CS 108 Project
Lost in the Maze: A Pygame Adventure

Ridam Jain
23B1049
Indian Institue of Technology Bombay

April 28, 2024

Contents

1 Introduction

This Project consists of building a Maze Game using Pygame, which is a cross platform set of Python modules designed for writing video games.

2 About the Game

The objective of the game is to navigate through a maze and reach the end point. The maze is a randomly generated network of walls and paths through which you need to navigate and reach the end point in order to win the game.
Moreover, the game also has a camera, which allows the user to see only a specific part of the maze. This makes the game both more exciting and difficult.
Moreover to make the game more engrossing, it consists of:

- An intuitive User Interface
- Maze having collectibles, powerups, enemies, traps
- Animated Sprites and Enemies
- Themes
- A High Score Leaderboard
- A Timer
- Background Music

3 Modules

The list of python modules used while making this game are:

3.1 Pygame

Pygame is the main module used in making the game. Pygame makes it easy to create games and multimedia applications in Python. It provides multiple functionalities for handling graphics, sound, user input methods (such as Keyboard and Mouse). You can easily detect keypresses, mouse movements, buttonclicks, hovers by mouse and many more. Pygame has functions for creating sprite groups, collision detections, creating game loops and hence very useful in creating games.

3.2 Sys

Sys provides access to some variables used or maintained by the Python Interpreter and functions that interact with the Python runtime environment.

- Sys was mainly used to exit the game when the Quit Button is pressed or the pygame window is quit.
- Sys was also used in taking custom music file as command line argument, and using default theme background music, if no command line input provided.

3.3 Random

The Random module in python provides functions for generating random numbers in a given range and also selecting random elements from lists, dictionaries, etc The use cases in the Maze Game are as follows:

- **Maze Generation**
  In the Maze generating algorithm, using DFS(Depth First Search), random was used in selecting a random unvisited cell from a set of neighbor cells
- **Collectibles**
  The collectibles, namely coins, gems, health are placed on the maze using randomly selected grid cells from a list of grid cells
- **Enemies**
  Firstly the enemies are spawned at random locations using Random module, and secondly, the movements of the enemies is also controlled by a random move selector again implemented using the Random module. Also the spawning of traps is also implemented using the Random module.

4 Directory Structure

The main submission directory is divided into multiple modules for making the code modular, readable, reusable. The main files are as follows:

- `button.py`
  This file contains implementation of the Button class, which is used thoroughly throughout the game. Since buttons are used in every screen of the game, this is a very essential class and file of the game. This button takes various attributes such as centerX, centerY of the button as well as the color, image of the button. The hovering feature is also implemented in the button so the text on the button changes colour on hovering.
- `cell.py`
  This file contains the basic class Cell. This class contains the properties of each cell of the grid of the maze. The basic declaration of the Cell contains its x and y indices, and the thickness of wall around it. This also contains some functions for the generation of the maze which is done in maze.py file.
- `collectibles_and_traps.py`
  This file contains 4 classes namely:
  - Coins
  - Gems
  - Health
  - Traps

  All these classes contain functions for declaring as well as updating the respective sprite objects. Each sprite object has its own x and y coordinates as well as frames for animating each respective sprite according to the time passed. Moreover the Trap class contains two different types of traps according to the theme chosen in the game.
- `enemy.py`
  This file contains the implementation of the enemy class. There are two types of enemies according to theme. If the theme is Jurassic then the enemies are flying dragons, if the theme is Zombie, then the enemies are the zombies. The implementation of these again involves

sprites in pygame. These are animated according to their direction of movement. In other words, these have different image frames for each direction of movement and these update instantaneously.

- `font.py`
  This is a small file for the implementation of a Pixel font for the whole game which helps in maintaining a uniformity throughout the game.

- `game.py`
  This file merely calls the Main Menu screen present in the screens.py file and initiates the game. It also takes in the music file path if provided by the user from command line. If none provided, then it takes it as default music preloaded in the resources section and played according to the theme selected. This has two functions. One for the leaderboard interface, and the other for the display of the leaderboard according to the level selected.

- `leaderboard.py`
  This file maintains the leaderboard and extracts the data of the leaderboard present in the leaderboards folder of the parent directory. The leaderboards are updated after each game session and displayed even at the end of the game session as well as in leaderboard tab under options tab. All the three levels have their individual leaderboards.

