Project Report

# Lab Group: 171

**Group Members(Name and roll number)**:

Toshendra Sharma (TA and Mentor) - 113050013

Tejas Soni (Group Leader) - 110010041

Vamsidhar Y. - 110050051

Vivan Sorab - 115060001

Tarkeshwar Singh - 115280020

Suresh Chand Meena - 100260011

Sunny Kataria - 110040060

Sai Krishna R. - 110040100

Vipul Snehi - 115090010

* **Introduction:**

­The aim of the project is to design a carrom-like game using C++. The reason of picking up this project is not only fun but also the learning that comes along with it. Designing a game like carrom might seem unimportant from a particular viewpoint. But it is the easiest way to understand how to handle problems with graphics in C++. It gives us an understanding of how particular objects can be made to appear like they’re moving by simply erasing and drawing it again over and over again at a different position. It is notable how simple laws of physics can turn out to be tough to apply in C++. But it’s a challenge we accepted. So, all in all, the project chosen by our team is an effort to put our knowledge and understanding of C++ to the test by creating something which can both be fun to create as well as play.

* **Purposes**:

The project is aimed at designing a carrom-like game using C++, and with graphics built using the EZWindows graphics library. The game requires two players, and does not permit a single player to compete against a computer opponent. The rules that follow are quite similar to those of the traditional carrom (explained later).

* **Outline:**

The game requires each player to move a carrom striker piece using the mouse until positioned as desired within the base lines running along each side of the board. Once positioned appropriately, the player must target a particular direction to shoot towards using the mouse. After doing so, the player must shoot the striker with an appropriate amount of force (which can be done using the mouse to select the power level from the power bar) at the carrom pieces and pocket those. Each piece has a particular point value. Pocketing the striker will result in negative points being awarded.

* **Game Rules:** As stated above the game is similar to the game popularly known as “carrom”, said to be introduced by India. The game is designed for two players. Both players get alternate turns to shoot. The “striker” is used to shoot other pieces on the board. The striker has to be placed between the base lines. There are 20 pieces, five of each of the following colors: Red, Blue, Black and Green. Each color has an associated value to it:

Cyan – 50 points

Blue – 40 points

Green – 25 points

Black – 10 points

All colors are open to both players and they score points corresponding to the piece they sink. For sinking the “striker” a penalty of 10 points is given. At the end of the game, the person with more points wins.

* **Division of Work:**

**SRS Documentation:**

Vivan, Vamsi, Tejas

**Background Code for Collisions:**

Tejas, Vivan, Vamsi

**Graphics:**

Tejas, Suresh, Sunny, Sai Krishna, Vipul, Tarkeshwar

**Game Controls:**

Tejas, Vamsi, Vivan, Sunny, Tarkeshwar

* **System requirements and how to run the game:**
* Ubuntu(10.04 or higher)
* Build-essential package/GCC compiler
* EzWindows for Linux/Ubuntu
* How to run the game :

First, open the terminal window. Go to the folder having the complete program using the cd command. Next, to compile the main program type: ./compile Main

Next, to run the program type: ./Main

Now, enter the player names and press enter. The Welcome Screen will appear. Select Start Game to start the game or explore more options like how to play and credits.

* **Implementation:**

Firstly, and introductory page was created displaying the different options available for the user such as How to play, Credits, Start game and Exit game. Each of the different pages were designed using EzWindows. The background of the carrom was created using EZWindows. The movement of pieces was shown using simple loops to create and erase the pieces over and over again at different positions to give the feeling of movement. The collision with other pieces and the walls of the carrom was shown by implementing simple physics laws of collision. Changing the line of motion by manipulating co-ordinates of the piece is an easier way to put it. Two objects named player\_1 and player\_2 were created and their scores maintained assigning points on every sink of a piece or the striker. And in the end the scores are displayed.

The main function contains only the codes to accept the names of the players and it calls the function which opens the welcome screen. Rest is taken care of by the header files.

#include<iostream>

#include<cstring>

#include<stdio.h>

#include<cassert>

#include"bitmap.h"

#include"position.h"

#include<cmath>

#include<cassert>

#include"ezwin.h"

#include"rect.h"

#include"circle.h"

#include"ellipse.h"

#include"shape.h"

#include"Player.h"

#include"WelcomeScreen.h"

#include"Credits.h"

#include"How\_to\_play.h"

#include"CARROM.h"

#include"Shoot.h"

#include"score\_screen.h"

using namespace std;

//The main function

int ApiMain(){

char player[20];

//Accepting the name of the two players

cout<<"Please enter the name of Player 1 and 2."<<endl;

cout<<"Player 1: ";

gets(player);

Player\_1.set\_name(player);//Copying the name to the Player's object

cout<<"Player 2: ";

gets(player);

Player\_2.set\_name(player);//Copying the name to the Player's object

Open\_Welcome\_Screen();//Calling function to display the WelcomeScreen

return 0;

}//Main function ends

Detailed description of all header files and the functions it contains:

* **#include<iostream>**

Main file for input-output streaming

* **#include<cstring>**

Header file with default string manipulative functions

* **#include<stdio.h>**

Header file for standard input and output functions. Used here for the function sprint().

* **#include<cassert>**

Header file with function assert() which asserts if given condition is satisfied or not.

* **#include"bitmap.h"**

Header file in EzWindows which was used to define a BitMap object of the various images used in the program.

* **#include"position.h"**

Header file in EzWindows used for the class Position to use various functions available to deal with the position of various objects.

* **#include<cmath>**

Header file with various mathematical functions. Used in the program for atan2() function to calculate tan inverse of the input given to the function.

* **#include"ezwin.h"**

Header file of EzWindows which includes uniform.h and WindowManager.h

* **#include"rect.h"**

Header file of EzWindows which includes the functions and class to make a rectangle in a SimpleWindow.

* **#include"circle.h"**

Header file of EzWindows which includes the functions and class to make a circle in a SimpleWindow.

* **#include"ellipse.h"**

Header file of EzWindows which includes the functions and class to make an ellipse in a SimpleWindow.

* **#include"shape.h"**

Header file of EzWindows which includes the functions and class to draw or erase a shape, etc.

* **#include"Player.h"**

Custom header file made by the team which contains the Player class and all functions needed to manipulate the Player object.

#ifndef PLAYER\_H\_INCLUDED

#define PLAYER\_H\_INCLUDED

#include<stdlib.h>//Used for the atoi() function

class Player{

private:

char name[20];//To store name of the Player

int score;//To store the score of the Player

public:

Player(){//Constructor which initializes the score 0

score=0;

}

void set\_name(char x[20]){//Function to copy the name

strcpy(name,x);

}

void update\_score(int x){//Function to update the score

if(x==0) score=0;//x=0 is used to make the score=0

else score=score+x;//else the score is updated by adding

}

int compare\_score(char\* charscore){//Function which compares //the score of the two players and return’s accordingly

int score2=atoi(charscore);//converting to int from char

if(score>score2) return 1;

else if(score<score2) return -1;

else return 0;

}

char\* give\_name(){//Function which returns the player’s name

return name;

}

char\* give\_score(){//Function which returns the score in char

char \*charscore=(char\*)malloc(4);

sprintf(charscore,"%d",score);

return charscore;

}

}Player\_1,Player\_2;//The two objects for the two players

#endif

* **#include"WelcomeScreen.h"**

Custom header file made by the team which contains the functions to display the Welcome Screen and the mouse click function.

#ifndef WELCOMESCREEN\_H\_INCLUDED

#define WELCOMESCREEN\_H\_INCLUDED

#include"Credits.h"

#include"How\_to\_play.h"

#include"CARROM.h"

SimpleWindow WelcomeScreen("Welcome",20.0,14.0);//Defining the WelcomeScreen window

int Open\_Welcome\_Screen();//Function to open the WelcomeScreen window

int WelcomeScreenMouseClick(const Position &p);//Function to manage the mouse clicks in the WelcomeScreen window

BitMap carrom(WelcomeScreen);//To display carrom image

BitMap the\_war(WelcomeScreen);

//Function to open the Welcome screen and display the contents of the window.

int Open\_Welcome\_Screen(){

WelcomeScreen.Open();

assert(WelcomeScreen.GetStatus()==WindowOpen);

//Background and main titles

WelcomeScreen.RenderRectangle(Position(0.0,0.0),Position(20.0,14.0),White);

//Main title "CARROM-171"

WelcomeScreen.RenderRectangle(Position(0.0,0.0),Position(20.0,4.0),Blue);

CircleShape a(WelcomeScreen,Position(2.5,2.0),Yellow,4.0);

a.Draw();

WelcomeScreen.RenderRectangle(Position(2.5,0.0),Position(4.5,4.0),Blue);

CircleShape f(WelcomeScreen,Position(2.5,2.0),Blue,3.75);

f.Draw();

WelcomeScreen.RenderRectangle(Position(2.5,0.0),Position(4.5,4.0),Blue);

EllipseShape b(WelcomeScreen,Position(8.0,1.0),Yellow,4.0,2.0);

b.Draw();

EllipseShape g(WelcomeScreen,Position(8.0,1.0),Blue,3.75,1.75);

g.Draw();

WelcomeScreen.RenderRectangle(Position(6.0,0.0),Position(8.0,4.0),Blue);

EllipseShape c(WelcomeScreen,Position(5.5,1.0),Yellow,4.0,2.0);

c.Draw();

EllipseShape h(WelcomeScreen,Position(5.5,1.0),Blue,3.75,1.75);

h.Draw();

WelcomeScreen.RenderRectangle(Position(3.5,0.0),Position(5.5,4.0),Blue);

WelcomeScreen.RenderLine(Position(4.0,0.0),Position(3.0,4.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(4.0,0.0),Position(5.0,4.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(3.25,2.5),Position(4.75,2.5),Yellow,0.1);

WelcomeScreen.RenderLine(Position(5.5,0.0),Position(5.5,4.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(5.5,2.0),Position(7.5,4.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(8.0,0.0),Position(8.0,4.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(8.0,2.0),Position(10.0,4.0),Yellow,0.1);

EllipseShape d(WelcomeScreen,Position(11.75,2.0),Yellow,2.5,4.0);

d.Draw();

EllipseShape e(WelcomeScreen,Position(11.75,2.0),Blue,2.25,3.75);

e.Draw();

WelcomeScreen.RenderLine(Position(13.5,0.0),Position(13.5,4.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(15.5,0.0),Position(15.5,4.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(15.5,0.0),Position(14.5,2.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(13.5,0.0),Position(14.5,2.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(16.0,2.0),Position(16.5,2.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(17.0,0.0),Position(17.0,4.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(17.0,0.0),Position(16.5,1.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(16.75,4.0),Position(17.25,4.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(17.75,0.05),Position(18.75,0.05),Yellow,0.1);

WelcomeScreen.RenderLine(Position(18.75,0.0),Position(18.0,4.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(19.5,0.0),Position(19.5,4.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(19.5,0.0),Position(19.0,1.0),Yellow,0.1);

WelcomeScreen.RenderLine(Position(19.25,4.0),Position(19.75,4.0),Yellow,0.1);

//Other 4 options, START GAME, HOW TO PLAY, CREDITS, EXIT and bitmaps

carrom.Load("carrom.xpm");

assert(carrom.GetStatus()==BitMapOkay);

carrom.SetPosition(Position(2.0,6.5));

carrom.Draw();

the\_war.Load("the\_war.xpm");

assert(the\_war.GetStatus()==BitMapOkay);

the\_war.SetPosition(Position(12.5,6.5));

the\_war.Draw();

WelcomeScreen.RenderText(Position(6.0,4.5),Position(14.0,6.5),"START GAME",White,Blue);

WelcomeScreen.RenderText(Position(6.0,6.5),Position(14.0,8.5),"HOW TO PLAY",White,Blue);

WelcomeScreen.RenderText(Position(6.0,8.5),Position(14.0,10.5),"CREDITS",White,Blue);

WelcomeScreen.RenderText(Position(6.0,10.5),Position(14.0,12.5),"EXIT GAME",White,Blue);

WelcomeScreen.RenderText(Position(5.0,12.5),Position(15.0,14.0),"CLICK ON THE OPTION YOU WISH TO SELECT TO PROCEED.",White,Blue);

WelcomeScreen.SetMouseClickCallback(WelcomeScreenMouseClick);

return 1;

}

//Funtion to control the mouse in the WelcomeScreen window

int WelcomeScreenMouseClick(const Position &p){

float x=p.GetXDistance(),y=p.GetYDistance();

if(x<14.0 && x>6.0 && y>8.5 && y<10.5){//Checking if click is on the CREDITS button and acting accordingly

WelcomeScreen.Close();

Open\_Credits\_Screen();

}

else if(x>6.0 && x<14.0 && y>6.5 && y<8.5){//Checking if click is on the HOW TO PLAY button and acting accordingly

WelcomeScreen.Close();

Open\_How\_to\_play\_screen();

}

else if(x>6.0 && x<14.0 && y>10.5 && y<12.5){//Checking if click is on the EXIT button and acting accordingly

WelcomeScreen.Close();

exit(0);

}

else if(x>6.0 && x<14.0 && y>4.5 && y<6.5){//Checking if click is on the START GAME button and acting accordingly

WelcomeScreen.Close();

Open\_CARROM\_screen();

}

else{//Otherwise check for another click

WelcomeScreen.SetMouseClickCallback(WelcomeScreenMouseClick);

}

}

#endif

* **#include"Credits.h"**

Custom header file made by the team which contains the functions to display the credits screen and the mouse click function.

