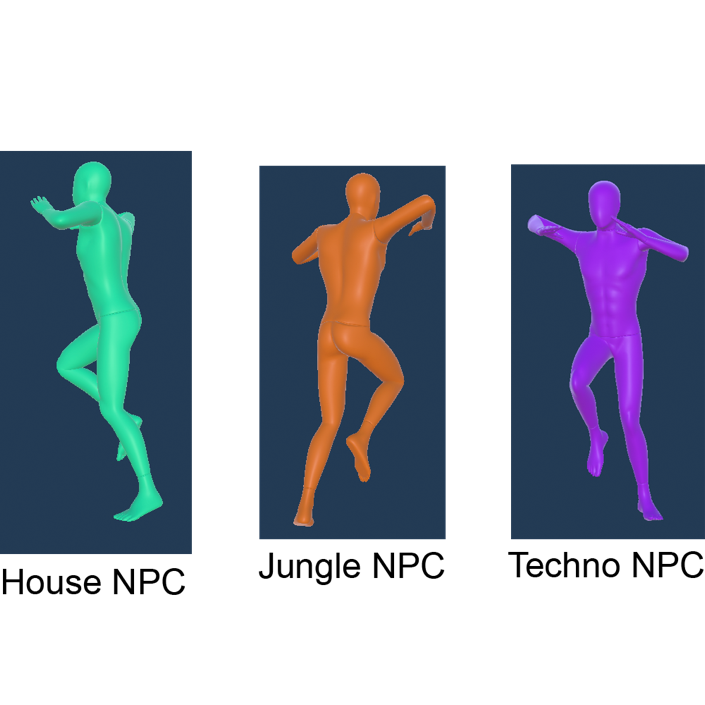
*Figure 9 – rough sketch of the map.*

The game world is designed to model a run-down, low-budget nightclub, which is in line with the experiences most beginner DJs have at venues [7]. The choice of minimal level design solidifies the low-budget, run-down nature of the club, and ensures that the player is not distracted from analysing the crowd. As expected, based on the contextual initialisation the player receives from the introductory cutscene, the player will find their DJ decks along with the crowd in front of them, and the music crates next to them. The player can also see the entrance, where NPCs will enter and join the crowd.

All of the DJ tools are designed in a way to be close, visible and accessible to the player as to not hinder the player’s performance within a timed setting. Likewise, the crowd is made to be strongly spaced out from any other objects on the map, so that the player can easily monitor and assess the crowd.

Club logos and decorations are played occasionally around the map in order to appeal to the low-budget aesthetic of the game world. It gently enhances the detail while not distracting the player from the action. Similarly, barriers are placed to ensure that the player cannot free roam beyond the intended map boundaries, and because barriers typically appeal to the mise-en-scene of most nightclubs. These features ensure that the level is functional, immersive but not too bust or cluttered. There is also a rotating disco ball present, which also increases immersion due to being typical nightclub furniture and is used as a diffusion surface for the club’s lighting system.

