Micro Machine Project Documentation

C++ course project

Huyen Do, Quan Hoang, Huyen Linh Nguyen, Duy To

December 11, 2023

1 Overview

1.1 What the game do

1. Basic features
   - Basic gameplay with simple driving physics
   - Multiple players on the same computer
   - Multiple tracks loaded from files
   - Game objects which affect gameplay
2. Additional features
   - Split screen
   - Different kind of vehicles
   - Sound effects

1.2 What the game does not do

1. Additional features
   - Different performance of vehicles on different terrains: The feature was dropped due to shortage of time and complexity.
2. Advanced features
   - Random generated maps: The feature was dropped due to shortage of time.
3. Notes
   In addition to some of the dropped features, we also simplified our game to have 2 default characters and 2 maps. Nevertheless, we always keep in mind scalability when building the project and we built a .json file to manage our resources. Therefore, new maps can be added at ease with less than 10 minutes of code modification.

2 Software Structure

2.1 Overall Architecture

2.1.1 Game Logic

The class structure can be divided into three main components: Game logic, Game manager, and GUI.

- **Game Logic:** Built upon Box2D - a 2D physics engine. The game logic utilizes the `b2Body` class from Box2D to define several classes that represent basic its components:
  - `Vehicle`: An abstract class used to define controllable objects in the game.

  - `Obstacle`: Represents obstacles in the game.
  - `OutsideArea`: Models different areas of a game map, such as starting lines and checkpoints.

To represent racing tracks, we define:

  - `Checkpoint`: Extending from `OutsideArea`, multiple `Checkpoint` objects represent a racing track. This class keeps track of all vehicles that pass through this area.
  - `StartingLine`: Extending from `Checkpoint`. Besides representing the starting point of a racing track, this class keeps track of all player points.

Additionally, classes that help represent items to make the game more compelling are defined:

  - `Timer`: A helper class that destroys itself after a certain number of simulation steps.
  - `Buff`: Derived from `Timer`, this class can change the physical properties of a vehicle.
  - `Collectable`: Represents a body that can be placed into the racing track.

Finally to model to interaction between various object of the game these class area defined:

  - `UserDataPointer`:A helper class that used to cast and maintain various pointer used.
  - `ContactListener`:Extending from Box2D, this class keep tracks all collisions that happens and react accordingly.

All of these classes are hosted in the `World` class, which uses a `b2World` object from Box2D to run the simulation and keeps track of various variables of the game.

2.1.2 Game Manager

The other part of the class structure concerns how to manage player inputs, allocate memory, and read game data from files:

- `Resource Manager`: Reads data from a JSON file.
- `Game`: This is where the game loops take place. This class combines all the components of the game and serves as a means of communication between game logic and the UI.

2.1.3 GUI

The GUI of our game is built upon SFML, utilizing the provided Graphics/`View` class of SFML to render game menus as well as gameplay. Using SFML/Sounds to implement Sound Effects.

