COP 290 Assignment 1

Kabir Chhabra , Harman Kumar , Haroun Habeeb

January 13, 2015

Abstract

Problem Statement : Assignment Link

The project was written in C++ using OpenGL, pthreads, mutex locks and it is aimed at ubuntu operating system. The collisions between the balls were considered to be perfectly elastic and the table was considered to be frictionless.

1 Overall Design

In the project three classes have been used namely Ball, Table and Screensaver. The user enters the number of balls and the velocity corresponding to the balls can be increased or decreased by the user. Every ball runs on a thread and collisions between two balls and between the balls and the boundary are handled.

2 Sub Components

The Class Ball has data members for storing the X and Y co-ordinates of it's centre, radius of the ball, X and Y components of it's velocity and 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 that are used to access the above mentioned data members and modify their values respectively, a function to display the ball and a function to modify the location in accordance with the screen size if it's changed.
When an object of type ball is created, it is randomly assigned a centre, radius, velocity.

The class Table has data members corresponding to the coordinates of the vertices and a vector for storing the colour of the table. It has a member function display to display the table and a member function reshape to resize the table if the screen size is changed.

An object of the Class ScreenSaver is called when the program starts executing. It creates N (Number of balls entered at the time of execution) objects of the class Ball, N pthreads 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 execution is paused or running.
2. isFullScreen (Boolean) : To check whether the screen is in fullscreen mode or not.
3. numThreads (Integer) : The number of balls the user wants on the screen.
4. windowID (Integer) : The ID of the window corresponding to the objectof the Class ScreenSaver.
5. table (Table) : The table in the background of the window.
6. balls (Array of type Ball) : 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 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 xCentres,yCentres, velocities of the balls. To do so, it also checks for collisions.
2. void exitter() : The exitter function is called upon pressing the Escape key. It handles all the threads, joins them, destroys all the custom made objects. Essentially just memory handling.
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 reshaping of the window.
6. void keyHandler(char,int,int) : This function handles the 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, seperated by functionality, are :

- GUI : The GUI was 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

The components that require testing are :

1. GUI : Because the display() functions are modular, we just need to test the display function of the ball and that of the table.
2. Physics : Since the physics is put into a function, testing the correctness of the function is easy, using dry runs.
3. Collision Handling : Perfected by eyesight and physics.
4. Memory Handling : valgrind will be used to ensure that there are no memory leaks.

4 Sub-component Interaction

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.

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.