THE LOST GIRL AND SIX COINS
FULL CODE EXPLANATORY DOCUMENTS

created by 闪电飞龙小队 - Lightning Dragon Team

1.1 Directory Structure

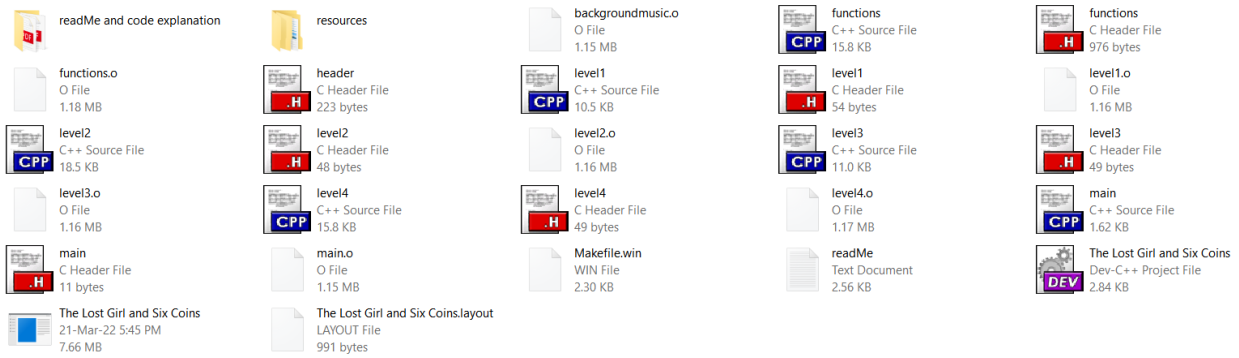
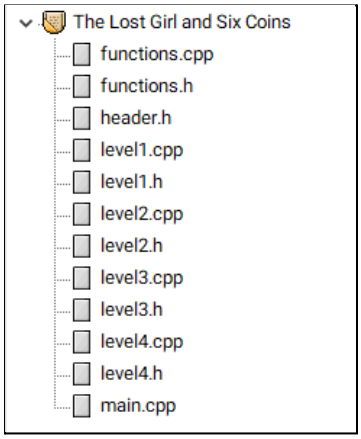


Image 1.1.1 - The zipped folder's content

Under the "**The Lost Girl and Six Coins**" zipped folder, several types of file are included, such as *.dev, .cpp, .h, .exe, .pdf*. The "*.h*" extension file stands for the library to be written in the code using "*#include*". Meanwhile, the "*.cpp*" extension stores all the C code to run the game. The "*.exe*" executes the compiled C code and runs the game in a new window. The "*readMe.pdf*" declares the general info, instructions to run the game, frequently asked questions, and much more useful information. Finally, a subfolder called "**resources**" stores the dialogues, tutorials, images, sound effects, and videos used throughout the game.



"**The Lost Girl and Six Coins.dev**" is the main project file that stores all the *.cpp* code, loads all external libraries, builds & compiles the game, and other many other crucial functions. Note that the project was built in Dev C++ 6.3 build systems. As for now, there are 12 files under the directory structure, namely functions.cpp, functions.h, header.h, level1.cpp, level1.h, level2.cpp, level2.h, level3.cpp, level3.h, level4.cpp, level4.h, main.cpp. Other external game development libraries include libEasyX.a, Libgdi32.a, libole32.a, msimg.dll, and winmm.dll.

Image 1.1.2 - The .dev project folder structure

1.2 Naming Scheme and Convention

The code adapted a pre-defined naming rule and convention for all the variables and functions used. This ensures the code is tidy and readable. The agreed naming rule is to

capitalize the first letter of each word, **except the first word**, and use no space between words. For example, livesLeft, playerWalk, and etc.

1.3 Code Explanation

The main code files that build the game are made of two types of files, ***.cpp* and *.h***. The *.cpp* file is straightforward, it contains all the lines of C code to compile and run the game. On the other hand, *.h* is mainly used to allow code partitioning. Partitioning code is a method used for making a large codebase or project manageable by breaking up different segments of it into smaller chunks that can be handled easily, as opposed to a large code that can have many areas of failure and take up large portions of a disk as well as take a very long time to compile.

In this project, the code is partitioned into several *.cpp* files for each level, menu, and header. The *.h* files act as a "bridge" that makes accessing functions written in other *.cpp* files possible. Each *.h* file must contain all the functions' names, return types, and arguments to function correctly. To access the functions from inside another *.cpp* file, simply include the target *.h* file by typing **#include "(*.h* file name)"** at the beginning of the code.

