Custom Physics Documentation

<8 bit pool>

dylan alvaro

2023

Contents

[1.0 - Custom Physics Simulation Class Diagram 2](#_Toc125539830)

[2.0 - Custom Physics Simulation Interactions 3](#_Toc125539831)

[3.0 - Custom Physics Simulation Potential Improvements 3](#_Toc125539832)

[3.1 - Improvement #1 3](#_Toc125539833)

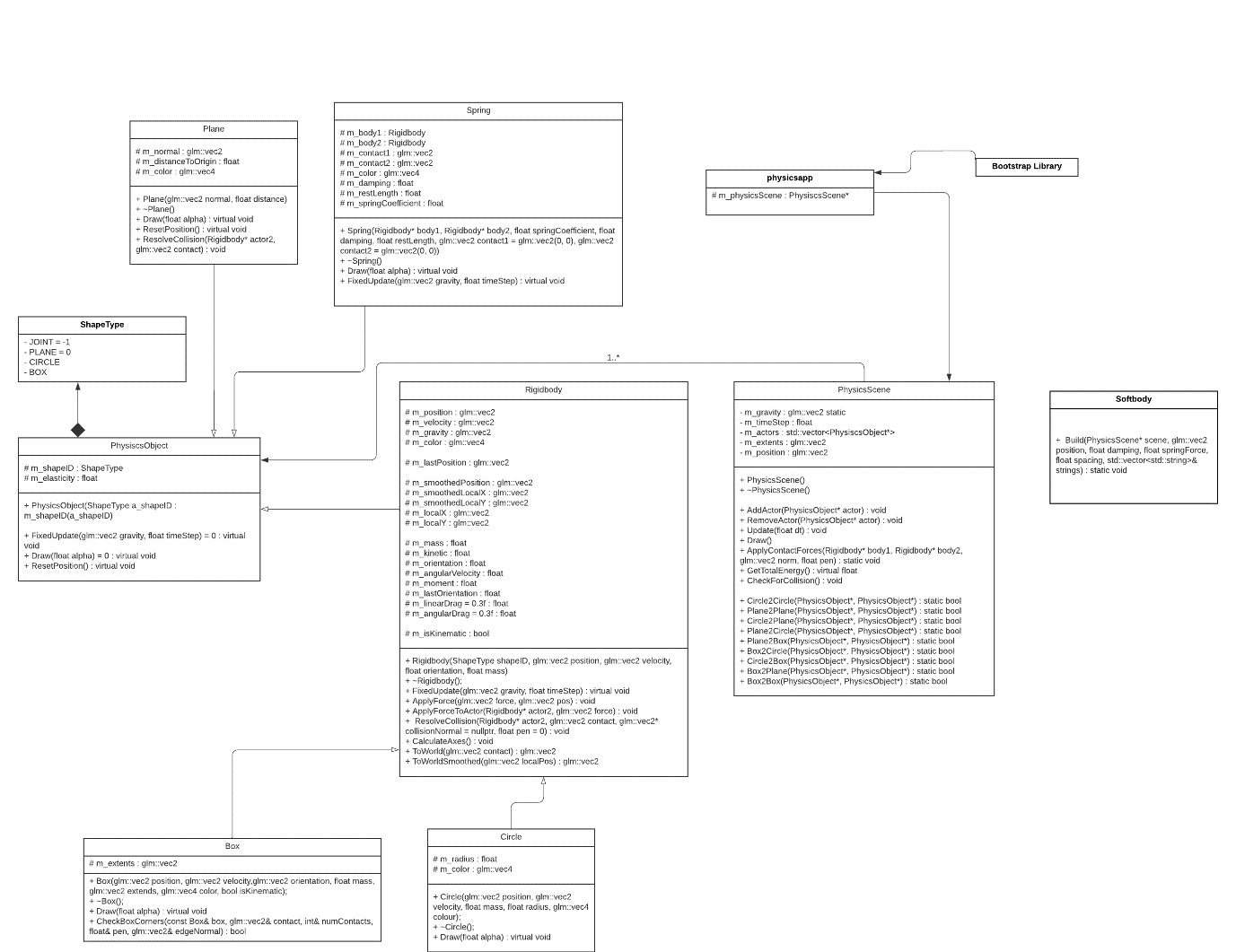
[3.2 - Improvement #2 3](#_Toc125539834)

