Timothy Kim
W01011895

Game Manual
Quebe

Pygame Platformer

CSCI 321 - Spring 2015 - Dr. Geoffrey Matthews

You are Quebe. ⬛

You must get to the White Door. ⬜

What awaits at the end? 🟥 🟩 🟦

User’s Guide

Your existence depends on an unexplained draw toward the White Door. Emptiness and death lies below you should you fall. Avoid contact with the out of place trying to stop you.

Controls

- Move/Jump: Arrow Keys | WASD
- Menu/Pause: ESC
  - Quit: q
  - Help: F1
- Restart Level: r
- Toggle Music: m

Cheats

- Skip to Next Level: Tab
- Toggle Jump Mode : j
  - allows for continued jumps while airborne

Modules

All source code for the game reside in the quebe.py file. Classes are not separated into individual files. Game requires the Pygame modules library in order to run.

Platform

Platform imports the Sprite parent class module. It holds the data for the grey platforms in which the player can walk and jump on. Creates a surface and a Rect. It bounds the player to the level by providing a barrier.

Warp

Warp uses the Sprite parent class. Similar to Platform but holds the 'White Door' object that transfers the player to the next level upon contact.

EnemyTri

Similar to above. Creates the pink triangles that represent enemies on the map.

EnemyPlat

Similar to Platform but handles the pink platforms that represents death on contact.

Player

The Player class represents Quebe, the black square, that the user controls. Attributes include its dimensions(35x35), surface image, Rect coordinates reference, current speed/acceleration, the current level, and its jump sound effect. The classes methods include:

- update: handles all movement of the player checking for collisions.
- calcGrav: determines the current gravity afflicted on the player and ground detection.
- jump: called when the player hits the jump button. Only allows jumps from a platform.
- goLeft/goRight: called when player wants to go left. Changes players Δx/y
- goUp/goDown: called only in jump cheat mode allowing for unlimited jumping.
- stop: on key up, stops all speeds to 0.

Level

Creation of individual levels is handled by the Level superclass. It takes a Player object as an argument. It handles all sprites on the level, the background image, the current x and y shifts of the world for camera position, and the players starting location on the map. Its methods include:

- update: updates the positions of platforms, enemies, and the warp door.
- draw: blits the background with shift ratio for depth of field, draws all sprites too surface screen.
- shiftWorldX: shifts all objects horizontally in the level by appropriate coordinate amounts.
- shiftWorldY: vertical shifts

Level_n

Each unique level is created by calling the Level parent constructor. All necessary images and values are loaded. Each level also has a level_limitX/Y attribute that sets the invisible boundaries of the map. Going outside of these values restarts the level. The general structure of a level is laid out like:

- background image
- background color key
- level limit x
- level limit y
- starting x position
- starting y position
- platforms
- warp White Door
- enemy triangles
- death walls

platforms, warp, triangles, and deathwalls are placed on the map using a list of 4 numeric values in the form [length, height, x pos, y pos]. A list of all the objects are then created and added to the appropriate sprite lists in Player.

main

The main procedure handles all the initialization of the modules and classes. The main game loop resides in here. Screen size is set to 800x600 and levels, sprites, fps, and music are loaded. main contains 3 functions:

- restart: restarts the current level
- menu: creates and displays the menu/pause screen
- help: creates and displays the in-game help screen

The main game loops until a quit command is issued. It checks for all necessary events and handles them accordingly. Cheats as well as certain collisions are handled in this loop as well.

About Me

I am currently pursuing my Computer Science degree at Western Washington University. The idea behind creating this game was to make a simple puzzle platforming game with a minimalistic aura. It began as a proof of concept of a sorts to get myself exposed to the world of game programming. As this was the first attempted game project I tried to reel in my expectations. Inspiration was taken from the Impossible Game. It was originally a web browser flash game I played while I was younger. It was very simple and minimal only making use of a single player controlled square and triangles. I wanted to see if an enjoyable game could be made using these same concepts and pleasing pastel color hues. The end result is a very simple game. However I did want this game to become something on a far more grand scale. Due to time I was unable to implement all gameplay aspects I had envisioned. Creating levels was a challenge that could have been solved with better planning before the initial codebase was written. As it is, it requires plotting individual objects using xy coordinates which is inefficient and time consuming. The physics behind the game is also another point of change I will look into at a later point. After creating a jump cheat mode where the player can have unlimited jumps while in the air, I sometimes found that this mode was more fun on some instances. This lead to more plans to perhaps creates different game modes. Optimization of the code would have to be fixed too as my attempts to run at a large resolution proved laggy and took away from the immersion. Overall I am satisfied with the amount I have learned and this project gave me a greater appreciation too all those games I have played and the hard work that goes into every one no matter how simple it may seem to the player.

Acknowledgements

Code Samples

Basic implementation of concepts was referenced from the two textbooks provided for the class online and the Pygame documentation.

- $https://inventwithpython.com/makinggames.pdf$
- $http://programarcadegames.com/$
- $http://www.pygame.org/docs/$

Assets

Several assets were used in development of this game from online resources.

Backgrounds

- $http://hdw.eweb4.com/wallpapers/9694/$
- $http://pichost.me/1640990/$

Fonts

- $@http://fonts.googleapis.com/css?family=Raleway$

Music

- Elegiac by Jon Hopkins from album Opalescent

Sounds

- $http://opengameart.org/content/8-bit-jump-free-sound-effects-1$