1.3.1 *main.cpp*

[*] main.cpp | header.h | level1.h | level1.cpp | functions.h | functions.cpp

```
#include "header.h"
#include "functions.h"
#include "level1.h"

int main() {

    initgraph(1280, 720);                                                                   // initialize graphic window (1280 = width, 720 = height)

    mciSendString(_T("open resources/video/game_intro.avi type MPEGVideo alias menuvideo"), NULL, 0, NULL); // open game intro video
    mciSendString(_T("open resources/sfx/cave.mp3 alias menusound"), NULL, 0, NULL);        // open music menu

    loadimage(NULL, _T("resources/image/menu_background.png"));                             // open menu image
    mciSendString(_T("play menusound"), NULL, 0, NULL);                                     // play music menu


    while (!_kbhit()) {                                                                     // Keep asking for input when keyboard has not been pressed
        char input = getKeyboardInput();                                                    // Press any key to continue from main menu
        if (input != NULL) {                                                                // if any key is pressed, load level 1
            cleardevice();
            mciSendString(_T("close menusound"), NULL, 0, NULL);                            // after pressing any button, close music menu
            mciSendString(_T("play menuvideo"), NULL, 0, NULL);                             // play game intro video
            settextstyle(16, 0, _T("Consolas"));                                            // set text font
            outtextxy(0, 0, _T("press any key in this window to start the game"));          // output text in screen
            outtextxy(910, 700, _T("press any key in this window to start the game"));
            _getch();                                                                       // press any key to load level 1
            mciSendString(_T("close menuvideo"), NULL, 0, NULL);                            // close game intro video

            loadLevel1();                                                                   // Start loading level 1
            break;
        }
    }

    _getch();
    closegraph();
}
```

Image 1.3.1.1 - main.cpp

The “*main.cpp*” includes several project-specific header files, *header.h, functions.h, and level1.h*. This file is mainly responsible for generating a new graphic window in HD resolution (1280x720) as well as loading the main menu image and music. Once the player has pressed any key or the video has finished playing, it will run the loadLevel1() function.

A clearer and more thorough line-by-line explanations of each code and function is available as comments in the *.cpp* file itself.

1.3.2 *level1.cpp*

[*] level1.cpp | level4.cpp | level2.cpp | functions.cpp | main.h | main.cpp | level1.h

```
#include "header.h"
#include "functions.h"
#include "level2.h"

void lvl1_resetPlayerPosition() {                           // this function is for reseting player position to the middle bottom of the screen
    IMAGE src;
    loadimage(&src, _T("resources/image/char_left2.png"));  // load character image char_left2 to IMAGE variable "src"
    transparentImage(NULL, 580, 630, &src, 0xffc4c4);       // load IMAGE variable "src" with transparent background to 580, 630
}

void loadLevel1() {

    // OPEN & PLAY MUSIC RESOURCES
    mciSendString(_T("open resources/sfx/background1.wav alias background1"), NULL, 0, NULL);
    mciSendString(_T("play background1"), NULL, 0, NULL);

    // LOAD DIALOGUE
    loadDialogue(1);

    // LOAD TUTORIAL
    loadLevelTutorial(1);

    // INITIALIZE LEVEL
    loadimage(NULL, _T("resources/image/level1_background.jpg"));   // load level 1 background
    int livesLeft = 5;
    loadLivesCounter(livesLeft);                                    // player starts with 5 lives

    // COINS VARIABLES
    int coin1PosX = 20;
    int coin1PosY = 440;
    int coin2PosX = 370;
    int coin2PosY = 250;
    bool coin1_isCollected = false;
    bool coin2_isCollected = false;
    loadCoin(coin1PosX, coin1PosY, coin1_isCollected);              // load coin1 image with the predefined variables
    loadCoin(coin2PosX, coin2PosY, coin2_isCollected);              // load coin2 image with the predefined variables
    int coinCounter = 0;
    loadCoinCounter(coinCounter);                                   // player starts with 0 coin collected

    // PLAYER POSITION VARIABLES
    int playerPositionX = 580;                                              // current player position in x-index
    int playerPositionY = 630;                                              // current player position in y-index
    int playerMovementCount = 0;                                            // number of times the player has moved
```

Image 1.3.2.1 - preview of level1.cpp

The “*level1.cpp*” includes several project-specific header files, *header.h, functions.h, and level2.h*. Firstly, the code loads and initialize all the necessary variables and functions, as well as runs the dialogue and tutorial cutscenes for level 1. It also initializes, loads, and checks for all the features in the level, including coins, remaining lives, speedboost, level help, lasers, lose, and the teleport conditions.

