Computer and Systems
Engineering

Python course(summer training)

1-Amal Mohammed Al-Saeed Ibrahim

2-Aya Mohammad Al-Qdry Ammer

3-Tassnem Abdul Rahman Hamed Abdul Wahab

4-Shrouk Mohammed Mahmoud Sabry

5-Rehab Mohammed Ahmed Lotfy

In this project we have made a group of games andused more than one library.

Libraries are a set of useful functions that eliminate the need for writing codes from scratch.

We made six games:

1-Connect_4

2-Tic tac toe

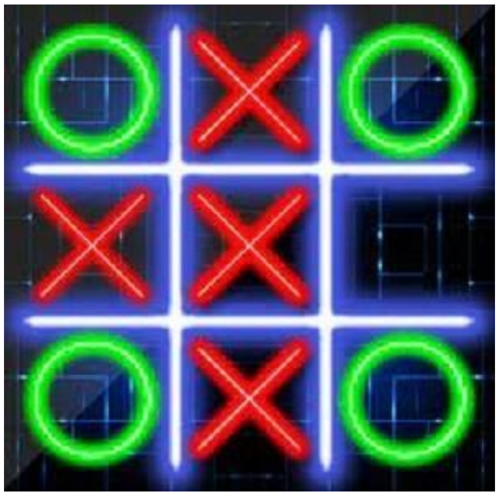
3-Flappy Bird

4-Ping-Pong Game

5-chicken Invaders

6-Runner

Tic tac toe



It is played by filling one of the squares formed by the intersection of two vertical and two horizontal lines with either an X or an O, and usually, X starts first. The player who succeeds in making 3 identical symbols vertically, horizontally or diagonally is the winner.

In this game we used a library called thinker.

First we created a window and set a background for it
We divided the window into 3 frames.

The first frame I created the game inside and the second frame inside it I created the buttons responsible for exiting the game and starting a new game, and the third frame I put the score for each player.

We used many functions:

Function	Description
checker(buttons)	This function makes one of the players to play first and the other to play after, and so on.
colorreturn()	At the end of the game, this returns the color of x,o to the original color.
Scorekeeper()	This function shows when the player wins and when winning the score increases by 1
reset()	When you press Restart game button, all the text inside the buttons are deleted.
newgame()	When you press start new game button, the texts are deleted and scores are equal to zero.

Flappy bird

Flappy Bird is an arcade-style game in which the player controls the bird Faby, which moves persistently to the right. The player is tasked with navigating Faby through pairs of pipes that have equally sized gaps placed at random heights.
This is done by using these functions:

Function	Description
welcome_main_screen()	Shows welcome images on the screen until one of these happen: 1-if user clicks on cross button, close the game 2-If the user presses

