MARCH 17, 2023

MULTIPLAYER CREATIVE SANDBOX

DRACOTT, PERCY
HILLS ROAD

Analysis

Project Overview

In this project, I will be exploring a Solution to a problem, to fulfil the needs of my client. Video game, since their creating have been a means of socialising, connecting and relaxing. A recent boom in popularity, owing to people spending more time gaming since the recent Covid-19 pandemic and lockdowns, has resulted in games becoming a staple in many people's lives. This ever-growing popularity creates a strong demand for new and unique games to enjoy.

Project Outline

My client is an avid video game enjoyer, who like many enjoy the social aspect of video games. They require a multiplayer game in the "Creative Sandbox" genre, so that they can play and create with their friends.

While this may be available through other games, their requirements for a game that runs well on very low powered hardware so that they can run the game on their laptops in a LAN configuration, vastly limits the choices of game. This excludes games such as terraria or Minecraft.

With all this in mind, the client has made it clear that they want a complicated and optimised map generation algorithm as well as multiplayer capability, as connecting on the same session is the most core component, and the most decorated feature.

My Client

My client in question is an avid video game fan. They particularly like retro style games, with simple and well executed features.

An Interview with Elliot – My Client

Q1. How would you like the game to be played?

> A1: The Game needs to be playable on PC, as my friends and I all have laptops which we play our games on. Therefore, the controls need to be suited to a keyboard and mouse, and with a trackpad friendly control scheme.

Q2. How would you like the game to be used and by who?

> A2: The game will be played by me and my friends using multiplayer. In either a LAN with all players on my Wi-Fi, or over a WAN hosted by me as I know how to set my router up for Port-Forwarding.

Q3. Which aspects of a "Creative Sandbox" do you find most enjoyable?

> A3: For me I find the building aspect of the game most enjoyable. Coming up with an idea for a build. then creating it in 'reality' most appeals to me. I also enjoy exploring the game world, as I can often find inspiration of new builds in the landscape.

Q4. Which features would you most associated with a “Creative Sandbox”?

> A4: In my opinion, gathering recourses for a build through mining is a key feature of a creative game. As it rewards the player with a real sense of achievement and satisfaction, knowing the work it takes to complete their build. I also enjoy the sense of progression, as getting new tools makes once difficult tasks seem trivial.

Q5. Which areas of a “Creative Sandbox” map are you most interested in?

A5: Of course, the landscape of a map is important, as much of the gameplay occurs there. However, I find exploring the thrill of exploring cave systems in games to be the most enjoyable.

Objectives

O1. Have functioning game element, which:
- 1.1. Allows the player to move around the world,
- 1.2. Contains hostile Mobile-Entities (mobs) in the world,
- 1.3. Has some form of combat between the player and the mobs,
- 1.4. Has a unique procedurally generated terrain and cave system,
- 1.5. Has a destructible world where blocks can be mined by the player,
- 1.6. Has a building system where the player can build structures,

O2. Have a randomly generating map, which:
- 2.1. Is randomly generated at game start,
- 2.2. Has a tunnelling cave generation algorithm,
- 2.3. Has randomly generated trees,

O3. Have a functioning Menu System, which:
- 3.1. Allows for new games to be created,
- 3.2. Allows for players to connect online,
- 3.3. Displays the game title,
- 3.4. Allows the user to close the program,

O4. Have a functioning Multiplayer System, which:
- 4.1. Allows multiple players to play in the same world,
- 4.2. Allows the world to be synced across each player,
- 4.3. Allows interactions between players,
- 4.4. Allow players to connect through LAN or WAN

O5. Satisfy the clients brief by:
- 5.1. Have a save-able and load-able world,
- 5.2. Have a Day / Night cycle,
- 5.3. Include fantasy mobs such as skeletons, or zombies,
- 5.4. Be playable on PC,
- 5.5. Be playable on a mouse and keyboard or trackpad,

O6. Be appealing to my End-Users by:
- 6.1. Having a consistent art style throughout,
- 6.2. Using a consistent pixel by pixel tile size throughout,

Similar Games

In order to create my program and explore possible features, I have researched into 2 existing “Creative Sandbox” games. Minecraft and Terraria.

Minecraft	Terraria

• Made for PC and other platforms.= • 3D • Able to create LAN multiplayer games, however there's no inbuilt way to play WAN multiplayer for free. • Uses 16x16 pixel art style. • Procedurally generated map with cave system.	• Made of PC and other platforms • 2D • Can create free multiplayer games • Uses pixel art textures. • Side Scroller • Items to enhance your character. •

Research

From these nots I have devised that me "Creative Sandbox" game will:

- Be a side-scrolling 2D game, to reduce computational load on a system.
- Use Keyboard and Mouse/Trackpad controls.
- Feature pixel art graphics (8p8px).
- Have free WAN multiplayer capability.
- Include a uniquely generating map and cave system.
- Include crafting/progression to enhance a player's character.
- Include game saving where the world can be saved and loaded from the server.

My solution will require two applications, a server and client. My client will run the server application and then, they can connect using client applications.

Outline of Solution

Random Map Generation

To meet the clients brief, I will be making use of random map generation, meaning that a uniquely generated map can be made for each new game. In order to do this, I will need to find a way to randomly generate a map.

