TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A PROJECT REPORT ON
OBJECT ORIENTED PROGRAMMING WITH C++

KNOWLEDGEKICKS

Submitted by:
BHRAMAN SHRESTHA (078BCT027)
JANARDAN BHETWAL (078BCT044)

SUBMITTED TO:
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

August 25, 2023

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our OOP lecturer Mr.Daya Sagar Baral for his guidance and support throughout the completion of this project. His insights and encouragement has been invaluable in developing our skills and understanding object oriented programming.

We express our heartfelt appreciation and acknowledgement to the esteemed faculty members of our institution for their unwavering guidance, mentorship, and encouragement throughout this journey. We would also like to thank the Department of Electronics and Computer Engineering, Pulchowk Campus for providing us the opportunity to develop our own project which will further help in understanding object oriented programming.

We are grateful to our friends for the help and feedback they provided whenever we needed. We would not have done this without the constant support and encouragement from our family, so we owe our deepest gratitude to our family.

Authors:
Bhraman Shrestha
Janardan Bhetwal

ABSTRACT

This project submitted to the Department of Electronics and Computer Engineering in the Second Year First Part of the Bachelors in Computer Engineering aims at the application of Object Oriented Programming.

Our aim in this project was to develop a simple user friendly program using Object Oriented Programming in C++. We made a project named 'KNOWLEDGEKICKS' which is a simple Quiz game. This project has the visually appealing graphics which is achieved through SFML. By combining the object oriented capabilities of C++ and SFML graphical features, this project serves as an example of how a theoretical program can be turned into practical along with enjoyable applications.

We have used Windows Operating System, Microsoft Visual Studio as the IDE to write the code and GNU GCC compiler. Though the program is still functioning, we plan to improve and add other features to make it more entertaining in the future.

Contents

1 OBJECTIVES

The major objectives of this project are:

i. To learn and understand the application of object oriented programming using C++.

ii. To develop problem solving skills.

iii. To learn the basics of game development.

iv. To build an attractive UI to interact with the users easily.

v. To learn to work and communicate with the team.

vi. To be prepared to work on a major project in the coming years.

2 INTRODUCTION

In the realm of modern education and entertainment, interactive digital applications can become a powerful medium to help in learning. Quiz game is one of the widely popular applications which serves as a medium of learning with entertainment. It not only challenges the player's knowledge but also enhances their understanding in various subjects.

Like every quiz game, every question has four options. Only one of them is correct. The player should choose one of the options and based on the input, the program decides whether the answer is correct or not. The set of questions can be selected through the genre. Every player gets only 10 questions. Every correct answer increases the score of the player. After the 10th question, the added score is the total score of the player. If the total score is the highest it displays in the high score section.

3 APPLICATION

This Quiz project is similar to the real life quiz competition and has a wide range of applications. Similar to the real life Quiz competition, this application also serves as the learning platform. Not only for the students, but also for the teachers to enhance the knowledge as well as gain some information, this project serves the best way. Not limited to this, it can be modified to make an exam preparation MCQs application as well. In conclusion, this project provides a reliable place to play a quiz game.

4 LITERATURE SURVEY

4.1 C++

C++ is an object oriented programming language developed by Bjarne Stroustrup at Bell Labs as an extension to C language. C++ is known for its flexibility and efficiency, making it suitable for a variety of applications such as software development, game development, system programming and many more.

Our main reference for C++ was "The Secrets of Object Oriented Programming using C++" by Daya Sagar Baral.

4.1.1 Structure of Program

A general c++ program includes the following parts:

- Standard Libraries Section
- Class Definition Section
- Functions Definition Section
- Main Function Section

Standard Libraries Section: This section is written at the top of the program. 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.

Class Definition Section: This section includes several class definitions, which are key building blocks of any object oriented program. We defined every class in separate header files.

Functions Definition Section: This section includes definition of functions used in the program which may be used in other parts of the project.

Main Function Section: The main function is the main body of the program which the compiler executes first after all preprocessing.