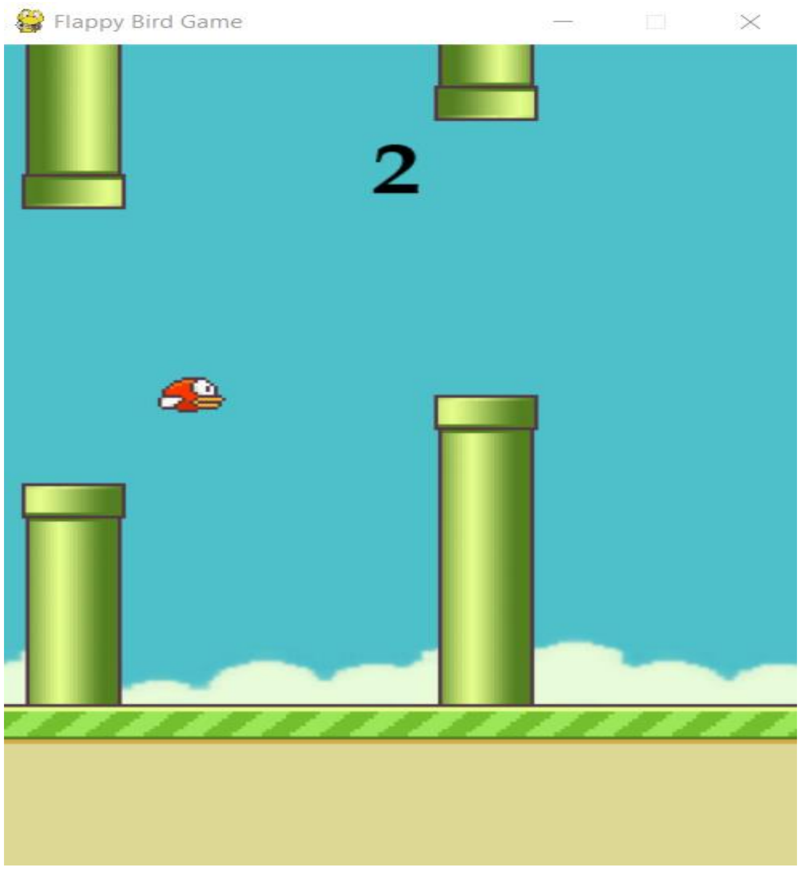
	space or up key, start the game for them
main_gameplay()	1-Set the coordinates of the pipes so it has constant gap between them using get_Random_Pipes(). 2-Start the game as when the user presses the up key or the space the bird fly otherwise it falls down to the ground. 3-if it falls and touch the ground it goes to the welcome_main_screen() 4- if the user presses the Escape key close the game 5- if the bird goes through the pips without crashing the pipes the score increases by 1.

is_Colliding(p_x, p_y,up_pipes, low_pipes)	Cheek if: 1-the bird crashed the pips 2- falls to the ground
get_Random_Pipes()	Generate positions of two pipes one bottom straight and one top rotated for blitting on the screen

Computer and Systems
Engineering

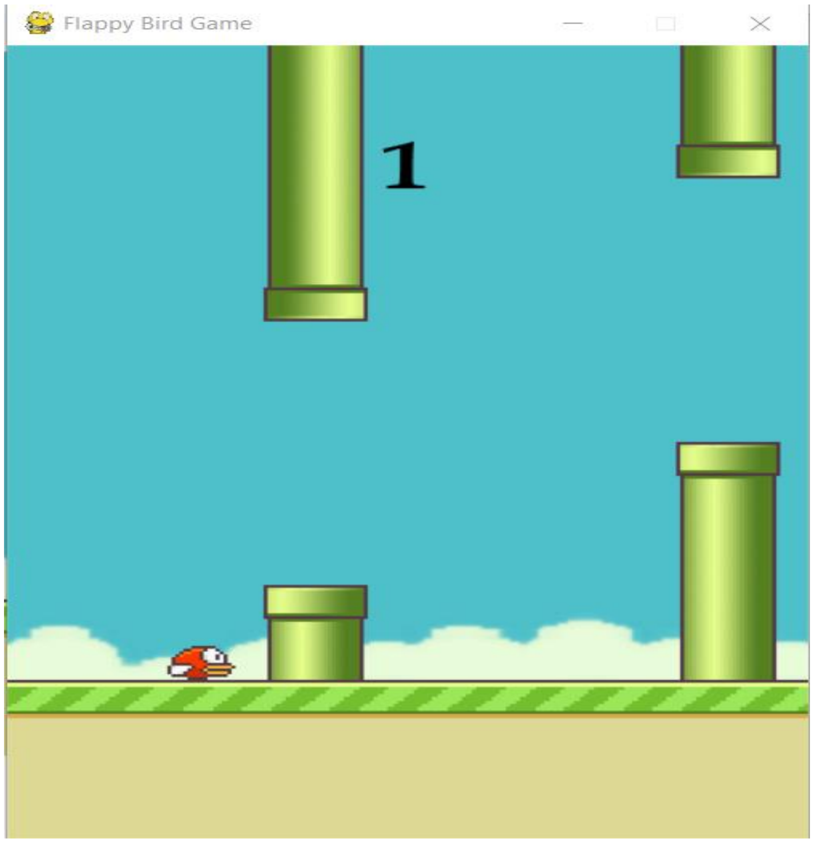
Python Report

Computer and Systems Engineering



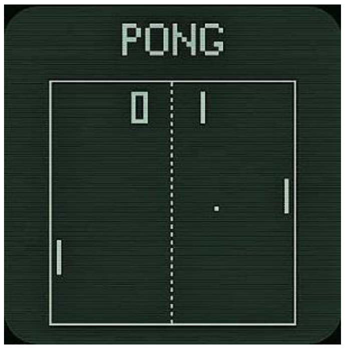
Python Report

Computer and Systems
Engineering



Python Report

Ping-Pong Game



About ping-pong game :

