PROJECT PROPOSAL

Project Description
Name: When mom says we have Fortnite at home

Description: This is a 3D third-person shooter game which involves constructing, modifying and destroying panels (walls, floors, ramps, etc) while aiming for different targets.

Competitive Analysis
The first comparison will be made against the renowned game ‘Fortnite’. Both these games are similar in the sense that they are 3D-shooters and have similar building mechanics. However, there will be many differences between the two. While Fortnite is a battle royale, ‘When mom says we have Fortnite at home’ will be an aim-trainer, where the game is won by eliminating pre-programmed targets rather than being the sole survivor amongst 99 other players. Apart from that, the Fortnite-generated map includes pre-constructed buildings, along with the concept of ‘materials’ to build. My game involves a much smaller map with no pre-existing constructions, and materials are not required in order to build.

The next comparison will be drawn against the game ‘1v1.lol’; this was actually the inspiration behind the idea for my game, and the mechanics and gameplay are thus quite similar. Both involve building and shooting, and both utilize a relatively small map. However, there are some differences, apart from how features will be more rudimentary in my game. The main difference in my game is the fact that it is an aim-trainer-based game, where the player shoots targets rather than another player/AI. Another difference is its implementation of projectile motion, where bullets travel based on the gravitational laws of physics as opposed to directly towards the opponent (the hit-scan method, which 1v1.lol uses).

Structural Plan
All of the objects in the game will be made through the use of a class entitled ‘model3D’, in which its constructor involves a points/vertices list, position, rotation, and color. The panels constructed by the player will all be part of a single ‘model3D’ object, Meanwhile, the ‘camera’ which follows the player around is also an object of its own class called ‘camera’, which includes position and rotation as its primary attributes. The 3D engine will be split into 2 functions. The first is ‘strictly3D’, a part of the Model which will take care of in-game 3D transformations of the object. The second is ‘translate3Dto2D’, a part of the View which will translate those 3d-coordinates onto the screen. An object class for guns will also be created, which will involve ammo capacities and projection angles, among others.

Algorithmic Plan
One of the most complex part of the project is the construction of the 3d-engine, which would convert in-world 3D coordinates into 2D-coordinates projected onto the screen. The 3D (in-world) transformation of the vertices in the points list will be done through the use of the function ‘strictly3D’, which will take care of all the translations as well as rotations of the objects. These translated coordinates will then be passed through the function ‘translate3Dto2D’, which

