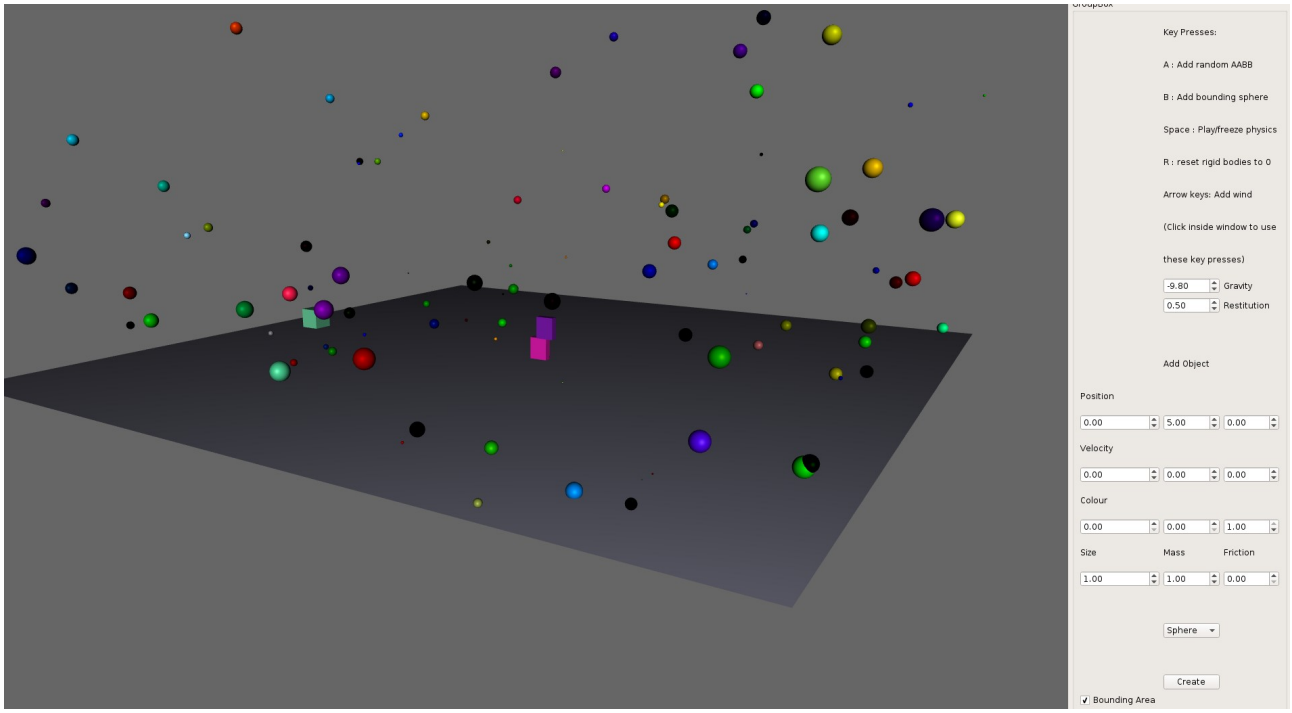
Physics Engine Report
Carla Moy
i7466612

Introduction

This report documents the approach used to create the physics engine for the computing for animation assignment.



Methods

The rigid body class stores a pointer to a collider object. This collider gets resolved to either an AABB or BoundingSphere upon instantiating it. When an object is added the rigid body constructor takes in a collider, which takes in the specific information for the size and position of each object. The size and position require being initialised inside of the colliders in order to resolve their respective collision detection requirements. The position and size remain consistent since the Physics Engine class accesses these values through the RigidBody class.

The getSize() method retrieves an ngl::Vec3 so that the width height and depth of the objects can be effected. For simplicity I kept the sphere's radius constant so that sphere-sphere collision detection works correctly.

I use verlet integration to update the position each frame based on this equation :

Fig 1

$$\vec{x}(t+\Delta t) = \vec{x}(t) + \vec{v}(t)\,\Delta t + \frac{1}{2}\,\vec{a}(t)\Delta t^2$$

$$\vec{v}(t+\Delta t) = \vec{v}(t) + \frac{\vec{a}(t) + \vec{a}(t+\Delta t)}{2}\Delta t$$

The forces are accumulated as world forces (gravity and wind) and object forces (friction and drag) and used to update the acceleration with reference to time and mass. This is then averaged to update the velocity.

Friction is calculated based on the equation :

$$Friction = -1 * \mu * N * \hat{v}$$

Fig 2

Where μ is the coefficient of friction
N is the normal between the two colliding surfaces
v is the velocity of the object
-1 gives the reverse direction to the velocity

Drag is calculated based on the equation :

$$F_d = -\frac{1}{2}\rho v^2 A C_d \hat{v}$$

Fig3

Where Fd refers to drag force
ρ(rho) is density of medium (not applicable in by physics engine so set to 1)
v is the magnitude of the velocity
A is front of object pushing through the medium (also not applicable)
Cd is the coefficient of drag.
v is the velocity normalized

An issue I had was correct collision with the ground and walls bounding area. When gravity is applied there is a floating point error which causes objects to sink into ground plane. This is a common problem for physics engines and has been addressed in thesis and papers such as by David Baraff "Fast Contact Force Computation for Nonpenetrating Rigid Bodies" [Baraff, D. 1994] and Brain Mirtich "Impulse-based Dynamic Simulation of Rigid Body Systems". [Mirtich, B. 1996]

To resolve this issue I attempted to implement impulse based collision response in accordance with Brian Mirtich's methods. This needs more work, but as a quick way to fix the issue to some degree I set the position of the object to a small degree opposite to the gravity's direction when the object reaches the wall and ground bounds.

