TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

PROJECT REPORT
ON
Penalty Fever

Submission Date: Falgun 22, 2076

Submitted by:

Sandeep Acharya (075BCT074)
Sangam Chaulagain (075BCT078)
Saujan Tiwari (075BCT083)

Submitted to:

Department Of Electronics and Computer Engineering

Acknowlegement

First of all, We would like to express our gratitude to the Department of Electronics and Computer Engineering, Pulchowk Campus for providing us the opportunity to build up the project.

We extend our sincere gratitude to Mr. Daya Sagar Baral sir for providing us not only the necessary object oriented concepts but also helping us regarding our real practical project.

Our special thanks go to our helpful seniors to provide us with better suggestions when needed. Lastly, We are grateful to our friends for highlighting the potential flaws in our system and helping us debugging it.

Abstract

The project “Penalty Fever” is a part of the Object Oriented Programming syllabus designed by IOE. This project was created for the better understanding of the Object Oriented Concepts through the real projects and to feel the practicality of creating programs, testing, adding features and debugging projects that are to be used by the real users.

For this purpose, we decided to make a game which is basically an imitation of the real life penalty in a football game. For the graphics purpose, we have used the SFML-2.5.1 library to make the game visually attractive. Since this is an imitation of a real game, we are not the first to make this type of game and there are dozens of pc and mobile games available already.

Table of contents

1. Objectives
2. Introduction
   - 2.1 Single Player
   - 2.2 Multi Player
3. Application
4. Literature survey
5. Existing System
6. Methodology
   - 6.1 Classes and Functions used
7. Implementation
   - 7.1 Block diagram
   - 7.2 Gameplay
8. Results
9. Problems Faced and solutions
10. Limitations and future enhancements
11. Conclusion and recommendations
12. References

1. Objectives

The major objectives of our project are:

- To learn about the object oriented approach and enhance our skills in C++ programming,
- To learn the basics of game development and become familiarized with SFML in C++ programming language,
- To learn to work in a team to make us able to work on major projects in the coming future.

2. Introduction

Penalty Fever, aptly named is a 2D football penalty shooting game. This game is inspired by the real penalty kick that may be awarded to either of the football teams when there’s a foul inside D-Box, also if there's a draw in an important game. Ourselves being football fans, we decided to develop the same phenomena as a game where users can interact with the game according to their choice.

Penalty Fever is a simple penalty game featuring two players. One is a goalkeeper and another is a shooter. Shooter shoots the ball towards the goal post according to the target set by the user and the goalkeeper tries to stop it. There are two teams in the game one is the user and another may be computer or user based on the mode of the game chosen at the main menu. The five kicks are awarded to either of the teams turn by turn exactly like in the real football game. The goals are counted each time the player scores. One scoring the most goals is declared the winner. There’s the hidden mode that’s triggered when the game is draw, For this mode, the winner is decided by a single kick.

2.1 Single Player

In this mode of the game, Only one player is needed to play the game while another player is the CPU. By default, User is assigned the control for the shooter. Users can control the target position by moving the cursor keys inside the goal post. When the user is finally ready and decides to hit the target at the specific post then the player can hit the SPACE button to kick the ball. After hitting the ball, We have defined some functions for the goalkeeper, goalkeeper accordingly dives and the game is continued. After the first shot, now the control is assigned to the Goalkeeper for the user and CPU plays the shooter part and the game progresses.

2.2 Multi Player

As the name suggests, In this mode we can control both the Shooter and the Goalkeeper. To make the game somewhat undecidable and entertaining, We have hidden the target location. One user controls the shooter and another controls the goalkeeper. After the shooter shoots the ball, the goalkeeper can make his move and stop the ball from hitting the target inside the goalpost. Controls are similar to that of Single player mode.

3. Application

This project “Penalty Fever” is a game. It's made for entertainment purposes. The game is simple but the score system used in the game makes it addicting. One can also get the idea about how the penalty shootout is performed in soccer as the penalty shootout in the game is much similar to the penalty shootout of real soccer game. The game implements the use of several mathematics theorems like pythagoras theorem in the motion of ball. So, the practical use of these theorems can be seen. This project can become one of the models to understand the implementation of the object oriented programming concept.