Secondly, the code loads the character image in the starting position in middle bottom of the screen utilizing the *transparentImage()* functions to remove the background of the image.

Next, it runs the main part of the code, which is the the character movement mechanic. This crucial mechanic affects all the aforementioned variables and functions.

For the player movement, the code check for “W A S D” keys input to move the character accordingly. “W” moves the character upward, “A” moves to the left direction, “S” moves downward and “D” moves to the right direction. For every player movement, it checks for all the affected variables and functions, the player-coin collision, player-laser collision, player-teleport collision, player-level help collision, and player-window border collision. If any of the collision are detected, it will run the appropriate function to update the UI shown to the player, such as the decrementing the remaining lives, incrementing the total of coins collected, showing level help, and many more. Lastly, the player is able to pause the game by pressing “ESC” key which allows them to resume, restart, or exit the game.

A clearer and more thorough line-by-line explanations of each code and function is available as comments in the *.cpp* file itself.

1.3.3 *level2.cpp*

level1.cpp | level4.cpp | level2.cpp | functions.cpp | main.h | main.cpp | level1.h

```
#include "header.h"
#include "functions.h"
#include "level3.h"

void lvl2_resetPlayerPosition() {                                   // this function is for reseting player position to the middle left of the screen
    IMAGE src;
    loadimage(&src, _T("resources/image/char_left2.png"));          // Load character image char_left2 to IMAGE variable "src"
    transparentImage(NULL, 20, 350, &src, 0xffc4c4);                // Load IMAGE variable "src" with transparent background to 20, 350
}

void loadLevel2(int livesLeft, int coinCounter) {

    // OPEN & PLAY MUSIC RESOURCES
    mciSendString(_T("open resources/sfx/background2.wav alias background2"), NULL, 0, NULL);
    mciSendString(_T("play background2"), NULL, 0, NULL);

    // LOAD DIALOGUE
    loadDialogue(2);

    // LOAD TUTORIAL
    loadLevelTutorial(2);

    // INITIALIZE LEVEL
    loadimage(NULL, _T("resources/image/level2_background.jpg"));   // Load background 2 image
    loadLivesCounter(livesLeft);                                    // Load lives counter based on the remaining lives from level 1

    // COINS VARIABLES
    int coin1PosX = 1200;
    int coin1PosY = 640;
    int coin2PosX = 50;
    int coin2PosY = 100;
    bool coin1_isCollected = false;
    bool coin2_isCollected = false;
    loadCoin(coin1PosX, coin1PosY, coin1_isCollected);              // Load coin1 image with the predefined variables
    loadCoin(coin2PosX, coin2PosY, coin2_isCollected);              // Load coin2 image with the predefined variables
    loadCoinCounter(coinCounter);                                   // Load coin counter based on the coins collected in level 1

    // PLAYER POSITION VARIABLES
    int playerPositionX = 20;                                       // current player position in x-index
    int playerPositionY = 350;                                      // current player position in y-index
    int playerMovementCount = 0;                                    // number of times the player has moved
    bool movement = false;                                          // movement false = player facing left
    int move = 5;                                                   // distance traveled (in pixel) for each player movement
```

Image 1.3.3.1 - preview of level2.cpp

The “*level2.cpp*” includes several project-specific header files, *header.h, functions.h, and level3.h*. Firstly, the code loads and initialize all the necessary variables and functions, as well

as runs the dialogue and tutorial cutscenes for level 2. It also initializes, loads, and checks for all the features in the level, including coins, remaining lives, speedboost, level help, wall, lose, and the teleport conditions.

Secondly, the code loads the character image in the starting position in middle left of the screen utilizing the *transparentImage()* functions to remove the background of the image. Next, it runs the main part of the code, which is the the character movement mechanic. This crucial mechanic affects all the aforementioned variables and functions.

