华南理工大學

South China University of Technology

《High-level Language Programming Project》
Report

Project Name: Yesternightmare

School : Future Technology

Major : Data Science and Artificial Intelligence

Student Name : Yue Zhang, Ziwen Huang, Xueyi Zhang

Teacher : Huaidong Zhang

Submission Date : 2024.7.7

Catalog

Project Name: Yesternightmare 1

Submission Date : 2024.7.7 1

1. Requirement Analysis for System 3

    1.1 The Background and Motivation of System 3

    1.2 System Objectives 4

        1.2.1 Basic Process 4

        1.2.2 Diagram of C++ inheritance used in the project 5

2. Program Analysis 8

    2.1 Key Issues for System 8

        2.1.1Animation Production 8

        2.1.2 User Interface Design 8

        2.1.3 Program Class Inheritance Structure Design 8

        2.1.4 Functionality and Data Design of Each Class in the Program 9

    2.2 Duty Assignments 9

3. Technical Routine 9

4. Programming Progress 14

5.Testing report 14

6. Summary 17

7. Reference 18

1. Requirement Analysis for System

1.1 The Background and Motivation of System

As students with years of gaming experience, we have observed that many games on mobile and social platforms suffer from obvious shortcomings in playability and design quality. These games often feature monotonous content, lack sufficient interactive design, and have rough graphics and effects, significantly limiting the gaming experience and market potential. Thus, we see a clear market opportunity to develop a two-player online game that combines innovative design, superior playability, and high-level technology to fill this gap.

The game we plan to develop is called "Yesterday's Nightmare," inspired by the River Styx in Greek mythology, and explores the meaning of death and life. The game is set in an endless nightmare, where the protagonist and their partner continuously challenge bizarre worlds and monsters in the dream, trying to break the curse of the nightmare and embrace the dawn of a new day. This game not only combines elements of reality and fantasy but also conveys the loneliness and helplessness humans face in the face of death through strong color contrasts and dim lighting effects.

In "Yesterday's Nightmare," players will experience a rich blend of adventure, sports, shooting, level-solving, and role-playing elements. Each level is a dual test of mind and courage, requiring players to exert their utmost effort to defeat the monsters in the dream and gradually break the cycle of the nightmare.

To enhance the game's appeal and replay value, we are introducing an engaging storyline. This will not only make "Yesterday's Nightmare" stand out in the fiercely competitive gaming market but also increase players' investment and interest in the game. Through the alternation of reality and dream, players will continuously reflect on the meaning of life and the essence of death, experiencing deep emotional resonance.

Additionally, this project is an excellent opportunity to practice and expand the knowledge we have acquired in our C++ courses. By applying C++ and related software development technologies in a real project, we can deepen our understanding of object-oriented programming, data structures, algorithms, and software engineering practices. This will not only enhance our programming skills and system design capabilities but also improve our ability to solve complex technical problems, laying a solid foundation for future advanced software development work.

Through this project, we hope to provide a high-quality game product that meets the market's demand for innovative video games. At the same time, the promotion of teamwork and innovative thinking will contribute to our academic and career

development. During the development process, team members will have the opportunity to explore various aspects of game design, from user interface design to backend logic processing, comprehensively improving individual and team technical skills.

1.2 System Objectives

1.2.1 Basic Process

Developing the game "Dreams of Yesterday" requires coordination and collaboration across multiple areas, including game design, programming, art design, sound production, and user experience. Here are the detailed steps for the development process:

A. Project Planning and Design:

A.(a) Game Concept and Background Setting

Game Background: The protagonist and their partners battle within dreams, attempting to escape from nightmares. The storyline will influence level design and monster settings.

Level Design: Each level contains different rooms, each with three types of monsters. Players need to kill all monsters to progress to the next level. Rooms include platforms and terrain for players to utilize.

A.(b) User Interface Design

Main Menu: Includes options for "Enter Dream", "Game Settings", "Personal Achievements", and "Leaderboard".

Enter Dream: Press Enter to start the game.

Game Settings: Configure game graphics and ID name.

Personal Achievements: Displays total kills, highest kills in a single run, highest level reached, and shortest time.

Leaderboard: Records the achievements of top-ranking players.

B. Technology Selection and Architecture Design

B.(a)Technology Stack Selection

Programming Language: C++

Version Control: Use Git for code version control.

