Proposal

1. Requirement Analysis for System

1.1 The Background and Motivation of System

Game Design Background:

As students with many years of experience playing games, we have observed that many games on mobile and social platforms have significant shortcomings in terms of playability and design quality.

The game we plan to develop is called "Yesterday's Nightmare," which is inspired by the great river of the Underworld in Greek mythology and the exploration of death and the meaning of life. The background of the game is set in an endless nightmare, in which the protagonist and his partners constantly challenge the treacherous world and monsters in the dream, trying to break the curse of the nightmare and meet the dawn again. This game not only combines reality and fantasy elements, but also conveys the loneliness and helplessness of human beings in the face of death through strong color contrast and dim light and shadow effects.

In "Yesterday's Nightmare", players will experience a variety of adventure, sports, shooting, level and role playing elements. Each level is a double test of heart and courage, players need to fight to defeat the monsters in the dream, and gradually crack the cycle of nightmares.

To enhance the game's appeal and replayability, we introduced a compelling storyline. This will not only make Yesterday's Nightmare stand out in a highly competitive gaming market, but will also increase player engagement and interest in the game. Through the alternation of reality and dreams, players will constantly ponder the meaning of life and the nature of death, and experience deep emotional resonance.

In addition, the project was a great opportunity to practice and extend what we learned in our C++ courses. By applying C++ and related software development techniques to real projects, 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 enhance our ability to solve complex technical problems, laying a solid foundation for more advanced software development work in the future.

Through this project, we hope to be able to provide a high quality gaming product that meets the market demand for innovative video games. At the same time, the promotion of teamwork and innovative thinking will also contribute to our academic and professional development. During the development process, team members will have the opportunity to delve into multiple aspects of game design, from user interface design to back-end logic processing, to improve the technical level of individuals and teams across the board.

Inspiration for the game:

The Dead of the Acheron

Author: Adolf Siremi Cich (1860-1933) Nationality: Hungary

Created in 1898

In the Greek gods, there are five great rivers in the underworld. The Acheron River is the first river in the Underworld, and those who enter the Underworld must pass through it first. Its water proportion is much lighter than the water in the world, even feathers can sink, even if the dead in the water for a long time, will be eroded by the river.

There are souls begging to be let back into the world, people afraid of the darkness of the underworld, people with garlands sitting as if they had accepted death peacefully, children sleeping without fear because they don't know enough to understand what death means.

The painting expresses the fear of death and the search for the meaning of life. Through strong color contrast and dim light and shadow effects, it conveys the loneliness and helplessness of human beings in the face of death.

Our games are also an exploration of deep human fears and curiosity, asking the experiencers to ponder the meaning of life and at the same time fantasize about the world after death

Game background:

One night, the hero falls into an endless nightmare. In his dreams, the deepest fears of reality haunted him as nightmares of all kinds. Fear flooded his mind, just as he was desperate, a mysterious old man appeared and said to him: "If you want to escape this endless nightmare cycle, you can choose a person to share your dreams with you and accept my trial." If we succeed in this challenge, we will break the curse of this nightmare."

After accepting the challenge of the old man, the hero and his partner experienced a strange world in the dream, met a thousand strange monsters, and was deeply trapped in the reincarnation of nightmares. They must constantly overcome the monsters in their dreams and do their best to break the curse. Each struggle is a double test of heart and courage, can they finally escape this endless nightmare cycle and meet the dawn of dawn again? There are obvious shortcomings in the surface.

The game we plan to develop is called "Yesterday's Nightmare," which is inspired by the great river of the Underworld in Greek mythology and the exploration of death and the meaning of life. The background of the game is set in an endless nightmare, in which the protagonist and his partners constantly challenge the treacherous world and monsters in the dream, trying to break the curse of the nightmare and meet the dawn again. This game not only combines reality and fantasy elements, but also conveys the loneliness and helplessness of human beings in the face of death through strong color contrast and dim light and shadow effects.

In "Yesterday's Nightmare", players will experience a variety of adventure, sports, shooting, level and role playing elements. Each level is a double test of heart and courage, players need to fight to defeat the monsters in the dream, and gradually crack the cycle of nightmares.