Ping-pong is a well-known computer game that is similar to table tennis. The two players in this game control the two paddles on either side of the game window. To hit the moving ball, they move the paddles up and down. A player's score rises when he or she hits the ball or when the opponent misses the hit. Game end when one of two players scores "ten" goals , then players can play new game with new score when press "Again" button.

Creating ping-pong game :

First to create this game we use "Turtle Library", which is a Python feature like a drawing board, which lets us command a turtle to draw all over it.

we create our window and start to draw on it using “Turtle Library”.

We draw the two paddles and one ball.

Then,we start to use functions to move paddles and ball.

We create two buttons using Buttons Library:

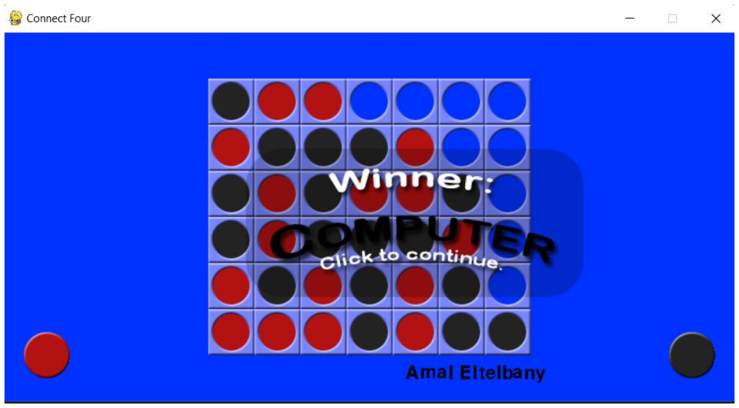
1- “Restart” : start new game with new score
2- “Again” : After one player win , players can start new game with new score

Libraries we used in this game:

Function	Description
Again()	This function has all functions I create in the game and it’s used to start new game
Paddle_L_up()	This function is used to moving left_paddle up , as we take the value of y and add to it specified value to go up
Paddle_L_down()	This function is used to moving left_paddle down

	, as we take the value of y and subtract from it specified value to go down
Paddle_R_up()	This function is used to moving Right_paddle up , as we take the value of y and add to it specified value to go up
Paddle_R_down()	This function is used to moving Right_paddle down , as we take the value of y and subtract from it specified value to go down

