Throughout the course of the semester, the overall design of my game engine was kinda complex and specific while making it for my platformer game because even though most of the classes written for my engine were mostly easy to understand and develop, the methods within the classes were sometimes complex in regards to the implementation of those methods. Additionally, the implementation of my classes and engine were overall kinda centered around the implementation of the platformer game I was making with it and the design wasn't properly planned for being made with other types of games in mind. For instance, with my platformer game, I decided to implement actual platforming physics into the game engine through various mechanics like jumping and moving platforms. In addition, my game engine also included additional functionality needed for my platforming game such as pausing, recording and replaying gameplay recordings, and changing the speed of the game. However, when trying to use the engine to create my maze and racer games respectively, since those games didn't require as much functionality as with the platformer game, some of the game engine functionality integrated into the design of the game engine such as pausing along with making and seeing game replays ended up being unnecessary for those games. Because my game engine was more focused and designed around the required functionality that was needed for the platformer game, I had to mainly simplify my game engine design a little with some of the common implementation/methods between the games when creating the implementation for my maze and racer games. When designing the implementation for my maze game, I was able to reuse most of the game engine implementation with my platformer game such as the implementation for my game object model along with my EventManager and ScriptManager classes, though I had to simply some of the method implementation for those classes. With my maze game, I designed it so that the player would have to navigate the maze level in order to reach the goal at the end of the maze, which was represented through my reused Medal object to serve as the main objective of the game. Since I only ended up designing 1 maze level for the game, I decided to make it kinda challenging by having implementation for the game so that if the player were to collide with any of the walls in the maze, the player would then respawn and have to restart at the beginning of the maze. To achieve these things with my maze game, some of the simplifications I had to make with my game engine included simplifying the code for my input() and collision() methods/events. When simplifying the input() method in the EventManager class, I had it so that it only focused on the player movement in the game and coded the implementation so that it simply moved the player either left, right, up, or down while traversing through the maze if the player pressed the respective arrow key through the use of the character.move() statement and passing the respective values that would move the player in the direction of the key pressed. When simplifying the collision() method in my EventManager class, I had it so if the player object's global bounds intersected the global bounds of any of the walls in the maze, the death() method in my EventManager class would then be called in order to spawn the player back to the start of the maze by calling the spawn() method in the EventManager class. While those were the main implementation changes with my game engine, some minor ones included changing the name of the Platform class to now be the Wall class and making small changes with the implementation for the spawn() method. Besides those changes, the other game changes I ended up making with the maze game included setting up new textures for my game objects along with adjusting the size and positions of those objects when rendering them in the game. When designing the implementation for my racer game, I designed it so that the player would have to navigate through the track in the game in order to reach the end of the track, which was again represented with my reused Medal object to serve as the main goal of the game. Like with my maze game, I also ended up only creating 1 track/level in the game, so I tried to add some extra challenge to it by having some game implementation where if the player touched any of the walls on the track, the player would respawn and restart at the beginning of the track. Because most of the things I was trying to do for my racer game was essentially the same as what I was doing with my maze game, I ended up basically reusing the game engine design from my maze game to incorporate into my racer game. As a result, most of the game engine design changes with my racer game were essentially the same as the ones I made with my maze game, which included simplifying the code for my collision() and spawn() methods/events in a similar fashion with how I did it in my maze game. Even though I wanted the player to still be able to move left, right, up and down similar to what the player could do in terms of movement in my maze game, I also wanted to change the movement a little so that the movement was instead automatic and that pressing the left/right/up/down arrow keys would instead change the direction of the automated movement. In order to achieve this effect in my racer game, when simplifying the input() method in my EventManager class, I first created new boolean variables called moveLeft, moveRight, moveUp and moveDown, where they would be set to true if their respective arrow key was pressed or false otherwise before then having checks on those variables to see which one was true to then call the character.move() statement with the respective values that would continuously move the player in the direction of the key pressed. Besides that main implementation change with my input() method for my racer game, the only other main changes made for the racer game were setting new textures for my in-game objects along with adjusting the size and position of those objects when rendering them on the screen. Overall, the game engine designs for my maze and racer games were pretty much the same and both ended up being simplified designs of my game engine design for my platformer game. When running "diff platformer.cpp maze.cpp | grep "^>" | wc -l" and "diff platformer.cpp racer.cpp | grep "^>" | wc -l" to determine the percentage of code that was different between my maze and racer games with my platformer game, I found that my maze game code was different by roughly 29% and that my racer game code was different by roughly 31% when comparing them with my platformer game code. Going into making the implementation for my maze and racer games based on the game engine design I had for my platformer game, I honestly thought that only 40-50% of my game engine code would end up being reused in those games since I was mostly designing it primarily for the functionality needed in my platformer game, so I'm glad to see that I was able to reuse about 70% of the original game design for my platformer game for those 2 games. The roughly 30% code difference between my maze and racer game codes with my platformer game code was likely mostly due to the simplified code I had in my maze and racer games for the input() and collision() method functionality along with not including code for some functionality that was implemented in my platformer game such as pausing and changing game speeds. Seeing how I ended up having to simplify the design of my game engine from my platformer game when creating the maze and racer games, if I were to go about designing my game engine differently from the beginning, I would likely try to have it more simplified and broad in general from the start such as the game engine designs I ended up having for my maze and racer games. However, since I was able to reuse most of my game engine code across all 3 games I created, I would probably still maintain some things such as the classes I made for my game engine along with the general abstract implementation for the methods in those classes. Overall, I was pretty satisfied with the evolution of the design for my game engine and the amount of reusability it ended up having with the other games I created, even though I probably could have done a better job with making it more applicable for other types of games.

New Game Texture Assets Links

Maze Character Texture: https://opengameart.org/content/die-for-the-empire-ships

Red/Gray Brick Wall Textures: https://opengameart.org/content/brick-wall-0

Racecar Character Texture: https://opengameart.org/content/2d-car-sprite-10