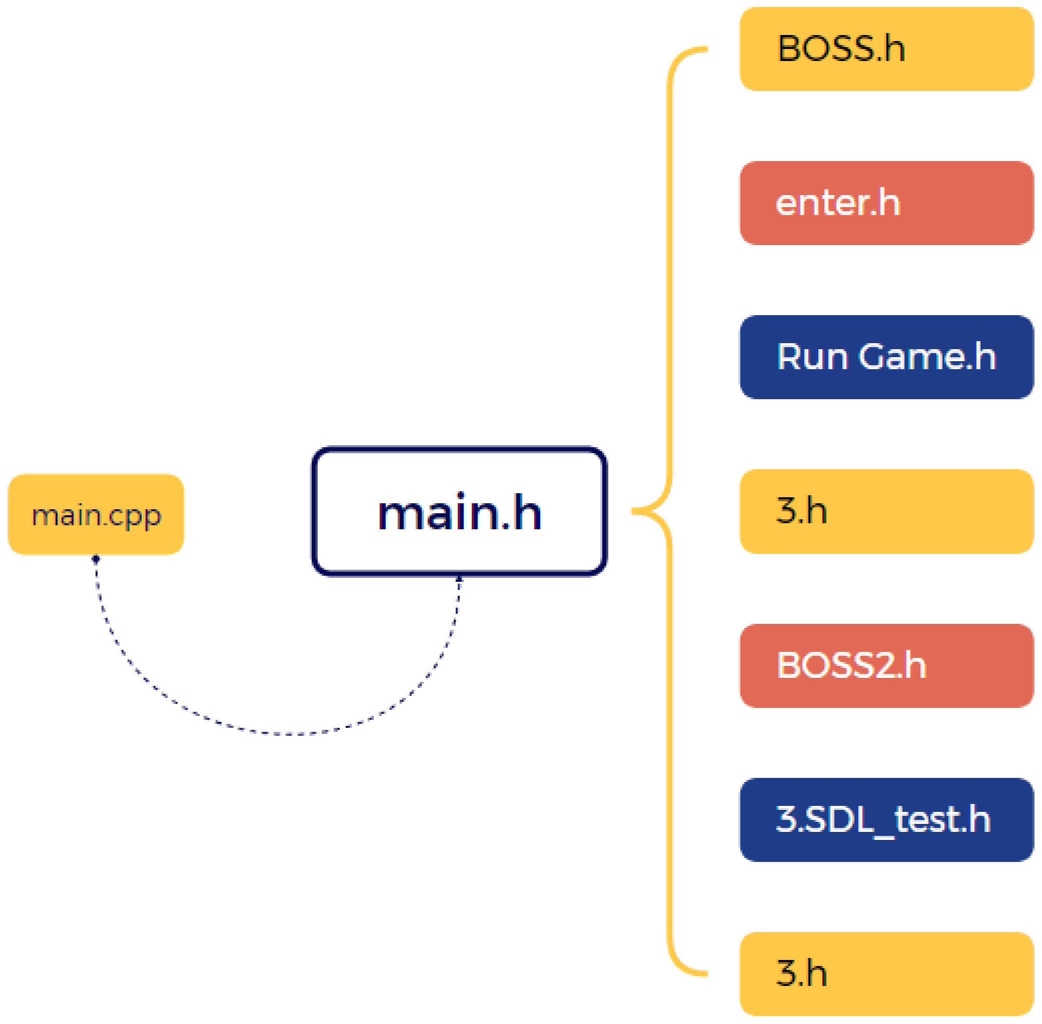
"Illuminati: The End of the World"

by Virtual Visionaries

How to Install the Game:

- method1:The compressed package contains a directly executable file of. exe, click here to run it. (maybe have not finish yet);
- method2:
- i. First, make sure to install Visual Studio (not Visual Studio Code), the newest version is sure to be better.
- ii. Second, download the SDL2 File that is provided in our Read-Me File.
- iii. Next Open the game file (the file contains an .sln at the end).
- iv. Once you are in Visual Studio go to project and press BirchEngine Properties.
- v. Now its time to configure the code but before that make sure you set so it is x64. Go to C/C++, press Additional Include Directories, edit.
- vi. Once you are in add a directory, and locate to all your SDL2 Files which are SDL2, SDL2 Image, SDL2 Mixer, SDL2 ttf, and select the include file inside.
- vii. After you have finished doing that, you can go to the Linker and edit the Additional Library Directories, Do the same with the Additional Include Directories, but this time when you open /lib, you will see x86 and x64(choose x64), and link the file.
- viii. Once you have finished doing that you can go to the Input section in Linker Additional Dependencies, and type SDL2.lib, SDL2main.lib, SDL2_image.lib, SDL2_mixer.lib, and SDL2_ttf.lib. Once you are done with all that make sure to press apply.
- ix. Finally, Go to Birch Engine that is located in the Solution Explore, right click and press open folder in File Explorer. Once you are in File Explorer put SDL2.dll, SDL2_mixer.dll, SDL2_ttf.dll, SDL2_image.dll (all of which you can find in the lib file of each SDL2).

Project Introduction



core of game

Core of the game (main.cpp)

Within the code, there is a main function code that works as the operating system of the program, however using SDL2, a type of header and source code that is necessary to run and maintain the game so it can run smoothly.

Project 1

dialogue

brief introduction

This project demonstrates how to use the SDL library to create a dialog interface and play audio files within it. By clicking on the dialog box, you can switch to another interface to display the image tutorial.

Functional characteristics

- Display the dialog box interface, displaying the text content word by word and sentence by sentence. Click on the dialog box to load all the text.
- Play audio files and pause or stop when needed.
- Clicking on the dialog box allows you to switch to another interface to display images.

Environmental requirements

- C compiler (such as gcc)
- SDL2 library
- SDL2_image library
- SDL2_ttf library

Installation steps

1. Install the SDL2 library and related development packages.

- Ubuntu: `sudo aptinstall libsdl2 dev libsdl2 image dev libsdl2 ttf dev`
- MacOS: Installing using Homebrew: `brew install sdl2 sdl2_image sdl2uttf`
- Windows: Download the SDL development library and configure it into the compilation environment.

2. Compile the program.

```
gcc main.c -o main -lSDL2 -lSDL2_image -lSDL2_ttf
```

Usage

1. Run the program.

In the program interface, the dialog box will display the content word by word and sentence by sentence, while playing audio.

3. Click on the dialog box area, and the interface will switch to displaying images.

Example

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <stdio.h>
#include <SDL2/SDL_ttf.h>
#include <stdlib.h>
#include <string.h>
int main(int argc,char *argv[]) {

    SDL_Init(SDL_INIT_AUDIO);
    SDL_Window *window = SDL_CreateWindow("Typewriter Effect", SDL_WINDOWPOS_UNDEFINED, SDL_WINI
    SDL_Renderer *renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED);
    TTF_Font *font;
    if(TTF_Init()<0){
        SDL_Log("TTF_Init failed:%s",SDL_GetError());
    return -1;
      }
    char a[100] = {"i am ljy ,this is a fridge"};
    char d[100] = {"i am dc ,this is not a fridge ,genius,damn"};
    char e[100] = {"i am wcy , this is a firdge"};
    char f[100] = {"i am wzm ,i agree with dc"};
    //一行最多50个字符（包括有空格）不然容易超出范围
    char b[100]={""};
    char c[100]={"dragon"};
    initphoto (window,renderer);
        SDL_Event event;
    while(1){

        if (SDL_PollEvent(&event)){
            switch(event.type){
                case SDL_QUIT:
                return 0;
            }

        }
                        present(window,renderer,font,a,b,c,"ljy.png");//只有这里用了
                        present(window,renderer,font,d,b,c,"dc.png");//只有这里用了
                        present(window,renderer,font,e,b,c,"wcy.png");//只有这里用了
                        //show_instruction(renderer,"ljy.png","wcy.png","wzm.png","dc.png","dia
                        present(window,renderer,font,f,b,c,"wzm.png");//只有这里用了
    }

    return 0;
}
```

Project 2

enter

Introduction

This project is an interface loading program implemented using the SDL library, which includes displaying logos, loading game logos, and saving and saving operations.

Functional characteristics

- Display team logo, display game logo.
- Play audio files and pause or stop when needed.
- Click enter to load the save and enter the game.

Environmental requirements

- C compiler (such as gcc)
- SDL2 library
- SDL2_image library
- SDL2_ttf library

Installation steps

1. Install the SDL2 library and related development packages.

- : `sudo aptinstall libsdl2 dev libsdl2 image dev libsdl2 ttf dev`
- MacOS: Installing using Homebrew: `brew install sdl2 sdl2_image sdl2uttf`

- Windows: Download the SDL development library and configure it into the compilation environment.

2. Compile the program.

   ```
   gcc main.c -o main -lSDL2 -lSDL2_image -lSDL2_ttf
   ```

Usage

1. Run the program.
2. In the program interface, the logo will be rendered, along with the loading interface and the 'press enter' option`
3. Click enter, the interface will switch to the game interface and automatically save and load saves.

Example

