**<Anything between pointed brackets> or states to [Delete this] around a subject and this text box must be addressed and the original template content deleted, also remember to remove the draft watermark before submitting.**

Custom Physics Documentation

Eight Ball Masters

Matthew Carver

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 4](#_Toc125539832)

[3.1 - Improvement #1 4](#_Toc125539833)

[3.2 - Improvement #2 4](#_Toc125539834)

[4.0 - Visualised Game Using Your Custom Physics Simulation 5](#_Toc125539835)

[5.0 - Third Party Libraries 6](#_Toc125539836)

[6.0 - References 6](#_Toc125539837)

# 1.0 - Custom Physics Simulation Class Diagram

# 

# 2.0 - Custom Physics Simulation Interactions

[Delete This] ***Define*** in your own words what your Custom Physics Simulation is demonstrating and ***outline*** how the physical bodies can interact together as dynamic and static objects. [/Delete This]

The Custom Physics Engine developed according to the tutorials is demonstrating a simplified physics simulation including force applications, simplified collisions, collision triggers, rotations and mass calculations.

The physical bodies can interact together as dynamic and static objects by virtue of the class structure of the bodies; all physical bodies are derived from the PhysicsObject class.

Static objects inherit directly from PhysicsObject and thus do not have the dynamic functionality for movement, mass, rotations etc. A Plane class is created for our static object.

Dynamic objects contain more functionality but are different in some fundamental ways which requires an intermediate class of inherit from. Rigidbody inherits from PhysicsObject and contains variables and functions that a common among all manner of primitive object interactions. A Circle and a Box class are the dynamic primitive classes created.

Dynamic and Static objects interactions are handled within the PhysicsScene class. The PhysicsScene checks for collisions amongst its actor list, identifies the PhysicsObject type, leverages a collision function array to assign a delegate for the appropriate collision function operation. The operation calculates and applies force to each actor (if they’re dynamic) according to their properties.

Kinematic versions of the dynamic objects is possible by setting a variable that controls the mass of the objects and some functions, essentially making them “static” or kinematic.

# 3.0 - Custom Physics Simulation Potential Improvements

[Delete This] The objective of this simulation is to demonstrate static and dynamic objects interaction in 2D space. ***Examine*** what improvements you could make to your simulation; this could be to:

* Support further features.
* provide a more accuracy.
* Make it more precise.
* Improve the quality.

(This refers to custom physics simulation library you are creating, not directly the game you have created. These are not always mutually exclusive however.) [/Delete This]

|  |  |
| --- | --- |
| Examine | Present in depth and investigate the implications. |

## 3.1 - Improvement #1

Friction - Torque

## 3.2 - Improvement #2

Oct-Tree

# 4.0 - 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 chosen visualisation utilising the custom physics simulation engine is Eight ball pool. Eight ball is a game with 16 balls on a table:

* 1 x White
* 1 x Black
* 7 x Solid Color
* 7 x Striped Color

The player must strike the white ball against the others on the table, abiding by the rules, sink their colored balls in the 6 pockets before sinking finally the black to win the game.

The project was created using the “Project2D” project as a template. The process of creation was as follows:

* Create a physics scene
* Creating a PoolBall class
* Loading balls into the game
* Creating wall “boxes”
* Creating pocket triggers
* Cue stick
* User Interface
* Implement rules

The game works by applying force to the white ball using the cue stick. The cue stick is measured against the white ball, the length is calculated, a force multiplier is applied and the direction vector is ascertained from the position of the mouse to the white ball. Using these variables, force is applied to the white ball.

The white ball, in turn, collides with the kinematic boxes

|  |  |
| --- | --- |
| Explain | Tell how things work or how they came to be the way they are, including perhaps some need to 'describe' and to 'analyse' (see above). |

# 5.0 - Third Party Libraries

[Delete This] ***Identify*** and ***explain*** third-party non-physics libraries used, if any. Otherwise ***identify*** why none where used [/Delete This]

|  |  |
| --- | --- |
| Identify | Pick out what you regard as the key features of something, perhaps making clear the criteria you use. |
| Explain | Tell how things work or how they came to be the way they are, including perhaps some need to 'describe' and to 'analyse' (see above). |

# 6.0 - References

[Delete This] List of references and research material used to influence the creation of your custom physics simulation and where you researched to improve the quality of the system.

Use the Harvard Citation Method to cite books and websites used. Here is a link to a good citing website if you are unsure how to do so <https://www.citethisforme.com/citation-generator/harvard> [/Delete This]