B.(b)System Architecture Design

Game Logic: Includes movement, jumping, shooting, and monster AI.

Interface Logic: Includes main menu, settings interface, achievements, and leaderboard display.

Data Storage: Store player achievements and leaderboard data in local files or a server database.

C. Development Process

B.(a)Game Prototype Development

Character Control: Implement A, D keys for movement, W key for jumping, E key for interaction, and Enter key for confirming actions. Use the left mouse button to attack, with the cursor controlling the attack direction.

Level Prototype: Create a simple level featuring platforms and basic behaviors for the three types of monsters.

User Interface: Initial implementation of the main menu and settings interface.

B.(b)Detailed Implementation

Character Control Optimization: Adjust the physical parameters of character movement and jumping for a smoother gameplay experience.

Shooting Mechanism: Implement the algorithm for shooting at different angles, ensuring precise attacks.

Monster AI: Design behaviors for three different types of monsters, including tracking the player and long-range attacks.

Level Design: Add more levels and rooms, each with unique terrain and monster combinations. Ensure the difficulty gradually increases, challenging the player's skills and strategy.

User Interface Refinement: Implement the display and updating of personal achievements and the leaderboard.

B.(c)Sound and Art Design

Sound Effects: Add background music, shooting sounds, monster attack sounds, etc., to enhance the game atmosphere.

Art Design: Design visual assets for characters, monsters, levels, and interfaces, ensuring a consistent overall style.

B.(d)Testing and Optimization

Functionality Testing: Conduct unit tests, integration tests, and system tests to ensure all game functions work correctly and there are no significant bugs.

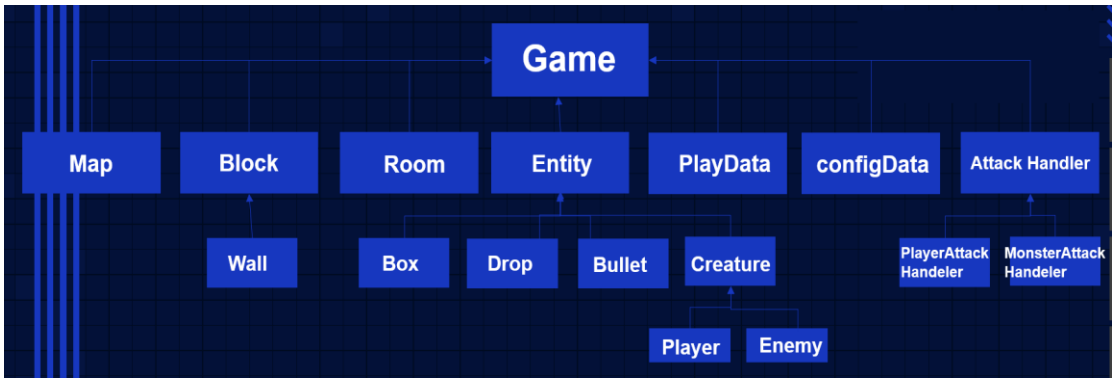
Performance Optimization: Optimize game performance to ensure smooth operation on various platforms.

User Experience Testing: Invite players for playtesting and adjust game difficulty and operation experience based on their feedback.

Through the detailed planning and implementation steps outlined above, we developed a high-quality game, "Dreams of Yesterday," with a rich background story, exciting gameplay experience, and deep user interaction.

1.2.2 Diagram of C++ inheritance used in the project

Sure, here is the translation:Using the design concept of inheritance, we roughly design the game into several classes with hierarchical relationships:



All elements in the game are implemented as classes. There are five base classes: Entity class, Map class, Room class, Player Data class, and Game Settings class, with the Entity class being the most important, as it is an indispensable part of the main content of the game.

The elements of the Entity class can be divided into four categories: Bullet class (main character), Creature class (monsters, players, etc.), Drop class, Box class, and Block class. The Creature class, which includes monsters, is further divided into various types of monsters. The Drop class is further divided into various items. This process of subdividing entities is implemented through class inheritance. Each class also has unique functions to implement various features.

Among these, the most essential is the Entity class. Elements in the game process are considered entities, and the Entity class records the basic information of entities such as coordinates and size. We attempt to abstract the common parts of various concepts and establish a relatively reasonable class relationship:

Initialization Function：

The initialization function, `Init()`, is responsible for initializing character information, importing game images, music, and other functionalities.

