Investigation into an infinite-world, resource gathering game

Contents

[**Analysis**](#_vniz7atxr9m8) **1**

[**Project Background**](#_txxebwwb5q1a) **1**

[**Description of the current system (similar games)**](#_bwlbb7wx3bx) **2**

[Terraria](#_r67ktpkdau1) 2

[Minecraft](#_i9o3r82k2why) 2

[**Identification of end user**](#_sc35r93czk4s) **2**

[Questionnaire](#_14idpwexh933) 2

[Conclusion](#_v5u2yacdp16d) 3

[**Objectives**](#_nllbu6fj0jfg) **3**

[General Objectives](#_6ecv8pfgfye4) 3

[Specific Objectives](#_qgerx79n7ba2) 4

[**Constraints and limitations**](#_5xd0bbbygjwv) **6**

[The game world](#_l2dj1orxfkjm) 6

[Game progression](#_9q1oofjd4pqc) 6

[Items/Crafting system](#_om7sgq5wiblv) 6

[**Model of the problem**](#_o768cqmfivlm) **6**

[Data flow diagram (Level 0)](#_e3hj2nf3mua5) 6

[Data flow diagram (Level 1)](#_dxwrsk2ybxlf) 7

[**Design**](#_d87qibarbvik) **7**

[**Class Diagram**](#_4zq9o24o5zsm) **7**

# Analysis

## Project Background

Over the course of the NEA project, I will be creating a game based around gathering resources, and exploring an infinitely generating world. The gameplay will consist of exploring this infinite world, collecting the resources, and crafting various weapons/tools to protect yourself against enemies.

The main problem will be generating new interesting terrain based on a seed, or loading existing serialised representations of the terrain, and doing so quickly enough that exploration of the world will look seamless to the end user. Code will also have to be written to support enemy pathfinding, crafting and inventory GUIs, and dealing with physics collisions.

## Description of the current system (similar games)

### Terraria



Terraria is an example of a game which randomly generates a 2D world based on a seed value, however the world is not infinite. The game uses a front view camera, and Terraria has utilised this well by creating an expansive underground world with various biomes, hidden structures, cave systems, etc., which gives the world more depth. The surface also contains multiple different biomes which are smoothly blended between, and also contains special structures such as the dungeon or the living tree. The player can also harvest various resources around the world and can kill enemies for various crafting materials, which they can then use to craft more powerful gear to fight stronger enemies or bosses.

### Minecraft



Minecraft generates a random infinite 3D world based on a seed value, with various biomes scattered across it. The underground has had much less interesting generation than Terraria’s, however it still has a cave system, with some biomes to create visually pleasing environments. Minecraft’s gameplay is more focused on gathering resources to create beautiful buildings and houses rather than for stronger equipment for fighting bosses, possibly because the 3D world allows for the creation of much more interesting structures and designs. Minecraft also has its own randomly generated structures, such as monster spawners, strongholds, and villages, however due to Minecraft being an infinite world, all structures are generated multiple times throughout the world, possibly making them less exciting to explore, since they’re mostly the same.

## Identification of end user

### Questionnaire

As the game will be targeted at younger audiences, I collected data using a questionnaire on my fellow classmates. For this I used the following questions to 21 students  
1) How old are you?

*All from ages 17-18*

2) What is your favourite game?

*Minecraft was most popular with 67% of the votes, followed by Fortnite with 19%*

3) Do you prefer a small game world packed with features and deep mechanics, or an expansive world with various POI scattered around, which you can explore freely?

*Small world was chosen by 14%, and large world 86%*

4) What are your least favourite aspects of a game?

*Unskippable cutscenes, long tutorials, or some other variation of a forced time-wasting event was chosen by 71% of students, with 14% choosing buggy gameplay/poor gameplay mechanics*

5) What importance would you place on the story in enjoying a game?

*(Rated on a scale 1-5, with 5 being essential, and 1 being useless)  
Had an average of 2.3, indicating story is not deemed as very useful to enjoyment of a game*

### Conclusion

From this data, I concluded that the majority of younger people who are already familiar with games prefer games that allow them to quickly get into the gameplay, and have larger worlds with the freedom to explore, rather than the restrictive but carefully constructed linear experience that comes with smaller worlds. This is also reflected in the current games industry, with most major titles being open-world or featuring a large map, such as Cyberpunk, Farcry 5, Elden Ring, and Horizon Forbidden West. Thus, this project will endeavour to create an infinite world with interesting terrain features, however it will not attempt to reach the same depth of the worlds of the previously listed games, as this would be infeasible with my current skills and time constraints.

