

**TRIBHUVAN UNIVERSITY**

**INSTITUTE OF ENGINEERING**

**THAPATHALI CAMPUS**

**A Project Report**

**On**

**Street Fighting Game Using C++**

**Submitted By:**

Ram Bharati (THA074BEX027)

Pukar Giri (THA074BEX025)

Sushil Khadka (THAO74BEX046)

Rikrish Shrestha (THA074BEX030)

**Submitted To:**

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

March, 2019

# COPYRIGHT

The author has agreed that the Library, Department of Electronics and Computer Engineering, Thapathali Campus, Institute of Engineering may make this report freely available for inspection. Moreover, the author has agreed that permission for extensive copying of this project report for scholarly purpose may be granted by the supervisors who supervised the project work recorded herein or, in their absence, by the Head of the Department wherein the project report was done. It is understood that the recognition will be given to the author of this report and to the Department of Electronics and Computer Engineering, Thapathali Campus, Institute of Engineering in any use of the material of this project report. Copying or publication or the other use of this report for financial gain without approval of to the Department of Electronics and Computer Engineering, Thapathali Campus, Institute of Engineering and author’s written permission is prohibited.

Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

Head

Department of Electronics and Computer Engineering

Thapathali Campus, Institute of Engineering

Lalitpur, Kathmandu

Nepal

# ACKNOWLEDGEMENT

We extend our sincere gratitude to Er. Saroj Shakya for providing the necessary guidelines and support for understating the feasibility and technical aspects for the project. Similarly, we also like to thank the Developers of SDL library for providing us with the necessary tools. We would also like to thank Thapathali Campus and Department of Electronics and Computer Engineering for providing us the opportunity to build up the project. Last but not the least; we would like to thank our friends and seniors for helping us in the project. Thanks for making the difference.

Ram Bharati (Class Roll No.: 074/BEX/027)

Pukar Giri (Class Roll No.: 074/BEX/025)

Sushil Khadka (Class Roll No.: 074/BEX/046)

Rikrish Shrestha (Class Roll No.: 074/BEX/030)

# ABSTRACT

Fighting games are a very niche genre that challenges the player and forces them to admit their own failings. Because of this high entry barrier, fighting game community is quite small and there are not that many new players. What forms this entry barrier is that player has to take in lots of information right from the beginning, not many mechanics are properly explained and they lack basic knowledge of how to play. Where this leads to is button mashing, frustration and eventually to drop of interest. I want to tackle that problem by creating a fighting game that teaches basic fighting game knowledges for new players and helps them move to other fighting games. Usually, new players do not understand differences between character archetypes, varying match situations or how to use their character’s tools.

While this may not appeal to veteran players, game like this would benefit the fighting game community by bringing in new players. While there has been fighting games that try to tackle the problem by simplifying the gameplay, teachings from them may not translate into other games. This way the contact into the world of fighting games is very shallow and will not improve new players. Usually, they also do not try to teach basic knowledge which would improve the experience

# **Table of Contents**

[COPYRIGHT i](#_Toc3103286)

[ACKNOWLEDGEMENT ii](#_Toc3103287)

[ABSTRACT iii](#_Toc3103288)

[Table of Contents iv](#_Toc3103289)

[List of Figures vi](#_Toc3103290)

[List of Tables vii](#_Toc3103291)

[List of Abbreviations viii](#_Toc3103292)

[1. INTRODUCTION 1](#_Toc3103293)

[1.1 Background Introduction 1](#_Toc3103294)

[1.2 Motivation 1](#_Toc3103295)

[1.3 Objectives 1](#_Toc3103296)

[1.4 Scope 2](#_Toc3103297)

[2. LITERATURE REVIEW 2](#_Toc3103298)

[3. METHODOLOGY 4](#_Toc3103301)

[3.1 Artificial Intelligence 4](#_Toc3103302)

[3.1.1 Reinforcement learning 5](#_Toc3103303)

[3.2 Networking/Socket Programming 5](#_Toc3103304)

[3.2.1 Socket APIs and Libraries 6](#_Toc3103305)

[3.2.2 Socket Interface Types 6](#_Toc3103306)