Drawing Function：

The drawing function, `show()`, primarily renders the various elements in the game. It is mainly divided into drawing characters, monsters, in-room items, etc. Each

element's drawing is implemented through specialized drawing functions within their respective classes, allowing the `show()` function to be organized and clear.

Collision Functionality：

Collision with walls limits the movement range of characters and monsters, providing footholds for characters and giving players more strategy options. This functionality checks whether two elements intersect on a 2D plane after movement; if they intersect, the movement is canceled. Collision with bullets is determined by checking if a bullet intersects with an element after moving. If they intersect, the bullet is deleted, and the corresponding element's health is reduced. Additionally, when monsters collide with characters, the character's health is also reduced.

Power-Ups：

To replicate the feature of characters becoming stronger during their adventure in the original game, players gain random power-ups after eliminating all monsters in a room. This functionality determines whether a character can receive a power-up by recording the number of monsters in the room and uses a random function to decide the type of power-up. Power-up types include increasing health, improving firing rate, etc.

Game Completion：

Strictly speaking, this game is an endless adventure game. When players eliminate all monsters in the rooms, the character is transported to the next level for a new round of adventure. The player's goal is to clear as many levels as possible.

Game Over：

The game determines a player’s failure when the character's health reaches zero. After failing, the system records information such as the number of monsters defeated by the player to calculate their score and upload it to the leaderboard. After failing, players can choose to exit the game or restart the adventure.

Leaderboard Functionality:

We record the total number of kills by players and the highest number of kills in a single round, implementing the leaderboard functionality.

2. Program Analysis

2.1 Key Issues for System

2.1.1Animation Production

Frame Synchronization and Smoothness: Ensuring the animation frame rate is synchronized with the game frame rate to achieve smooth animation effects.

Resource Management: Managing a large number of animation resources (such as sprite sheets, PNG sequences, etc.) and optimizing them for memory usage and loading speed.

Complex Animation Logic: Designing and implementing complex animation logic, such as smooth transitions between various states like walking, jumping, and attacking.

Performance Optimization: Ensuring game performance is not impacted even with high-complexity animations and a large number of animation objects.

2.1.2 User Interface Design

User Experience (UX): Designing an intuitive and user-friendly interface to ensure players can easily navigate and use the game functions.

Adaptive Layouts: Creating an interface that adapts to different screen resolutions and devices, ensuring consistent visual effects and functionality.

Interactive Feedback: Providing timely and clear user interaction feedback, such as button clicks and menu selections, to enhance the user experience.

UI Performance: Optimizing the rendering and responsiveness of UI elements, especially when the interface is complex or contains many elements.

2.1.3 Program Class Inheritance Structure Design

System Architecture Design: Designing a clear and reasonable class inheritance structure, ensuring that each class has a distinct responsibility and clear inheritance relationships, avoiding confusion and redundant code.

Scalability: Ensuring the system architecture is highly scalable, allowing for easy addition of new features or modification of existing ones without breaking the original structure.

Polymorphism and Reusability: Effectively using object-oriented polymorphism and reusability to reduce code duplication and improve code maintainability.

Dependency Management: Controlling dependencies between classes to avoid tight coupling, ensuring the modularity and independence of the system.

2.1.4 Functionality and Data Design of Each Class in the Program

Data Encapsulation and Privacy: Properly encapsulating data to ensure the privacy of internal data, preventing external direct access and modification of internal states.

Function Responsibility Division: Clearly defining the responsibilities of each class, adhering to the single responsibility principle, where each class is responsible for only one thing.

Data Consistency: Ensuring data consistency and correctness, especially in multi-threaded or asynchronous operations.

Performance Optimization: Designing efficient data structures and algorithms to ensure performance when handling large-scale data processing in the game.

2.2 Duty Assignments

Everyone devotes equally to the game!!!

Person	Duty
Yue Zhang	Game and UI Design、Animation
Ziwen Huang	Initialization creation, file reading
Xueyi Zhang	Class structure design and Inheritance and polymorphism design

3. Technical Routine

Based on the provided information, the system employs several technical routines to achieve its objectives. These routines encompass various aspects of the game, from data display and synchronization to configuration management and visual effects. Here are the key technical routines used in the system:

Text Rendering and Conversion:

The system uses text rendering functions such as outtextxy to display player game statistics on the screen. This involves converting strings to wide characters using

MultiByteToWideChar before rendering them. For example, player statistics like "Total Kills" and "Highest Single Round Kills" are converted and displayed at specific coordinates.