```c
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <stdio.h>
#include <SDL2/SDL_ttf.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char* args[]) {
    if (setwindows() != 0) {
        return -1;
    }
    // 加载图片
    renderTextOneByOne("Virtual Visionaries");

    init_audio();
    initphoto();


    loadGame(&gameData, currentSave);
    printf("存档 %d 加载成功\n", currentSave);
    printf("目前人物位置x: %d,目前人物位置y: %d, 地图:%d\n",gameData.playerplacex,gameData.playerpl


    char entertext[]="press enter";
    SDL_Color color={100,100,100};
    SDL_Event event;
    textFadeInOut(win,rdr,font,entertext,color);
    // 等待窗口关闭
    while (SDL_PollEvent(&event))
        {

            if (event.type == SDL_QUIT) {
                saveGame(gameData, currentSave);
                printf("存档 %d 保存成功\n", currentSave);
                printf("目前人物位置x: %d,目前人物位置y: %d, 地图:%d\n",gameData.playerplacex,gameD

                break;
        }
    }
    closeSDL();
    return 0;
}
```

Project 3

Pause

Introduction

This project is a game pause interface implemented using the SDL library, where there are options for game operations, returning to the game, and exiting the game.

Functional characteristics

- Click 'esc' to enter the pause interface.
- A tutorial on loading games, which can be selected independently.
- Click 'quit' to exit the game.
- Click 'back' to return to the game.

Environmental requirements

- C compiler (such as gcc)
- SDL2 library
- SDL2_image library
- SDL2_ttf library

Installation steps

1. Install the SDL2 library and related development packages.

- Ubuntu: `sudo aptinstall libsdl2 dev libsdl2 image dev libsdl2 ttf dev`
- MacOS: Installing using Homebrew: `brew install sdl2 sdl2_image sdl2uttf`

- Windows: Download the SDL development library and configure it into the compilation environment.

2. Compile the program.

   ```
   gcc main.c -o main -lSDL2 -lSDL2_image -lSDL2_ttf
   ```

Usage

1. Run the program.
2. In the program interface, click 'esc' to enter the paused interface, and then click again to exit.
3. Click 'quit' to exit the game.
4. Click 'back' to return to the game.
5. Click on 'instruction' to load the game's operation tutorial, which can be selected independently.

Example

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <stdio.h>
#include <SDL2/SDL_ttf.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char* argv[]) {
    if (!initSDL() || !loadImages()) {
        fprintf(stderr, "Failed to initialize SDL or load images.\n");
        return 1;
    }
    Uint32 frameStart, frameTime;
    int k=1;
    bool quit1=true;
    while (quit1) {
        frameStart = SDL_GetTicks();

        // 清除屏幕
        SDL_SetRenderDrawColor(renderer, 0, 0, 0, 255);
        SDL_RenderClear(renderer);

        // 渲染当前帧的图像
        SDL_RenderCopy(renderer, images[currentFrame], NULL, NULL);
        SDL_RenderPresent(renderer);

        while (SDL_PollEvent(&event)) {
            if (event.type == SDL_QUIT) {
                goto end;
            }

            if(event.type == SDL_MOUSEBUTTONUP){
                    SDL_Log("SDL_MOUSEBUTTONDOWN x = %d, y = %d, button = %d, clicks = %d",
                        event.button.x, event.button.y, event.button.button, event.button.click
                    SDL_Point pt = {event.button.x, event.button.y};
                    if(SDL_PointInRect(&pt,&instruction)){
                        show_instruction(renderer,"photo/aRRow_4.png","photo/aRRow_3.png","phot
                        //quit1=false;
                    }
                    if(SDL_PointInRect(&pt,&quit)){
                        goto end;
                    }
                    if(SDL_PointInRect(&pt,&back)){
                        goto end;
                    }
            }
        }
        // 更新当前帧
        if(currentFrame<IMAGE_COUNT&&k==1){
            currentFrame = (currentFrame + 1) % IMAGE_COUNT;
            }
        if(currentFrame>=IMAGE_COUNT-1){currentFrame=100;
        k=2;}
        if(currentFrame<IMAGE_COUNT&&k==2){
            currentFrame = (currentFrame + 1) % IMAGE_COUNT;
        }

        // 控制帧率
        frameTime = SDL_GetTicks() - frameStart;
        if (frameTime < 1000 / FRAME_RATE) {
            SDL_Delay(2000 / FRAME_RATE - frameTime);
        }

    }
end:
    closeSDL();
    return 0;
}
```

