COP290 : Assignment 1
Design Document

Sahil Bansal (2016CSJ0008) & Sahil Singh(2016CSJ0025)

January 28, 2018

1 Overall Design :

- An object-oriented design is intended to be followed where all the properties of the screen are defined in a class **Screen** including all the GUI options provided.
- Similarly, there is a class for **Ball** containing all the information about a ball and the related functions to detect and handle collisions.
- The third main class named **Physics** is the one where we define functions to perform the velocity calculations using vector algebra, so that the ball position after collision can be conveniently updated using those.
- The movement of balls is controlled by **threads** and the communication between them is described in separate section.

2 Sub-Components :

2.1 GUI :

The class **Screen** contains the properties of the box in which the balls are present (width, height & depth). It also includes all the GUI options available including **mouse selection** and **slider/buttons** to update the speed of the balls.

2.2 Balls :

This class defines all the properties of the balls and the related functions. The constructor will intialize the properties(center, velocity, radius and color) randomly whereas the mass is kept proportional to the radius of the ball for convenience.

2.3 Physics :

The physics is handled by the following equations which are obtained after conserving the momentum along the common normal direction ($\hat{n}$) and using the coefficient of

restitution equation.
The center of the balls are represented by vectors $\vec{r_1}$ & $\vec{r_2}$.
So, the direction of common normal is given by :

$$\hat{n} = \frac{\vec{r_1} - \vec{r_2}}{\|\vec{r_1} - \vec{r_2}\|}$$

The final velocities ($\vec{v_1}$ & $\vec{v_2}$) can be then expressed in terms of initial velocities ($\vec{u_1}$ & $\vec{u_2}$), masses of the balls (m_1 & m_2) (considering the coefficient of restitution $e = 1$) as :

$$\vec{v_1} = \vec{u_1} + \left(\frac{2m_2}{m_1 + m_2}(\vec{u_1} - \vec{u_2}).\hat{n}\right)\hat{n} \qquad \rightarrow (1)$$

$$\vec{v_2} = \vec{u_2} + \left(\frac{2m_1}{m_1 + m_2}(\vec{u_2} - \vec{u_1}).\hat{n}\right)\hat{n} \qquad \rightarrow (2)$$

All tools and functions to do these computations like dot product, are defined in the Physics class.

3 Testing Sub-Components :

3.1 GUI :

- The GUI will be tested by seeing that the box is rendered correctly.
- The balls are created with random positions, speed and color. Hence, this should be seen in GUI.
- Upon selection of a ball by mouse click, it must become white.

3.2 Physics :

- The functions can be checked by creating a test file which calls them with some parameters and it can be verified whether the output is correct.

3.3 Balls :

- The collision detection function can be checked to function properly since it will change the color of both the balls (if ball - ball collision) to **black** (or some other color which is not reserved for any ball). If it is a ball to wall collision, then also the color of the ball will be changed.
- The speed updation functions can be verified by printing the updated speed on the console whenever the slider is moved. It can then be seen whether the ball speed is in sync with the slider.

4 Sub-Component Interaction :

4.1 Interaction between balls :

Whenever the balls are close enough, i.e. their center to center distance is less or equal to sum of their radii, the ball collision handler function in Balls class is called which in turn calls the update velocity function in Physics class to get the appropriate velocities for the balls which are then updated.

4.2 Interaction between ball and wall :

When the perpendicular distance between the wall and the center of the ball is less than or equal to its radius, the wall collision handler function in Balls class is called.

5 Thread Interaction :

Each thread controls a ball and in starting, they are in active state. To modify the position and velocity of the ball, they try to pick mutex lock by calling the update velocity (when collisions occurs or slider is moved) and update position functions. The threads are synchronized so that the state of the balls keep on updating. This is the barrier mode of communication. It is being used since it is easy to implement.

6 Variable Ball Speeds :

To update the speed of a ball, it has to be selected with a mouse click after which its color will be changed to white using a function in the Screen class related to mouse input. A slider is displayed on the screen, which then shows the current speed of the ball and then it can be moved to update the speed accordingly. This happens since the velocity variable of ball object is updated. A limit is kept on the maximum speed of the ball due to a limited size of slider, moreover to make the graphics look smooth. Also, a minimum speed is set so that the balls do not move very slowly.

7 Additional Features :

Counter to display average no. of collisions per second :
Each time a collision is encountered, a counter will be incremented. Also, using time library in C++, we will keep track of how much time have been passed since the start of the program. So, we can find the average no. of collisions and display them using a function in Screen class.