[3.3 Simple DirectMedia Layer(SDL) 7](#_Toc3103307)

[4. SYSTEM DESCRIPTION 9](#_Toc3103308)

[4.1 Use of SDL 10](#_Toc3103309)

[4.2 Game Modes 10](#_Toc3103310)

[4.3 Controls 10](#_Toc3103311)

[5. RESULTS AND ANALYSIS 10](#_Toc3103312)

[5.1 Comparison with actual game 11](#_Toc3103313)

[5.2 Comparison with other AI's 12](#_Toc3103314)

[6. CONCLUSION AND FUTURE ENHANCEMENT 13](#_Toc3103315)

[6.1 Conclusion 13](#_Toc3103316)

[6.2 Limitations 13](#_Toc3103317)

[6.3 Future Enhancement 13](#_Toc3103318)

[References 5](#_Toc3103322)

# List of Figures

[Figure 3-1: Block Diagram of AI 4](#_Toc476258198)

[Figure 3-1-1: Block Diagram of Reinforcement Learning 5](#_Toc476258199)

[Figure 3‑2: Block Diagram of Socket Programming 7](#_Toc476258200)

[Figure 3-1: Block Diagram of Program Flow 8](#_Toc476258198)

[Figure 4-1-1: Spritesheet 10](#_Toc476258198)

[Figure 4‑1-2: Background 11](#_Toc476258201)

[Figure 5: Screenshot of Output 13](#_Toc476258198)

# List of Tables

[Table 4‑1 Controls 12](#_Toc476258538)

[Table 5‑1 Comparison of AI 14](#_Toc476258538)

# List of Abbreviations

|  |  |
| --- | --- |
| CPP | C Plus Plus |
| IDE | Integrated Development Environment |
| API | Application Programming Interface |

# INTRODUCTION

## Background Introduction

While most games focus on giving players relaxing time, fighting games go against this way of thinking. Because of their high entry barrier, the fighting game genre has always been niche. Even among competitive players, fighting games have never reached as huge popularity as other competitive games. Nevertheless, fighting game community attracts dedicated players that are willing to spend the time to learn these difficult games.

While the high entry barrier makes it difficult for the new players, the deep and challenging gameplay has created a dedicated group of followers. This thesis tries to determine what the reason is for this by studying the gameplay of fighting games. Using the found knowledge, this thesis also aims to create a new fighting games.

## Motivation

Fighting games as we know them nowadays emerged during the golden period of arcade halls. Fighting games short play sessions, player versus player and competitive nature suited well into the dark rooms of arcades. During the downfall of arcade halls, fighting games also saw a drop in popularity. This lead into a decade of obscurity and silence as the genre saw few new games. The rise of online gaming ushered in a new era of fighting games as the need of physical arcade machines disappeared and players could find other players from their couch. Fighting games gained new rise in popularity and once again new games flooded the market

## Objectives

The main objectives of our project are listed below**:**

To design a fighting game that teaches basic fighting game knowledge to new players and to elevate their interest in fighting game genre.

To implement the concepts of Object Oriented Programming.

## Scope

* This project will certainly help us to become a competent C++ and/or game programmer.
* It can be a good tool for recreation among youngsters.
* This project teaches us advance topics like AI, networking, 2D graphics.
* This project demonstrates the basic usage and creation of simple physics engine.
* This game can be commercially viable.

# LITERATURE REVIEW

Fighting games are a genre of games where the focus is on one versus one matches against another player or computer using a wide variety of characters. The objective usually is to deplete other character’s health with different means. Fighting games are highly skill-based games, which leads to their competitive nature. Because fighting games focus on one versus one, they are mostly played against other players. This can be seen in the lack of single player content. (Dargenio 2014, Fanell 2011, Racketboy, 2011).