String conversion routines are used to format and concatenate strings with integer values, typically using the sti function. This function converts a string and integer into a concatenated string, which is then used for display purposes.

```
void showMenu_3() {
    int firstflash = 1; //第一次进入的时候要先显示了UI再读入，不然输入前会黑屏
    int select = 1;
    cleardevice();
    BeginBatchDraw();
    while (1) {
        cin.clear();
        if (firstflash == 1) {
            firstflash = 0;
        }
        else {
            char k = 'f';
            k = getch();
            if (k == '\r') {
                if (select == 1) {
                    break;
                }
            }
        }
        gifCreater("menu.1", WIDTH / 2 - 300, 0, tick, 600, 800, 0, 1);

        settextcolor(RGB(133, 96, 208));
        setbkmode(TRANSPARENT);
        settextstyle(20, 0, _T("宋体"), 0, 0, 800, 0, 0, 0); \
            string t;
        TCHAR tszWord[1024];
        char szWord[100];
        t = sti("总击杀数:", pdata.totilKilled);
        strcpy(szWord, t.c_str());
        MultiByteToWideChar(CP_ACP, 0, szWord, -1, tszWord, 1024);
        outtextxy(470, 370, tszWord);
        t = sti("最高单轮击杀:", pdata.maxOnceKilled);
        strcpy(szWord, t.c_str());
        MultiByteToWideChar(CP_ACP, 0, szWord, -1, tszWord, 1024);
        outtextxy(470, 400, tszWord);
        t = sti("最高层数:", pdata.level);
        strcpy(szWord, t.c_str());
        MultiByteToWideChar(CP_ACP, 0, szWord, -1, tszWord, 1024);
```

```
        setbkmode(TRANSPARENT);
        settextstyle(20, 0, _T("宋体"), 0, 0, 800, 0, 0, 0); \
            string t;
        TCHAR tszWord[1024];
        char szWord[100];
        t = sti("总击杀数:", pdata.totilKilled);
        strcpy(szWord, t.c_str());
        MultiByteToWideChar(CP_ACP, 0, szWord, -1, tszWord, 1024);
        outtextxy(470, 370, tszWord);
        t = sti("最高单轮击杀:", pdata.maxOnceKilled);
        strcpy(szWord, t.c_str());
        MultiByteToWideChar(CP_ACP, 0, szWord, -1, tszWord, 1024);
        outtextxy(470, 400, tszWord);
        t = sti("最高层数:", pdata.level);
        strcpy(szWord, t.c_str());
        MultiByteToWideChar(CP_ACP, 0, szWord, -1, tszWord, 1024);
        outtextxy(470, 430, tszWord);
        t = sti("最短用时:", (int)pdata.lastTime / 1000);
        t += "s";
        strcpy(szWord, t.c_str());
        MultiByteToWideChar(CP_ACP, 0, szWord, -1, tszWord, 1024);
        outtextxy(470, 460, tszWord);

        if (select == 1) {
            settextcolor(RGB(64, 207, 235));
        }
        else {
            settextcolor(RGB(133, 96, 208));
        }
        outtextxy(580, 500, _T("确认"));
        FlushBatchDraw();
    }
}

void showMenu_4() {
```

Configuration File Management:

The system reads from or creates configuration files to manage game settings. This ensures that user preferences, such as sound effects, are maintained across gaming

