Report for 2D Physics Puzzler

Name: Jamie Cropley

P Number: P15188432

Module: IMAT2605 **-** Object Oriented Design and Development with C++

For attention of: Peter Cooke / Rafael Ktistakis

Date: 24/04/2017

Summary:

This report contains a complete breakdown of my game / physics simulation in c++, whilst utilising the box2D library as well as the SFML library. It explains in detail the game mechanics, design, choices made throughout, the level of functionality that I achieved and the testing process that I used.

Page count: 7 (Not including page 1 and 9).

Contents:

1 Introduction

With a game in the most basic sense as a starting point for myself, I initially setup in Visual Studio 2015 a template for my C++ code, that also included by means of files and in the code the SFML and Box2D library. I then began to break down what I needed to achieve in terms of code, my aims and objectives as such:

1.) Set the Bounciness of the ball via restitution.

2.) Set a static graphical background.

3.) Set screen bounds so the player ball or balls cannot go off screen.

4.) Make only one ball appear at a time via a timer or mouse click rather than mouse down method.

5.) Figure out how to display the blocks on the left and right of the screen.

6.) A Left Block and a Right Block to move up and down and detect a collision of the ball or balls thus adding a player score.

7.) Implementation of defined rules for winning and/or losing

8.) Scoring system, levelling system and game countdown timer contained within a Heads Up Display.

9.) Ensure code follows dynamics of Object Orientated Programming.

10.) Testing.

11.) Physics: Ball undergoes realistic motion

12.) Physics: Uses sensors or contact listeners.

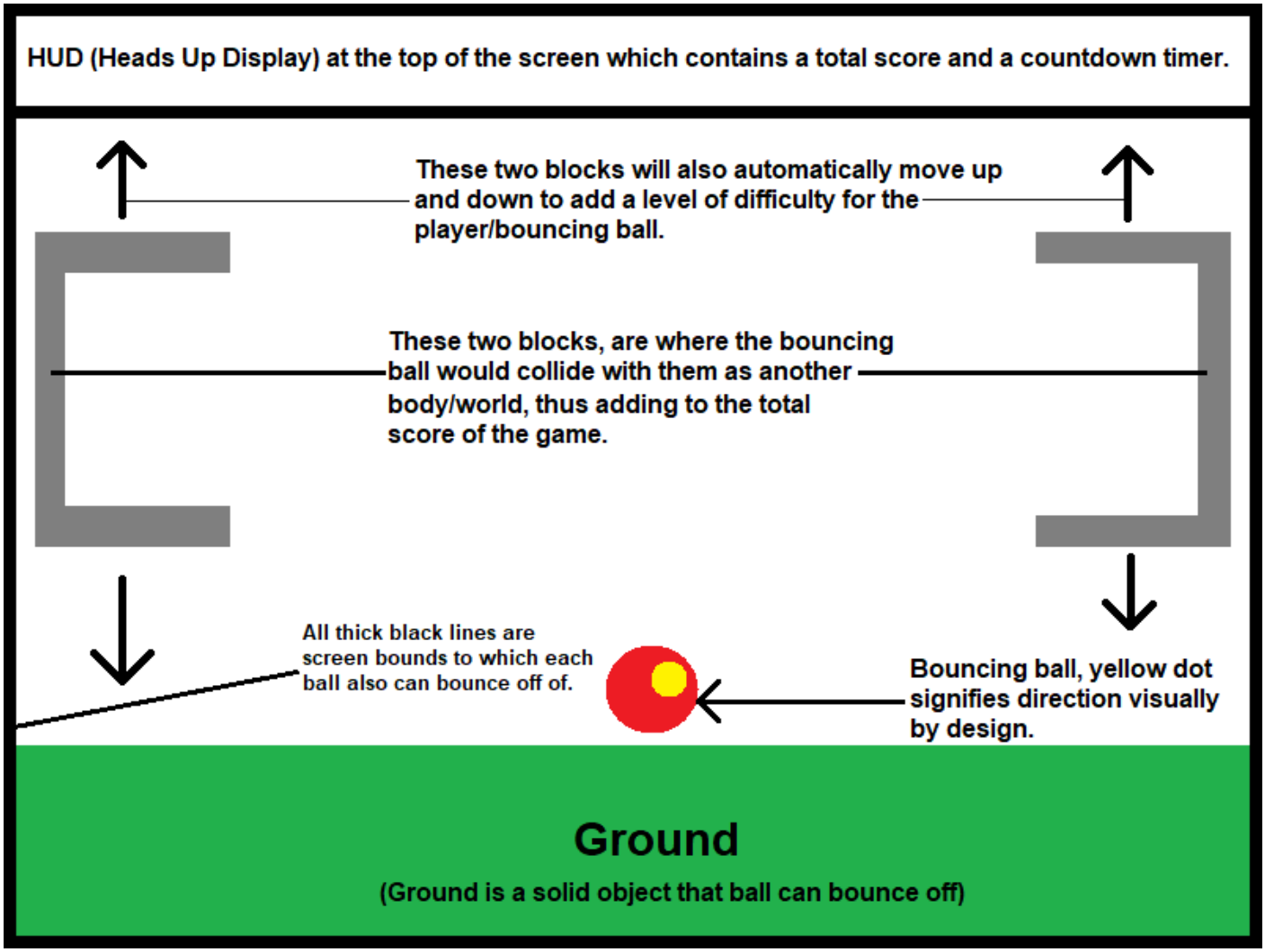
13.) Physics: Also uses joints and sensors. / Collides with Left block and right block.