Fighting games have many easily recognizable features that you can find in almost all of this type of games. Character rosters contain many easily recognizable characters with distinct appearance and fighting style. This makes them easily distinguishable during hectic matches. The user interface in fighting games is similarly very unique to the genre. It is polished to give players the needed information quickly and easily. (Dargenio 2014, Fanell 2011, Racketboy, 2011

Many 2D fighting games also share gameplay mechanics between each other. Combos, which are moves strung together, are an important aspect of fighting games. Because of this fighting games include a variety of punches and kicks. They vary by strength and speed, which allows players to do a different type of combos. Also, one very essential feature is special moves which are on every 2D fighting game. Special moves are unique skills that allow characters to do unusual attacks, such as throwing fireballs and they are done by doing certain but-ton patterns. These moves give the characters their own unique playstyle and make them stand out from each other. (Dargenio 2014, Fanell 2011, Racketboy, 2011).

# METHODOLOGY

We used C++ core concepts as the backend framework and for physics engine. SDL was used to provide 2D graphics support in frontend. Advanced techniques like AI and networking would made our game more approachable.

## Artificial Intelligence

We used current machine learning technique of reinforcement learning using Q-learning algorithm. This made the game more fun even in single player mode.

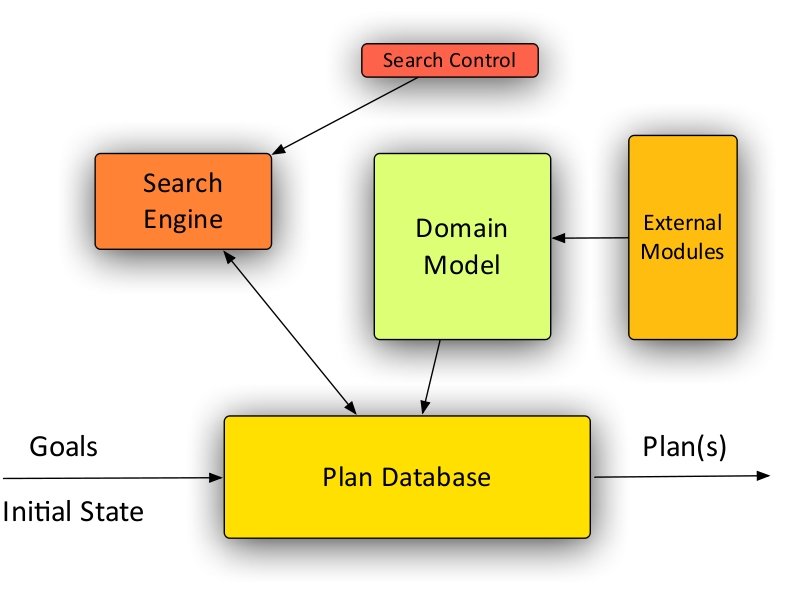


Figure3‑: Block Diagram of AI

### Reinforcement learning

Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Reinforcement learning is considered as one of three machine learning paradigms, alongside supervised learning and unsupervised learning. Main points in Reinforcement learning:

Input: The input should be an initial state from which the model will start

Output: There are many possible output as there are variety of solution to a particular problem

Training: The training is based upon the input, The model will return a state and the user will decide to reward or punish the model based on its output.

The model keeps continues to learn.

The best solution is decided based on the maximum reward.

###### battleship2.png

###### Figure3‑1-1: Block Diagram of Reinforcement Learning

## Networking/Socket Programming

Socket programming was used to provide peer to peer network ability over local area network which would support for multi-player mode of the game.

### Socket APIs and Libraries

Several libraries that implement standard application programming interfaces (APIs) exist on the Internet. The first mainstream package - the Berkeley Socket Library is still widely in use on UNIX systems. Another very common API is the Windows Sockets (WinSock) library for Microsoft operating systems. Relative to other computer technologies, socket APIs are quite mature: WinSock has been in use since 1993 and Berkeley sockets since 1982.

The socket APIs are relatively small and simple. Many of the functions are similar to those used in file input/output routines such as read(), write(), and close(). The actual function calls to use depend on the programming language and socket library chosen.

### Socket Interface Types

Socket interfaces can be divided into three categories:

Stream sockets, the most common type, requires that the two communicating parties first establish a socket connection, after which any data passed through that connection will be guaranteed to arrive in the same order in which it was sent - so-d connection-oriented programming model.

