Reflection

By Max Morphy

Despite having used ‘object oriented programming before, this was the first time I have actually made use of the concept of inheritance. Previously with each new class I would rewrite the same code multiple times. This time I have saved myself lots of time and improved the readability, and improved the simplicity of my code. This is a major positive of this task.

The biggest issue with the coding in my program by far is the positioning of text on the screen. Virtually all of the text has its position hard coded, this means that its position is absolute not relative so if the user resizes the window the text formatted could be wrong. This greatly decreases the portability of the code as if a screen with different dimensions is used the size of the canvas element will be different and the position of the text relative to the canvas and relative to other elements will be off. I tried to use the .measureText() method that is inbuilt in javascript, but the value it returned was not in pixels despite ChatGPT saying it should be. This meant the 100px of screen size was not equivalent to 100 units of whatever that method was returning. So trying to dynamically render text using an algorithm such as xPosition = windowWidth/2 - textWidth/2 which I would normally use to center text would not work as the width of the text couldn’t be determined.

I also hard coded the text that drew text on the screen, In hindsight I could have made a drawText() method that would take parameters of: text, x, y then the text on the screen at the given location. As it currently stands I have many lines of code devoted to drawing text on the screen.

I was also considering If I should have rearranged some of the classes in terms of what classes are parents/ children of what classes. For example the Ground class uses only methods and variables of the rect class. Despite being a child of the PhysicsBody it does not use any method or variables of the PhysicsObject class as drawing rectangles is handled by the Rect object, the ground’s hitbox is just its rect object and the logic regarding collision/ hitboxes are handled in the players/ enemies update method anyway. In summary, there is an argument to be made that Ground should just be a child of Rect class. I also considered creating an entity class that would “sit” between the player/ enemy class and the physics object class so that the ground/platforms would be physics objects with hitboxes and then entities would be PhysicsObjects that also had code regarding movement. I also considered making PhysicsObject an extension of the rect class as every object has an instance of the Rect class acting as its hitbox. I even considered making circles PhysicsObjects, they obviously would require a different draw method but they still have a pos, hitbox, and in some cases velocity. The reason I am considering a restructuring of the classes like this is that there were some cases where seemingly unique classes did inevitably require the same code, indicating that perhaps greater abstraction is possible.

There are also a few cases where I believe more code could be moved from the game class to the object classes. For example the list containing all the bullets could very well have been a property of the player class, that is, I would be using this.player.bullets[] for example. I believe my game class became “overcrowded” and I could have made more function / variables properties of the other classes.

An issue I have run into before when developing using javascript is the CORS policy of google chrome. This prevents the accessing of different files locally. I am not entirely sure what this does or why. I know it is a safety feature that is supposed to prevent malicious use of chrome and potential harm to the computer but in my case it is very inconvenient. This means that to run the application the user would have to host a local server, probably through a VS Code package such as “Live Server”. I know Mr Howse knows how to use VS Code and will have no problem performing this to run the code, but in the hypothetical situation where this application is to go live to a real audience this would be a major strike against the accessibility of the program.