## Character design

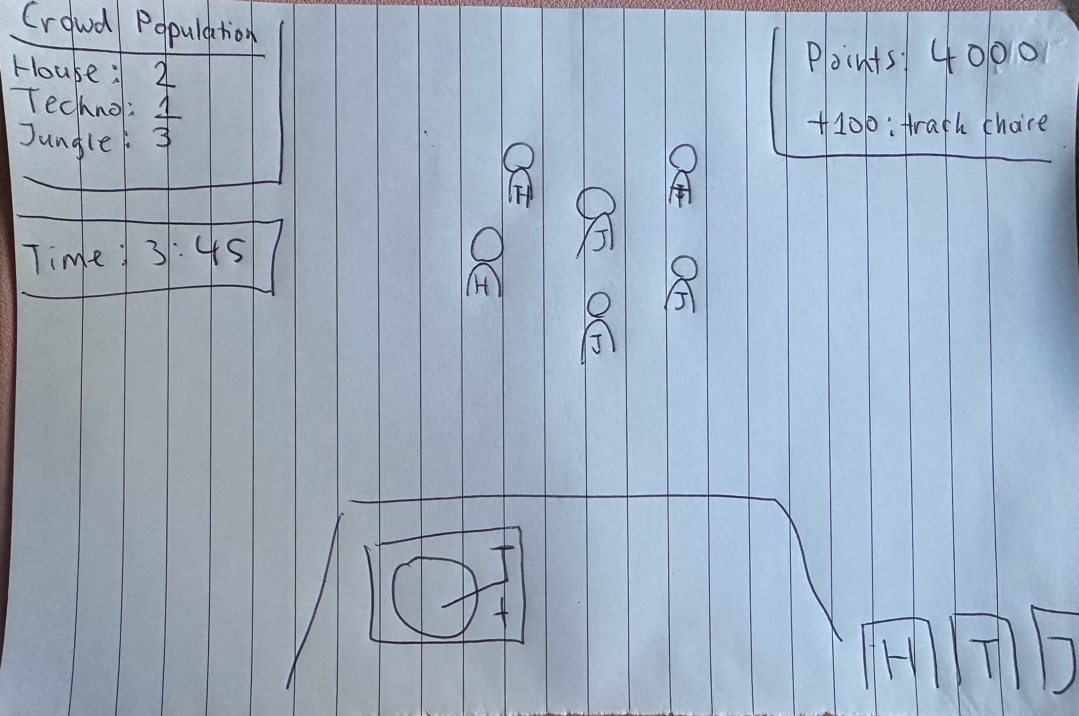


*Figure 10 – design of in-game NPCs.*

As with many other design elements within the game, the NPCs are designed in a way that is functional, but not distracting. Utilising ambiguous models ensures no provocative or offensive imagery, and also promotes inclusivity by not alienating in-game characters to any specific demographic. The NPCs are colour coded based on the genre they enjoy the most. Being such a simple design means that the player can instantly see what genre the NPC enjoys. This is an example of efficient and functional visual cues.

The player never will see their own player model due to the first-person perspective of the game. Therefore, to not introduce distractions, no visible player modelling exists within the game, and even shadows for the player are removed.