sessions. Configuration management routines involve checking for the existence of configuration files and loading or initializing settings as needed.

```
void showMenu_2() {
    int firstflash = 1; //第一次进入的时候要先显示了UI再读入，不然输入前会黑屏
    int select = 1;
    int isInput = 0;
    string name = "";
    cleardevice();
    BeginBatchDraw();
    while (1) {
        cin.clear();
        char k = 'f';
        string pri = "游戏ID:       ";
        if (firstflash == 1) {
            name = pdata.name;
            firstflash = 0;
        }
        else {
            k = getch();
        }
        if (k == 0x08) {
            if (isInput == 1 && !name.empty()) name.pop_back();
        }
        else {
            if (isInput == 1) {
                if (select == 1) {
                    if (k == 'a' || k == 'A' || k == 'd' || k == 'D') {
                        pcfg.changeEffects();
                    }
                }
                if (select == 2) {
                    if (('a' <= k && k <= 'z') || ('A' <= k && k <= 'Z') || k == '_' || k == '-') {
                        name += k;
                    }
                }
            }
            else {
                if (k == 'w' || k == 'W') {
                    if (select > 1) {
                        select--;
                    }
                }
                if (k == 's' || k == 'S') {
                    if (select < 3) {
                        select++;
                    }
                }
            }
        }
        if (k == '\r') {
            if (select == 1 || select == 2) {
                isInput = isInput == 1 ? 0 : 1;
            }
            if (select == 3) {
                pdata.clear();
                strcpy(pdata.name, name.c_str());
                pdata.save();
                pcfg.save();
                break;
            }
        }

        pri += name;

        gifCreater("menu.1", WIDTH / 2 - 300, 0, tick, 600, 800, 0, 1);

        setbkmode(TRANSPARENT);
        settextstyle(20, 0, _T("宋体"), 0, 0, 800, 0, 0, 0);
        TCHAR tszWord[1024];
        char szWord[100];
        strcpy(szWord, pri.c_str());
        MultiByteToWideChar(CP_ACP, 0, szWord, -1, tszWord, 1024);
        if (select == 1) {
            if (isInput == 1) {
                settextcolor(RGB(255, 0, 0));
```

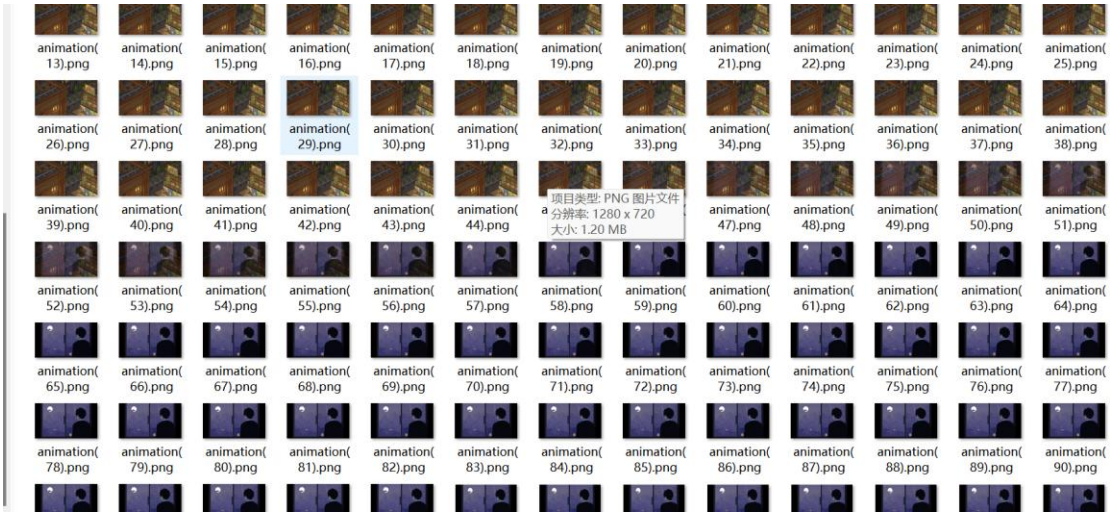
```
        settextcolor(RGB(255, 0, 0));
        }
        else {
            settextcolor(RGB(64, 207, 235));
        }
    }
    else {
        settextcolor(RGB(133, 96, 208));
    }
    if (pcfg.getEffects() == 1) {
        outtextxy(510, 390, _T("特效等级:     ◂ 低 ▸"));
    }
    else if (pcfg.getEffects() == 2) {
        outtextxy(510, 390, _T("特效等级:     ◂ 高 ▸"));
    }

    if (select == 2) {
        if (isInput == 1) {
            settextcolor(RGB(255, 0, 0));
        }
        else {
            settextcolor(RGB(64, 207, 235));
        }
    }
    else {
        settextcolor(RGB(133, 96, 208));
    }
    outtextxy(510, 420, tszWord);
    if (select == 3) {
        settextcolor(RGB(64, 207, 235));
    }
    else {
        settextcolor(RGB(133, 96, 208));
    }
    outtextxy(580, 450, _T("确认"));
    FlushBatchDraw();
}
```

Visual Effects and Animations:

The system enhances the visual appeal by playing animations and displaying animated gifs. Routines such as videoCreater and gifCreater are used to load and render video content and gifs at specified positions on the screen. This adds dynamic visual elements to the game, making it more engaging for the player.

