Nick Hippen

Intro to Game Programming

Design Document

March 1, 2017

Cannon Ball Game Design Document

# **Game Design**

I decided to go with 8 city blocks for my game. I made the aspect ratio 16:9. I felt that it made the cities large enough to not be too easy to defend while small enough to be manageable. I made gravity noticeable, but not too crazy so that players are able to still click the circles when they are near the bottom. I also made the spawn rate of cannon balls increase over time, as required, with a cap of 0.5 seconds for each meteor spawn.

Wind: Wind varies from -1 (pushes to left) to 1 (pushes to right). Every 4 seconds, it has a chance to change the wind value. This chance increases as the game goes on, and can reach 100%.

Score: Every meteor destroyed gives a base 100 points. This is multiplied by your combo, so if you have a combo of 2, you get 200 points, if you have a combo of 10 you get 1000 points, etc.

Combo: Your combo increases by 1 every time you hit a cannon ball. If you miss a cannon ball (click where there are no cannon balls) your combo will be reset back to 0.

# **edu.unomaha.nhippen.cannonball**

This package contains all of the base files to the game. Most of the files are taken from the book source except for CannonBallApplication.java

## Application.java

This class is the base of the application. It contains all of the provides methods/code to sub-classes.

## CannonBallApplication.java

This class contains a lot of the base game logic. It handles all of the global properties such as wind, and it handles spawning meteors and controlling deletion of city blocks.

## KeyboardInput.java

This class is taken from the book with no changes.

## RelativeMouseInput.java

This class is taken from the book with no changes.

## Utility.java

This class is taken from the book with no changes.

# **edu.unomaha.nhippen.cannonball.vectors**

This package contains all of the base vector classes for the game. None of the files in this package have any direct logic to the game.

## Drawable.java

This class is taken from the slides with no changes.

## Matrix3x3f.java

This class is taken from the book with no changes.

## PointVectorObject.java

This class is a sub class of the VectorObject class. It defines a VectorObject made up of a list of points. This class is intended to allow for different object types in the future, such as a circle, by putting the polygon-specific logic here instead of in VectorObject.class.

## RegularPolygonObject.java

This class is a sub class of the point vector object and simply takes a number of sides and a radius and generates the points automatically.

## Vector2f.java

This class is taken from the book with no personal changes. It has some of the vector math logic discussed in the slides.

## VectorObject.java

This class is similar to the original VectorObject from last assignment. One change is that I added the ability for the vector object to hold the viewpoint matrix. I also made it an abstract class, with the PointVectorObject class containing the rest of the VectorObject logic (see PointVectorObject.java). I also added a new method called updateObject() to handle object update logic inside the object’s class instead of the application class. The final thing added was the ability to mark an object as deleted, for use in other places as needed.

# **edu.unomaha.nhippen.cannonball.vectors.game**

## CityBlock.java

This class is a sub class of PointVectorObject. It defines a city block that the player tries to protect in the game. It has a utility function to help determine if a point is within its hitbox. Its hitbox is defined as the tightest rectangle that can cover all points of the city block.

## Meteor.java

This class is a sub class of RegularPolygonObject. It has 50 sides with a radius of 0.3, making it look like a circle. It has a method to determine if a point is within the shape, with the hitbox of the shape being treated as a circle centered around the shape. It also has the updateObject() method overridden to handle the meteor falling and wind.