

INFO-0004: Object-Oriented Programming Projects

Project 3: Drunk Run 2.5D - Report

Goffart Maxime
180521

Joris Olivier
182113

Academic year 2020 - 2021

1 Architecture

In this section, we will present the classes we build in this project as long with their responsibilities and their interactions.

Our classes and their interactions can be represented by the following UML static diagram¹:

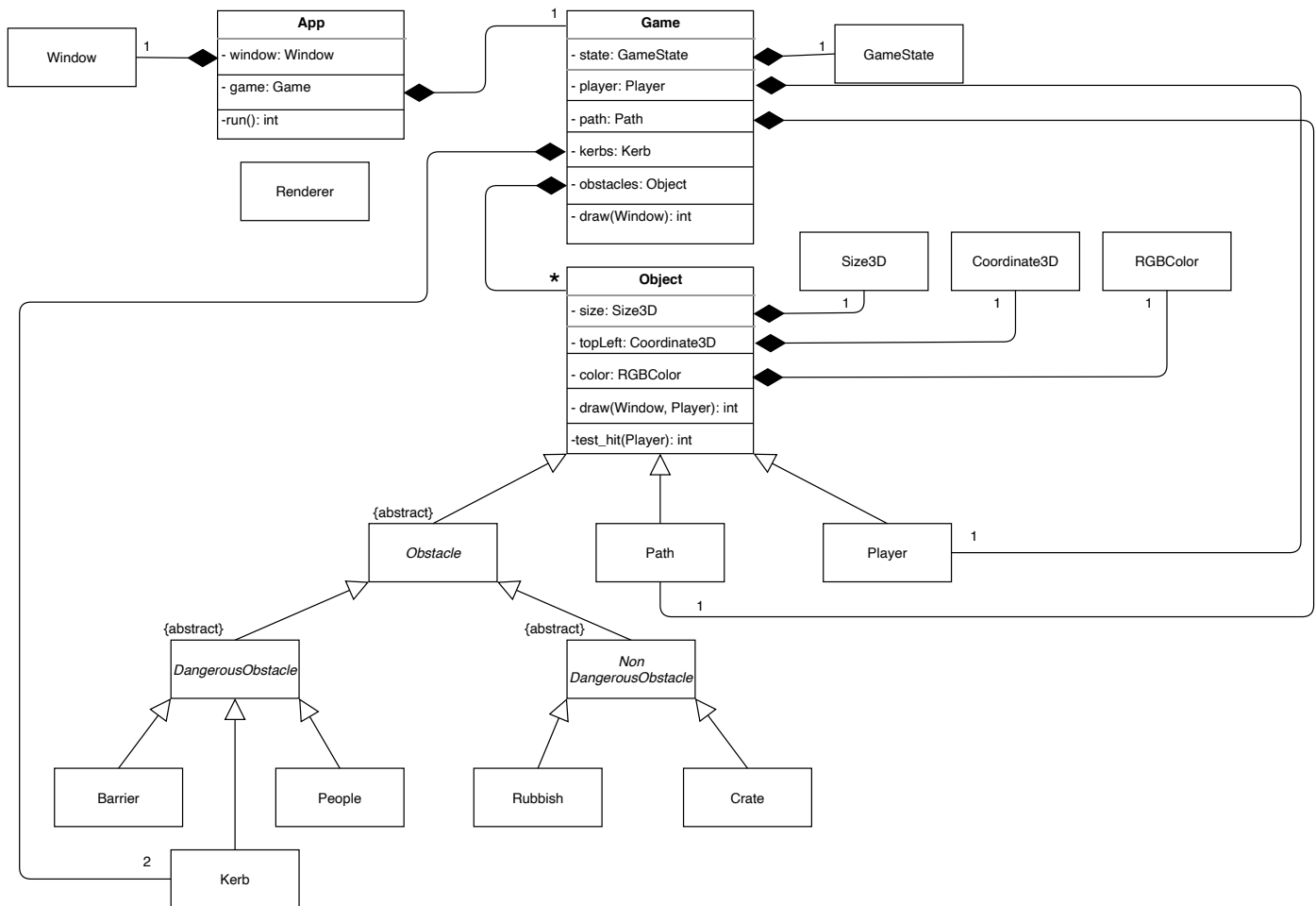


Figure 1: UML static diagram

The class `Window` is a wrapper around a `SDL_Window` which stores some information about the window it is representing as long as a few functions that are wrappers around SDL functions. These functions allow to draw a line, a rectangle, or a text on the window.

The class `App` represents the application. It is responsible for running the game, handling the events, and quitting the game properly at the end.

¹We purposefully did not represent every field and function of the classes because we wanted to keep the diagram as general as possible. You can look at the documented header files of the classes for more information on their fields and functions.

The goal of the class **Renderer** is to compute the coordinates of the perspective of a given point and to calculate the coordinates of the point on the window coordinate system.

The class **Game** is representing the game. It is responsible for the logic of the game and for maintaining references to all the elements needed by the game. It has a lot of responsibilities with the most important being to draw the game on a given window, test if the player hit something, and generating random obstacles.

The class **GameState** is responsible for maintaining the state of the ongoing game. It is remembering if the game is still ongoing and the distance travelled by the player.

The class **Object** represents every object of the game. It stores information about an object, including its size, position, and color. These properties are respectively represented by the class **Size3D**, **Coordinate3D**, and **RGBColor**. It also has a function to draw the object on a given window and another one to test if the player hit the object.

The class **Path** represents the path on which the player is walking.

The class **Player** represents the player (Cymi) of the game.

The abstract class **Obstacle** represents every obstacle of the game. It is further divided into **DangerousObstacle** and **NonDangerousObstacle** which respectively represent obstacles on which the player will trip and obstacles on which it will not trip.

The class **DangerousObstacle** is itself specialized into **Barrier**, which represents a barrier, **People**, which represents a person standing on the path, and **Kerb**, which represents a kerb on the side of the path.

The class **NonDangerousObstacle** is also specialized in **Rubbish**, which represent a rubbish left by the people on the path, and **Crate**, which represent a crate on the path.

Each class has two files associated to it. The names of the files associated to a given class are the name of the class then **.hpp** or **.cpp** depending on whether you are searching for the interface or the implementation of the class.

2 Design

How we solved non-obvious specific problems (generation of obstacles, drawing of side faces, etc). Justification behind your implementation choices.

3 Feedback about the assignment

In this section, we will explain the main difficulties we encountered during the development of this project as long with an approximation of the time we spent on this project.

Undoubtedly, the hardest part was the rendering process. We encountered some troubles to take the field of view into account when drawing the obstacles on the window. The other parts of the project were not that hard once we had our architecture figured out.

Regarding the time we spent, Maxime spent approximately 15 hours on the project while Olivier spent approximately **ADD TIME SPENT** hours on the project