TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A PROJECT REPORT ON APPLICATION OF OBJECT-ORIENTED PROGRAMMING USING C++ "MARIO"

Submitted To:
Department of Computer and Electronics Engineering

Submitted By:
Nijiya Maharjan (078BCT052)
Rubika Bashyal (078BCT068)
Sadhana Panthi (078BCT069)

1. Acknowledgement

We express our gratitude to all those who have helped us throughout the duration of this project. We would like to thank our Object Oriented Programming teacher, Mr. Daya Sagar Baral. His insightful feedback and unwavering encouragement have been instrumental in shaping our understanding of object-oriented programming.

We would also like to extend our thanks to the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering for providing us with the platform and resources to undertake this project.

Lastly, we extend our sincerest appreciation to the ingenious minds behind the creation of Mario, whose visionary work laid the foundation upon which our project stands. Their creativity continues to inspire us and countless others in the gaming world. We thank them for shaping the magic that has sparked our own journey.

This project has not only enhanced our programming skills but has also been an enjoyable journey of learning and growth. The knowledge and skills we have acquired during this endeavor will undoubtedly serve as a strong foundation for our future endeavors in the field.

Authors:
Nijiya Maharjan (078BCT052)
Rubika Bashyal (078BCT068)
Sadhana Panthi (078BCT069)

2. Abstract

This project report outlines the project work submitted in the partial fulfillment of the requirements for the course on Object Oriented Programming as prescribed in the syllabus for BCT II/I.

The main aim of this project was to develop a user-friendly program using an Object Oriented Programming language in C++. For this project we have created a clone of the classic platformer game Mario. We have recreated most of the features of the original game, including gameplay style, artwork, and sound.

While the game is fully functioning, there is room for improvement. We plan to add improvements in the future to make the game more entertaining.

Keywords: Mario, SFML, OOP, C++

3. Table of Contents

4. Objectives

The main objectives of our project are as follows:

- To apply the concepts of Object Oriented Programming in practical application.
- To implement external libraries like SFML for game development.
- To learn the basics of game physics, particularly world building and character interaction.
- To work and communicate in a team environment.
- To get familiar with version control with git and collaboration tools like GitHub.
- To make a fully functional Mario game which is entertaining to play

5. Introduction

For this project, we recreated the classic platformer game Mario using object oriented programming in C++. We reproduced most of the features of the original game, including gameplay style, artwork, and sound.

The player controls the titular character Mario using keyboard controls (up, left and right). The objective is to get through the level as fast as possible to reach the end. The game world includes coins for Mario to collect and special bricks marked with a question mark (*?*), which when hit from below by Mario reveals a Super Mushroom. If Mario gets hit in this mode, then instead of dying he turns back to regular Mario.

The game starts with Mario having 3 lives. He loses a life if he takes damage while small or falls in a bottomless pit. The game ends when the player runs out of lives or reaches the end of the level.

Enemies of various kinds are found throughout the level, consisting of plants, spikey, normal turtle and flying turtle. The turtles can be killed by jumping on top of them, while the other enemies cannot be killed.

6. Application

Mario is not only fun to play, it offers a range of benefits to hand-eye coordination and spatial awareness. The game can be challenging, requiring players to try and retry to succeed. This promotes a sense of resilience and perseverance. Mario games can be a fun and relaxing way to spend time. They can help to take your mind off of your troubles and relieve stress. Additionally, the retro graphics and sound effects can be nostalgic for many players.

Mario has been made many times using different programming languages. Even if the project is not on par with the projects made by mainstream developers, our version of Mario provides a reliable platform for entertainment.