#ifndef CREDITS\_H\_INCLUDED

#define CREDITS\_H\_INCLUDED

SimpleWindow Credits("Credits",16.6,14.0);//Defining Credits window

int CreditsMouseClick(const Position &p);//Function to control the mouse in the credits window

void Open\_Credits\_Screen();//Function to open the Credits window

int Open\_Welcome\_Screen();//Function to open the WelcomeScreen window

//Function to control the mouse in the credits window

int CreditsMouseClick(const Position &p){

float x=p.GetXDistance(),y=p.GetYDistance();

if(x>12.4 && x<16.4 && y<14.0 && y>13.5){//Checking if the click is on the CLICK HERE TO GO BACK button and acting accordingly

Credits.Close();

Open\_Welcome\_Screen();

return 1;

}

else{//Otherwise, looking for another click

Credits.SetMouseClickCallback(CreditsMouseClick);

}

}

//Function to open the Credits window

void Open\_Credits\_Screen(){

Credits.Open();

assert(Credits.GetStatus()==WindowOpen);

//Background and details

Credits.RenderRectangle(Position(0.0,0.0),Position(16.6,14.0),Black);

//Top title "TEAM-171"

Credits.RenderLine(Position(1.0,0.05),Position(3.0,0.05),Cyan,0.1);

Credits.RenderLine(Position(2.0,0.0),Position(2.0,4.0),Cyan,0.1);

Credits.RenderLine(Position(3.5,0.0),Position(3.5,4.0),Cyan,0.1);

Credits.RenderLine(Position(3.5,0.05),Position(5.5,0.05),Cyan,0.1);

Credits.RenderLine(Position(3.5,2.0),Position(4.5,2.0),Cyan,0.1);

Credits.RenderLine(Position(3.5,4.0),Position(5.5,4.0),Cyan,0.1);

Credits.RenderLine(Position(7.0,0.0),Position(6.0,4.0),Cyan,0.1);

Credits.RenderLine(Position(7.0,0.0),Position(8.0,4.0),Cyan,0.1);

Credits.RenderLine(Position(6.25,2.75),Position(7.75,2.75),Cyan,0.1);

Credits.RenderLine(Position(8.5,0.0),Position(8.5,4.0),Cyan,0.1);

Credits.RenderLine(Position(10.5,0.0),Position(10.5,4.0),Cyan,0.1);

Credits.RenderLine(Position(10.5,0.0),Position(9.5,2.0),Cyan,0.1);

Credits.RenderLine(Position(8.5,0.0),Position(9.5,2.0),Cyan,0.1);

Credits.RenderLine(Position(11.0,2.0),Position(12.0,2.0),Cyan,0.1);

Credits.RenderLine(Position(12.5,0.0),Position(12.5,4.0),Cyan,0.1);

Credits.RenderLine(Position(12.5,0.0),Position(12.0,1.0),Cyan,0.1);

Credits.RenderLine(Position(12.125,4.0),Position(12.875,4.0),Cyan,0.1);

Credits.RenderLine(Position(13.0,0.05),Position(14.5,0.05),Cyan,0.1);

Credits.RenderLine(Position(14.5,0.0),Position(13.25,4.0),Cyan,0.1);

Credits.RenderLine(Position(15.25,0.0),Position(15.25,4.0),Cyan,0.1);

Credits.RenderLine(Position(15.25,0.0),Position(14.75,1.0),Cyan,0.1);

Credits.RenderLine(Position(14.875,4.0),Position(15.575,4.0),Cyan,0.1);

//Team members

Credits.RenderText(Position(5.3,4.5),Position(11.3,5.5),"TOSHENDRA SHARMA (T.A. AND MENTOR)",Cyan,Black);

Credits.RenderText(Position(5.3,5.5),Position(11.3,6.5),"TEJAS SONI (TEAM LEADER)",Cyan,Black);

Credits.RenderText(Position(5.3,6.5),Position(11.3,7.5),"VAMSIDHAR Y.",Cyan,Black);

Credits.RenderText(Position(5.3,7.5),Position(11.3,8.5),"VIVAN SORAB",Cyan,Black);

Credits.RenderText(Position(5.3,8.5),Position(11.3,9.5),"TARKESHWAR SINGH",Cyan,Black);

Credits.RenderText(Position(5.3,9.5),Position(11.3,10.5),"SURESH CHAND MEENA",Cyan,Black);

Credits.RenderText(Position(5.3,10.5),Position(11.3,11.5),"SUNNY KATARIA",Cyan,Black);

Credits.RenderText(Position(5.3,11.5),Position(11.3,12.5),"SAI KRISHNA R.",Cyan,Black);

Credits.RenderText(Position(5.3,12.5),Position(11.3,13.5),"VIPUL SNEHI",Cyan,Black);

Credits.RenderText(Position(12.4,13.5),Position(16.4,14.0),"CLICK HERE TO GO BACK",Red,Cyan);

Credits.SetMouseClickCallback(CreditsMouseClick);

}

#endif

* **#include"How\_to\_play.h"**

Custom header file made by the team which contains the functions to display the how to play screen and the mouse click function.

#ifndef HOW\_TO\_PLAY\_H\_INCLUDED

#define HOW\_TO\_PLAY\_H\_INCLUDED

SimpleWindow How\_to\_play("How to play",16.0,14.0);//Defining the How\_to\_play window

int How\_to\_playMouseClick(const Position &p);//Funtion to control the mouse in the How\_to\_play window

void Open\_How\_to\_play\_screen();//Function to open the How\_to\_play window

int Open\_Welcome\_Screen();//Function to open the WelcomeScreen window

//Funtion to control the mouse in the How\_to\_play window

int How\_to\_playMouseClick(const Position &p){

float x=p.GetXDistance(),y=p.GetYDistance();

if(x>11.0 && x<15.0 && y<14.0 && y>13.5){//Checking if the click is on the CLICK HERE TO GO BACK button and acting accordingly

How\_to\_play.Close();

Open\_Welcome\_Screen();

return 1;

}

else{//Otherwise, looking for another click

How\_to\_play.SetMouseClickCallback(How\_to\_playMouseClick);

}

}

//Function to open the How\_to\_play window

void Open\_How\_to\_play\_screen(){

How\_to\_play.Open();

assert(How\_to\_play.GetStatus()==WindowOpen);

//Background and the details

How\_to\_play.RenderRectangle(Position(0.0,0.0),Position(16.0,14.0),Black);

How\_to\_play.RenderText(Position(1.0,0.0),Position(15.0,1.0),"..::RULES OF THE GAME::..",Red,Black);

How\_to\_play.RenderText(Position(1.0,1.0),Position(15.0,1.5),"The game is similar to the original game. The game was introduced in",Red,Black);

How\_to\_play.RenderText(Position(1.0,1.5),Position(15.0,2.0),"India. This game is designed for two players. Both players get alternate",Red,Black);

How\_to\_play.RenderText(Position(1.0,2.0),Position(15.0,2.5),"turns to shoot. The striker is used to shoot other pieces on the board.",Red,Black);

How\_to\_play.RenderText(Position(1.0,2.5),Position(15.0,3.0),"The striker has to be placed between the base lines. There are 16 pieces,",Red,Black);

How\_to\_play.RenderText(Position(1.0,3.0),Position(15.0,3.5),"four of each of the following colors: Cyan, Blue, Green and Black.",Red,Black);

How\_to\_play.RenderText(Position(1.0,3.5),Position(15.0,4.0),"Each color has an associated value to it:",Red,Black);

How\_to\_play.RenderText(Position(1.0,4.0),Position(15.0,4.5),"Cyan - 50 points",Red,Black);

How\_to\_play.RenderText(Position(1.0,4.5),Position(15.0,5.0),"Blue - 40 points",Red,Black);

How\_to\_play.RenderText(Position(1.0,5.0),Position(15.0,5.5),"Green - 25 points",Red,Black);

How\_to\_play.RenderText(Position(1.0,5.5),Position(15.0,6.0),"Black - 10 points",Red,Black);

How\_to\_play.RenderText(Position(1.0,6.0),Position(15.0,6.5),"All colors are open to both players and they score points corresponding",Red,Black);

How\_to\_play.RenderText(Position(1.0,6.5),Position(15.0,7.0),"to the color of the piece they sink in any of the four pockets at the",Red,Black);

How\_to\_play.RenderText(Position(1.0,7.0),Position(15.0,7.5),"corners. For sinking the striker a penalty of 10 points is given. At",Red,Black);

How\_to\_play.RenderText(Position(1.0,7.5),Position(15.0,8.0),"the end of the game, the person with more points wins.",Red,Black);

How\_to\_play.RenderText(Position(1.0,8.0),Position(15.0,9.0),"..::HOW TO PLAY::..",Red,Black);

How\_to\_play.RenderText(Position(1.0,9.0),Position(15.0,9.5),"The first step is to set the position of the striker. Using the mouse,",Red,Black);

How\_to\_play.RenderText(Position(1.0,9.5),Position(15.0,10.0),"click on the position between the base lines where you want the striker",Red,Black);

How\_to\_play.RenderText(Position(1.0,10.0),Position(15.0,10.5),"to be placed. Once you select the position, confirm it by clicking on",Red,Black);

How\_to\_play.RenderText(Position(1.0,10.5),Position(15.0,11.0),"POSITION SET at the bottom-right of the board. Next, select the",Red,Black);

How\_to\_play.RenderText(Position(1.0,11.0),Position(15.0,11.5),"direction in which you want to shoot the striker by clicking in the",Red,Black);

How\_to\_play.RenderText(Position(1.0,11.5),Position(15.0,12.0),"direction. Once you select the angle, confirm it by clicking on ANGLE",Red,Black);

How\_to\_play.RenderText(Position(1.0,12.0),Position(15.0,12.5),"SET. Next, select the power with which you want to shoot the striker",Red,Black);

How\_to\_play.RenderText(Position(1.0,12.5),Position(15.0,13.0),"by clicking in the power bar at the appropriate level. Once done,",Red,Black);

How\_to\_play.RenderText(Position(1.0,13.0),Position(15.0,13.5),"click on POWER SET and then click on SHOOT! to make your shot. Enjoy!",Red,Black);

How\_to\_play.RenderText(Position(11.0,13.5),Position(15.0,14.0),"CLICK HERE TO GO BACK",Cyan,Red);

How\_to\_play.SetMouseClickCallback(How\_to\_playMouseClick);

}

#endif

* **#include"CARROM.h"**

Custom header file made by the team which contains the functions to display the carrom screen, mouse click function, display the pieces, repair the carrom background and calculate theta function. It also contains the definitions to the bitmaps of the objects of striker and the carrom pieces.