4.2 SFML

SFML(Simple and Fast Multimedia Library) is an open-source C++ library designed to simplify the process of creating multimedia applications and games. SFML provides a wide range of facilities for graphics, audio, networking, and window management, making it an excellent choice for developers looking to build interactive and visually appealing applications.

Our main reference for SFML was "SFML Game Development" by Jan Haller and

the documentation provided by the official SFML website.

Furthermore, various youtube videos and websites such as stackoverflow.com helped us with C++ and SFML.

5 EXISTING SYSTEM

This type of game is not a new concept as we can see various such games across multiple platforms. They may be tailored to entertainment or to expand the knowledge of the players. Some of the applications are specifically tailored to help students prepare for exams. A lot of quiz game apps have multiplayer support allowing people to compete with their friends. They also have a large number of questions and multiple 'game modes'. On the other hand there are also a lot of applications which simply ask questions and give the answers. Since our aim is to learn through this project, we chose this topic to make a simpler version with some of our modifications.
As far as our choice goes, we have chosen this specific project as we feel that building an application like a quiz game can give us the best opportunity to learn as such applications can be scaled from simple to quite complex.

6 METHODOLOGY

For this project completion we roughly followed the following methods:

6.1 Choosing the topic and Graphics Library

After forming the team, we started researching some of the interesting projects which can be done using C++. After some research, we decided to choose a Quiz game as our project. The next decision we had to make was to choose the appropriate library for the graphics. We found some libraries like SDL, WxWidgets, SFML, Qt, etc but ultimately chose SFML as our library.

6.2 Initiating and Planning

After choosing the topic and library, we started planning the making of the project and divided the work among the team members.

6.3 Project Proposal

Before coding, we made the basic framework for the project and documented that information on our proposal.

6.4 Algorithm Design

We then made the basic algorithm and flowchart of this project. Considering the validity and time limitation we decided the features to be included in this project.

6.5 Learning C++ and Libraries

After algorithm design, we started refreshing our C++ knowledge and learning C++ libraries. Learning SFML was time consuming as we had no prior knowledge about it.

6.6 Software Design

After our proposal was accepted, we started coding the main logic of this project. We divided the work among two of us into implementing questions/answers and developing graphics.

6.7 Debugging and Testing

After making the minimal Viable Product(MVP), we started testing and debugging. We then added different features in the game and made the project more attractive and interactive. We were constantly finding and removing bugs in our program throughout this process.

6.8 Documentation

After the completion of the project, we summarized all the process and works done during the project and started making the final report.

