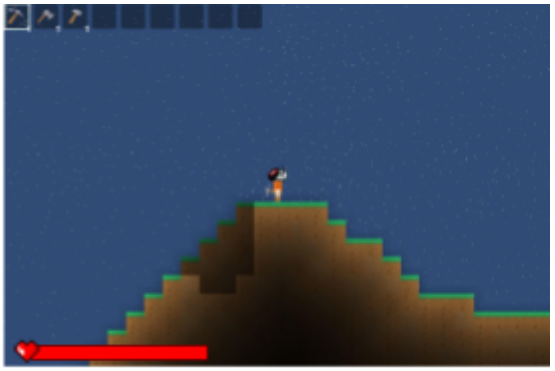
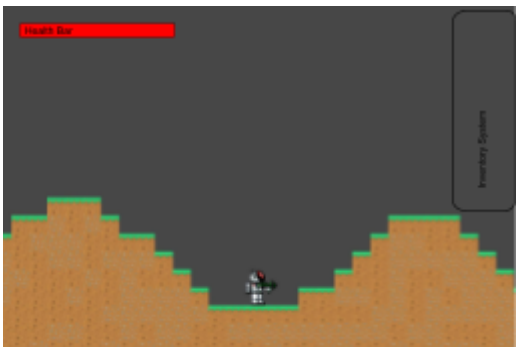
Changes From Project Plan

Design Changes

I have made a couple changes to the design of my game throught the development and testing stages. I will start with covering the UI and visual aspects first. I changed my settings menu from the original design and simplified it as you can see from comparing the original design (Left) and the final implementation (Right). I did this as although useful I did not see it totally necessary to implement the screen resolution drop down menu. I also changed the text on all my menus to a darker black to make it more clear.

Another change I made from the origonal plan was how the player inventory and health bar are displayed. I decided to implement a hotbar to display the top row of the inventory.This decicion was made after testing as the tester would struggle with the scroll through the inventory functionality when it was vertical. You can see how these changes where made bellow. The image on the left is the running game and the image on the right is the plan.





GamePlay Changes

The main gameplay change I have made is that the player no longer fights multiple enemies. This is something that I would have liked to implement if I had not spent so much time on other aspects of the game. In this stage of my game the player now fights a Boss that has multiple different movesets and stages. The boss fight will trigger on the third night attacking the player until the boss is dead. The boss has two stages, a normal stage and a “rage” state which is triggered when it’s health drops bellow 40%. In this second stage the boss does more damage and moves faster aswell as doing a different attack. This was implemnted using a state machine inside Unitys animator.

Another aspect of the game that I have changed is the player health and respawning loop. Now when the players health drops to 0 they are sent back to the spawn point but their inventory is not emptied. This was implemented as the contents of the players inventory when emptied would then be destroyed when the player was moved back to the spawn point if the chunks where they died where no longer loaded. I decided to remove the empty inventory feature as from testing players found it draining to loose all their items with no hope of getting them back. I decided this was the less important aspect as the occlusion culling impacted the games performance when generating large worlds a lot.

The biggest change I have made to my design is that I have been unable, as of this hand in to implement the end game function. Currently this game will run indefinitely with no game end goal. This is something that I am disappointed about and will be covering in the Changes I would make if I was to do the project again section. I choose to do this as I believe a polished open world game allows for far more creativity and enjoyment for the player than reaching an end goal after unsatisfying gameplay.