## Objectives

# 

### General Objectives

*Game engine*

-> Abstract majority of OpenGL API

-> Batch rendering API

-> Pathfinding

-> Serialising data (POD, containers - including string, classes which inherit a streamable generic)

-> Collisions between (rotated) rectangles

-> Generating seeded noise

-> Playing sounds/sound API

-> Particle system

Game

-> Start menu

-> World map

-> Infinite world

-> Inventory system

-> Crafting system

-> Mining tiles

-> Randomly spawning animals/enemies

-> Combat system

### Specific Objectives

*Game engine*

-> Create classes to handle binding/unbinding of (and changing the data of): vertex buffer, index buffer, shader, texture, and framebuffer

-> Create a batch class which allocates an appropriate amount of memory for the amount of shapes to be batched when constructed, taking into account the layout of the vertex buffer. Class should have multiple methods for setting the data for each shape in different formats

-> Pathfinding function should implement the A\* algorithm, and should provide ADTs for the final path generated, and a map representing the walkable/unwalkable areas of the world

-> Serialiser should provide streamable and streamable POD base classes which can be inherited to indicate an object can be serialised/deserialised (with the latter indicating the class can be serialised by simply copying the binary data for the class). Serialiser should also provide various functions for writing/reading PODs, vectors, arrays, and maps from a binary file.

-> Physics system should be able to resolve collisions between rotated rectangles (using impulse resolution). It will divide all physics bodies into layers, and whether layers interact with each other will be specified when creating the layer (similar to Unity collision matrix)

-> Various noise functions (specifically white, interpolated, and cellular/voronoi) should be provided through classes, which are initialised with the noise function’s parameters. The noise values will be calculated given a coordinate, and will take into account the noise function’s seed

-> API for playing sounds using irrklang library, should allow sounds to be played on repeat, some basic time scheduling, change the master volume and specific volume, and stop/pause sounds

-> Particle system should provide a base class for a polymorphic Particle, which will store the required data for each particle, and a ParticleSystem class which handles the position/motion of particles, stores particles, and provides a simple function for rendering all particles

*Game*

-> Start menu with GUI that allows you to navigate to an options screen, a create world screen (where you enter a world name, and optionally a world seed), and a load world scene (which displays all the worlds on your system inside a certain directory)

-> Small world map in corner of the screen which displays a simplified rendition of the region around the player (1 tile -> 1 pixel)

-> World should generate infinitely around the player, with regions far out of sight being serialised and deleted from primary memory, and new regions either being generated or loaded from secondary memory if already generated. For each tile generated, a biome (forest/desert/etc.) will have to be selected based on noise (voronoi?), and the height of that tile will be selected based on another noise function (interpolated?). The terrain will also be randomly populated with trees/cacti, various ores, plants, and animals.

-> Inventory system should contain a GUI which displays all items the player currently has, and allows the player to move items around with drag/drop features

-> A crafting GUI should display all possible recipes the player can create, taking into account the items they have in their inventory. Clicking one of these displayed recipes will craft the item, subtracting the recipe’s ingredients from the inventory, and returning the item crafted

-> Holding down LMB on a mineable tile will break the tile, randomly selecting an item and quantity based on a predetermined drop table. The resulting item will be automatically picked up by the player when in close enough proximity, or will be pulled towards them when within a larger range.

-> Enemies and animals should spawn around the world. Animals will ensure there are not too many other animals around them when they spawn, and will randomly wander within a distance of their original spawn location. Enemies will spawn at night-time, pathfinding towards the player to attack them, whether that be right next to the player, or anywhere within their attack range.

-> The player will be able to select from a variety of melee/ranged weapons to defeat the enemies, and weapons will be swung/fired in the direction of the cursor. Successful hits on enemies will be detected, and an appropriate amount of damage will be dealt or the enemy will be killed (with enemies’ hits doing damage to the player). Health will restore naturally over time or artificially through certain actions such as eating.

## Constraints and limitations

### The game world

There will be severe limitations of the depth of the game world due to the time. It is likely only a couple biomes will be implemented, and few different tile types as well. While the algorithms to be implemented could support many more biomes, it would be too time consuming to program the individual terrain generation algorithms for each biome, or there would be a wide variety of biomes, but with roughly the same generation in all of them. Structures would also likely be very basic, since all the tiles of the structure must be hardcoded, thus adding a variety of structures would take too long.

### Game progression

Another major limitation of the game world is blocking progression. Blocking progression is essential to making a player feel like they’ve made progress; if a player can do the same things (defeat the same enemies, obtain the same resources, etc.) they can at the start of the game as they can 2 hours into playing, it is difficult to feel enthusiastic about continuing to play - there is no end goal. Terraria blocks progression into the world through certain bosses, and Minecraft blocks progression through obtaining certain ores (the rarer of which are hidden deeper underground), and finding certain structures. While I do plan to force the player to create better tools to access some ores, finding the ores is unlikely to be difficult as long as the player runs around the world long enough, thus there is no long-term goal to work towards, nor rare resource which the player can struggle to obtain.