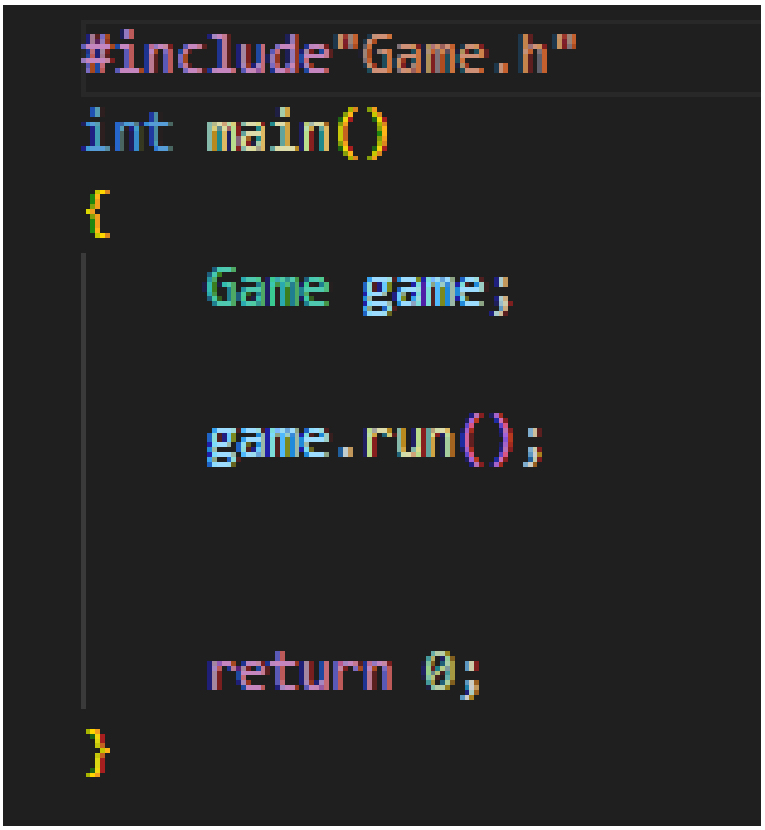
7 OBJECT ORIENTED APPROACH

Since, the main aim of our project is the application of Object Oriented Programming, we tried our best to developed this project with object oriented approach.

7.1 Abstraction

Abstraction refers to the concept of simplifying complex reality by modeling classes of objects or systems with essential characteristics and behavior while ignoring or hiding unnecessary details. It allows us to focus on the essential aspects of an entity, making it easier to understand, analyze, and work with.

For instance, we have created a Game class and made it available in the main program. Now we can create a game object and manipulate as our need without knowing its implementation.



Similarly, we have created a button class and made it available in different states. We can create the button objects in any state and use its operation. This helps in managing complexity as well as memory management.

7.2 Encapsulation

Encapsulation refers to the practice of binding data (attributes) and the methods (functions) that operate on that data into a single unit called a class. The key prin-

ciple of encapsulation is to hide the internal details of an object's implementation from the outside world while providing a well-defined interface for interacting with the object.

```
class GenreState :
    public  State
{
private:
    sf::Font font;
    std::map<std::string, Button*> buttons;
    sf::Text text1;
    sf::Text text2;
    std::fstream file;

    void initFonts();

public:
    GenreState(sf::RenderWindow* window, std::stack<State*>* states);
    virtual ~GenreState();

    //functions
    void initButtons();
    void updateButtons();
    void renderButtons(sf::RenderTarget* target);
    void initText();
    void renderText(sf::RenderTarget *target);
    void endState();
    void updateInput(const float& dt);
    void update(const float& dt);
    void render(sf::RenderTarget* target = NULL);
};
```

In this project, we have created many classes. Each class has its own data members and funtions. It helps in data protection, abstraction, access control and code reusability.

7.3 Inheritance

is a mechanism that allows you to create a new class based on an existing class, inheriting its attributes (data members) and methods (member functions). By allowing one class to inherit the attributes and methods of another, it promotes code reusability, reducing redundancy and accelerating development. Inheritance also facilitates the creation of a clear and hierarchical class structure, enhancing code organization and modularity.. Moreover, inheritance supports polymorphism, enabling derived classes to override inherited methods for tailored behavior.

```
class MainMenuState :
    public State
{
private:


    sf::RectangleShape background;
    sf::Font font;

    std::map<std::string, Button*> buttons;
```

In this project, we have inherited the features of State class (base class) into other different classes such as MainMenu class (derived class). Every members of base class is inherited into derived class but only protected and public members are visible in derived class.

7.4 Polymorphism

Polymorphism is a concept in OOP where objects of different classes are treated as objects of a common base class, allowing the use of common interfaces to interact with diverse types of objects. It allows you to write code that can work with objects of different classes in a unified way, regardless of their specific types.

Pure virtual fuctions is a virtual function that only has a declaration but doesn't have a definition. Since they have no definition, these functions cannot be called and object consisting of pure virtual function cannot be created. It's usefulness comes from the fact that any class that derives from a base class consisting of a pure virtual function must implement the function for the derived class.
An abstract class is a class that cannot be instantiated on its own and is meant to serve as a blueprint for other classes. In other words, an abstract class is a class that has at least one pure virtual function.