#ifndef CARROM\_H\_INCLUDED

#define CARROM\_H\_INCLUDED

#define SIZE\_FACTOR 1.2//Defining the size-factor for the carrom dimensions

SimpleWindow CARROM("Carrom", 15.0\*SIZE\_FACTOR, 12.0\*SIZE\_FACTOR);

Position striker,cyan1,cyan2,cyan3,cyan4,green1,green2,green3,green4,black1,black2,black3,black4,blue1,blue2,blue3,blue4,end\_point;//For the position of the striker and the pieces

int flag=0,turn=0,balls\_remaining=16,foul=0;//For miscellaneous uses, apparent from the variable name

float power=0.0;//To store the power which user selects

float striker\_theta=100.0, cyan1\_theta=100.0, cyan2\_theta=100.0, cyan3\_theta=100.0, cyan4\_theta=100.0, blue1\_theta=100.0, blue2\_theta=100.0, blue3\_theta=100.0, blue4\_theta=100.0, green1\_theta=100.0, green2\_theta=100.0, green3\_theta=100.0, green4\_theta=100.0, black1\_theta=100.0, black2\_theta=100.0, black3\_theta=100.0, black4\_theta=100.0;//To store the slope/theta at which each piece moves

char player\_score[15]="Score: ";

BitMap Striker(CARROM);

BitMap Cyan1(CARROM);

BitMap Cyan2(CARROM);

BitMap Cyan3(CARROM);

BitMap Cyan4(CARROM);

BitMap Green1(CARROM);

BitMap Green2(CARROM);

BitMap Green3(CARROM);

BitMap Green4(CARROM);

BitMap Blue1(CARROM);

BitMap Blue2(CARROM);

BitMap Blue3(CARROM);

BitMap Blue4(CARROM);

BitMap Black1(CARROM);

BitMap Black2(CARROM);

BitMap Black3(CARROM);

BitMap Black4(CARROM);

int CARROMMouseClick(const Position &p);//Function to control the mouse in the CARROM window

void Open\_CARROM\_screen();//Function to open the CARROM window initially

float calculate\_theta(Position X,Position Y);//Function to calculate the slope of the line joining the two given points

void repair\_carrom\_background();//Function to show the repaired background of the carrom

void display\_pieces();//Function to display the pieces

void Open\_Display\_Score\_Screen();//Function to display the score after the game finishes

void Shoot();//Function to shoot the striker included in Shoot.h

//Function to control the mouse in the CARROM window

int CARROMMouseClick(const Position &p){

float x=p.GetXDistance(),y=p.GetYDistance();

if(x>2.85\*SIZE\_FACTOR && x<8.8\*SIZE\_FACTOR && flag==0){//Setting the position of the striker

Striker.Erase();

if(turn%2==0){

Striker.SetPosition(Position(p.GetXDistance(),2.4\*SIZE\_FACTOR));

striker=Striker.GetPosition();

//Restoring CARROM structure and displaying striker

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,2.5\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,2.5\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,2.9\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,2.9\*SIZE\_FACTOR),Black,0.0);

CircleShape G(CARROM,Position(2.7\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

G.Draw();

CircleShape H(CARROM,Position(2.7\*SIZE\_FACTOR,3.01\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

H.Draw();

CircleShape K(CARROM,Position(9.3\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

K.Draw();

CircleShape L(CARROM,Position(9.3\*SIZE\_FACTOR,3.01\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

L.Draw();

CircleShape O(CARROM,Position(3.0\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

O.Draw();

CircleShape P(CARROM,Position(3.01\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

P.Draw();

CircleShape Q(CARROM,Position(9.0\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

Q.Draw();

CircleShape R(CARROM,Position(9.0\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

R.Draw();

CARROM.RenderLine(Position(2.0\*SIZE\_FACTOR,2.0\*SIZE\_FACTOR),Position(4.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(2.0\*SIZE\_FACTOR,10.0\*SIZE\_FACTOR),Position(4.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Black,0.0);

Striker.Draw();

}

else{

Striker.SetPosition(Position(p.GetXDistance(),9.0\*SIZE\_FACTOR));

striker=Striker.GetPosition();

//Restoring CARROM structure and displaying striker

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,9.5\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,9.5\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,9.1\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,9.1\*SIZE\_FACTOR),Black,0.0);

CircleShape I(CARROM,Position(2.7\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

I.Draw();

CircleShape J(CARROM,Position(2.7\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

J.Draw();

CircleShape M(CARROM,Position(9.3\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

M.Draw();

CircleShape N(CARROM,Position(9.3\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

N.Draw();

CircleShape S(CARROM,Position(3.0\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

S.Draw();

CircleShape T(CARROM,Position(3.01\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

T.Draw();

CircleShape U(CARROM,Position(9.0\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

U.Draw();

CircleShape V(CARROM,Position(9.0\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

V.Draw();

CARROM.RenderLine(Position(10.0\*SIZE\_FACTOR,10.0\*SIZE\_FACTOR),Position(8.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(10.0\*SIZE\_FACTOR,2.0\*SIZE\_FACTOR),Black,0.0);

Striker.Draw();

}

display\_pieces();//displaying pieces

CARROM.SetMouseClickCallback(CARROMMouseClick);

}

else if(x<12.0\*SIZE\_FACTOR && flag==1){//Setting the angle

Position r=striker;//Getting position of Striker

r.SetXDistance(r.GetXDistance()+0.34);//Setting position as the center of the striker

r.SetYDistance(r.GetYDistance()+0.34);//Same

striker\_theta=calculate\_theta(r,p);//calculating the slope/theta which is made by the line joining the center of the striker and the position of click

end\_point.SetXDistance(r.GetXDistance()+2.0\*cos(striker\_theta));

end\_point.SetYDistance(r.GetYDistance()+2.0\*sin(striker\_theta));

//Restoring structure of CARROM and drawing the angle line

if(turn%2==0){

CARROM.RenderRectangle(Position(1.0\*SIZE\_FACTOR,1.0\*SIZE\_FACTOR),Position(11.0\*SIZE\_FACTOR,4.45\*SIZE\_FACTOR),White);

CircleShape C(CARROM,Position(1.3\*SIZE\_FACTOR,1.3\*SIZE\_FACTOR),Black,0.6\*SIZE\_FACTOR);

C.Draw();

CircleShape E(CARROM,Position(10.7\*SIZE\_FACTOR,1.3\*SIZE\_FACTOR),Black,0.6\*SIZE\_FACTOR);

E.Draw();

CARROM.RenderLine(Position(2.5\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(2.5\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(2.9\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(2.9\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(9.5\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(9.5\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(9.1\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(9.1\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,2.5\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,2.5\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,2.9\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,2.9\*SIZE\_FACTOR),Black,0.0);

CircleShape G(CARROM,Position(2.7\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

G.Draw();

CircleShape H(CARROM,Position(2.7\*SIZE\_FACTOR,3.01\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

H.Draw();

CircleShape K(CARROM,Position(9.3\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

K.Draw();

CircleShape L(CARROM,Position(9.3\*SIZE\_FACTOR,3.01\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

L.Draw();

CircleShape O(CARROM,Position(3.0\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

O.Draw();

CircleShape P(CARROM,Position(3.01\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

P.Draw();

CircleShape Q(CARROM,Position(9.0\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

Q.Draw();

CircleShape R(CARROM,Position(9.0\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

R.Draw();

CARROM.RenderLine(Position(2.0\*SIZE\_FACTOR,2.0\*SIZE\_FACTOR),Position(4.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(10.0\*SIZE\_FACTOR,2.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(4.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(3.8\*SIZE\_FACTOR,3.6\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(4.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(3.6\*SIZE\_FACTOR,3.8\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(8.2\*SIZE\_FACTOR,3.6\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(8.4\*SIZE\_FACTOR,3.8\*SIZE\_FACTOR),Black,0.0);

Striker.SetPosition(striker);

Striker.Draw();

}

else{

CARROM.RenderRectangle(Position(1.0\*SIZE\_FACTOR,7.55\*SIZE\_FACTOR),Position(11.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),White);

CircleShape D(CARROM,Position(1.3\*SIZE\_FACTOR,10.7\*SIZE\_FACTOR),Black,0.6\*SIZE\_FACTOR);

D.Draw();

CircleShape F(CARROM,Position(10.7\*SIZE\_FACTOR,10.7\*SIZE\_FACTOR),Black,0.6\*SIZE\_FACTOR);

F.Draw();

CARROM.RenderLine(Position(2.5\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(2.5\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(2.9\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(2.9\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(9.5\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(9.5\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(9.1\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(9.1\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,9.5\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,9.5\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,9.1\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,9.1\*SIZE\_FACTOR),Black,0.0);

CircleShape I(CARROM,Position(2.7\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

I.Draw();

CircleShape J(CARROM,Position(2.7\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

J.Draw();

CircleShape M(CARROM,Position(9.3\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

M.Draw();

CircleShape N(CARROM,Position(9.3\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

N.Draw();

CircleShape S(CARROM,Position(3.0\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

S.Draw();

CircleShape T(CARROM,Position(3.01\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

T.Draw();

CircleShape U(CARROM,Position(9.0\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

U.Draw();

CircleShape V(CARROM,Position(9.0\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

V.Draw();

CARROM.RenderLine(Position(10.0\*SIZE\_FACTOR,10.0\*SIZE\_FACTOR),Position(8.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(2.0\*SIZE\_FACTOR,10.0\*SIZE\_FACTOR),Position(4.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(4.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Position(3.6\*SIZE\_FACTOR,8.2\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(4.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Position(3.8\*SIZE\_FACTOR,8.4\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Position(8.2\*SIZE\_FACTOR,8.4\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Position(8.4\*SIZE\_FACTOR,8.2\*SIZE\_FACTOR),Black,0.0);

Striker.SetPosition(striker);

Striker.Draw();

}

display\_pieces();//Displaying pieces

CARROM.RenderLine(r,end\_point,Yellow,0.1);//Drawing the angle line

CARROM.SetMouseClickCallback(CARROMMouseClick);

}

else if(x>13.0\*SIZE\_FACTOR && x<14.0\*SIZE\_FACTOR && y>3.0\*SIZE\_FACTOR && y<8.0\*SIZE\_FACTOR && flag==2){//Setting the power

CARROM.RenderRectangle(Position(13.0\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(14.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),White);

CARROM.RenderRectangle(Position(13.0\*SIZE\_FACTOR,p.GetYDistance()),Position(14.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Red);//Displaying the power

power=8.0\*SIZE\_FACTOR-p.GetYDistance();//Calculating the power

CARROM.SetMouseClickCallback(CARROMMouseClick);

}

else if(x>12.0\*SIZE\_FACTOR && x<15.0\*SIZE\_FACTOR && y>11.0\*SIZE\_FACTOR && y<12.0\*SIZE\_FACTOR && flag==3){//Checking if click is on the SHOOT! button

Shoot();//Initiating the shooting process

turn++;//Changing turns

Striker.Erase();

flag=0;

power=0.0;//Getting ready for the next shot

CARROM.RenderRectangle(Position(13.0\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(14.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),White);//Changing the button

CARROM.RenderRectangle(Position(12.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,12.0\*SIZE\_FACTOR),Red);

CARROM.RenderText(Position(13.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),Position(14.0\*SIZE\_FACTOR,12.0\*SIZE\_FACTOR),"POSITION SET",Yellow,Red);

if(turn%2==0){

Striker.SetPosition(Position(5.7\*SIZE\_FACTOR,2.4\*SIZE\_FACTOR));

Striker.Draw();

striker=Striker.GetPosition();

}

else{

Striker.SetPosition(Position(5.7\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR));

Striker.Draw();

striker=Striker.GetPosition();

}

if(balls\_remaining!=0)CARROM.SetMouseClickCallback(CARROMMouseClick);

else {

cyan1\_theta= cyan2\_theta= cyan3\_theta= cyan4\_theta= blue1\_theta= blue2\_theta= blue3\_theta= blue4\_theta= black1\_theta= black2\_theta= black3\_theta= black4\_theta= green1\_theta= green2\_theta= green3\_theta= green4\_theta= 100.0;//Assigning the defaults again before next game

turn=0;

balls\_remaining=16;

foul=0;

Open\_Display\_Score\_Screen();//To open the screen to display the final score.

}

}

else if(x>12.0\*SIZE\_FACTOR && x<15.0\*SIZE\_FACTOR && y>11.0\*SIZE\_FACTOR && y<12.0\*SIZE\_FACTOR && flag==0){//Checking if click is on POSITION SET button

CARROM.RenderRectangle(Position(12.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,12.0\*SIZE\_FACTOR),Red);

CARROM.RenderText(Position(13.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),Position(14.0\*SIZE\_FACTOR,12.0\*SIZE\_FACTOR),"ANGLE SET",Yellow,Red);

flag++;

if(turn%2==0){//Assigning default values of angle

CARROM.RenderLine(Position(striker.GetXDistance()+0.34,striker.GetYDistance()+0.34),(end\_point=Position(striker.GetXDistance()+0.34,4.0\*SIZE\_FACTOR)),Yellow,0.1);

striker\_theta=3.141592654/2;

}

else{

CARROM.RenderLine(Position(striker.GetXDistance()+0.34,striker.GetYDistance()+0.34),(end\_point=Position(striker.GetXDistance()+0.34,8.0\*SIZE\_FACTOR)),Yellow,0.1);

striker\_theta=-3.141592654/2;

}

CARROM.SetMouseClickCallback(CARROMMouseClick);

}

else if(x>12.0\*SIZE\_FACTOR && x<15.0\*SIZE\_FACTOR && y>11.0\*SIZE\_FACTOR && y<12.0\*SIZE\_FACTOR && flag==1){//Checking if click is on ANGLE SET button

CARROM.RenderRectangle(Position(12.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,12.0\*SIZE\_FACTOR),Red);

CARROM.RenderText(Position(13.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),Position(14.0\*SIZE\_FACTOR,12.0\*SIZE\_FACTOR),"POWER SET",Yellow,Red);

flag++;

CARROM.SetMouseClickCallback(CARROMMouseClick);

}

else if(x>12.0\*SIZE\_FACTOR && x<15.0\*SIZE\_FACTOR && y>11.0\*SIZE\_FACTOR && y<12.0\*SIZE\_FACTOR && flag==2){//Checking if click is on POWER SET button

CARROM.RenderRectangle(Position(12.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,12.0\*SIZE\_FACTOR),Red);

