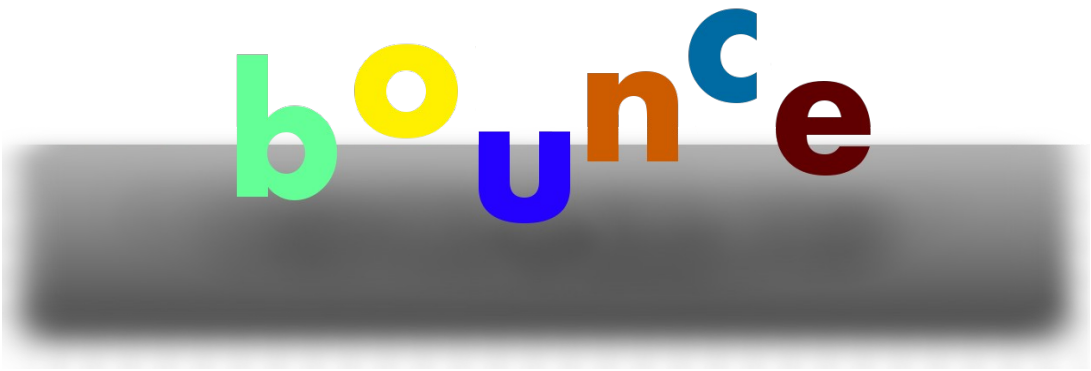
Bounce2D 1.0 Manual



1 Introduction

Description

Bounce2D is a 2D rigid body physics simulator. It is open source and free for commercial and non comerical use; under the BSD licence.

Features

- 2D rigid body physics simulator
- Collision detection and reporting
- Developed in Java
- Open Source

Download

- The source code is available from <http://code.google.com/p/bounce2d/ >
- Documentation is available on the wiki page <http://code.google.com/p/bounce2d/wiki/Bounce2D >
- Demos are available from <>

Contact

- Donovan Benoit <donovan.benoit@gmail.com>

2 Using Bounce

Integration

Bounce has been designed to be easily integrated into an existing project or incorporated into a new project.

- Add the Bounce package to your project
- Instantiate a new Bounce World
- Add PhysicsObjects and Modifers to that world
- Add a CollisionListener to the world to report collisions
- Advance the simulation by the desired time step
- Get the updated posisitions and rotations from the objects

Modification

Bounce has been designed to be easily modified. New types of Modifiers and PhysicsObjects can be created to add functionality to the engine.

3 Bounce Library Overview

Introduction

Bounce is tasked with simulating forces applied to bodies, collision detection, collision resolution, and simulation of other constraints. This section will give an overview of the components in the Bounce Package and how they interact.

Software Design

Bounce has been designed from the start to be easily customized and modular. The five main components used by the library are World, PhysicsObject, Modifer, PhysicsListener, and Vector2.

VPP fig

World

- World
  The `World` class is responsible for simulating the physical world. This world consists of `PhysicsObject's` `Modifier's` and other properities. The units used by the world are 100 units per meter and time is in milliseconds. The positive x-axis points right and the positive y-axis points down. The world has gravity which acts on some `PhysicsObject's` (see `PhysicsObject`). The force of gravity can be changed through the set gravity method.

- Advance Simulation
  The `advanceSimulation` method advances the physicsl simulation by a time step. If the time step is larger then the `maxTimeStep` then it advances by `maxTimeStep`. `maxTimeStep` can be set using the `setMaxTimeStep` method. `AdvanceSimulation` integrates the `PhysicsObject's` over time then recalculates verticies of the objects and checks for collisions. If a penetration is detected then the simulation is steped back, the time step is cut in half and resimulated until there is no penetration or the timestep is suficiently small.

- Integration
  Integration is responsible for taking the forces and torques applied to bodies and integrating them over time. This produces velocities which are then integrated to obtain the new positions and rotations of `PhysicsObject's`.

- Collisions
  The engine checks all non static `PhysicsObject's` against all other `PhysicsObject's`; this results in the order of 2^n checks. The colision detection algorithm uses a bounding circle test to see if two objects are close enough to collide. If the circles overlap the function goes into a more accurate collision detection algorithm. The algorthm used for this is a Barycentric point in triangle test. <http://www.blackpawn.com/texts/pointinpoly/default.html > If the point is outside the triangle no collision is detected, if it is inside then a collision is taking place. Depending how close the point is to the outter edge of the triangle it either resolves the collision or reports a penetration and exits the collision check.

- Resolving Collisions
  When a collision is detected the `resolveCollision` function takes the two bodies that collided and applies an impulse and moment to the bodies dependant on the collision normal, collision point.

- Collision State
  `CollisionState` is an enumeration used by the `World` to report when a collision or penetration takes place. The three states are: `CLEAR, COLLIDING, PENETRATING.`

Physics Object

`PhysicsObject` is an abstract class from which all objects simulated by the `World` are extended. To create an `PhysicsObject` you need a name position and mass. If the mass is greater then 0 an dynamic object is created if the mass is 0 then a static object is created. Static objects cant move.

- `RigidBody` consists of a convex mesh of verticies. The max number of vertices per object is 32. `RigidBody's` have a moment of inertia associated with them and can be rotated.

- `Particle` is a point that can be integrated over time. `Particles` are checked for collisions with other `PhysicsObject`'s but the collisions are not resolved so they have no physical interaction with other objects.

Modifer

A `Modifier` is defined to be anything that modifies the properities of a set of `PhysicsObject`'s and has an update method. `Modifier` is an abstract class from which all modifiers are extended.

- `Spring`
  `Spring` is an example of a modifier; `Spring` simulates connecting two bodies with a spring. It applies a force to the objects about their center of mass proportional to the spring constant the distance between the objects and the objects masses.

Collision Listener

The `CollisionListener` class is a way to report collisions outside the `Bounce` Package. It has a method that gets called with the objects names and collision normal. Extend the collision listener and add it to the `Bounce World` to get this information.

Vector2

`Vector2` is a class that extends `Point2D.Float`, it has aditional methods for finding normal vectors and vectot opperations. Vector2 is not part of the Bounce package instead it is in a `Util` package. This is because it is used by all aspects of the game not just `Bounce`.

4 Future

For future iterations:

- Added more modifiers
- Add friction
- Improve RigidBody rotation
- Add more constraints to world