Report A

## Game Concept

For my assignment I have create a 2D strategy / puzzle game. In the game you have a battle field that is split up into a grid. On this grid you will see blocks everywhere with a number on, these numbers represent the health of that block. You will have a small arsenal of guns and turrets that you can place on to the grid to attack the enemy. All guns will only shoot once so the guns you place should be strategic and the smarter your placement the more points you can get. All the guns only shoot once when the start button is pressed, so when placing guns, you must think about its placement the blocks around it and how close it is to other guns otherwise you can perhaps destroy your own guns in the process.

## Analysis

The game is quite simple in design, but I know I am going to need a few different base classes to help flesh out the guts of the game. Fundamentally the game is going to need a game loop, window manager, state manager and resource manager. And from the game I’m going to need to have a TileMap class for the battlefield, a Button class for various buttons in the game, a Label class for things like the score and other information on the screen, and a Ball class specifically for the bullets that will be shot out of the guns when you hit play.

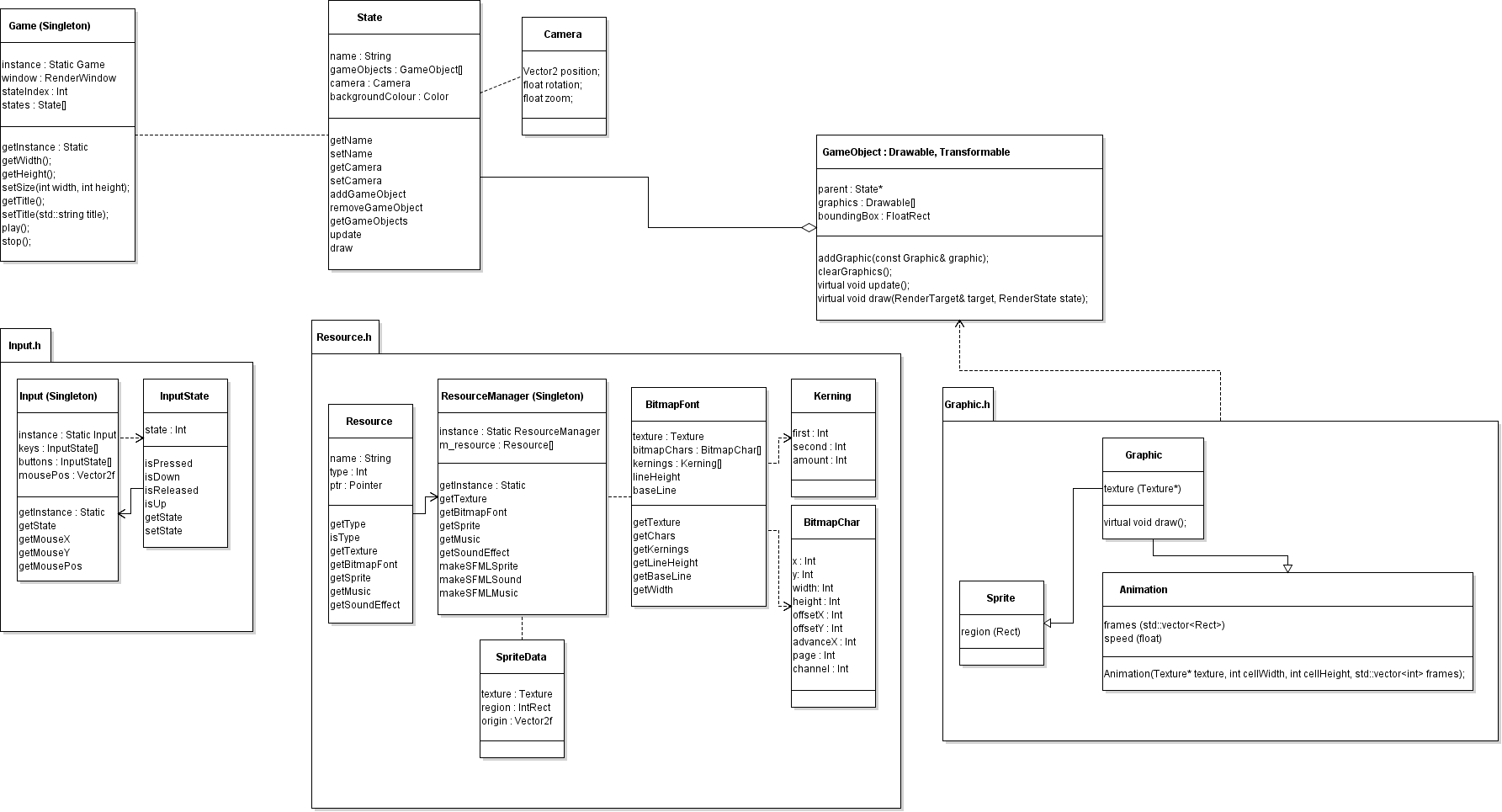
Since I’m using SFML I have areas such as texture, audio and basic drawing already handled for me, however they do not have the ability to make use of grids, animations or update and draw loops. With this I have identified a GameObject class which will encompass all these features into one class that can be extrapolated into the Ball, Button, TileMap and Label classes that I described above.

