SRS Document
Lab Batch 37
CS101 Course Project
FOOSBALL

Team Members:

AI Team:

Abhishek khanna*	10D070045 (Team Leader)
Hardik Agarwal	10D070042
Rohan pillai	10D070053
Karthik sharma	10D070014

Collisions Team:

Pavan jatale	10D070004
Prakhar Khandelwal*	10D070040 (Team Leader)
Shanay shah	10D070007
Virat shejwalkar	10D070021

Graphics and UI Team:

Ashwin Kachhara**	10D070048 (Team Leader and Batch Coordinator)
Himanshu soni	10D070050
Mrinal Joshi	10D070028
Prashant Borkar	10D070025
Mayank Maloo	10D070063

Introduction:

Foosball is basically table football. There's a Foosball table with two sets of players controlled by rods.There is a goal at each end and a place to serve on each side in the middle. Each player/team uses their handles to move their men to strike the ball into the opposing teams goal.

In our project , we captured the same essence of the game . Here we have two teams , Team Blue and Team Red, with 7 players each.

- 3 in attack.
- 3 in Mid-Field.
- 1 Goal-keeper.

Team Blue is A.I . Team Red is User Controlled. You can control the movement of all the players by clicking on left and right Bitmap images.There's a stop buttuon to stop the movement. Its a 5 Goals Match.

The game is fast and funny and everyone cheers and laughs.Don't be surprised if you get so excited you sweat a little bit from the excitement!!!

Enjoy!

Functions Created:

MovePlayer1 & MoveaiPlayer:

void MoveaiPlayer(Position &p,float pposxu,float pposy);
void MovePlayer1(Position &p,float pposx,float pposy)

This function moves the player from position p and increments the x coordinate by amount pposx on each invocation. the player moves only in one direction i.e x direction. The y coordinate is fixed.Class used to create the shape is RecctangleShape which is in rect.h library.

How it functions:

We have used the usleep function of time.h library to provide delay between drawing and erasing the object.usleep takes one parameter i.e the time to pause the run in microseconds.
All Shape classes in EzWindows by default have a border which remains even if you erase the object.The same problem as in MoveBall function is present here. Hence we have used the ClearBorder() member function after each Draw() function to counter this problem.
With some delay time the position is incremented, initial player is erased and redrawn at new position. Executed in a loop,this leads to continous movement of the player. The corresponding usleep() function is in the RunGame() function

Limitation:
the functions create multiple object of class rectangleshape. As per the function described above,it erases the player and redraws it at the new increamented position. The erase function changes the colour of the player to the colour of the background.we also repeatedly use fieldform() function.

MClick:

It is a mouse callback click function for playing in the foosballWindow.Function prototype declaration is as follows:

int MClick(const Position &p1)

It detects clicks made on the bitmaps Left,Right and Stop.If Left is clicked it sets the user player X position increment to a negative value.In case of Right sets the X increment to a positive value.In case of Stop sets the increment to zero.

Loop:

It is a timer callback function for checking the collision conditions and AI execution in the foosballWindow. Function prototype declaration is as follows:

int Loop()

1.It directs movement of players and ball for various cases of collision like

-With the wall,

-With player or the ball as the case may be.

2.It executes the AI conditions.
3.It calculates time elapsed since the start of the game and displays it at the appropriate position

W0MouseC1ick1:

It is the mouse click function for the W0 window which leads to W1 window.Its declaration is: int W0MouseClick1(const Position &p1)
where p1 is the position where the mouse was clicked.
It closes W0,opens W1, loads the Bitmaps of the W1 window(Main Menu) namely Play, Instructions and Exit, sets their positions and draws them using the BitMap::Draw() function.
It also sets the Mouseclick callback of W1 window to W1MouseClick1.

W1MouseClick1:

Declaration: int W1MouseClick1(const Position &px)
It is the mouse click function for the W1 window which leads to

- W4 if Instructions is clicked
- foosballWindow if Play is clicked
- Closes W1 if Exit is clicked

It also sets the various display attributes in the windows foosballWindow and W4
incase of foosballWindow it invokes function to generate field,sets the callbacks (timer and mouse click which are essential for playing).

Incase of W4, it draws the necessary bitmaps and sets the callback function as W4MouseClick1 which closes W4, and reopens W1(Main Menu) and loads the necessary bitmaps.

W4MouseClick1:

It is the mouse click function for the W4 window.
It closes W4, and reopens W1(Main Menu) and loads the necessary bitmaps.

MoveBall1 function:

This is a modification of MoveBall function which is made to erase the ball if it enter the goal(it was looking

unrealistic if ball was remaining in the goal).It removes the usleep statements and does not increment the posx and posy variables and in the end it does not redraw the ball.It is called only if goal occurs.