7. Literature Survey

7.1. C++

7.1.1. Introduction

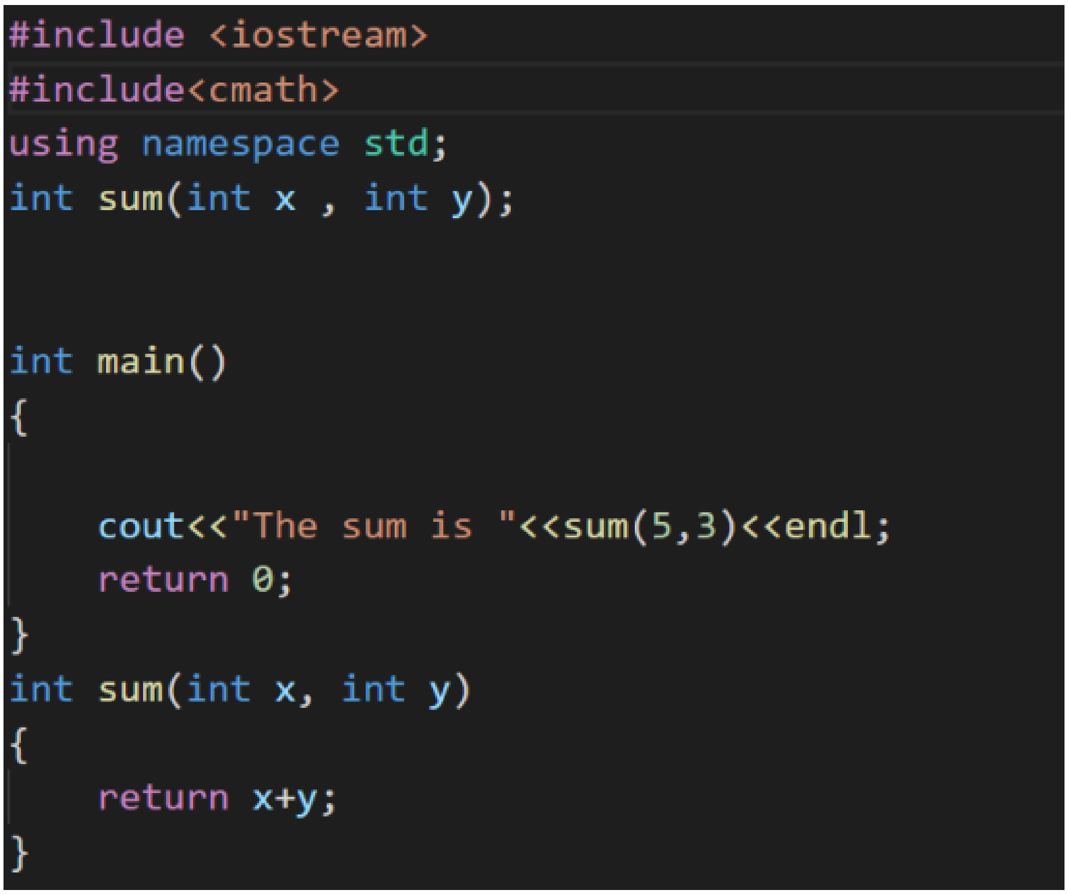
C++ is a general purpose programming language that was developed by Danish computer scientist Bjarne Stroustrup as an extension of the C programming language.It is an intermediate level language, as it comprises a confirmation of both high level and low level language features.

It was originally created as a successor to C with more features, hence the name ”C++”. It is an Object Oriented Programming language but is not purely Object Oriented. It is also sometimes referred to as ”C with classes” due to its Object Oriented features and machine level control.

7.1.2. Structure of Code

A C++ program is structured in a specific and particular manner. In C++, a program is divided into the following four sections:

1. Standard Libraries Section
2. Class Definition Section
3. Functions Definition Section
4. Main Function Section

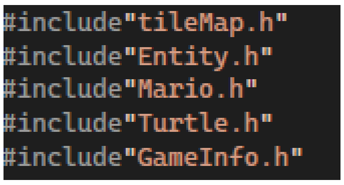


```
#include <iostream>
#include<cmath>
using namespace std;
int sum(int x , int y);


int main()
{

    cout<<"The sum is "<<sum(5,3)<<endl;
    return 0;
}
int sum(int x, int y)
{
    return x+y;
}
```

1. **Standard Libraries Section**

This section is written at the top of the program and consists of the first lines of code read by the compiler. This is also known as pre-processor section. This section is where all the header files and namespaces used in the program such as iostream and std. This includes standard libraries as well as user-defined header files and programs.

2. **Class definition Section**



```
#include"tileMap.h"
#include"Entity.h"
#include"Mario.h"
#include"Turtle.h"
#include"GameInfo.h"
```

The classes are used to map real world entities into programming. The classes are key building blocks of any object oriented program. A C++ program may include several class definitions.In our case, the classes are defined in separate header files that have been included in the Standard Libraries Section.

3. **Functions Definition Section**

Functions are a feature of Procedure Oriented Programming but are very useful in OOP as well. In the above program sum() is a function declared and used in the main function.

4. **Main Function Section**

main() function is the function called when any C++ program is run. The execution of all C++ programs begins with the main() function and also ends with it, regardless of where the function is actually located within the code.

7.1.3. Features of C++

C++ is a general-purpose programming language that was developed as an enhancement of the C language to include object oriented paradigm. It is an imperative and a compiled language. Some of its main features are:

1. **Namespace**: A namespace is a declarative region that provides a scope to the identifiers (the names of types, functions, variables, etc) inside it. Namespaces are used to organize code into logical groups and to prevent name collisions that can occur especially when the code base includes multiple libraries.

2. **Inheritance**: Inheritance is a process in which one object acquires some (or all) of the properties and behaviors of its parent object automatically. In such way, one can reuse, extend or modify the attributes and behaviors which are defined in other classes, from existing classes. The class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class. The syntax for derived class is:

3. There are three visibility modes in which a derived class inherits from its base class. They are private, public and protected. Following table clarifies the concept:

4. **Polymorphism**: The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. Polymorphism is considered as one of the important features of Object Oriented Programming.

5. In C++ polymorphism is mainly divided into two types:
(a) Compile time Polymorphism
(b) Runtime Polymorphism

6. **Templates**: A template is a simple and yet very powerful tool in C++. The simple idea is to pass data type as a parameter so that we don't need to write the same code for different data types. For example, a software company may need sort() for different data types. Rather than writing and maintaining the multiple codes, we can write one sort() and pass data type as a parameter.