CARROM.RenderText(Position(13.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),Position(14.0\*SIZE\_FACTOR,12.0\*SIZE\_FACTOR),"SHOOT!",Yellow,Red);

flag++;

CARROM.SetMouseClickCallback(CARROMMouseClick);

}

else if(x>12.0\*SIZE\_FACTOR && x<15.0\*SIZE\_FACTOR && y>0.0\*SIZE\_FACTOR && y<1.0\*SIZE\_FACTOR){//Checking if click is on the EXIT button

CARROM.Close();

exit(0);

}

else//Otherwise, looking for another click

CARROM.SetMouseClickCallback(CARROMMouseClick);

}

//Function to open the CARROM window initially

void Open\_CARROM\_screen(){

CARROM.Open();

assert(CARROM.GetStatus()==WindowOpen);

//For the carrom frame

CARROM.RenderRectangle(Position(0.0\*SIZE\_FACTOR,0.0\*SIZE\_FACTOR),Position(12.0\*SIZE\_FACTOR,12.0\*SIZE\_FACTOR),Black);

CARROM.RenderRectangle(Position(1.0\*SIZE\_FACTOR,1.0\*SIZE\_FACTOR),Position(11.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),White);

//For the power bar and players and instruction message

CARROM.RenderRectangle(Position(12.0\*SIZE\_FACTOR,0.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,12.0\*SIZE\_FACTOR),Yellow);

CARROM.RenderRectangle(Position(13.0\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(14.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),White);

CARROM.RenderRectangle(Position(13.0\*SIZE\_FACTOR,(8.0-power)\*SIZE\_FACTOR),Position(14.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Red);

CARROM.RenderText(Position(13.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Position(14.0\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),"Power Bar",Red,Yellow);

CARROM.RenderRectangle(Position(13.0\*SIZE\_FACTOR,0.0\*SIZE\_FACTOR),Position(14.0\*SIZE\_FACTOR,1.0\*SIZE\_FACTOR),Black);

CARROM.RenderText(Position(13.0\*SIZE\_FACTOR,0.0\*SIZE\_FACTOR),Position(14.0\*SIZE\_FACTOR,1.0\*SIZE\_FACTOR),"Exit",Cyan,Black);

CARROM.RenderText(Position(12.0\*SIZE\_FACTOR,1.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,2.0\*SIZE\_FACTOR),Player\_1.give\_name(),Blue,Yellow);

CARROM.RenderText(Position(12.0\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,10.0\*SIZE\_FACTOR),Player\_2.give\_name(),Blue,Yellow);

strcat(player\_score,Player\_1.give\_score());

CARROM.RenderText(Position(12.0\*SIZE\_FACTOR,2.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),player\_score,Blue,Yellow);

strcpy(player\_score,"Score: ");

strcat(player\_score,Player\_2.give\_score());

CARROM.RenderText(Position(12.0\*SIZE\_FACTOR,10.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),player\_score,Blue,Yellow);

strcpy(player\_score,"Score: ");

CARROM.RenderRectangle(Position(12.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,12.0\*SIZE\_FACTOR),Red);

CARROM.RenderText(Position(13.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),Position(14.0\*SIZE\_FACTOR,12.0\*SIZE\_FACTOR),"POSITION SET",Yellow,Red);

//For the corner pockets

CircleShape C(CARROM,Position(1.3\*SIZE\_FACTOR,1.3\*SIZE\_FACTOR),Black,0.6\*SIZE\_FACTOR);

C.Draw();

CircleShape D(CARROM,Position(1.3\*SIZE\_FACTOR,10.7\*SIZE\_FACTOR),Black,0.6\*SIZE\_FACTOR);

D.Draw();

CircleShape E(CARROM,Position(10.7\*SIZE\_FACTOR,1.3\*SIZE\_FACTOR),Black,0.6\*SIZE\_FACTOR);

E.Draw();

CircleShape F(CARROM,Position(10.7\*SIZE\_FACTOR,10.7\*SIZE\_FACTOR),Black,0.6\*SIZE\_FACTOR);

F.Draw();

//For the side lines

CARROM.RenderLine(Position(2.5\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(2.5\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(2.9\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(2.9\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(9.5\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(9.5\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(9.1\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(9.1\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,2.5\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,2.5\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,2.9\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,2.9\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,9.5\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,9.5\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,9.1\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,9.1\*SIZE\_FACTOR),Black,0.0);

//For the smaller red circles

CircleShape G(CARROM,Position(2.7\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

G.Draw();

CircleShape H(CARROM,Position(2.7\*SIZE\_FACTOR,3.01\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

H.Draw();

CircleShape I(CARROM,Position(2.7\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

I.Draw();

CircleShape J(CARROM,Position(2.7\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

J.Draw();

CircleShape K(CARROM,Position(9.3\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

K.Draw();

CircleShape L(CARROM,Position(9.3\*SIZE\_FACTOR,3.01\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

L.Draw();

CircleShape M(CARROM,Position(9.3\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

M.Draw();

CircleShape N(CARROM,Position(9.3\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

N.Draw();

CircleShape O(CARROM,Position(3.0\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

O.Draw();

CircleShape P(CARROM,Position(3.01\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

P.Draw();

CircleShape Q(CARROM,Position(9.0\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

Q.Draw();

CircleShape R(CARROM,Position(9.0\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

R.Draw();

CircleShape S(CARROM,Position(3.0\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

S.Draw();

CircleShape T(CARROM,Position(3.01\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

T.Draw();

CircleShape U(CARROM,Position(9.0\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

U.Draw();

CircleShape V(CARROM,Position(9.0\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

V.Draw();

//For the lines between the red circles

CARROM.RenderLine(Position(2.0\*SIZE\_FACTOR,2.0\*SIZE\_FACTOR),Position(4.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(2.0\*SIZE\_FACTOR,10.0\*SIZE\_FACTOR),Position(4.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(10.0\*SIZE\_FACTOR,10.0\*SIZE\_FACTOR),Position(8.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(10.0\*SIZE\_FACTOR,2.0\*SIZE\_FACTOR),Black,0.0);

//For the centeral circle

CircleShape m(CARROM,Position(6.0\*SIZE\_FACTOR,6.0\*SIZE\_FACTOR),White,3.0\*SIZE\_FACTOR);

m.Draw();

CircleShape n(CARROM,Position(6.0\*SIZE\_FACTOR,6.0\*SIZE\_FACTOR),Black,0.44\*SIZE\_FACTOR);

n.Draw();

CircleShape l(CARROM,Position(6.0\*SIZE\_FACTOR,6.0\*SIZE\_FACTOR),Red,0.4\*SIZE\_FACTOR);

l.Draw();

//For the arrow at the end of lines between small red circles

CARROM.RenderLine(Position(4.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(3.8\*SIZE\_FACTOR,3.6\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(4.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(3.6\*SIZE\_FACTOR,3.8\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(8.2\*SIZE\_FACTOR,3.6\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(8.4\*SIZE\_FACTOR,3.8\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(4.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Position(3.6\*SIZE\_FACTOR,8.2\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(4.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Position(3.8\*SIZE\_FACTOR,8.4\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Position(8.2\*SIZE\_FACTOR,8.4\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Position(8.4\*SIZE\_FACTOR,8.2\*SIZE\_FACTOR),Black,0.0);

//For striker and pieces

Striker.Load("Striker.xpm");

Striker.SetPosition(Position(5.7\*SIZE\_FACTOR,2.4\*SIZE\_FACTOR));

Striker.Draw();

striker=Striker.GetPosition();

Cyan1.Load("Cyan.xpm");

Cyan1.SetPosition(Position(4.7\*SIZE\_FACTOR,4.7\*SIZE\_FACTOR));

Cyan1.Draw();

cyan1=Cyan1.GetPosition();

Cyan2.Load("Cyan.xpm");

Cyan2.SetPosition(Position(5.45\*SIZE\_FACTOR,5.45\*SIZE\_FACTOR));

Cyan2.Draw();

cyan2=Cyan2.GetPosition();

Cyan3.Load("Cyan.xpm");

Cyan3.SetPosition(Position(6.95\*SIZE\_FACTOR,6.95\*SIZE\_FACTOR));

Cyan3.Draw();

cyan3=Cyan3.GetPosition();

Cyan4.Load("Cyan.xpm");

Cyan4.SetPosition(Position(6.2\*SIZE\_FACTOR,6.2\*SIZE\_FACTOR));

Cyan4.Draw();

cyan4=Cyan4.GetPosition();

Green1.Load("Green.xpm");

Green1.SetPosition(Position(4.7\*SIZE\_FACTOR,6.2\*SIZE\_FACTOR));

Green1.Draw();

green1=Green1.GetPosition();

Green2.Load("Green.xpm");

Green2.SetPosition(Position(5.45\*SIZE\_FACTOR,6.95\*SIZE\_FACTOR));

Green2.Draw();

green2=Green2.GetPosition();

Green3.Load("Green.xpm");

Green3.SetPosition(Position(6.2\*SIZE\_FACTOR,4.7\*SIZE\_FACTOR));

Green3.Draw();

green3=Green3.GetPosition();

Green4.Load("Green.xpm");

Green4.SetPosition(Position(6.95\*SIZE\_FACTOR,5.45\*SIZE\_FACTOR));

Green4.Draw();

green4=Green4.GetPosition();

Blue1.Load("Blue.xpm");

Blue1.SetPosition(Position(4.7\*SIZE\_FACTOR,5.45\*SIZE\_FACTOR));

Blue1.Draw();

blue1=Blue1.GetPosition();

Blue2.Load("Blue.xpm");

Blue2.SetPosition(Position(5.45\*SIZE\_FACTOR,6.2\*SIZE\_FACTOR));

Blue2.Draw();

blue2=Blue2.GetPosition();

Blue3.Load("Blue.xpm");

Blue3.SetPosition(Position(6.2\*SIZE\_FACTOR,6.95\*SIZE\_FACTOR));

Blue3.Draw();

blue3=Blue3.GetPosition();

Blue4.Load("Blue.xpm");

Blue4.SetPosition(Position(6.95\*SIZE\_FACTOR,4.7\*SIZE\_FACTOR));

Blue4.Draw();

blue4=Blue4.GetPosition();

Black1.Load("Black.xpm");

Black1.SetPosition(Position(4.7\*SIZE\_FACTOR,6.95\*SIZE\_FACTOR));

Black1.Draw();

black1=Black1.GetPosition();

Black2.Load("Black.xpm");

Black2.SetPosition(Position(5.45\*SIZE\_FACTOR,4.7\*SIZE\_FACTOR));

Black2.Draw();

black2=Black2.GetPosition();

Black3.Load("Black.xpm");

Black3.SetPosition(Position(6.2\*SIZE\_FACTOR,5.45\*SIZE\_FACTOR));

Black3.Draw();

black3=Black3.GetPosition();

Black4.Load("Black.xpm");

Black4.SetPosition(Position(6.95\*SIZE\_FACTOR,6.2\*SIZE\_FACTOR));

Black4.Draw();

black4=Black4.GetPosition();

CARROM.SetMouseClickCallback(CARROMMouseClick);

}

//Function to calculate the slope of the line joining the two given points

float calculate\_theta(Position X,Position Y){

Y.SetXDistance(Y.GetXDistance()-X.GetXDistance());//Shifting the origin

Y.SetYDistance(Y.GetYDistance()-X.GetYDistance());

float theta=atan2(Y.GetYDistance(),Y.GetXDistance());//Calculating the theta by slope method

return theta;

}

//Function to show the repaired background of the carrom

void repair\_carrom\_background(){

//For the carrom frame

CARROM.RenderRectangle(Position(0.0\*SIZE\_FACTOR,0.0\*SIZE\_FACTOR),Position(12.0\*SIZE\_FACTOR,12.0\*SIZE\_FACTOR),Black);

CARROM.RenderRectangle(Position(1.0\*SIZE\_FACTOR,1.0\*SIZE\_FACTOR),Position(11.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),White);

//For the score

strcat(player\_score,Player\_1.give\_score());

CARROM.RenderRectangle(Position(12.0\*SIZE\_FACTOR,2.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Yellow);

CARROM.RenderText(Position(12.0\*SIZE\_FACTOR,2.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),player\_score,Blue,Yellow);

strcpy(player\_score,"Score: ");

strcat(player\_score,Player\_2.give\_score());

CARROM.RenderRectangle(Position(12.0\*SIZE\_FACTOR,10.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),Yellow);

CARROM.RenderText(Position(12.0\*SIZE\_FACTOR,10.0\*SIZE\_FACTOR),Position(15.0\*SIZE\_FACTOR,11.0\*SIZE\_FACTOR),player\_score,Blue,Yellow);

strcpy(player\_score,"Score: ");

//For the side lines

CARROM.RenderLine(Position(2.5\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(2.5\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(2.9\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(2.9\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(9.5\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(9.5\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(9.1\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Position(9.1\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,2.5\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,2.5\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,2.9\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,2.9\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,9.5\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,9.5\*SIZE\_FACTOR),Black,0.05);