For the player movement, the code check for “W A S D” keys input to move the character accordingly. “W” moves the character upward, “A” moves to the left direction, “S” moves downward and “D” moves to the right direction. For every player movement, it checks for all the affected variables and functions, the player-coin collision, player-wall collision, player-teleport collision, player-level help collision, and player-window border collision. If any of the collision are detected, it will run the appropriate function to update the UI shown to the player, such as the decrementing the remaining lives, incrementing the total of coins collected, showing level help, and many more.

In this level, player will need to use hole in each row to move between rows. The player’s current position affects all the boolean variables of each row. Player will need to collect all coins and proceed to the next level in under 15 seconds. Finally, the player is able to pause the game by pressing “ESC” key which allows them to resume, restart, or exit the game.

A clearer and more thorough line-by-line explanations of each code and function is available as comments in the *.cpp* file itself.

1.3.4 *level3.cpp*

level1.cpp | level4.cpp | level2.cpp | functions.cpp | main.h | main.cpp | level1.h | level3.cpp

```
include "header.h"
include "functions.h"
include "level4.h"

oid lvl3_resetPlayerPosition() {                          // this function is for reseting player position to the middle bottom of the screen
    IMAGE src;
    loadimage(&src, _T("resources/image/char_left2.png"));      // load character image char_left2 to IMAGE variable "src"
    transparentImage(NULL, 580, 630, &src, 0xffc4c4);           // load IMAGE variable "src" with transparent background to 580, 630

oid resetPuzzle() {                                       // this function is for resetting the puzzle (load all lamp_off images)
    IMAGE lamp_off;                                       // load lamp_off image
    loadimage(&lamp_off, _T("resources/image/lamp_off.jpg"));

    transparentImage(NULL, 306, 296, &lamp_off, 0xffc4c4);      // load lamp_off image in position 1
    transparentImage(NULL, 447, 296, &lamp_off, 0xffc4c4);      // load lamp_off image in position 2
    transparentImage(NULL, 588, 296, &lamp_off, 0xffc4c4);      // load lamp_off image in position 3
    transparentImage(NULL, 729, 296, &lamp_off, 0xffc4c4);      // load lamp_off image in position 4
    transparentImage(NULL, 870, 296, &lamp_off, 0xffc4c4);      // load lamp_off image in position 5

ool loadPuzzleBox(int round) {                                  // take round argument, and output true/false depending on the user's input

    IMAGE puzzleBox;
    loadimage(&puzzleBox, _T("resources/image/level3_puzzlebox.jpg"));      // load puzzle box image
    transparentImage(NULL, 280, 207, &puzzleBox, 0xffc4c4);
    resetPuzzle();                                              // call the resetPuzzle() function

    Sleep(1000);                                                // wait for one second
    outtextxy(500, 480, _T("Press any key to start the sequence"));         // output text
    _getch();

    // PLAYER INPUT STORAGE VARIABLES
    char round1_input[10];
    char round2_input[10];

    if(round == 1) {                                                                    // for round 1
        mciSendString(_T("open resources/video/Round_1.avi type MPEGVideo alias round1"), NULL, 0, NULL);
        mciSendString(_T("play round1"), NULL, 0, NULL);                                // play round 1 puzzle video
        Sleep(7000);                                                                    // wait for 7 seconds
        mciSendString(_T("close round1"), NULL, 0, NULL);                               // close round 1 p

        InputBox(_T(round1_input), 10, _T("Enter Round 1 Sequence"));       // load input box and store the input in round1_input variable
```

Image 1.3.4.1 - preview of level3.cpp

The "*level3.cpp*" includes several project-specific header files, *header.h, functions.h, and level4.h*. Firstly, the code loads and initialize all the necessary variables and functions, as well as runs the dialogue and tutorial cutscenes for level 3. It also initializes, loads, and checks for all the features in the level, including coins, remaining lives, level help, puzzle key, lose, and the teleport conditions.

Secondly, the code loads the character image in the starting position in middle bottom of the screen utilizing the *transparentImage()* functions to remove the background of the image. Next, it runs the main part of the code, which is the the character movement mechanic. This crucial mechanic affects all the aforementioned variables and functions.

For the player movement, the code check for "W A S D" keys input to move the character accordingly. "W" moves the character upward, "A" moves to the left direction, "S" moves downward and "D" moves to the right direction. For every player movement, it checks for all the affected variables and functions, player-key collision, player-teleport collision, player-level help collision, and player-window border collision. If any of the collision are detected, it will run the appropriate function to update the UI shown to the player, such as the decrementing the

remaining lives, incrementing the total of coins collected, showing level help, loading the puzzle, and many more. In addition, the player is able to pause the game by pressing “ESC” key which allows them to resume, restart, or exit the game.