Specific functions are used to initialize and control different animations, such as initNutsSnakeGif, initNutsJellyFishGif, and initOpeningAnimationGif. These functions set up the animations with parameters like position, size, and frame rate, contributing to the overall immersive experience.



Color and Text Style Management:

The system sets text colors and styles to enhance readability and provide visual feedback based on user actions. Functions like settextcolor and settextstyle are used to change the appearance of text elements dynamically. For example, the text color changes when a menu option is selected, providing immediate feedback to the player.

```
    setbkmode(TRANSPARENT);
    settextstyle(20, 0, _T("宋体"), 0, 0, 800, 0, 0, 0); \
        string t;
    TCHAR tszWord[1024];
    char szWord[100];
    t = sti("总击杀数:", pdata.totilKilled);
    strcpy(szWord, t.c_str());
    MultiByteToWideChar(CP_ACP, 0, szWord, -1, tszWord, 1024);
    outtextxy(470, 370, tszWord);
    t = sti("最高单轮击杀:", pdata.maxOnceKilled);
    strcpy(szWord, t.c_str());
    MultiByteToWideChar(CP_ACP, 0, szWord, -1, tszWord, 1024);
    outtextxy(470, 400, tszWord);
    t = sti("最高层数:", pdata.level);
    strcpy(szWord, t.c_str());
    MultiByteToWideChar(CP_ACP, 0, szWord, -1, tszWord, 1024);
    outtextxy(470, 430, tszWord);
    t = sti("最短用时:", (int)pdata.lastTime / 1000);
    t += "s";
    strcpy(szWord, t.c_str());
    MultiByteToWideChar(CP_ACP, 0, szWord, -1, tszWord, 1024);
    outtextxy(470, 460, tszWord);

    if (select == 1) {
        settextcolor(RGB(64, 207, 235));
    }
    else {
        settextcolor(RGB(133, 96, 208));
    }
    outtextxy(580, 500, _T("确认"));
    FlushBatchDraw();
}
void showMenu_4() {
```

Collision Detection:

The system includes routines for collision detection, particularly for interactions between bullets and enemies or the player. These routines check for overlaps between entities' coordinates and update their states accordingly. For instance, when a bullet collides with an enemy, the system reduces the enemy's health and marks the bullet as inactive.

```
int collisionBetweenEntity(float X1, float Y1, float X2, float Y2, float X3, float Y3, float X4, float Y4) //判断实体和墙有没有相交
{

    float a, b, s;
    a = min(X2, X4) - max(X1, X3);
    a = max(0, a);
    b = min(Y2, Y4) - max(Y1, Y3);
    b = max(0, b);
    if (a * b > 0)return 1;
    else return 0;
```

```
int Bullet::isCollideRocket(Creature creature) // 判断生物是否和子弹碰撞
{
    float distance_x = abs(creature.posx - posx);
    float distance_y = abs(creature.posy - posy);

    if (collisionBetweenEntity(posx, posy - 2 * radius, posx + 2 * radius, posy, creature.posx, creature.posy - creature.height, creature.posx + creature.width, creature.
    else return 0;
}
```

By leveraging these technical routines, the system ensures efficient data management, user interaction, visual presentation, and overall gameplay experience. Each routine

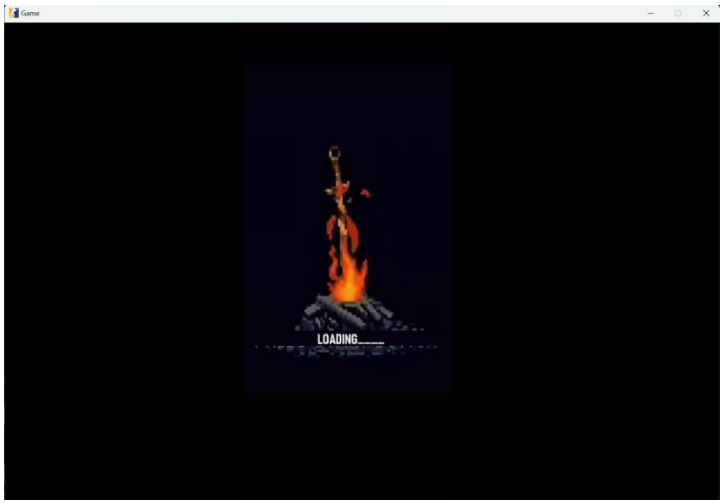
plays a crucial role in maintaining the functionality and responsiveness of the game, contributing to a seamless and enjoyable experience for the player.