CARROM.RenderLine(Position(3.0\*SIZE\_FACTOR,9.1\*SIZE\_FACTOR),Position(9.0\*SIZE\_FACTOR,9.1\*SIZE\_FACTOR),Black,0.0);

//For the smaller red circles

CircleShape G(CARROM,Position(2.7\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

G.Draw();

CircleShape H(CARROM,Position(2.7\*SIZE\_FACTOR,3.01\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

H.Draw();

CircleShape I(CARROM,Position(2.7\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

I.Draw();

CircleShape J(CARROM,Position(2.7\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

J.Draw();

CircleShape K(CARROM,Position(9.3\*SIZE\_FACTOR,3.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

K.Draw();

CircleShape L(CARROM,Position(9.3\*SIZE\_FACTOR,3.01\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

L.Draw();

CircleShape M(CARROM,Position(9.3\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

M.Draw();

CircleShape N(CARROM,Position(9.3\*SIZE\_FACTOR,9.0\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

N.Draw();

CircleShape O(CARROM,Position(3.0\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

O.Draw();

CircleShape P(CARROM,Position(3.01\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

P.Draw();

CircleShape Q(CARROM,Position(9.0\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

Q.Draw();

CircleShape R(CARROM,Position(9.0\*SIZE\_FACTOR,2.7\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

R.Draw();

CircleShape S(CARROM,Position(3.0\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

S.Draw();

CircleShape T(CARROM,Position(3.01\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

T.Draw();

CircleShape U(CARROM,Position(9.0\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Black,0.4\*SIZE\_FACTOR);

U.Draw();

CircleShape V(CARROM,Position(9.0\*SIZE\_FACTOR,9.3\*SIZE\_FACTOR),Red,0.35\*SIZE\_FACTOR);

V.Draw();

//For the lines between the red circles

CARROM.RenderLine(Position(2.0\*SIZE\_FACTOR,2.0\*SIZE\_FACTOR),Position(4.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(2.0\*SIZE\_FACTOR,10.0\*SIZE\_FACTOR),Position(4.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(10.0\*SIZE\_FACTOR,10.0\*SIZE\_FACTOR),Position(8.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(10.0\*SIZE\_FACTOR,2.0\*SIZE\_FACTOR),Black,0.0);

//For the centeral circle

CircleShape m(CARROM,Position(6.0\*SIZE\_FACTOR,6.0\*SIZE\_FACTOR),White,3.0\*SIZE\_FACTOR);

m.Draw();

CircleShape n(CARROM,Position(6.0\*SIZE\_FACTOR,6.0\*SIZE\_FACTOR),Black,0.44\*SIZE\_FACTOR);

n.Draw();

CircleShape l(CARROM,Position(6.0\*SIZE\_FACTOR,6.0\*SIZE\_FACTOR),Red,0.4\*SIZE\_FACTOR);

l.Draw();

//For the arrow at the end of lines between small red circles

CARROM.RenderLine(Position(4.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(3.8\*SIZE\_FACTOR,3.6\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(4.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(3.6\*SIZE\_FACTOR,3.8\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(8.2\*SIZE\_FACTOR,3.6\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,4.0\*SIZE\_FACTOR),Position(8.4\*SIZE\_FACTOR,3.8\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(4.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Position(3.6\*SIZE\_FACTOR,8.2\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(4.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Position(3.8\*SIZE\_FACTOR,8.4\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Position(8.2\*SIZE\_FACTOR,8.4\*SIZE\_FACTOR),Black,0.0);

CARROM.RenderLine(Position(8.0\*SIZE\_FACTOR,8.0\*SIZE\_FACTOR),Position(8.4\*SIZE\_FACTOR,8.2\*SIZE\_FACTOR),Black,0.0);

//For the corner pockets

CircleShape C(CARROM,Position(1.3\*SIZE\_FACTOR,1.3\*SIZE\_FACTOR),Black,0.6\*SIZE\_FACTOR);

C.Draw();

CircleShape D(CARROM,Position(1.3\*SIZE\_FACTOR,10.7\*SIZE\_FACTOR),Black,0.6\*SIZE\_FACTOR);

D.Draw();

CircleShape E(CARROM,Position(10.7\*SIZE\_FACTOR,1.3\*SIZE\_FACTOR),Black,0.6\*SIZE\_FACTOR);

E.Draw();

CircleShape F(CARROM,Position(10.7\*SIZE\_FACTOR,10.7\*SIZE\_FACTOR),Black,0.6\*SIZE\_FACTOR);

F.Draw();

return;

}

//Function to display the pieces

void display\_pieces(){//Checking if the pieces are not at rest and also not already potted

if(cyan1.GetXDistance()!=-100.0 && cyan1\_theta!=101.0) Cyan1.Draw();

if(cyan2.GetXDistance()!=-100.0 && cyan2\_theta!=101.0) Cyan2.Draw();

if(cyan3.GetXDistance()!=-100.0 && cyan3\_theta!=101.0) Cyan3.Draw();

if(cyan4.GetXDistance()!=-100.0 && cyan4\_theta!=101.0) Cyan4.Draw();

if(blue1.GetXDistance()!=-100.0 && blue1\_theta!=101.0) Blue1.Draw();

if(blue2.GetXDistance()!=-100.0 && blue2\_theta!=101.0) Blue2.Draw();

if(blue3.GetXDistance()!=-100.0 && blue3\_theta!=101.0) Blue3.Draw();

if(blue4.GetXDistance()!=-100.0 && blue4\_theta!=101.0) Blue4.Draw();

if(black1.GetXDistance()!=-100.0 && black1\_theta!=101.0) Black1.Draw();

if(black2.GetXDistance()!=-100.0 && black2\_theta!=101.0) Black2.Draw();

if(black3.GetXDistance()!=-100.0 && black3\_theta!=101.0) Black3.Draw();

if(black4.GetXDistance()!=-100.0 && black4\_theta!=101.0) Black4.Draw();

if(green1.GetXDistance()!=-100.0 && green1\_theta!=101.0) Green1.Draw();

if(green2.GetXDistance()!=-100.0 && green2\_theta!=101.0) Green2.Draw();

if(green3.GetXDistance()!=-100.0 && green3\_theta!=101.0) Green3.Draw();

if(green4.GetXDistance()!=-100.0 && green4\_theta!=101.0) Green4.Draw();

return;

}

#endif

* **#include"Shoot.h"**

Custom header file made by the team which is the heart of the program. It has the functions to shoot and the two functions for collision.

#ifndef SHOOT\_H\_INCLUDED

#define SHOOT\_H\_INCLUDED

#include"CARROM.h"

SimpleWindow Display\_Foul("FOUL!",5.0,2.0,Position(8.0,7.0));//Window to display the message when striker sinks

void Shoot();//Function to shoot the striker

void collision\_pieces(Position a, Position b, float &theta1, float &theta2);//Function for the collisions of pieces taking place during the shooting process

void collision\_striker(Position a, float &theta1);//Function for the collisions of striker taking place during the shooting process

//The most important function Shoot, to shoot the striker and move the pieces according to the laws of physics

void Shoot(){

float i=1.0;

while(1){

if((cyan1.GetXDistance()<=1.6\*SIZE\_FACTOR && cyan1.GetYDistance()<=1.6\*SIZE\_FACTOR) || (cyan1.GetXDistance()<=1.6\*SIZE\_FACTOR && cyan1.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (cyan1.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && cyan1.GetYDistance()<=1.6\*SIZE\_FACTOR) || (cyan1.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && cyan1.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){//Checking if the piece hits the pocket. Also, the upcoming if loops are same

if(turn%2==0) Player\_1.update\_score(50);

else Player\_2.update\_score(50);

Cyan1.Erase();

cyan1.SetXDistance(-100.0);//Used as a flag to determine if the piece has been pocketed

cyan1.SetYDistance(5.0);//Used so that the above conditions don't match in the following executions of the loop

cyan1\_theta=101.0;//Used as a flag

balls\_remaining--;

}

if((cyan2.GetXDistance()<=1.6\*SIZE\_FACTOR && cyan2.GetYDistance()<=1.6\*SIZE\_FACTOR) || (cyan2.GetXDistance()<=1.6\*SIZE\_FACTOR && cyan2.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (cyan2.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && cyan2.GetYDistance()<=1.6\*SIZE\_FACTOR) || (cyan2.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && cyan2.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(50);

else Player\_2.update\_score(50);

Cyan2.Erase();

cyan2.SetXDistance(-100.0);

cyan2.SetYDistance(5.0);

cyan2\_theta=101.0;

balls\_remaining--;

}

if((cyan3.GetXDistance()<=1.6\*SIZE\_FACTOR && cyan3.GetYDistance()<=1.6\*SIZE\_FACTOR) || (cyan3.GetXDistance()<=1.6\*SIZE\_FACTOR && cyan3.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (cyan3.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && cyan3.GetYDistance()<=1.6\*SIZE\_FACTOR) || (cyan3.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && cyan3.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(50);

else Player\_2.update\_score(50);

Cyan3.Erase();

cyan3.SetXDistance(-100.0);

cyan3.SetYDistance(5.0);

cyan3\_theta=101.0;

balls\_remaining--;

}

if((cyan4.GetXDistance()<=1.6\*SIZE\_FACTOR && cyan4.GetYDistance()<=1.6\*SIZE\_FACTOR) || (cyan4.GetXDistance()<=1.6\*SIZE\_FACTOR && cyan4.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (cyan4.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && cyan4.GetYDistance()<=1.6\*SIZE\_FACTOR) || (cyan4.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && cyan4.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(50);

else Player\_2.update\_score(50);

Cyan4.Erase();

cyan4.SetXDistance(-100.0);

cyan4.SetYDistance(5.0);

cyan4\_theta=101.0;

balls\_remaining--;

}

if((blue1.GetXDistance()<=1.6\*SIZE\_FACTOR && blue1.GetYDistance()<=1.6\*SIZE\_FACTOR) || (blue1.GetXDistance()<=1.6\*SIZE\_FACTOR && blue1.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (blue1.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && blue1.GetYDistance()<=1.6\*SIZE\_FACTOR) || (blue1.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && blue1.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(40);

else Player\_2.update\_score(40);

Blue1.Erase();

blue1.SetXDistance(-100.0);

blue1.SetYDistance(5.0);

blue1\_theta=101.0;

balls\_remaining--;

}

if((blue2.GetXDistance()<=1.6\*SIZE\_FACTOR && blue2.GetYDistance()<=1.6\*SIZE\_FACTOR) || (blue2.GetXDistance()<=1.6\*SIZE\_FACTOR && blue2.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (blue2.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && blue2.GetYDistance()<=1.6\*SIZE\_FACTOR) || (blue2.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && blue2.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(40);

else Player\_2.update\_score(40);

Blue2.Erase();

blue2.SetXDistance(-100.0);

blue2.SetYDistance(5.0);

blue2\_theta=101.0;

balls\_remaining--;

}

if((blue3.GetXDistance()<=1.6\*SIZE\_FACTOR && blue3.GetYDistance()<=1.6\*SIZE\_FACTOR) || (blue3.GetXDistance()<=1.6\*SIZE\_FACTOR && blue3.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (blue3.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && blue3.GetYDistance()<=1.6\*SIZE\_FACTOR) || (blue3.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && blue3.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(40);

else Player\_2.update\_score(40);

Blue3.Erase();

blue3.SetXDistance(-100.0);

blue3.SetYDistance(5.0);

blue3\_theta=101.0;

balls\_remaining--;

}

if((blue4.GetXDistance()<=1.6\*SIZE\_FACTOR && blue4.GetYDistance()<=1.6\*SIZE\_FACTOR) || (blue4.GetXDistance()<=1.6\*SIZE\_FACTOR && blue4.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (blue4.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && blue4.GetYDistance()<=1.6\*SIZE\_FACTOR) || (blue4.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && blue4.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(40);

else Player\_2.update\_score(40);

Blue4.Erase();

blue4.SetXDistance(-100.0);

blue4.SetYDistance(5.0);

blue4\_theta=101.0;

balls\_remaining--;

}

if((black1.GetXDistance()<=1.6\*SIZE\_FACTOR && black1.GetYDistance()<=1.6\*SIZE\_FACTOR) || (black1.GetXDistance()<=1.6\*SIZE\_FACTOR && black1.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (black1.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && black1.GetYDistance()<=1.6\*SIZE\_FACTOR) || (black1.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && black1.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(10);

else Player\_2.update\_score(10);

Black1.Erase();

black1.SetXDistance(-100.0);

black1.SetYDistance(5.0);

black1\_theta=101.0;

balls\_remaining--;