In this level, once the player collides with the key’s coordinates, it will trigger the loadPuzzleBox() functions. Next, the puzzle hint video will be played upon key press, and once the video is done showing, an input dialogue box will be appear for the player to input his answer. If the player input the right sequence, the loadPuzzleBox() function will be retriggered for the second time, loading the second round of the puzzle. In contrast, everytime the player fails to guess the order, they will lose one health. If the player manages to finisih both rounds of the puzzle, two more coins will be added to the coin counter and the player can proceed.

A clearer and more thorough line-by-line explanations of each code and function is available as comments in the *.cpp* file itself.

1.3.5 *level4.cpp*

level1.cpp | level4.cpp | level2.cpp | functions.cpp | main.h | main.cpp | level1.h | level3.cpp

```
#include "header.h"
#include "functions.h"

void lvl4_resetPlayerPosition() {                               // this function is for reseting player position to the top-left of the screen
    IMAGE src;
    loadimage(&src, _T("resources/image/char_left2.png"));      // load character image char_left2 to IMAGE variable "src"
    transparentImage(NULL, 300, 90, &src, 0xffc4c4);            // load IMAGE variable "src" with transparent background to 300, 90
}

void loadLevel4(int livesLeft, int coinCounter) {

    // OPEN & PLAY MUSIC RESOURCES
    mciSendString(_T("open resources/sfx/background4.wav alias background4"), NULL, 0, NULL);
    mciSendString(_T("play background4"), NULL, 0, NULL);

    // LOAD DIALOGUE
    loadDialogue(4);

    // LEVEL TUTORIAL
    loadLevelTutorial(4);

    // INITIALIZE LEVEL
    loadimage(NULL, _T("resources/image/level4_background.jpg"));

    // ITEMS VARIABLE
    int item1PositionX = 100;
    int item1PositionY = 110;
    int item2PositionX = 100;
    int item2PositionY = 335;
    int item3PositionX = 100;
    int item3PositionY = 550;

    bool item1IsCollected = false;
    bool item2IsCollected = false;
    bool item3IsCollected = false;

    bool item1 = false;
    bool item2 = false;
    bool item3 = false;
    int itemCounter = 0;

    // LOAD ITEM IMAGE
    IMAGE imgitem1;
```

Image 1.3.5.1 - preview of level4.cpp

The “*level4.cpp*” includes several project-specific header files, *header.h and functions.h*. Firstly, the code loads and initialize all the necessary variables and functions, as well as runs the dialogue and tutorial cutscenes for level 4. It also initializes, loads, and checks for all the features in the level, including coins, remaining lives, level help, boss, win, lose, speedboost, and the teleport conditions.

Secondly, the code loads the character image in the starting position in top left of the screen utilizing the *transparentImage()* functions to remove the background of the image. Next, it runs the main part of the code, which is the the character movement mechanic. This crucial mechanic affects all the aforementioned variables and functions.

For the player movement, the code check for “W A S D” keys input to move the character accordingly. “W” moves the character upward, “A” moves to the left direction, “S” moves downward and “D” moves to the right direction. For every player movement, it checks for all the affected variables and functions, player-boss collision, player-laser collision, player-items collision, player-drop off area collision, player-teleport collision, and player-window border collision. If any of the collision are detected, it will run the appropriate function to update the UI shown to the player, such as the decrementing the remaining lives, incrementing the total of coins collected, showing level help, picking and dropping items, and many more. In addition, the player is able to pause the game by pressing “ESC” key which allows them to resume, restart, or exit the game.

In this level, the player will need to move all items to the designated circle one-by-one. Player can pick up an item by standing on top of it and drop it by either dying to the boss/laser or getting inside the drop-off area. The boss will try to catch the player by calculating its current positions and comparing it that of the player’s. This allow the boss to chase the player according to the player movement. Once all items has been dropped off correctly, the player will win the game.

A clearer and more thorough line-by-line explanations of each code and function is available as comments in the *.cpp* file itself.

