|  |
| --- |
| **Concordia University**  **Department of Computer Science**  **and Software Engineering** |

|  |
| --- |
| Animation |

|  |
| --- |
| **Final Project – Jenga Simulator** |

|  |
| --- |
| **COMP 477**  **Winter 2013**  **Professor: Dr. Grogono** |
| **By: Patrick Modafferi**  **9401377**  **03/22/2013** |

I certify that this submission is my original work and meets the Faculty’s Expectations of originality

# Instructions

## Starting the game

1) Go to the JengaSimulator folder

2) Open JengaSimulator.sln with visual studio 2010

3) Press F5 to compile and run in debug mode

If you do not have visual studio

1) go to JengaSimulator \ JengaSimulator \bin\x86\Debug

2) Double click on JengaSimulator.exe

## Controls

|  |  |
| --- | --- |
| TOGGLE KEY(S) | ACTIONS |
| F11 | Full Screen Mode (Recommended) |
| SPACE | Pause / Navigate Menus |
| Space , I | Instructions screen showing all available controls |
| 1 / Shift + 1 | Toggle Rain |
| 2 / Shift + 2 | Toggle Snow |
| F / Shift + F | Toggle Fireworks |
| G / Shift + G | Toggle day times |
| B / Shift + B | Hide/Show bounding boxes |
| CAMERA CONTROLS |  |
| LEFT/RIGHT ARROW | Pan Left and Right |
| UP/DOWN ARROW | Pan Up and Down |
| I/K | Rotate Camera Up and Down |
| J/L | Rotate Camera Left and Right |
| Q/E | Move Camera Forward and Backward |
| Z/C | Rotate the world |
| X | Reset camera |
| HAND CONTROLS |  |
| W/S | Move Hand Up and Down |
| A/D | Move Hand Left and Right |
| Left Click (Hold) | Poke hand forward |
| Right Click + W/S | Rotate Hand Up and Down |
| Right Click + A/D | Rotate Hand Left and Right |
|  |  |

# C:\Users\Patrick\Documents\GitHub\JengaSimulator\ClassDiagram.pngArchitecture

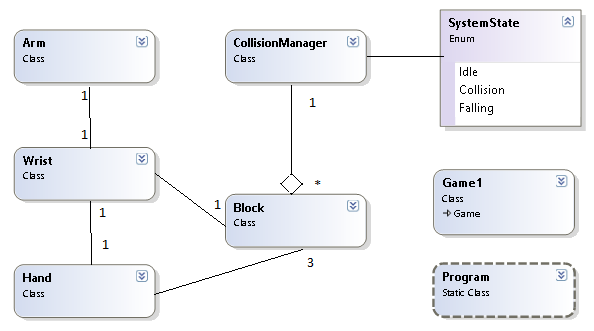


Figure 2‑2‑1 Class Diagram for Particle Systems

Figure 2‑2 Class Diagram for Game Elements

# Particle Systems

For the assignments we created a particle system of particle systems. To accomplish this, all particle systems contain an object called “Bursts” which is a list of Particle systems. This creates a recursive type of relationship in the update and draw functions used for fireworks and shooting stars.

## Star System

The static stars obtain a random 3D coordinate based on two angles (Phi and Theta) mapped to a large sphere. The complexity rises when shooting stars are added. Each shooting star is more than just another particle; it also has a trail and behaviour of its own. Therefore, when we reach the specified spawn rate, another shooting star is added to the “Bursts” discussed earlier. Their initial velocity is then calculated as a tangent to the sphere located at a random static star’s location.

## Fireworks

The fireworks are made of three main components: Firework Rockets, Firework Explosions and Firework Rocket Trails. Each of these components is considered as its own particle system. The rockets get fired upwards with a large vertical velocity and random sideways direction. When the vertical speed reaches zero, the rocket explodes and a new system is created at the source of the rocket’s explosion. Random lifespan and color are given along with directions calculated like in the star system. When all the exploding particles expire, we re-spawn the rocket at the initial source with a new random direction. In addition to what was done for the assignment requirements, extra look and feel was expanded on to make the project more interesting. To avoid recurring patterns and add realism, there is a random offset in the x and z directions for the fire

## Trails

Fireworks and shooting stars contain trails. These trails are simply particle systems that get new particles added to them every time the object updates. Basically, a new stationary particle gets created at the location of the leading object. These trailing particles then have their alpha value decreased and die when the alpha reaches zero.

## Weather System

For this project, we thought it would be interesting to expand on the particle systems made for the assignments and introduce a weather system. The user can toggle between rain, snow and fair weather. Each state modifies the sky color and ground material properties. The snow introduces a simple matte effect with white. The rain adds a specular color with a high shiny constant.

### Rain

The rain particle system simply requires a series of raindrops with downwards velocities. These raindrops start from a random height then drop with an acceleration equal to gravity. Their y-direction scale depends on their downwards velocity to give the appearance of the long faster drops elongating and stretching out.

### Snow

The snow draws a lot of algorithms from the rain system. The main difference is in the aesthetics on the particle itself. To add more realism, a “flurryness” value is introduced. Each particle falls in the general downwards direction, but obtains a random value for X, Y and Z (between negative FLURRY and FLURRY) each frame to offset its position. This adds the effect of being in a blizzard.

# The Hand

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**DISCALIMER**

This project was meant for educational purposes only and is not intended

to be sold or distributed without consent

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Patrick Modafferi

Chris DI Fulvio

04/21/2013