7. C++ adds two new keywords to support templates: 'template' and 'type-name'. The second keyword can always be replaced by the keyword 'class'.

7.2. Simple Fast Media Library (SFML)

SFML (Simple and Fast Multimedia Library) is a powerful and user-friendly C++ multimedia library designed to aid developers in creating multimedia applications and games. With a focus on simplicity, efficiency, and cross-platform capabilities, SFML provides a wide range of features for handling graphics, audio, networking, and more, making it an ideal choice for both beginners and experienced programmers.

SFML abstracts the complexities of low-level multimedia programming, allowing developers to concentrate on creating engaging content without getting bogged down in platform-specific intricacies. Its modular design enables users to seamlessly integrate different components into their projects while maintaining a consistent programming interface.

Originally developed by Laurent Gomila, SFML has gained popularity within the game development community due to its ease of use and efficiency. It supports various platforms, including Windows, macOS, Linux, and more, making it suitable for creating applications that run on multiple operating systems.

Key features of SFML include:

- Graphics Rendering: 2D graphics rendering, shapes, textures, and custom shaders.
- Audio Handling: Sound effects, music playback, and audio recording.
- Input Management: Keyboard, mouse, and joystick input handling.
- Networking: Support for multiplayer games and networked applications.
- Window Management: Simplified window creation, event handling, and application loop.
- System Utilities: File and path manipulation, time management, threading.
- Cross-Platform: Consistent API across Windows, macOS, Linux, and more.

- C++ Library: Designed for C++ programming and user-friendly development.
- Open Source: Available under the zlib/png license, fostering community contribution.
- Community and Documentation: Active community, tutorials, and comprehensive documentation.

8. Existing System

Super Mario Bros was originally developed in 6502 assembly language, made to be played in the Nintendo Entertainment System (NES), an 8-bit third-generation home video game console. Since then, it has been recreated countless times on every platform.

As beginners to OOP and programming in general, we have made a simpler version of this popular game as a way to learn.

9. Methodology

Following steps were followed in the development of this project:

9.1. Resource Gathering and Planning:

In the very first stage, we brainstormed on the overall concept of the game, focusing on the gameplay, art style, and character and world design. We will then identify the required assets for the game including graphics, sound effects, music, and character animations. The scope of the game was defined where we decided on the features to be added, taking into account the time and resources available as well as our skill level.

After that we moved on to plan the technical aspects of the game. We gathered information on the aspects of SFML to be used in the code. The workload was then divided among the three members for a smoother workflow.

9.2. System Model and Design:

Based on the resources gathered, we designed a high-level representation of the overall code. We decided on the specifics of the game mechanisms and components. We planned the structure of the codebase, including organization of files and assets. Here, we also worked on how to implement modularization and object-oriented design in the game.

9.3. Software Development:

In this step, we coded the game logic, integrating artwork and audio assets, implementing user interface elements, and incorporating the required physics and animations. We wrote the code in Microsoft Visual Studio IDE using an object-oriented approach in C++ language. The compiler was MSVC for Windows OS. We used SFML (Simple and Fast Multimedia Library) for graphics and audio.

9.4. Testing and Implementation:

We first developed a Minimum Viable Product(MVP) sample of our project for testing and debugging. Further testing and debugging were done to add features and edit the project code as per our need and capability.

9.5. Deployment and Documentation:

Once the game was thoroughly tested and approved, it was deployed to the production environment. This will involve installing it on target machines and making it available for end-users. At this point, we will also created comprehensive technical documentation of our completed project.