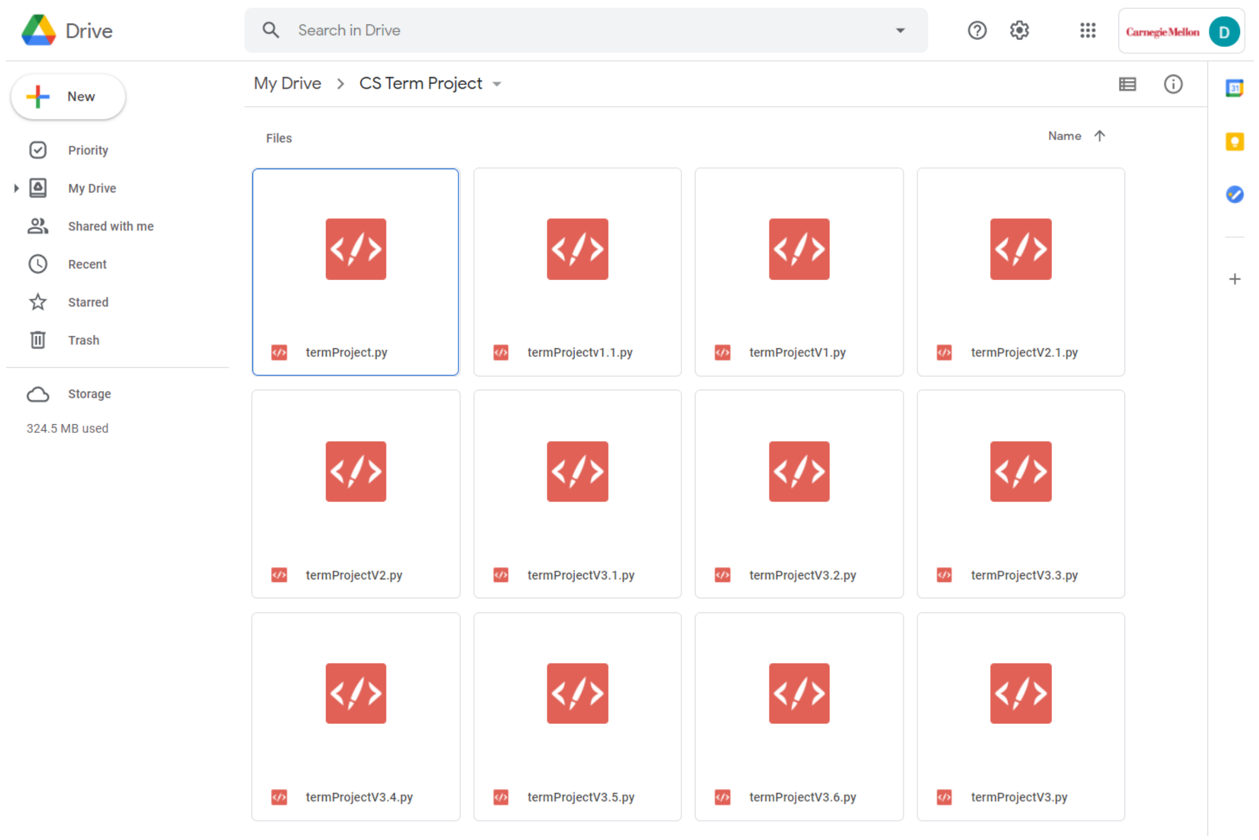
will go through a series of processes and draw the resultant points-converted lines. The implementation of the view matrix (which converts points from worldspace into viewspace) was probably one of the more complex parts, where camera position and look-at direction also had to be taken into account before multiplying different matrices together. Another more complicated facet was the clipping of triangles, which functions to remove triangles outside of the 'viewing plane'. This would account for objects going offscreen or the camera being positioned too near to an object. The algorithm involves checking whether the points of a triangle is inside/outside the viewing plane; triangles with all points outside meant that they were off-screen, triangles with all points inside meant that the whole triangle was in-screen, and triangles in-between would either be rescaled or 'sliced' into more triangles so that they could be properly classified.

Another complex part of the project that is yet to be coded is the addition of projectile motion for bullets. This will necessitate the use of several physics equations involving velocity, angles and time. In addition to this, the collision between the bullet and an object is also something that will need to be taken into account. One of the potential solutions to this is using ray-casting, which would help check if the projectile is actually hitting a physical target or not.

Timeline Plan

Features	**Finished By**
Building Mechanics	28 Apr
Projectile Bullets	30 Apr

Version Control Plan
I am backing up my code by storing every update on a designated Google Drive folder, meaning that my code is safely stored on the cloud. More sizable updates are denoted by a change in the one's place, while smaller changes are denoted by a change in the tenths place (after the decimal).



termProject.py
termProjectv1.1.py
termProjectV1.py
termProjectV2.1.py
termProjectV2.py
termProjectV3.1.py
termProjectV3.2.py
termProjectV3.3.py
termProjectV3.4.py
termProjectV3.5.py
termProjectV3.6.py
termProjectV3.py

Module List

1. cmu_112_graphics
2. numpy

TP2 Update

-New Project Description

Name: When Mom says we have Temple Run at home

Description: This is an endless 3D running game in which a player must strive to survive for as long as possible. The player must avoid stepping on illegal boundaries, which can be done through side-by-side evading or by building ramps to travel along. The player must also win duels against opponents when the opportunity arises, a situation of shoot or be shot. The goal is to earn as many points as possible; these points are earned through survival time as well as the winning of duels.

-Adjusting Building Features

Initially, I planned on involving ramps, floors, walls and roofs which could be connected with one another to create interesting structures and platforms. This was reduced to only ramps to accommodate the gameplay necessities and requirements.

-Adjusting Platform

Since the game was changed, instead of having a finite platform in which the user can move on, the ‘floor’ has become a sort of path which extends infinitely, where the tiles that a user passes are removed, and new tiles are added to the end of the path to create that illusion. In addition, the user is forced to always move forwards as opposed to the free movement in the TP1 program.

TP3 Update
-New Project Name: CubeWalker
-added a leaderboard
-added music through pygame
-added screens(main menu, instructions, etc) to make for better visuals
-removed the lines from the 3d objects
-cube is now seen in front of the ramp (was behind before)
-added visuals for score, timers, etc
-added feature to change projectile angle