4. Literature Survey

The project is done to show the application of the concept of object oriented programming using c++. Our main reference for the OOP using c++ was "The secrets of object oriented programming" by DayaSagarBaral. The graphics library of cpp was not sufficient to make the game interactive. So, SFML library was used. Our reference for SFML was “Beginning c++ game programming” by John Horton ,”SFML Game Development“ by Jan Haller, HenrikVogelius Hansson, Et al and the documentation provided on the official SFML site.

5. Existing System

As of present time, We can find various similar games on various online platforms at the present time. These games are based on similar concepts and are of different names. ‘Final Kick’ by Ivanovich Games, ‘Perfect Kick’ by Gamegou Limited are some of the widely used penalty games. Similarly other penalty games like ‘World Cup penalty’, ‘Soccer shootout’, ‘Goalkeeper Premier’ etc are found in various online platforms.

6. Methodology

This project is based on the OOP concept using C++ programming language. Hence, different classes were created with a number of member data and member functions for the smooth running of the program. The events for each object have been handled by the different member functions so that they form a final outlook working together simultaneously.

This project needed the concept of graphics but since we only had the knowledge of C++ graphics which may not fulfill the required objectives of our project having limited features. For development of the game, we used SFML which can definitely fulfill the objectives of our project. So,we started learning SFML through various sites and e-books and understood the basic gaming logic through various gaming sites and forums along with the books regarding C++ and OOP itself.

Then we had to decide the right compiler and IDE. So,we decided to use Code::Blocks as IDE, which uses GCC Compiler. For graphics and other game related works, we decided to use SFML library. After collecting necessary materials, we developed the algorithm of our project and we worked according to it. The coding has been done according to the basic idea of OOP. We have created various classes and related functions and data binding has been used. Features of OOP has been implemented in our project.
For the completion of the projects different methods were implemented that are summarized as follows.

- Idea hunting: After forming the team, we collected and researched some of the projects done already in C++. After a long research we decided to make a penalty shootout game.
- Graphics Library dilemma: Getting the idea was the first part but implementing it is a completely different one. We found some graphics

libraries used in c++ like SDL, SFML, WxWidgets, Qt. For our needs and given time limit, our dilemma ended with SFML.

- Proposal submission: Before coding, we made the basic framework for the project and documented that information on our proposal.
- Coding: We already made the basic layout of the project before proposal submission and after our proposal was accepted we started coding slowly to make the projects components one by one.
- Testing, Modifications and Debugging: After finishing the basic model of the game and implementing the function we started testing and adding features to it. We were constantly finding and removing bugs in our program throughout this process.
- Documentation: After finishing our project, Now we had to document it properly. We summarized all the process and works we carried out till the completion of the project.

6.1 Classes and Functions Used:

Main file:

- Main.cpp
    - game.run( ) : calls run function from Game class
- Classes
    - Game
    - Football
    - Mainmenu
    - Goalkeeper
    - Goalpost
    - Target
    - Collision: For Pixel perfect collision, Taken from Sonar Systems.
    - Shooter

Class Descriptions:

- Game
    - Functions:
        - Game( ) : Constructor, initializes all necessary variables for the game
        - run( ) : Calls processEvents( ), update( ), render( ) functions
        - processEvents( ) : Handle all keyboard and mouse inputs
        - update( ) : to keep fixed time frame
        - render( ) : All draw functions inside
        - resetScore( ) : Resets score every new game

    Mainmenu

    - Functions:
        - mainmenu() : Constructor
        - sf::Sprite getsprite() : Get mainmenu sprites to draw from the Game class
        - sf::Text gettext() : Same purpose as getsprite
        - bool checker() : Checks if mouse is over any text
        - void setcolor() : Sets color when mouse is over the text
        - void resetcolor() : Resets color when mouse is removed from text
        - Void setpos(float, float) : Set position of text in main menu
        - void setstring(std::string) : Fills string inside text content