Project 4

Puzzle3 bomb disposal

Introduction

This project is a bomb disposal game implemented using the SDL library, where players need to dismantle bombs through different operations

Functional characteristics

-A unique bomb disposal game, coupled with circuit analysis, is addictive.

Environmental requirements

- C compiler (such as gcc)
- SDL2 library
- SDL2_image library
- SDL2_ttf library

Installation steps

1. Install the SDL2 library and related development packages.

- Ubuntu: `sudo aptinstall libsdl2 dev libsdl2 image dev libsdl2 ttf dev`
- MacOS: Installing using Homebrew: `brew install sdl2 sdl2_image sdl2uttf`
- Windows: Download the SDL development library and configure it into the compilation environment.

2. Compile the program.

```
gcc main.c -o main -lSDL2 -lSDL2_image -lSDL2_ttf
```

Usage

1. Run the program.
2. Click on the red button to turn on the switch.
3. Click on the green button to cut the wires short.
4. Yellow represents the OR gate.
5. Blue represents the AND gate.
6. It is required that the three green lights on the right side light up, and the red light does not light up throughout the entire process.

Example

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <stdio.h>
#include <SDL2/SDL_ttf.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char *argv[]) {
    // 初始化SDL
    if (!initSDL()) {
        return 1;
    }

    // 初始化所有元件的位置和状态
    // 开关
    switches[0].x = 360;
    switches[0].y = 212;
    switches[0].type = SWITCH;
    switches[0].state = false;
    switches[0].rect = (SDL_Rect){switches[0].x, switches[0].y, 20, 40};



    switches[1].x = 444;
    switches[1].y = 212;
    switches[1].type = SWITCH;
    switches[1].state = false;
    switches[1].rect = (SDL_Rect){switches[1].x, switches[1].y, 20, 40};

    switches[2].x = 535;
    switches[2].y = 212;
    switches[2].type = SWITCH;
    switches[2].state = false;
    switches[2].rect = (SDL_Rect){switches[2].x, switches[2].y, 20, 40};

    switches[3].x = 620;
    switches[3].y = 212;
    switches[3].type = SWITCH;
    switches[3].state = false;
    switches[3].rect = (SDL_Rect){switches[3].x, switches[3].y, 20, 40};
```

```
switches[4].x = 708;
switches[4].y = 212;
switches[4].type = SWITCH;
switches[4].state = false;
switches[4].rect = (SDL_Rect){switches[4].x, switches[4].y, 20, 40};

// 电线
wires[0].x = 583;
wires[0].y = 315;
wires[0].type = WIRE;
wires[0].state = true;
wires[0].rect = (SDL_Rect){wires[0].x, wires[0].y, 45, 10};

wires[1].x = 529;
wires[1].y = 453;
wires[1].type = WIRE;
wires[1].state = true;
wires[1].rect = (SDL_Rect){wires[1].x, wires[1].y, 45, 10};

wires[2].x = 413;
wires[2].y = 146;
wires[2].type = WIRE;
wires[2].state = true;
wires[2].rect = (SDL_Rect){wires[2].x, wires[2].y, 65,30};

// AND门
andGates[0].x = 410;
andGates[0].y = 390;
andGates[0].type = AND_GATE;
andGates[0].state = false;
andGates[0].rect = (SDL_Rect){andGates[0].x, andGates[0].y, 33, 30};

andGates[1].x = 600;
andGates[1].y = 375;
andGates[1].type = AND_GATE;
andGates[1].state = false;
andGates[1].rect = (SDL_Rect){andGates[1].x, andGates[1].y, 33, 30};

