TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A minor project report on

A 2d action game using Raylib library

Submitted by:

Jeeshan Shrestha(THA081BCT012)

Pranay Tuladhar(THA081BCT020)

Hari Bahadur Basnet(THA081BCT011)

Submitted to:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

March, 2025

DECLARATION

We hereby declare that the report of the project entitled “A 2d action game using Raylib library” which is being submitted to the Department of Electronics and Computer Engineering, IOE, Thapathali Campus, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in Electronics and Communication Engineering, is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any university or institution for the award of any degree, and we are the only author of this complete work. No sources other than the ones listed here have been used in this work.

Hari Bahadur Basnet (Class Roll No: 081/BCT/011) _____

Jeeshan Shrestha (Class Roll No: 081/BCT/011) _____

Pranay Tuladhar (Class Roll No: 081/BCT/011) _____

Date: March, 2025

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a minor project work entitled "**A 2d action game using Raylib library**" submitted by **Hari Bahadur Basnet, Jeeshan Shrestha,** and **Pranay Tuladhar** in partial fulfillment for the award of Bachelor's Degree in Electronics and Communication Engineering. The Project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled, and ready to undertake any related work to their field of study, and hence we recommend the award of partial fulfillment of Bachelor's degree of Electronics and Communication Engineering.

_____

Project Supervisor
Saroj Shakya
Department of Electronics and Computer Engineering, Thapathali Campus

_____

External Examiner
Prof. Dr. Dinesh Kumar Sharma
Department of Electronics and Computer Engineering, Pulchowk Campus

_____

Project Co-ordinator
Dinesh Baniya Kshatri
Department of Electronics and Computer Engineering, Thapathali Campus

_____

Janardan Bhatta

Head of the Department,

Department of Electronics and Computer Engineering, Thapathali Campus

March, 2025

COPYRIGHT

ACKNOWLEDGEMENT

We would like to express our gratitude to everyone who supported us in the development of this 2D game project. Special thanks to our instructors, whose guidance helped us understand key concepts, and to our friends and family for their encouragement. We also appreciate the open-source community for providing valuable resources and inspiration.

This project has been a great learning experience, and we look forward to improving our skills further.

Jeeshan Shrestha (THA081BCT012)

Pranay Tuladhar (THA081BCT020)

Hari Bahadur Basnet (THA081BCT011)

ABSTRACT

This project aims to develop a 2D action game titled **"ALL AS PLANNED"** using the RayLib library in C. The game is inspired by the popular anime *Bleach* and features a protagonist, Ichigo, who must defeat enemies and a final boss, Aizen, to complete the game. The game incorporates various mechanics such as player movement, projectile attacks, enemy spawning, and boss battles. The project focuses on creating an engaging and interactive gaming experience while ensuring smooth gameplay, efficient resource management, and a user-friendly interface.

The game is designed to be a side-scrolling action game where the player controls Ichigo, who can move left and right, jump, and attack using a projectile called **Getsuga Tensho**. The game also includes a **Flash Step** ability, which allows the player to move faster for a limited time. The enemies spawn at regular intervals, and the game features two boss battles with increasing difficulty. The project emphasizes the use of object-oriented programming principles, efficient collision detection, and sound effects to enhance the gaming experience.

This project was possible due to the cooperation of the entire team members which managed different aspects of this projects like handling the files and managing it, programming the certain features and abilities of the character, cool new designs and cool new effects which enhanced the game, Even though the game is near its completion, few bugs are to be expected because this project is made by junior developers and students who have very less experience.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Background Introduction

In recent years, the gaming industry has seen exponential growth, with 2D games remaining a popular choice due to their simplicity and nostalgic appeal. The development of 2D games involves various aspects such as game design, physics, collision detection, and user interaction. This project focuses on creating a 2D action game using the **Raylib** library, which provides a simple and efficient framework for game development in C.