    Football

    - Functions:
        - Football() : Constructor
        - sf::Sprite getSprite() : Returns the sprite of football
        - bool kick(sf::Vector2f,sf::Vector2f &newpos) : Generates the motion of football from intial position to target position
        - void spawn() : Resets the postion of football to its original (shooting) position
        - sf::Vector2f getPosition() : Returns the position of the football

- void goalSound() : Plays sound when goal is scored
- void missSound() : Plays sound when shot is missed
- void whistleSound() : Plays sound when plays gets ready to shoot the ball
- void draw(sf::RenderWindow&window) : Draws the football sprite and the circle in the intial position of ball in ground
- void draw(sf::RenderWindow&window,int,int,int,int) : Draws the sprites for goal scored and goal missed according to situation for both players.

Target

- Functions:
  - Target () : Constructor, Defines target parameters
  - void handle_input() : Handle keyboard input for moving target
  - void moveUp();
  - void stopUp();
  - void moveDown();
  - void stopDown();
  - void moveLeft();
  - void stopLeft();
  - void moveRight();
  - void stopRight();
  - void spawn() : Reset position for next game
  - void setSpritePosition(int,int) : Set target position for single player mode
  - void setSpritePosition(sf::Vector2f) : Set target position for multiplayer mode
  - sf::Vector2f getPosition() : Pass target position to the game class
  - void update(sf::Time) : Update target after moving without altering time frame

        - void checkPosition() : Checks position of target inside goalpost stops from going outside goalpost

- Shooter
    - Functions:
        - Shooter() : Constructor, Defines shooter parameters
        - sf::Sprite getSprite() : Pass sprite to game class for drawing purpose
        - void spawn() : Reset after new game
        - void kick() : Kick the ball to specified coordinates
        - void setToBlue() : Set color to blue
        - void setToRed() : Set color to red
        - sf::Vector2f shootPosition() : Pass coordinates to shoot

Goalkeeper

- Functions:
    - Goalkeeper() : Constructor, Defines goalkeeper parameters
    - boolhandle_input() : Handle keyboard input for moving goalkeeper
    - void kick(int,float) : Dive function for goalkeeper
    - void spawn() : Reset after new game
    - sf::Vector2f getPosition() : Pass position of goalkeeper to Game class
    - intdivePosition() : Return position for diving according to the conditions defined inside
    - void setToBlue() : Change color to blue
    - void setToRed() : Change color to red
    - void update() : Update goalkeeper
    - void resetPosition() : Reset position after every kick

Goalpost

- Functions:
    - Goalpost() : Constructor, Initializes goal post parameters

- sf::Sprite getSprite() : Method for passing sprite to game class
- void draw(sf::RenderWindow&window) : Draw goalpost function