Datagram sockets offer "connection-less" semantics. With datagrams, connections are implicit rather than explicit as with streams. Either party simply sends datagrams as needed and waits for the other to respond; messages can be lost in transmission or received out of order, but it is the application's responsibility and not the sockets to deal with these problems. Implementing datagram sockets can give some applications a performance boost and additional flexibility compared to using stream sockets, justifying their use in some situations.

The third type of socket is the raw socket, bypasses the library's built-in support for standard protocols like TCP and UDP. Raw sockets are used for custom low-level protocol development..

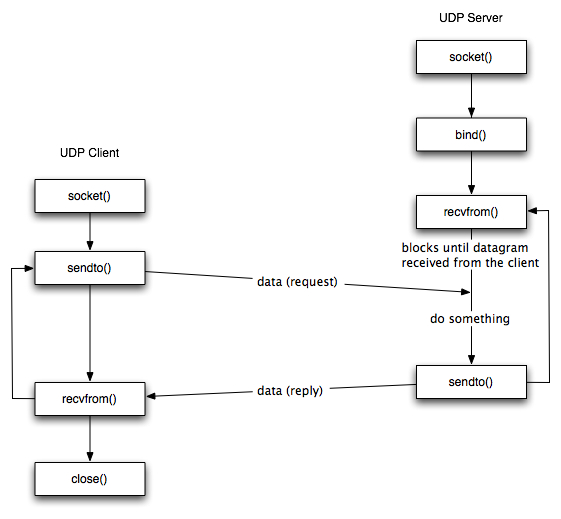


Fig3-2-2: Block Diagram of Socket Programming

## Simple DirectMedia Layer(SDL)

Simple DirectMedia Layer (SDL) is a cross-platform software development library designed to provide a hardware abstraction layer for computer multimedia hardware components. The Simple DirectMedia Layer(SDL) is used in our project to provide the low level graphics rendering and to interface with various input devices at low level.

# SYSTEM DESCRIPTION

A simple menu is provided whenever game is lauched. User can select in between single player or multiplayer. Single player mode allows user to play with automated bot while multiplayer is between two player.

Menu

Single Player

AI

Multiplayer:

Same Machine

Via Network

Fig4-1: Block Diagram of Program Flow

## Use of SDL

The Simple DirectMedia Layer(SDL) is used in our project to provide the low level graphics rendering and to interface with various input devices at low level. The library has only been used as a supportive low level framework upon which all other capabilities of the game are built. Our project revolves around a simple technique of rendering the rectangles on windows. The rendering was done by using SDL\_RenderPresent() function.Our job was to manipulate the rectangle. We used a clever technique of selecting sets of animation frames with vector of type SDL\_Rect. The rectangle contained the carefully searched coordinates as well as height and width of the every frames needed. Later the rectangle was pointed to a spritesheet which contained the actual image. Then we used various functions to execute the appropriate frames as per user input. The user input was captured using SDL\_PollEvent() function.SDL\_PollEvent function provided us realtime user inputs. Then we called appropriate functions as per user input which inturn called the appropriate sequence of animation frames. Some of the most used SDL tools:

SDL\_Windows: SDL\_Window is the struct that holds all info about the Window itself: size, position, full screen, borders etc.

SDL\_Renderer: SDL\_Renderer is a struct that handles all rendering. It is tied to a SDL\_Window so it can only render within that SDL\_Window. It also keeps track the settings related to the rendering.

SDL\_Surface and SDL\_Texture: The SDL\_Renderer renders SDL\_Texture, which stores the pixel information of one element. It's the new version of SDL\_Surface which is much the same. The difference is mostly that SDL\_Surface is just a struct containing pixel information, while SDL\_Texture is an efficient, driver-specific representation of pixel data.

SDL\_Rect: The simplest struct in SDL. It contains only four shorts. x, y which holds the position and w, h which holds width and height.

Considering of SDL\_Window as physical pixels, and SDL\_Renderer a place to store settings/context helped a lot.

1. **SpriteSheet**

Spritesheet is bitmap image file that contain several smaller graphics in a tiled grid arrangement. By compiling a several graphics into a single file, you enable animate and other applications to use the graphics while only needing to load a single file.