I have expanded all the types of classes I am going to need into this grid below. The separation in their grid is classes specific to the game and classes specific to an engine.

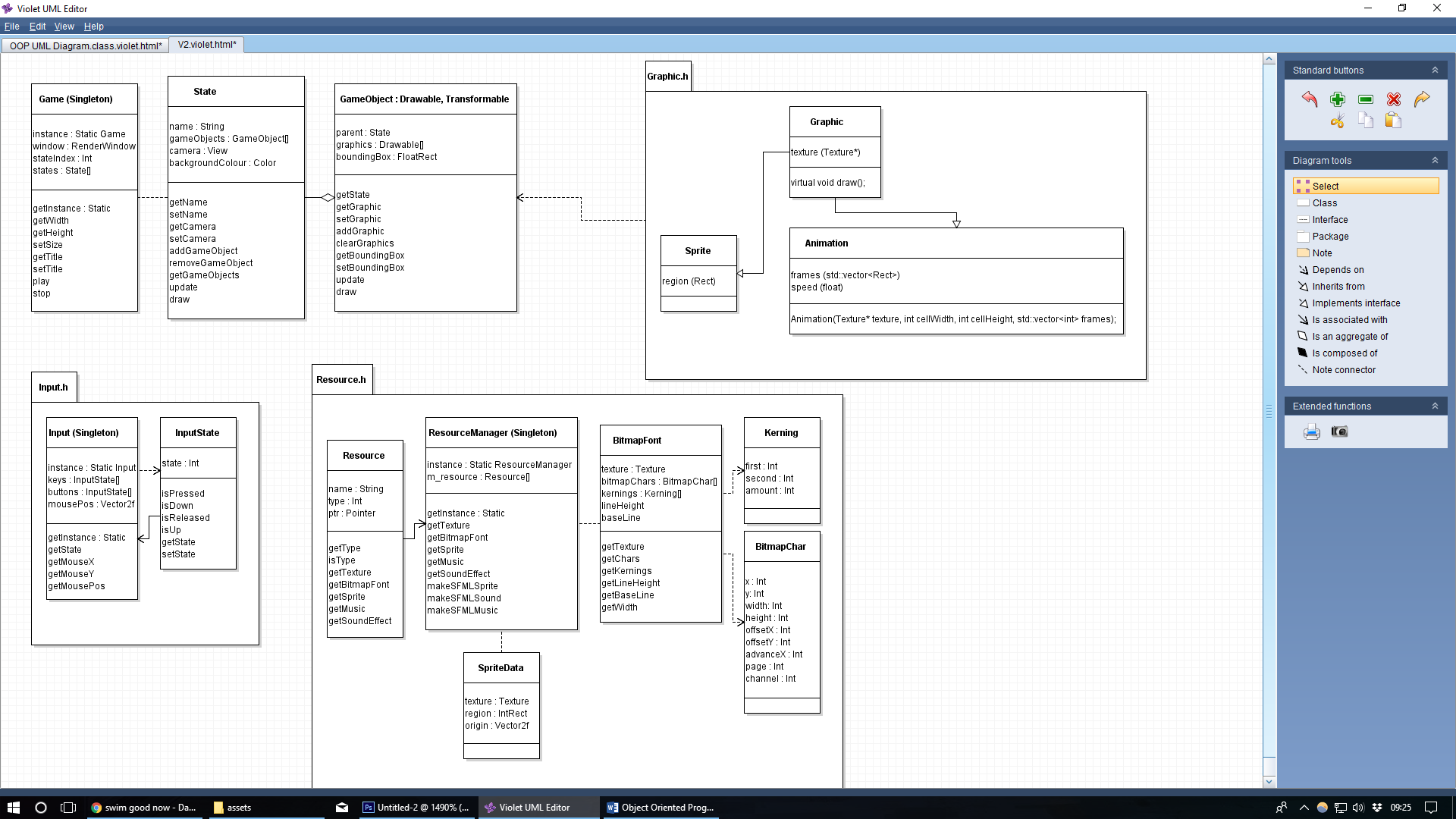
|  |  |
| --- | --- |
| Class Name | Purpose |
| Game | A class that will handle the main game loop and perform updates to the active State class that has been set. |
| Input | Handles all the input events for the game. It will record keyboard and mouse input in the form of an InputState. Previous development in games has informed me to make more detailed input. And the InputState class will record if the button is just pressed, down, just released or up which will give me greater control. |
| State | State is a shell class that will handle the GameObject’s that I add to it by calling the update and draw functions for each one. |
| GameObject | GameObject is the most commonly used class that has update and draw functions as well as a stored list of graphics and collision information. |
| ResourceFactory | A Singleton class which will load resources (textures, sprites, sounds etc.) and create graphics to be used with the GameObject class. |
| BitmapFont | I have chosen to use BMFont by AngleCode. So this class would load and make use of BMFont exported fonts. |
| Label | A Label class makes use of the BitmapFont class by creating a graphic with its own text, position, size etc. This can then be added to a GameObject. |
|  |  |
| TileMap | TileMap is an extension of GameObject which makes use of SFML’s VertexArray class, so I can create an optimised tile grid. There will also be flags to say which tiles are solid and which ones are not. |
| Ball | Ball inherits GameObject and will move and collide with the tiles on the tilemap. |
| Gun | Gun inherits GameObject and will define the size, direction and type of gun it is. It will also create Ball instances when it shoots. |
| MenuState | A main menu which has play and exit buttons available. |
| LevelSelectState | A state which shows all the levels in the game and buttons to play. |
| PlayState | This is the main game state where the gameplay takes place. |