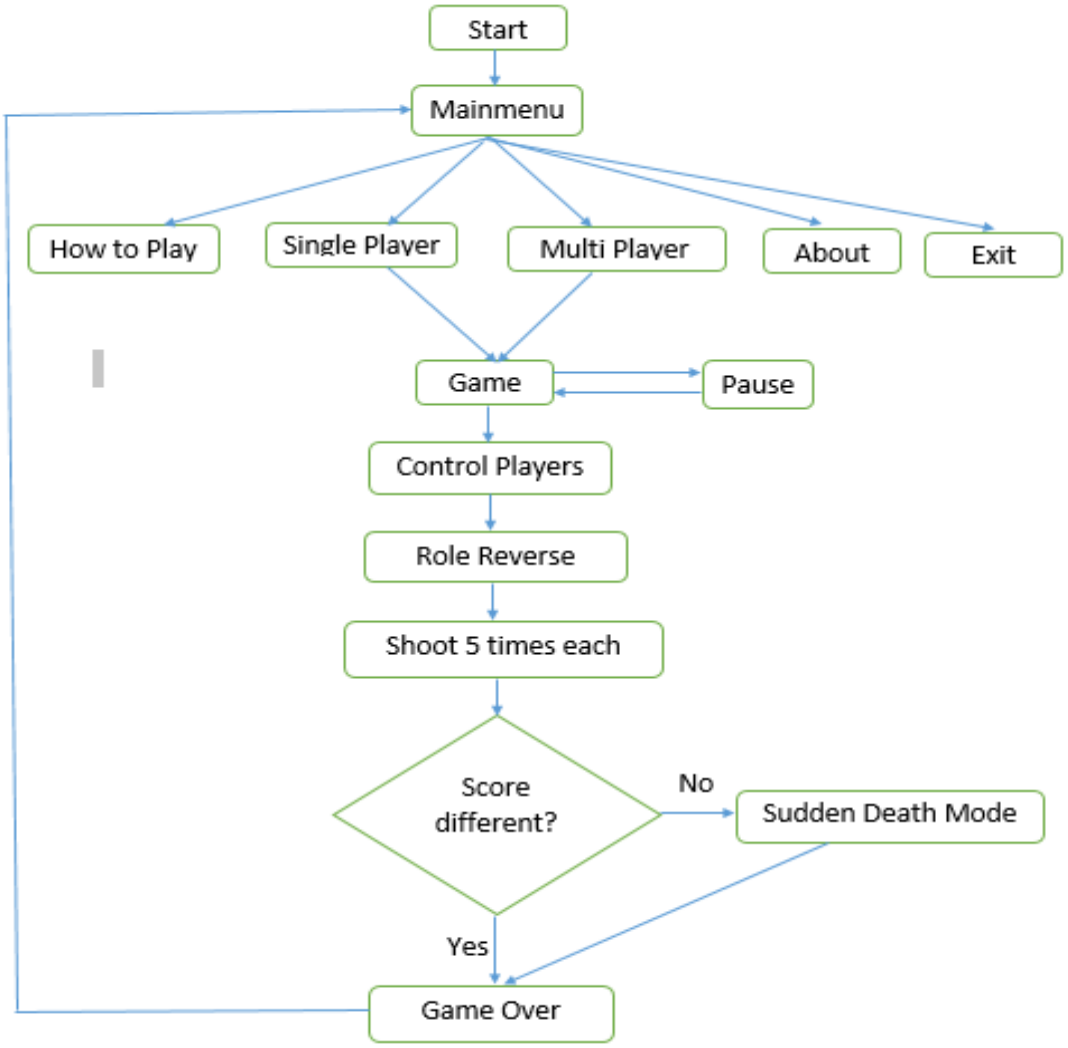
7. Implementation

The first interaction of the user with the game is the main menu that consists of altogether five options:

- Single Player: By clicking this option, User agrees to play with the CPU turn by turn, By default user is team red and s/he can’t change the team.
- Multiplayer: This option allows the user to play with another user, both players compete with one another to win his counterpart.
- How to play: This option is the tutorial section to show the user how to play the game for the first time.
- About: This option allows the user to find out more information about the game itself and the developers.
- Exit: It is simply the graphical option to exit the game.

