Sam Blackwood
Cole Kisicki

OOAD Project 7 - Final Project

Project Title: The FNCD Race Game

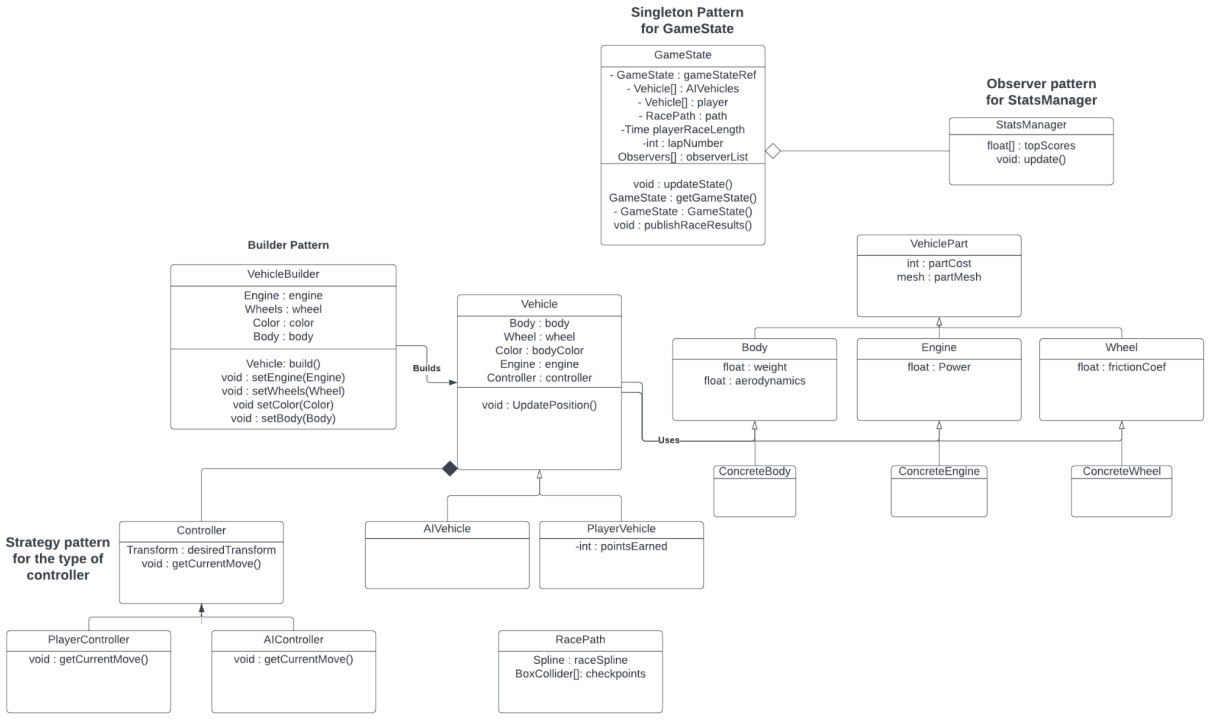
Final State of System Statement

Features Implemented: we implemented vehicle customization, a leaderboard, and a difficulty slider to go along with the racing aspect of the game.

