Acknowledgement:

It’s a great opportunity for us to be given a chance to work on self-motivated graphics-based Object-Oriented Programming project. Since practical projects act as a reflecting mediator of the theoretical knowledge we have gained and the extent to which we can implement in the real life platform, the successful accomplishment of this project is assumed to help us grow in terms of game development basics, team work and understanding core concepts of pointers, arrays as well as the SFML library.

We would like to thank and show immense gratitude towards **Mr. Daya Sagar Baral** (Department Of Electronics and Computer Engineering, IOE Pulchowk), our respected mentor and lecturer, who has always been helpful and supportive in giving the core concepts on Object Oriented Programming, guiding and motivating us through for this project.

We would also like to show deepest gratitude to our Department and our lab teachers for carving our way through. We would also like to appreciate the help from our friends who made the idea a possibility by helping us through constructive criticism on the proposed idea and the explaining the possibility with SFML library.

This report is supposed to give a basic insight to our 2D game project that is going to be written based on SFML library. We as a team of 3 people fully dedicated ourselves into this project and with proper guidance from our mentors, we ae ready to go all out on meeting what our proposal demanded on the final Project.

Abstract

This project work is a minor project for our academic session B.E.(Computer) Second Year First part as prescribed in the syllabus designed by IOE. The main aim of the project was to develop a program that would provide student the knowledge of object-oriented programming and its importance for the development of software. For this project we have made a 2D game in which the player combats in a hand battle with the CPU (in Campaign mode) or with another player (Multiplayer). The CPU gets tougher to fight as the level progresses. With each hit the health of the opposition decreases and game ends when the health is fully drained. We have added background environment, story and appropriate sounds/music to bring an enthusiastic vibe to the game.

Table of Contents

Objectives

The main objectives to be met with this project can be summarized as follows:

- To clearly define and implement the OOP programming principles as a part of your code.
- To be familiar with resource reusability by using multiply linked files and inheritable classes.
- To learn the basics of game development, particularly world building,
- character interaction and implementation of real-world scenarios as a part of your game logic.
- To understand the use of a graphics library, particularly in the implementation of SFML Modules to create a 2-D game.
- To reuse the resources within the program and use various low-level manipulation techniques (i.e. pointers, references) to create a small sized, optimized and fast application.

Introduction

The Iron Throne

This is the 2d animated game written in C++ using OOP concepts. We used SFML library for multimedia. The basic mechanics of the game are:

- The player uses controls to move and hit the opponent.
- The opposition hits the player with certain frequency which increases with levels.
- The health of the player drains when hit by the opponent.
- The game ends when the health bar is fully depleted.

The main features of the game are:

Main menu: The player can select to play, look at the options menu for controls or quit.

Single Player mode: In this mode the player's character is automatically selected to Jin the protagonist of the story who seeks to save the princess from Vader's detention. The player will battle with three opponents Hulk, Kazuya and finally Vader in a best of three combat battle. As level progresses the opponent becomes harder to fight. The three battles take place in three different locations with different arena music playing in the background. The game reverts back to main menu when 'escaped'.

Multiplayer mode: In this mode, the players select from the four characters mentioned above and battle against one another. Then they are allowed to select the arena in which they wish to fight. The input from each player is given through the same keyboard and different buttons.

Health bar: The player's and opponent's health are viewed as a green bar in the screen. As the players take hit the green bar turns red and finally when all health is depleted the one with remaining health wins.

Characters: The players are allowed to select any of four characters in multiplayer mode. This feature is not available in Campaign mode as the players must progress according to the story.

Arena: The players have three arenas to select from in multiplayer mode. The arenas cannot be selected for campaign as the player progresses according to the story-line.

Application

A fascinating statistic we found was that 2d arcade game have a huge prospect of market in the gaming industry. Tekken 3 a 2d fighting game 1.11 million copies and made 48.5 million dollars in revenue alone in the year 1998. The latest version of the game has sold 2.5 million copies. Being said that other applications of our game are listed below:

- The Iron Throne is a realization of a 2-D combat game which has large prospects in gaming industry. For an addictive easy to grab entertaining gaming environment.
- Recreational graphics and musical background
- Multiple mode available for dual as well as single player
- Story for content seeking gamers
- may not be competitor to modern games but guarantees nostalgia to early fighting game players

Literature Survey

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

‘**Secrects of Object-Oriented Programming in C++**’ - Daya Sagar Baral, Diwakar Baral

‘**SFML Game Development by Example**’ - Raimondas Pupius

‘**Programming Principles and Practice Using C++ (2014)**’ - Bjarne Stroustrup

‘SFML Game development’ – Jan Haller

‘Beginning C++ Game Programming’ -John Hutton

We also surfed through SFML forums and documentation to better understand the implementation of SFML in our game.

Existing Systems

There exist games like this before. Most famous are Tekken series, Mortal Combat and Street fighter. Our characters are also based on Tekken and voices are adopted from original mortal combat announcer. Game consists of two players going head to head in hand combat. The basic mechanics of the game are that the two players engaged in battle hit each other with their fists or feet and drain the other player's health. When the health of the player is completely depleted, the other player wins.