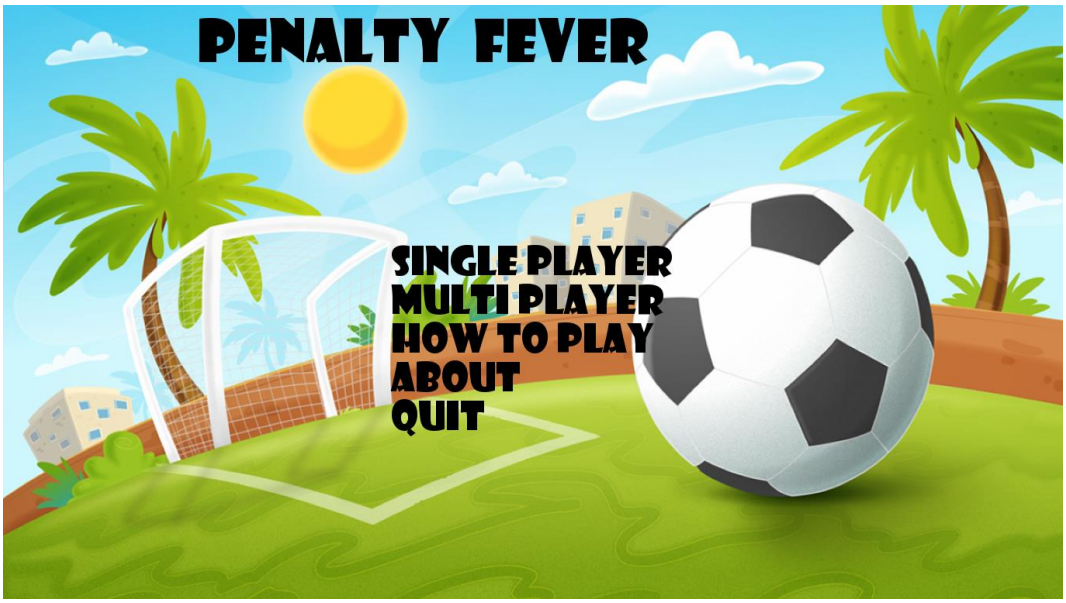
After choosing the first two options users have five kicks to spare or the user can pause the game if he finds something important to do, Progress is not lost. Finally the game over menu appears with the scores and congratulation texts.