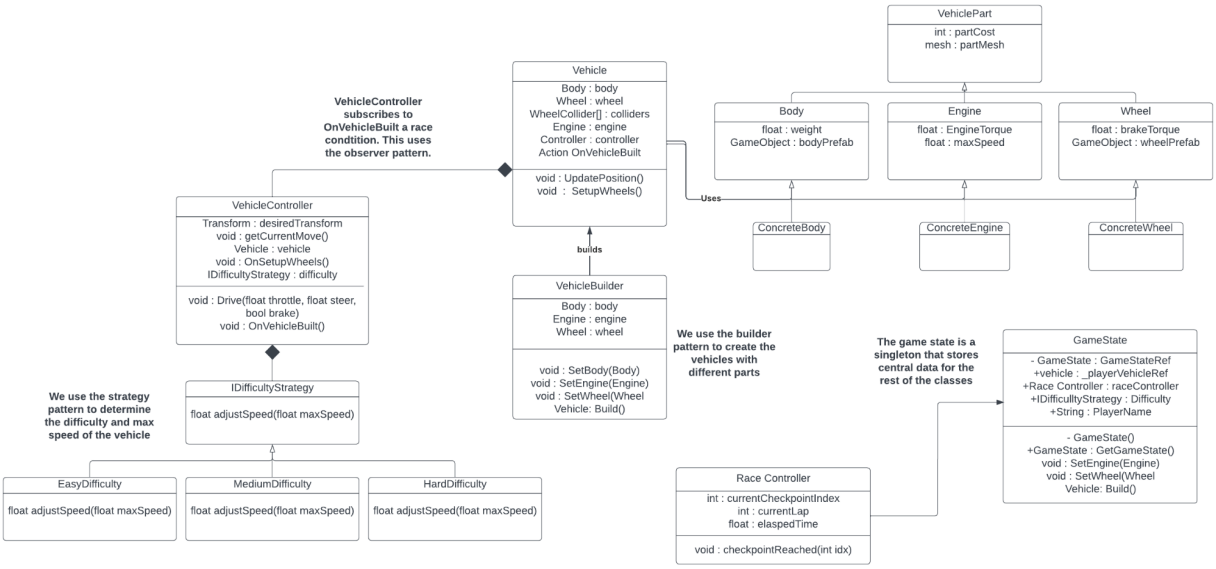
Features Not Implemented: the primary feature that we did not implement was allowing the user to race against other AI vehicles. This proved to be too much work given the time constraint for completing the project. Another minor feature we did not implement was the UI on the race screen that shows the race time and lap number. If we had a few more days, we certainly could have implemented this. We also did not get around to creating a pause menu in the race scene, but this was not one of our original features we intended to implement anyways.

For the most part, the reason that we did not implement these features was because Unity has a steep learning curve and it took more time than we had anticipated to familiarize ourselves with the Engine and how to build out certain elements.

Original Class Diagram



Final Class Diagram



Comparison Statement

- Our final program architecture is very similar to what we had originally planned with a few key differences. To start we were unable to implement AI in the time we had so in order to use the strategy design pattern, we decided to make the difficulty (in the form of the maximum vehicle speed) the strategy instead of the type of controller used for a vehicle. We also ended up using the observer strategy in a different location than we originally had planned to. Instead of using an observer to subscribe to the stats produced in the game state, we used it to eliminate a race condition when the vehicle controller and vehicle are created. Since the vehicle controller has to reference all the parts of the vehicle, if the vehicle has not finished spawning in the world before those references are accessed it causes the game to crash. To eliminate this we used an observer in the vehicle controller that subscribes to an event in the vehicle that tells the controller that it's safe to access all elements of the vehicle after they have finished instantiating.

Third Party Code vs. Original Statement

Unity Engine: Unity supplies some built in functionality and code scaffolding to expedite the development process. We also used its built-in observer implementation.

Youtube vehicle movement tutorial: This tutorial was used to make the wheels of the vehicle rotate like they should using the wheel colliders.

YouTube Leaderboard Tutorial: used this tutorial to help build the functionality to save entries to the leaderboard. Link: https://www.youtube.com/watch?v=iAbaqGYdnyI

The rest of the code developed was our own code.

Statement on the OOAD Process

1. Among the design elements we worked out in Project 5, the UML diagram proved to be the most valuable to us. It gave us a clear sense of what classes we needed to implement and how they would interact with each other. Once we started programming, we knew exactly what files we needed to create and that helped speed up some of the development.
2. The wireframe was probably our second most helpful design element in our project. We referenced it several times to get a clear vision for how we would design and implement some of the UI screens.
3. In our developing phase of the project, we noticed that it is rather difficult to implement classes in the way that we had been doing in Java throughout the semester. This is because in Unity, you attach a script to a Game Object to give it a behavior. In a way, Game Objects act as their own classes. Therefore, abstract classes were not as useful and it was harder to implement the design patterns in the way that we had intended.