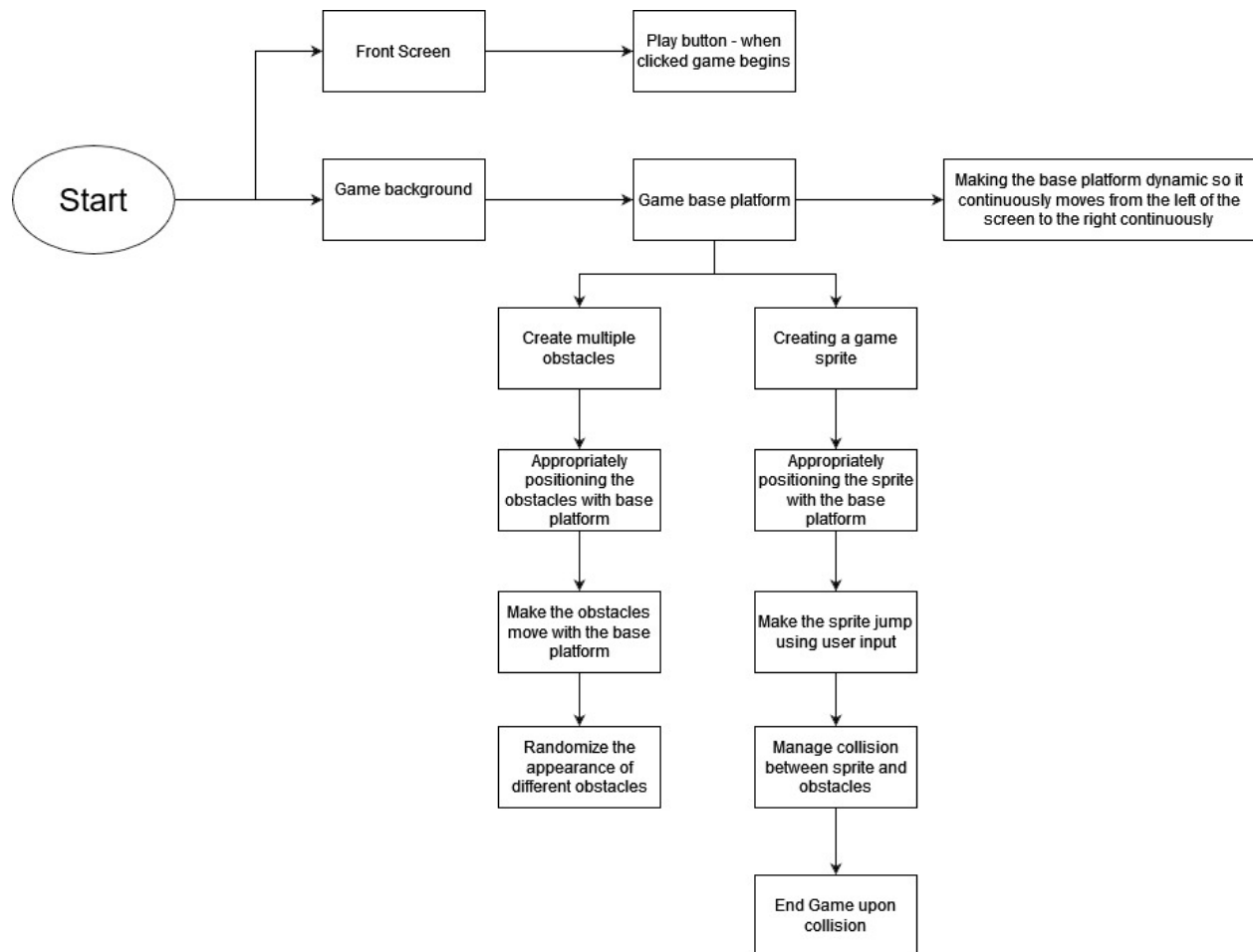


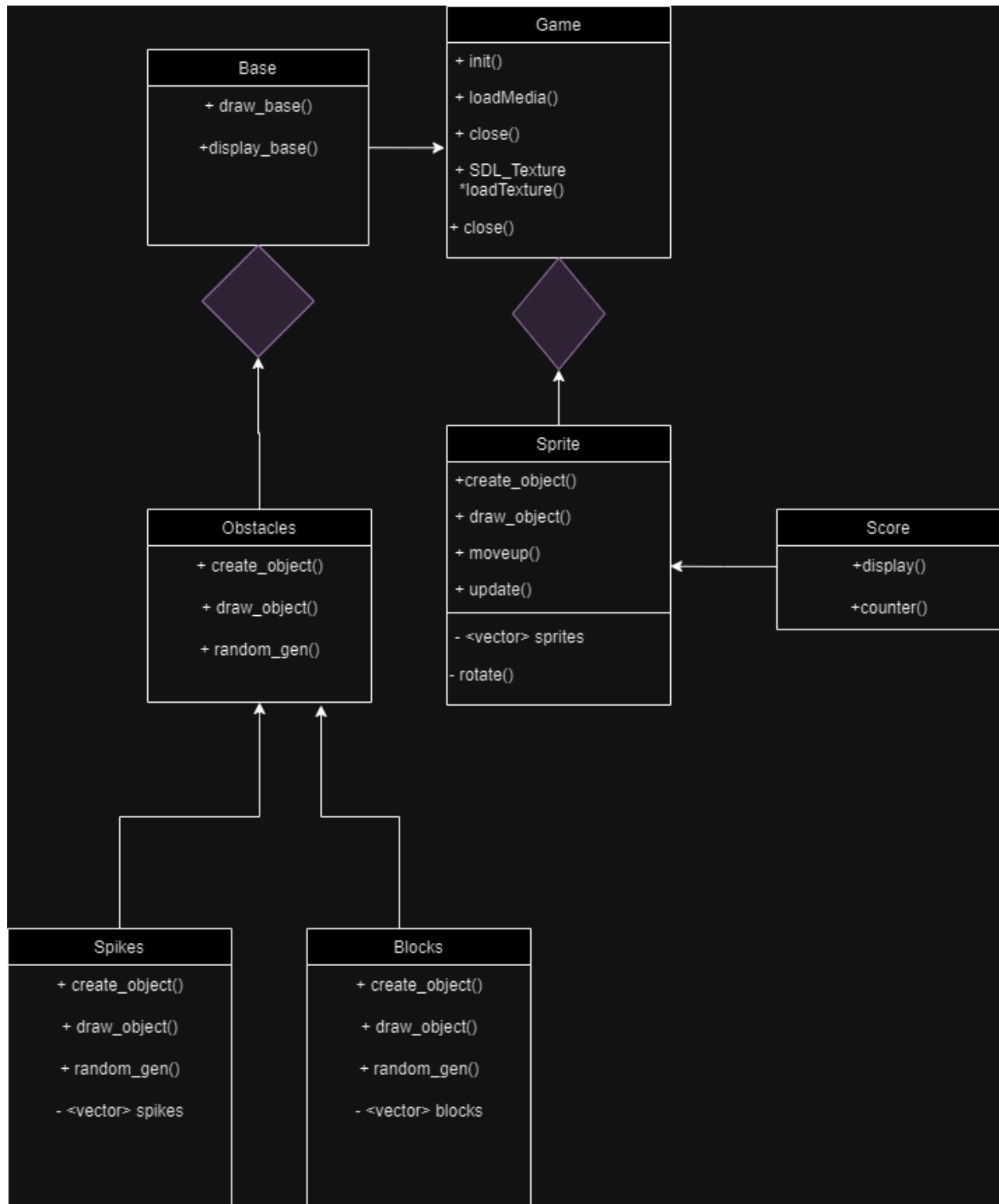
UML and Project Synopsis

Project Overview

A diagram describing the flow and aspects of our project (game).



UML Diagram



Description

Base Class:

The Base class serves as a fundamental component. It encapsulates basic functionality for drawing and displaying a base element. The **drawBase()** function is responsible for rendering the base, and **displayBase()** handles the presentation of the base element.

Game Class:

The Game class plays a pivotal role in managing the overall game state. It is associated with the Base class, indicating a relationship where the Game class utilizes functionalities from the Base class.

Functionality within the Game class includes:

- **init():** Initializes the game, setting up necessary components.
- **loadMedia():** Loads game media, such as textures or images.
- **close():** Handles the closure of the game, freeing up resources.
- **loadTexture():** Loads an SDL texture, potentially used for rendering.

Sprite Class:

The Sprite class is intricately tied to the Game class through a composition relationship, suggesting that a Sprite is a vital part of the Game.

Functionalities encapsulated by the Sprite class include:

- **createObject():** Creates a sprite object.
- **drawObject():** Renders the sprite object on the screen.
- **moveUp():** Initiates an upward movement of the sprite.
- **update():** Manages the update logic of the sprite.
- **rotate():** Enables rotation of the sprite.

Obstacle Class:

The Obstacle class inherits from the Base class, indicating that an obstacle is a specialized form of the base element. This relationship implies that the Obstacle class inherits basic functionalities from the Base class.

Core functionalities of the Obstacle class encompass:

- **createObject()**: Generates an obstacle object.
- **drawObject()**: Renders the obstacle object on the screen.
- **randomGenerator()**: Generates random obstacles.

Additionally, the class has a composition relationship with the Base class, implying that an obstacle is composed of or contains a base element.

Spikes Class:

The Spikes class inherits from the Obstacle class, signifying that spikes are a specialized form of obstacles. This relationship implies that the Spikes class inherits functionalities from the Obstacle class.

The class maintains a vector (spikes) to manage multiple spike instances.

Blocks Class:

The Blocks class inherits from the Obstacle class, signifying that blocks are a specialized form of obstacles. This relationship implies that the Blocks class inherits functionalities from the Obstacle class.

The class maintains a vector (blocks) to manage multiple block instances.

This updated description reflects the additional details about the inheritance relationships between the Obstacle, Spikes, and Blocks classes, as well as the composition relationship between the Obstacle class and the Base class.