14.) Physics: Uses many aspects of engine capabilities.

15.) Physics: Has worlds and bodies.

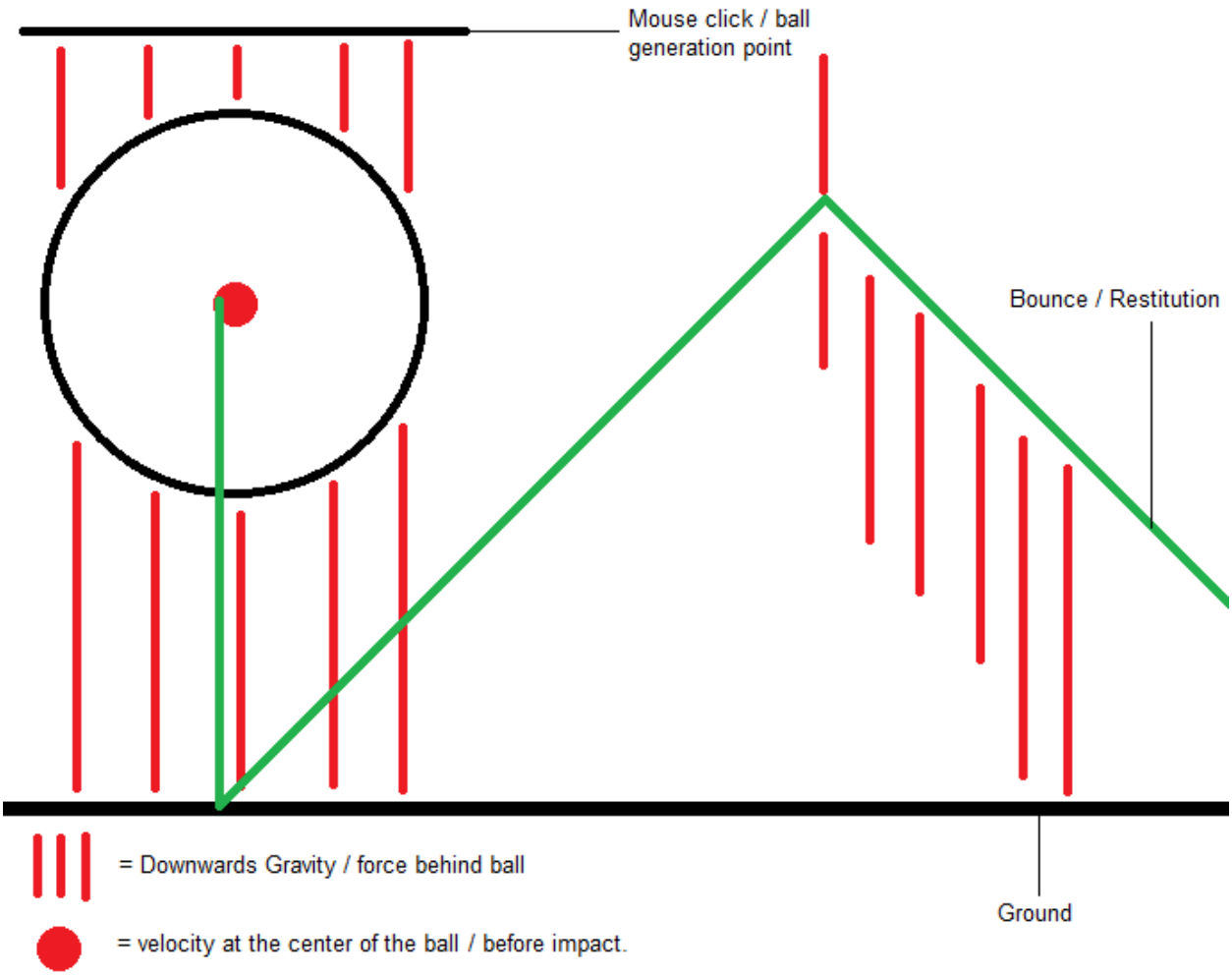
Throughout this project, I used various methods to solve the above list of goals that I wanted to achieve throughout my code, mainly that of research however I personally felt that the documentation I read on Box2D was very difficult to adapt to my goals in this respect thus achieving what I could from other various online resources and such.