So, in a nutshell, the players hit each other and deplete health of the opposition and finally win the game.

Methodology

This project is based on C++ programming language utilizing SFML graphics library, and "Object Oriented Programming" concept. The private access specifier provided us with the power of data hiding and abstraction, thus preventing accidental changes and code reusability among the team members. The events for each object were handled by the different member functions and that they formed a single unit, capable of running all the task required on its own. C++ has the capability to manage the memory allocation/deallocation as well as memory manipulation through references and pointers on any objects that we've created which can increase the performance of our game. Game is performance critical software that requires 100% usage of the hardware user has, and C++ is only popular language that gives you such abilities:

- Efficient abstraction and data hiding from the user.
- Optimized encapsulation of all the data members and functions required to operate on them in a single unit.
- Deterministic and memory/address level control of the resources you use.

The strength of C++ when it comes to game development is the ability to exactly layout the data-structures that your software will use. C++ provides the ability to override important performance bottlenecks such as memory allocation. **C++ is a high-level language that lets you get close to the hardware.** It has the ability to structure and place things exactly where they want in the memory. C++ has a variety of compilers that create binaries specific to the platform directly without the need to first be compiled into an intermediate language then some form of virtual machine thereafter which significantly slows down code. Code-Blocks was chosen as our preferred IDE, which uses GCC compiler. However, C++ graphics would not fulfill the graphical and design objectives of our project having many graphical interfaces and objects to draw. For development of the game, we used SFML which has high resolution which can definitely fulfill the objective of our project. So, we started learning SFML through various sites and e-books as mentioned above. The latest, stable version of SFML library i.e. version 2.5.1 is thus used as the core graphics manipulating library of our project.

Classes Used in the Program

1. **Game**
   It is the class in which the main game loop occurs. All components in our game are drawn in this loop. The member function of game class Run () is the main loop of our program. The players, opponents, textures, fonts are loaded in the constructor of this class. The sprites are updated constantly as the loop runs. All the states loaded in different phases of the game are defined and controlled in this class.

2. **Player**
   In this case all the actions of the players i.e. the player and the CPU are defined in this class. This class includes the Anime class and controls the flow of animation of the players and the opponent CPU. The collision detection between the sprites is detected in this class. The health of the players is also defined in this class.

3. **Anime**

   This class defines data members and functions to animate a certain row of sprite which is required in case of certain button pressed by the user or certain action by the CPU. E.g. when 'I' is pressed during gameplay assigned for punch, the row of sprite which contains punching stance is selected animated over the stance pictures to produce animation.

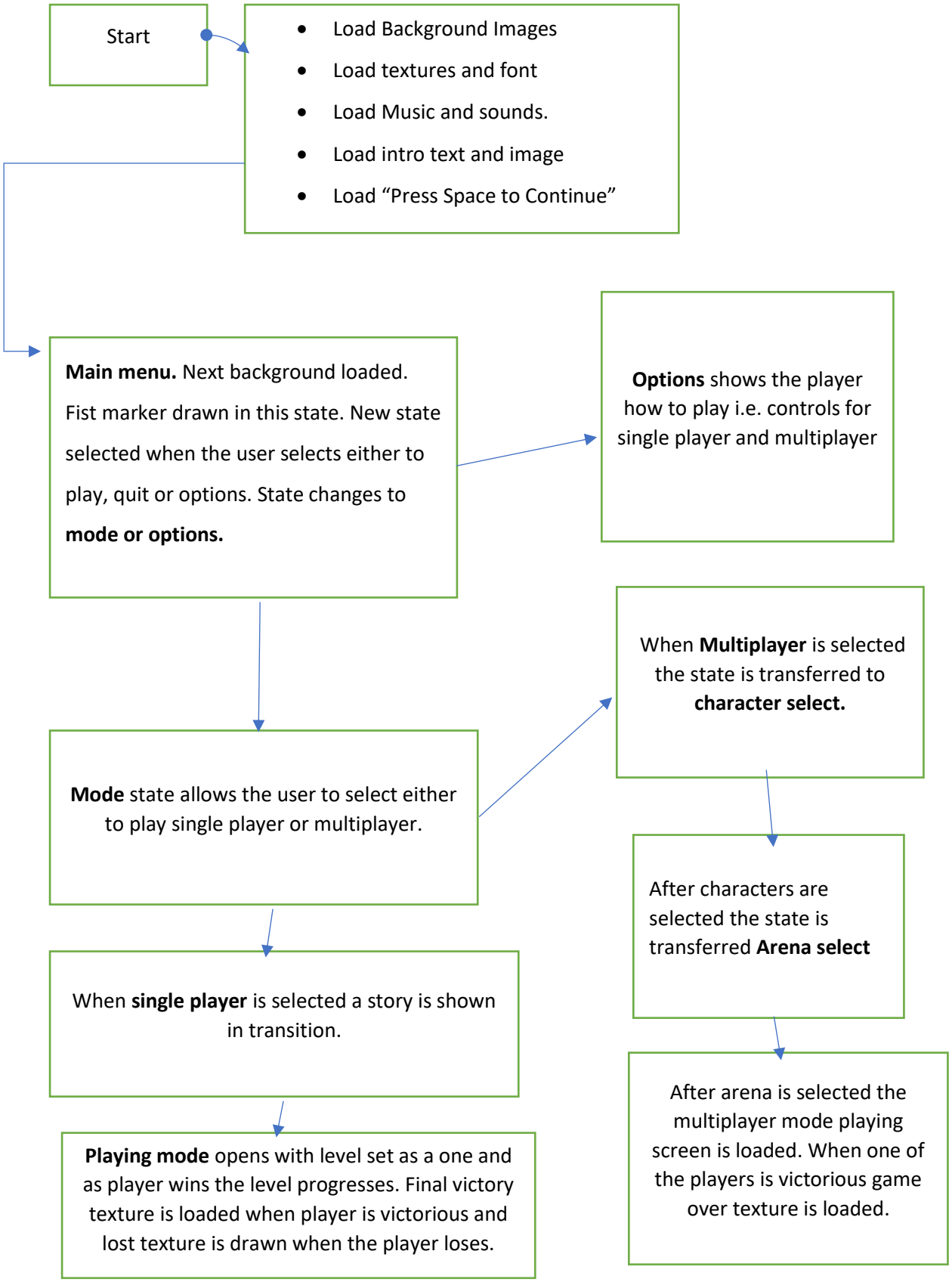
4. **Music**
   The constructor of this class loads all the sound and music files.
   The sound files are loaded into the buffer for quick access during gameplay and avoid lag while the music files are directly streamed by the music class of SFML to avoid excessive memory consumption.
   For each of the sounds a function is created to play the sound whenever needed simply with a function call.

