COP 290
Assignment 1
Screensaver App

Atul Rai
2016CSJ0074

Dwij Upadhyay
2016CSJ0009

1 Overview

Our whole design is modular which consists of many classes including `BounceBall` and `Box`. `Ball` defines the properties of a ball while `Box` defines the area properties in which the balls are moving. The number of balls n is accepted as user input and n balls of different sizes and different masses moving at different velocities are simulated in a box in random places. The user has the option to change the speeds of individual balls by traversing all the balls using keyboard keys. We have included many other keyboard functionalities in our application.

2 Components

1. Class `BounceBall`

   The `BounceBall` Class defines the properties of the ball. It has the following members:

   - `Centre` – This is a `Vector` of the centre coordinates of the ball. [`Vector` is a user-defined class for a mathematical 3-dimensional vector.]
   - `Velocity` – This is a `Vector` which defines the velocity of an individual ball.
   - `Radius` – The radius of the ball.
   - `Mass` – Mass of the ball which is proportional to the radius of the ball.
   - `Red, Green, Blue` – Data members which together define the color of the ball.
   - `appeartBall()` – Member function which initializes the centre, velocity, radius and color of the ball uniformly at random in a specified range using random generator engine.

2. Class `Box`

   The `Box` class defines the properties of the box in which balls are confined. It has the following members:

   - `boxWidth` - The width of the box.
   - `boxHeight` - The height of the box.
   - `boxDepth` - The depth of the box.

3. Class `Terrain`

   The `Terrain` Class defines the properties of the terrain. It has the following members:

   - `Centre` – This is a `Vector` of the centre coordinates of the sphere shaped terrain. [`Vector` is a user-defined class for a mathematical 3-dimensional vector.]
   - `Radius` – The radius of the sphere shaped terrain..
   - `Red, Green, Blue` – Data members which together define the color of the obstacle.
   - `appeartTerrain()` – Member function which initializes the centre, radius and color of the obstacle uniformly at random in a specified range using random generator engine.

4. Class `Button`

   The `Button` Class defines the properties of the button. It has the following members:

   - `xPos` – This is the x-position of the button on the screen.
   - `yPos` – This is the y-position of the button on the screen.
   - `butWidth` – This indicates the width of a button.
   - `butHeight` – This indicates the height of a button.
   - `Red, Green, Blue` – This is a `Vector` of the colour used for background of button.
   - `butText` – Data menber for storing the text displayed on the button.
   - `isPressed` – This shows whether the button is pressed or not.

5. GUI

   The `GUI` class is the class which we have formed in main(). This is responsible for all of out on-screen display items.

- `showBall()` – It is responsible for drawing the ball .
- `showBox()` – It is responsible for drawing the box .
- `showButton()` – It is responsible for drawing the button .
- `showTerrain()` – It is responsible for drawing the terrain .
- `randInitBalls()` – It randomly initializes all the properties of all the balls and push them into an array.
- `randInitObst()` – It randomly initializes all the properties of all the obstacles and push them into an array.
- `keyboardInput()` – It detects keyboard input from the user and acts accordingly. It can be used as an alternative to the GUI button to increase/decrease ball speed (More in the *Variable Ball Speed* section).

6. Physics

   The `Physics` Class defines the whole physics used in the application. It has the following member functions:

   - wallCollision(): This function accepts one ball and box, and apply the physics associated with the collision of single ball with box which includes the collision detection with the box and update the velocity of the ball after the collision with the box.Collision of a wall with a ball is detected when:

     $$|Pos_{wall} - Pos_{centre}| \leq radius$$

   - ballCollisionDetection(): This function accepts the array of the balls in the form of vector and detects if there is collision taking place in between the balls. It stores the state of balls in a boolean 2D array and updates the array if ball collision is detected. Then it again traverse the array to check which of the balls are getting collided and calls the update() function for them. Collision of a ball is detected when

     $$||\vec{r_1} - \vec{r_2}|| \leq radius_1 + radius_2$$

     where $\vec{r_1}$ and $\vec{r_2}$ denote the centers of the two balls.
     The new velocities are computed according to the equations mentioned below.
   - updateBalls(): This function accepts two balls, say ball 1 and ball 2, which have collided and updates their velocities.We are calculating their final velocities using vector calculations which eventually saves us from the lengthy calculations by using Trignometry. For carrying out vector operations we have made our own class named `Vector` which includes operations like `add`, `subtract`, `modulus`,

`dotProduct`, `normalize` and `scale`. Using these functions, the new velocities are computed using the equations:

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

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

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

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

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

.