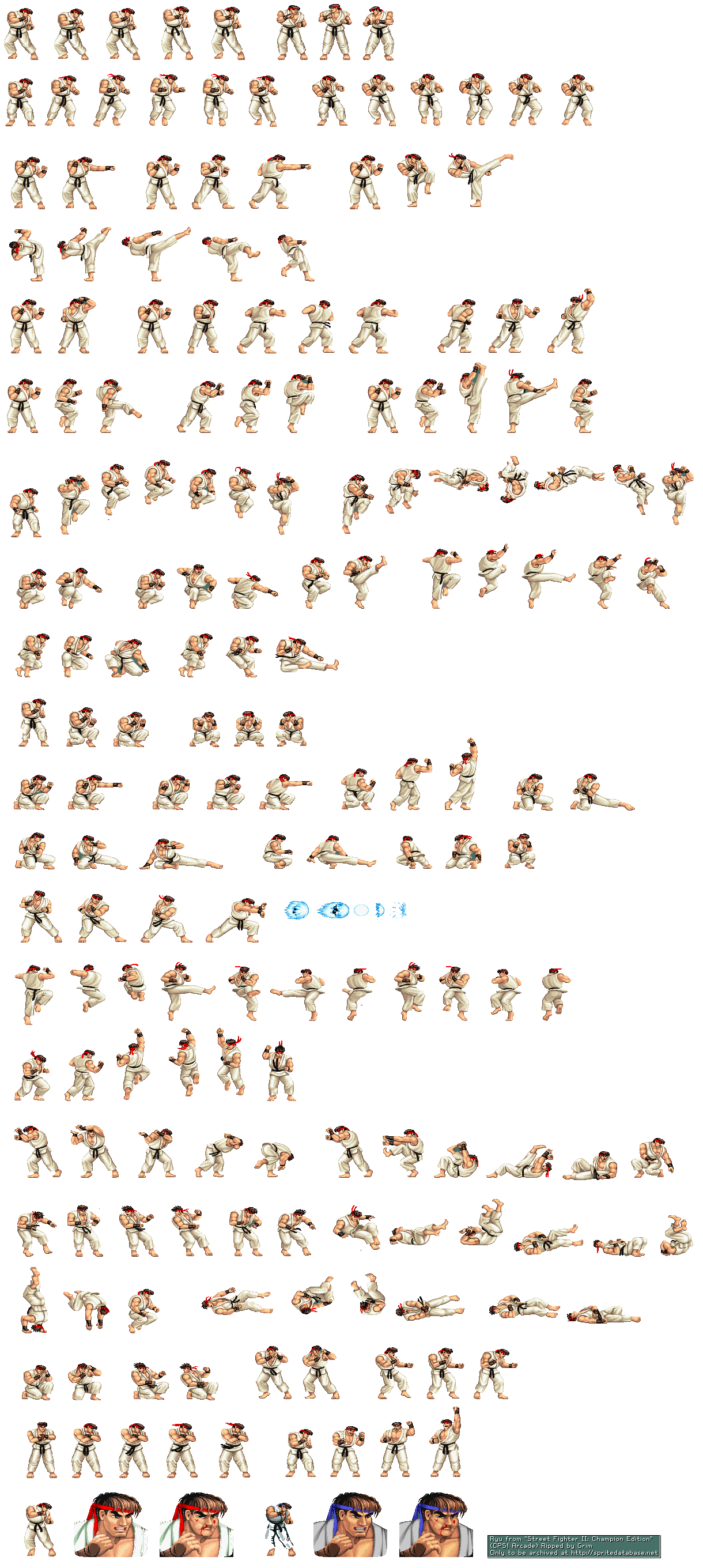


Figure4-1-2:Spritesheet 1

1. **Background Surface**

Background surface is used as a background in our game. Every other rectangles are drawn in top of this background surface. So this acts as a stage for our game.



Figure4-1-1: Background

## Mode

* 1. Single Player(against AI)

This mode allows player to play with AI. AI is powered with machine learning algorithm belonging to reinforcement learning called Q learning.

* 1. Multiplayer:

This mode allows two players to play against each others either in same machine or via network(LAN). Networking feature is provided via the native Linux Socket API and syscalls.