7.1 Block diagram

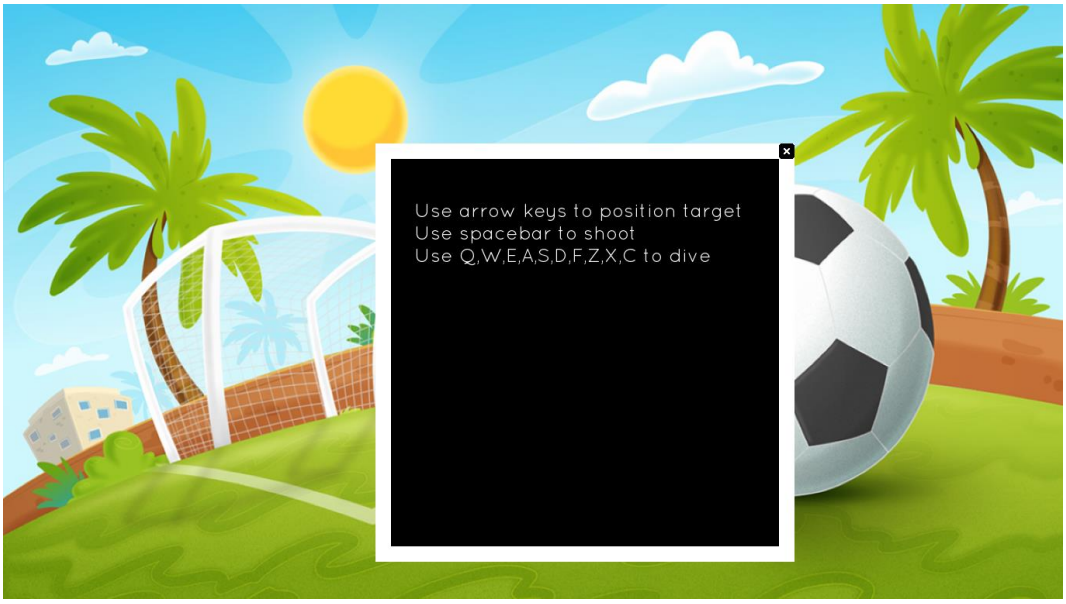


7.2 Game play

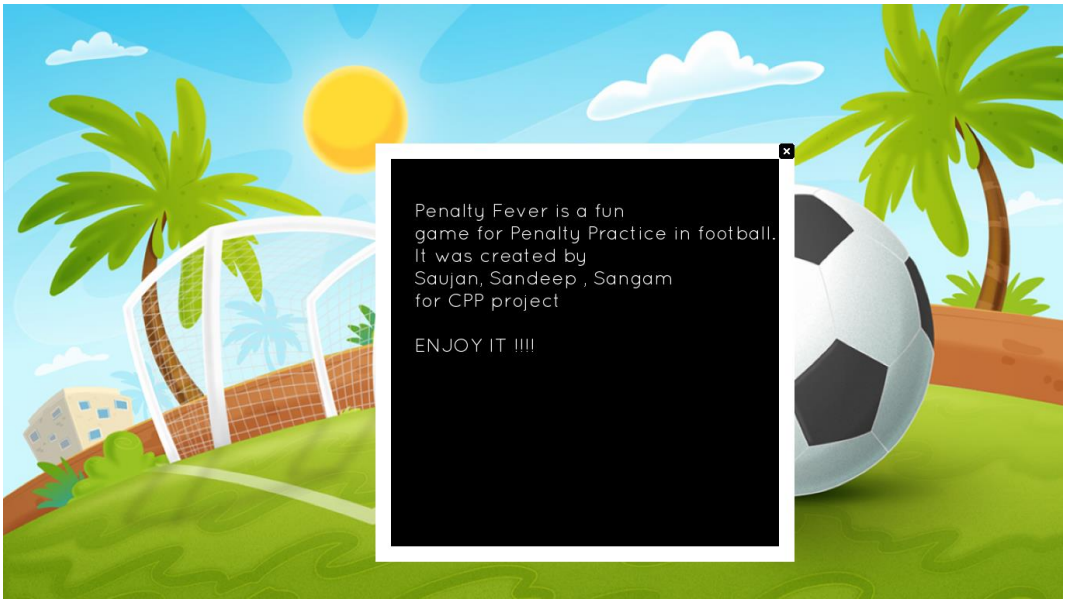
Main Menu



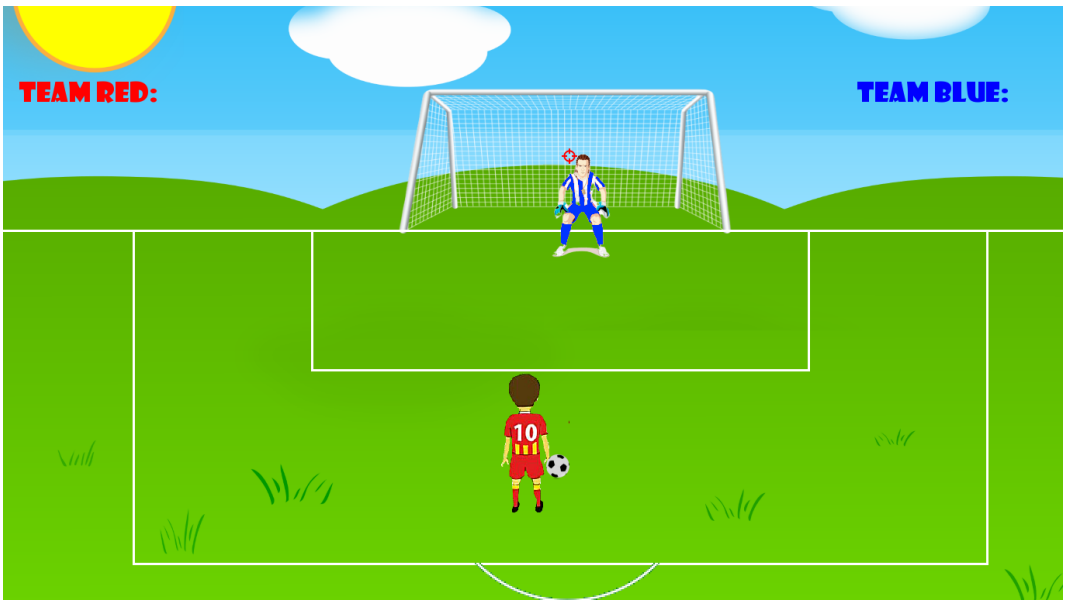
How to Play



About



Playing state



GameOver

8. Results

The main objectives of the project were completed with its completion. We designed the complete game with the core features which makes it a penalty shootout game.