The game, **"ALL AS PLANNED"**, is inspired by the anime *Bleach* and features the protagonist, Ichigo, who must defeat enemies and a final boss, Aizen. The protagonist have their own sets of skills and moves, as for now there is only two moves that Ichigo can perform, they are Flash step and Getsuga Tensho, the enemy spawns from time to time, from a random side, if Ichigo is touched by the enemy then he will get damaged and if he kills the enemy with his moves then the score will increase, and once certain score is reached then the boss Aizen will be spawned, This is the first Aizen Boss, which has large Hit Point and if Aizen touches Ichigo than Ichigo lives is subtracted from his main health. Aizen's main attack freezes the target on touch, preventing the player from moving for a certain time.
Ichigo has a total of 10 lives and if he loses all of them, then the Game is over.

The programming Language used in making this game was C. With the help of RayLib library, it was possible to make this game visually more entertaining and more interactable, RayLib helps to make the Game have UI, Controls, logic, and much more.

1.2. Motivation

The motivation behind this project is to explore the field of game development and understand the intricacies involved in creating a functional and entertaining game. The project also aims to apply programming concepts such as collision detection, and resource management in a practical scenario. By developing a game inspired by a popular anime, the project also seeks to engage fans of the series and provide them with an interactive experience.

As people who were exposed to video games and anime at a very young age, we were inspired by them to make one for ourselves by combining the two things we love. We aim to combine video games with our favorite characters from the anime *Bleach.*

We, as the students of 081BCT, worked on this project not only for the project completion but also for the development of our knowledge on working with projects, which is a vital skill in the field of Technology.

1.3. Problem Definition

The project aims to develop a 2D action game with the following features:

- Player movement (left, right, jump)
- Projectile attacks (Getsuga Tensho)
- Enemy spawning and AI
- Boss battles with increasing difficulty
- Sound effects and background music
- Efficient collision detection and resource management

1.4. Project Objectives

The specific objectives of the project are:

- To study and understand the concepts of game development using Raylib.
- To implement player movement, projectile attacks, and enemy AI.
- To design and implement boss battles with increasing difficulty.
- To incorporate sound effects and background music to enhance the gaming experience.
- To ensure efficient collision detection and resource management.

1.5. Project Scope and Applications

The proposed game can be used for:

- Entertainment and leisure.
- Educational purposes to teach game development concepts.
- Showcasing the capabilities of the Raylib library in game development.

The scope of the project includes the development of a 2D action game with the aforementioned features. The game will be developed using the Raylib library in C, and the focus will be on creating a smooth and engaging gaming experience. The project will also include testing and debugging to ensure the game runs efficiently on various platforms.

1.6. Report Organization

This should briefly explain all the chapters.

2. LITERATURE REVIEW

Game development has evolved significantly over the years, with 2D games remaining a popular choice due to their simplicity and nostalgic appeal. The use of libraries such as Raylib has made game development more accessible, allowing developers to focus on game design and mechanics rather than low-level programming.

Raylib is a simple and easy-to-use library for game development in C. It provides functions for rendering graphics, handling input, and playing audio, making it an ideal choice for developing 2D games. The library is lightweight and cross-platform, making it suitable for developing games that can run on various platforms.

The game **"ALL AS PLANNED"** draws inspiration from the popular anime *Bleach*, which has a large fan base worldwide. By incorporating elements from the anime, the game aims to engage fans and provide them with an interactive experience.

3. REQUIREMENT ANALYSIS

This project required a personal computer with a proper GPU support and with gcc or g++ installed to compile and run the C code. This project further requires the use of the Raylib graphics library to render game objects and window objects.