Connect Four

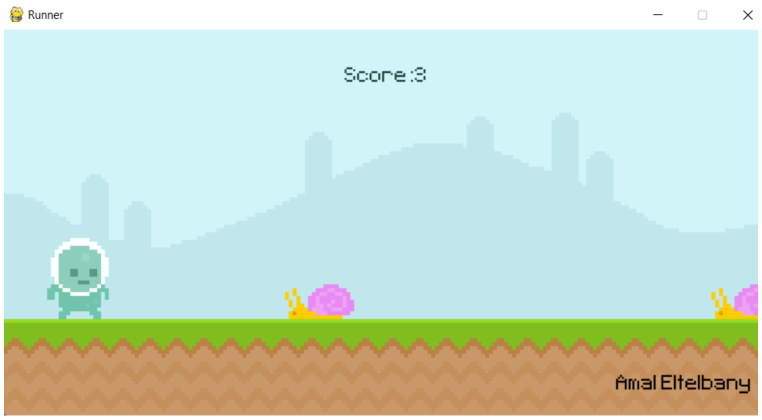


The game ***Connect Four*** has a 7 x 6 board where the players take turns dropping tokens from the top of the board. The tokens will fall from the top of each column and come to rest on the bottom of the board or on top of the topmost token in that column. A player wins when four of their tokens line up in a row either horizontally, vertically, or diagonally. The AI for this game is pretty good. It simulates every possible move it can make, then simulates every possible move the human player can make in response to each of those moves, and then simulates every possible move it can make in response to that, and then simulates every possible move the human player could make in response to each of those moves! After all that thinking, the computer determines which move is most likely to lead to it winning. So the computer is kind of tough to beat. I usually lose to it. Since there are seven possible moves you can make on your turn (unless some columns are full), and seven possible moves the opponent could make, and seven moves in response to that, and seven moves in response to that, that means that on each turn the computer is considering 7 x 7 x 7 x 7 = 2,401 possible moves. You can make the computer consider the game even further by setting the DIFFICULTY constant to a higher number, but when I set to a value larger than 2, the computer takes a long time to calculate its turn. You can also make the computer easier by setting DIFFICULTY to 1.

Then the computer only considers each of its moves and the player's possible responses to those moves. If you set the DIFFICULTY to 0, then the computer loses all intelligence and simply makes random moves.

Function	**Description**
runGame(isFirstGame)	Let the computer go first on the first game, so the player. can see how the tokens are dragged from the token piles. Randomly choose who goes first, contain the main loop
makeMove(board, player, column)	Make the player move either human or computer
drawBoard(board,extraToken=None)	draw tokens, draw the extra token, draw board over the tokens, draw the red and black tokens off to the side
getHumanMove(board, isFirstMove)	Show the help arrow for the player's first move, start of dragging on red token pile, update the position of the red token being dragged
getComputerMove(board)	get the best fitness from the potential moves, find all potential moves that have this best fitness
getPotentialMoves(board,tile,diff)	Figure out the best move to make, a winning move automatically gets a perfect fitness, don't bother calculating other moves, do other player's counter moves and determine best one, a losing move automatically gets the worst

Runner



The game ***Runner*** has player animates over the game’s time until
He hits the obstacles which is a snail and flyer and the game a counter to count score