}

if((black2.GetXDistance()<=1.6\*SIZE\_FACTOR && black2.GetYDistance()<=1.6\*SIZE\_FACTOR) || (black2.GetXDistance()<=1.6\*SIZE\_FACTOR && black2.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (black2.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && black2.GetYDistance()<=1.6\*SIZE\_FACTOR) || (black2.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && black2.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(10);

else Player\_2.update\_score(10);

Black2.Erase();

black2.SetXDistance(-100.0);

black2.SetYDistance(5.0);

black2\_theta=101.0;

balls\_remaining--;

}

if((black3.GetXDistance()<=1.6\*SIZE\_FACTOR && black3.GetYDistance()<=1.6\*SIZE\_FACTOR) || (black3.GetXDistance()<=1.6\*SIZE\_FACTOR && black3.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (black3.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && black3.GetYDistance()<=1.6\*SIZE\_FACTOR) || (black3.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && black3.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(10);

else Player\_2.update\_score(10);

Black3.Erase();

black3.SetXDistance(-100.0);

black3.SetYDistance(5.0);

black3\_theta=101.0;

balls\_remaining--;

}

if((black4.GetXDistance()<=1.6\*SIZE\_FACTOR && black4.GetYDistance()<=1.6\*SIZE\_FACTOR) || (black4.GetXDistance()<=1.6\*SIZE\_FACTOR && black4.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (black4.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && black4.GetYDistance()<=1.6\*SIZE\_FACTOR) || (black4.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && black4.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(10);

else Player\_2.update\_score(10);

Black4.Erase();

black4.SetXDistance(-100.0);

black4.SetYDistance(5.0);

black4\_theta=101.0;

balls\_remaining--;

}

if((green1.GetXDistance()<=1.6\*SIZE\_FACTOR && green1.GetYDistance()<=1.6\*SIZE\_FACTOR) || (green1.GetXDistance()<=1.6\*SIZE\_FACTOR && green1.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (green1.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && green1.GetYDistance()<=1.6\*SIZE\_FACTOR) || (green1.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && green1.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(25);

else Player\_2.update\_score(25);

Green1.Erase();

green1.SetXDistance(-100.0);

green1.SetYDistance(5.0);

green1\_theta=101.0;

balls\_remaining--;

}

if((green2.GetXDistance()<=1.6\*SIZE\_FACTOR && green2.GetYDistance()<=1.6\*SIZE\_FACTOR) || (green2.GetXDistance()<=1.6\*SIZE\_FACTOR && green2.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (green2.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && green2.GetYDistance()<=1.6\*SIZE\_FACTOR) || (green2.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && green2.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(25);

else Player\_2.update\_score(25);

Green2.Erase();

green2.SetXDistance(-100.0);

green2.SetYDistance(5.0);

green2\_theta=101.0;

balls\_remaining--;

}

if((green3.GetXDistance()<=1.6\*SIZE\_FACTOR && green3.GetYDistance()<=1.6\*SIZE\_FACTOR) || (green3.GetXDistance()<=1.6\*SIZE\_FACTOR && green3.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (green3.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && green3.GetYDistance()<=1.6\*SIZE\_FACTOR) || (green3.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && green3.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(25);

else Player\_2.update\_score(25);

Green3.Erase();

green3.SetXDistance(-100.0);

green3.SetYDistance(5.0);

green3\_theta=101.0;

balls\_remaining--;

}

if((green4.GetXDistance()<=1.6\*SIZE\_FACTOR && green4.GetYDistance()<=1.6\*SIZE\_FACTOR) || (green4.GetXDistance()<=1.6\*SIZE\_FACTOR && green4.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR) || (green4.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && green4.GetYDistance()<=1.6\*SIZE\_FACTOR) || (green4.GetXDistance()+0.4>=10.5\*SIZE\_FACTOR && green4.GetYDistance()+0.4>=10.5\*SIZE\_FACTOR)){

if(turn%2==0) Player\_1.update\_score(25);

else Player\_2.update\_score(25);

Green4.Erase();

green4.SetXDistance(-100.0);

green4.SetYDistance(5.0);

green4\_theta=101.0;

balls\_remaining--;

}

if(cyan1\_theta!=101.0 && cyan1\_theta!=100.0 && (cyan1.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || cyan1.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

cyan1\_theta=3.141592654-cyan1\_theta;//Checking if the piece hits the wall and re-bounds it. Following are the same.

else if(cyan1\_theta!=101.0 && cyan1\_theta!=100.0 && (cyan1.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || cyan1.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

cyan1\_theta=-cyan1\_theta;

if(cyan2\_theta!=101.0 && cyan2\_theta!=100.0 && (cyan2.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || cyan2.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

cyan2\_theta=3.141592654-cyan2\_theta;

else if(cyan2\_theta!=101.0 && cyan2\_theta!=100.0 && (cyan2.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || cyan2.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

cyan2\_theta=-cyan2\_theta;

if(cyan3\_theta!=101.0 && cyan3\_theta!=100.0 && (cyan3.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || cyan3.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

cyan3\_theta=3.141592654-cyan3\_theta;

else if(cyan3\_theta!=101.0 && cyan3\_theta!=100.0 && (cyan3.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || cyan3.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

cyan3\_theta=-cyan3\_theta;

if(cyan4\_theta!=101.0 && cyan4\_theta!=100.0 && (cyan4.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || cyan4.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

cyan4\_theta=3.141592654-cyan4\_theta;

else if(cyan4\_theta!=101.0 && cyan4\_theta!=100.0 && (cyan4.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || cyan4.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

cyan4\_theta=-cyan4\_theta;

if(blue1\_theta!=101.0 && blue1\_theta!=100.0 && (blue1.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || blue1.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

blue1\_theta=3.141592654-blue1\_theta;

else if(blue1\_theta!=101.0 && blue1\_theta!=100.0 && (blue1.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || blue1.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

blue1\_theta=-blue1\_theta;

if(blue2\_theta!=101.0 && blue2\_theta!=100.0 && (blue2.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || blue2.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

blue2\_theta=3.141592654-blue2\_theta;

else if(blue2\_theta!=101.0 && blue2\_theta!=100.0 && (blue2.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || blue2.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

blue2\_theta=-blue2\_theta;

if(blue3\_theta!=101.0 && blue3\_theta!=100.0 && (blue3.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || blue3.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

blue3\_theta=3.141592654-blue3\_theta;

else if(blue3\_theta!=101.0 && blue3\_theta!=100.0 && (blue3.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || blue3.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

blue3\_theta=-blue3\_theta;

if(blue4\_theta!=101.0 && blue4\_theta!=100.0 && (blue4.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || blue4.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

blue4\_theta=3.141592654-blue4\_theta;

else if(blue4\_theta!=101.0 && blue4\_theta!=100.0 && (blue4.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || blue4.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

blue4\_theta=-blue4\_theta;

if(green1\_theta!=101.0 && green1\_theta!=100.0 && (green1.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || green1.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

green1\_theta=3.141592654-green1\_theta;

else if(green1\_theta!=101.0 && green1\_theta!=100.0 && (green1.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || green1.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

green1\_theta=-green1\_theta;

if(green2\_theta!=101.0 && green2\_theta!=100.0 && (green2.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || green2.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

green2\_theta=3.141592654-green2\_theta;

else if(green2\_theta!=101.0 && green2\_theta!=100.0 && (green2.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || green2.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

green2\_theta=-green2\_theta;

if(green3\_theta!=101.0 && green3\_theta!=100.0 && (green3.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || green3.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

green3\_theta=3.141592654-green3\_theta;

else if(green3\_theta!=101.0 && green3\_theta!=100.0 && (green3.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || green3.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

green3\_theta=-green3\_theta;

if(green4\_theta!=101.0 && green4\_theta!=100.0 && (green4.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || green4.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

green4\_theta=3.141592654-green4\_theta;

else if(green4\_theta!=101.0 && green4\_theta!=100.0 && (green4.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || green4.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

green4\_theta=-green4\_theta;

if(black1\_theta!=101.0 && black1\_theta!=100.0 && (black1.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || black1.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

black1\_theta=3.141592654-black1\_theta;

else if(black1\_theta!=101.0 && black1\_theta!=100.0 && (black1.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || black1.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

black1\_theta=-black1\_theta;

if(black2\_theta!=101.0 && black2\_theta!=100.0 && (black2.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || black2.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

black2\_theta=3.141592654-black2\_theta;

else if(black2\_theta!=101.0 && black2\_theta!=100.0 && (black2.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || black2.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

black2\_theta=-black2\_theta;

if(black3\_theta!=101.0 && black3\_theta!=100.0 && (black3.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || black3.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

black3\_theta=3.141592654-black3\_theta;

else if(black3\_theta!=101.0 && black3\_theta!=100.0 && (black3.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || black3.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

black3\_theta=-black3\_theta;

if(black4\_theta!=101.0 && black4\_theta!=100.0 && (black4.GetXDistance()+0.4>=11.0\*SIZE\_FACTOR || black4.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR))

black4\_theta=3.141592654-black4\_theta;

else if(black4\_theta!=101.0 && black4\_theta!=100.0 && (black4.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || black4.GetYDistance()+0.4>=11.0\*SIZE\_FACTOR))

black4\_theta=-black4\_theta;

//For all possible combinations of collisions of the striker

collision\_striker(cyan1,cyan1\_theta);

collision\_striker(cyan2,cyan2\_theta);

collision\_striker(cyan3,cyan3\_theta);

collision\_striker(cyan4,cyan4\_theta);

collision\_striker(blue1,blue1\_theta);

collision\_striker(blue2,blue2\_theta);

collision\_striker(blue3,blue3\_theta);

collision\_striker(blue4,blue4\_theta);

collision\_striker(black1,black1\_theta);

collision\_striker(black2,black2\_theta);

collision\_striker(black3,black3\_theta);

collision\_striker(black4,black4\_theta);

collision\_striker(green1,green1\_theta);

collision\_striker(green2,green2\_theta);

collision\_striker(green3,green3\_theta);

collision\_striker(green4,green4\_theta);

//For all possible combination of collisions of cyan1

collision\_pieces(cyan1,cyan2,cyan1\_theta,cyan2\_theta);

collision\_pieces(cyan1,cyan3,cyan1\_theta,cyan3\_theta);

collision\_pieces(cyan1,cyan4,cyan1\_theta,cyan4\_theta);

collision\_pieces(cyan1,blue1,cyan1\_theta,blue1\_theta);

collision\_pieces(cyan1,blue2,cyan1\_theta,blue2\_theta);

collision\_pieces(cyan1,blue3,cyan1\_theta,blue3\_theta);

collision\_pieces(cyan1,blue4,cyan1\_theta,blue4\_theta);

collision\_pieces(cyan1,green1,cyan1\_theta,green1\_theta);

collision\_pieces(cyan1,green2,cyan1\_theta,green2\_theta);

collision\_pieces(cyan1,green3,cyan1\_theta,green3\_theta);

collision\_pieces(cyan1,green4,cyan1\_theta,green4\_theta);

collision\_pieces(cyan1,black1,cyan1\_theta,black1\_theta);

collision\_pieces(cyan1,black2,cyan1\_theta,black2\_theta);

collision\_pieces(cyan1,black3,cyan1\_theta,black3\_theta);

collision\_pieces(cyan1,black4,cyan1\_theta,black4\_theta);

//For all remaining possible combination of collisions of cyan2

collision\_pieces(cyan2,cyan3,cyan2\_theta,cyan3\_theta);

collision\_pieces(cyan2,cyan4,cyan2\_theta,cyan4\_theta);

collision\_pieces(cyan2,blue1,cyan2\_theta,blue1\_theta);

collision\_pieces(cyan2,blue2,cyan2\_theta,blue2\_theta);

collision\_pieces(cyan2,blue3,cyan2\_theta,blue3\_theta);

collision\_pieces(cyan2,blue4,cyan2\_theta,blue4\_theta);

collision\_pieces(cyan2,green1,cyan2\_theta,green1\_theta);

collision\_pieces(cyan2,green2,cyan2\_theta,green2\_theta);

collision\_pieces(cyan2,green3,cyan2\_theta,green3\_theta);

collision\_pieces(cyan2,green4,cyan2\_theta,green4\_theta);

collision\_pieces(cyan2,black1,cyan2\_theta,black1\_theta);

collision\_pieces(cyan2,black2,cyan2\_theta,black2\_theta);

collision\_pieces(cyan2,black3,cyan2\_theta,black3\_theta);

collision\_pieces(cyan2,black4,cyan2\_theta,black4\_theta);