- `maze.py`
  This file contains the maze class, which contains the functions for generation of maze by Depth Search Algorithm. And also storing the maze solution path in a file named path.txt

- `screens.py`
  This file contains the main screens of the game such as:
  - Main Menu
  - Options
  - Play
  - Levels
  - Game Over Screens

- `sidebar.py`
  This file contains the blitting instruction of the sidebar which stores the game scores, instructions, health and time left on the side of the game screen. The timer also turns red when the time left is less than 10 seconds.

- `sound.py`
  This file has all the sounds loaded so that the sounds can be called by other screens just by calling the functions from this file and hence easing accessibility.

- `sprite.py`
  This is our main character class. Our main characters move according to the keyboard inputs provided to it. Also the animation frames change according to the direction of movement which gives a real character like feel and hence enhances the gaming experience. This also has collision detection implemented with the enemies, traps, coins, gems, health. All of which is implemented in the score. The functions are also present for preventing passing through walls by the player.

- `themes.py`
  This file provides the UI for selection of the Theme which is asked before each level. The theme is either Jurassic or Zombie with their own enemies, traps, background, sound effects, background sounds.

- `user_name.py`
  This file asks for the Username of the player before the start of the game. This helps in maintaining the Leaderboard with the names of the players.
- `path.txt`
  This file stores the solution path extracted from the maze generating algorithm.
- `leaderboards`
  This folder has 3 files for maintaining the data of the leaderboard each of the game levels.
- `resources`
  This folder has all the necessary graphics and sound effects used in the game.

5 Running Instructions

For running the game, you simply need to open the parent directory and type any of the following commands in the terminal:

- `python3 game.py music_file_path`
- `python game.py music_file_path`

As per the version of python in your system. If music file path is not provided, then default theme music is played in the game.

6 Basic Completion

The basic completion of the game consisted of the following tasks:

- **Random Maze Generation**
  The random maze generation algorithm used is borrowed from The Python Code [1] website. This algorithm first generates a full grid with walls across all the cells. Next it starts from the Top-left cell and randomly picks a neighbor cell which it it has not visited and breaks the walls between them. It continues doing this until it encounters a cell with all neighbor cell visited, then, it backtraces, and goes to the last cell which has some neighbor unvisited, doing this the algorithm traverses all through the maze, thereby visiting each cell atleast once. A unique solution is also guaranteed in this algorithm.
- **Player Implementation**
  The Player implementation is done using the Sprite object of Pygame which helps us to choose the Rect and Image of the sprite, also allows us to do animations on the sprite and update the sprite after each game loop.
- **Collision Detection**
  The wall collision detection was quite simple to implement. The `cell.wall[‘direction’]` gives the status of the walls surrounding the current cell. You just need to carefully check the conditions in which the player crosses the walls and prevent that.
- **Camera Implementation**
  The camera implementation was done using multiple surfaces of pygame. The maze is generated on a separate surface, and a part of that surface is blitted on top of the screen, the camera moves with the player, but the whole maze is not visible at once.

- **Maze Solution Path File**
  For this I had to go through the maze generating algorithm. I stored all the visited cells in a list in order. Whenever a cell is coming twice, it means there were no available unvisited neighbors in the previous cells so all cells between the repeated cells are removed from the array including one of them. This is repeated unless we reach the Final Cell. The moves are extracted from the properties of the cell and then later put into `path.txt` file.

- **Intuitive User Interface with various screens**
  All the screens used in the game are present in the `screens.txt` file. All these are called upon as functions when that button is clicked. This is well integrated across all screens, to ensure smooth gameplay.

- **Difficulty Levels**
  There are three levels with increasing difficulty. The difficulty is varied by varying the maze size, number of enemies, traps, deadliness of enemies, traps, less relative time.

- **Score, Timer, Health**
  The score, timer, health are visible on the sidebar of the Gameplay screen. Timer is a reverse clock. Positive Health is required for the completion of the level. This is reduced on encounter of enemies, and increased on collecting Health randomly spawned. The score is calculated using the formula below:

$$s = (\frac{x+y}{5}) * (\frac{t_e}{t_t})^2 + 2h * (\frac{t_e}{t_t})^2 + 50g + 20c + (t_t - t_e) * (\frac{t_e}{t_t})^2 \quad (1)$$

  Where t_t=totalTime, t_e=elapsedTime, g=collectedGems, c=collectedCoins, h=currentHealth.

7 Advanced Completion

After completion of the Basic part, I proceeded to the advanced tasks. This consisted of the following tasks:

- **High Score Leaderboard**
  There is a leaderboard for each level which gets updated after each gaming session. These leaderboards are displayed inside the options menu or just after the game gets over.