// OR门
orGates[0].x = 398;
orGates[0].y = 324;
orGates[0].type = OR_GATE;
orGates[0].state = false;
```

```
    orGates[0].rect = (SDL_Rect){orGates[0].x, orGates[0].y, 34, 14};

    orGates[1].x = 535;
    orGates[1].y = 379;
    orGates[1].type = OR_GATE;
    orGates[1].state = false;
    orGates[1].rect = (SDL_Rect){orGates[1].x, orGates[1].y, 33, 14};

    orGates[2].x = 659;
    orGates[2].y = 324;
    orGates[2].type = OR_GATE;
    orGates[2].state = false;
    orGates[2].rect = (SDL_Rect){orGates[2].x, orGates[2].y, 35, 14};

    // 运行游戏循环
    gameLoop();

    // 清理资源
    SDL_DestroyTexture(backgroundImage);
    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    SDL_Quit();

    return 0;
}
```

Boss1

-Header file:

BOSS.h

Used to cover the definitions and functions used in boss 1.

Installation

1. Copy the header file to your project directory.
2. Include the header file in your source file:

```
#include "BOSS.h"
```

defination

```
#define W 1250
#define H 650
extern SDL_Window* Puzzle1_Window;
extern SDL_Renderer* Puzzle1_Renderer;
#define win Puzzle1_Window
#define rdr1 Puzzle1_Renderer
```

function

```
int Boss(int ct);
int Random_atk_Mod(int atk_Round);
int Random_Star_Position(int atk_Round);
int fROg(int count_of_Frog, int Stop_frOG);
int Flies(int atk_Scd, int atk_Dly, int atk_Mod, int atk_Num);
void Fly_Circle(int* F_x, int* F_y, int atk_Scd, int atk_Dly, int atk_Num, int which_FLY);
void Fly_Crash(int* F_x, int* F_y, int atk_Scd, int which_FLY, int atk_Dly);
int wall(int i);
void TestText(int image_index);
int ColliSion(int x1, int y1, int w1, int h1, int x2, int y2, int w2, int h2);
void ID_CarD(int Num_of_Stars);
int bOss1_For_sciENtist2();
void cLaEn_All_iN_bOSs1();
```

- The function explanation is presented in boss1.cpp.

boss2

- Header file:

BOSS2. h

Used to cover the definitions and functions used in boss2.

Installation

1. Copy the header file to your project directory.
2. Include the header file in your source file:

```
# Include "BOSS2. h"
```

Definition

```
#define W 1250
#define H 650
extern SDL_Window* Puzzle1_Window;
extern SDL_Renderer* Puzzle1_Renderer;
#define win Puzzle1_Window
#define rdr1 Puzzle1_Renderer
```

function