2 Game concept and Design



The above image clearly illustrates my desired game concept and design. A new ball would generate upon each new mouse click ideally.

2.1 Game mechanics and physics

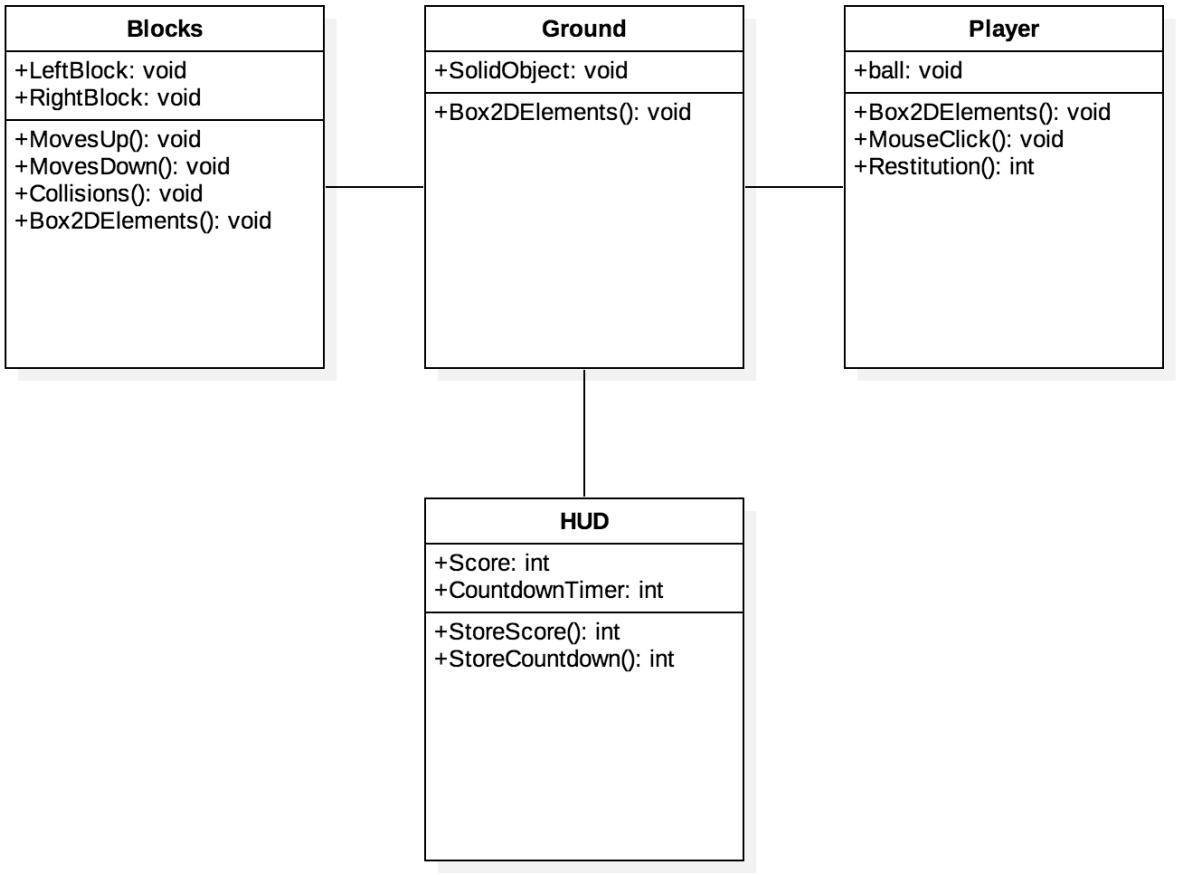


The game mechanics in this game all relate to the Box2D library where the SFML library was mainly used for drawing related media to the screen, with the ball being generated by the player clicking the left mouse button on each instance the ball would work in terms of physics in the following context:

The above diagram shows the basics of the momentum in terms of restitution, thus when hitting the ground, in the code I have set the coefficient of the restitution to '1.0f' which means there is elasticity to the ball causing it to bounce off the ground at its point of collision, its angle would depend on where you drop it, such as the height or near a colliding object such as the blocks used in my game. The initial force by gravity pulls the ball towards the ground changing its initial velocity, again the velocity is slightly changed when the ball hits the ground and bounces off it towards the angle that it goes towards.