1.3.6 *functions.cpp*

level1.cpp | level4.cpp | level2.cpp | functions.cpp | main.h | main.cpp | level1.h | level3.cpp

```cpp
#include "header.h"
#include "level1.h"
#include "main.h"

void transparentImage(IMAGE* dstimg, int x, int y, IMAGE* srcimg, UINT transparentcolor) {  // this function is used to generate an image with transparent background
    HDC dstDC = GetImageHDC(dstimg);
    HDC srcDC = GetImageHDC(srcimg);
    int w = srcimg->getwidth();
    int h = srcimg->getheight();

    TransparentBlt(dstDC, x, y, w, h, srcDC, 0, 0, w, h, transparentcolor);
}

bool pause() {                                              // pause function

    ExMessage m;

    m = getmessage(EM_MOUSE | EM_KEY);                      // get mouse click input

        IMAGE pause;
        loadimage(&pause, _T("resources/image/pause_menu.png"));        // load pause menu image
        transparentImage(NULL, 0, 0, &pause, 0xffc4c4);

    switch(m.message)
    {
        case WM_LBUTTONDOWN:

        if(m.x >= 475 && m.x <= 805 && m.y >= 285 && m.y <= 340)        // resume button coordinate
        {
            return false;                                       // the game is no longer pausing
        }
        if(m.x >= 475 && m.x <= 805 && m.y >= 360 && m.y <= 420)        // restart button coordinate
        {
            mciSendString(_T("close all"), NULL, 0, NULL);
            main();                                             // run main menu
        }
        if(m.x >= 475 && m.x <= 805 && m.y >= 445 && m.y <= 500)        // exit button coordinate
        {
            mciSendString(_T("close all"), NULL, 0, NULL);
            closegraph();                                       // close game window
        }
    }
```

Image 1.3.6.1 - preview of level4.cpp

The "*functions.cpp*" file contains all the application-wide functions, that can be accessed throughout the project by including the "*function.h*" header files. The functions are as followed:

1. transparentImage()
2. pause()
3. loadEndMenu()
4. loadCoin()
5. loadLivesCounter()
6. loadCoinCounter()
7. loadHorizontalLaser()
8. loadVerticalLaser()
9. loadLevelTutorial()
10. loadDialogue()
11. getKeyboardInput()
12. playerWalk()
13. item1Move()
14. item2Move()

15. item3Move()
16. loadItemCounter()
17. bossMove()
18. playerWalkLevel4()

A clearer and more thorough explanations of each code and function is available as comments in the .cpp file itself.

1.3.7 *header.h*

header.h | level1.cpp | level4.cpp

```
1   #include <conio.h>
2   #include <graphics.h>
3   #include <stdio.h>
4   #include <stdlib.h>
5   #include <mmsystem.h>
6   #include <windows.h>
7   #pragma comment( lib, "MSIMG32.LIB")
8   #pragma comment(lib, "winmm.lib")
9   #include <time.h>
10
11
```

Image 1.3.7.1 - header.h

The "*header.h*" file contains all the application-wide headers and libraries, that can be accessed throughout the project by including the "*header.h*" header files. The headers and libraries are as followed:

1. <conio.h>
2. <graphics.h>
3. <stdio.h>
4. <stdlib.h>
5. <mmsystem.h>
6. <windows.h>
7. <time.h>
8. MSIMG32.LIB
9. winmm.lib

1.4 Level & Feature Documentations

Main Menu (main.cpp)

Intro Video (main.cpp)

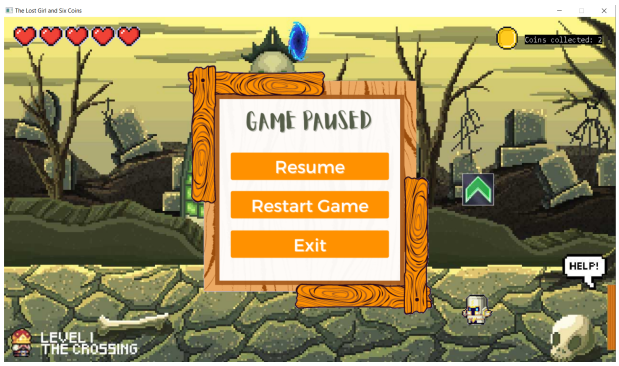
Dialogue Cutscene (functions.cpp)

Level Tutorial (functions.cpp)