```
int bOss2_For_dRaGoN(int sTaTe_oF_bOSs);
int bIg_bOMb_iS_cOMiNG(int atk_Scd, int atk_Round);
int On_The_Floor_Or_Not(int Circle_Center_x, int Circle_Center_y);
int bOMbs_Controller(int nUm_oF_bOMbs, int atk_Scd, int atk_Round);
int Boss2(int ct);
void TestText(int image_index);
void cLeAn_All_iN_bOSs2();
int eXtRa_bIg_bOMbS(int atk_Scd, int atk_Round);
```

-The function explanation is presented in boss2.cpp.

project

Parkour

Introduction

This project demonstrates how to use the SDL library to create a game that allows players to manipulate characters to jump up and down to avoid oncoming lasers, while recording the player's survival time through a timer and displaying the amount of health while deducting health.

Functional characteristics

- Click the spacebar to elevate the character
- Combo can make characters jump faster, and the game has a gravity field
- Lose health and display health after hitting the laser
- After jumping and surviving for a certain period of time, the level will be cleared

##Environmental requirements

- C compiler (such as gcc)
- SDL2 library
- SDL2_image library
- STDBOOL library
- Stdio library
- Stdlib library

Installation steps

1. Install the SDL2 library and related development packages.

- Ubuntu: `sudo aptinstall libsdl2 dev libsdl2 image dev libsdl2 ttf dev`
- MacOS: Using Homebrew to install: `brew install sdl2 sdl2_image`
- Windows: Download the SDL development library and configure it into the compilation environment.

2. Compile the program.

Usage

1. Run the program.

In the game interface, the laser will come from the right side and players need to press the spacebar to avoid the laser.

3. If you encounter a laser, you will lose health, and if you survive until the end, you will win.

Example

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <stdbool.h>
#include<stdio.h>
#include<stdlib.h>
// 定义屏幕宽度和高度
#define SCREEN_WIDTH 1250
#define SCREEN_HEIGHT 650
#undef main
// 主函数
int main(int argc, char* argv[]) {
    initialize();  // 初始化游戏
    life=24;
    while (running) {

        SDL_Event event;
   // SDL_RenderPresent(renderer);  // 更新屏幕显示
     if(pipe_x==SCREEN_WIDTH)cout=generate_pipe_height();
     if(cout>=600)cout=generate_pipe_height();
     top_pipe_height=cout;
        while (SDL_PollEvent(&event)) {
            if (event.type == SDL_QUIT) {
                running = false;  // 如果接收到退出事件，停止游戏循环
            } else if (event.type == SDL_KEYDOWN) {
                if (event.key.keysym.sym == SDLK_SPACE) {
                    if(bird_velocity>=0 ){
                     bird_velocity = -10;  // 处理空格键按下事件，使小鸟向上跳
                    }else bird_velocity -=8;
                }
            }
        }
          update();  // 更新游戏状态
        if(time++<=625){
            bottom_pipe_height=440- top_pipe_height ;
            gethurt=meet( bird_x, pipe_x, pipe_x2, top_pipe_height, bottom_pipe_height, top_pipe
            if(!gethurt){
                gethurt=meet( bird_x, pipe_x, pipe_x2, top_pipe_height, bottom_pipe_height, top_
            }
            if(gethurt==60){
                life--;
            }
            if(gethurt)gethurt--;
            tu_x1-=10;
            tu_x2-=10;
            if(tu_x1<=-1250)tu_x1=0;
            if(tu_x2<=0)tu_x2=1250;
           render();  // 渲染游戏画面
        }else if(time>625&&time<=2500){
            bottom_pipe_height=470- top_pipe_height ;
            gethurt=meet( bird_x, pipe_x, pipe_x2, top_pipe_height, bottom_pipe_height, top_pipe
            if(!gethurt){
                gethurt=meet( bird_x, pipe_x, pipe_x2, top_pipe_height, bottom_pipe_height, top_
            }
            if(gethurt==60){
                life--;
            }
            if(gethurt)gethurt--;
           top_pipe_height2=200;
           bottom_pipe_height2=250 ;
            pipe_velocity=15;
            if(pipe_x>=-100&&pipe_x<=650){
                pipe_x2=pipe_x+600;
            }else{
                pipe_x2=pipe_x-750;
            }
            pipe_x2 -= pipe_velocity;  // 更新管道的水平位置
            tu_x1-=10;
            tu_x2-=10;
            if(tu_x1<=-1250)tu_x1=0;
            if(tu_x2<=0)tu_x2=1250;
        render2();
        time ++;
        }else if(time>2500){
            //win!!
        }
        if(life<0) {//GAME OVER!!
           cleanup();
           return 0;
        }
        SDL_Delay(16);  // 控制帧率，避免过快渲染
    }
    cleanup();  // 游戏结束，执行清理操作
```

```
    return 0;
}
```

Enter

- Header file:

Enter. h

Used to cover the definitions and functions used in enter.

##Installation

1. Copy the header file to your project directory.
2. Include the header file in your source file:

```
# Include "enter. h"
```

Definition

```
extern SDL_Window* Puzzle1_Window;
extern SDL_Renderer* Puzzle1_Renderer;

//SDL_Window *win=NULL;
SDL_Surface *surf=NULL;
//SDL_Renderer *rdr=NULL;
TTF_Font *font=NULL;
SDL_Texture *imageTexture = NULL;

Uint8 *audio_buf_1;
Uint32 audio_len_1;
Uint32 audio_pos_1 = 0;
SDL_AudioDeviceID device_id_1;//以上四个音量所需

int flag=0;




#define SAVE_FILE "S/gamesave.dat"
#define NUM_SAVES 3
#define win Puzzle1_Window
#define rdr Puzzle1_Renderer
#define init_audio() init_audio_1();
#define audio_buf audio_buf_1
#define audio_len audio_len_1
#define audio_pos audio_pos_1
#define device_id device_id_1
// 游戏数据结构
typedef struct {
    int playerplacex;
    int playerplacey;
    int map;
    int disk;
    bool s2;
    bool ID;
    int is_time_machine_dialogue;
    int boss1_result;
    int boss2_result;
    int lose_count;
    int parkour_result;
    // 其他游戏数据
} GameData;

GameData gameData;
int currentSave = 1; // 当前选中的存档编号
```