## Graphical User Interface design



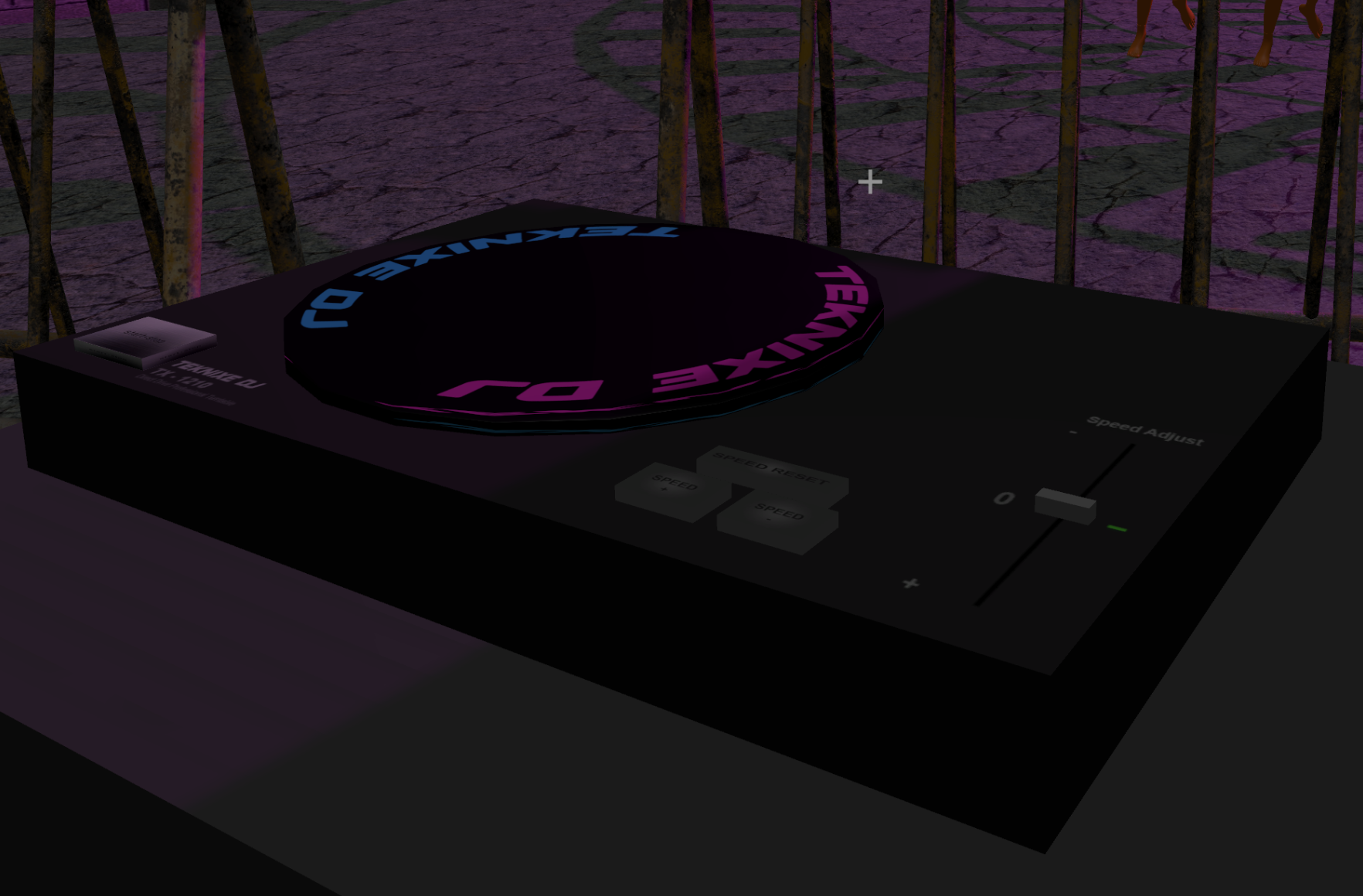
*Figure 11 – rough plan of visual cues.*

DJ Simulator must be able to provide a large amount of data in visual cues for the player. A sketch has been made purposefully for how the GUI can be used to provide and assist with visual cues *(see Figure 11)*.

The Points List is there to show the player what the game has rewarded the player for. It serves as a points breakdown and feedback system informing the player of any mistakes that they may be making that can cost them points. This is a very useful performance indicator and teaching tool showcasing how the player can improve and gain more points.

The Crowd Population tally (not available in hard mode) is an assist for the player, designed to help navigate the crowd. Dependant on the situation, the crowd may sometimes be very tightly packed together, which can make it difficult to assess the situation. It also serves its purpose from an accessibility perspective, allowing colour-blind players to use the Crowd Population tally instead of relying on colours they cannot see.

## Meeting requirements via design



*Figure 11 – in-game prototype of the DJ decks.*

As covered by the analysis of Vinyl Reality, it is important that the systems modelled within DJ Simulator follow the niche conventions of DJing. Hence, many DJ system design choices have followed the conventions set by industry standard hardware, such as the in-game decks which resemble the Technics SL-1210. This meets the goal of simulating DJ hardware in accordance with industry standards.

Straight forward visual cues are an important requirement identified within earlier analysis. As prior discussed, most design elements are made in a way that has enough fidelity to convey the run-down, low-budget nature of the environment, however not in excess as to cause distraction and confusion for the player seeking visual cues.

To conclude, it is important that DJ Simulator has clear and unobstructed visual cues. The player cannot be distracted by unnecessary visual elements, since this would ruin immersion and cause a frustrating artificial difficulty spike. To ensure that is not the case, DJ Simulator employs a near-minimalist design language, where no unnecessary detail is present.