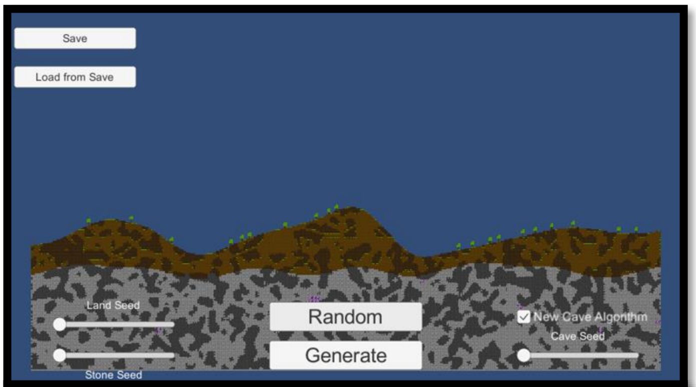
Although, Minecraft is a 3D game its' early beta methods for world generation are great to analyse. As it is made for 3D, an altered 2D version of its' system producing a single slice would be highly efficient. For my research of Procedural generation therefore, I will have researched the solution used by beta versions of Minecraft. Minecraft uses a Perlin noise function to give the height at each given coordinate, in addition to many other functions to add trees and caves when creating a map. As my game will be 2D I will have to research other methods of generation of cave systems and other world features. For cave generation, I will produce a prototype to explore different generation methods; giving me a chance to analyse and evaluate each solution. In this I will explore both a Perlin noise approach and an iterative cellular automata approach.

Multiplayer

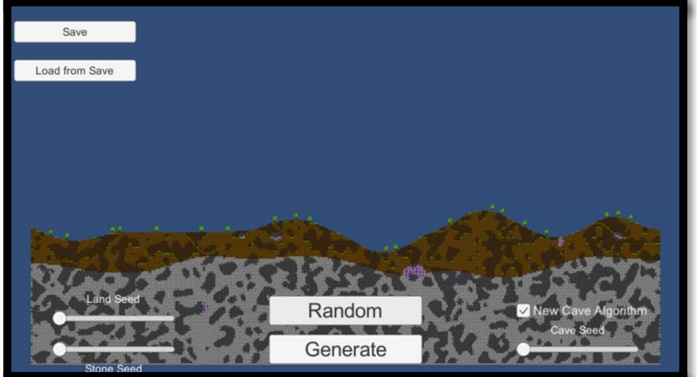
For multiplayer I will be using Riptide Networking through Unity, I chose this as it offers a high degree of control of network entities through code, with open-source code. For my client, I will produce two applications, a server and client build. With clients many clients connecting to one server. Clients will have to send the position of their player to the server as player movement will be managed client side. Ordinarily this is a bad idea for a multiplayer game, as this makes cheating much easier; a cheater can simply change the data in messages sent to the server, however Elliot has said that this won't be a problem as only a few trusted friends will have access to the game. As well as this, managing movement client side will spread the processing load across multiple systems in a thick-client type solution, reducing the load placed on the computer running the server application, Elliot notes that he is likely to run both the client and server applications simultaneously on his laptop and therefore optimisation of the networking side is key.

World Generation Prototype

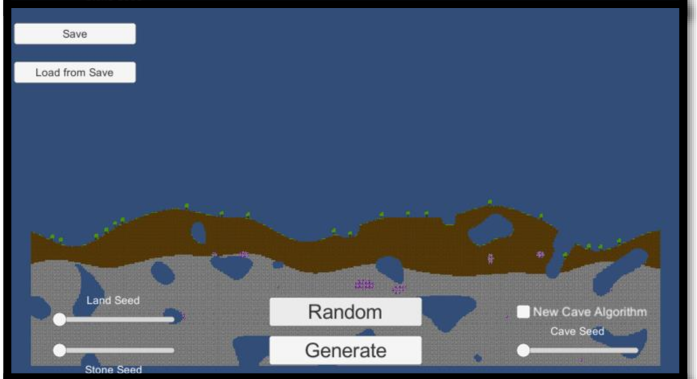
In this prototype, terrain generation is implemented using Perlin noise, by giving the function the x-coordinate and a seed value instead of a y-coordinate, for each value of x. This produces a smooth and varied terrain, before using Perlin noise I had experimented with using a Sin() based function to generate terrain, however this led to a far too periodic looking terrain with repeating hills and few areas of flat land. Elliot says it's important for



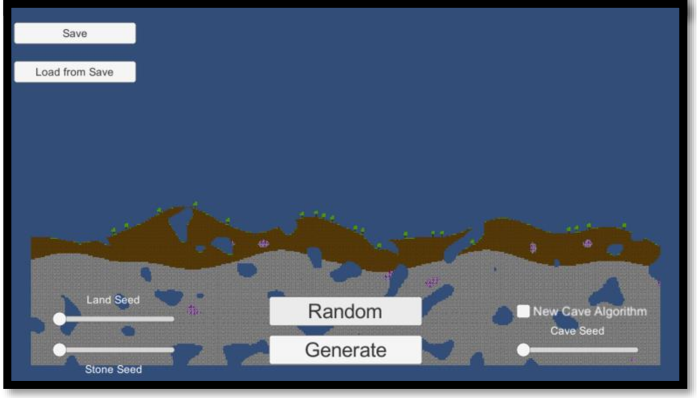
☑ New Cave Algorithm



☑ New Cave Algorithm



□ New Cave Algorithm

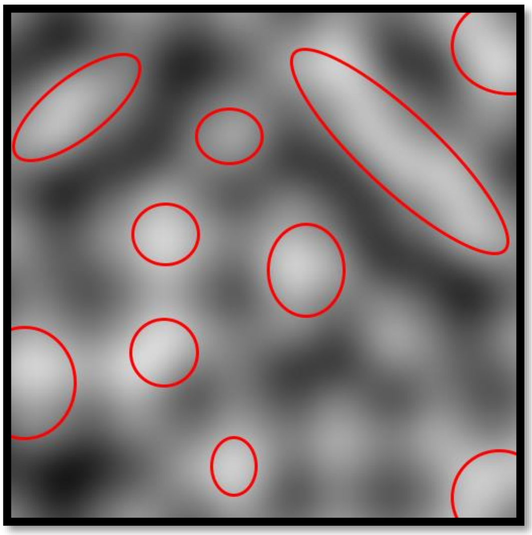


□ New Cave Algorithm

varied terrain to keep worlds looking interesting, however having a few flatter spaces is ideal for building.