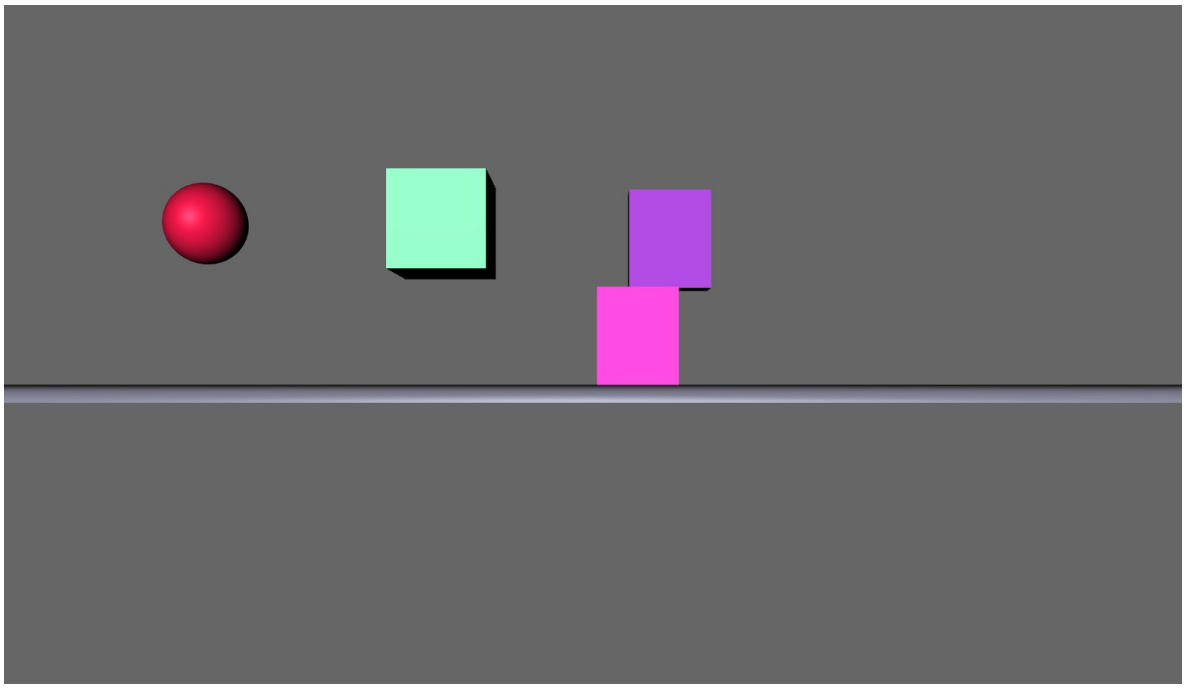
The collision response I implemented instead of the impulse based approach is :

$$v_a = \frac{C_R m_b (u_b - u_a) + m_a u_a + m_b u_b}{m_a + m_b}$$
$$v_b = \frac{C_R m_a (u_a - u_b) + m_a u_a + m_b u_b}{m_a + m_b}$$

Fig4

Where va is the final velocity of the first object after impact
vb is the final velocity of the second object after impact
ua is the initial velocity of the first object before impact
ub is the initial velocity of the second object before impact
ma is the mass of the first object
mb is the mass of the second object
CR is the coefficient of restitution (if it is 1 the collision is perfectly elastic, if 0 collision is perfectly inelastic)

This method conserves momentum between the rigid bodies when they collide.



Conclusion

There are still issues to resolve with ground collision and much more can be added. The structure of the code I have implemented allows for lots of developments to be made to expand and improve the code. With more time I would have implemented the spatial grid or octree to divide the space into cells to improve the efficiency of the system. I would also add more types of colliders, improve the collision responses and include more physics forces, such as attraction force between objects which was began, and surface sliding. Another major area to add would be rotation for OBB's and sphere rolling.

References

Fig1
anonymous. Verlet Integration. Creative Commons Attribution-Share Alike 3.0 Unported. URL : https://www.saylor.org/site/wp-content/uploads/2011/06/MA221-6.1.pdf

Fig2 & Fig3
Daniel Shiffman [2012] Nature of Code. Chapter 2. Forces. ISBN-13: 978-0985930806. Creative Commons, 444 Castro Street, Suite 900, Mountain View, California 94041, USA. URL : http://natureofcode.com/book/

Fig4
Ferdinand Beer, Jr. and E. Russell Johnston (1996). ***Vector equations for engineers: Dynamics*** **(Sixth ed.). McGraw Hill. pp. 794–797. ISBN 978-0070053663. URL : https://en.wikipedia.org/wiki/Inelastic_collision**

David Baraff [1994] ***Fast Contact Force Computation for Nonpenetrating Rigid Bodies.*** **Robotics Institute Carnegie Mellon University. Pittsburgh, PA 15213. SIGGRAPH 94, Orlando, July 24–29. URL : https://www.cs.cmu.edu/~baraff/papers/sig94.pdf**

Brian Vincent Mirtich [1996] ***Impulse-based Dynamic Simulation of Rigid Body Systems.*** **Arizona State University, University of California.**
URL : http://www.kuffner.org/james/software/dynamics/mirtich/mirtichThesis.pdf