Building a Game

Project Plan

Romina Petrozzi

CS3810 - BSc Final Year Project

Supervised By: Julien Lange

Department of Computer Science
Royal Holloway, University of London

Contents

1 Abstract

The video game industry has grown into one of the largest sectors in the entertainment industry, with video game sales being about five times higher than global music revenues, higher than consumer book sales, and similar to movie revenues (Marchand A, Hennig-Thurau T, 2013). This growth has been driven by many factors, including the growth of software, improvement of internet technology, and innovation (Zackariasson, P, Wilson T. 2012) . Video games have evolved significantly over the years, progressing from simple 2D graphics to immersive 3D experiences, and with the oncoming development of virtual reality (VR) and artificial intelligence (AI) technology, they are set to become even more immersive and engaging. My motivation for this project stems from my interest in the VR industry. By working on this game with Unity 3D, I will gain experience using one of the most widely used engines for VR development and practice working with 3D environments. These skills will transition well into VR development, providing a good foundation for future projects in that field.

The game I am developing is a 3D platformer set in a fantasy forest, where the player controls an axe-wielding hero tasked with rescuing trapped animals guarded by goblin enemies. There will be a set number of levels, each with increasing difficulty that the player must traverse through. They will platform across obstacles by jumping over gaps, avoiding hazards, and fighting goblins. Points are awarded based on the number of enemies defeated in a level, and to clear each level the player must rescue all the animals being guarded. In addition, there will be extra collectables hidden throughout the levels which will grant the player bonus health and temporary power ups. This level design aims to cater to both "segments of players labelled as 'experiencers' versus 'achievers"' (Yi Z. et al. 2022) by offering contained, linear levels which also encourage exploration. A highly successful example of this design is Super Mario 3D World, which my level designs will take inspiration from.

For the development of this game, I have chosen to use Unity game engine. 'Game engines are platforms that make it easier to create computer games. They allow you to integrate and combine into single unit individual game elements such as animations, interaction with the user, or detection of collisions between objects' (Barczak M, Woźniak H, 2019). There are many game engines available for 3D development, with notable examples including Unity, Unreal Engine, and CryEngine. As Barczak and Wozniak (2019) explain, these engines provide reusable components, allowing developers to focus on gameplay and design rather than redeveloping fundamental systems. Unity stands out as the best choice for my project due to its powerful capabilities, user-friendly interface, and well documented resources. According to

Christopoulou and Xinogalos (2017), Unreal Engine 4 and Unity are the two most developed engines, with Unreal Engine being more suited to experienced users providing remarkable graphics, while Unity is more suited to beginners with a large asset library and simpler user interface.

Optimisation techniques are crucial in game development, as performance lag can significantly impact the player's gameplay experience in many game genres. In particular, 3D platformers require precise movement and jumps for navigating levels and obstacles, and experiencing frame drops can make the gameplay frustrating for the player. Some of the main optimisation techniques used for games are level of detail (LOD) management, dynamic batching, occlusion culling (OC) and shader optimisation. The techniques I plan to use for my game are OC and LOD management. Occlusion culling 'is the mechanism by which Unity avoids rendering computations for GameObjects that are fully hidden from view by other GameObjects' (Singh, B. Sharma and A. Sharma, 2022). By not rendering objects outside of the player's view, performance is significantly improved as resources are not wasted on hidden elements. Unity has a built in function for this which I will make use of. LOD management adjusts the level of detail of an object depending on how far it is from the camera. In Unity, this is done using LOD levels, which specify the amount of detail of an object's geometry to be rendered based on the object's distance from the camera (Unity Technologies, 2024). By rendering fewer polygons for distant objects, the GPU workload is decreased, freeing up resources for closer objects.