# Technical Game Design

# 

# Implementation

# 

# Testing and Evaluation

# 

# Legal, Social, Security and Ethical Issues

# 

# Critical Review and Conclusion

# 

# Technical Appendices

# References

[1] Ler digital studio (2025). Beat Saber - VR rhythm game. [online] Beatsaber.com. Available at: https://beatsaber.com/documentation/environment-specific-lighting-attributes-and-explanations/index.html [Accessed 22 Apr. 2025].

[2] Sunil Khobragade (2024). Bringing Stories to Life in Casual Gaming Experiences [online] Linkedin.com. Available at: https://www.linkedin.com/pulse/visual-storytelling-casual-game-development-sunil-khobragade-xjvhf/ [Accessed 22 Apr. 2025].

[3] Venue Lights. (2024). Nightclub Lighting | Venue Lights. [online] Available at: https://venuelights.co.uk/lights/venue-lighting/nightclub-lighting/ [Accessed 22 Apr. 2025].

[4] Vinyl Reality. (2021). Vinyl Reality - DJ in VR. [online] Available at: https://www.vinyl-reality.com/ [Accessed 22 Apr. 2025].

[5] Vinyl-reality.com. (2019). audio\_analysis [Vinyl Reality Wiki]. [online] Available at: https://vinyl-reality.com/wiki/doku.php?id=audio\_analysis [Accessed 22 Apr. 2025].

[6] Technics (n.d.). SL-1200 ‒ Heritage - Technics. [online] www.technics.com. Available at: <https://www.technics.com/global/home/sl1200/heritage.html>.

[7] www.djgym.co.uk. (n.d.). How Do DJs Get Their First DJ Gig: A Guide For New DJs. [online] Available at: https://www.djgym.co.uk/post/how-do-djs-get-their-first-dj-gig-a-guide-for-new-djs.

[8] Simoes, R. (2024). A Complete Guide To The DJ Club Standard Setup For The Industry In 2024 - If You’re In Search Of DJ Gigs, Here’s What You Should Know - Learningtodj.com. [online] Learning to DJ. Available at: <https://learningtodj.com/blog/the-club-standard-dj-setup-in-2024/>.

[9] Fandom.com. (2025). GTA Online Nightclubs. [online] Available at: https://gta.fandom.com/wiki/Nightclubs [Accessed 22 Apr. 2025].

[10] Unity Technologies (2024). Unity - Scripting API: MonoBehaviour. [online] Unity3d.com. Available at: <https://docs.unity3d.com/6000.0/Documentation/ScriptReference/MonoBehaviour.html>.

[11] Epic Games Developer. (2025). Designing Visuals, Rendering, and Graphics | Unreal Engine 4.27 Documentation | Epic Developer Community. [online] Available at: <https://dev.epicgames.com/documentation/en-us/unreal-engine/designing-visuals-rendering-and-graphics?application_version=4.27>.

[12] Unreal Engine. (2024). Animation. [online] Available at: <https://www.unrealengine.com/en-US/uses/animation>.

[13] Unity Technologies (2025). Unity - Scripting API: AudioSource.GetSpectrumData. [online] Unity3d.com. Available at: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/AudioSource.GetSpectrumData.html [Accessed 22 Apr. 2025].

[14] Unity Technologies (2025). Unity - Manual: Animation State Machines. [online] docs.unity3d.com. Available at: <https://docs.unity3d.com/Manual/AnimationStateMachines.html>.

[15] Raph Koster (2005). A theory of fun for game design. Scottsdale, Az: Paraglyph Press.

‌[16] ProductPlan (2023). Agile vs. Waterfall | Pros, Cons, and Key Differences. [online] www.productplan.com. Available at: https://www.productplan.com/learn/agile-vs-waterfall/.

‌[17] Wikipedia Contributors (2018). Agile software development. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Agile\_software\_development.

[18] Starloop Studios. (n.d.). Best Agile Practices in Game Development | Starloop Studios Blog. [online] Available at: https://starloopstudios.com/best-agile-practices-in-game-development/.

‌[19] Lucidchart (2017). Scrum for One: Adapting Agile Scrum Methodology for Individuals. [online] Lucidchart. Available at: https://www.lucidchart.com/blog/scrum-for-one.