Top Down RogueLike Survival Shoot’em Up based Video Game

Project Plan

Alin Gabriel Mican

Full Unit BSC Final Year Project

Supervised By: Adrian Johnstone
Department of Computer Science
Royal holloway, University of London

1 Abstract

First developed in the 1960s, video games have provided a path to escape reality for hours on end. An industry that is worth over 138 billion dollars and growing [1], providing people with entertainment, but also improving hand to eye coordination, problem solving skills, accuracy with decision making skills and more [2]. So why not develop a game where you can have endless fun with and on top improve your cognitive skills?

To begin developing and deploying the game we need to establish a methodology plan, to test releases of the game. Agile methodology is what I will be using to develop and test the progress of my game. Using prototypes to test various game mechanics, allows me to evaluate the proper outcomes for what I would like the game to produce, by playing simulations of different input values, such as prototyping with weapon recoil. Furthermore, an agile methodology will allow me to quickly backtrack if a change needs to be made, unlike waterfall methodology. Since planning/ ideas can change over the time, this will be beneficial, especially when designing games. Lastly, the concept of using divide and rule, will allow me to develop integrations in fast successions to produce and test playable aspects of the game, working fast and effectively [3].

For this project, I will be working on a 2 dimensional angled top down view video game. As the title says the genre of this game will be a top down rogueLike survival shoot'em up type video game. The reason I choose this genre is because of the idea of replayability and satisfaction. It is a wave based shooter against hordes of zombies whilst the player will try to collect various weapons to survive as long as possible. The artstyle will be a simplistic 64x64 pixel art, for various reasons such as being able to run the game more easily as the graphics are not as intense (less GPU work!) and because of efficiency, being able to produce sprites at a reasonable pace.

The main goal of the game is to survive as long as you can, collect experience to level up, every level increment, the user will be prompted with a menu screen that will allow them to choose one ability/buff or a weapon. This idea of getting different weapons and buffs adds replayability and if done right with items, it can add satisfaction. With the in-game currency you can also unlock two other characters (subject to change) that have different buffs, in addition you can unlock more powerful weapons to survive for a longer time; powerful weapons will have a chance to pop up when the player levels up in-game.

1.1 Game Mechanics

- Zombies: - There will be different types of zombies, including: ranged, melee, fast, tough, mini-bosses and bosses and using the shortest path to find the player.
- Wave System: - The wave system will include a timer, every 2.5 minutes an enhanced version of an enemy will be spawned. Every 5 minutes an event will happen such as more enemies spawned, or mini bosses spawned. The more the player survives; the spawn rate increases, an increase in enemy health, and harder to gain experience.
- Player controller: - The player will be using “WASD” movement , aiming and shooting with the mouse and changing weapons with number keys.
- Leveling System: - Every time the player starts the game they are reset to level zero, enemy kills give them experience, once their level has incremented, game pauses and the user is prompted with a screen that allows a random selection of a buff or a weapon.
- Currency System: - Player earns currency from playing the game. Killing bosses earns them currency and depending how long they survive a multiplier is added to their total “currency” collected at the end of the game. The currency allows the player to unlock different characters/ weapons and buff upgrades.
- Pick ups: - Enhanced enemies, or bosses have the chance of dropping a perk such as, removing every enemy around a certain radius, to pick up the item the player all it has to do is go over the proposed item and it will activate.
- Storage System: - This system will include saving highscore, saving user data such as their characters unlocked, currency and unlocked weapons.

1.2 Algorithms

Throughout the project I will be exploring different complex algorithms that will help with game performance during run time and user experience.

- Object Pooling : - In the context of spawning lots of different types of enemies, we need some type of algorithm to optimize CPU time. The main idea is that we will instantiate most objects at the beginning of the, so that once in run time, we can “pull” those objects and spawn them in at a different coordinate. I will be exploring this idea, finding a balance between loading game time and runtime in game performance [4].
- 2D Line of Sight or shadow casting :- This algorithm uses the idea of hiding things from a player in a 2D plain. Using raycasting to find all points visible to the player, we form a map of what is visible. Further taking this, I also want to explore occlusion culling, where the idea is to improve performance by doing less rendering calculations on what is not seen. Why should the whole map be rendered when only a subsection that can be seen can be rendered? Using this idea of visibility we can render in what can be seen by the

player, contiguously, however if there isn't a lot of performance difference, I still plan on keeping this feature for user experience[5].

