**CS 207 Final Project Report**

*Authors: Yaxiong Cai, Zhenyang Pan*

Table of Contents

1. Project Description 2

2. Collaboration Breakdown 3

3. Specifications and documentation 4

4. Simple Steps to Get and Use The Code 9

5. Pre-Requirement 10

6. Working Demonstration 10

7. Collaboration Evaluation 11

# 1. Project Description

In our project, we designed a robust SDLViewer that can provide user a simple and very cool GUI interacted by mouse and keyboard. After consulting various needs from different groups, our SDLViewer extension have achieved features that are able to:

* Pause/ accelerate/ decelerate/ recover your visualizer by mouse and keyboard at any time.
* Move your objects in 3 dimensions by keyboard at any time.
* Change your objects with 7 different color choices at any time.
* Add/ Remove/ Alter any external forces or constaints at any time.

Moreover, users can add whatever other specific customized functions they like by very simple steps!

Specifically in the demonstration, we combined our model with Meshed Mass - spring group and Collision Detection Extension group to conduct ball collision physics simulations in various ways. In Meshed Mass – spring, we can alter the property of these two balls. Through our SDLViewer, we can move one ball onto the other ball; hit one ball to the other ball through external wind force; and propel one ball to the other ball through internal speed - up.

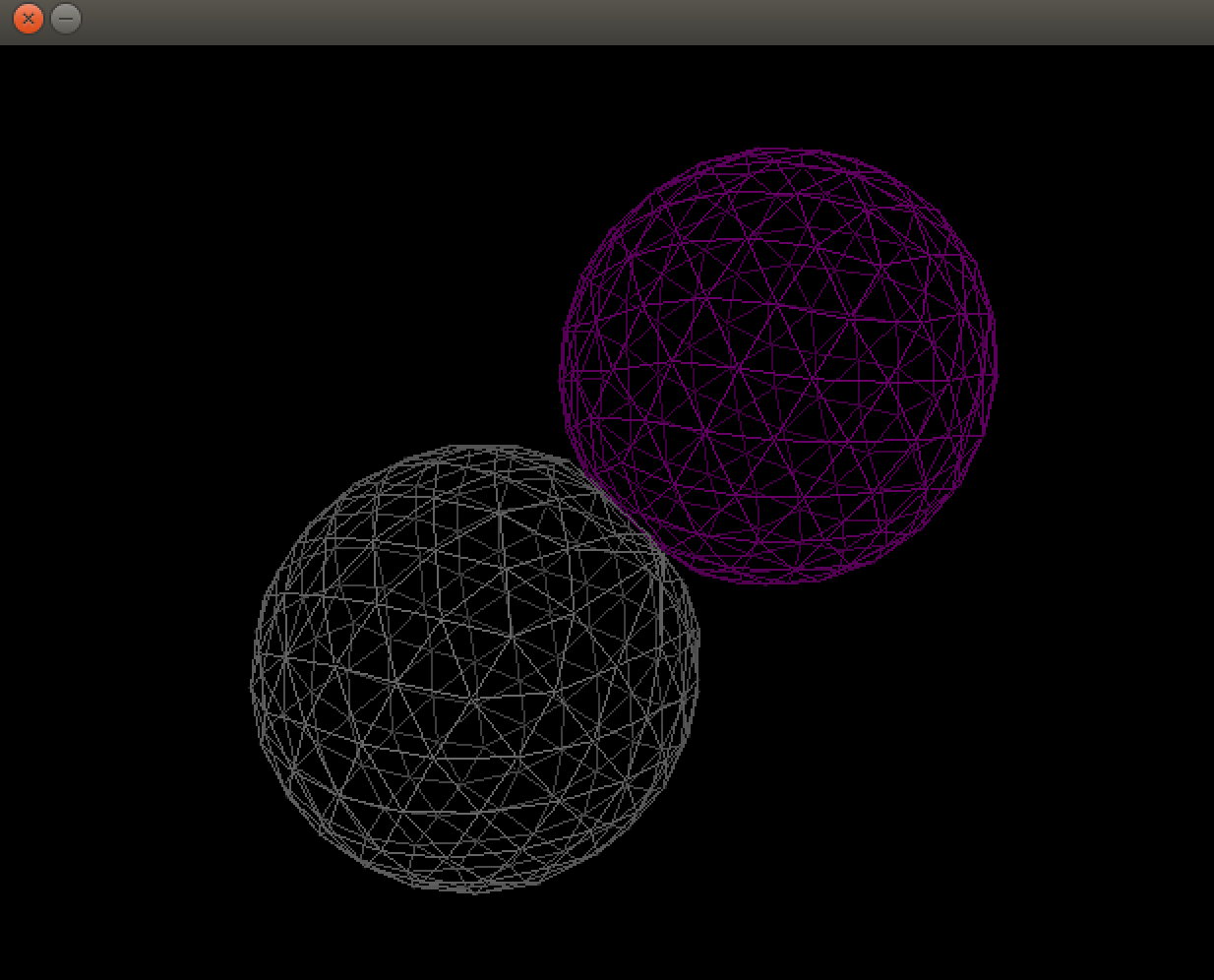


Figure 1. Two balls collision

# 2. Collaboration Breakdown

* **Visualizer extensions**: By our group
* Pause/ accelerate/ decelerate/ recover your visualizer by mouse and keyboard at any time
* Move your objects in 3 dimensions by keyboard at any time
* Change/ Alter/ Hide balls with different color choices
* Initiate/ Increase external wind force to one ball
* Propel one ball directly toward the other ball
* **Meshed Mass-Spring**: By Tian Lan & Xide Xia
* Forces (wind force, gravity, pressure, etc) implementation applying to triangle meshed surface
* An inflated ball based on meshed mass-springs
* **Collision Detection Extension + Morton Code**: By Lukas & Erik
* Detect collision when the two balls collide with each other
* Using Morton Code to a accelerate the searching speed!