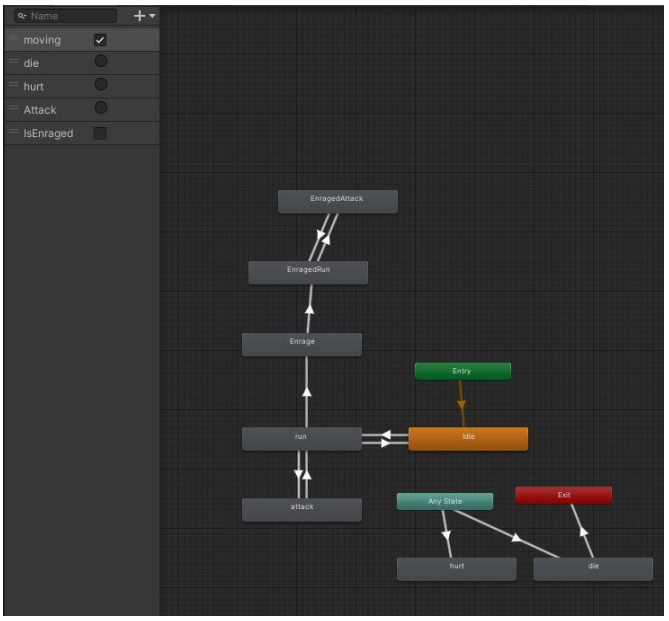
Architectural design

As I was using Unity to develop my game I opted to use the Entity Component System Architectural design as Unity is an EC based development platform. I chose this as it would allow me to most efficiently manage and maintain my project over time. Examples of this within my game are entities which are instances of GameObject such as my player that contain multiple components and Systems such as Transform or scripts such as Player Controller or the Player Inventory. I used this by attacking the components to the entity using Unitys Hierarchy Panel and Inspector Panel.
I did consider using the Model View Controller architecture for my game as from my research I found that it can be easier to develop, update and debug in the long run as well as allowing for rapid development. This would have been good as I believe I had quite a hard game idea to implement in the time frame considering I have very little game development experience and no prior experience working in Unity. Therefor I decided to opt for the standard Unity Entity Component System architectural design.

Game design patterns used

Throught my game I have tried to used multiple different patterns. I have used the **State** pattern for my player, boss and boss AI. For the player I have implemented this by using a boolean variable to check if the player is on the ground. If this boolean is true then they are able to jump and if it is false they are not able to jump as they are not touching the ground. This check is done through the use of a Capsule Colider 2D component attached to the player Entity.
 I have also used the state design pattern in my AI for the boss and boss battle. I have implemented this by creating a State Machine using Unitys Animator. This allows the boss to trigger actions only while in the required state. An example of this could be the boss entering its second stage, when the boss' health drops below a certain level the IsEnraged boolean is set to true and the boss enters its enraged mode. During this stage the boss changes to its Enrage animation and then to the EnragedRun state in which It has different characteristics

than the original stage. This State Machine can be seen below and code for this can be found in the Boss_Enrage script located in the Enemy folder.



I have also implemented the **Flywieght** pattern in my Terrain generation to store each individual tiles/blocks information. This was done so that I could procedurally generate the world using perlin noise and to make the game more light weight in terms of data and storage as there are thousands of these blocks. Each individual block stores its position and inherits other shared data like the model. While implementing this I followed an example on generating trees in “Game Programming Patterns” by Robert Nystrom. My example of this can be found in my ItemClass using enums and the TileClass, OreClass, TileAtlas, BiomeClass scripts.

I also used the **Update Method** Sequencing Pattern this was implemented using Unity’s Update Method. I choose this pattern because I often had to update GameObjects each frame, such as the player using the PlayerController script. This allowed me to make sure variables such as player position where accurate as my game has a number of objects or systems that need to run simultaneously.

Required Componets

Artifical Intelligence

I used AI in my game to control and effect my Boss battle event. This event is triggered on the third night of the game cycle. During this event the Boss will locate the player and attack them. After the player does enough damage to the boss the second stage of the boss’ behavour is triggered. While transferring to this stage the boss will become invincible as this was a recommendation from testers inspired from boss fights in other games. In this stage the boss has a different attack which does more damage and moves faster. This system runs using a State Machine coded within Unity’s Editor. This system means that the boss can only be in one state at the a time and can only transfer between states if there is an active and acceptable transition between the two states.
The Boss fight is meant to be a key aspect of the game, so implementing multiple different stages allows for the fight be more interesting and exciting, especially for new players.

Using the Animator for the for this State Machine was extremely helpful as it has a good UI and allows me to intuitively show the AI and build on it to make it more complex easily.