//For all remaining possible combination of collisions of cyan3

collision\_pieces(cyan3,cyan4,cyan3\_theta,cyan4\_theta);

collision\_pieces(cyan3,blue1,cyan3\_theta,blue1\_theta);

collision\_pieces(cyan3,blue2,cyan3\_theta,blue2\_theta);

collision\_pieces(cyan3,blue3,cyan3\_theta,blue3\_theta);

collision\_pieces(cyan3,blue4,cyan3\_theta,blue4\_theta);

collision\_pieces(cyan3,green1,cyan3\_theta,green1\_theta);

collision\_pieces(cyan3,green2,cyan3\_theta,green2\_theta);

collision\_pieces(cyan3,green3,cyan3\_theta,green3\_theta);

collision\_pieces(cyan3,green4,cyan3\_theta,green4\_theta);

collision\_pieces(cyan3,black1,cyan3\_theta,black1\_theta);

collision\_pieces(cyan3,black2,cyan3\_theta,black2\_theta);

collision\_pieces(cyan3,black3,cyan3\_theta,black3\_theta);

collision\_pieces(cyan3,black4,cyan3\_theta,black4\_theta);

//For all remaining possible combination of collisions of cyan4

collision\_pieces(cyan4,blue1,cyan4\_theta,blue1\_theta);

collision\_pieces(cyan4,blue2,cyan4\_theta,blue2\_theta);

collision\_pieces(cyan4,blue3,cyan4\_theta,blue3\_theta);

collision\_pieces(cyan4,blue4,cyan4\_theta,blue4\_theta);

collision\_pieces(cyan4,green1,cyan4\_theta,green1\_theta);

collision\_pieces(cyan4,green2,cyan4\_theta,green2\_theta);

collision\_pieces(cyan4,green3,cyan4\_theta,green3\_theta);

collision\_pieces(cyan4,green4,cyan4\_theta,green4\_theta);

collision\_pieces(cyan4,black1,cyan4\_theta,black1\_theta);

collision\_pieces(cyan4,black2,cyan4\_theta,black2\_theta);

collision\_pieces(cyan4,black3,cyan4\_theta,black3\_theta);

collision\_pieces(cyan4,black4,cyan4\_theta,black4\_theta);

//For all remaining possible combination of collisions of blue1

collision\_pieces(blue1,blue2,blue1\_theta,blue2\_theta);

collision\_pieces(blue1,blue3,blue1\_theta,blue3\_theta);

collision\_pieces(blue1,blue4,blue1\_theta,blue4\_theta);

collision\_pieces(blue1,green1,blue1\_theta,green1\_theta);

collision\_pieces(blue1,green2,blue1\_theta,green2\_theta);

collision\_pieces(blue1,green3,blue1\_theta,green3\_theta);

collision\_pieces(blue1,green4,blue1\_theta,green4\_theta);

collision\_pieces(blue1,black1,blue1\_theta,black1\_theta);

collision\_pieces(blue1,black2,blue1\_theta,black2\_theta);

collision\_pieces(blue1,black3,blue1\_theta,black3\_theta);

collision\_pieces(blue1,black4,blue1\_theta,black4\_theta);

//For all remaining possible combination of collisions of blue2

collision\_pieces(blue2,blue3,blue2\_theta,blue3\_theta);

collision\_pieces(blue2,blue4,blue2\_theta,blue4\_theta);

collision\_pieces(blue2,green1,blue2\_theta,green1\_theta);

collision\_pieces(blue2,green2,blue2\_theta,green2\_theta);

collision\_pieces(blue2,green3,blue2\_theta,green3\_theta);

collision\_pieces(blue2,green4,blue2\_theta,green4\_theta);

collision\_pieces(blue2,black1,blue2\_theta,black1\_theta);

collision\_pieces(blue2,black2,blue2\_theta,black2\_theta);

collision\_pieces(blue2,black3,blue2\_theta,black3\_theta);

collision\_pieces(blue2,black4,blue2\_theta,black4\_theta);

//For all remaining possible combination of collisions of blue3

collision\_pieces(blue3,blue4,blue3\_theta,blue4\_theta);

collision\_pieces(blue3,green1,blue3\_theta,green1\_theta);

collision\_pieces(blue3,green2,blue3\_theta,green2\_theta);

collision\_pieces(blue3,green3,blue3\_theta,green3\_theta);

collision\_pieces(blue3,green4,blue3\_theta,green4\_theta);

collision\_pieces(blue3,black1,blue3\_theta,black1\_theta);

collision\_pieces(blue3,black2,blue3\_theta,black2\_theta);

collision\_pieces(blue3,black3,blue3\_theta,black3\_theta);

collision\_pieces(blue3,black4,blue3\_theta,black4\_theta);

//For all remaining possible combination of collisions of blue4

collision\_pieces(blue4,green1,blue4\_theta,green1\_theta);

collision\_pieces(blue4,green2,blue4\_theta,green2\_theta);

collision\_pieces(blue4,green3,blue4\_theta,green3\_theta);

collision\_pieces(blue4,green4,blue4\_theta,green4\_theta);

collision\_pieces(blue4,black1,blue4\_theta,black1\_theta);

collision\_pieces(blue4,black2,blue4\_theta,black2\_theta);

collision\_pieces(blue4,black3,blue4\_theta,black3\_theta);

collision\_pieces(blue4,black4,blue4\_theta,black4\_theta);

//For all remaining possible combination of collisions of green1

collision\_pieces(green1,green2,green1\_theta,green2\_theta);

collision\_pieces(green1,green3,green1\_theta,green3\_theta);

collision\_pieces(green1,green4,green1\_theta,green4\_theta);

collision\_pieces(green1,black1,green1\_theta,black1\_theta);

collision\_pieces(green1,black2,green1\_theta,black2\_theta);

collision\_pieces(green1,black3,green1\_theta,black3\_theta);

collision\_pieces(green1,black4,green1\_theta,black4\_theta);

//For all remaining possible combination of collisions of green2

collision\_pieces(green2,green3,green2\_theta,green3\_theta);

collision\_pieces(green2,green4,green2\_theta,green4\_theta);

collision\_pieces(green2,black1,green2\_theta,black1\_theta);

collision\_pieces(green2,black2,green2\_theta,black2\_theta);

collision\_pieces(green2,black3,green2\_theta,black3\_theta);

collision\_pieces(green2,black4,green2\_theta,black4\_theta);

//For all remaining possible combination of collisions of green3

collision\_pieces(green3,green4,green3\_theta,green4\_theta);

collision\_pieces(green3,black1,green3\_theta,black1\_theta);

collision\_pieces(green3,black2,green3\_theta,black2\_theta);

collision\_pieces(green3,black3,green3\_theta,black3\_theta);

collision\_pieces(green3,black4,green3\_theta,black4\_theta);

//For all remaining possible combination of collisions of green4

collision\_pieces(green4,black1,green4\_theta,black1\_theta);

collision\_pieces(green4,black2,green4\_theta,black2\_theta);

collision\_pieces(green4,black3,green4\_theta,black3\_theta);

collision\_pieces(green4,black4,green4\_theta,black4\_theta);

//For all remaining possible combination of collisions of black1

collision\_pieces(black1,black2,black1\_theta,black2\_theta);

collision\_pieces(black1,black3,black1\_theta,black3\_theta);

collision\_pieces(black1,black4,black1\_theta,black4\_theta);

//For all remaining possible combination of collisions of black2

collision\_pieces(black2,black3,black2\_theta,black3\_theta);

collision\_pieces(black2,black4,black2\_theta,black4\_theta);

//For all remaining possible combination of collisions of black3

collision\_pieces(black3,black4,black3\_theta,black4\_theta);

if(cyan1\_theta!=100.0 && cyan1\_theta!=101.0){//Checking if the piece has already not been potted or is not at rest. Following are the same.

Cyan1.Erase();

cyan1.SetXDistance(cyan1.GetXDistance()+0.005\*power\*cos(cyan1\_theta));

cyan1.SetYDistance(cyan1.GetYDistance()+0.005\*power\*sin(cyan1\_theta));

Cyan1.SetPosition(cyan1);

Cyan1.Draw();

}

if(cyan2\_theta!=100.0 && cyan2\_theta!=101.0){

Cyan2.Erase();

cyan2.SetXDistance(cyan2.GetXDistance()+0.005\*power\*cos(cyan2\_theta));

cyan2.SetYDistance(cyan2.GetYDistance()+0.005\*power\*sin(cyan2\_theta));

Cyan2.SetPosition(cyan2);

Cyan2.Draw();

}

if(cyan3\_theta!=100.0 && cyan3\_theta!=101.0){

Cyan3.Erase();

cyan3.SetXDistance(cyan3.GetXDistance()+0.005\*power\*cos(cyan3\_theta));

cyan3.SetYDistance(cyan3.GetYDistance()+0.005\*power\*sin(cyan3\_theta));

Cyan3.SetPosition(cyan3);

Cyan3.Draw();

}

if(cyan4\_theta!=100.0 && cyan4\_theta!=101.0){

Cyan4.Erase();

cyan4.SetXDistance(cyan4.GetXDistance()+0.005\*power\*cos(cyan4\_theta));

cyan4.SetYDistance(cyan4.GetYDistance()+0.005\*power\*sin(cyan4\_theta));

Cyan4.SetPosition(cyan4);

Cyan4.Draw();

}

if(blue1\_theta!=100.0 && blue1\_theta!=101.0){

Blue1.Erase();

blue1.SetXDistance(blue1.GetXDistance()+0.005\*power\*cos(blue1\_theta));

blue1.SetYDistance(blue1.GetYDistance()+0.005\*power\*sin(blue1\_theta));

Blue1.SetPosition(blue1);

Blue1.Draw();

}

if(blue2\_theta!=100.0 && blue2\_theta!=101.0){

Blue2.Erase();

blue2.SetXDistance(blue2.GetXDistance()+0.005\*power\*cos(blue2\_theta));

blue2.SetYDistance(blue2.GetYDistance()+0.005\*power\*sin(blue2\_theta));

Blue2.SetPosition(blue2);

Blue2.Draw();

}

if(blue3\_theta!=100.0 && blue3\_theta!=101.0){

Blue3.Erase();

blue3.SetXDistance(blue3.GetXDistance()+0.005\*power\*cos(blue3\_theta));

blue3.SetYDistance(blue3.GetYDistance()+0.005\*power\*sin(blue3\_theta));

Blue3.SetPosition(blue3);

Blue3.Draw();

}

if(blue4\_theta!=100.0 && blue4\_theta!=101.0){

Blue4.Erase();

blue4.SetXDistance(blue4.GetXDistance()+0.005\*power\*cos(blue4\_theta));

blue4.SetYDistance(blue4.GetYDistance()+0.005\*power\*sin(blue4\_theta));

Blue4.SetPosition(blue4);

Blue4.Draw();

}

if(black1\_theta!=100.0 && black1\_theta!=101.0){

Black1.Erase();

black1.SetXDistance(black1.GetXDistance()+0.005\*power\*cos(black1\_theta));

black1.SetYDistance(black1.GetYDistance()+0.005\*power\*sin(black1\_theta));

Black1.SetPosition(black1);

Black1.Draw();

}

if(black2\_theta!=100.0 && black2\_theta!=101.0){

Black2.Erase();

black2.SetXDistance(black2.GetXDistance()+0.005\*power\*cos(black2\_theta));

black2.SetYDistance(black2.GetYDistance()+0.005\*power\*sin(black2\_theta));

Black2.SetPosition(black2);

Black2.Draw();

}

if(black3\_theta!=100.0 && black3\_theta!=101.0){

Black3.Erase();

black3.SetXDistance(black3.GetXDistance()+0.005\*power\*cos(black3\_theta));

black3.SetYDistance(black3.GetYDistance()+0.005\*power\*sin(black3\_theta));

Black3.SetPosition(black3);

Black3.Draw();

}

if(black4\_theta!=100.0 && black4\_theta!=101.0){

Black4.Erase();

black4.SetXDistance(black4.GetXDistance()+0.005\*power\*cos(black4\_theta));

black4.SetYDistance(black4.GetYDistance()+0.005\*power\*sin(black4\_theta));

Black4.SetPosition(black4);

Black4.Draw();

}

if(green1\_theta!=100.0 && green1\_theta!=101.0){

Green1.Erase();

green1.SetXDistance(green1.GetXDistance()+0.005\*power\*cos(green1\_theta));

green1.SetYDistance(green1.GetYDistance()+0.005\*power\*sin(green1\_theta));

Green1.SetPosition(green1);

Green1.Draw();

}

if(green2\_theta!=100.0 && green2\_theta!=101.0){

Green2.Erase();

green2.SetXDistance(green2.GetXDistance()+0.005\*power\*cos(green2\_theta));

green2.SetYDistance(green2.GetYDistance()+0.005\*power\*sin(green2\_theta));

Green2.SetPosition(green2);

Green2.Draw();

}

if(green3\_theta!=100.0 && green3\_theta!=101.0){

Green3.Erase();

green3.SetXDistance(green3.GetXDistance()+0.005\*power\*cos(green3\_theta));

green3.SetYDistance(green3.GetYDistance()+0.005\*power\*sin(green3\_theta));