function

```
void loadGame(GameData *data, int saveNum);
void saveGame(GameData data, int saveNum);
int setwindows();
void renderTextOneByOne(char *text) ;
SDL_Texture* loadTexture(const char *path) ;
void closeSDL();
int initphoto();
void callback_1(void *userdata, Uint8 * stream, int len);
void init_audio_1();
int textFadeInOut(SDL_Window* window, SDL_Renderer* renderer, TTF_Font* font, const char* text,
int Load_IN();
```

- The function explanation is presented in enter. h.

3

- Header file:

3. h

Used to cover the definitions and functions used in puzzle3.

Installation

1. Copy the header file to your project directory.
2. Include the header file in your source file:

```
# Include "3. h"
```

Definition

```
#define SCREEN_WIDTH 1250
#define SCREEN_HEIGHT 650
extern SDL_Window* Puzzle1_Window;
extern SDL_Renderer* Puzzle1_Renderer;
#define window Puzzle1_Window
#define renderer Puzzle1_Renderer
#define initSDL() initSDL_1()
typedef enum {
    SWITCH,
    WIRE,
    AND_GATE,
    OR_GATE
} ComponentType;

typedef struct {
    int x, y;
    ComponentType type;
    bool state;
    SDL_Rect rect;
} Component;
```

function

```
int Puzzle3();
void checkMouseClick(int mouseX, int mouseY);
void gameLoop();
bool initSDL();
bool loadBackgroundImage();
void renderComponent(Component component);
void renderphoto();
void renderwire();
```

- The function explanation is presented in enter. h.

Run Game

- Header file:

Run Game. h

Used to cover the definitions and functions used in enter.

Installation

1. Copy the header file to your project directory.
2. Include the header file in your source file:

```
#Include "Run Game. h"
```

Definition

```
#define SCREEN_WIDTH 1250
#define SCREEN_HEIGHT 650
extern SDL_Window* Puzzle1_Window;
extern SDL_Renderer* Puzzle1_Renderer;
#define window Puzzle1_Window
#define renderer Puzzle1_Renderer
```

function

```
void initialize();
void callback(void* userdata, Uint8* stream, int len);
void init_audio();
void update();
int meet(int bird_x, int pipe_x, int pipe_x2, int top_pipe_height, int bottom_pipe_height, int 
void render();
void render2();
void cleanup();
int run_forest_run();
```

- The function explanation is presented in Run Game.cpp.

SDL2_test

- Header file:

SDL2_test. h

Used to cover the definitions and functions used in SDL2_test.

Installation

1. Copy the header file to your project directory.
2. Include the header file in your source file:

```
#Include "SDL2_test. h"
```

Definition

```c
#define SDL_MAIN_HANDLED  // give back my main��
#define FR 40  // Frame Rate
#define FT 1000 / FR
#define MT 300  // One Motion Time(ms)

#define W 1250
#define H 650

const int Bullet_Speed = 5;  /*
This cannot be changed anymore!
The accuracy of targeted strikes is relatively high, while the slow and sluggish ones are fewer
*/
#define Bullet_Normal_Speed 5
#define Bullet_Max_Speed 10
#define Plane_Speed 5
#define No_Hit_Time 500
/*
*       Include
*/
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_mixer.h>
#include <stdbool.h>
/*
*       Structure
*/
typedef struct Player {
        int x;  // center point
        int y;
        int v_x;
        int v_y;
        int w;
        int h;
        int r;  // radius of circular collision box
}Player;

typedef struct Object {
        int x;  // top-left coordinates
        int y;
        int w;
        int h;
        int l_u[2];  // top-left coordinates
        int r_u[2];  // top-right coordinates.
        int l_d[2];  // bottom-left coordinates
        int r_d[2];  // bottom-right coordinates.
}Object;  // rectangle collision box structure

typedef struct Circle {
        int x;  // center coordinates
        int y;
        int r;
        int v_x;
        int v_y;
        bool is_exist = false;
}Circle;  // circle collision box structure

typedef struct Keycontrol {
        int w;
        int a;
        int s;
        int d;
        int E;
        char l_cmd;
}Keycontrol;  // Store the key command

typedef struct Puzzle1 {
        int register_row = 0;
        int register_column = 0;
        int player_row = 0;
        int player_column = 0;
        int puzzle_1_w = (1133 - 42) / 29 + (1133 - 42) * 1.5 / (29 * 22);
        int puzzle_1_h = (604 - 96) / 12 + (604 - 96) * 1.3 / (12 * 12);
        int round = 0;  // 0 represent player's turn 1 represent dragon's turn
        int is_controlled = 0;  // 1 represent has been (dragon��controlled  0 represent has n
        int result = 10; // 1 represents dragon win��0 repreents player win
        int last_result = 0;  // 0 win 1 lose
        int is_dialogue = 0;  // �Ƿ�h���
}Puzzle1;
```