Trees are added randomly, using a random integer value for the x-position, then checked for collisions to prevent trees from spawning onto of each other.

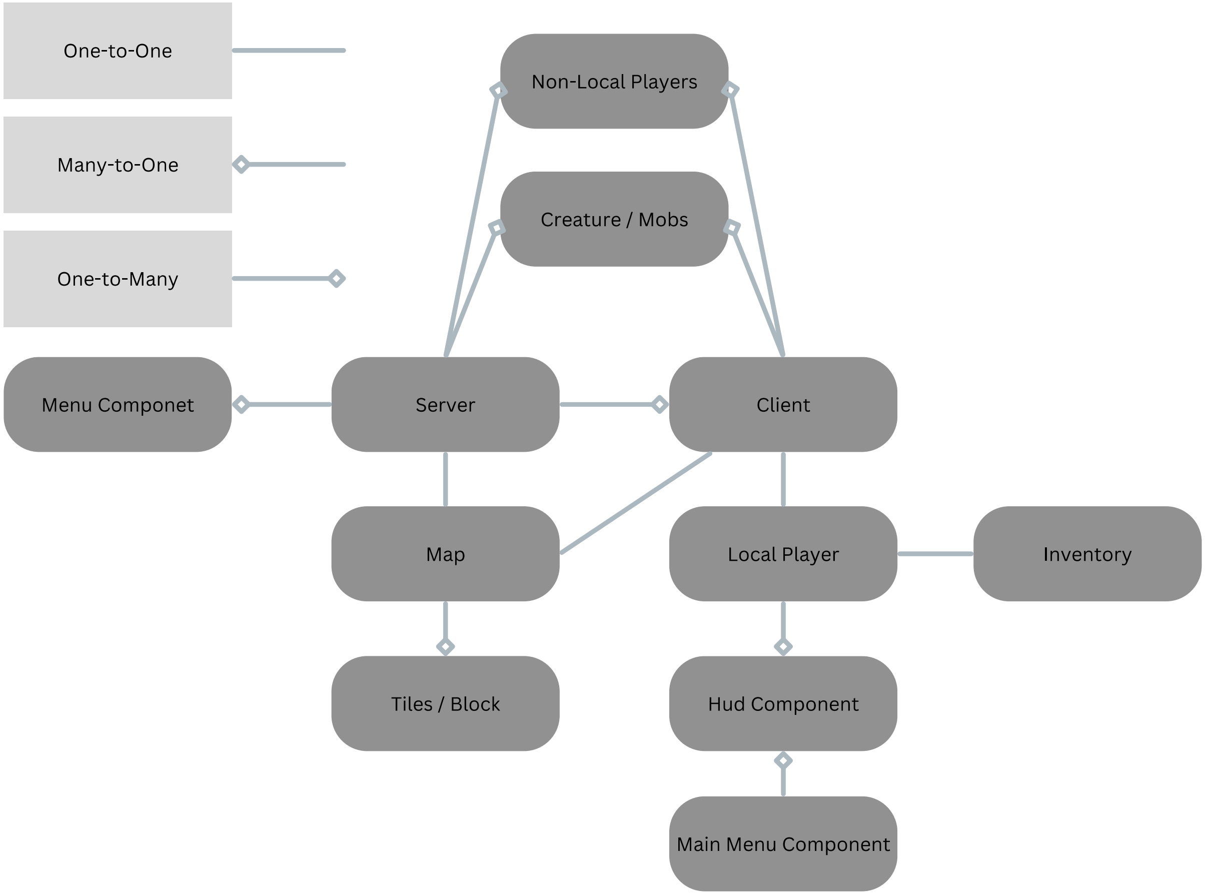
The first two screenshots show a cellular automata approach to cave generation. This code will be explained in Documented Design.



Screenshots 3 and 4 show the Perlin noise cave generation, where parts of the terrain are cut out using a Perlin noise map. Areas on the map are cut out if the resulting Perlin noise output of their position are above a threshold value, as shown. Any area of the map within the ellipse will be cut out as caves.

Elliot says that he greatly prefers the Cellular Automata based cave generation algorithm, as the caves produced by the algorithm are far more complex and varied.