Green3.SetPosition(green3);

Green3.Draw();

}

if(green4\_theta!=100.0 && green4\_theta!=101.0){

Green4.Erase();

green4.SetXDistance(green4.GetXDistance()+0.005\*power\*cos(green4\_theta));

green4.SetYDistance(green4.GetYDistance()+0.005\*power\*sin(green4\_theta));

Green4.SetPosition(green4);

Green4.Draw();

}

if(foul!=1){//Checking if the Striker is not potted and then Striker is drawn

if(striker.GetXDistance()+0.68>=11.0\*SIZE\_FACTOR || striker.GetXDistance()-0.04<=1.0\*SIZE\_FACTOR)//Condition for re-bounding from walls

striker\_theta=3.141592654-striker\_theta;

if(striker.GetYDistance()-0.04<=1.0\*SIZE\_FACTOR || striker.GetYDistance()+0.68>=11.0\*SIZE\_FACTOR)//Same

striker\_theta=-striker\_theta;

if((striker.GetXDistance()<=1.6\*SIZE\_FACTOR && striker.GetYDistance()<=1.6\*SIZE\_FACTOR) || (striker.GetXDistance()<=1.6\*SIZE\_FACTOR && striker.GetYDistance()+0.6>=10.5\*SIZE\_FACTOR) || (striker.GetXDistance()+0.6>=10.5\*SIZE\_FACTOR && striker.GetYDistance()<=1.6\*SIZE\_FACTOR) || (striker.GetXDistance()+0.6>=10.5\*SIZE\_FACTOR && striker.GetYDistance()+0.6>=10.5\*SIZE\_FACTOR)){//Checking if the striker hits the pocket

foul=1;

Striker.Erase();

if(turn%2==0) Player\_1.update\_score(-10);

else Player\_2.update\_score(-10);

continue;

}

Striker.Erase();

striker.SetXDistance(striker.GetXDistance()+0.005\*power\*cos(striker\_theta));

striker.SetYDistance(striker.GetYDistance()+0.005\*power\*sin(striker\_theta));

Striker.SetPosition(striker);

Striker.Draw();

}

display\_pieces();

if(50\*i/power>5000) break;//Condition for stopping the moving pieces

usleep(50\*i/power);//The main equation of the program which enables the pieces to appear as if they're actually moving

i+=1.0;

}

if(foul!=0){//Checking if a foul takes place and displays the foul message

Display\_Foul.Open();

assert(Display\_Foul.GetStatus()==WindowOpen);

Display\_Foul.RenderRectangle(Position(0.0,0.0),Position(5.0,2.0),Black);

Display\_Foul.RenderText(Position(0.0,0.0),Position(5.0,2.0),"FOUL! -10 points!",Cyan,Black);

sleep(2);

Display\_Foul.Close();

foul=0;

}

if(cyan1\_theta!=101.0)cyan1\_theta=100.0;//Assigning defaults

if(cyan2\_theta!=101.0)cyan2\_theta=100.0;

if(cyan3\_theta!=101.0)cyan3\_theta=100.0;

if(cyan4\_theta!=101.0)cyan4\_theta=100.0;

if(blue1\_theta!=101.0)blue1\_theta=100.0;

if(blue2\_theta!=101.0)blue2\_theta=100.0;

if(blue3\_theta!=101.0)blue3\_theta=100.0;

if(blue4\_theta!=101.0)blue4\_theta=100.0;

if(black1\_theta!=101.0)black1\_theta=100.0;

if(black2\_theta!=101.0)black2\_theta=100.0;

if(black3\_theta!=101.0)black3\_theta=100.0;

if(black4\_theta!=101.0)black4\_theta=100.0;

if(green1\_theta!=101.0)green1\_theta=100.0;

if(green2\_theta!=101.0)green2\_theta=100.0;

if(green3\_theta!=101.0)green3\_theta=100.0;

if(green4\_theta!=101.0)green4\_theta=100.0;

sleep(2);

repair\_carrom\_background();

display\_pieces();

}

//Function for the collisions of pieces taking place during the shooting process

void collision\_pieces(Position a, Position b, float &theta1, float &theta2){

if((theta1==100.0 && theta2==100.0) || theta1==101.0 || theta2==101.0) return;//Checking if both are at rest or if either one is already potted

else if((a.GetXDistance()>=b.GetXDistance() && a.GetXDistance()<=b.GetXDistance()+0.4 && a.GetYDistance()>=b.GetYDistance() && a.GetYDistance()<=b.GetYDistance()+0.4) || (a.GetXDistance()+0.4>=b.GetXDistance() && a.GetXDistance()+0.4<=b.GetXDistance()+0.4 && a.GetYDistance()>=b.GetYDistance() && a.GetYDistance()<=b.GetYDistance()+0.4) || (a.GetXDistance()+0.4>=b.GetXDistance() && a.GetXDistance()+0.4<=b.GetXDistance()+0.4 && a.GetYDistance()+0.4>=b.GetYDistance() && a.GetYDistance()+0.4<=b.GetYDistance()+0.4) || (a.GetXDistance()>=b.GetXDistance() && a.GetXDistance()<=b.GetXDistance()+0.4 && a.GetYDistance()+0.4>=b.GetYDistance() && a.GetYDistance()+0.4<=b.GetYDistance()+0.4)){//Checking if collision takes place

a.SetXDistance(a.GetXDistance()+0.2);//Setting the position as the center of the pieces

a.SetYDistance(a.GetYDistance()+0.2);

b.SetXDistance(b.GetXDistance()+0.2);

b.SetYDistance(b.GetYDistance()+0.2);

if(theta1==100.0)theta1=calculate\_theta(b,a);//Calculating it's theta of motion

else if(theta2==100.0)theta2=calculate\_theta(a,b);//Calculating it's theta of motion

else{

float temp=theta1;

theta1=atan2(sin(theta2),cos(temp));//Deflecting both the moving pieces according to the laws of physics

theta2=atan2(sin(temp),cos(theta2));

}

}

}

//Function for the collisions of striker taking place during the shooting process

void collision\_striker(Position a, float &theta1){

if((striker.GetXDistance()>=a.GetXDistance() && striker.GetXDistance()<=a.GetXDistance()+0.4 && striker.GetYDistance()>=a.GetYDistance() && striker.GetYDistance()<=a.GetYDistance()+0.4) || (striker.GetXDistance()+0.68>=a.GetXDistance() && striker.GetXDistance()+0.68<=a.GetXDistance()+0.4 && striker.GetYDistance()>=a.GetYDistance() && striker.GetYDistance()<=a.GetYDistance()+0.4) || (striker.GetXDistance()+0.68>=a.GetXDistance() && striker.GetXDistance()+0.68<=a.GetXDistance()+0.4 && striker.GetYDistance()+0.68>=a.GetYDistance() && striker.GetYDistance()+0.68<=a.GetYDistance()+0.4) || (striker.GetXDistance()>=a.GetXDistance() && striker.GetXDistance()<=a.GetXDistance()+0.4 && striker.GetYDistance()+0.68>=a.GetYDistance() && striker.GetYDistance()+0.68<=a.GetYDistance()+0.4)){//Checking if collision takes place

if(theta1==101.0) return;//Checking if the piece is already potted

Position temp1;

temp1.SetXDistance(striker.GetXDistance()+0.34);//Setting the position as the center of the pieces

temp1.SetYDistance(striker.GetYDistance()+0.34);

a.SetXDistance(a.GetXDistance()+0.2);

a.SetYDistance(a.GetYDistance()+0.2);

if(theta1==100.0)theta1=calculate\_theta(temp1,a);//Calculating theta of the piece

else{

float temp=theta1;

theta1=atan2(sin(striker\_theta),cos(temp));//Deflecting both the moving pieces according to the laws of physics

striker\_theta=atan2(sin(temp),cos(striker\_theta));

}

}

}

#endif

* **#include"score\_screen.h"**

Custom header file made by the team which contains the functions to display the score screen and the mouse click function.

#ifndef SCORE\_SCREEN\_H\_INCLUDED

#define SCORE\_SCREEN\_H\_INCLUDED

SimpleWindow Score\_Screen("Score:",11.0,11.0,Position(5.0,2.0));//Defining the simple window

void Open\_Display\_Score\_Screen();//Function to display the score after the game finishes

int Score\_ScreenMouseClick(const Position &p);//Function to control mouse clicks in the Score\_Screen window

//Function to control mouse clicks in the Score\_Screen window

int Score\_ScreenMouseClick(const Position &p){

if(p.GetXDistance()<=4.0 && p.GetXDistance()>=0.0 && p.GetYDistance()<=11.0 && p.GetYDistance()>=10.0){//Checking if the click is in the GO TO MAIN MENU button and acting accordingly

Score\_Screen.Close();

Open\_Welcome\_Screen();

}

if(p.GetXDistance()<=9.0 && p.GetXDistance()>=4.5 && p.GetYDistance()<=11.0 && p.GetYDistance()>=10.0){//Checking if the click is on the PLAY AGAIN button and acting accordingly

Score\_Screen.Close();

Player\_1.update\_score(0);

Player\_2.update\_score(0);

Open\_CARROM\_screen();

}

if(p.GetXDistance()<=11.0 && p.GetXDistance()>=9.0 && p.GetYDistance()<=11.0 && p.GetYDistance()>=10.0){//Checking if the click is on the EXIT button and acting accordingly

Score\_Screen.Close();

exit(0);

}

}

//Function to display the score after the game finishes

void Open\_Display\_Score\_Screen(){

int temp;

CARROM.Close();

Score\_Screen.Open();//Opening window to display the final score

assert(Score\_Screen.GetStatus()==WindowOpen);//Checking if window opened

Score\_Screen.RenderRectangle(Position(0.0,0.0),Position(11.0,11.0),Black);

Score\_Screen.RenderText(Position(2.0,0.0),Position(4.0,2.0),Player\_1.give\_name(),Red,Cyan);

Score\_Screen.RenderText(Position(2.0,2.0),Position(4.0,4.0),Player\_2.give\_name(),Red,Cyan);

Score\_Screen.RenderText(Position(5.0,0.0),Position(6.0,2.0),":",Red,Cyan);

Score\_Screen.RenderText(Position(5.0,2.0),Position(6.0,4.0),":",Red,Cyan);

Score\_Screen.RenderText(Position(7.0,0.0),Position(9.0,2.0),Player\_1.give\_score(),Red,Cyan);

Score\_Screen.RenderText(Position(7.0,2.0),Position(9.0,4.0),Player\_2.give\_score(),Red,Cyan);

temp=Player\_1.compare\_score(Player\_2.give\_score());

if(temp==1){//That is, if Player\_1 has a higher score

Score\_Screen.RenderText(Position(3.0,4.0),Position(5.5,6.0),Player\_1.give\_name(),Red,Cyan);

Score\_Screen.RenderText(Position(5.5,4.0),Position(11.0,6.0),"Wins!",Red,Cyan);

Score\_Screen.RenderText(Position(3.0,6.0),Position(8.0,8.0),"CONGRATULATIONS! Well Played!",Red,Cyan);

Score\_Screen.RenderText(Position(0.0,8.0),Position(5.5,10.0),"Better luck next time",Red,Cyan);

Score\_Screen.RenderText(Position(5.5,8.0),Position(11.0,10.0),Player\_2.give\_name(),Red,Cyan);

}

else if(temp==-1){//That is, if Player\_2 has a higher score

Score\_Screen.RenderText(Position(2.0,4.0),Position(5.5,6.0),Player\_2.give\_name(),Red,Cyan);

Score\_Screen.RenderText(Position(5.5,4.0),Position(11.0,6.0),"Wins!",Red,Cyan);

Score\_Screen.RenderText(Position(3.0,6.0),Position(8.0,8.0),"CONGRATULATIONS! Well Played!",Red,Cyan);

Score\_Screen.RenderText(Position(0.0,8.0),Position(5.5,10.0),"Better luck next time",Red,Cyan);

Score\_Screen.RenderText(Position(5.5,8.0),Position(11.0,10.0),Player\_1.give\_name(),Red,Cyan);

}

else{//In case of a tie

Score\_Screen.RenderText(Position(3.0,4.0),Position(5.5,7.0),"It's a tie!",Red,Cyan);

Score\_Screen.RenderText(Position(5.5,7.0),Position(8.0,10.0),"Well played, both of you!",Red,Cyan);

}

Score\_Screen.RenderText(Position(0.0,10.0),Position(4.0,11.0),"GO TO MAIN MENU",Red,Cyan);

Score\_Screen.RenderText(Position(4.5,10.0),Position(9.0,11.0),"PLAY AGAIN",Red,Cyan);

Score\_Screen.RenderText(Position(9.0,10.0),Position(11.0,11.0),"EXIT",Red,Cyan);

Score\_Screen.SetMouseClickCallback(Score\_ScreenMouseClick);

}

#endif