C++ Programming Project Plan: Micro Machines

The goal of the project is to create a simple 2D version of the Micro Machines game, inspired by the Nintendo hit game series Mario Kart. Mario Kart is characterized by not only the characters of the series being playable, but also its several power-ups, which alter the state of the player and of the chosen map.

Several of the graphical properties and features of the game will be inspired by characters in the Mario Brothers –series. However, the original series has never been played from an up-perspective, providing a fun differentiating twist for our project.

Scope of the project

The plan is to implement a local, 2 player game with split screen. The game will include several different cars and terrain types. Playable maps will include 4 premade options and a random map generator.

The Cars are controlled through the arrow keys and WASD-keys. Power-ups are deployed with the E and Shift keys respectively.

Cars will have physical properties which affect the handling of the car, such as

- Acceleration, which will affect the pace at which the car gains velocity.
- Max Speed, which will state the maximum velocity the car can accelerate to.
- Weight, which will affect the interactions between cars when they collide with other cars and walls.
- Maneuverability, which will state how quickly the car's direction can be altered.

To mix up the game, it will include – like the game it is inspired by – power-ups aquired from item blocks, such as

- star (boosts acceleration and speed for the user, causes the other player to lose speed for a short time)
- ghost (allows the user to drive through others without affecting them)
- mushroom (boosts the speed of the user for a short time),

and weapons, such as

- Banana peel (causes the player that drives into it to stop and spin for a short time)
- Trap (causes the player that drives into it to lose their unused power-up or weapon to the player owning the trap)

The player can hold a single power-up or weapon at a time.

The game will include different car handling on different terrains like:

- Asphalt: the standard terrain type where the car will maneuver and move its best.
- Ice: all the driving commands will be delayed
- Grass: grass will cause the car to slow down
- Item Tile: adds an item/weapon to inventory
- Wall: causes a collision and prevents driving through

The game will include sound effects such as:

- Acceleration
- Stopping
- Crashing
- Picking up an item
- Using an item
- Driving into a weapon

Libraries

We will utilize the libraries recommended in the project instructions. SFML will be utilized for graphics and sound effects. Box2D will be used to implement the simple physical properties of each sub class

SFML provides methods for accessing the keyboard and mouse commands so it will be utilized for selecting the car and map before each game as well as controlling the car. It is also capable of drawing graphical objects to a game window, based on a position.

Box2D provides options for basic physical concepts such as rudimentary gravity, friction, density and a take on the physical understanding of vectors. Especially the b2vec2-class will most definitely prove to be highly adaptable for our purposes as it seems to provide an excellent way to access and modify the direction and velocity of our Car objects.

Cars and walls will be Box2D rigid bodies with shapes and fixtures. The map will be a 2DBox world in which collisions and interactions between cars and tiles can take place.