2.2 UML and Diagram Object Orientated Programming design

My Object orientated design hit a lot of obstacles throughout and I could not implement it exactly how I wanted to into my code. I initially tried to do it as per the below UML diagram:



I tried to implement it via this UML diagram across header files and such but this was unsuccessful, I could not seem to get it working (as commented out near the top of my code (Line 4), I then went on further to break down an Object Orientated piece of code to better implement it into my code and build it up from there without using header files but I could not get the following to work with my code either:

```
#include <iostream>
using namespace std;

class Name {
	int value;

public:
	Name(int variable) {
		value = variable;
	}

	int content() {
		return value;
	}
};

int main() {
	Name

	//Rest of my code here.
}
```

I could not seem to get it working like the above coding method either, I think that this was due to me trying to put too much in one stream of a function (int main) rather than separating it all out into difference classes, members, objects, instances etc… I found this improbable to do within one file so concluded it was best to leave my code as it was. I believe my code still followed a slight Object Orientated Design however mainly because I structured it in an order that is readable relative to each object in the game and what each object was doing for that block of code. (Some elements in SFML especially, must be put after others in terms of how the program runs through, beginning to end.)

3 What I have achieved

Overall in this project I believe I have achieved a physics simulation of sorts, where I have made balls simulate movement and bounciness. I have achieved an Object Orientated approach to my code in some respects. I have also achieved using SFML to make the physics simulation look more aesthetically pleasing.

3.1 Testing

Debug version:

Does it compile and run OK: Yes

Game Mechanic	Does it work as intended?	If no what happens?	Expected fix?
Blocks (Specifically the left one)	No	The implemented one did not display.	To make it display, I believed the origin to be set wrong on it, even when I could get it to display the ball went right through it.
Ground	Yes	N/A	N/A
Player/Ball	Yes	N/A	N/A
HUD System	No	Nothing	Needs coding into the game.

Release version:

Does it compile and run OK: No

Game Mechanic	Does it work as intended?	If no what happens?	Expected fix?
Blocks (Specifically the left one)	No	64 Debugger errors to do with it being stuck in debug mode	To figure out in the vastness of visual studio 2015 settings what aspect I need to set from a ‘2’ to a ‘0’
Ground			
Player/Ball			
HUD System			

3.2 Reflection

I believe the key initial thing that would of made my project develop the way I wanted it based on my initial game design and goals therein, would have been to organise more how I was going to code it, especially within regards to object orientation.

This would have not only allowed me to organise my coding and objectives better but pin point what was wrong exactly and how it related to other objects within my game.

I did not realise as well how much physics would play a part in terms of figuring out how game items interact and such in terms of the Box2D library, this was something I seemed to overlook at first thinking it was more like the SFML library, this made learning a lot of the concepts difficult for me especially as mathematics is not my strongest of skills.

My initial aims and objectives list in section 1, I believe I could have condensed this a lot more to set myself more achievable goals from the onset of my project.

4 Conclusion

Overall I got my game albeit a physics simulation running and compiling in debug mode, I found overall that Box2D was immensely more difficult than I initially anticipated and writing my code initially in one function was not a great idea as it caused me issues later with Object Orientated Design. My game upheld although it has no gamified aspects to it, I am happy overall that it turned into a type of physics simulation.

5 Bibliography

- Erin Catto (2013) Box2D v2.3.0 User Manual. [Online] Box2D. Available from: http://box2d.org/manual.pdf [Accessed 24/04/17].

- Veendeta (2012) Tutorial: Getting Started with Box2D and SFML. [Online]. Available from: https://veendeta.wordpress.com/2012/02/16/tutorial-getting-started-with-box2d/ [Accessed 24/04/17].

- Iforce2d (2017) Box2D Tutorials. [Online] iforce2d. Available from: https://www.iforce2d.net/ [Accessed 24/04/17].

- David M. Bourg and Bryan Bywalec. (2013) Physics for Game Developers. 2nd ed. UK: O'REILLY.