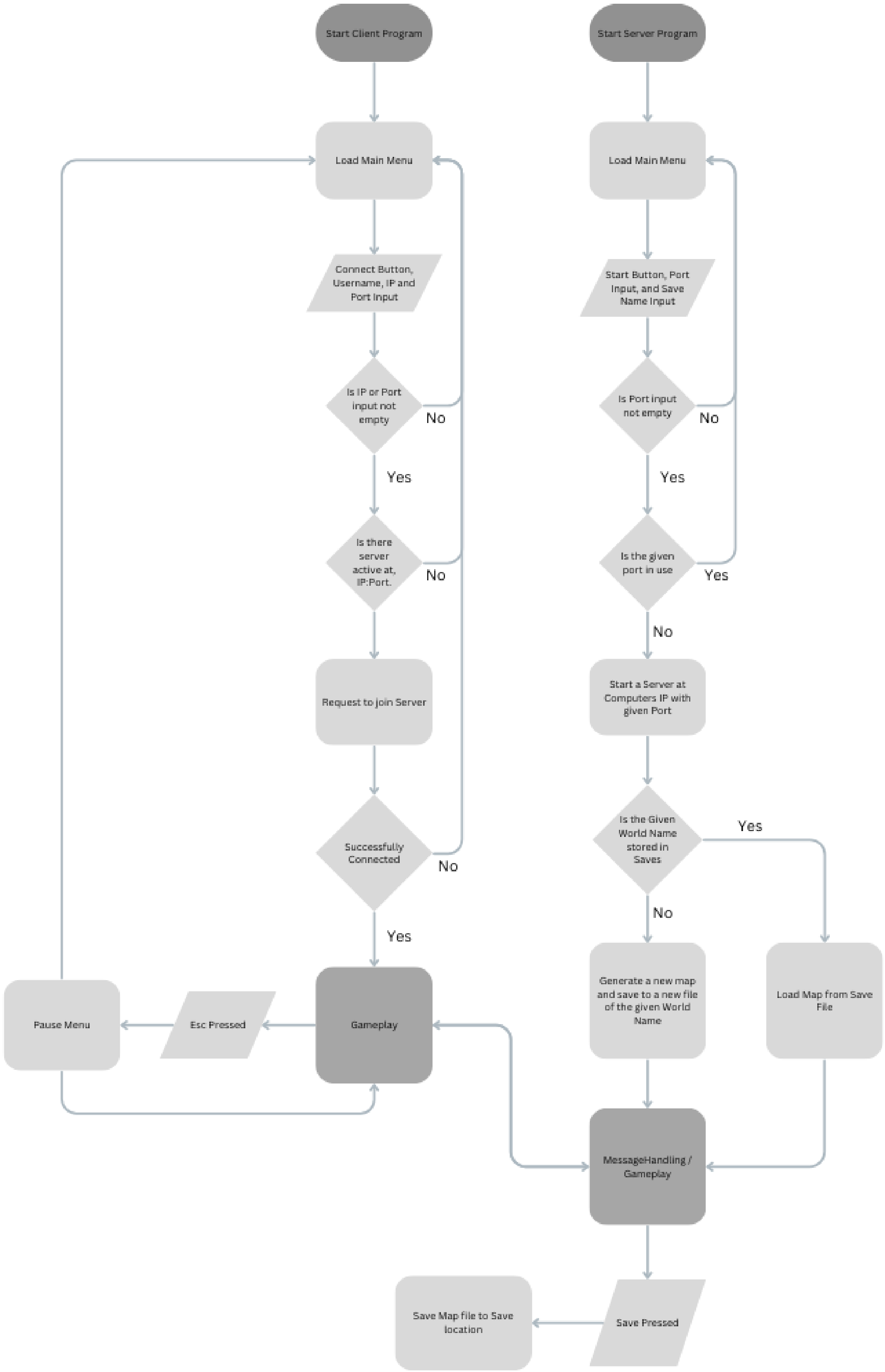
Entity Relationship & Scene Diagram



Within Unity entities, known as “game objects”, can be added to scenes, using the Instantiate() function. As shown above, on the client side a local player will be added to a scene in addition to non-local players. A local player contains all the control and interaction scripts which takes in the users’ inputs. A non-local player is a projection of another clients’ local player, whose position is sent to all other clients through the server to update the position of its’ corresponding non-local player.

Map, Mobs and Non-Local players, will be shared between both Server and all Clients connected. With. Mobs being controlled, spawned and respawned by the server.

System Flowchart


Start Client Program
Load Main Menu
Connect Button, Username, IP and Port Input
Is IP or Port input not empty
No
Yes
Is there server active at, IP:Port.
No
Request to join Server
Successfully Connected
No
Yes
Gameplay
Esc Pressed
Pause Menu
Start Server Program
Load Main Menu
Start Button, Port Input, and Save Name Input
Is Port input not empty
No
Yes
Is the given port in use
Yes
No
Start a Server at Computers IP with given Port
Is the Given World Name stored in Saves
Yes
No
Generate a new map and save to a new file of the given World Name
Load Map from Save File
MessageHandling / Gameplay
Save Pressed
Save Map file to Save location

Unity Engine Framework

Required Runtime Data

Player GameObject		
Data Name/ Access Name	Data Type	Represents
transform.postition	Vector3	Position of the player in the scene
transform.scale	Vector3	The overall scale of the player in the scene
transform.localScale	Vector3	The scale of the player relative to its current scale
playerHealth	integer	Current player health
inventory	Integer array	An array of the current block held in the inventory
hasAxe	bool	Whether the player has a Pickaxe
hasSword	bool	Whether the player has a Sword
itemHolding	integer	The ordinal number for the inventory array
ScreenToWorldPoint(Input.mousePosition)	Vector3	Returns the position of the mouse in relation to its position in the physical scene
velocity	integer	The velocity of the player in the scene
OnGround	bool	A physics overlap at the players feet to determine if they are on the ground
Id	ushort	The players Id in the scene
IsLocal	bool	Whether the player is the Local player

Zombie GameObject		
Data Name/ Access Name	Data Type	Represents
transform.postition	Vector3	Position of the entity in the scene

transform.scale	Vector3	The overall scale of the entity in the scene
transform.localScale	Vector3	The scale of the entity relative to its current scale
playerInViewRange	bool	Whether a player is in sight range
PlayerInAttackRange	bool	Whether there's a playing the attacking range
closestPlayer	Vector3	The closest player entity to the zombie
Id	ushort	The Id of the zombie in the scene