Goaldisplay function:
This function will be invoked when a goal is occured and it basically prints the goal at the centre. It is an int function as it returns -1 if the goal count exceeds 3 because we have kept max goals to be 4. It takes team as the parameter and displays goal and increments the goal of the team which is passed in the parameter.Then as the goal increments, we want to print in the score box, so we converted goal into string variable by sstream library. If goal of any team is greater than 3, then a text “Match ends” is printed at the centre of the screen and text “You win” and “You lose” are printed by checking which team (user or AI) has scored more goals. Then Final score is printed at the centre of screen.

Rungame function:
In this function, ball is drawn by function Circleshape with the parameters “window, position, colour and diameter of the ball”. Then to move players Moveplayer1 and Moveaiplayer are called giving the parameters as positon of the player and its increments in x and y direction. Then positon of the ball is incremented and usleep is given to make the ball visible for that much time at that particular position. Then Erase functions(R.Erase and C.Erase) are called to erase the players and ball from their previous positions. Then fieldform is again called to reform the field as it was necessary for the smooth functioning of the game. Then the ball is again drawn at the incremented position.

The following are not functions but work as a block, so their description is given here

Collision of ball with player:
For this purpose we have used **if** statements.For each player the **if** conditions are written separately. Each player is divided into 3 parts. Now if the ball hits player's specific part the ball is deflected in a specific direction by changing the value of position increment variable(posx & posy).
e.g. If the ball hits player's right part the ball is deflected to it's right .
If the ball hits player's middle part the ball is deflected slightly from it's straight path.
If the ball hits player's left part the ball is deflected to it's left.

Collision of ball with wall:

For this purpose again we have used **if** statements. For two vertical walls the conditions are written separately & for horizontal it is differently written.
e.g. If ball collides any vertical wall its posx is
reversed. Similarly for the horizontal wall, posy is reversed.
Also if ball goes in the goal, there is no collision and the moveball1 function as mentioned above is invoked, which erases the ball ,then "goal" is displayed and the ball's X-coordinate is randomly set by using rand() function & y-coordinate is
set to 10.

Collision of player with wall:
The condition is checked for four corner players, as if they collide with the wall and if we reverse their direction, it will automatically be implemented for all the players. So, we make the sign of pposx and posxu (the players' increment variables) negative of the previous variable.

6. Acceptance criteria

Our whole project is divided into three parts:-

1.Graphical User Interface team.
2.Ball collision with walls and players and merging all teams' code to produce a compilable program.
3.A.I. Team defining computer's movements.

Graphical User Interface team.

Desired status at the end of project

1. A. window consisting of a field, scorecard, timer.
2. Field will have green colour and all appropriate markings like goal, centre line, D . Goal of a team will be of same colour as the players of that team.
3. Players(7 each team) will be assigned their appropriate positions (formation of 3-3-1) with respective team colours.
4. Movement of ball according to incrementation of xcoordinate and ycoordinate.
5. Registering callback from user.

Completion status till date

1.A. window consisting of a field.
2.Field has green colour and all appropriate markings like goal, centre line, D .
3.Movement of ball according to incrementation of xcoordinate and ycoordinate has been done.
4.Players are assigned their appropriate positions.
5.Input from user is taken through mouse click as input through keyboard wasn't working properly.

Ball collision with walls and players and merging all teams' code to produce a compilable program.

Desired status at the end of project

1. Ball colliding smoothly with walls following angle of reflection laws.
2.If ball enters goal, it should display goal and ball should reset at the middle position.
3. Ball colliding with players and getting deflected according to the part of player it hits the ball.
If ball hits leftmost part in figure it will be deflected 45 degrees towards the direction shown in figure, similarly for the right part. If ball hits central part, it will be deflected back. These all are valid if ball hits the face of player facing the goal(in which he has to kick ball). If ball hits other face, ball's coordinates will be changed so that it comes on the face facing the goal and above function will be executed.
If ball hits vertical faces,it will be deflected straight
5. Merging every code done by each team suitably.

Completion status till date

1. Ball colliding smoothly with walls following angle of reflection laws.
2. If ball enters goal, it displays goal and ball should reset at the middle position with x coordinate is set randomly.
3. The collsion of the ball with the player has been completed with slight varuiation from the desired part as ball rebounding straight has been not included as if the ball is set straight at the corner position, no player is able to reach to the ball, instead the ball is slightly deviated from the straight position.
4. The collision of the players with the walls is working except the part that players are somewhat going out of the field.
5. Merging and integration has been done perfectly.

Description of data

From the point of view of variables used, the program has 2 main entities

1. the players

2. the ball

There are other variables which are used to enhance the features of players and ball

*PLAYERS *

There are 7 players per teamwith a 3-3-1(goalkeeper) formation.
Each player is stored as a rectangle with some initialized centre position and colour.

```
Position q1(8.0,2.4);
Position q2(4.0,6.0);Position q3(8.0,6.0);Position q4(12.0,6.0);
Position q5(4.0,9.0);Position q6(8.0,9.0);Position q7(12.0,9.0);
Position q8(4.0,11.0);Position q9(8.0,11.0);Position q10(12.0,11.0);
Position q11(4.0,14.0); Position q12(8.0,14.0); Position q13(12.0,14.0);
Position q14(8.0,17.6);
//q1 to q14 are the player positions initialisation
```

MOVEMENTS**:** The movement of players is restricted to left /right. The computer moves its players by incrementing their position by suitable incrementors and giving a time-delay after each incrementation. The increment in position is fixed for the players as is the time-delay.

