ESCAPE FROM THE PIT

Game Manual

JJ Small
W00992440

Dr. Geoffrey Matthews
CSCI 321 - Spring 2015

Back Story

You are the Lone Adventurer. You spend your days exploring the far off reaches of the world as you seek treasure and hidden mysteries. During one such adventure in the mountainous region of The Devils Cauldron, you find yourself lost in the deep network of tunnels and cave that carve through the mountain.

As you make your way through the labyrinthine the ground starts to shake. Stone crumbles from the walls and the ground opens up beneath your feet and you fall into the lava filled cavern below. You now have to jump from platform to platform and stay alive for as long as you can. Are you ready for the challenge?

User's Guide

The rules of the game are fairly simple and straightforward. However, though it make take just a minute of two to get accustomed to the controls it will take some time to master them and start surviving for long periods of time.

When the game starts you fall from the top of the screen and land on a platform. From here on out you need to jump from platform to platform and avoid the fire balls. If you get hit by a fire ball or fall into the lava below you die and have to start over.

Your goal is to stay alive for as long as you can and try and beat your high score.

Controls:

Move Left:	Left arrow
Move Right:	Right Arrow
Jump:	Up Arrow
Down:	Down Arrow
Pause Menu:	ESC key

Module Documentation

Main.py

Imports: main_menu, player, Platforms, fireball
Functions: main(), game_loop(screen), game_over(screen), pause_menu(screen)

This is the main file for the game that needs to be run. It consists of two functions, main() and game_loop(). The main function is pretty self explanatory, as it initializes the screen and the audio and then starts the game by calling the main_menu() function. Once the menu (described later) finishes and the user wants to play the game, it calls the game_loop() function. This function initializes all of the game objects and then starts the loop. Gets user input, updates the sprites, and then draws everything to the screen.

Platforms.py

Functions: update(self, plat_list), resef_pos(self, plat_list)
This is the class for the platforms. Each platform object has an image and a rectangle. The update() function is used to move the platform and check if the platform has fallen into the lava, and if it did it triggers the reset_post() function which will spawn the platform at some random location at the top of the screen.

There's some collision detection for each platform that will make sure that platforms don't spawn on top of each other or touch.

Player.py

Imports: Platforms
Functions: update(self, collision_list, fireball_list), jump(self), move_left(self), move_right(self)

The player class is the largest for this game. The player constructor has an array of images that are used to do the movement animations. It also has a lot of variables pertaining to position, the state of the player (standing, falling, jumping), and which platform the player is on.

Movement is governed by the left, right, and jump functions. The state of the player is changed depending on what the player is doing. Such as when the player is jumping they can't jump again until they are standing. There's also boundary detection which stops the player from moving too far to the left or right.

Most of the logic is in the update() function. It first gets the keyboard input and calls the appropriate function. Then it checks each state to see which action to do. If you're falling, calculate gravity and move the player down. If you're standing, the player moves down at the rate of the falling platform. When you're jumping, your movement is based on the forces and gravity.

There's also code interspersed throughout the update() function which tests for collisions with the various objects and also tests if you have died.

Fireball.py

Functions: update(self)

The final module for this game is the fireball class. It creates fireball objects that move up from the bottom of the screen. Is has an image array that is used to do the animation loop. The update function just changes the y coordinate of its rectangle and makes it move.

How all of these modules come together is pretty simple. The player object can collide with the platforms and land on one if they are in the falling state. If the player collides with any fireball in the fireball collision list then the game is over and you have to start over. Same thing if you fall into the lava pit.

The menu is just another loop that gets input and blits the menu images to the screen.

Assets

There are two types of assets for this game; images and audio. The images are for the backgrounds, the menus, the platforms, and the player sprites. I tried to make the images in such a way that there was as little redrawing to the screen as possible in order to save on CPU time. The other type of asset is the sounds for the game. There's only two of them, the background music and fireball sound effect, and they can be found in the 'audio' folder.

Acknowledgments and Credits

Though 95% of the game was written and designed by myself, I did use a couple of resources that were available free for use online.

The character sprite was obtained from Sithjester's RMMXP Resources website, under the Steampunk category. All of the sprite sheets are available for free as long as you mention where you got them from.

http://untamed.wild-refuge.net/rmxpresources.php?characters

The second resource I obtained online was for the background music which plays throughout the duration of the game. It was downloaded from No Soap Radio's royalty free music generator site. There are a ton of options available so check it out if you want.

http://www.nosoapradio.us/

The final resource was for the sound effect for the fireball, which was downloaded from the freeSound website under the noncommercial Creative Commons license.

https://www.freesound.org/people/Robinhood76/sounds/248116/

Autobiographical Info

Video games have always been a favorite hobby of mine. From console to PC, to handhold to mobile, it's seems like there's always a game or two that is taking up my free time. With that being said however, before this class I've never really considered designing or making my own games. Game development is a long and involved process, and though there a ton of good games that have been developed by a single person, I've just always thought it was something that was beyond my level of skill. That changed though when I started working on this game and realized that not only making a game is manageable and even possible for me, I found out that it was something I really enjoyed.

My programming history does not extend much outside of what I've done in school, unfortunately, and though I like to spend time creating programs and learning about computers and technology, one of the challenges that I encountered early on was that this is the first time I've programmed in Python. I had heard good things about the language so I was glad that I now had the chance to explore it, and it turned out to be a pretty fun and easy to use language. So once I spent some time learning the basics of Python and then Pygame, going about creating my game was pretty straightforward.

Speaking of creating my game, since I knew I was just a beginner game designer (though with a lot of game knowledge), I aimed to make a simple arcade style game that a lot of mobile phone games nowadays are like. That is, a game where you can pick it up and play for a few minutes and better your own individual skills. I knew that a game of this style was in my range of skill and I think that I managed to accomplish this.