Input field Component		
Data Name/ Access Name	Data Type	Represents
text	string	The text held in the interactable area
position	Vector2	Position on the UI canvas

Button Component		
Data Name/ Access Name	Data Type	Represents
DisplayText	string	The text held in the text area
position	Vector2	Position on the UI canvas
colour	Color	The colour applied to the texture of the button

Tile Component		
Data Name/ Access Name	Data Type	Represents
TileMap	TileMap	The tilemap the tile belongs to.
Tile	RuleTile	The current tile being rendered
position	Vector2	The position of the tile in the TileMap's grid
tilemapCollider	Collider	The physics collider of the tile

Hud Component		
Data Name/ Access Name	Data Type	Represents

text	string	The text held in the text area
position	Vector2	Position on the UI canvas
texture	Texture	Current rendered texture

Asset Requirements

Text fonts used will be acquired from www.dafont.com, specifically using fonts that are completely royalty free, all fonts used will be credited in the source tracker.

All textures are of my own making, made using Adobe Photoshop.

Design

Function Abstraction – From Flowchart

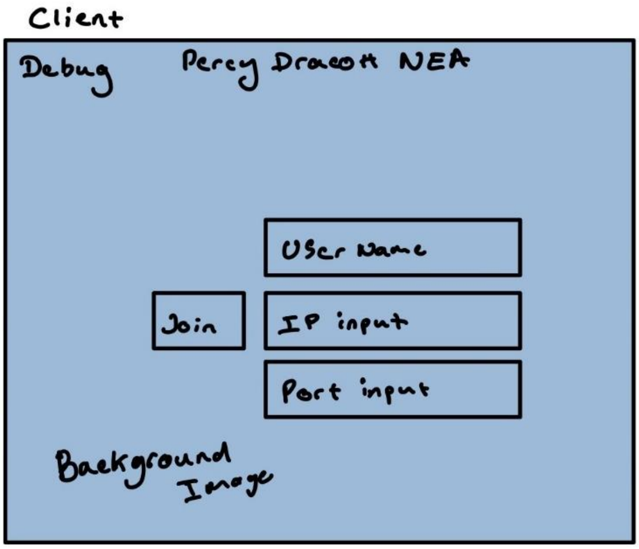
Server	
Function	Definition
Load Main Menu	All Menu Components, and background images in the scene are enabled, input text boxes are unlocked
Request to Start Server	Port Number is taken in from the input field and passed into the constructer for the Server class
Generate New Map	Landscape and Rock seeds are randomly generated. Empty 2D array of map size and width is generated. Perlin noise-based generation function and cellular automata cave generation function is run. This is copied to the 2d array and then rendered to the Tile map.
Load Map from Save	A new Stream Reader is created, then this cycles through each line of the save .txt file and the added to the 2D map array.
Save Map	A new Stream Writer is created. Foreach row in the 2D map array, the content of the row is written to the file with StreamWriter.Write().
Gameplay	Many functions will run on gameplay, MessageHandling will include syncing player positions, block placing and breaking, in addition to passing the map file between the server and client upon joining.

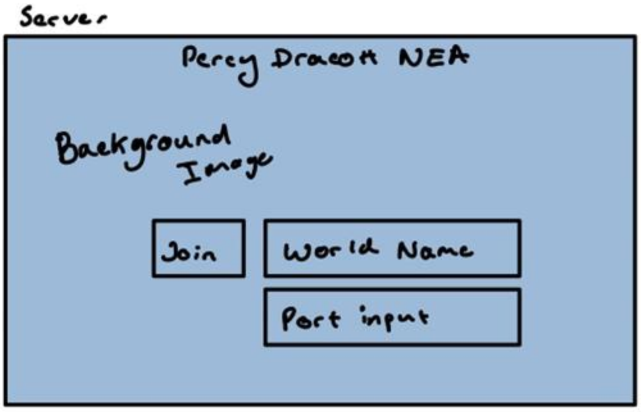
Client	
Function	Definition
Load Main Menu	All Menu Components, and background images in the scene are enabled, input text boxes are unlocked
Request to Join	Port number and IP address will be taken in as a ushort and string from the input fields. All menu Components are disabled, and input fields are locked. A request is sent to join any server at the given Ip and Port. This will return a bool; a failed connection will reload the Main Menu.
Gameplay	First the map will be loaded from the server and then a player spawned. Positions of the local player and any modifications made to the map will be sent to the server.
Pause Menu	Pressing the Esc key will toggle the pause menu. On a press, all pause menu components will be enabled and all player controlling scripts will be disabled. This will allow a player to resume or leave the server, returning them to the main menu.

User Interface Diagram

Throughout the design phase, I have sent various diagrams, algorithms, and interface designs to Elliot, my client, to see if he thinks they meet the objectives, or if they need to be improved on. I will have different versions of these throughout the document.