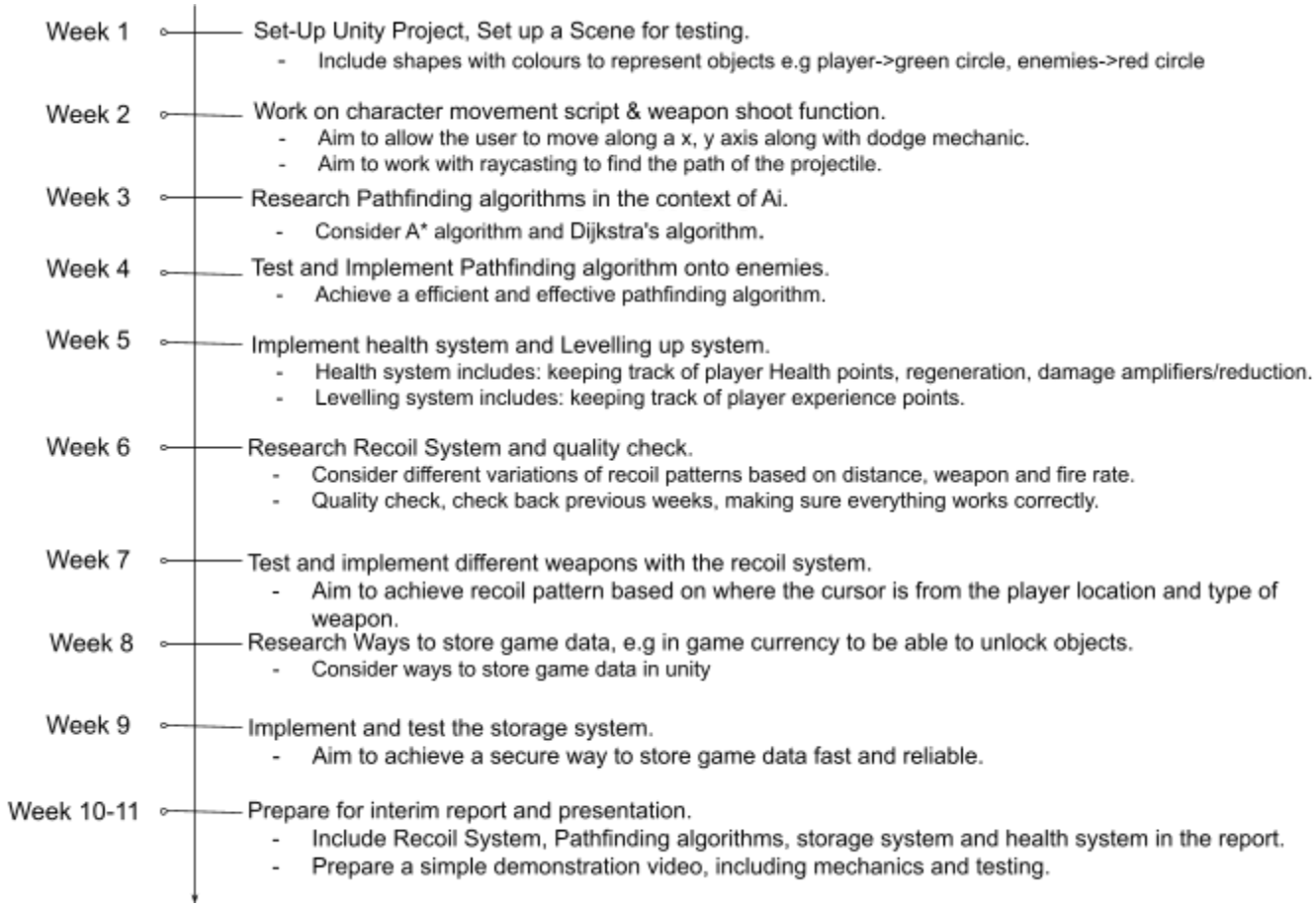
- Recoil System: - In the context of user experience I want to make the game challenging for the user. Idea is to track the player position and cursor position, the further the cursor position is from the player position the bigger the recoil the spread is, and the more they spam shoot function the bigger the bullet spread is or type of weapon the user is currently holding.
- Ai PathFinding Algorithm- For the enemies to find the position of the player, I will be using a search algorithm to optimally find the player's location whilst avoiding obstacles. I will be exploring different approaches and path finding approaches that are optimized well for performance [6].
- Map System: - The map will be designated into five zones, the player will start in the first zone, keeping track of the player position and zones location and time. When the time reaches 5 minutes, enemies from the second zone will start to come towards the player, 10 minutes enemies from the third zone will start to come, etc. However the player can challenge themselves and go to whatever zone they want before that certain time is met. For example the player could go to zone 3 at minute 2, however it could be very challenging since now you have enemies that were meant to spawn at minute 10 coming towards you and zone 1 enemies also coming, not giving the player enough levels to be able to combat these enemies, becoming more of a dodging game.

With this project I aim to achieve a fun and enjoyable experience, whilst keeping a competitive aspect to try to aim for a high score. In addition, having unlockables will keep the user hooked to try out new mechanics and allow them to further progress into the game, giving them higher scores.

2 Timeline

This will be the plan I will be carrying out throughout my project lifetime. Implementing most of the core mechanics in the first term, and in the second term focusing more on game algorithms.

2.1 Term 1



2.2 Term 2

Week 1
Research 2D line of sight/ Shadow Casting algorithms.
- Consider this algorithm with occlusion culling and consider using it with AI behaviours.
- Consider this algorithm with movable objects.