5. **Opponent(multiplayer)**
   This is an inherited class from class Player to replicate the actions of our player in single player mode. It has additional code to detect buttons from the keyboard and perform actions accordingly.

The programming methods we've used can as summarized below:

- Discussion among team members and initial research.
- Scheduling the project and dividing the tasks among the group members.
- Initial coding for creating logic, including defining a clear-cut diagram of the classes required to fully implement OOP principles.
- Coding the program and updating it among team members
- Execution and testing the program.
- Debugging.
- Program Documentation.

Implementation Block Diagram



Result

With our major objective being implementing the concept of OOP to achieve a smooth Window based gaming experience, we were nearly able to achieve almost all major objectives that we set for this proposal. We did have a hard time figuring out the implementation of some additional features to this game like, adding sprite sheet animation and this was due to the lack of proper graphic resource for adding the animation. Besides some minor noise to the purposed objectives, we would consider this as a successful project fulfilling almost all objectives and has helped us gain a mirror understanding of the concept of OOP and Game development with SFML.

Problems Faced and Solutions

Obviously, the first problem we faced was understanding the architecture of game development, we almost had 2 failed attempts of creating a properly manageable game architecture. With that being said, here are some other problems we faced:

- Finding the correct sprites and editing them so that they could be used as animation turned out to be a lengthy process.
- We had great difficulties in developing proper logic due to lack of proper reference materials on such game.
- Problems occurred while merging the code from different members like linking the game logic with the already drawn sprites.
- The SFML graphics library was completely new for us considering that we had not done our first-year project using a graphics library.

Now, to the solutions. Though we do know that the solution of every problem is "Google" obviously but being more precise. Here are some solutions we applied to the above-mentioned problems:

i) We decided to make a template of the SFML linked C++ file and then we didn't have linker errors coming up every time.

ii) After lot of test and debug we found a collision detection system which works to first find whether collision is taking place or not and then again check which animation is taking place during the collision phase to deduct the health of the player getting hit.

Limitation and Future Enhancement

- This game could be enhanced by adding different types of weather and increasing the arena number and character numbers.
- Graphics could be enhanced by making sprites more realistic and
- The game can be extended for multiple players to play in LAN

Conclusion and Recommendation

This project was unquestionably a good way of learning and implementing the way for programming practice. The coding is not the initial step for emergent of any program, rather a good planning on the basic framework and making decisions on the way of implementing the program is the most. As important coding is, debugging and testing are equally important aspects of application deployment. After the completion of the system, its management takes, is another most required obsession that is to be handled with great care. Thus, after the completion of our project, we can conclude proper judgment and implementation of the problems or topic leads to good programming practice which might lead to having desired output.

References

Books:

Secrets of Object-Oriented Programming in C++ - Daya Sagar Baral

Game Development with SFML – Jan Haller

'Beginning C++ Game Programming' -John Horton

'Programming Principles and Practice Using C++ (2014)' - Bjarne Stroustrup

Websites:

Sfml-dev.org

Stackoverflow.com

Geeksforgeeks.org