Class Structure

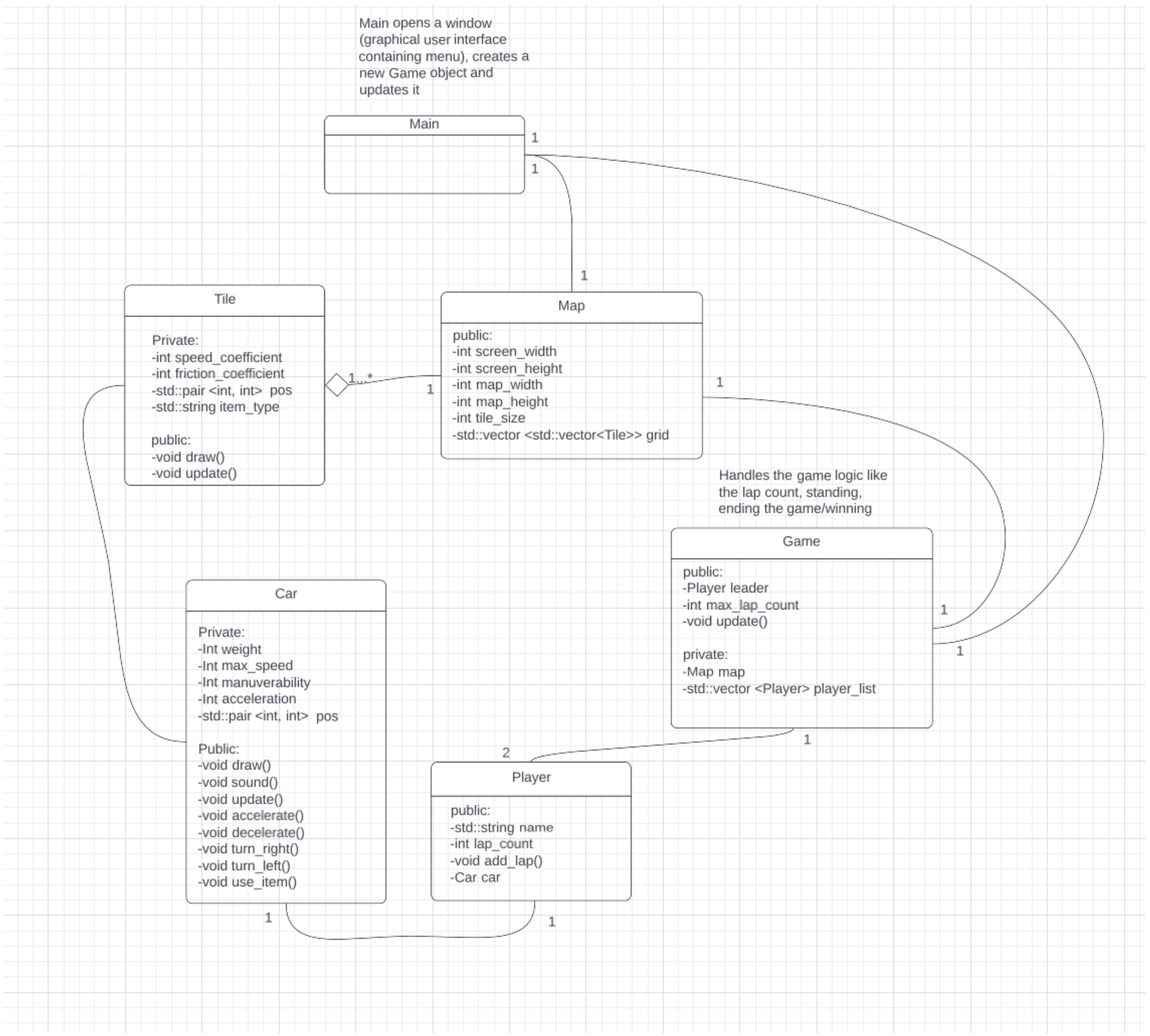

Fig 1. Basic class structure and relationships

Fig 1 contains the planned class structure of the program. As with any other C/C++ based program, launching the game calls the Main-function, which will open the graphical user interface, which will include a menu for selecting the Map and a Car for each Player. The main will start an instance of the Game which will include 2 Players and the selected Map.

Game-object will keep track of the basic game logic such as the lap count, the player standings and who wins. The Game-object also updates the state of the current game at a certain refresh rate.

The Map consists of Tiles of a fixed size, which keep track of whether a Car object is on them, based on a position value. The information of the racetracks are stored on the Maps as a 2-dimensional array. The Map will contain a whole racetrack and every object that resides in it.

The Tile objects contain properties defining the terrain and altering the Cars' movement. They also store information about possible items placed on the Tiles as alluded to in Fig 1.

The Player class holds information for the name, lap count and the type of car that the Player has. Each Player object is connected to a Car object. The lap count can be increased with a member function of the class.

As shown in Fig 1. the Car class holds information about the specific car chosen that affects how it interacts with the game world. The Car class also holds the controls of the car which will change its course and speed as it moves. Its update function calculates the car's next state. Its draw function is used on every loop iteration to make the car appear in its new state.

Responsibilities:

Name	**Responsibilities**
Aaro Saastamoinen	Game, SFML, Box2D. Initial goal is to create a Game (have a functional constructor) based on parameters given in Main
Alex Lietsala	Main, SFML Initial goal is to get to a point where one can open a window and start a game instance.
Heikki Penttinen	Tile, Car (Box2D physics), Player, sound effects (SFML) Initial goal is to create some sort of tiles, cars, and players that allow for testing.
Jussi Lehtonen	Map, Box2D. Initial goal is to create a flexible 2-dimensional array which stores tiles.

*The responsibilities given at this stage are not set in stone and might change down the line as we begin working on the project.

Schedule:

Week number	**The goal**
Week 45	Planning the submission, getting acquainted with SFML and Box2D and seeing if the responsibilities are spread evenly.
Week 46	Plan review with assistant The goal is to at very least have a window that spawns a car which can move to some extent and maybe interact with the world and its tiles.
Week 47	The goal is to be able to do one lap of a very simple track with one player.
Week 48	Demo with assistant: The goal is to have multiple tracks and working interactions with the game world.
Week 49	Deadline: The goal is to submit a finished product on time.

*Weekly meet up on Tuesdays at 16:00