Week 2
Test and implement 2D line of sight algorithm.
- Test the performance when multiple shadows are casted at once.

Week 3
Work on GUI system and graphics. E.g Camera shake/ Particles.
- Aim to get a working menu screen
- Camera shake for various actions such as shooting and explosions.
- Produce particles based on object impacts.

Week 4
Research Object Pooling.
- Consider using object pooling with particles and enemies

Week 5
Test and implement Object Pooling.
- Aim to achieve a performance boost using object pooling design pattern.

Week 6
Work on enemy system.
- Consider different enemy types e.g ranged, melee etc.
- Consider enemies with different buffs.

Week 7
Work on the wave system.
- Implement a timer, that will increase enemy spawn rate, health, speed, types of enemies that may spawn, making the game more challenging as we progress further into the game.

Week 8
Research a way to implement the map system.
- Consider an efficient system.
- Consider using tags to keep track of what zones the player has visited.

Week 9
Work on the map system and test several functions.

Week 10-11
Prepare for final report and demonstration.
- Include algorithms I have been working on.
- Ideas and concepts explored
- Implement last minute changes if necessary

3 Risk Mitigations

Throughout the lifetime of a project there are bound to be risks and unavoidable errors. Since there are some enthusiastic tasks, they can come with their own risks. In this section, a discussion about plausible and general risks and mitigations that can happen.

3.1 Performance issues

A risk that is always involved in game design is performance. How could we achieve the best possible optimal performance whilst handling lots of data? There is also the idea of doing too much work, for a little performance boost. To mitigate this I will be using various algorithm techniques and testing to find the best possible solution that could lead to an enjoyable experience. Having said so, it is important to work on algorithms that will have a low algorithmic complexity.

3.2 Poor Estimation of Tasks

As with all projects, time management is a crucial step for success. With the concept of game development, there will always be tasks that will take more time than others to complete, so finding the right balance when to carry out what task when and for how long it is an essential step. To mitigate this all tasks should be divided into subproblems and with research, I can gain an idea of what is to be expected.

3.3 Prototyping (Wasting)

As with any game development prototyping is a core aspect of testing and creating. However an issue can arise when there is a lot of time and resources that can be wasted/spent on prototyping mechanics that will never see the light of day, further delaying other tasks that need to be completed. To keep prototyping at a minimum research and assurance will go before making any prototype, so that i can keep prototypes at a minimum, avoiding unnecessary prototype making.

3.4 Data Organization

When it comes down to development of games another factor that needs to be considered is, being organized. In an environment with lots of assets, snippets of code, scenes, animations and more, it is easy to get lost and overwhelmed which can become very time inefficient, trying to find or call certain functions I wrote previously. To not allow this to happen, setting up the

project correctly and in a hierarchical manner will allow me to be organized and keep track of files/folders.

3.5 Logical Errors/ bugs

A glitch or a bug in a system is something that even big companies can't avoid in game development. These errors tend to happen because of a fault in the program logic or the structure of the problem, and it can go unnoticed since it doesn't tend to crash the program. To help reduce this, I will be using several testing methods such as black box testing, to test inputs and its expected outputs, and also have other people (mostly colleagues) test the game and find errors, mainly alpha testing.

3.6 Losing data / hardware failure

There is a possibility of losing progress due to a failure in the system, since this cannot be predicted, it is a good idea to keep a repository to avoid this data loss. Using Gitlab and Github and following good github standards, such as committing, and working in branches, it will allow me to keep a track of data and avoid unnecessary data loss.

3.7 Game Balancing

Another issue can arise when the game is not balanced. In the context of weapons, sometimes a weapon can be too strong and make the game too easy or the weapon can be too weak which results in the player dying too fast. To reach a balanced state, I will be testing and tweaking values to reach a "balanced" state and asking others for opinions as well.

REFERENCE

[1] Clement, Jessica. “Market size of the video games industry in the United States from 2010 to 2022” *in value of the video game market in the U.S,* written in May 2022. https://www.statista.com/statistics/246892/value-of-the-video-game-market-in-the-us/

[2] Adair, Cam. *Positive Effects of video games.* https://gamequitters.com/positive-effects-of-video-games/

[3] Eden,Martin. “Divide and Rule” In *Agile in Game Development.* https://meliorgames.com/game-development/agile-in-game-development/

[4] Unity Technologies. “Introduction to object pooling”, Last updated May 2022. https://learn.unity.com/tutorial/introduction-to-object-pooling#5ff8d015edbc2a002063971b

[5] redblobgames.com. "2D Visibility", Last Updated Apr 2020. https://www.redblobgames.com/articles/visibility/

[6] Kumar Pawan, Bottaci Len, Mehdi Quaism, Gough Norman, Natkin Stephane. “Efficient pathfinding for 2D games” https://wlv.openrepository.com/bitstream/handle/2436/31520/CGAIDE%202004_Kumar%20et%20al.pdf?sequence=1&isAllowed=y