IMAT2912 Lab Book 3 Report
Jack Hedges - P2551027

Introduction

The task for this lab book was to synthesise a real-time simulation of rigid body physics in 2D, using the Box2D and SFML libraries. Specifically, we were to create a game vaguely similar to pool, with some additional functionality only possible in a computer environment (such as moving joints).

Objectives

The objective was to create the game as per the coursework specification. This consisted of the following aspects:

- A game world
- A controllable ball, which projects a ray. The ray can be rotated, and the ball moves in the direction of the ray when space is pressed
- A collection of uncontrolled balls which react to physics collisions
- A block which can be moved and rotated with key presses
- A number of Box2d joints around the world - in this implementation, these were used to create obstacles

Tools

The project is being created using Visual Studio Community 2017, utilising the Box2d and SFML libraries to handle physics simulation and graphics drawing, respectively.

Implementation

As this project was similar to Lab Books 1 and 2, Lab Book 2 was used as a basis. From there, the game world was constructed as a rectangular square, and the game screen was made larger to accommodate UI elements. The key difference from Lab Book 2 was the lack of gravity, as the game is now viewed as if from above, instead of from the side. Disabling gravity and adding damping to the existing balls achieved this simulation perfectly.

Next, the joints were reimplemented. The trapdoor and slider used in Lab Book 2 were updated and placed into the game world in the bottom-left and bottom-right corner respectively. The spinners from Lab Book 1 were used in the top-right corner, and the pendulum was discarded due to not fitting the top-down theme of the game. The controllable block was also implemented at this time, as a modified version of the original controllable block from Lab Books 1 & 2, with added rotation and y-axis movement.

After this, sensors were implemented to control some of the joints. It was decided that there would be two sensors; a large one in the centre of the screen for the spinners, and a smaller one above for the trap door. For the sensors, the first change was to add polymorphic functions to the root PhysicalObject to handle sensor input. Once this was implemented, the sensor could accept a Vector of PhysicalObjects, and would iterate through the objects to call relevant functions when the sensor was triggered or released. In this specific case, triggering the blue sensor would invert the direction of the spinners each time it was touched, and triggering the orange sensor would open the trap door when touched, and close it again when released. These colours match up with the colours used for the objects they control. The controllable block was used for initial testing of these sensors.

Next, the controllable ball and ray cast were implemented. The controllable ball is referred to as “White Ball” in code, as a reference to the ball usually struck with the cue in most games of this style. It was also approached with the perspective that the player controls an invisible cue in order to manipulate the ball, with the casted ray being a preview of where the ball will go. Due to this approach, it was decided that the movement of the ball should actually be a charged shot instead of applying static amounts of power. The ball can be fired by holding space and releasing up to a maximum power, which is displayed as a graph in the lower-right of the UI. The ray cast itself is handled by creating two points; one at the origin and another at the first collision, and drawing a line between them to represent these points.

Finally, to truly represent the base inspiration of the game, four pockets were added to the game world. If a ball enters one of the pockets, it is hidden from the world - however, if the white ball goes in, it simply resets to its starting position. This gives the game an objective of trying to pocket all the balls, getting past the various obstacles made using joints.

Issues

There were a few issues to overcome when programming this game. The first issue experienced was when updating the collision listener code from Lab Book 2. Initially, this code had only been used to change the colour of one object to purple, and back to green on release. However, this was now handling many objects of many colours. The first iteration of fixes was for objects to keep a reference of their own colour upon their creation, and to simply return to this colour after the collision listening ended. However, due to the collision listeners implementation, it was impossible to access the required variable on the original objects, and as such the collision listener was simply setting to a semi-transparent colour on exit.

To work around this, textures were used for all parts. The collision listener could then set the colour to any available colour on the beginning of the collision, and once collision ends simply reset to white, returning the objects to their original colour.

A second issue was with trying to pocket the balls. The original iteration was simply trying to remove the balls from the vector entirely, similar to how the timer did on balls in Lab Books 1 and 2. However, this proved unreliable and would sometimes remove incorrect balls, or often even nothing at all. A simple workaround for this issue was to set the collision mask for these balls to 0, stopping them from colliding with anything, and no longer drawing them. By doing this the objects were effectively removed from the world without causing problems. This workaround is not ideal, but due to there only being 15 balls there is no major performance impact of doing so, so for the sake of this project it's an appropriate solution.

Outcome

Overall, the project has been mostly successful. Given more time it would have been nice to implement a scoring system, and perhaps a way to reset the game on victory or loss, as well as sound - however, it already meets all requirements of the project specification with the pocketing system acting as an additional aspect to make it more of a game than just a simulation. If the project was restarted from scratch, it's possible the issues mentioned above with the collision listener could have been avoided; but beyond that, the project has overall been very successful.