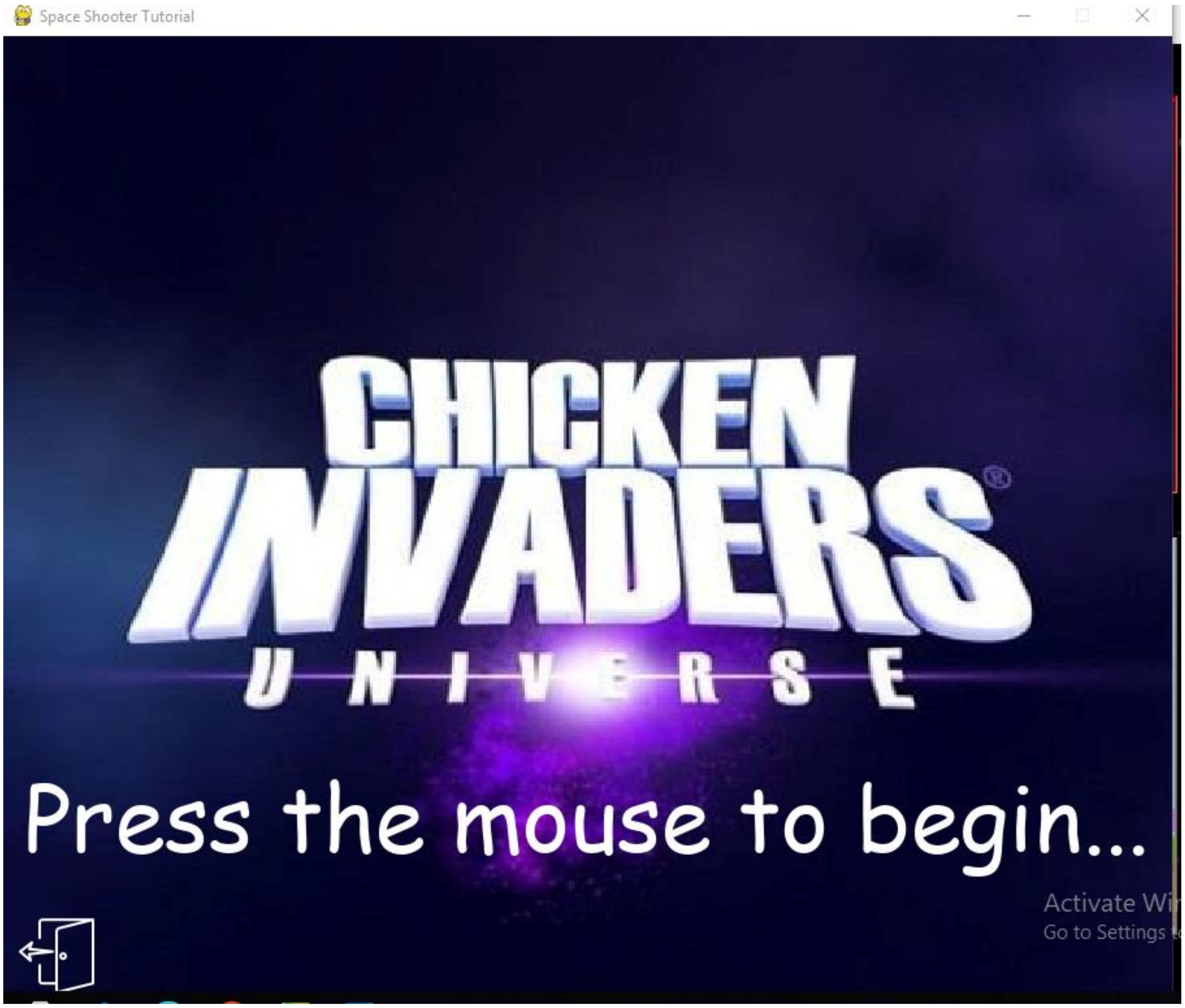
Function	**Description**
player_input()	Takes the input from user to start game which is space key
apply_gravity()	Set the player’s gravity to make him stable on the ground and increase it to make him jump
destroy()	Kill the player when he collides one of the obstacles (snail,fly)
display_score()	Count the score and display it on the screen

chicken invaders

About the game:

Chicken invaders is a famous game which we try to make it , chicken invaders game is based off the old Atari game where you have those spaceships coming down the screen and you is the big shooter and you are trying to shoot it all of the sky and then you win , but in chicken invaders we have a lot of more features that we have lives and levels and lives and player healths are the main things that control the game .
Lives decrease be one as one chicken is move under our shooter and players health is decreased by one every time the egg hit the shooter and the level increments by one every time the shooter hit all the chicken in the screen so we increment the level to make a new row of chicken come from up of the screen in different height, the player will lose if the number of levels become equal to zero or the player health become less than zero .

