2D Physics Engine

CS 293 Project

Kumar Pallav *100050046*

Pulkit Bansal *100050045*

Introduction

* We tried to build a basic 2D Physics Engine which worked on simple circular objects constrained in a simple world
* Physics Engine forms the basis of most of the modern online/mobile games which involves physics e.g. Angry Birds, Doodle Jump

Input and Output

* User defines a World object to which he adds objects
* He provides the initial conditions
* And with his choice of time resolution he inquires the evolved state of the world in terms position, velocity and acceleration of the bodies.

*High Level Structure

Vect.h

- 3 mutually perpendicular components
- Accessor functions and mutator functions
- General operations defined on vectors such as *dot*, *cross*, *scalar multiplication*, *magnitude* etc.
- Assignment, comparison, addition and subtraction operators defined

Kinematic.h

- Stores kinematical quantities such as position, velocity and acceleration
- Provides their accesor and mutator functions
- Provides an integrate(t) function which returns a reference to the evolved kinematical state after t time

Body.h

* Simple circular objects with mass, radius, restitution and a Kinematics object
* Provides accessor functions for its data members

World.h

- Contains the boundaries of the stage where all the action takes place
- Contains vector of bodies contained in it
- Has a Gravity associated with it
- Provides the user a reference to the vector of bodies
- Has an Integrate(t) function which evolves the world over time t

```
void World::integrate (float dt)
{
        if (dt < nextCollision)
        {
                        setNewKinematics(dt);
                        nextCollision = nextCollision - dt;
        }
        else
        {
                setNewKinematics(nextCollision);
                collisionHandler(); dt = dt - nextCollision;
                nextCollision = findNextCollision();
                integrate(dt);
        }
}
```

*integrate(t) [World.h]

Work Distribution

- Work evenly distributed
- Class Implementation of World.h done by Pulkit
- GUI done by Pallav (using QT)
- We also implemented an algorithm to compute minimum distance between two polygons
- Almost 1100 lines of code plus 500 lines of markup