- **Sound Effects**
  There are sound effects for button clicks, turning mute button ON or OFF, typing sound effects, coin collection effects, gem collection effects, health collection effects, enemy collision effects, trap encounter sound effects, game won sound effects, game lost sound effects. Also a background music for the interface menus.

- **Custom Themes**
  There are two themes namely Jurassic and Zombie, each with its own features, custom enemies, sound effects, default background music, default background, default background padding. Thereby providing interesting gaming options.

- **Collectibles**
  There are three types of collectibles.

  – Coins
    Increase the score of the player

  – Gems
    These are rare.Increase the score by a larger amount. These increase the speed of the player for 5 seconds and hence help in completing the game.

  - Health
    These are also rare, these increase the total health of the Player.

- **Traps**
  There are random traps in the form of fire in Jurassic theme, and haunted pumpkins in Zombie theme, these are randomly spawned on the map, these cause a reduction in health upon coming near them. These traps are also animated.

- **Enemies**
  There are random enemies on the map, which on coming near to the player cause a reduction in the health of the player. These are animated enemies, which move as per a random function which randomly selects movement elements from a Moves list. The enemy is a dragon in the Jurassic theme, and a zombie in the zombie theme.

- **Custom Background Music**
  There is an option to provide the background music file in command line, this music will be played in the gameplay. If music is not provided, the default theme music will be played.

- **Particle, Player, Enemy, Collectible Animations**
  Each of the collectible, trap, player, enemy is animated according to some frame speed. This makes the game look more intuitive and visually appealing.

- **Sound Effects on Coin, Gem, Health collection as well as with Traps and Enemies**
  There are different sound effects on collection of the coins, gems, health, as well traps and enemies according to the type of enemy and trap depending upon the theme selected.

- **Colored Timer and Health**
  The timer is colored, in other words, when you are running out of time (less than 10 seconds left, it appears red and blinks). If the Health is greater than 100, it appears green. Otherwise, it appears red.

- **Mute Button** This is a toggle switch for turning ON or OFF the background music of the Gameplay.

8 My Project Journey

The Project Journey began with a lot of enthusiasm. First I became familiar with Pygame by learning from the GeeksForGeeks Website.[2] Also I got hold of the official Pygame documentation [3] and searched for some functions here and there on the internet.
I learnt a lot from this project especially the importance a deadline and doing the assigned tasks before the deadline. Also since python was used, I became much more confident in dealing with functions, classes, objects, modules, etc in python. Also some challenges were encountered in the process of making the project. These were:

- The first major challenge which I encountered was implementing collision detection.
- Another challenge was to implement camera.
- Another challenge was to make the solution path of the maze from the algorithm used.

I overcame all these challenges with some difficulty, but at the end this adds to the major learnings from the project. Also the DFS algorithm used in the maze generation as well as some part of the maze generation code is taken from The Python Code[1] website. Some of the images from the game are as follows:

Figure 1: Main Menu

Figure 2: Leaderboard

Figure 3: Theme Selection

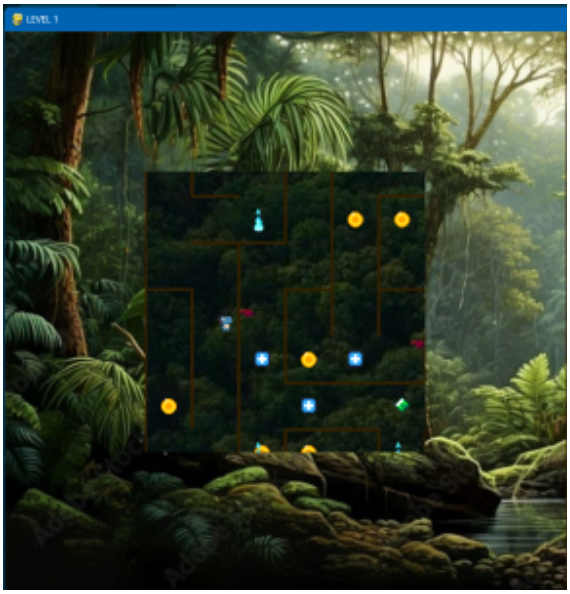

Figure 4: Gameplay

Figure 5: Game Over

9 Bibliography

References

[1] https://www.thepythoncode.com/article/build-a-maze-game-in-python.

[2] https://www.geeksforgeeks.org/pygame-tutorial/.

[3] https://www.pygame.org/docs/.