2.2 Class diagram

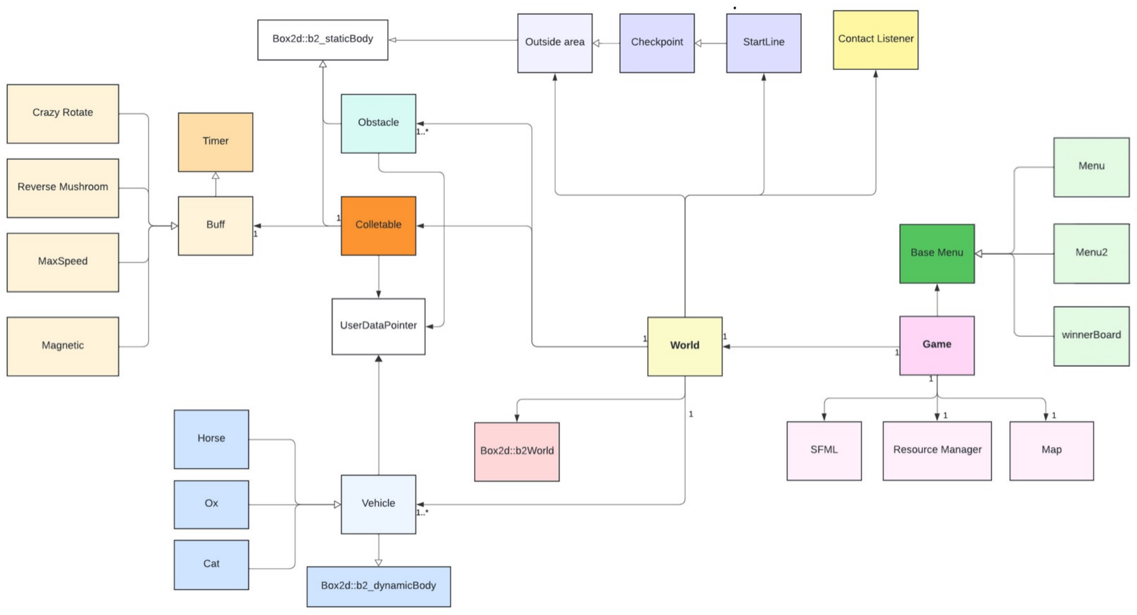

Figure 1: UML Class Diagram of Micro Machine

2.3 Interface with external library

- Box2d: Game physics
- SFML: GUI and sound effect

3 Instructions for building and using the software

3.1 Building instruction

1. Install SFML to your local computer
2. After cloning the game from Git, delete the default build folder.
3. Enter the following command in your terminal:
   (a) mkdir build: Make a new build folder
   (b) cd build: Switch to build folder
   (c) cmake ..: Run cmake file
   (d) make: Build the game
   (e) ./MicroMachine: Run the game
4. After the first build, follow from step (d) to restart the game

3.2 Using the software

After building the game, a menu pop up. Choose between one player and two players mode using Up-Down arrows on your keyboard and press Enter to continue. Next, choose your map and press Enter to start the game. Control your vehicle using Up-Left-Right arrow keys and W-A-D keys. The player can activate their superskill by pressing Enter. Race around the track, the first person to reach 3 rounds win the games. The game has a 15-minute time limit, after time out, the session will end automatically.

4 Testing

The modules in the software were tested using GoogleTest testing framework. The unit tests and the responsible CMakeLists.txt file are included in the tests/ folder.

4.1 Test of Timer class

- Involved classes: Timer
- Test File: timerTest.cpp
- Testing method: Unit testing. A Timer object was constructed, then its Tick() function was tested.
- Results: 1/1 Test passed

4.2 Test of Vehicle class

- Involved Classes: b2World, b2Vec, sf::Fixture, Vehicle
- Test File: vehicleTest.cpp
- Testing method: Unit testing. A b2World object was constructed to enable the construction of Vehicle objects. For each test, a new Vehicle object was constructed. The test suite tested two methods of class Vehicle: GetPosition(), and ToggleForce().
- Results: 2/2 Test passed

5 Work log

5.1 Division of work

1. Duy To
    - Obstacle class, multiple player, outside area implementation
    - Split screen
    - Camera view
    - Testing
    - Cmake file
    - Project plan and weekly meeting host
    - Project documentation
2. Huyen Do
    - Game class, resource manager implementation
    - Main GUI design and implementation/render: Map, menu; character, obstacle, collectable design
    - Handle conflicted merge request
    - Project plan and weekly meeting host
    - Software documentation
    - Presentation
3. Linh Nguyen
    - Collectable, buff, contact listener and real-time timer implementation
    - Sound effect
    - Project plan and weekly meeting host

- Software documentation
- Main responsibility for project documentation
- Presentation

4. Quan Hoang
    - Vehicle, buff, contact listener, custom pointer, checkpoint implementation
    - Memory management: Exit button
    - Project plan and weekly meeting host

5.2 Weekly work log

Date	Member	Progress	Time
Week 1	All members	- Finish game logic - External library decided - Basic project structure - Finish writing project plan	8 hours each
Week 2	Duy To	Cmake file, obstacle class	8 hours
	Huyen Do	Game class, experiment with GUI	9 hours
	Linh Nguyen	Collectable class and contact listener	8 hours
	Quan Hoang	Vehicle class	7 hours
Week 3	Duy To	Cmake file, obstacle class	9 hours
	Huyen Do	Game, world class, render collectable	9 hours
	Linh Nguyen	Collectable class, contact listener, timer	9 hours
	Quan Hoang	Vehicle class, contact listener, custom pointer	15 hours
Week 4	Duy To	Obstacle class, outside area, multiple player, split screen	12 hours
	Huyen Do	Rendering, multiple track loaded from files, time GUI	12 hours
	Linh Nguyen	Buff and timer class, forest sound effect	12 hours
	Quan Hoang	Improve vehicle and implement buff class	10 hours
Week 5	Duy To	Formatting, outside area, checkpoint, project documentation, testing	15 hours
	Huyen Do	Game menu, GUI for ocean map, software documentation	13 hours
	Linh Nguyen	Implement more buff, ocean sound effect, software and project documentation	12 hours
	Quan Hoang	Improve vehicle, checkpoint, memory management	13 hours

Table 1: Work log