# 3. Specifications and documentation

* **Specification for code:**

1. Class and Struct
   1. Pause\_listener: a listener to pause the screen when the right key of the mouse is clicked. When constructed, we store a copy value of the original dt. When this listener is triggered, the dt is set to 0. When clicking again, it will recover back to original dt value.



* 1. Speed\_listener: a listener to adjust the dt value. When constructed, we store a copy value of the original dt. When this listener is triggered, the dt added by 3e-4 if down arrow key is pressed to slow down, or decreased by 3e-4 if up arrow key is pressed to speed up. When clicking left arrow key, it will recover back to original dt value.



* 1. XYZ\_listener: a listener to adjust the x,y,z value of all the node by iterating through the mesh node. Every single click increase or decrease the position value by 0.5.



* 1. Color\_listener: a listener to change the color of the graph by changing the integer values of the color functor. Number key 1,2,3,4,5,6,7 will trigger different color for the graph.



* 1. Force\_listener: a listener to apply the force to the graph. By pressing key f, a force will apply to the graph. The force is controlled by a point with (0,0,0) as the empty force and others as the position of the force.



1. Creating the object

For pause listener and speed listener, the dt needs to be plug in to create the object. For XYZ listener, we need to plug in the mesh which should have an iterator of all the nodes. For the color listener, we need to plug in three integers as the reference for the color. For the force, we need to plug in three integers as the reference for the point.



1. Adding the listener

Using the object created, just add the listener in the viewer.



# 4. Simple Steps to Get and Use The Code

* For CS207 Staffs for Project Demonstration:
* **Getting the package:**

Download the whole package “CS207FinalProject” from:

!!!

* For CS207 Students who want to use this extension:
* **Getting the code:**

Download the SDLViewer.hpp, Viewer\_Extension.hpp files from <https://github.com/JasonCyx/Project-Code>

* **Using the code:**
* Replace your SDLViewer.hpp with our SDLViewer.hpp in your CS207 folder, and place the Viewer\_Extension.hpp in your working directory
* Create the customized listener in your main function and add the reference of the listener objects into the viewer
* On the top of your
* Run the program and press the keys and mouse button
* **Extending the code:**
* Create your own listener class or any functors in Viewer\_Extension.hpp
* Create the customized listener in your main function and add the reference of the listener object into the viewer
* Run the program with the customized keys and mouse button

# 5. Pre-Requirement

* Your code should have the following objects
* Mesh Class with nodes iterator
* Graph Class
* Point Class

# 6. Working Demonstration

* **To run the code in terminal:**
* In terminal change the directory to “CS207FinalProject”
* In terminal $ make two\_balls\_collision
* In terminal $ ./two\_balls\_collision data/sphere200\* data/sphere200\*
* **Interesting features to try:**
* RightClick: Pause/ Unpause the simulation
* Keyboard f: Activate/ Increase external wind force to the pass-in object
* Keyboard m: Propel internal speed to the pass-in object
* Keyboard UP: Accelerate the simulation (Cannot accelerate too much due to limitation of meshed mass spring)
* Keyboard DOWN: Decelerate the simulation
* Keyboard LEFT: Recover the simulation to original simulation setting
* Keyboard w (s): Moving the pass-in object along x-axis increase (decrease)
* Keyboard d (a): Moving the pass-in object along y-axis increase (decrease)
* Keyboard z (x): Moving the pass-in object along z-axis increase (decrease)
* Keyboard number key 1 2 3: Changing different colors for the pass-in object

# 7. Collaboration Evaluation

This demonstration contains three (or four) different prompts:

* **Visualizer extensions** by ZhenyangPan & me**:**

Basically our code is robust and very easy to use as reflected by other groups. Details contained in the group evaluations

* **Meshed Mass-Spring developed** by Tian Lan & Xide Xia **:**

Tian Lan and Xide Xia developed an excellent and cool Meshed Mass-Spring. We obtained their code from Piazza and communicated with them frequently. Their code is very easy to use and well adapted to our code. The wind force and the ball both work quite awesome. We collaborated with Tian Lan & Xide Xia a lot too and they helped us understanding their code as well as with merging their code with other groups.

Some minor limitations:

The properties of the two balls are hard to determine. That is, we need to try different combinations in the main function of value “K”, amount of air inside the ball, and value “b”, the pressure outside the ball, to determine the softness of the two balls almost iteratively.

* **Collision Detection Extension + Morton Code** by Lukas & Erik**:**

Lukas and Erik developed a robust Collision Detector. We obtained the code through Piazza and we have not communicated with Lukas or Erik in person. It is really awesome that their code integrates the Morton Code, for we previously tried collision detector of another group and the one developed by Lukas and Erik is much faster.

However, there is a bug that both my group and other groups found: when the object is very small or very large in terms of distance between two nodes, either the collision cannot be detected or the program fails.