**Project Name:** Minecraft 2D Lite

**Project Description:** A sandbox game similar to Minecraft / terraria where the player is able to break and destroy blocks. Each block correspond to some type of resource which in turn can be crafted into more superior tools/gear. There will be a combat system with a variety of enemies, all of which have some sort of pathfinding mechanism. The terrain that the player encounters will be completely procedurally generated, and the worlds are all infinite (no limit on how big the player’s world can be). With this sort of freedom, the purpose of the game will be completely up to the player’s own discretion!

**Competitive Analysis:**

* My project will be most similar to (and has been hugely inspired by) games such as Minecraft and Terraria
* Minecraft is programming in Java, rendered with OpenGL vertex arrays. Terraria is based on the .NET framework. Similarly, they all use chunk-based rendering which helps optimize the rendering performance which I will implement in my own project too.
* The game mechanics will be a mash of both Minecraft and Terraria, with it leaning more towards Terraria (since it is platformer/2D based)
* However, I’m implementing enemies (mobs) that will be able to break blocks, path find and dig towards the player, so that no matter the block that is placed down, the player will not be indefinitely safe passively through building structures. This will differ to Minecraft and Terraria, whose mobs wont actively break locks as part of their abilities

**Structural Plan**

* Different files:
  + Player.py – contains all the player and player-like classes (like mobs), that have movement and have some sort of AI
  + Worldgen.py – contains all the world generating functions including caves and terrain
  + Game.py – contains the world initializing functions and classes, also responsible for initiating game mechanics and spawning mobs from player.py
  + Entities.py – contains all the block (superclass) and blocks (inheriting from it) in the game
  + Physics.py – contains all the physics functions to be used on players and mobs
  + Draw.py – the main file that will be run to render the other objects where the user will interact with the tkinter framework

**Algorithmic Plan**

* Terrain generation
  + Ground level – grass level including generation of trees
  + Caves: cellular automaton (quite algorithmically difficult)
  + Voronoi noise to generate block generation underground (e.g stone, dirt, ore groupings)
* Mob pathfinding: bfs (pre mvp) and star/dijkstra’s (after mvp)
  + This is algorithmically the most complex part of my project
  + I will implement the mob enemy pathfinding in different stages of difficulty:

1. “flying” BFS pathfinding mob by taking in graphs (nodes and edges as connections between airblocks) and generating the shortest path taken by flying (not through blocks)
2. “walking” BFS pathfinding mob that takes into account gravity and jumping mechanics to find the shortest path to player – if it exists
3. “tunneling” dijkstra’s/A star mob that is similar to the second mob, but is able to dig through walls, and finds the most optimal path by calculating edge weights between the different blocks (with breaking blocks with the highest feasible edge weight)

* Infinite world generation
  + Generating “chunks” of the world as the player moves across their screen, and saves it into an array
  + Making sure that the game rendering is optimized by only rendering what is on the player’s screen at a given time
  + Also making sure that the game’s simulation is run 1-2 chunks behind and ahead of the player at a given time, instead of the entire generated world
* Saving progress
  + Without using pickle: when saving progress, writing all objects to a text file by converting it into a string with repr, then initializing the appropriate saved progress by reading the file upon starting the game again

**Timeline Plan**

TP0 – TP1

* Drawing the world with simple blocks
* Adding a movement mechanism
* Breaking and placing block mechanism
* Cave generation

TP1 – TP2:

* Implementing **infinite** world generation, optimizing world generation
* Adding crafting and inventory system
* Combat and health system, death screen
* (tackling algorithmic complex sections) finished BFS mob pathfinding algorithm
* More blocks added to the game
* Physics and collision mechanics
* Splash screen

TP2 – TP3:

* Adding dijkstra’s and a star algorithms to improve mob’s pathfinding abilities
* Adding biomes/spawning certain blocks with Voronoi noise
* Saving features (either using pickle or implementing it myself)
* Polishing UI and adding more blocks and items in the game
* Adding sound

**Version Control Plan**

Using Github for version control and backing up. Pushing to github through the VSCode git bash command line.

Additionally using Onedrive as another method to locally store my files.

A screenshot of a computer

Description automatically generated

**Module List**

* Planning to use pickle to save game progress (file handling, writing objects to files) **post mvp**

**Storyboard**

See included file storyboard.png

**TP2 Updates**

* All intended checkpoints were met in the original proposal except for the following:
  + Crafting system
  + More blocks still need to be added
* Problems:
  + Procedural generation needs fixing
  + Efficiency needs addressing, game runs extremely slowly
  + Crash with MVC violation when moving in a specific way
* Solutions to problems:
  + Make things run faster by making the enemies regenerate a new bfs path every 10 ‘ticks’ rather than every ‘tick’ so that the algorithm has to run less per mob
* New idea: using Perlin noise or simplex noise to map the top level terrain rather than sin and cosine functions

TP3 Updates

* Procedural generation fixed
* Game runs a bit faster by updating pathfinding mechanism by mob. Also updated the spawning mechanisms of the mob
* UI updated and polished
* Crafting mechanic added
* Changes to intended: didn’t get to implement A\*/Dijkstra’s algorithm, biomes, sound or saving features