COP290: Design Practices
Design Document
n-Ball Screensaver

Vaibhav Vashisht 2016CSJ0002
Pratik Parmar 2016CSJ0049

January 28, 2018

n-Ball ScreenSaver

1 *Overall Design*

GUI window of the Project consists of ScreenSaver along with GUI components for controlling the ScreenSaver,showing the date and time at the top of Window and inspirational quotes at the bottom of the window.Background of the screensaver changes with time. Command line interface of the program will also give access to various functionallity. Number of balls in the screensaver initially can be given as input with the makefile and can also be given through commandline.Positions, size and velocity of the balls will be random initially and can be changed by the user accordingly. Source code of the program is supposed to contain class for GUI Components, balls and Thread, which might increase or decrease as the project moves further.

2 *Sub-components*

1.Class Ball
It holds-: Coordinates of centre of the ball,(x,y,z). Velocity components of the ball. Radius of the ball. Mass of the ball proportional to

$$r^3$$

Red,Green,Ball-define color of the ball. Various functions-to handle collsions,resize function which is called when user changes the size of the ball,function to change velocity vector of the ball
2. Class Cube
Contains detail about size of the box,functions used for rotating the box,moving ground frame are also in this class.
3.GUI

It is controlled by the following functions:
display(): It initializes the state of the screen,with balls at random positions and cube of given size,front view is displayed intially.It also displays the buttons,background and text-field as mentioned in the previous document.
mouseInput(): Detects mouse input from the user and performs the action defined according the button on which its clicked,we will try also to extend this feature for selecting balls.
keyBoardInput(): Detects keyboard input from the user and performs required action, currently we have planned for using space button for play/pause,we may program some more buttons for some other features.
3.Physics: Threads handle the physics of the repective ball. Collision of ball is handled by the function collsionHandler(). Mechanics of the ball is explained below: Velocity of ball along the direction of collsion changes as per the eqn:

$$|\vec{v}_{1f_{11}}| = \frac{\{m_1\vec{v}_1 + m_2\{(1+e)\vec{v}_2 - e\vec{v}_1\}\}.\vec{u}}{m_1+m_2}$$
$$|\vec{v}_{2f_{11}}| = \frac{\{m_2\vec{v}_2 + m_1\{(1+e)\vec{v}_1 - e\vec{v}_2\}\}.\vec{u}}{m_1+m_2}$$

where ,

$$\vec{u} = \frac{\vec{r}_2 - \vec{r}_1}{|\vec{r}_2 - \vec{r}_1|}$$

.

$$\vec{v}_{1f}\perp = \vec{v}_{1i}\perp$$

$$\vec{v}_{2f}\perp = \vec{v}_{2i}\perp$$

Final velocity can be obtained by vector sum of these velocities resp. User can also pass an extra argument in the terminal for specifying the coefficient of restitution e for collsions.

3 ***Testing***

GDB has been used to debug any bug occcuring within the program and Valgrind has been used to check any memory leakage is occuring because of the threads.
Balls: Each functionality of the ball such as size,speed etc can be tested individually by putting random value in and comparing the output with results expected.
Collisions: print statements has been used to check collisions occuring between the balls and between balls and walls.
Threads: With the use of valgrind we can check if the threads are working properly or not.
Physics: By putting value in the formulae we can check if we are obtaining desired output or not.
Buttons: Each button can be checked individually by pressing and observing if we obtain the tested result,print command can be used to check if we get right value of the attribute.

4 ***Interaction***

Both type of collisions are handled by collisonHandler function. Interaction b/w different components is held as follows:
Ball-Ball collsion: If the distance b/w two balls is equal to r1+r2 then collsion occurs b/w them,it is handled by conservation of momentum and using coefficient of restitutuion eqn.
Wall-Ball collsion: This occurs when distance b/w ball and wall is less than its radius.Wall is assumed to be fixed hence momentum is not conserved here(assumed as wall actually moves).Hence the velocity in the direction of wall(wall vector) reverses and becomes e times.This can also be proved by using the equations given above. Other velocity components remain same.

5 *Thread Interaction*

Interaction between the threads have been done using barrier synchronization and we plan to change it to one to one interaction with the help of messageQueue.Through MessageQueue whenever an event occurs to a thread it will put a message in the queue to which other thread can see and respond to it.
The reason behind changing the mode of interaction is that barrier synchronization is ineffective as while a thread is checking for collisions other threads have to wait,hence this process is time taking.We also plan to divide the cube into 8 parts so that we have to check collisions only b/w adjacent quadrants which reduces computation time.

6 *Thread Synchronization*

Synchronization between the threads has been done using mutex. With the help of mutex we can lock a thread and then that thread will start working until the condition variable has changed it's value and unlocked this thread.
This thing will happpen so fast that it would seems like all threads are working simultaneously.

7 *Variable Speed*

We have stored each component velocity of each ball in its respective object of the class Ball.Max Limit of speed is defined for each ball, speed of ball cannot go above that.We have given GUI option for changing complete velocity vector of the selected ball,this will be done through a function having ball object as its argument.This function will change the velocity vector of the ball to the velocity entered in the

text field.
Another option has been provided which will only change the speed of the ball keeping the direction same.It will also be implented using the same function.Each velocity component is made 1.5 times hence the speed becomes 1.5 times in the same direction.