Contents

[Physics Engine 3](#_Toc507493256)

[PhysicsScene 3](#_Toc507493257)

[IFixedUpdater 3](#_Toc507493258)

[PhysicsObject 3](#_Toc507493259)

# Introduction

\*\*\*\*\*\*TO DO\*\*\*\*\*\*

# Dependencies

\*\*\*TODO say how it used Bootstrap, glm\*\*\*

# Physics Engine

Classes in the PhysicsEngine library are in the physics namespace.

## PhysicsScene

class PhysicsScene

A physics scene contains physics objects which interact with each other and manages the simulation.

During each fixed duration physics step the following actions are taken:

* FixedUpdater objects observing the scene have their fixedUpdate method called
* earlyUpdate is called on all physics objects. This is used to apply forces from springs.
* fixedUpdate is called on all physics objects. This is where rigidbodies move.
* Collisions are checked between physics objects. On a hit, the objects inform observers about it and resolve the collision

After all physics steps in an update, the physics objects are drawn. This interpolates their current and previous position by the amount of time left in the update. This means the object is consistently drawn one timestep before the current time, preventing temporal aliasing.

PhysicsObjects and FixedUpdaters are stored as shared pointers, since the scene will probably share responsibility for them with some other object. While FixedUpdaters can be removed at any time, PhysicsObjects should not be removed during a collision. Instead, the object’s kill() method should be called. This will flag it for removal at the end of a fixed update. Also, Clear should never be called within fixedUpdate or OnCollision functions.

## IFixedUpdater

class IFixedUpdater

This purely abstract class is an interface to allow derived classes to be added to a PhysicsScene and have their fixedUpdate function called every physics timestep. These are held by the PhysicsScene as shared pointers.

## PhysicsObject

class PhysicsObject

This is an abstract base class for all objects in a physics scene.

A physics object can have CollisionObserver objects subscribed to it. When BroadcastCollision is called, these objects will be sent the collision object, allowing them to react to the collision. They are stored as weak pointers by the PhysicsObject, to avoid reference cycles between the observer and object.

To remove PhysicsObjects from a scene safely, and let anything else with references to it know it has been removed, the kill and isAlive methods can be used. Calling kill will set the object to dead, and after the next physics update it will be removed from the physics scene. Similarly, if it has been remove from a scene using removeActor, it will be set as dead. Before a killed object is returned to a physics scene, resetAlive must be called, or it will be removed again the next frame.

## Plane

class Plane : public PhysicsObject

This PhysicsObject represents a plane. The plane is one sided (anything behind the plane is considered to be colliding with it) and infinitely long. Planes are always considered static objects.

## RigidBody

class RigidBody : public PhysicsObject

A rigidbody represents solid, non-deformable object.

Rigidbodies can be dynamic, kinematic, or static. Dynamic bodies move, are affected by drag and gravity, and can have forces applied to them both through collisions and with the applyForce or applyImpulse methods. Kinematic bodies move and rotate, but aren’t affected by any forces. Static bodies do not move, except by setting their position directly.

A body is made kinematic by setting its mass as 0 or infinity, and is made static with the setStatic method. Collisions aren’t tested between two static bodies.

Rigidbodies store both their current and past positions and local axes. This is so they can be rendered at a position and orientation interpolated between these values, ensuring smooth movement regardless of update time.

## Sphere

class Sphere : public RigidBody

This is a spherical rigidbody. The

## Collision

struct Collision

TODO describe collision struct