In my Game class the main function that I will use it called start which is a function that takes no arguments and sets up a window and runs a game loop.

## Design



In the above UML diagram I have laid out the ResourceManager and its related classes. The ResourceManager is a singleton class which will hold a dynamic list of Resource objects. The Resource structure has a key, the key is used by the rest of the program to access the resources that have been loaded. I have chosen this because its not demanding on todays systems and provides a nice robust interface to managing resources, which I personally think is easier than managing the pointers yourself and parsing them around them program, this is like the texture id system in OpenGL however it’s not used for the same purpose. This way any part of the engine or game can access any resource by getting the singleton and using a key as a look up.



In this diagram I have outlined the State, GameObject and Game (Singleton) classes. State and GameObject have strong relationship with GameObject sending requests to State and vice versa. GameObject is a class that will be extended from by other game objects, in my games case it will be things like TileMap, Ball and Gun.

State contains functions such as add and remove objects, get and set name and camera as well as a personal update and draw function. The draw function makes use of SFML’s sf::Drawable interface which is used by SFML for propagating the RenderTarget and RenderStates through the drawing pipeline. Both update and draw will be virtual functions that can be overridden by the inherited version of State such as the PlayState or MenuState that I have identified in my analysis.

GameObject will have properties such as parent, a dynamic list of graphics (which in turn allows me to have multiple graphics per GameObject), and a bounding box which is used for collision. Each of these properties have getters and setters. Update and draw are also in this class as well in similar fashion to the State class, however GameObject extends from the sf::Drawable interface and also sf::Transformable abstract class. The sf::Transformable abstract class contains position, origin, scale and rotation properties that are used to create a 3x3 matrix that is then used for the RenderStates function allowing me to transform the GameObject as a whole and all the graphics contained with in it.