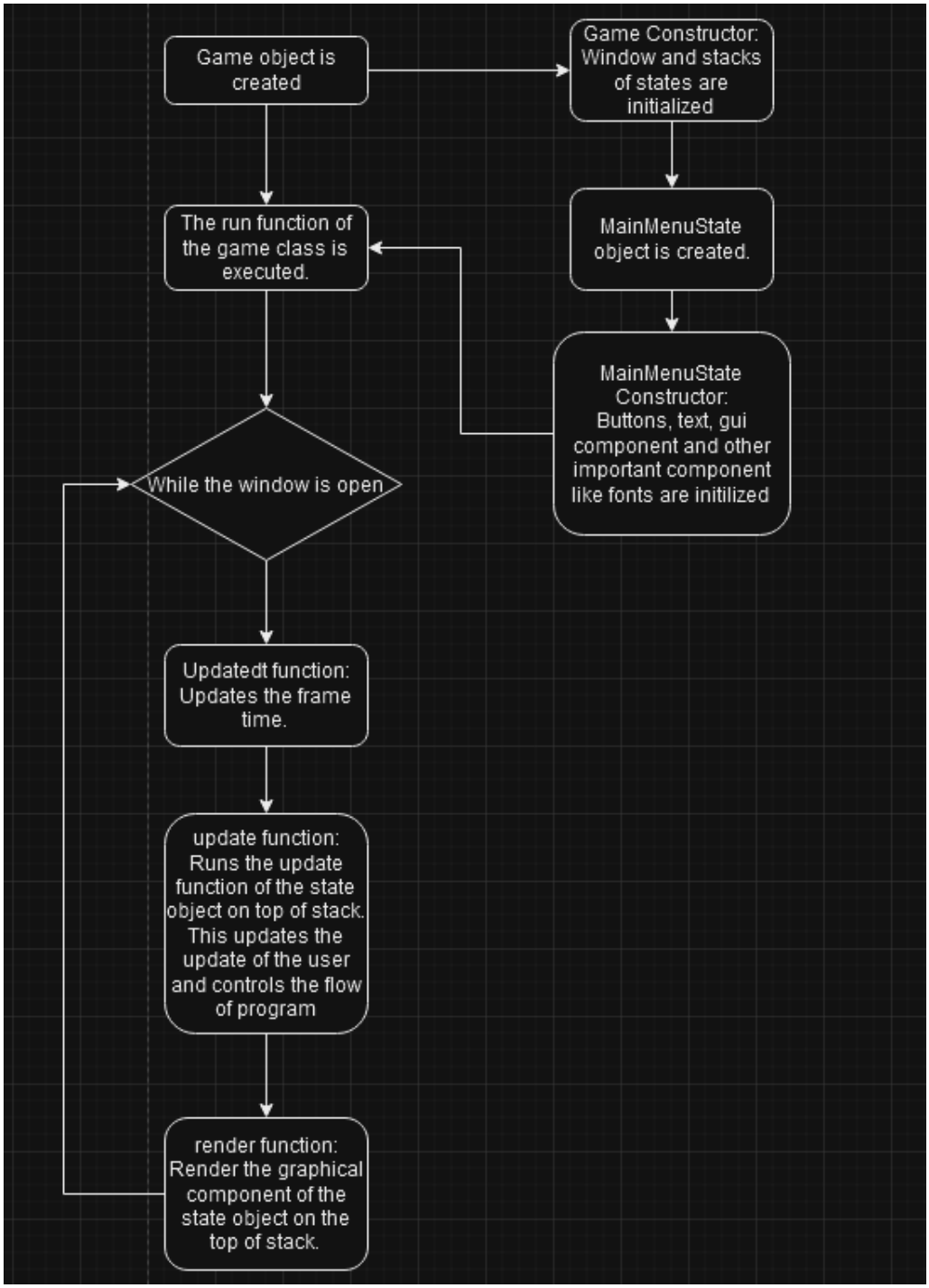
```
virtual void updateInput(const float & dt) = 0;
virtual void update(const float &dt) = 0;
virtual void render(sf::RenderTarget* target = NULL) = 0;
```

We have created a state class that acts as an abstract class as it has pure virtual functions. It provides common interface for its derived classes : MainMenuState class, GenreState class, etc.