- terrainCollisionDetection(): This function accepts the array of the balls and array of obstacles forming terrain in the form of vector and detects if there is collision taking place in between the balls. It stores the state of balls and obstacles in a boolean 2D array and updates the array if terrain- ball collision is detected. Then it again traverse the array to check which of the terrain-ball are getting collided and calls the updateObstacles() function for them.Collision of a ball with terrain is detected when

  $$||\vec{r_1} - \vec{r_2}|| \leq radius_1 + radius_2$$

  where $\vec{r_1}$ denote centre of ball and $\vec{r_2}$ denote the center of spherical obstacle.

- updateObstacle(): This function accepts a ball and a terrain, between which the collision have taken place and update the velocity of the ball after collsion.We again here use the similar type of physics as used in updateBalls() function to update the velocity of the ball.

3 Multi-Threading

There are n threads for n balls, wherein each thread is responsible for controlling a single ball. Currently barrier synchronization is used to synchronize the threads. Firstly, the ball positions are updated. Then wall collisions are detected and updated. Then ball collisions are detected and handled inside a mutex lock to prevent race conditions.

4 Variable Ball Speeds

The user can select a ball with a left mouse click which will make the ball white in color. The ball's speed can then be increased or decreased between a pre-defined maximum or minimum by pressing the relevant button or through keyboard input, whichever is preferred by the user. The ball's velocity which is maintained in the `Ball` object will be updated accordingly. The model for movement and collisions ensures that the physics does not break. When the next ball is selected the previously selected ball will return to its original color and the new ball will become white.

5 Controls

We have include following controls in our application:

- Keyboard Controls :
  - Number of Balls : First of all the user has power to enter(input) any number of balls, that he or she want to enter into the box.
  - Increase Speed : '+' allow the user to increase the speed of a selected Ball.
  - Decrease Speed : '-' allow the user to decrease the speed of a selected Ball.
  - Selection : A ball can be selected using 'a' and 'd' keyboard keys. When a ball is selected its colour will get changed to white so that it can be easily determined which ball is selected by user.
  - Pause/Play : Space Bar allows the user to pause the simulation, and resume the same, as per his convenience. And one more thing, you can observe that as we pause or play using spacebar, the GUI Button's text will also get changed to "Play" or "Pause". Isn't it interesting!
  - Camera Controls : The Up, Down, Left and Right arrow keys allow the user to change the view of the camera, allowing the user to view the Box, and the Balls from different angles,while facilitating the selection of Balls. Up and Down arrow keys will move the camera in and out of the box respectively whereas Left and Right arrow keys allow user to left and right around the box.
  - Change Shape: We have additionally added the the functionality for changing the shape of our moving balls using the keyboard keys 's' and 'w'. This functionality is just for fun and if still is less then we have another functionality of changing the shape of our terrain using keys 'f', 'h', 't' and 'g'.
- Mouse Controls :

- Decrease Speed : Sometimes using the keyboard controls can make you fall in some trouble. It's because sometimes you may increase the speed of the particular ball too much that it transfers its speed to other balls getting you in trouble as the speed balls of all the balls will eventually will increase to a heck. So, for the same, we have designed a GUI Button '-' which will help you taking the speed of all the balls down to a safe value so that you can still enjoy this application without any issue.

6 Other Specifications

- We have additionally added texture to our walls. We have tried to give this application a theme of a tunnel so we have added the textures to the walls and the bottom surface is blue coloured which depits the water flowing below.
- We have made three GUI Buttons but unfortunately we were not be able to made them all functioning. Our '-' button is functioning properly. The Pause button is working through keyboard controls as you can see while pressing spacebar on keyboard the text written on Pause onscreen button changes to "Play" and again pressing spacebar,it again changes back to "Pause". Also as you can see while pressing onscreen Buttons they change their colour.This is an additional functionality added by us.
- Our Application can take input of number of balls somewhere in between 60-80.
- You can even have the feel of walking down the cave using the arrow keys in our application