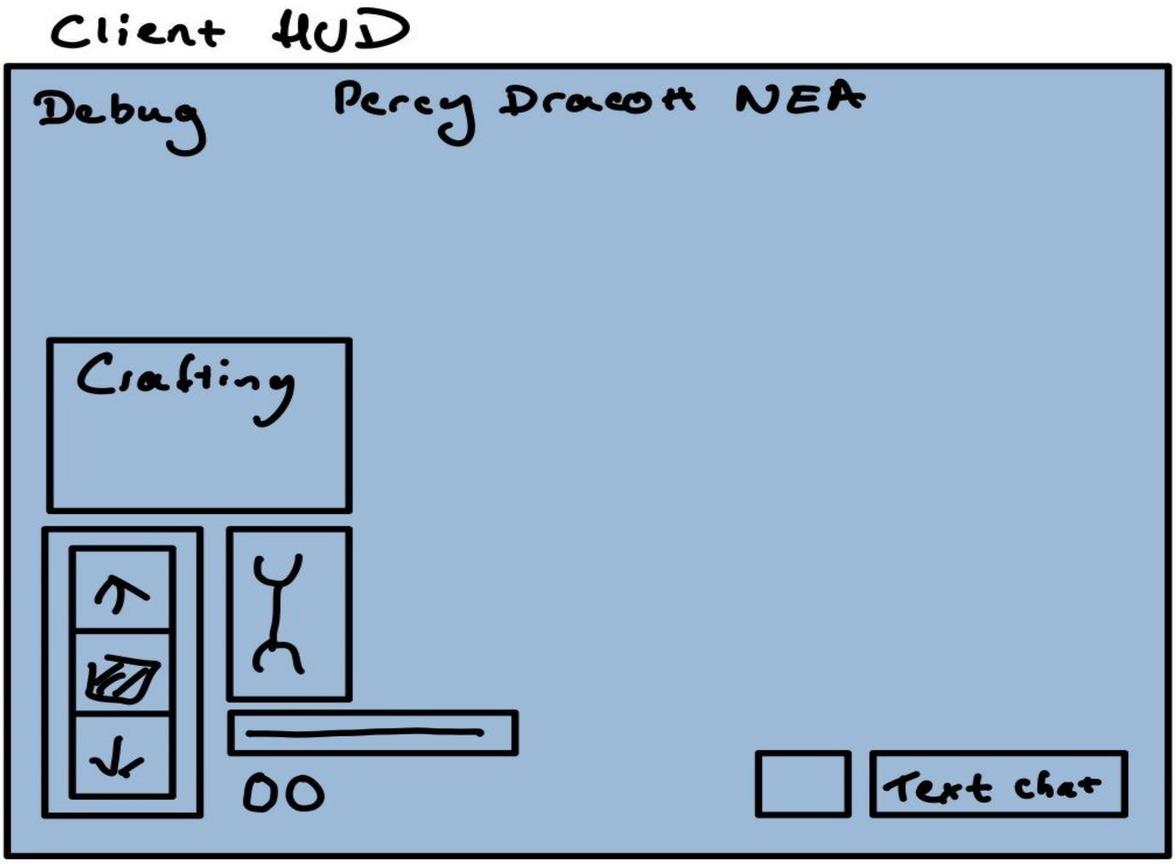
Here's the initial and final designs for the user interface; main menu and HUD.





The Main menu scene of both the client and server applications will obviously contain a few collections of graphics, including text components, buttons, and input fields. These will all need to be unloaded when a game is started.

The HUD layout, as planned is below,



The HUD layout contains a prototype for the inventory system, where the player can scroll through items represented by the shaded square. Each element on the HUD drawing will be rendered above the gameplay. Below the crafting icon, which will toggle the 'crafting' element, is the health bar which will display the hp of the local player.

Algorithms

I've given each algorithm a rough difficulty level to implement, easy, medium, or hard – the harder algorithms may need more revisions as other elements of the design, such as data structures or data storage, develop.

Map Generation

As I've made a world generation prototype, I will go over the procedures and functions contained in that first.

Name: Generation	Definition: Overall function that call each part of the generation script, and initialises the map
Input: worldWidth, worldHeight	Output: N/A
Subroutine Generation(worldWidth, worldHeight)	

```
    ForgroundTileMap.ClearAllTiles()
    BackgroundTileMap.ClearAllTiles()
    Seed = Random.WithinRange(0,1000)
    StoneSeed = Random.WithinRange(0,1000)
    CaveSeed = Random.WithinRange(0.02,0.05)
    OreSeed = Random.WithinRange(0.03,0.05)

    map = new int[worldWidth,worldHeight]
    If IsFromSave()
        LoadMap()
        SaveMap()
    else
        OptimisedTerrainGeneration(map, worldWidth, worldHeight, GrassSoil, Stone)
        ApplyCaves(map);
        AddTrees(TreePopulation, TestTileFG, Log, Leaf)
    Endif

    Renderer(map, TestTileFG, TestTileBG)
    GenerateSaveFile(worldName)
    SaveMap()
endSubroutine
```

Relevant Objective: 2.1	Time Complexity: O(1)
Difficulty: medium	

Name: TerrainGeneration	Definition: A function that creates the soil and stone layer of the map, in addition to adding ores. This is done in a loops per column of the map for efficiency.
Input: worldWidth, worldHeight, worldMap, GrassTexture, StoneTexture	Output: N/A

```
Subroutine TerrainGeneration(worldWidth, worldHeight, worldMap, Grass, Stone)
    perlinNoiseSoil = 0
    perlinNoiseStone = 0
    For x = 0, x < worldWidth, x++
        perlinNoiseSoil = RoundToInt(PerlinNoise(x / (smoothness), Seed) * worldHeight / 5)
        perlinNoiseSoil += worldHeight / 3
        For y = 0, y < perlinNoiseSoil, y++
            worldMap[x,y] = 1
        Endfor
        perlinNoiseStone = RoundToInt(PerlinNoise(x / (smoothness), StoneSeed) * worldHeight / 8)
        perlinNoiseStone += worldHeight / 4
        For y = 0, y < perlinNoiseStone, y++
            worldMap[x,y] = 2
        Endfor
        For y = 0, y < perlinNoiseSoil, y++
            PerlinNoiseOre = RoundToInt(PerlinNoise(x * OreSeed, y * OreSeed))
            If  PerlinNoiseOre <= OreThreshhold
                map[x,y] = 3
            Endif
        Endfor
    Endfor
endSubroutine
```

Relevant Objective: 2.1	Time Complexity: O(n^2)
Difficulty: medium	

Adding Caves

Adding Caves was specified by my clients as a major aspect of the generation system, therefore Elliot and I had, many conversations to find the most appropriate generation system that fitted with his expectations.

