COP 290 Assignment 1

Haroun Habeeb 2013CS10225
Kabir Chhabra 2013CS50287
Harman Kumar 2013CS10224

January 14, 2015

Problem Statement : Assignment Link

1 Overall Design

In the project three classes have been used namely Ball, Table and Screensaver.The number of balls is accepted from the user and the velocity corresponding to the balls can be increased or decreased by the user using predefined keys. The motion of every ball is controlled by a separate thread and collisions between two balls or between the balls and the boundary are handled.

2 Sub Components

- **Class Ball :**

  The Class Ball has data members for storing the radius of the ball and the X and Y co-ordinates of it's centre and its velocity. The mass of the ball is assumed to be proportional to the cube of it's radius.
  The member functions of the class Ball include the set and get functions, a display function and a reshape function. The set and get functions are used to access the aforementioned data members and modify their values respectively. The display function is used to display the ball and the reshape function is used to handle the case when the window is resized by the user.
  When an object of type ball is created, it is randomly assigned a centre, a radius, and a velocity. It is also ensured that the balls do not overlap at the time of creation.

- **Class Table :**

  The Class Table has data members corresponding to the coordinates of the vertices of the table and a vector for storing the colour of the table.

The member function display is used to display the table and the function reshape is used to resize the table if the screen size is changed.

- **Class ScreenSaver :**

  An object of the Class ScreenSaver is created when the program starts executing. It creates N (Number of balls entered at the time of execution) objects of the class Ball, N pthreads, one to control the motion of each ball, N mutex locks and one object of the class table. The data members are as follows:

  1. **isPaused** (Boolean) : To check whether the screensaver has been paused or is running.
  2. **isFullScreen** (Boolean) : To check whether the screen is in fullscreen mode or not.
  3. **numThreads** (Integer) : To store the number of balls the user wants on the screen.
  4. **windowID** (Integer) : To store the ID of the window corresponding to the object of the Class ScreenSaver.
  5. **table** (Table) : To store the table in the background of the window.
  6. **balls** (Array of type Ball) : To store the N balls that move on the table.
  7. **threads** (vector of pthread_t) : These are the N threads that control one ball each.
  8. **threadUpdate** (Vector of Boolean) : This controls the individualThread function.
  9. **vecMutex** (vector of mutex locks): It's used to lock the parameters of each ball so that only one thread can access it at a certain point of time.

  The member functions of the class are the following:

  1. **void* individualThread(void*)** : This function is what each of the n-pthreads executes. It is triggered after every iteration of the display() function. The purpose of this function is to update the position of the ball by updating the xCentre and the yCentre. It also checks for collisions and in case of one, it updates the velocities of the balls.
  2. **void exitter()** : The exitter function is called upon pressing the Escape key. It joins the threads, closes the window and ensures that there are no memory leaks.
  3. **void init()** : The init() function handles the initialization of the glut window. It takes care of one-time statements for the GUI to

run.

4. **void makeObjects(args)** : The args are references to the locations where objects will be made. The objects we are talking about are the table and the balls.
5. **void reshape(int,int)** : This function handles the reshaping of the window.
6. **void keyHandler(char,int,int)** : This function handles the inputs from the user in the form of pressing of keys. For example, the Escape key quits the screenSaver.
7. **void display()** : The display function is called by glut's mainLoop . It calls the display function of the balls and the table, then proceeds to tell each individual thread to update its variables. It also posts a redisplay request.
8. **void execute()** : This is where the flow of the program is controlled. It calls functions in the right order, initializes variables and then starts the glutMainLoop.

The subcomponents, segregated by functionality, are :

- **GUI** : The GUI is handled by giving the initialization part to one class, and then allowing the balls and the table to display themselves.
- **Physics** : The physics are handled by the threads. The threads call a solve() function which accepts as parameters velocities, masses and co-ordinates. It then returns the same values AFTER collision.
- **Threading** : The updates of the co-ordinates and velocities of balls are handled by threads. They communicate through barrier communication (accessing static global variables). Mutex is applied at appropirate locatons.