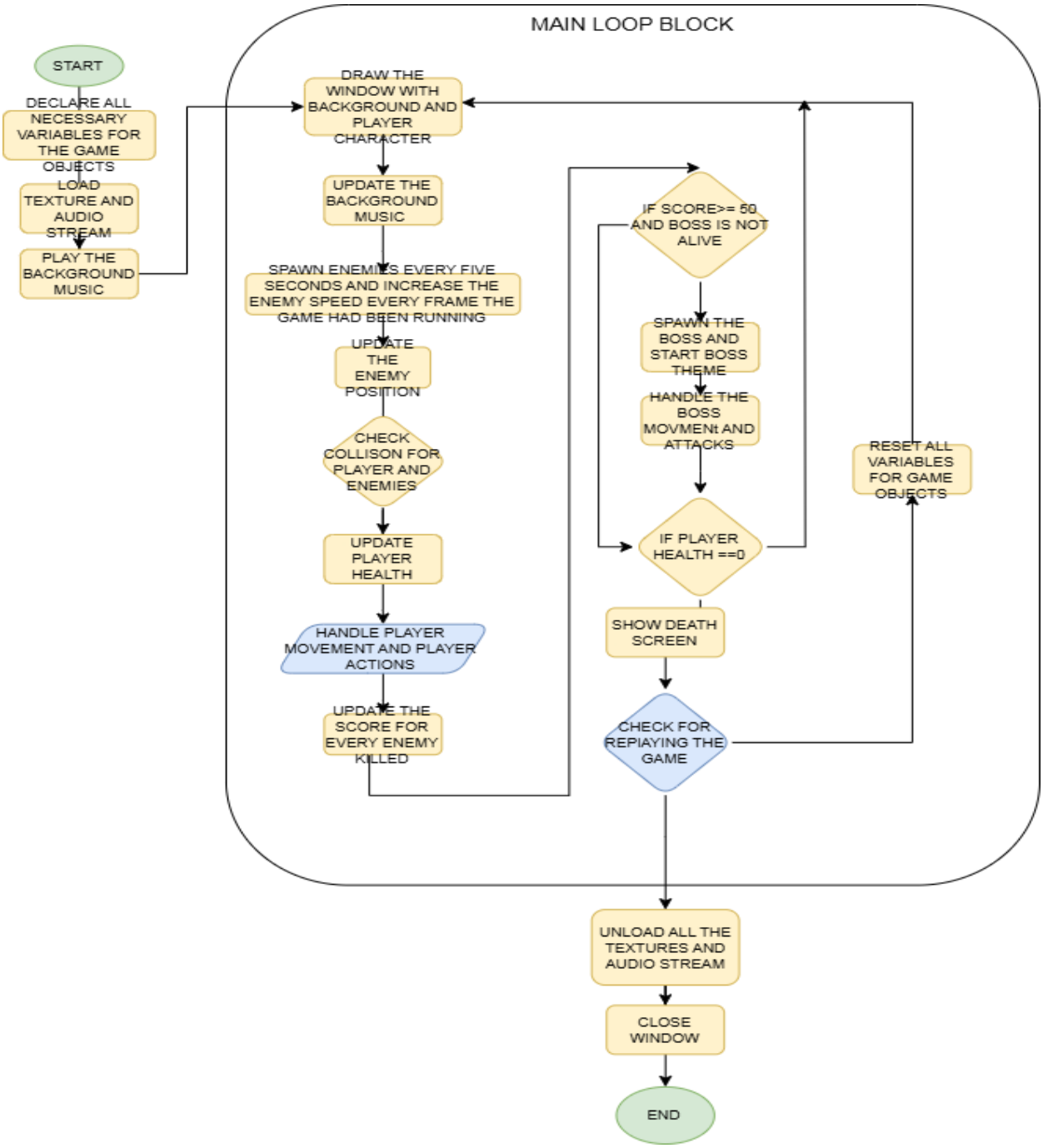
3.1. Project Requirement

- Computer
    - Required for writing, compiling and running the C program.
- CPU and GPU
    - CPU for processing the game login, enemy spawning, and collision detection
    - GPU for rendering the 2D sprites of the game objects
- Mouse and Keyboard
    - The keyboard is used for controlling player movements
    - The mouse for using attacks
- C compiler
    - Required for compiling the C source code to an executable
- Raylib Library
    - Required to use functions to render 2D graphics, handle background audio, and sound effects.
- IDE/Text Editor
    - Used for debugging and writing C code efficiently

4. SYSTEM ARCHITECTURE AND METHODOLOGY

Algorithms and methods and block diagrams go here

.



5. IMPLEMENTATION DETAILS

The main working of this project is solely dependent on Raylib and the logics used to build this project. On this project, there are lots of Game Development concepts that have been implemented, such as Vector2, DeltaTime, MainLoop, KeyboardInput, and a lot more. Rather than a whole bunch of functions and modules, this entire game runs on one game loop, followed by logically built conditions and flags. Below are the methods that have been implemented to create the game and its logic.