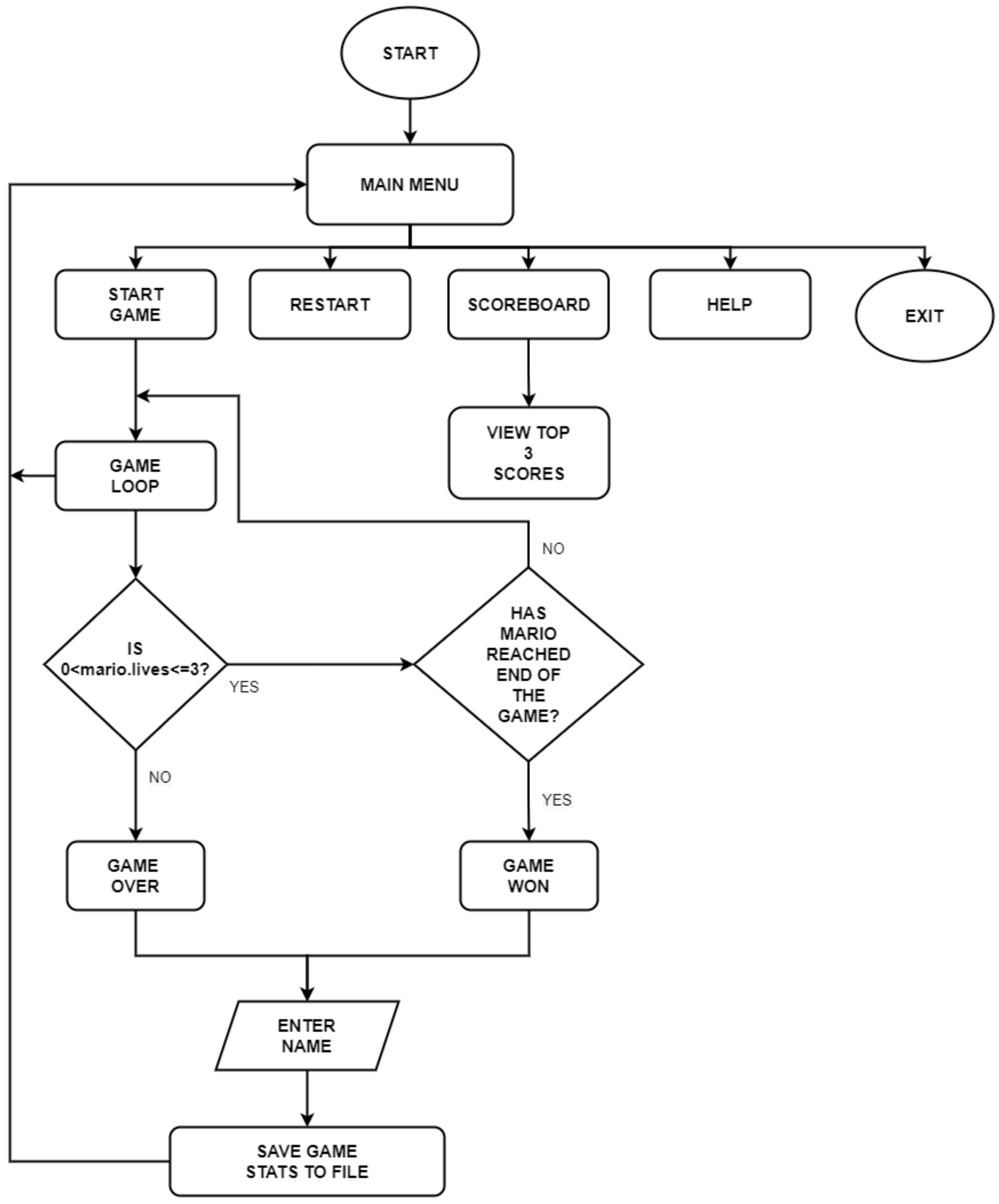
9.6. Maintenance and Support:

After this stage, the game will require ongoing maintenance and support. This includes addressing bugs, implementing enhancements, and providing technical support to users. Maintenance will involve iterative cycles of development to continuously improve the game based on user feedback and changing requirements. Also, monitoring changes in hardware, operating systems, and platforms to ensure that the game remains compatible with the latest technologies might be necessary.

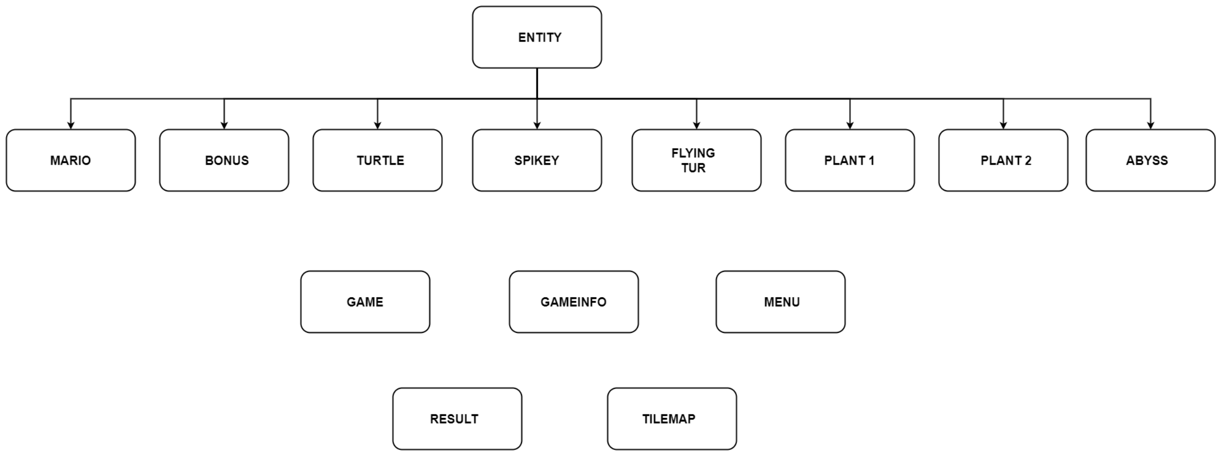
10. Implementation

10.1. Block Diagram

10.1.1 Game Flow



10.1.2 Code Structure



10.2. Main Menu

The game starts with a Main Menu, which has options to start and restart the game, view the top three scores, view the help window consisting of game instructions and also an exit button. The menu is controlled using keyboard inputs.