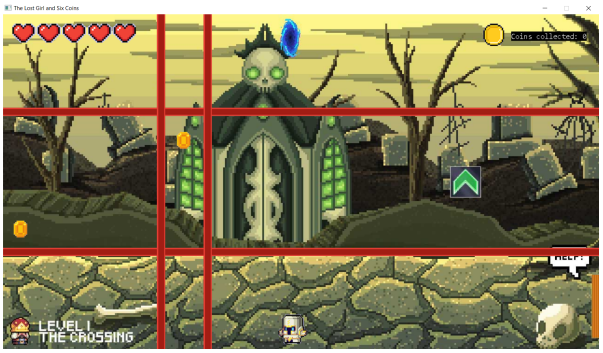
Lives Counter (functions.cpp)

Coin Counter (functions.cpp)

Level Help (level1.cpp)

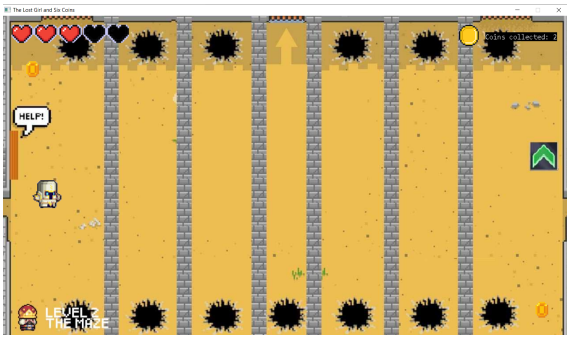


Game Pause (function.cpp)



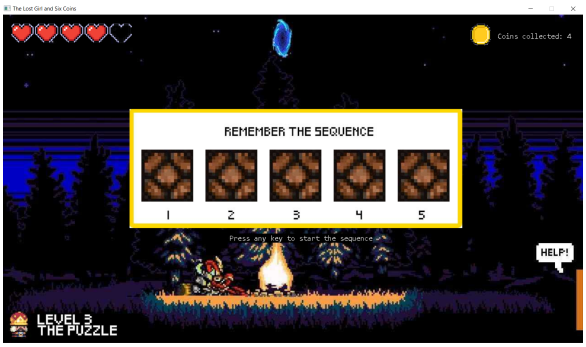
Level 1 (level1.cpp)

Player Movement (functions.cpp)

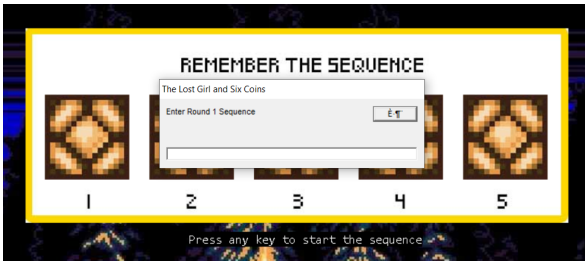


Level 2 (level2.cpp)

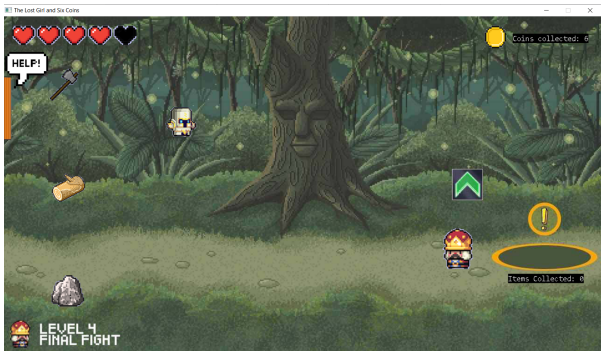
Level 3 (level3.cpp)



loadPuzzleBox (level3.cpp)

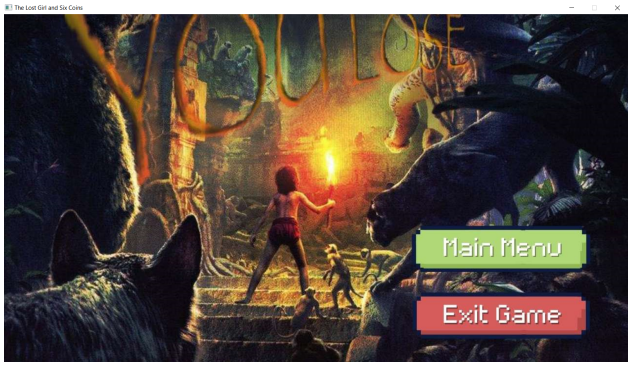


Input Dioalogue Box (level3.cpp)



Level 4 (level4.cpp)

Final Dialogue (functions.cpp)



Lose Menu (functions.cpp)

Win Menu (functions.cpp)