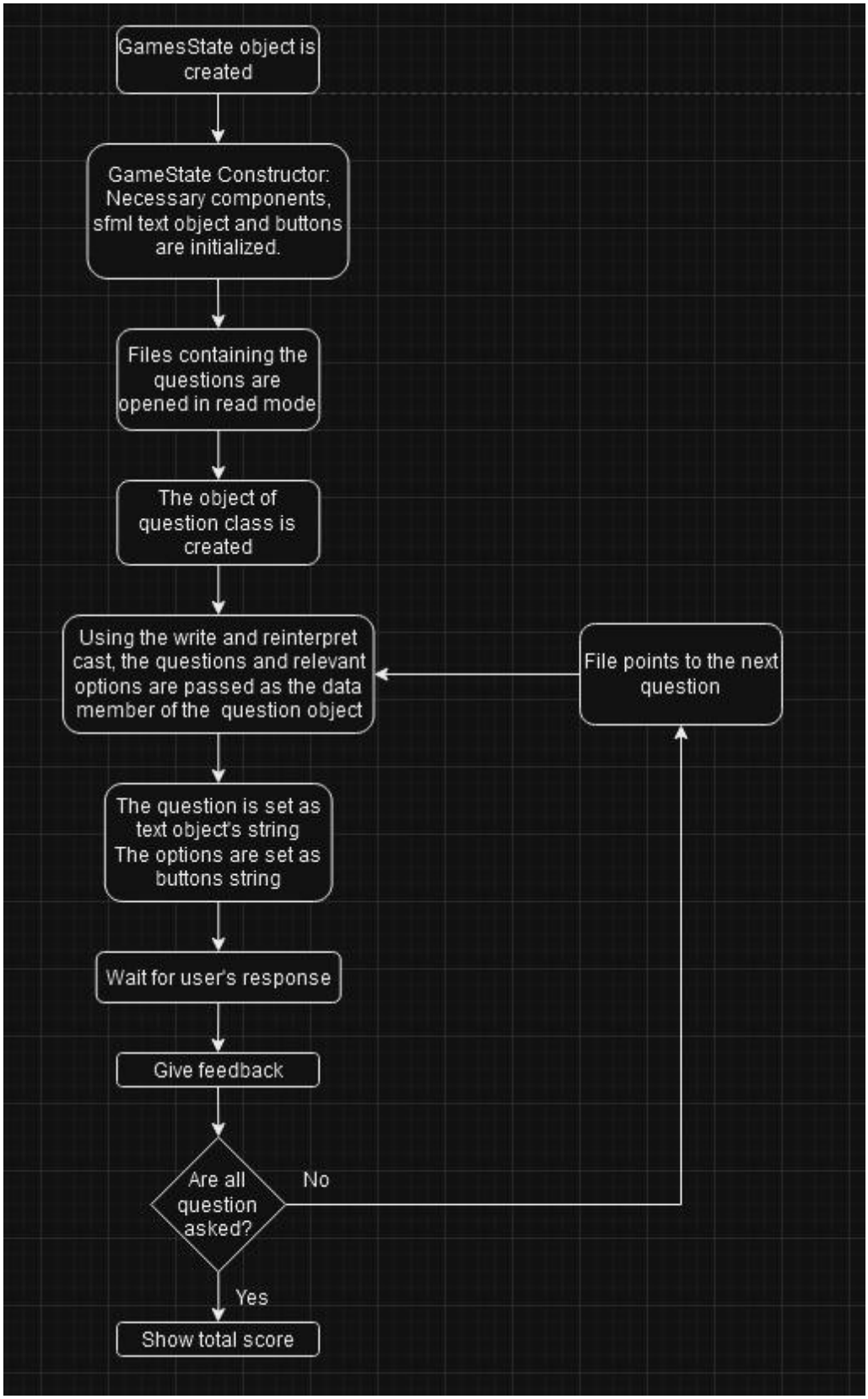
8 IMPLEMENTATION

We began this project with the idea of the application of object oriented programming.