10.3. Game World

Pressing ‘Start Game’ begins the game, introducing the player to the titular character Mario. The game follows the basic principles of platformer games. There is a mechanism of side-scrolling, which means the game screen follows Mario when he goes left and right. On the top, the number of lives, score, and time elapsed are displayed.

10.4. Graphics and Sound Effects

The original Mario music plays in the background throughout the game. Different sound effects are also played when Mario jumps, gets hit, kills an enemy or reaches the end of the game.

10.5. Enemy Characters

Enemies are present throughout the game. Spikey, Normal Turtle and Flying Turtle move left and right, while the plants stay stationary. The turtles can be killed by Mario jumping on top of them.

10.6. Collectibles

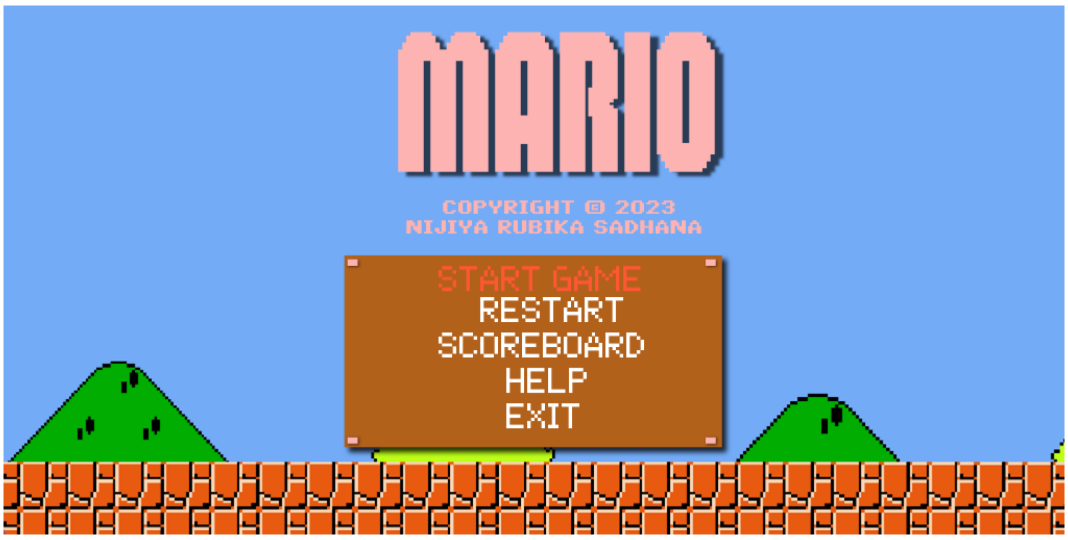
Mario can collect coins to increase score, or hit the question mark (?) brick to collect a Special Mushroom, changing Mario into Big Mario and gaining an additional life.

10.7. Scoreboard

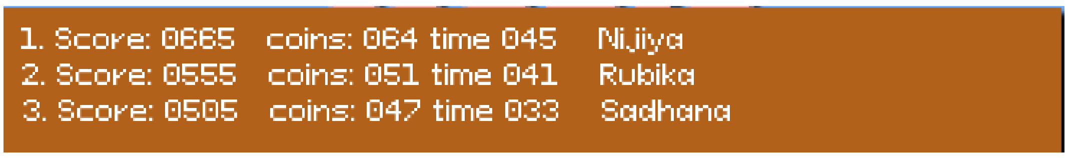
Losing all three lives or reaching the end of the game prompts the user to enter their name. If the score of the player is in the top three, their name is displayed in the scoreboard.

11. Results

After the final program was completed and bug-free, the result obtained was close to what was expected. The major objective, the application of OOP in C++, was fulfilled. The following screenshots from different states throughout the game illustrate the final result of the project:

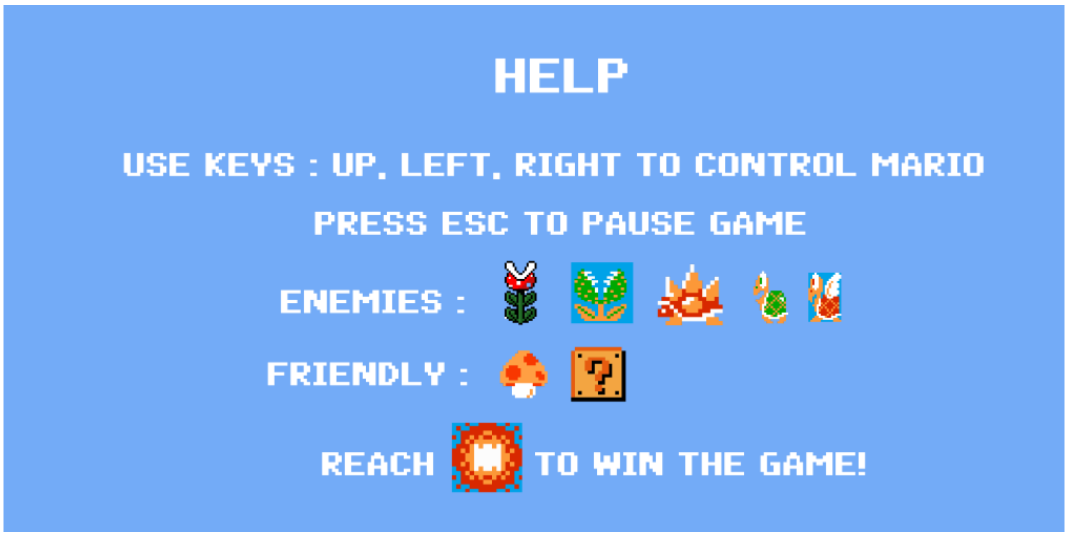


Main Menu



```
1. Score: 0665   coins: 064 time 045   Nijiya
2. Score: 0555   coins: 051 time 041   Rubika
3. Score: 0505   coins: 047 time 033   Sadhana
```

Scoreboard


HELP

USE KEYS : UP, LEFT, RIGHT TO CONTROL MARIO

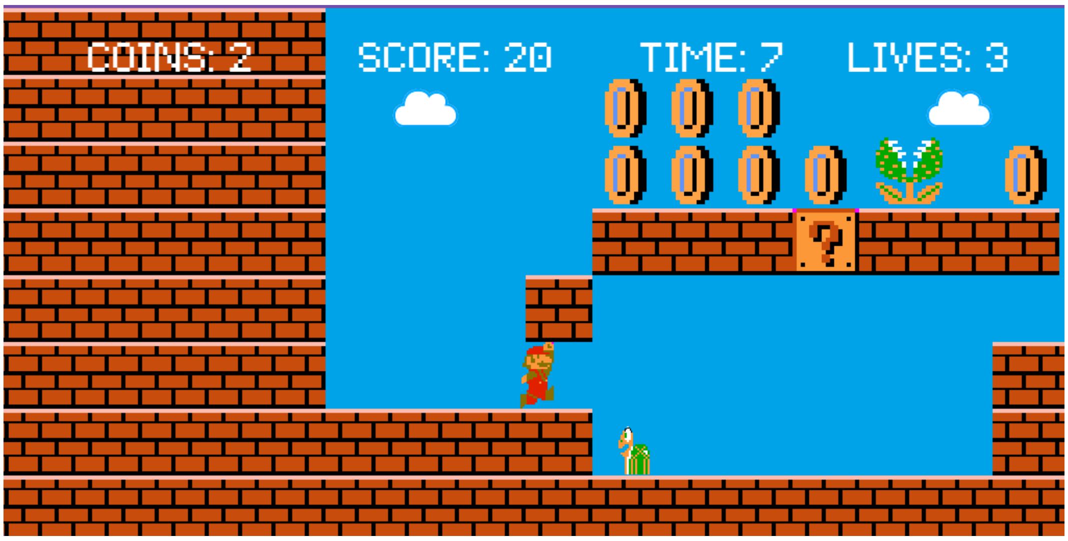
PRESS ESC TO PAUSE GAME

ENEMIES :

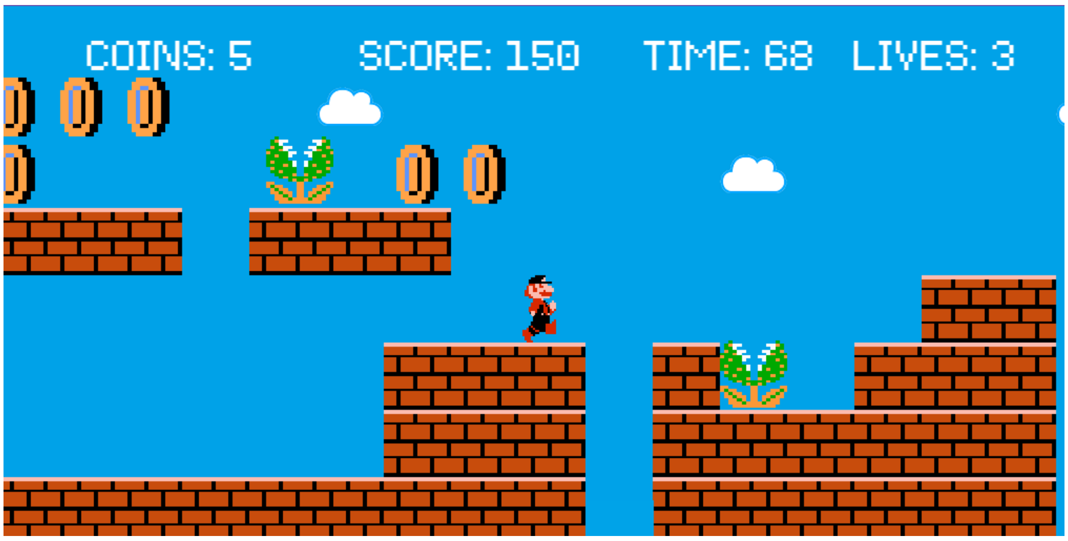
FRIENDLY :

REACH TO WIN THE GAME!