Name: ApplyCaves Version 1	Definition: Function that cuts caves out of the 2D array map file.
Input: worldWidth, worldHeight, worldMap	Output: N/A

```
Subroutine ApplyCaves(worldWidth, worldHeight, worldMap)
    For x = 0, x < worldWidth, x++
        For y = 0, y < worldHeight, y++
            PerlinNoiseCave = RoundToInt(PerlinNoise(x * CaveSeed, y * CaveSeed))
            If PerlinNoiseCave >= CaveThreshhold
                map[x,y] = 0
            Endif
        Endfor
    Endfor
endSubroutine
```

Relevant Objective: 2.2	Time Complexity: O(n^2)
Difficulty: easy	

Name: ApplyCaves Version 2	Definition: Function that cuts caves out of the 2D array map file, using cellular automata
Input: worldWidth, worldHeight, worldMap	Output: N/A

```
Subroutine ApplyCaves(worldWidth, worldHeight, worldMap)
    RandomFillGeneration()
    Automata(iterations)
    For x = 0, x < worldWidth, x++
        For y = 0, y < worldHeight, y++
            int topOfWorld = RoundToInt(PerlinNoise(x / (smoothness), Seed) * worldHeight / 5)
            perlinNoiseSoil += worldHeight / 3
            If  cavemap[x, y] == 0 && y != 0 && y != topofworld - 1 && y != topofworld - 2
                If  map[x,y] == 1
                    map[x,y] = 8
                Endif
                If  map[x,y] == 2 OR map[x,y] == 3
                    map[x,y] = 9
                Endif
            Endif
        Endfor
    Endfor
endSubroutine
```

Relevant Objective: 2.2	Time Complexity: O(n^2)
Difficulty: easy	

Name: RandomFillGeneration	Definition: Function that fills a 2D array randomly with a random fill percentage.
Input: worldWidth, worldHeight	Output: N/A

```
Subroutine RandomFillGeneration(worldWidth, worldHeight, caveMap)
    caveMap = new int[worldHeight, worldWidth]
    For x = 0, x < worldWidth, x++
        For y = 0, y < worldHeight, y++
            caveMap[x,y] = Random.WithinRange(0,100) < RandomFillPercentage ? 1 : 0
        Endfor
    Endfor
endSubroutine
```

Relevant Objective: 2.2	Time Complexity: O(n^2)
Difficulty: easy	

Name: Automata	Definition: Function that fills a 2D array randomly with a random fill percentage.
Input: iterationCount, worldWidth, worldHeight	Output: N/A

```
Subroutine Automata(iterations, worldWidth, worldHeight)
    For i = 0, i < count, i++
        int[,] tempMap = caveMap.Clone
        For xs = 0, xs < worldWidth – 1, xs++
            For ys = 0, ys < worldHeight – 1, ys++
                int neighbours = 0
                neighbours = GetNeighbours(tempMap, xs, ys, neighbours)
                If  neighbours > 4
                    caveMap[xs,ys] = 1
                else
                    caveMap[xs,ys] = 0
                Endif
            Endfor
        Endfor
    Endfor
endSubroutine
```

Relevant Objective: 2.2	Time Complexity: O(n^2)
Difficulty: medium	

Name: GetNeighbours	Definition: Finds the number of neighbouring active cells in the array, around a given point.
Input: tempMap, xs, ys, neighbours	Output: int neighbours

```
Subroutine GetNeighbours(tempMap, xs, ys, neighbours)
    For x = xs – 1, x <= xs, x++
        For y = ys – 1, y <= ys, y++
            If  y <= worldHeight AND x <= worldWidth
                If  y != ys AND x != xs
                    If  tempMap[x,y] = 1
                        neighbours++
                    Endif
                Endif
            Endif
        Endfor
    Endfor
    Return neighbours
endSubroutine
```

Relevant Objective: 2.2	Time Complexity: O(1)
Difficulty: medium	

Adding Trees

Adding Trees was also pointed out by Elliot for part of the world generation, in addition this opens avenues for weapon crafting and addition building blocks that can be obtained through crafting, such as the plank block.

Elliot notes that have blocks of multiple colours and textures is important for keeping builds looking interesting.

This algorithm ensures that trees are not placed atop each other, by checking the position of each past tree. This is done by looping through the positions of all placed trees, and subtracting the x-coordinate of a proposed tree, from the position of a placed tree, and ensuring that the absolute value of the sum is greater than 6; as this is the padding value given so trees aren't overly clustered.