We couldn’t include all the features that could have been added to the game which would make it even more interactive. It was mainly because of the lack of time. As both core concepts in the game, object oriented programming and SFML were new to us. Huge portion of our time was taken to take these concepts to the level so that implementation can be done in an efficient manner in our project. Also, the complexity of some of the topics made us back out.

The project made us clear about basic game development. We learnt about teamwork and it’s value while working on the project. It has become our motivation for the future projects minor or major.

9. Problems faced and solutions

Throughout the completion of this project, we faced different types of problems related to logic, syntactical, installing libraries and many more. We tried our best to solve the problems ourselves but being as a complete noob in graphics libraries like SFML, it was a difficult task.We sought help from the professionals, friends, discussion forums, official documentation, teachers, books and most importantly google as a search engine. The major problems faced and solutions used are as follows:

- As installing libraries in C++ was new to us, we faced many difficulties there. At first we were unknown where to put static libraries, dynamic libraries, to configure linkers and compilers. Through some research and online guides we were able to link the libraries properly.
- While working on the Collision problem, we first used the rectangular collision detection. This could be carried out easily using the function given in SFML (sf::Sprite::getGlobalBounds( ) const). But it looked unrealistic and wonky. Thanks to sonar systems, there was a pixel perfect collision function made already. Because of our time limitations and little to no knowledge of graphics we used the function straight up for that.
- We had planned to make the game with an accuracy and power slider to hit the ball, but it seemed target selection inside the goalpost would look nice and more realistic than the slider. Hence, we ditched that idea and implemented target selection.
- We were not much experienced with the version control systems like git and haven't used it for the team project. We faced many problems while merging code and putting it all together.
- Problem also occurred when moving the ball towards the target from the ground. We solved it ourselves using simple logic.

10. Limitations and future enhancements

Though the game is completed for now, future enhancements are possible in the game. The game can be made more attractive and realistic using different techniques. Some of the limitations and the possible future enhancements are discussed below:

- Since SFML is itself a limited library with basic graphics capabilities. We could choose other high class graphics like OpenGL, Direct3d, Vulkan to make the game 3d and more realistic.
- Because of the time limitation we couldn't add a free-kick version of the game. In that mode we could add different players, ball movement and celebrations.
- The game can be made more interesting by using networking in the multiplayer mode where one computer becomes the shooter and another becomes the goalkeeper. This way, Both users need not be on the same computer that would make it thrilling.
- A section can be made entirely for the player customization and selection of backgrounds. Users could choose different players and backgrounds according to their likes and dislikes. Ball rebound feature can also be added.
- We had little knowledge of data structures and algorithms. Hence we may not have chosen the right containers for holding the different data in the game and may not have used efficient algorithms. We could improve it after gaining some insight in the respective field of study.
- Game over section can be made mode interactive by adding different celebration animations. E.g player holding the trophies, jumping with excitement.

11. Conclusion and recommendations

This project based learning was a great way of understanding the concepts of programming and implementing them. This project helped us to understand the ways of developing new softwares. More than just the coding it taught us other variables to be considered when building software. First of all, we need to find the idea that we work upon. Of course, the idea hunting should be taken more seriously and a lot of research and analyzing of the older projects is required to know if our proposed project is fit enough for us. We need to choose the libraries that best fit for us considering time limit, quality, knowledge e.t.c. and we can't just hop into any library and start coding. Coding is by all means the important part of the project. It helps us to solidify our concepts and also helps our knowledge to benefit the real world. Finishing the project is only half way through, many bugs are found here and there by testing it with a bunch of people and more testing, more bugs and more debugging is required. So maintenance is also an important part of a project that's often overlooked.

12. References

- https://www.sfml-dev.org
- SFML Game Development: Jan Haller, HenrikVogelius Hansson, Et al
- https://github.com/SonarSystems/SFML-Box2D-Tutorials/tree/master/SFML/Tutorial%20013%20-%20Pixel%20Perfect%20Collision%20Detection : For pixel perfect collision detection
- DayaSagarBaral and DiwakarBaral, "The Secrets of Object Oriented Programming in C++", BhundipuranPrakasan
- https://www2.lunapic.com
- John Horton,"Beginning c++ game programming"