8.1 BLOCK DIAGRAM



Block Diagram 1



Block Diagram 2

8.2 MAIN MENU

The game starts with the main menu. Main menu contains four options: play game, select genre, high score, and exit. The player has to choose one of the options by clicking on the respective button.

8.2.1 Play Game

This option starts the game. A question with its four options displays on the screen when the player choose this button. After choosing one option, new screen appears in the screen which shows whether the option selected is correct or not. The continue button, on clicking, displays another question and its four options. Every player is asked 10 questions. The total score achieved by the player displays after the 10th question.

8.2.2 Select Genre

This option enables the player to choose the genre. Every genre contains different set of questions. The player can choose any one genre to play the game related to particular topics.

8.2.3 High Score

After the game, the total score of the game is matched with the high score of that genre. If the total score exceeds high score, then the total score is recorded as the new high score. So, when this option is selected, it displays the highest score for each genre.

8.2.4 Exit

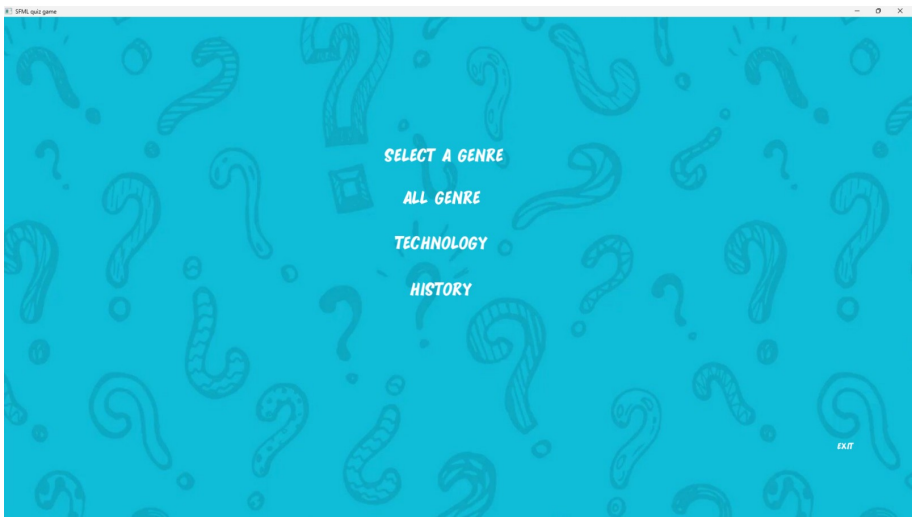
After playing the game, the player can play again or exit the game. The exit operation is performed by this option.

9 RESULTS

After the completion of the project using object oriented programming using C++, our goal is achieved. Though we can add many more features and make it more attractive and entertaining in the future, it is still functioning and fun to play. Here are the obtained results from the project.