4. Programming Progress

Phases of Mission	**Period**	**Planned Completion**	**Actual Completion**
Determining the topic and submit the proposal	2024.4.4-2024.4.10	Survey the interesting topics, and writing the proposal.	Finished
Project Setup	2024.4.15-2024.4.28	Design the game	Finished
Code structure	2024.4.29-2024.5.12	Design the class structure	Finished
Class realiztion	2024.5.13–2024.6.23	Accomplish the classes	Finished
Perfect the figure and animation	2024.6.24-2024.30	Find good resourses of figures and animation	Finished
Writing Report and Making Presentation Video	2024.7.1-2024.7.6	Summarize the project Design.	Finished

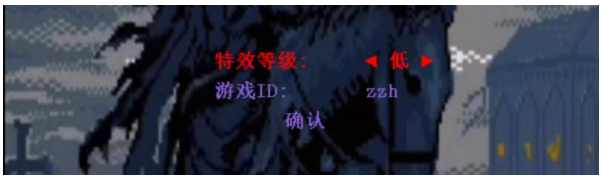
Note: The project will be started from April 4, 2024 and ended at July 6, 2024.

5.Testing report



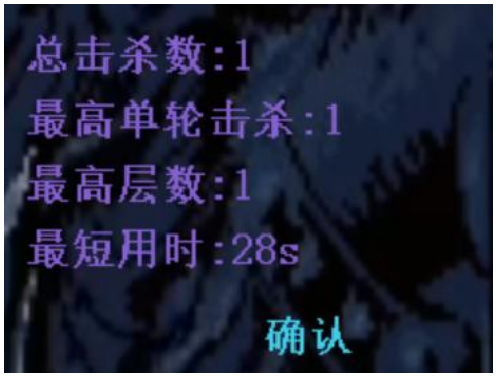
When opening the executable file, it will enter the loading screen.

The UI interface records the achievements of different players.

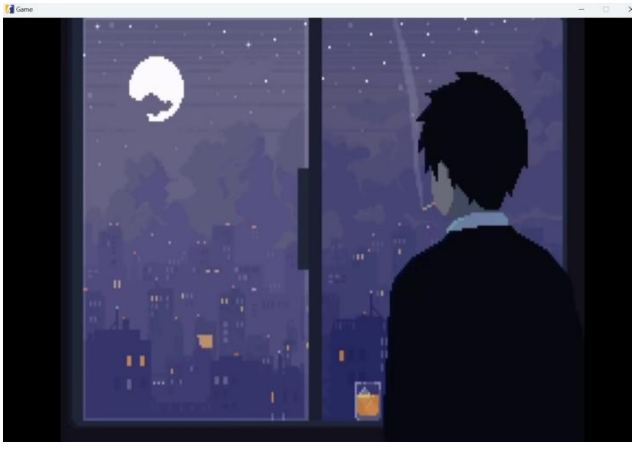


The game settings allow players to change the effect level.

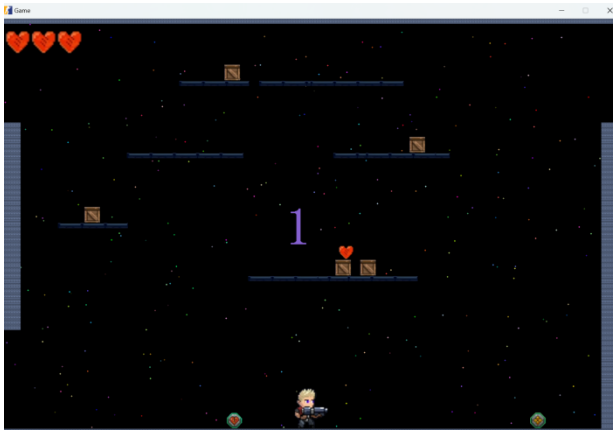
The game settings allow players to change their in-game name.



