**Team Members:**

Ahmad *- Designer/Artist, Level Designer, Programming*

Diego *- Input and Movement, Programming, Main Features*

Adel *- Functionality, Effects/SFX, Damage Event Behavior*

**Game Information:**

**Title:** Retrowave Rider

**Tagline:** Chase the Neon Dream

**Genre:** Subway Surfer-style Runner Game

**Target Audience:** Teenagers

**Goal:** Reach the sunset at the end of the run while avoiding obstacles

**Major Mechanics Descriptions**

The player is under constant movement, where the car that the player is in control of cannot be stopped or slowed down. The speed of the car will be slowly increased over time, making it more difficult for the player to stay alive.

Powerups will pop up occasionally on the path of the player and the player can choose whether to pick them up or not by running them over with their car. Powerups include a flying mode, where the car can temporarily fly above all the obstacles and the map below the player, in order to advance further. A second powerup will be a speed-up powerup, where the player will reach such high speeds that they enter a “hyperspace” style tunnel that brings them forward on the map. Finally, our last powerup includes a Wave slingshot. In order to add a special effect that matches our game style, we will create a literal “retroWave”, in the style of the environment around the player that slingshots the player forward.

There will also be world events occurring that will hinder the player’s ability to move forward. Some will be obstacles, some will be dynamic events, all will damage the player until the player can no longer drive, therefore forcing the player to avoid these obstacles and try their best to not take any damage. An obstacle that will be easy to avoid will be lasers that will be spread across the course in some sections. However, this will deal 100% damage and split the player’s car in half. Other obstacles may be harder to avoid, but will cause less damage. For example, there will be some parts of the map that will have neon fireballs shooting at your car, which will damage your car and deal a certain amount of health damage. There will also be barricades, cop cars, trains and other props which will deal small bits of damage every time that you hit one before allowing you to continue. There will be neon puddles that will be present in some parts of the map. These puddles will slow down your car, before you suddenly regain all your speed in one moment, therefore making it hard for the player to control right after the effect is done and potentially hitting these stationary obstacles like the cop cars, which will deal damage to the player.

There will also be collectibles spread across the map, similar to the coins in subway surfer that give the player a suggested path and allow the player to collect something while they are making their way through the level. The use for these coins are still undetermined, potentially we will let it be to purchase different cars.

Once the player makes it to the end of the level, a.k.a the sunset, they will be greeted with a game won screen and further options to continue.

The level will not be the same for each playthrough, we will make blocks where the order can be interchanged for a specific length that we desire, therefore making each playthrough randomly generated based on premade blocks.



**World/Level Design Draft**





**Task Breakdown for Next Deliverables**

* Creating a level prototype - **Ahmad**
* Add input functionality - **Diego**
* Design the level for immersive visuals (road, sunset, the sides view…) - **Ahmad**
* find/create a car for the game - **Diego & Ahmad**
* Creating Health+Damage System - **Diego**
* Programming car movement - **Diego**
* Programming Powerups - **Diego**
* Implementing Car modes - **Diego & Adel**
* Creating obstacles described earlier - **Diego & Adel**
* Programming help advantages ( time slower, turbo?,...) - **Adel**
* Make effects for each obstacles or advantages - **Adel**
* Find & Implement sound - **Adel**
* Design and implement UI menu system ( start game, options, settings) - **Ahmad**
* Design a HUD with details about the game – **Ahmad**

[Atlassian Jira](https://diegoarriaga.atlassian.net/jira/software/projects/CCS/boards/1?assignee=unassigned%2C712020%3A6761e318-c11d-42a8-af7d-199245192663&atlOrigin=eyJpIjoiM2E2OTM4ZGU0ZjkwNGZlNWE2MzQwNThkODQ3ODZiMTQiLCJwIjoiaiJ9)

**Design Patterns Utilizations:**

**Singleton Pattern**

* **Game Manager & Configuration Settings**
  + Manages game-wide states like score tracking, difficulty scaling, and pause/resume.
  + Handles background music, sound effects, and global settings like graphics and controls.

**Adapter Pattern**

* **Cross-Platform Input Handling**
  + Adapts different input methods (touch, keyboard, gamepad) into a unified input system.
  + Enables seamless compatibility between mobile and PC versions of the game.

**Factory Pattern**

* **Power-Up and Obstacle Spawning**
  + Dynamically creates different types of obstacles and power-ups without modifying existing code.
  + Ensures proper instantiation based on difficulty scaling and player progression.

**Decorator Pattern**

* **Character & Vehicle Upgrades**
  + Applies enhancements like speed boosts, shields, and magnet effects to the base runner without altering its core class.
  + Allows stacking of power-ups while maintaining modular code.

**Observer Pattern**

* **UI Updates & Event Handling**
  + Automatically updates UI elements (score, power-up timers) when relevant game states change.
  + Notifies enemy AI or environmental effects when specific player actions occur.

**MVC Pattern (Model-View-Controller)**

* **UI and Game State Management**
  + Separates core game logic (Model), visual representation (View), and user input handling (Controller).
  + Keeps UI elements (menus, HUD) separate from gameplay mechanics for maintainability.