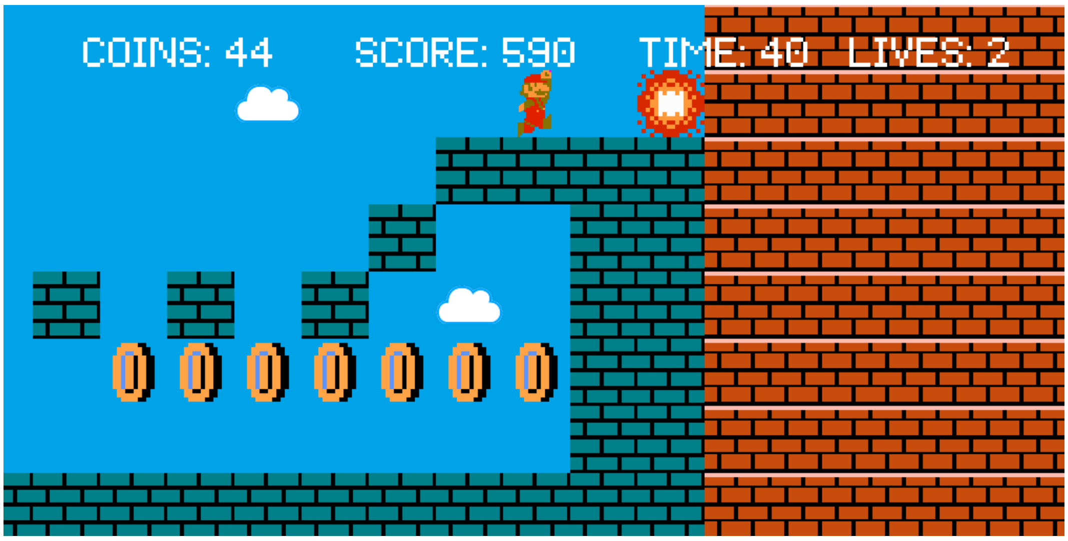
Help Menu



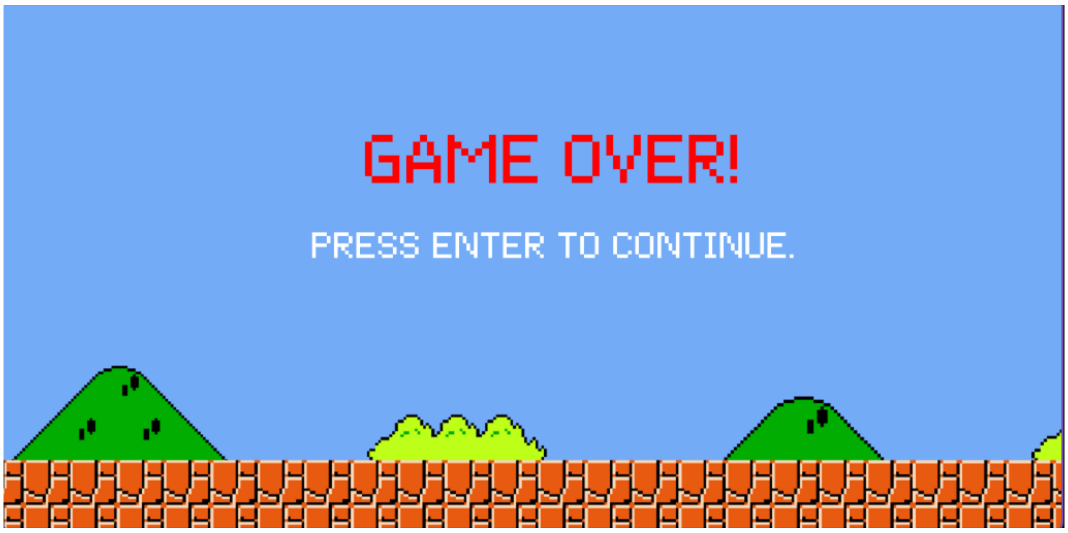
Gameplay



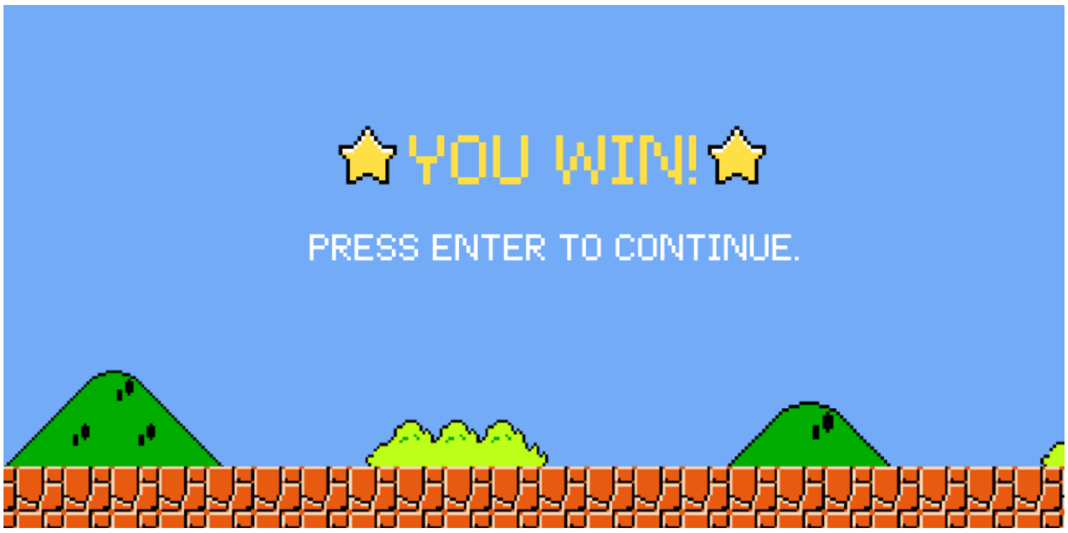
Big Mario, after collecting Special Mushroom



End of the game



Display after losing all three lives



Display after completing the level

12. Problems Faced and Solutions

1. **Setup :** The setup itself was quite challenging. We ran into several problems while installing SFML on our computers.

   We solved these issues by referring to the official SFML documentation, trying methods recommended in forums and YouTube videos and also consulting with our classmates

2. **Code partition** : While the use of Object Oriented programming helped us in dividing the code into manageable chunks, navigating between certain parts of the code and finding specific lines were slightly tedious due to the sheer volume of the code.

   We mitigated this problem by separating the class into header files. This helped us quickly view the member functions in any class. We also heavily relied on the search function of Visual Studio to track where certain functions and variables were used.

3. **Compiling Errors** : As this was our first time creating a large project, we had a hard time getting the application to compile successfully. The problems we encountered were primarily related to Mario's movement and interaction with various objects in the game.

   We added error handling and exceptions to root out the cause of the bug. Then we removed the bugs through trial and error and extensive research on the internet. We also referred our problems to AI-based chatbots Bard, Bing AI and ChatGPT; this was extremely helpful as they provided tailored advice to solve our bugs.

4. **User Experience :** The last problem faced was to make our game engaging and intuitive to play. We had to strike the perfect balance between easy to play and challenging, make the controls intuitive and smooth. Additionally, we had to make the game design attractive and the graphics pleasing.

Throughout the process of coding, we played the original Mario ourselves to understand the look and feel of the game. After achieving a MVP (Minimum Viable Product), we had other people play the game. Through this, we gained user feedback and improved different elements of the game.

13. Limitations and Future Enhancements

Our version of Mario is nowhere near perfect. Due to time constraints as well as our beginner skill level in programming, our game contains several limitations. Some limitations are :

- The game has only one level.
- We have added only one type of power-up.
- There is no animation when Mario jumps or dies.