5.1. Defining the structures

Initially, we define the structures for the projectile's detail and the enemies' detail. The main reason for this structure was to be used instead of classes, as there is no concept of classes in C, we assume those projectile and enemies as classes from which we will create pseudoObjects for our code later.

The use of Structure covers up for the lack of class concept in C programming, although we can't imitate the constructor on structure, we can still make use of the values stored in the structure to create a new datatype, or we can say object.

5.2. Defining the Macros

After defining our structures, which we will be using as pseudoClass to create pseudoObjects, we then create the macros that we will be frequently using throughout the code like screen_width, screen_height and so on.

Defining macros is super useful for ease of understanding and also to not memorize the values for certain methods. Let's take an example, when initializing the window with Raylib we need to pass 3 arguments: int Screen_Height, int Screen_Width, and char[] title. And if we define our screen width and size in the beginning, we don't have to keep writing those numbers, and also we will be using screen width and sizes a lot in this project, so macro is a must.

5.3. Main Function

After defining the macros, our main function finally starts, which includes all our code. It contains every logic, flags, variables, and loops that were used in this project. Inside the main function, we first initialize the Raylib window with desired width and height, and import our audio and texture that we will be using in this project on the start of our main function. Inside the main method, we have also described the player's data and boss data, this could have been done through a structure, but since we only have 1 player, it didn't seem necessary to create an entire class just for the player.

5.4 Main Game Loop

This is the main loop that we will be executing our entire code in, and this loop will render graphics and logic every frame, and since our project is built on 60FPS, it will render the graphics and execute the code every 1/60 second.

5.5 PlayerMovement and Attack Logic

By using the Raylib inbuilt functions we can get the keystrokes that has been pressed by the user and receive that keystroke and if that keystroke is either W A S D than we will change the position of the player by incrementing and decrementing it by the player speed * deltaTime to keep the movement independent of the frames and solely depend on playerSpeed. We also prevented player texture from going outside the bounds by adding some restrictions into the player positions.

Using Raylib's inbuilt mouse functions, we can get the MouseClicks that have been clicked by the user, and use that intel to simulate an attack. For example, when the user clicks the left click, then we will render a new texture which inherits the property of projectile that we defined, and if the texture sprites collide with any enemies, then that enemy will take damage. We also added attack cooldown so that the player can’t spam attacks. And with the left click button we just gave the player the extra movement speed to avoid enemies and boss attacks.

5.6. Enemy Spawn Logic

By using another Raylib function we can keep track of real time that has been passed in the game. The function used is GetFrameTime(), it returns the time interval between every frame, and if we add it every frame, then we can keep track of the time passed in the game, and with this time we can create a logic of enemy spawning every defined interval of time.

We also added the macro and defined it 0 and 1 which helps to create a random side where the enemy spawns from. By using the Raylib Random function we can generate a random number between 0 and 1 and if it's 0 the enemy will spawn from left and if it's 1 then the enemy will spawn from right.

5.7. Boss Spawn Logic

When the player clicks the left mouse button the texture generated If it collides with the enemy texture then the enemy gets unrendered and the score gets incremented. If the score reaches 50 then the boss flag is set true and we draw the boss texture having huge health and a projectile skill. Boss skill freezes the player and if the player gets touched by the boss then the player loses health.

There is also a secret final boss that spawns after the user gets 1000 score and it is the final boss of the game and it has super massive health.

5.8. Displaying text on the screen

By using another Raylib inbuilt function like DrawText, we can give the player the guide on how to play the game, and keep track of the scores and lives of the player. This will help in giving necessary information to the player, and once the player dies, we can display the Game over screen using this inbuilt function.

5.9. Game Restart Condition

After the player dies, they can press R to restart the game. We can use the Raylib inbuilt method to get the player input, and if the player presses R, then we can reset every condition, such as enemy speed, player health, restart the game music, despawn the boss if necessary, and all the important things.

On pressing R, our Entire GameLoop restarts, then the player can enjoy the fresh start of the game.