Name: AddTrees	Definition: Adds a given number of trees to the surface of the terrain, without placing then atop one another
Input: worldWidth, worldHeight, DensityCount	Output: N/A

```
Subroutine AddTrees(worldWidth, worldHeight, Map, density)
    //Check for overpopulation
    If  density >= worldWidth / 6
        density = worldWidth / 6
    Endif
    int temporaryX = 0
    int[] positionHistory = new int[density]

    For i = 0, x < density, i++
        bool collisionAvoidance = true
        While(collisionAvoidance)
            int holdingPositionVariable = Random.WithinRange(2, worldWidth - 2)
            collisionAvoidance = false
            For j = 0, j <= i, j++
                If  AbsoluteValue(holdingPositionVariable - positionHistory[j]) < 6
                    collisionAvoidance = true
                Endif
            Endfor
            If  collisionAvoidance == False
                temporaryX = holdingPositionVariable
            Endif
        Endwhile
        positionHistory[i] = temporaryX

        int temporaryY = worldHeight - 1
        While Map[temporaryX,temporaryY - 1] == 0
            temporaryY--
        Endwhile

        Map[temporaryX,temporaryY] = 4
        Map[temporaryX, temporaryY + 1] = 4
        Map[temporaryX, temporaryY + 2] = 5
        Map[temporaryX, temporaryY + 3] = 5
        Map[temporaryX, temporaryY + 4] = 5
        Map[temporaryX + 1, temporaryY + 2] = 5
        Map[temporaryX – 1, temporaryY + 2] = 5
        Map[temporaryX + 1, temporaryY + 3] = 5
        Map[temporaryX – 1, temporaryY + 3] = 5
        Map[temporaryX + 1, temporaryY + 4] = 5
    Endfor
endSubroutine
```

Relevant Objective: 2.3	Time Complexity: O(n!)
Difficulty: hard	

Name: Renderer	Definition: Function that takes the 2D map array and displays the content to the Tile map
Input: worldWidth, worldHeight, worldMap	Output: N/A

```
Subroutine Renderer(worldMap)
    ForgroundTileMap.ClearAllTiles()
    BackgroundTileMap.ClearAllTiles()
    For x = 0, x < worldMap.GetHorizontalLength, x++
        For y = 0, y < worldMap.GetVerticalLength, y++
            If  worldMap[x,y] == 1
                ForgroundTileMap.SetTile(new Vector3Int(x,y,0), GrassSoil)
                BackgroundTileMap.SetTile(new Vector3Int(x,y,0), GrassSoilBackground)
            Endif
            If  worldMap[x,y] == 2
                ForgroundTileMap.SetTile(new Vector3Int(x,y,0), Stone)
                BackgroundTileMap.SetTile(new Vector3Int(x,y,0), StoneBackground)
            Endif
            If  worldMap[x,y] == 3
                ForgroundTileMap.SetTile(new Vector3Int(x,y,0), Ore)
                BackgroundTileMap.SetTile(new Vector3Int(x,y,0), StoneBackground)
            Endif
            If  worldMap[x,y] == 4
                ForgroundTileMap.SetTile(new Vector3Int(x,y,0), Log)
            Endif
            If  worldMap[x,y] == 5
                ForgroundTileMap.SetTile(new Vector3Int(x,y,0), Leaf)
            Endif
            If  worldMap[x,y] == 6
                ForgroundTileMap.SetTile(new Vector3Int(x,y,0), Plank)
            Endif
            If  worldMap[x,y] == 8
                BackgroundTileMap.SetTile(new Vector3Int(x,y,0), GrassSoilBackground)
            Endif
            If  worldMap[x,y] == 9
                BackgroundTileMap.SetTile(new Vector3Int(x,y,0), StoneBackground)
            Endif
        Endfor
    Endfor
endSubroutine
```

Relevant Objective: 2.1	Time Complexity: O(n^2)
Difficulty: easy	

Map Saving and Loading

Elliot notes that the ability to save and load past maps is really important, as it will enable him and his friends to spend multiple sessions on one project.

Name: SaveMap	Definition: Function that saves the 2D map array to a file
Input: worldMap	Output: N/A

```
Subroutine SaveMap(worldMap)
    Using StreamWriter sw = new
StreamWriter($"WorldSaves/{worldName}/worldSave.txt")
        For y = 0, y < worldMap.GetVerticalLength, y++
            For x = 0, x < worldMap.GetHorizontalLength, x++
                sw.Write(worldMap[x,y])
            Endfor
            sw.WriteLine()
        Endfor
    EndUsing
endSubroutine
```

Relevant Objective: 5.1	Time Complexity: O(n^2)
Difficulty: medium	

The map will be located in a save folder: WorldSaves/{worldName}/worldSave.txt, where worldName is a string variable passed in from the input fields upon starting a server.

Name: LoadMap	Definition: Function that fills an empty 2D array with the map data.
Input: worldMap	Output: N/A

```
Subroutine LoadMap(worldMap)
    int y = 0
    Using StreamReader sr = new
StreamReader($"WorldSaves/{worldName}/worldSave.txt")
        string value
        While (value = sr.ReadLine()) != Null
            For x = 0, x < worldMap.GetHorizontalLength, x++
                worldMap[x,y] = ConvertToByte(value[x]) – 48
            Endfor
            y++
        Endwhile
    EndUsing
    Renderer(worldMap)
endSubroutine
```

Relevant Objective: 5.1	Time Complexity: O(n^2)
Difficulty: medium	

The following two functions will be used on the server application to check if a map is being loaded from a save of is new, and then create the saving directory if necessary

Name: GenerateSaveFile	Definition: Function that creates the file saving directory upon loading a new world, not a previous save.
Input: worldName	Output: N/A

```
Subroutine GenerateSaveFile(worldName)
    if File.Exists($"WorldSaves/{worldName}") == false
        Directory.Create($"WorldSaves/{worldName}")
    Endif
endSubroutine
```

Relevant Objective: 5.1	Time Complexity: O(1)
Difficulty: medium	

Name: IsFromSave	Definition: Function that returns a Boolean if the given world name has a file directory
Input: worldName	Output: Boolean

```
Subroutine Bool IsFromSave(worldName)
    if File.Exists($"WorldSaves/{worldName}")
        return true
    else
        return false
    Endif
endSubroutine
```

Relevant Objective: 5.1	Time Complexity: O(1)
Difficulty: medium	