Custom Physics Documentation

Pool

Sofi Wesson

2022

Contents

[Visualised Game Using Your Custom Physics Simulation 2](#_Toc95731610)

[Custom Physics Simulation Interactions 4](#_Toc95731611)

[Custom Physics Simulation Class Diagram 5](#_Toc95731612)

[Custom Physics Simulation Potential Improvements 6](#_Toc95731613)

[Improvement #1 6](#_Toc95731614)

[Improvement #2 6](#_Toc95731615)

[Improvement #3 6](#_Toc95731616)

[Third Party Libraries 7](#_Toc95731617)

[References 7](#_Toc95731618)

# Visualised Game Using Your Custom Physics Simulation

[Delete This] Define what your visualisation (chosen game) is and then explain how you created it and how it works. (Include Image/s). [/Delete This]  
The game chosen is Pool

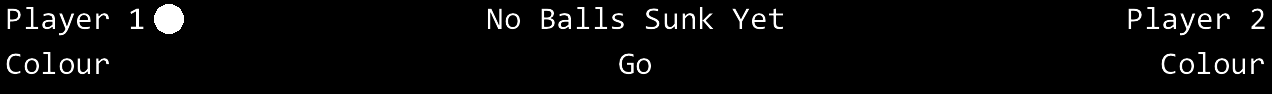
A screenshot of a video game

Description automatically generated with medium confidence

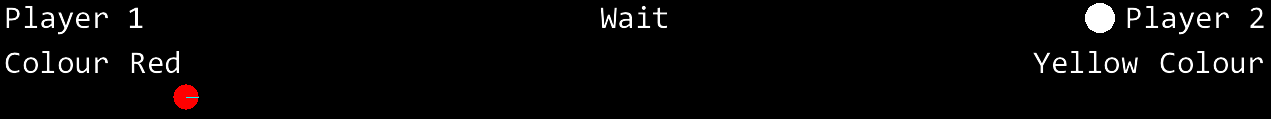
It was made using the 2D physics engine built in class, the base of the table is a kinematic box with no collider while the borders are kinematic boxes and circles with colliders, the pockets are circles with a trigger.

The balls are a custom class inheriting from the circle class that uses an enum to determine what type of ball it is.

The UI uses bootstraps 2DRenderer to draw text and circles that indicate what’s going on in the game.



Example of how the UI has changed.



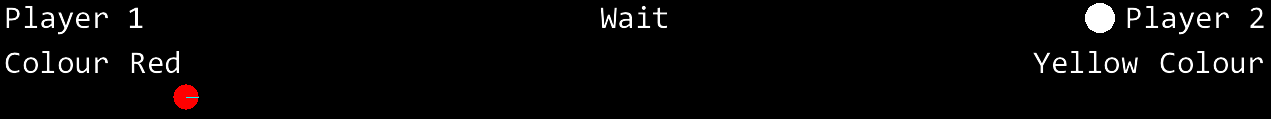
The player uses the mouse to determine which direction they want the ball to go in and strength it is propelled at by clicking and dragging, the direction and strength is visualised by a vector line pointing in the direction the ball will go and its length determining the strength the ball will propelled at.

When the player lets go of the left mouse button, the cue ball is propelled in the direction and speed indicated by the vector line.

A screenshot of a game

Description automatically generated with medium confidence

Once the first ball is sunk, the text “No Balls Sunk Yet” disappears for the rest of the game and a colour is assigned to both players using text.



When a ball has been sunk, it is added to a list containing the balls that player has sunk and moved from the table to the side of the player that sunk it.

A screenshot of a game

Description automatically generated with medium confidence

Who’s turn it is, is determined by a white ball next to that player.

# Custom Physics Simulation Interactions

The custom physics simulation demonstrates how 2D physics bodies interact with each other in a 2D environment. Each body in the simulation checks its collision against every other body in the simulation every frame, the physics bodies in the simulation consist of a plane, a circle, and a box, each body has its own collision logic against each body including itself.

The plane is a flat, one sided object that is considered to be infinitely long across both sides of its edge extending from its normal. The plane is the only body that doesn’t collide with itself and can only be collided with in one direction. The plane checks the extents of it normal and distance from the world origin against the constraints of the other body to check for collisions. Because the plane doesn’t have any height to it, it is possible for the other body to ‘pass through’ the plane between two frames if it is traveling fast enough, as it may never actually physically overlap the plane, to combat this, the plane also checks the velocity of the other body against its normal to check if the body has potentially ‘passed through’.

The circle is a perfectly round object whose boundary consists of points that are an equal distance from one another originating at a fixed length from its centre point. To get its collision, we get its position in the world and then check the displacement of its radius from its centre against the other body’s constraints that its colliding with.

The box is a rigid, four-sided object consisting of two identical right-angled triangles both joining along the hypotenuse. The boxes collision detection is a bit more complicated as we must do a check against the displacement of each of its corners and extents from its centre as well as factor in the displacement from its rotation. This is done by checking each end of the boxes extents against the local position of the other body then checking this against the displacement of the constraints from the other body’s position.

When interacting together, the physics bodies use combined versions of their collisions to accurately detect any overlaps between each other, once a collision is detected, it is then passed on to get resolved so that the bodies don’t phase through each other, we do this by getting the collision normal between both bodies, the velocity of the contact points, and the effective mass of both bodies, and use this to calculate the impact force. The impact force is then used to calculate a new velocity for the physics bodies, resulting in them bouncing off of eachother.

# Custom Physics Simulation Class Diagram

# Custom Physics Simulation Potential Improvements

## Improvement #1

Implement calculations for angular velocity.

## Improvement #2

Only have physics bodies check collisions for other bodies near it, instead of everyone physics body in the scene.

## Improvement #3

Have a single function to detect collisions between any two possible shapes instead of a separate function for each collision possible.

# Third Party Libraries

Bootstrap – Basic 2D and 3D engine and math library used as the base that the custom physics engine was built upon.

# References

Baker, M., 2022. Physics - Collision in 2 dimensions - Martin Baker. [online] Euclideanspace.com. Available at: <https://www.euclideanspace.com/physics/dynamics/collision/twod/index.htm> [Accessed 18 August 2022].