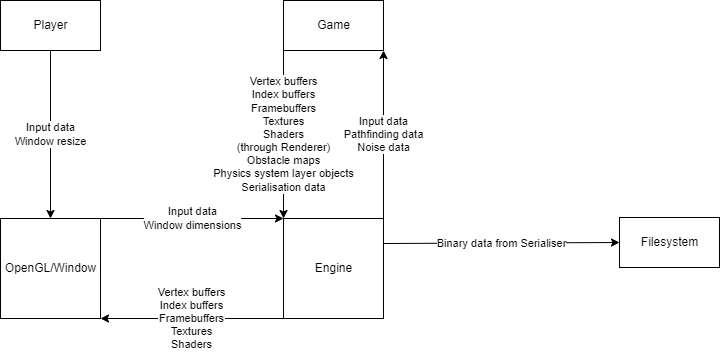
### Items/Crafting system

It is unlikely that there will be many recipes to choose from, since the resource variety is low, and creating new items takes a lot of work. If tools/weapons feel too similar, there is no incentive to pursue the resources to create more, unless they are drastically stronger.

## Model of the problem

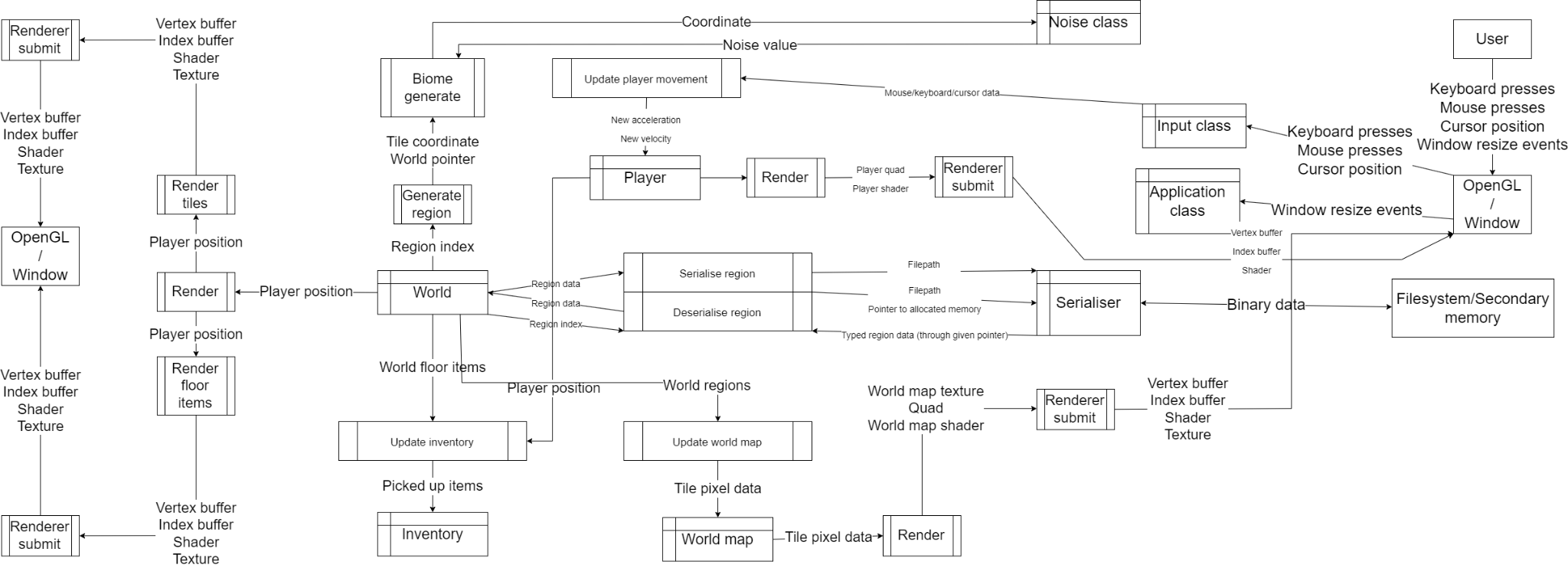
### Data flow diagram (Level 0)

Brief overview of the system with very generalised connections between entities



### Data flow diagram (Level 1)

Expanded overview of data flow in the system. Note that not all connections are shown, and some entities have been duplicated to make the diagram cleaner-looking (e.g. Renderer submit process is duplicated in multiple locations)



# 

# Design

## Class Diagram

Class diagram of classes in Game only  
  