6. RESULTS AND ANALYSIS

The game's performance was analyzed based on player movement, enemy spawn rates, and ability cooldowns. Player movement was smooth, while jumping had a slight landing delay due to gravity effects. Enemy spawn rates, set between 3 to 5 seconds, were observed to vary from 3.2 to 5.1 seconds, confirming the randomization function worked within acceptable limits..

Minor errors were observed, mainly due to frame rate delays affecting cooldowns and collision detection. Occasional enemy hits registered a few pixels off, and spawn timing fluctuated slightly due to random number generation. Despite these small deviations, the game functioned as intended. Future optimizations could refine cooldown precision, collision accuracy, and spawn timing, but overall, the game met its design objectives effectively.

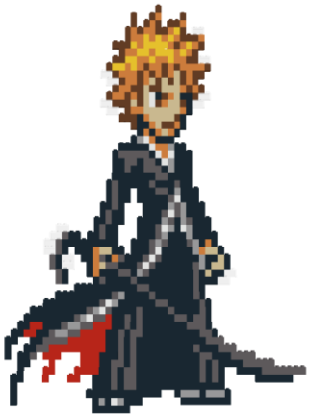


Fig 1: Player Ichigo

Fig 2: Enemy

Fig 3: Aizen Boss

Fig 4: Screenshot of Game

7. FUTURE ENHANCEMENT

This game is far from complete but with future enhancement this game will be very enjoyable and fun to pass time. Here are some of the enhancements that might be added to increase the games reach and add more flexibility to the game.

7.1. Visual and Animation Improvements:

By adding More complex and beautiful visuals this game will catch more eyes and currently the game has no animation which makes it looks dull and plain but after adding the animation we can make the game more interactive and fun.

7.2. Gameplay Mechanics:

For the Gameplay we can add more characters from bleach with each new set of skills and we can add more levels and stages to this game. We can also implement AI controlled enemies to make the game more realistic and challenging.

7.3. User Interface:

Currently there is no menu and user interface in the game right now but in the future we can add the user interface to customize our game and customize the player attributes and all to make the game feel much more versatile and fun. We can also add the health bar and energy meter to the player to make the game feel more competitive and hard and not so dull.

Implementing a pause and resume menu might be a good idea too.

7.4. Multiplayer and Networking:

Although this seems super out of reach as of right now but maybe in the future we can host a server and make the game multiplayer which would instantly make the game much more fun and we can enjoy it with out friends.

7.5. Story Mode and Progression:

We can also add the story mode into this game to make this game feel more interactable and where the player gets all the skill as the story progresses and also we can express the emotion through the story which might make the game feel more engaging.

8. CONCLUSION

The development of the 2D game based on the Bleach anime, utilizing the Raylib library and the C programming language, has been a rewarding and insightful experience. Through this project, a comprehensive understanding of game design principles, graphics rendering, and real-time interaction was achieved. The game's simplicity allowed for a focus on core mechanics and efficient performance while staying true to the aesthetic and thematic elements inspired by the Bleach universe.

The project successfully implemented fundamental features such as character control, basic combat mechanics, and visual effects. The use of Raylib proved to be advantageous due to its lightweight nature and ease of handling 2D graphics. Additionally, the application of C programming enabled fine-grained control over memory management and performance optimization, which was crucial for smooth gameplay.

Despite the accomplishments, there remains significant potential for future enhancements. These include improved animations, advanced AI for enemy behavior, multiplayer capabilities, and a more immersive storyline. Integrating sound effects, background music, and a robust user interface can further enhance the player's experience.

In conclusion, this project serves as a solid foundation for further development and expansion. The knowledge gained in graphics programming and game development frameworks will be invaluable for future endeavors in the gaming industry. With continued effort and innovation, this game can evolve into a more complex and engaging experience for fans of the Bleach anime and the gaming community alike.

REFERENCES

Raylib Documentation: https://www.raylib.com/

Bleach Anime: https://www.crunchyroll.com/bleach

Game Development Concepts: https://www.gamedev.net/

Error Debugging: https://chatgpt.com/

Game Development Techniques:https://www.youtube.com/watch?v=wVYKG_ch4yM&list=PLwR6ZGPvjVOSRywn9VCQ3yrRVruxzzuo9&ab_channel=ProgrammingWithNi