Akshay Gupta
Shreya Gaur

Final Program: Adventure Time

Design Document

Program Requirements

1. There must be a "**main character**" that the user controls, with the following requirements:
    1. User input must control the character.
    2. Must be able to change position within the game
    3. Must have at least 2 attributes (for example: current position and number of lives left).
2. There must be a "**board**" for the main character to move around in
    1. Must have at least 16 possible positions (4x4, 2x8, 1x16)
3. There must be **things** on the board for the main character to interact with
    1. Must have a position on the board. They may move or be stationary.
4. Must include **at least 3 topics** from the last third of the course. We have used:
    1. File I/O
    2. Objects
    3. Turtle graphics

Game Design

1. We decided to make a version of the game, “Space Invaders” called, “Earth Defense”.
2. It is a 2D arcade style game that runs using the turtle module that is in-built in certain softwares of Python, for example, Anaconda.
3. In this game, you control a spaceship protecting Earth from a never-ending supply of enemies. Aliens from outer space are trying to invade Earth!
4. There are always 5 enemy UFOs that are slowly approaching you while moving from left to right.
5. Every UFO you shoot down, you gain 10 points.
6. However, if an UFO hits your spaceship, you lose a life. You only have 3 lives. If you lose all 3 lives, Earth loses all hope and the game ends.
7. If you let an UFO pass you and invade Earth, you lose.
8. The game window should display your spaceship, the enemy UFOs, current score, and the number of lives you have left.

Variables

1. border_pen is a turtle object which is used to create the boundary of the board
2. score_pen is a turtle object that is used to display the score on the game screen
3. lives_pen is the turtle object that is used to display the number of lives that the player has left in the game. The number of lives that the player starts off with is three.

4. player is a turtle object that creates the “fighter” that is user-controlled. It can be operated using the left and right arrow keys.
5. number_of_enemies defines the number of enemies that will appear on the game screen, which is set to 5
6. enemies is a list which is originally empty, and creates 5 identical “enemies” in terms of their size, shape, color and speed
7. bullet is a turtle object which creates the bullet that the “fighter” can fire using the spacebar to shoot the enemies
8. x within the function move_left() stores the x coordinate (position) of the player and reduces the value of the coordinate as the user moves the player left using the left arrow key
9. x within the function move_right() stores the x coordinate (position) of the player and increases the value as the user moves the player right using the right arrow key
10. bulletstate is a global variable that helps to determine whether the user wants to fire the bullet or not
11. distance in the function isCollision() determines the distance between the bullet and the enemy

Functions

1. move_left()- This function takes the x coordinate of the player and moves it to the left when the user presses the left arrow key by subtracting 15 from the coordinate each time. It also checks to make sure that the player does not move beyond the leftmost boundary of the game board

2. move_right()- This function takes the x coordinate of the player and moves it to the right when the user presses the left arrow key by adding 15 to the coordinate each time. It also checks to make sure that the player does not move beyond the rightmost boundary of the game board

3. fire_bullet()- Checks if a bullet is ready to be fired. If it is, then it released from the player and moves up until it either collides with the boundary or an enemy

4. isCollision()- This function computes the distance between the released bullet and the enemy. If the bullet hits the enemy, then the score goes up by ten and the enemy re-appears in a new position. It also is used to compute the distance between the enemies and the player. If they collide, then the number of lives that the user has left goes down by one. The enemy will reappear in a new position.

Error Handling

IOError- This could be caused if the image file for the background does not exist. This is handled by the if statement which checks if the file exists. If it doesn't, then the program executes with a plain black background.