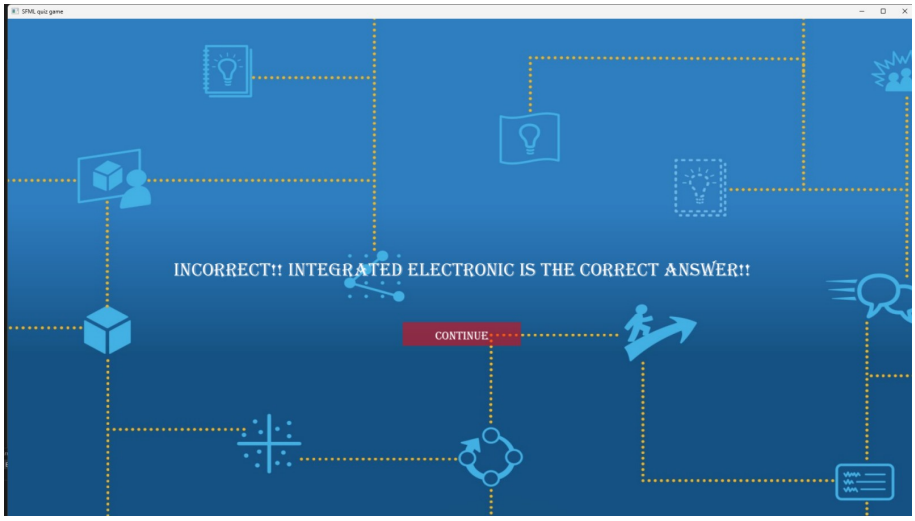
9.1 Main Menu

9.2 Select Genre

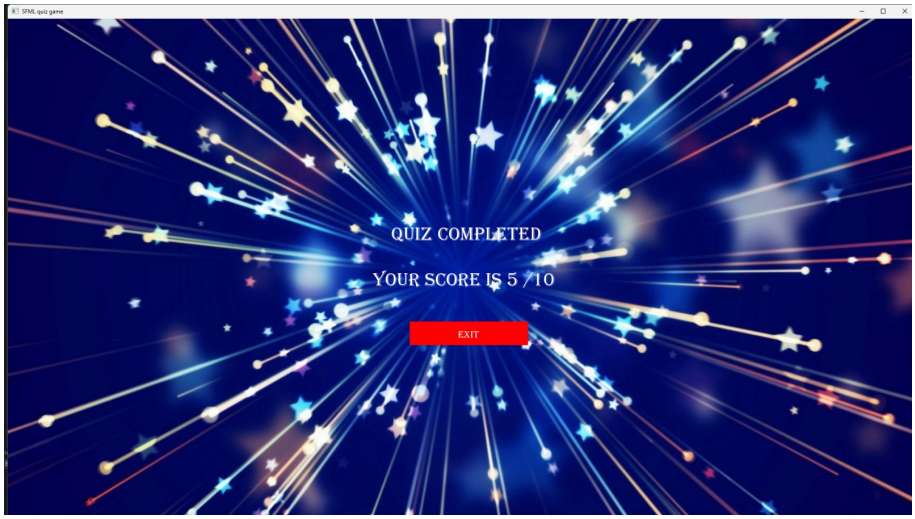


9.3 Play Game

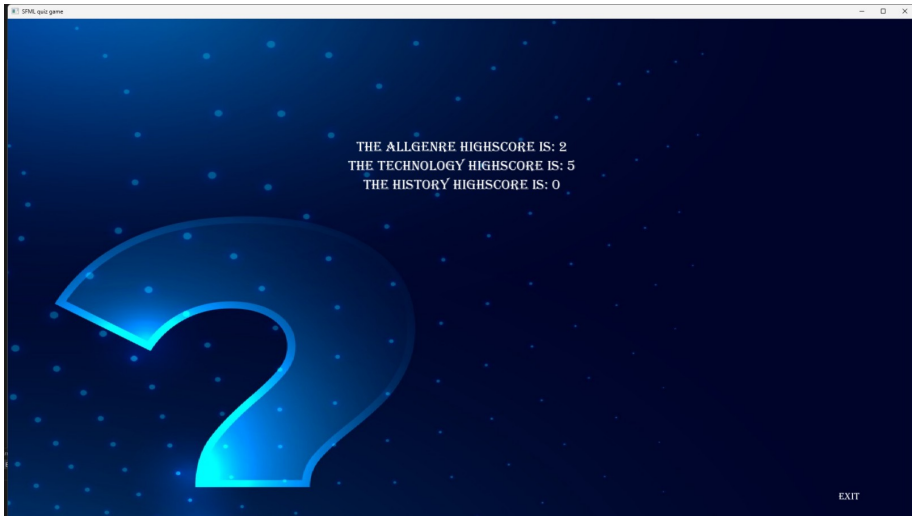
9.4 Feedback



9.5 Total Score



9.6 High Score



Due to lack of enough time, we couldn't include some of the important features like Hint options, Coins collected, etc. We couldn't add many types of Genres as well. This project can be modified into much better form.
From this project, we learnt about the C++, its applications and its object oriented approach, we learnt about the basics of game development and SFML as well. We learnt to collaborate with a team and its value which will be helpful in our future.