BALL

The most important entity of the game which the players have to hit inside the opponents goal.

MOVEMENTS: The movement of the ball is on same similar lines as that of players but a bit complicated. The incrementors are not constant as they were for the players.
Initialization of the x coordinate of ball is randomized after every goal.the y coordinate is always set to the centre of the field i.e 10

Position p=Position(7.0,10.0);//sets the initial position of ball
OTHER VARIABLES

scoreA and scoreB, these two variables are used as counters for goals of both the teams. Although they are integers, they are initially stored as strings and later on converted to numerical form.

Timeval: functioning is similar to scoreA and scoreB but this variable stores time in seconds.

INCREMENTS: 1.posx and posy – these are x and y coordinate incrementors for the ball
2.pposx and pposy -these are x and y coordinate incrementors for the AI players.
3.pposxu and pposyu -same as pposx and pposy, but for user players.

GoalR and goalB,they store number of goals scored by the user and computer respectively in the form of integers. The process of increasing goal count is done through them.they are initialized to zero.

Tstart : it stores the number of milliseconds elapsed from a prefixed time till the loading of the foosball window.
Tnow : it stores the number of milliseconds elapsed from a prefixed time till the point it is called.
Telapsed: it stores the number of milliseconds from the time foosball window loads to the point where tnow is called.this is the variable which is used for displaying time on screen.the three timer variables are of the type long int.

float pposxu=0.0,pposx=0.2,pposy=0.0;//postion incrementers are initialised(for movement)

User Interface requirements

Method of playing

The user plays this game using mouse. Below the timer there are three rectangles provided.
1.The left-most rectangle which has a 'left arrow ' sign when clicked, moves the user players leftwards.
2.The rightmost rectangle which has a 'right arrow ' sign gives the user players a speed towards right when clicked
3.When the user clicks on the central rectangle(one with “STOP” written on it) his players stop.
Thus with these three mouse-driven operations, the user can conveniently control his players and position them properly to make glorious saves, retain possession of ball in the mid-field or hit supperb winners.

Further details

The ball shoots off in three different directions(left, almost straight and right) depending upon which of the three parts of the player (left, middle and right) it collides with. The new dierction attained by the ball is independent of

the previous direction of the ball. It only depends on the part of the player it collides with and nothing else.
In a particular row any position can be attained only by one player. Thus the user must carefully make a good guess as to which is the player who will be able to reach out to the ball.

Timer Callback:
The SimpleWindow class supports a timer mechanism. We use this to ivoke the Loop() function every 5 milliseconds

AI Part:

1.)INTRODUCTION:
The theme of A.I. is that the computer players are moved in such a way that they reach such a place where the ball is **expected** to come.
2.)DETAILS:
The field is divided into 7 regions corresponding to each player.
When the ball is moving towards opposing , i.e the user's goal , it is left alone.
The attacking of AI players only occurs when the ball is moving towards their *own* goal. In this case the computer acts as follows :
First , the system checks the position of the ball (region). For this the coordinates of ball are received from its Position object 'p' using

a)GetXDistance()
b)GetYDistance()

Now, the system predicts the x-coordinate of the ball when the y-coordinate of the ball will be equal to y-coordinate of player of that region. In calculating the predicted coordinates, we used the initial coordinates of the ball and the direction of its velocity or slope, i.e. ,

m = (increment in y/increment in x) = posy/posx.

The ball can also collide with the wall , therefore simple laws of reflection have been used in predicting the future x coordinate of the ball.
Second , all players are moved by 0.4 cm , in such a way that the player *whose region the ball is in* comes closer to the predicted coordinates of the ball .This is done by setting
pposx = +0.4 ; // or pposx = -0.4;

More specifically , since the player is itself divided into 3 regions which will reflect the ball differently, we move the players till the required players reaches the desired coordinates for optimum reflection *towards* the goal.
Eg: players on the left will hit the ball with their right regions to direct the ball towards the goal.
When the required player reaches the desired coordinates , they are commanded to remain stationary , pposx = 0;
As a result,they are available to hit the ball when the ball comes to them .
Additional Features : Since this code is included in the same loop as that is moving the ball , many a time the players will not be able to reach the ball because it would have moved past owing to its faster velocity. This adds to the realism of the game attackers.
The above is implemented in the Program by making separate cases for the possible positions of the ball and is evaluated using if() statements. This code is included in the Loop() function and is called periodically.
3.)DESCRIPTION OF REGIONS-
A diagram is shown in which seven different regions ,red are the computer players and green are the user players