## Controls

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Player | Move left | Move right | Jump | Punch | Kick |
| Player 1 | A | D | W | E | Q |
| Player 2 | Left Arrow | Right Arrow | Up Arrow | Num-pad 0 | Slash-key |

Table4-3: Controls

# RESULTS AND ANALYSIS

Overall, player moves ,attacks and jumps according to user's need.All audio, animations, UI, gameplay were well-handled by our code with the help of SDL library.The AI bot in this game is still in learning phase. But continous progress was observed. As for networking, two computer were able to be connected and play against each other simultaneously (connection to same network is required ).



Figure5 :Screenshot of Output

## 5.1 Comparison with actual game

This game is simply a small remake of Street Fighter 2 except of the fact that we have use AI algorithm in it. SF2 has no such AI. It has a bunch of simple scripts chosen randomly based on difficulty choosed by user and time left in that round. But, our algorithm learns from the user playing style and tries to adapt to it. AI in SF2 is simply good because it can cheat. The Computer can know your move as soon as you have input it, before the first animation frame has even displayed. As such it gets one more frame of advantage on top of your reaction time.

###### download.jpgFigure5‑: Screenshot from actual game

## Comparison with other AI's

As this project was made for learning purposes. We focused on using simple C++ programming and OOP concepts with AI rather than trying to make a perfect AI for any game. Other worldwide programmers have built AI's that can play and adapt to not only one but many games without even changing the code. And even giant tech industry like Google has been trying to make so-called Universal AI that can adapt to any situation like a human being.So, relatively our project would'nt seem that extraordinary but it has been like crossing a big milestone for us as students.

|  |  |  |  |
| --- | --- | --- | --- |
| Others AI | Static | Fixed values | If else tree |
| Our AI | Machine learning | Values learned | Reinforcement learning/Q learning |

Table 5‑1 Comparison of AI

# CONCLUSION AND FUTURE ENHANCEMENT

## Conclusion

The purpose of this project was to learn about OOP concepts in C++. But, extra features required graphics and AI which helped us to learn even more. We now know that OOP helps us to write efficient, readable, reusable code. We also learned that making a game and AI algorithm selection is not always straightforward. For instance, if we had made a 2D scrolling game insted of 2D Fighting game, it would require a different asset, game system, and even a different AI technique might be needed. The AI algorithm and its hyperparameters used affects the accuracy and precision of the AI bot. So, choosing the most suitable one requires good level of experience.

## Limitations

The limitations are as follow:

* The game exhibits a small lag due to the design defect.
* Only two player are supported in any multiplayer mode.
* AI can be improved using better hardware and better algorithms like Deep Q learning.

## Future Enhancement

There is a lot of room left to improve in this game as this is just minimum viable product. Some of main areas to improve are:

1. Variety of characters , powerups, backgrounds . Different voices for each character are still needed to be implemented.
2. As different characters , sounds , levels get added, code design can change (not the whole concept).New classes, design patterns may be added.
3. Online multiplayer mode could be added so that people all around the world could challenge each other or play with AI which will then make it even smarter.
4. Lots improvement can be on how the AI learns.

# References

|  |  |
| --- | --- |
| [1] | L. Prechelt, B. Unger, W.F. Tichy, P. Brossler, L.G. Votta, A controlled experiment in maintenance comparing design patterns to simpler solutions, IEEE Transactions on Software Engineering 27 (12) (2001) 1134–1144. |
| [2] | M. Voka´c, W. Tichy, D.I.K. Sjøberg, E. Arisholm, M. Aldrin, A controlled experiment comparing the maintainability of programs designed with and without design patterns – a replication in a real programming environment, Empirical Software Engineering 9 (2004) 149–195. |
| [3] | L. Bishop, D. Eberly, T. Whitted, M. Finch, M. Shantz, Designing a PC game engine, IEEE Computer Graphics and Application (1998) 46–53. |
| [4] | S. Bjork, S. Lundgren, J. Holopainen, Game design patterns, in: Lecture Note of the Game Design track of Game Developers Conference 2003, March 4–8, San Jose, CA, USA, 2003. |
|  |  |