Design patterns offer solutions to common problems faced by software developers. In game development, many common design issues have been solved through specific patterns that are widely used across many types of games. The use of these patterns helps maintain clean, readable code, making it easier to expand the game without disrupting existing functionality. Two common patterns that I will use in my game are State and Observer. The state pattern addresses player/object states for example, walking, idle and running and encapsulates each as an object. This pattern solves two problems: an object should change it's behaviour when its internal state changes, and adding new states does not impact the behaviour of existing states (Unity Technologies, 2022). Within my game I will use this pattern for the player's animation and movement transitions, as well as basic enemy AI states. The observer pattern is used when objects need to be notified without explicitly referencing them. This will be required when multiple different game objects need to be notified about events such as collecting a power up or defeating an enemy. The pattern involves observer and subject classes, where the observer contains a method that defines what action to take when notified by the subject, while the subject keeps a list of observers and notifies them when a specific event occurs (Nystrom R, 2014). This decouples the objects, allowing the subject to notify multiple observers without explicitly referencing them.

2 Timeline

The project timeline is divided into two terms. Term 1 focuses on developing and refining the core game mechanics, while Term 2 is dedicated to expanding features and optimising performance.

2.1 Term 1

Week 1:	— • Learning and understanding the fundamentals of Unity game engine
Week 2:	— • Finalise the core game concept and gather all assets
Week 3-4:	— • Prototyping: Animations and movement for player and enemies
Week 5-6:	— • Prototyping: Enemy interactions and health mechanics
	— • **Initial player and enemy movement prototype working by Week 7**
Week 7-8:	— • Design and User Interface: First level design and creating base user interface screens
Week 9:	— • Initial play-testing: Bug fixing and gathering feedback from users to refine core gameplay
	— • **Level one finished by Week 10**
Week 10-11:	— • Work on interim report and presentation

2.2 Term 2

Week 1-3:	— • Feature Expansion: Multiple levels, score system, weapon upgrades
Week 4-5:	— • Audio and sound effects implementation
Week 6-7:	— • Improvements to current graphics and game optimisation
Week 8:	— • Improving and expanding on current mechanics
Week 9-10:	— • Final play-testing : Bug fixing and gathering feedback from users to refine style and gameplay
	— • **Final game with expanded features finished by Week 10**
Week 10-11:	— • Work on final report

3 Risks and Mitigations

Given the substantial scope of this game, there are several risks associated with its progress and completion. In this section, I will discuss these risks and outline my strategies for mitigating them, along with risk level (Low, Medium, High).

3.1 Asset Acquisition - *Low*

Creating a 3D game requires numerous 3D models and assets. As I lack experience creating 3D models and animations, doing this myself would require significant amounts of time and present a very steep learning curve. This creates a significant risk as the quality and visuals of assets will impact the game's graphic consistency and playability. To mitigate this risk, I will set dedicated time in the project for asset gathering. Prioritising the use of well-documented and optimised assets to ensure they integrate smoothly into the game.

3.2 Learning New Tools - *Medium*

As I have limited experience using Unity engine, learning this new software with its many capabilities could take a significant amount of time, which could impact the rate of game development. To mitigate this, I will make use of the vast amount of tutorials that exist, and dedicate some time early in the project to learn the fundamentals.

3.3 Time Management - *Medium*

There is a risk of falling behind schedule on both the project timeline and report writing. This is due to the fact that I have commitment to part time work and expect to spend most of my time working on the project code rather than the report. I will mitigate this by breaking down each timeline block into smaller, manageable tasks and prioritise the most important tasks. Furthermore, to make sure that I have sufficient time to complete the report, I will allocate time throughout the timeline rather than only at the end of each term to begin working on it gradually.

3.4 Hardware Failure - *Low*

There is risk of the hardware I am using to work on the project to fail, such as a malfunctioning laptop or data loss, which could result in lost work. To mitigate this, I will use Git version control system (VCS) to regularly back up my code and related files. This means that if my hardware fails I will be able to recover the project from my most recent commit.

3.5 Hardware Constraints - *High*

3D games generally require certain hardware specifications to run smoothly and avoid performance issues. As I will primarily be working on this project on a laptop, I may encounter performance issues if my device does not meet these specifications. To mitigate this, I will research optimisation strategies as challenges arise and consider using a more powerful machine when available.

