*****Note: TA Max was helping us with our project, and allowed us to turn in our project an hour late, sorry for the inconvenience**

README

UFO Fighters
Anthony Lu, Mario Zuniga
TA: Bryant

Intro

UFO Fighters is a game where two players control fighter UFOs as they try to shoot each other to eliminate the other players' lives. Player 1 turns left, turns right, and shoots bullets using the A, D, and W keys. Player 2 does the same with the left, right, and up arrow keys. Bullets will vanish after hitting the arena boundaries or a player, and each player can only have two active bullets at a time. A life powerup will spawn in a random place every fifteen seconds, and touching it will grant a player an extra life. Each player starts with five lives and can only have up to five lives. The game ends when a player reaches 0 lives.

Features

1. *Life counters; the game should automatically end when one player reaches 0 lives.*
2. *Power-ups that grant extra lives spawn on random places on the board.*

Justification for Complexity

Our project includes a variety of complex features, including the sprite's movements (rotation and continuous forward movement), detecting collision (with bullets & power-ups), and a life-counter if the player obtains a power-up or loses a life. In building these features, we overcame several complicated tasks. We tackled these problems by thoroughly researching how to write the function, implemented and wrote it into our own code, and tested it. If we ran into a problem or a bug, we broke down every individual line of code and reevaluated its purpose and function.

Lists & Script Variables

We used lists to organize player, powerup, and bullet sprites in order to adjust how both players interact with and react to different sprites. For example, we created separate lists for player 1's and player 2's bullets in order to make it so that a sprite won't get hurt from its own bullets, but will hurt the other player.

Script variables were always determined through each class' constructor and were mainly used within their methods. This was especially useful for rendering different sprites for each player's life counter and bullets.

Function Table

Each row will describe the functionality of one custom block/function, with the relevant information placed in the relevant columns (as defined above). You must include a separate row for EVERY custom block/function you create

Block / Function Name	**Domain (inputs)**	**Range (outputs)**	**Behavior (role in the context of the project)**
Bullet.__init__(self, speed, x, y, image)	Strings, numbers	Bullet sprite	Constructor for the Bullet class; determines its speed, spawn position, and sprite image
Bullet.update(self)	Self	Bullet sprite	The function moves the bullet in the xy plane and destroys it when it hits the border or the enemy player
Player.__init__(self, left, right, image, shoot, x, y, speed, bullet, small)	Keyboard inputs, strings, numbers	Player sprite	Constructor for the Player class; dictates the player's control for its movement and shooting, its starting position, its speed on the game board, and all the sprites associated with it
Player.update(self, function)	Self, function	Player sprite	This function is responsible for each player's movement on the arena, rotates their velocity vector when the right inputs are passed, increases / reduces their lives counter depending on what sprites they touch, and ensures that they don't fly off the screen
Player.livescounter(self)	Self	Sprite drawings	This draws smaller versions of each player's sprite on the top of the screen to indicate how

			many lives they have left
Powerup.__init__	Strings, number	Powerup sprite	Constructor for the Powerup class; determines the sprite image and loads it in on a random place within the arena
Powerup.update(self)	Self	Powerup sprite	This will check if a player has touched the sprite; if a player touches it, the sprite will disappear

Extra Credit (optional)

We built our game using pygame, and were required to self-learn the mechanics using video and website tutorials. We also had to research how to implement and code each function in pygame ourselves. By using pygame, we were able to implement a wide range of functions and features to our game.