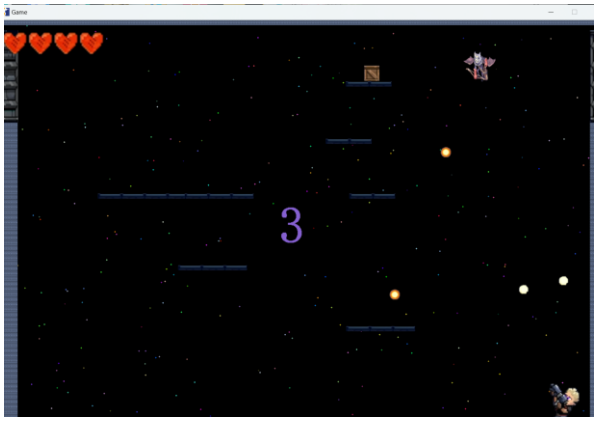
The UI interface allows players to view their personal records.



Story mode CG playback.

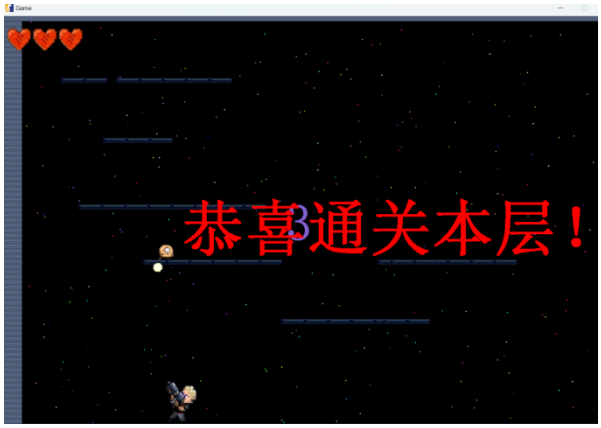


Animation interface and items.



Player attacks and monster attacks, with some monsters chasing the player.

The game ends if the health bar is depleted.



The game is won when all monsters are defeated.

6. Summary

"When we decided to develop 'Yesterday's Nightmare,' I was deeply attracted to its unique background and profound themes. Through this project, I gained a deeper understanding of the complexities of game design and experienced how to closely integrate storyline with game mechanics. Balancing the game's difficulty with player experience was a huge challenge during the design process. We experimented with various methods to convey the loneliness and helplessness humans feel in the face of fear and death, ultimately achieving the desired effect through strong color contrasts and lighting effects. Seeing players not only experience the thrilling adventure but also feel deep emotional resonance while exploring the meaning of life and death brings me immense satisfaction. I believe this experience will lay a solid foundation for my future work in game design."

We learned how to translate complex game mechanics into intuitive and engaging user interfaces. We wanted players to enjoy the challenge of the game while experiencing comfort through visual and operational aspects, deeply feeling the emotions and storyline conveyed by the game. During the design process, I constantly communicated with other team members to ensure our design fully expressed the core ideas of the game. Seeing players smoothly operate within the game and feel the eerie and mysterious atmosphere of the dreamscape through visual effects makes me very proud. This project not only improved my design skills but also deepened my

understanding of the importance of user experience in the success of a game. I believe these valuable experiences will greatly benefit my future career development."

"'Yesterday's Nightmare' was an excellent opportunity for me to apply the knowledge I learned in my C++ course to a real-world project. Throughout the development process, I deeply realized the importance of object-oriented programming, data structures, and algorithms. In solving the complex technical problems within the game, my team members and I constantly collaborated to find the best solutions, which not only enhanced my programming skills but also improved my problem-solving abilities. Particularly in implementing the game's backend logic and optimizing performance, we faced many challenges, but each successful resolution brought a great sense of accomplishment. This project made me more aware of the importance of teamwork and technological innovation in software development, laying a solid foundation for my future work in more advanced software development."

The development process of "Yesterday's Nightmare" was not just a blend of technology and creativity but also a result of teamwork and mutual growth. Everyone leveraged their strengths in this project and gained valuable experience and growth. We believe this game will not only meet the market's demand for innovative video games but also bring players deep emotional resonance and a unique gaming experience. The experiences we gained from this project will be a precious asset in our future careers.

7. Reference

None!