To enhance the game's appeal and replayability, we introduced a compelling storyline. This will not only make Yesterday's Nightmare stand out in a highly

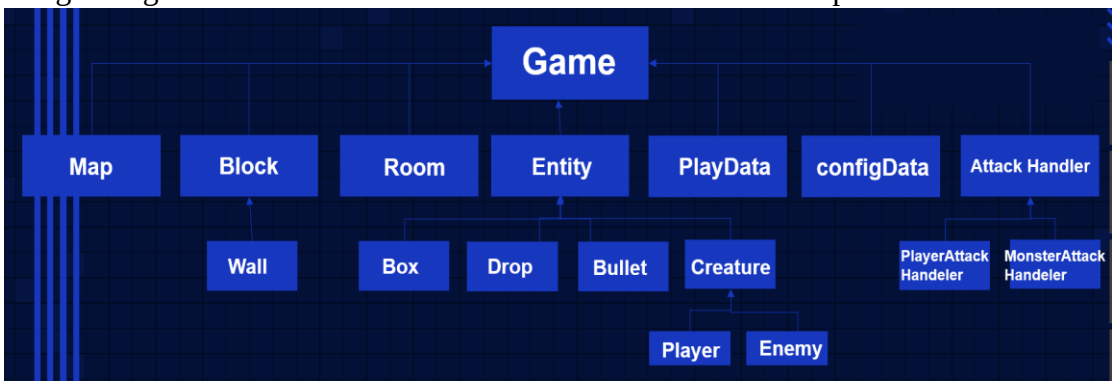
competitive gaming market, but will also increase player engagement and interest in the game. Through the alternation of reality and dreams, players will constantly ponder the meaning of life and the nature of death, and experience deep emotional resonance.

In addition, the project was a great opportunity to practice and extend what we learned in our C++ courses. By applying C++ and related software development techniques to real projects, 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 enhance our ability to solve complex technical problems, laying a solid foundation for more advanced software development work in the future.

Through this project, we hope to be able to provide a high quality gaming product that meets the market demand for innovative video games. At the same time, the promotion of teamwork and innovative thinking will also contribute to our academic and professional development. During the development process, team members will have the opportunity to delve into multiple aspects of game design, from user interface design to back-end logic processing, to improve the technical level of individuals and teams across the board.

1.2 System Objectives

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

Data Display and Update:

Display Player Data: The system must display various player game statistics on the interface, such as total kills, highest kills in a single round, highest level achieved, and shortest time taken to complete a level. This is achieved using functions such as outtextxy to render text on the screen. For example, displaying "Total Kills" and "Highest Single Round Kills" on the game interface involves converting data to the appropriate format and rendering it at specific coordinates.

Update and Sync Data: The system must synchronize local player data with the server and save it to local files. This ensures that the player's progress and statistics are up-to-date and stored safely. Functions like pdata.updateToServer() and pdata.save() are used to upload and save the game data, respectively.

Configuration Management:

Read/Create Configuration Files: The system needs to manage game configurations by reading from or creating a configuration file named config.gm. This file includes various settings such as sound effects, which are essential for customizing the game experience. Effective configuration management ensures that user preferences and settings are maintained across gaming sessions.

Game State Management:

Manage Game States: The system must handle different game states, including starting the game, pausing, and restarting. Efficient state management is critical to providing a seamless gaming experience. Functions and logic are implemented to transition between these states smoothly, ensuring that the game responds appropriately to user inputs and actions.

Display Server Status: In the game menu, the system should show the server connection status. This is crucial for informing players about the current state of their connection to the game server, impacting their ability to sync data and play online. The showServerStatus function is used to display this information.

User Interaction:

Menu Options and Selection: The system provides interactive menu options allowing players to confirm or cancel actions. For instance, when a player selects an option, the text color changes to provide visual feedback, enhancing the user experience. This involves using functions to set text color based on the selection state.

Handle User Input: The system must process user inputs to update player data and download the latest data from the server. When a player enters their name or other information, the system updates the local data structure and syncs it with the server.

This ensures that the latest player information is always available.

Visual Effects:

Play Animations: At the start of the game, the system should play animations using the videoCreater function to enhance the visual appeal and provide an engaging introduction. This function is responsible for displaying video content at specified coordinates on the screen, adding to the game's immersive experience.

Show Animated Gifs: In the game menu, the system utilizes the gifCreater function to display animated gifs. This adds dynamic visual elements to the menu, making it more attractive and interactive. The function specifies the gif file, position, size, and other parameters to control how the animation is presented.

2.2 Duty Assignments
Zhang yue: User interface Design:
Zhang xue yi: Level design; Inheritance and polymorphism of design classes
Huang zi wen: After the class design is completed, initialization creation, file reading and other operations are carried out

Everyone does the same amount of work!!!

3. Reference

None