Dept. of Computer Science & Technology,

University of Dhaka.

Project Report

Fundamentals of Programming Lab (CSE-1211)

Project Name

Game.exe

Game Name

Hunt

Submitted By

Group-3

Team Members

Alve Reduan (FH 59)

Raufun Nazin Srizon (FH 49)

Mustakim Hossain Jim (FH 53)

INTRODUCTION

This is a single player 2D platform based game. Where your character moves/jumps/climbs from platform to platform collecting coins and clearing enemies on the way. The game consists of 3 levels. If you get close to enemies or if you fail to jump/climb and fall your character will die and the game will give you the option to start from the current level you are in or from the start or quit. To kill enemies your character will have a pistol. Sometimes the enemies can be deceiving, they won't die after getting hit and this feature is set in random order. So the game might not look repetitive and even if you are doing speedrun you will need to do it carefully because you might need to shoot the enemy again. For graphical interference, SDL2 libraries have been used.

Game controls

The left and right cursor keys to move left and right you can also use A and D. To climb the ladder up cursor key or W and to climb down the down cursor key or S. You can jump with SPACE key and shoot with LMB. These are the controls of the game.

Game Enemies

The game consists of 3 types of enemies on each level except for the third. In the third level you will have 2 types of enemies and in the end we have a special boss fight, so a special one and only enemy. You can kill the enemies with your pistol or you can jump over them. But I wouldn't recommend avoiding them and jumping because the enemies are constantly moving and you will die getting near them. The enemies will have random hitpoints, some of them will require multiple bullets to kill. This feature helps to keep the game a little bit randomized. Though the placement of enemies will always be the same but health will not always be the same.

Objectives

Though there are 3 types of enemies on each level, the enemy types will not all be the same to make sure each level is unique in it's own way. Each level has a different background, different platform design, different props, different level based background music, But the idea of going from one place to another is the same. When your character dies you will be given the option to retry the same level from the beginning or the entire game from level 1or quit. Basically if you die the level will reset you will be back at the front and the enemies will return or you can always reset all your level progress and make your run perfect by starting all of it all over again from level 1.

Project Features

It is a 2D based platform game. Featuring 5 types of different enemies over 3 levels. Each type of enemy has a different type of death sound, each level has a different background music. Our character can shoot bullets, enemies die only when the bullet hits them, The bullet won't travel out of the current frame and kill enemies who are further away, our character can jump. The character can't move up or down unless it is near a ladder. Beware of cliffs, because if you fall you will die. Timing is crucial when you are jumping from platform to platform because if you miss the jump you will fall and you will die.

Project Modules

main.cpp

Game starts from this file. When the game launches main() function gets called.

Main function first sets some initial variable for the game states like if the game is over or has started or is won.

Then the initializerParamter() function is called. Which sets some initial variable value for window size and all that.

Then the init() function gets called. The init function initializes sdl library, sdl image Library, sdl mixer Library and get the player's level progress from saved.txt file. If everything loads fine this function returns true.

The init function calls init_level() function which loads calls functions to load current level background, character, enemy, block and bullet texture. If everything loads fine then this function returns true.

If init() function returns true then the program calls startScreen() function. Which loads the Start Screen texture. If texture is loaded the game continues otherwise the program stops.

If startScreen() returns true Background music is started repeatedly on loop. Then the game loops starts.

In the game loop, first we count the frame. We find the matrix index for player current position.

After that we look for events or inputs from the keyboard. We check for these input:

A or Left Arrow: Player goes left

D or Right Arrow: Player goes right

S or down Arrow: Player goes down

W or Up arrow: Player goes up

Space: Player Jumps

Before input is taken we check if the player at the current position can do that assigned operation or not. If that move is allowed at that position we call characterPositionHandle() which updates the character position.

After that we look for input from the mouse. Here check the game state and calculate the mouse position and do things accordingly.

If the game is started and the player clicks the left mouse button a bullet is fired and we play the sound of the bullet fired.

After this section we update the position or state of everything following the user inputs.

First, we check if the level is completed. That happens when the player reaches the end of the current level. As this happens, we increase the level if there is any or game won screen is shown. If There is a next level we increase the level variable and save it in a file using saveLevel() function. Then we load that level's background, enemy and all other elements by the init_level() function. Then we set the parameters for this level by initializerParam() function.