10 PROBLEM FACED AND SOLUTIONS

i. **Problem with navigation**
The problem we faced at the beginning of the project was the problem with navigation. We were rendering certain sprites on the screen and to navigate we had to stop rendering those things, and render others. This was tedious. So,

we solved the problem by having multiple state classes. Each class rendered specific things on the screen depending on its state and we simply jumped through different states to navigate through the program

ii. **Problem with communication between the states.**

a. As the game runs the user is shown multiple screens to ask questions and to give feedback. This meant that the essential datum from the previous state could be lost. So a problem we faced was that only one question was asked again and again and the program did not move onto the next question.

b. Also, the genre is selected in the Genre state, while the game is played on the GameState. Which meant that the Genre state and the Game state had to communicate which genre was selected. Using fstream objects as data members to communicate, or using static members to communicate was very tedious, prone to error and memory leaks.

The problems were mainly solved using file handling. To make the states communicate with each other, the essential data were saved in a file before the next state was called. Then the new state will always extract the essential datum from the file before other codes are executed.

iii. **Final state glitch** After all the questions had been asked and we needed to report how much the gamer scored, we were not moving on to the final state. Multiple exceptions were thrown and instead of moving to the final state, the last question was asked again and again.

This, although a trivial problem, was hauntingly a difficult problem to solve mainly due to our incompetency in writing organized code. The problem was solved by writing code concisely and organizing the code properly.

iv. **Code organization and cleaning up messy code** We utmostly used the c++ abstraction feature to clean up our code and reduce mess. We created multiple functions based on the required tasks as well as multiple classes.

11 LIMITATIONS AND FUTURE ENHANCEMENTS

We worked on this project to make it as good as possible. Though this project is functioning and can provide good experience, it has many limitations which are listed below:

- The player cannot choose the difficulty level of the game.
- No hints and lifeline features in expense to the coins collected to help the player.
- No separate profile for each player so cannot record each player's achievements.
- No multiplayer mode to play 1 vs 1 or as a team online.
- Lack of proper graphical interface.

There are many possible enhancements for this project in the future.
Some of them are:

- Implementing difficulty level helps the player to play the game according to their difficulty. This also challenges the player to improve their difficulty level.
- Using the coin system that increases with the score helps in the use of hints and lifelines.
- Multiplayer mode. This allows two or more players to team up and play against each other online via networking.
- Use of attractive interfaces and animations could enhance the user experience.

12 CONCLUSION

In this way, we completed our project. This project was based on the application of object oriented programming in C++. This project has been developed using most of the features of OOP.

This project has been a great learning experience. We learned how powerful the object oriented paradigm is, with the help of key concepts like inheritance, polymorphism, encapsulation and abstraction. We learned about the application of OOP, features of C++, basics of game development and Simple and Fast Multimedia Library (SFML). We learned about the collaboration and teamwork from this project.

This project teaches us to deal with the development cycle, including planning, problem solving, analysing, debugging and testing. Overall we learned how to apply object oriented approach of C++ to develop a project in efficient manner.

13 REFERENCES

13.1 Book References

“The Secrets of Object oriented Programming”, Daya Sagar Baral, Diwakar Baral

“The C++ Programming Language”, Bjarne Stroustrup

“C++ How To Program” , Paul Deitel, Harvey Deitel

“SFML Game Development”, Jan Haller, HenrikVogelius Hansson, Et al

13.2 Web References

https://www.sfml-dev.org

SFML Tutorials by Suraj Sharma