[4.0 - Visualised Game Using Your Custom Physics Simulation 4](#_Toc125539835)

[5.0 - Third Party Libraries 4](#_Toc125539836)

[6.0 - References 5](#_Toc125539837)

# 1.0 - Custom Physics Simulation Class Diagram



# 2.0 - Custom Physics Simulation Interactions

The custom physics simulation that I’ve created is able to successfully demonstrate real life physics through the implementation of triggers, collisions, drag and gravity between different shapes such as Boxes, Circles, and Planes. The physics engine I’ve created is able to showcase the effect that collision has on dynamic rigid bodies through the use of forces and other hard coded functions.

The physic bodies are able to interact together through the use of collision between each object. Objects that are dynamic are able to move due to factors such as the addition to forces and the velocity/ rate that the objects move at. Successful collision is implemented by effectively calculating the mass at the contact point for each object and then seeing how far it will move due to the force applied, this force can be altered to get different results. When two shapes collider they can exert large forces onto one another, this is called impulsive forces. This large force causes any other force present to not have any effect on what has already happened. An example of this in my pool game would be if the player was to hit the white ball with the imaginary cue, the white ball will experience a change in momentum due to the impulsive force that was exerted by the cue. If the white ball was to be collided with it would be neglected.

# 3.0 - Custom Physics Simulation Potential Improvements

## 

## 3.1 - Improvement #1

The first improvement that could be made to the Custom Physics Engine that I have made would be to improve/change the existing collision within the simulation. Currently the physics engine has working collision between circles, boxes, and planes. In the future the addition of other shapes such as triangles and hexagons would allow for a fun challenge, implementing different shapes would allow for different simulations to be created. In the current system when a ball is shot fast enough into the wall it phases through. Through the implementation of better collisions this could be fixed.

## 

## 3.2 - Improvement #2

## the second improvement that could be made to the Physics Engine could be to make the system more efficient and effective. I believe that in it’s current state things such as the collision and way triggers are handled can be changed to be more efficient and more streamlined.

# 4.0 - Visualised Game Using Your Custom Physics Simulation

The game I chose to make was 8-bit pool, traditionally pool is a two-player game where the player to sink all their balls before the other does, wins. The type of ball is assigned to first person to sink the ball at the start, either being the solids or stripes.

Graphical user interface

Description automatically generated

I was able to create the game through the creation of a series of dynamic circles that would function as the balls for the game, with this I could change the colour of the balls to represent the stripes and solids shown in a traditional game of pool. To distinguish the different between the white ball and the other balls in the game I made the mass of the white ball lower. To play the game I created a way for the players to drag their mouse to allow the ball to move in the direction that they chose to aim at, this is done through velocity. To showcase the players turns in the game I made a an int called playersturn, after the initial move whether the first person gets a ball in or not it moves to the next person and changes the colour of the initial line and circle, the line and circle take in a worldPos which is the mouses X and Y coordinates.

# 

# 5.0 - Third Party Libraries

The third party libraries that were used within this custom physics engine were things such as OpenGL and GLM (OpenGL Mathematics). The use of GLM within my custom physics engine allowed me to make use of a lot of functionality that I normally wouldn’t be able to access including vectors.

# 6.0 - References

*The UML 2 class diagram* (no date) *IBM Developer*. Available at: <https://developer.ibm.com/articles/the-class-diagram/>.

*UML 2 class diagrams: An agile introduction* (no date) *AgileModeling.com*. Available at: http://www.agilemodeling.com/artifacts/classDiagram.htm (Accessed: February 20, 2023).

Potasky, J. (no date) *Gilbert-Johnson-Keerthi Distance Algorithm*, *Efficient Collision Detection*. Available at: https://cse442-17f.github.io/Gilbert-Johnson-Keerthi-Distance-Algorithm/.

*Ch. 11 introduction - university physics volume 1* (no date) *OpenStax*. Available at: https://openstax.org/books/university-physics-volume-1/pages/11-introduction (Accessed: March 7, 2023).

Alrasheed, S. (1970) *Impulse, momentum, and collisions*, *SpringerLink*. Springer International Publishing. Available at: https://link.springer.com/chapter/10.1007/978-3-030-15195-9\_5#Sec4 (Accessed: March 7, 2023).