3 Testing

We intend to use gdb for testing. The components that require testing are :

- **Table :**

  An object of type table was created by passing 4 points to it's parameterized constructor and was then displayed on the window. Then the window was then resized in order to check the correctness of the reshape member function.

- **Ball :**

The ball provides most of the functionality to our application. Hence we test it by testing various tiny portions.

The ball can be tested in progressive steps , such as drawing it, moving it, multiple balls, managing them on threads and so on and so forth.

To test for threads' funtioning, we will allow the display loop to run once while the thread controls the ball's state parameters. We will then ensure that the ball's variables have changed as expected. This is to identify potential issues with thread synchronization.

- **Physics :** Since our physics part is a set of equations in a seperate function, we execute this function for a set of values to ensure that the output is correct, afterwhich, we ensure that all corner-cases are handled. The following equations were solved to obtain

$$\hat{n} = \frac{\vec{R_1} - \vec{R_2}}{\parallel \vec{R_1} - \vec{R_2} \parallel} \tag{1}$$

$$\vec{V_1'} := \vec{V_1} + (\frac{m_2}{m_1 + m_2}((1-e)\vec{V_1} - (1+e)\vec{V_2})\cdot \hat{n})\hat{n} \tag{2}$$

$$\vec{V_2'} := \vec{V_2} + (\frac{m_1}{m_1 + m_2}((1-e)\vec{V_2} - (1+e)\vec{V_1})\cdot \hat{n})\hat{n} \tag{3}$$

- **Memory Handling** : Valgrind will be used to ensure that there are no memory leaks i.e. to ensure that all dynamically allocated memory is released by the time the program is terminated.
- **Thread Synchronization** : We check the number of active threads at a given point of time and ensure that its always one.

4 Sub-component Interaction

This heading could be divided into the following sub-components:

- **Collision between two Balls :**

  If the distance between the centres of twoballs is less than or equal to the sum of their radii then the collision condition for the two balls is satisfied else not. If there was a collision between two balls then the solveBallCollision function was called that took two objects of type Ball and computed their new velocities based on the equations of motion and assuming perfectly elastic collision.

- **Collision between a Ball and a Wall :**

In this type of collision the wall is assumed to have infinite mass.
If the perpendicular distance between the centre of the ball and the wall is less than the radius of the ball then this collision occours else not.
This collison was also handled by the solveWallCollision function that took an object of type Ball as parameter and modified its velocity using the equations of motion and assuming perfectly elastic collision.

5 Interthread Communication

Interthread communication is done through barrier communication. Threads modify global static variables, namely, the state variables of the balls.

To ensure that this part doesn't face issues, we lock the portion of code where the thread is modifying the values.

Our intention is to use conditional variables to signal between threads. The same idea should also allow us to implement a one-to-one communication between threads.

6 Ball Speed

The X and Y components of the velocity corresponding to each ball are stored in the object of the class Ball and can lie in a certain range. When a collision between two balls or between a ball and the wall occours, these parameters are modified in accordance with the equations derived from Newton's Laws of Motion.

7 Future Endeavours :

During the execution of the screensaver, the user would be provided with the following options:

- **Selection of a ball:**

  The user would be given the option of selecting a ball using the left and right arrow keys. Once the desired ball has been selected, the following operations could be performed :

  1. The user can increase or decrease the speed of the ball using the up and down arrow keys.
  2. The user can delete the ball by pressing the 'D' character key.
  3. The user can change the colour of the ball by pressing the 'C' character key.

- **Efficiency of collision detection :** Presently, the detection is $\mathcal{O}(n^2)$. Improving it by using pixel hashing is a possiblity.
- **Adding a ball :**

  The user can add a ball by pressing the 'A' character key.
- **Pause :**

  The user can pause and unpause the screensaver by pressing 'Spacebar'.
- **FullScreen Mode :**

  The user can toggle between fullscreen mode by pressing the 'F' character key.