Physics

My physics are built within the Unity editor using Capsule Colliders. As my game doesn't require massive amounts of physics calculations I decided this would be the best. My physics are able to be edited within the player controller script or within the inspector window for quick changes. I can change the players speed and the world gravity through the JumpForce variable. Although this isn't changing throughout the game as the player is the only Entity using this it gives the perception of changing gravity.

Ethical and Security Implications

Ethical

All video games affect people who play them, some of the ethical issues include: violence, education and gambling. While designing my game I have decided to minimise these effects as much as I can. I have made sure to not implement any pay walls or lootboxes to combat gambling in games. I have tried to minimise the violence in my game as much as possible, my game does include a shooting and fighting mechanic but as a large population of games do I belive the way I have done this is acceptable, and minimises any risk. I believe as my game is very open ended it promotes creativity, I would like to emphasize this more as I continue with development.

Security

I have been careful while building my game to minimise any security risks for the game and its players. As my game isn't networked/multiplayer there isn't a whole lot of risks involved. I would consider this more if I had more time to implement some features such as a save and load function. A save function would need to be designed well as to not to tamper with users file system on their computer. And a Load function would need to be implemented securly as not to have incorrect files loaded into the game.

Changes I would make if I was to develop this game again

There is very little I would change if I could start this project from scratch, but there are a couple. Such as, I wouldn't spend so much time on the terrain generation as although I love where it's at currently it impacted a lot of my features after it due to time restrictions. I would have liked to finish the win game condition as it may apply a bit more story to the game in general. I also struggled with implementing the AI as I didn't really know what would classify as AI in this project. I kept changing my mind about how I was implementing it which impact multiple things down the line of the project, but after some research I am glad at where that ended up.

Testing

As well as constant Self playtesting throught development I followed a Confidants Playtesting strategy as I believe this was the way to get the best feedback and improve my game experience as testers would give the most accurate opinions and results.
I had 6 testers for my game who tested the game over 2 rounds. The first Testing stage was an early build of the game, in this test I wanted mostly feedback about UI and gameplay.

Recruitment

- 6 Participants
- 5 Male
- 1 Female
- All participants were able-bodied, and able to use a computer without assistance.
- All participants had played games on a PC before and were familiar with basic standard inputs.

Testers undertook a brief five minute training task, by testing the controls and familiarising themself with them.
Testers where then given 10 minutes to play the game however they liked while making note of any bugs/issues they came across. After this stage I also took feedback from the testers on any gameplay issues or improvements they would like to see in the next stage.

Game testing round 1 errors found

Player issues

1. Player sometimes gets stuck in blocks if player is standing on the corner then jumps and places block below them.
    a. Unable to fix in timeframe - not a serious issue as player can jump and escape from being inside block.
    b. I think it may be a game update() issue calling the block place when it is not obstructed by player and then player falls into the block after it has been approved to be placed but not placed yet.
2. Sometimes blocks in inventory glitch giving the player infinite blocks of this type – unable to find reason.

Boss issues

1. Boss can attack immediatly affter attacking player.
    a. Fixed with adding reset trigger in OnStateExit() in StateMachine (Boss_Run)
2. Players preferred not going when the boss had been triggered.

General Game issues

1. Game sometimes lags on placing blocks and updating lighting.
    a. Changed the lighting radius so that a smaller area is updating therefor not as demanding on graphics.

Suggested Edits

- Change orientation of inventory
- add hot keys for tools (1-3).

Game testing round 2 errors found

No serious or game breaking errors found! Players enjoyed the game more with suggested edits.

References

- Game Programming Patterns” by Robert Nystrom
    - Design Patterns Revisited
        - Singleton
        - State
    - Sequencing Patterns
        - Update Method

Assets used in game

All game assets in this project where free to use, personal or commercial. There is no need to ask permission before using any assets. Giving attribution is not required for any Assets.

Tiles used in terrain generation

- Kenny Assets - Voxel Pack
    - https://www.kenney.nl/assets/voxel-pack
    - CC0 1.0 Universal License