3.6 Scope - *Medium*

The ambitious scope of my project may prevent me from completing the game within the given time frame. Given the complexity of developing a 3D game while simultaneously learning a new game engine at the same time could mean that I cannot achieve the desired features and visual quality. To mitigate this risk, I will focus on establishing a minimum viable product (MVP) for the game by deciding on core, achievable aspects of the game. With this approach I can then build on these essential features as time allows.

4 Acronyms

- VR: Virtual Reality.
- AI: Artificial Intelligence.
- VCS: Version Control System.
- MVP: Minimum Viable Product.
- OC: Occlusion Culling
- LOD: Level Of Detail
- API: Application Programming Interface
- LFS: Large File System

5 Bibliography

5.1 Cited References

1. Marchand A, Hennig-Thurau T. (2013) Value Creation in the Video Game Industry: Industry Economics, Consumer Benefits, and Research Opportunities. *Journal of Interactive Marketing*, 27(3), 141-157. - This resource, along with

(Zackariasson, P, Wilson T. 2012) have provided me with more background on the history and economics of the game industry.

2. Zackariasson, P. and Wilson, T.L. (2012) *The video game industry: Formation, Present State, and future.* New York: Routledge.

3. Yi Zhao. *et al.* (2022) A Dynamic Model of Player Level-Progression Decisions in Online Gaming. *Management Science* 68(11):8062-8082. - This paper has helped me understand player level-progression which can be applied to the level design of my game.

4. Barczak Andrzej Marian, Woźniak Hubert. (2019) Comparative study on game engines. *Studia Informatica. Systems and Information Technology*, no. 1-2, pp. 5-24. - This source, along with (Christopoulou and Xinogalos 2017) provided me with insights into selecting the most appropriate game engine for this project.

5. Christopoulou, E, Xinogalos S. (2017) Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices. *International Journal of Serious Games*, 4(4).

6. N. P. Singh, B. Sharma and A. Sharma. (2022) Performance Analysis and Optimization Techniques in Unity 3D. *3rd International Conference on Smart Electronics and Communication (ICOSEC)*, Trichy, India, pp. 245-252. - This resource gave an overview of the main optimisation techniques for 3D games and how they are implemented in Unity engine specifically. I will refer to this when optimising my game.

7. Unity Technologies (2024) Level of Detail (LOD) for meshes. Available at: https://docs.unity3d.com/Manual/LevelOfDetail.html (Accessed: 10 October 2024). - This resource refers to the official Unity documentation on controlling LOD for game objects. I will use this when implementing LOD optimisation.

8. Unity Technologies. (2022) *Level up your code with game programming patterns.* Available at: https://unity.com/resources/level-up-your-code-with-game-programming-patterns (Accessed: 10 October 2024) This book, along with Nystrom R, (2011), provides an overview of key design patterns in game development, with this source specifically detailing their implementation in Unity C# scripts. I will reference this when implementing design patterns in my game.

9. Nystrom, R. (2011) *Game Programming Patterns.* gb.

5.2 Further Research

1. Unity Technologies. (2022) Unity User Manual 2022.3 (LTS). Available at: https://docs.unity3d.com/Manual/index.html (Accessed: 10 October 2024). - I will refer to the official Unity documentation for help with the Unity editor and Scripting API.

2. Petterson, T. (2016) The complete guide to Unity & Git. Available at: https://www.gamedeveloper.com/programming/the-complete-guide-to-unity-git#unity_gitignore. (Accessed: 7 October 2024). - This website helped me set up my project to be properly managed with Git with a Git Large File System (LFS) and appropriate .gitignore file.

3. Unity Technologies. (2022) Best practices for organizing your Unity project. Available at: https://unity.com/how-to/organizing-your-project (Accessed: 7 October 2024). - I will refer to this article when trying to maintain a clear and consistent file structure for my project.