function

```
void INIT();
void LOAD();
void PrintPlayer();  // print player's animation
void QUIT();
void Center_Text(int x, int y, int w, int h, const char* text);  // put the text in the center 
void Key_detect();
void Player_move(int start, int end);
void Map_0();
void Map_1(int start, int end);
void Map_2(int start, int end);
void Map_3(int start, int end);
void Map_4(int start, int end);
void Map_5(int start, int end);
void Map_6(int start, int end);
void Map_7(int start, int end);
void Map_8();
void Map_21();
void Map_22();
void Map_24();
void Puzzle_1();
void Bullet_INIT();  // init all bullet
void Are_You_Tough_Man();  // Plane_fight(endure for 30 seconds if you are a man)
void Bullet_Print();  // render bullet
void Map_Shift(int map_num, int x, int y);  // shifting map with a gradually disappear effect
void Puzzle_1_Init();  // init puzzle 1
int Is_SameQuadrant(Object object, int x, int y);  // detect if the rect are in the same quadra
int Is_collide(Object object, int player_w, int player_h, int x, int y, int r);  // detect if t
int Is_collide_Rect(Object object, int x, int y, int w, int h);  // detect if they(rect and rec
int Collision_Detect(int x, int y, int r, int start, int end);  // detect all object from start
void Object_INIT();  // init all object
void Object_Update();  // update all object
void Object_INIT_Point(int i);  // init all objects' points
void Plane_Object_Update();  // update the collision box of plane

//int textFadeInOut(SDL_Window* window, SDL_Renderer* renderer, TTF_Font* font, const char* tex
int Find_Max(int a, int b);  // find absolute maximum number from two int
void present(SDL_Window* window, SDL_Renderer* renderer, TTF_Font* font, char* a, char* b, char
```

- The function explanation is presented in SDL2_test.cpp.

test2

- Header file:

Test2. h

Used to cover the definitions and functions used in test2. cpp.

Installation

1. Copy the header file to your project directory.
2. Include the header file in your source file:

```
#Include "test2. h"
```

Definition

```
extern SDL_Window* Puzzle1_Window;
extern SDL_Renderer* Puzzle1_Renderer;
#define FRAME_RATE 60
#define IMAGE_COUNT 150  // ������10��ͼT
#define WINDOW_WIDTH 1250
#define WINDOW_HEIGHT 650
#define window Puzzle1_Window
#define renderer Puzzle1_Renderer
#define closeSDL() closeSDL_1()
```

function

```
bool initSDL();
int Pause();
void closeSDL_1();
bool loadImages();
int show_instruction(SDL_Renderer* renderer, char* next, char* before, char* yes);
```

- The function explanation is presented in test2.cpp.

Contribution Guide:

Thank you to programmers Lv Jiayang, Wang Chenyuan, Deng Chen, and Wang Zimo;
Thank you to Lamyae, Fatima Sajjad, and Ellie for their dedicated contributions.

Disclaimers

All content of this project is for reference and learning purposes only and does not constitute any form of legal, financial, or professional advice.
We make every effort to ensure the accuracy of information in the project, but do not guarantee its completeness, accuracy, timeliness, or applicability.
We are not responsible for any loss or damage caused by anyone's use or reliance on the information of this project.
This project may include links to third-party websites or resources, which are provided for user convenience only. We do not assume any responsibility for the content or reliability of these websites or resources.