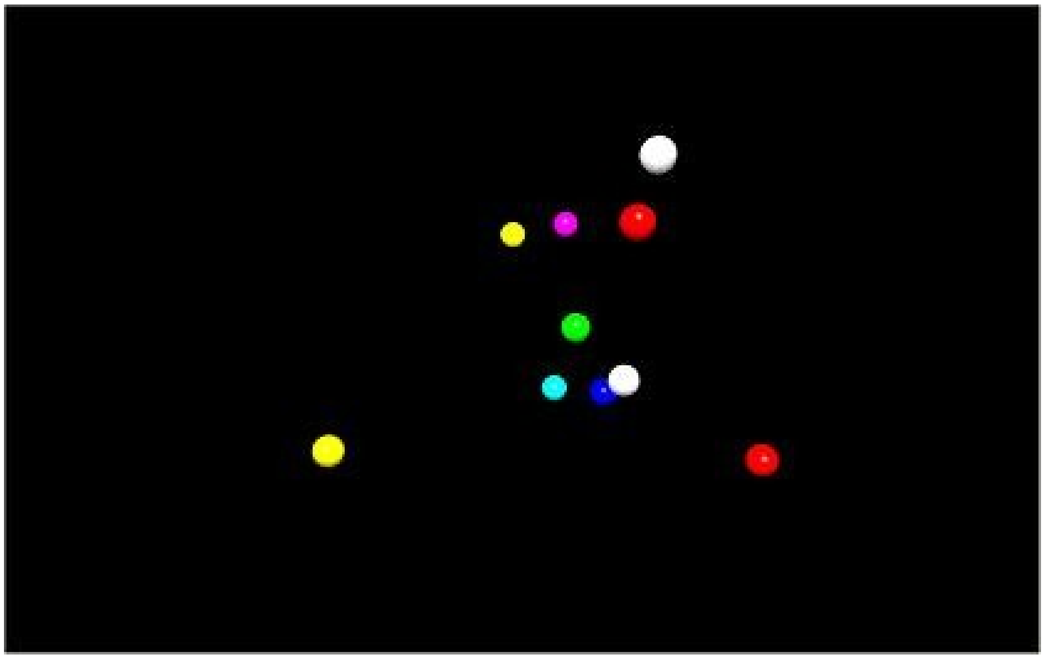
Design Document

COL 290

Assignment1-Screen Saver Application



Author:

Tabish Shaikh(14)

Kunal Parihar(07)

IIT Jammu

1 Overall Design

The design of this application consists of two main features ,a room in which the floor changes as the frame shifts rightward and 3d balls which are programmed to collide inside the room.The user enters the number of balls n at the start of the program .All the n balls are of same size but different color and start at a random initial position in a gravity free space.The user can change the speed of the balls using a slider in the GUI. Threads would communicate with each other during collision with each other and the walls of the room.

2 Sub Components

2.1 GUI

1.display():It will be used to display the application on screen.
2.scrollInput():This will take input from the scroll bar on GUI and increase/decrease the speed of all balls simultaneously.
3.control():Taking input from freeze and move buttons on the GUI it will do the respective action according to users choice.

2.2 Physics

2.2.1 Ball to Wall collision:

Collision of a wall with a ball is detected when:

$$|Poswall - Poscentre| <= radius$$

The appropriate changes are made to the ball velocity according to the elasticity (=1 by default).

2.2.2 Ball to Ball Collision:

Collision of a ball is detected when

$$||\vec{r1} - \vec{r2}|| <= radius1 + radius2$$

where $\vec{r1}$ and $\vec{r2}$ denote the centers of the two balls. The new velocities are computed according to the equations mentioned below.

2.2.3 InterBallCollisionFunc():

This function takes position of ball as argument and checks for collision and if collision takes place updates their respective velocities. Collision will be governed by using the vector form of equations and by using Vector class in C++ as they provide an efficient way of computation.

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

$$w_1 = \frac{m_2 e(u_2 - u_1) + m_1 u_1 + m_2 u_2}{m_1 + m_2}$$

$$w_2 = \frac{m_1 e(u_1 - u_2) + m_1 u_1 + m_2 u_2}{m_1 + m_2}$$

$$\vec{v_1} = \vec{u_1} - (\vec{u_1} \cdot \hat{n})\hat{n} + w_1 \hat{n}$$

$$\vec{v_2} = \vec{u_2} - (\vec{u_2} \cdot \hat{n})\hat{n} + w_2 \hat{n}$$

3 Sub-Component Testing

3.0.1 Automated testing ->

Inorder to test each subcomponent we will working with Unit testing application in openGL because it is fast , productive and verifies results efficiently.

3.0.2 Manual Testing ->

1. **Physics :** We will make sure that physics laws work properly.We will manually check for different ball speed and position and try some corner

cases ,using the function BallWallCollision() and InterBallCollisionFunc() , and match with correct output.

2.**GUI:** we will make sure that display() function work. It should give different color balls and button appearance and working is fine.

4 MultiThreading

There are n threads for n balls, wherein each thread is responsible for controlling a single ball. We will be using barrier synchronization.Barrier synchronization is used to synchronize the threads by updating the balls position then wall collisions are detected and updated. Then ball collisions are detected and handled inside a mutex lock to prevent race conditions.

5 Synchronization

For the purpose of synchronising threads we will be using mutex as mutex has a locking mechanism which allows safe race free execution of application.

6 Variable Speed of Ball

Initially all balls will have a certain speed which can controlled using the scrollbar (speed will be changed only after releasing the mouse click from scrollbar) to increase or decrease the speed within a given maximum and minimum speed(maxima and minima are defined so that application works properly). Accordingly the respective speed will be updated in 3d properties class.All the balls then will start moving at that speed.