- The game supports only one screen size.

In the future, we intend to improve the game in the following ways:

- Better animation
- Different power-ups
- Different levels consisting of different challenges
- Dynamic screen sizing for the game to work in multiple screen sizes.

14. Conclusion

In reflection, we have successfully attained our project's objectives, resulting in the creation of a fully functional Mario game. By applying the fundamental principles of Object-Oriented Programming (OOP), we thoroughly understood how to translate the theoretical concepts we learnt in the classroom into practical application.

The project journey allowed us to delve into the intricate realm of software development. We saw how creating an application not only requires programming skills but also patience, perseverance and a touch of creativity.

One remarkable aspect of this endeavor was the strength of our teamwork and collaboration. Our effective communication, harmonious coordination, and shared efforts were indispensable in reaching our project milestones. Working collaboratively not only polished our technical proficiencies but also emphasized the significance of interpersonal dynamics in accomplishing collective goals.

Our familiarity with version control tools, particularly Git, and platforms like GitHub, played a pivotal role in orchestrating a seamless workflow. Through these tools, we managed code revisions, tracked progress, and deftly resolved conflicts, showcasing the pivotal role of organized version control in project success and cohesive teamwork.

Ultimately, our dedication and proficiency harmonized to yield a delightful Mario game. The successful amalgamation of OOP principles, external libraries, teamwork, and version control expertise not only realized our outlined objectives but also furnished us with an invaluable learning experience. The skills we honed throughout this project are bound to serve us well in future software development endeavors and various other domains.

15. References

We referred to various books and resources for understanding the concepts behind game development using C++ and SFML. Our resources are listed as below:

Book References :

1. 'Beginning C++ Game Programming, 2nd Edition' by John Horton
2. 'Secrets of Object-Oriented Programming in C++' by Daya Sagar Baral, Diwakar Baral
3. 'SFML Game Development by Example' - Raimondas Pupius
4. 'Programming Principles and Practice Using C++ (2014)' - Bjarne Stroustrup
5. 'SFML Game development' – Jan Haller

Web References :

1. https://en.sfml-dev.org/forums/
2. https://stackoverflow.com/
3. https://www.sfml-dev.org/documentation/2.6.0/