After that we check for y axis movement of the character.

If the player has jumped we update the player position accordingly.

If the player is in the air we call the gravity() function which moves the player position down.

Then we check for x axis movement.

If the player was moving right or left we update the player position by characterPositionHandle() function.

Then we check for enemy movement.

We update each enemy position by calling findEnemyLocation() function.

And then set the enemy movement direction according to the frame. Enemy rotates each second or each 60 frames.

Then we propagate the fired bullets.

After that we check if any bullets have hit any enemy. If any enemy is hit we set that enemy's enemyState to false which means that enemy is dead and wont render from this frame anymore.

Then we check if the character has been in contact with any enemy. If character has been in contact with any enemy character is dead and the game is over. We don't run the rest of the frame functionalities and in the next frame Game Over screen is loaded as gane_over has been set to true.

Then we play a set of frames for animation if character has fired any bullets.

After that we come to the render function.

Checking the current state, if game is won, Game won texGameWon texture is rendered.

If game is over texGameOver texture is rendered.

If Game has not started yet texStartScreen texture is rendered.

If game has started we render updated background, blocks, player, bullets and enemies.

We delay for (1000/FPS) microseconds to run the game in already set FPS frames per second.

After the game is quit, we call the close() function to free all the pointers and memory.

LoadEverything.cpp

A few functions are here which all load images for the game from directory files.

startScreen() function is called to load texture for start screen, Game over screen and Game Win image.

loadGameBackground() function takes a parameter of Player Current Level. According to this level, this function loads the game background image when called. If loading is successful this function returns true other prints an error message and returns false

loadGamePathway() function takes a parameter of Player Curren Level. According to this level, this function loads game blocks when called. If loading is successful this function returns true, otherwise prints an error message and returns false.

loadCharacter() function loads all the character frames and stores it in texcharacter array. It then loads all the image/frames textures of character shooting animation which is rendered by the main function when character shoots bullet.

loadMatrix() function takes a parameter of Player Curren Level. According to the level this function loads the matrix of the current level from the directory file.

loadEnemy() function takes a parameter of Player Curren Level. According to the level this function loads all the enemies and stores them in a texture array for enemies and returns true if loaded successfully otherwise prints an error message and returns false.

loadBullet() function loads the bullet image into texBullet texture and returns true if loaded successfully otherwise prints an error message and returns false.

loadSound() function takes a parameter of Player Curren Level. According to the level this function loads sound for bullet fired, enemy died and background and returns true if loaded successfully otherwise prints an error message and returns false.

CharecterMovement.cpp

This file contains two functions to update character position.

characterPositionHandle() takes some parameter to know which state the character is in and moves forward, backwards or up and down according to the state.

gravity () function takes no parameter.When this function is called character position goes down.

globalVariables.h

This function defines variables to be used across all the files.

Team Member Responsibilities

Alve Reduan: main.cpp

Raufun Nazin Srizon: LoadEverything.cpp, Image Graphic

Mustakim Hossain Jim: CharecterMovement.cpp

Platform, Library & Tools

Platform: Linux

Library: SDL2, SDL2_Image, SDL2_Mixer

Tools: VScode, Adobe Photoshop, Photopea

Limitations

The game is limited in certain ways. Player can shoot 10000 bullets in his play time. This particular number is possible to increase but that is a constant number.

When the player shoots while in the ladder the player goes straight up the ladder or goes down and dies, based on the player's direction.

Conclusions

We learnt about the sdl2 library, group development and building, and lots of bug fixing.

We had a lot of fun making this game.

We worked together as a team, and each of us was helpful to the other.

Our knowledge of C and C++ has grown, and finding the best solutions to problems like enemy character interaction has been fun to put into practice in our project.

Future plan

- More smooth gameplay
- Adding More Level
- Difficulty Level

Repositories

Github Repository:

https://github.com/reduan2660/Game_1111/tree/master/

Gameplay Video:

https://youtu.be/2lTyJydS8MY

References

http://lazyfoo.net/

https://stackoverflow.com/