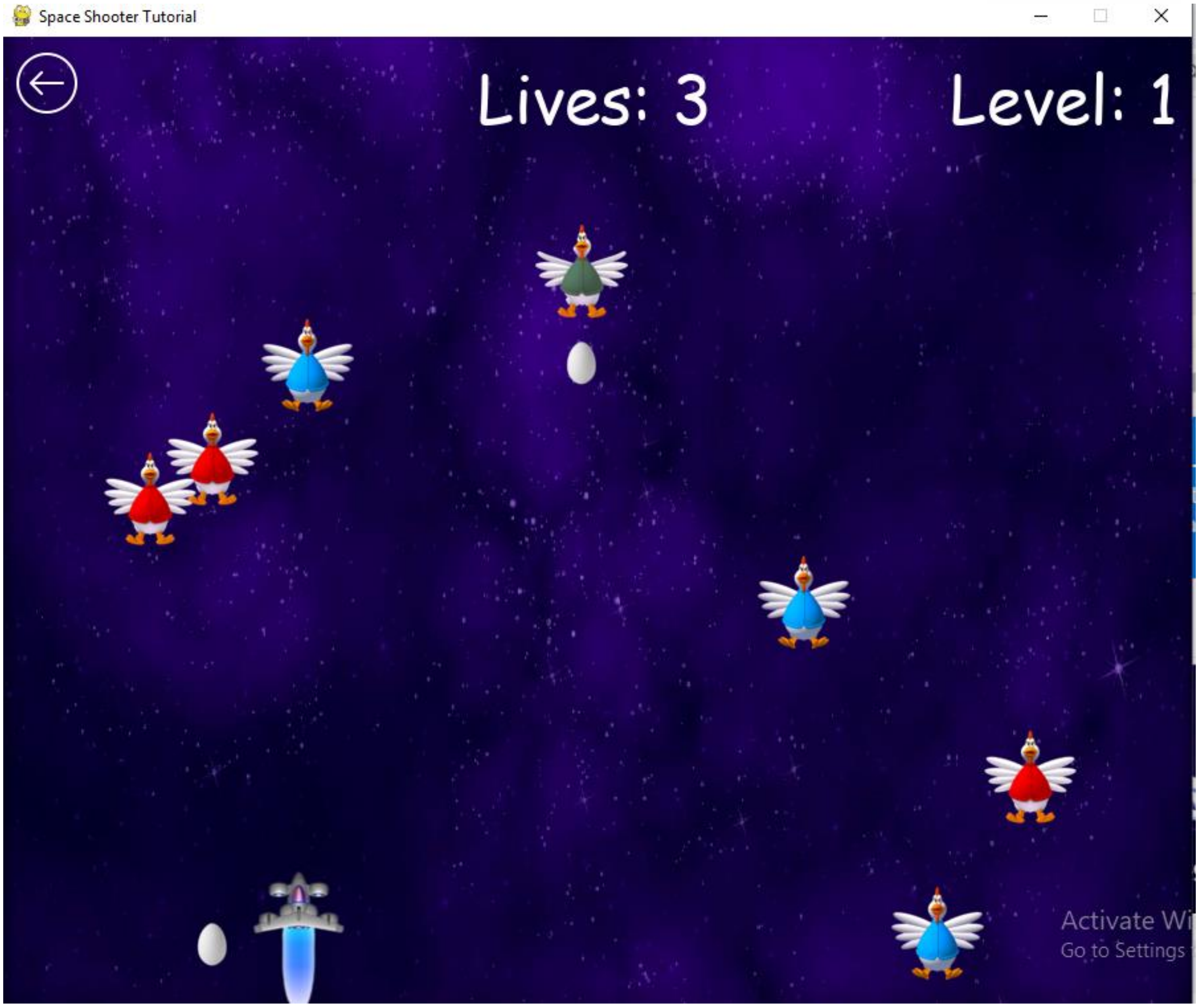
Computer and Systems
Engineering



Python Report

Computer and Systems
Engineering

HELWAN UNIVERSITY



Python Report

Function/class	Description
main()	We create a main loop which going be kind of what handles all our events so it is going handle the collision and it is going be what calls things to be draw on the screen and it is what allow us to quit the game and run the game as all.
Redraw()	it is a function inside the main function which is going to draw everything for us in the screen and it is going to handle all our drawing , anything that needs to rendered and refresh it so that the update version.
Class ship:	ship class is an abstract class that we going to inherit from it later by enemy ship and player ship it have general properties for both of them

Class laser	Contain all the properties of laser and it’s function
Move()	It is responsible for move things from place to another
Collide()	Function of collide which is going to tell us if this collides with an object so laser collide object and what this going to do is calling a function that we have to define and return the collide of object itself collide functions check if things are colliding when removing the laser , determine if the two objects are overlapping , the pixels between them are overlapping then we say yes the two objects are colliding
Shoot()	It cool_down_counter variable which tell us to show a laser or to wait for another second
Main menu()	It contain the “press mouse to continue ” which is let us start play the game and run the menu before the game and also contain the quit from the game and the back button to big main menu of all games

`Class button ()`	It contains the properties of the button that get back to the main menu and the second button that go back to the big menu of all games

Computer and Systems
Engineering

Python Report