COSC 1436 - Assignment 4

## 1. Brief Introdunction to Pygame Zero:

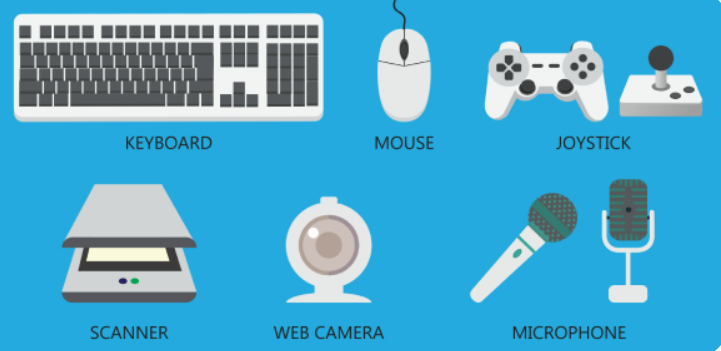
We are going to use Pygame Zero for Assignment 4. Assignment 4 is continued from Assignment 2. For this assignment, we need more concepts of Pygame Zero with the concepts from previous Assignments:

- on key down

- on key up

- update

I would like to start with explaining the input for the computer. Let’s think of how could you command to the computer. Can you command right away to the computer like ‘play the game!’? Of course, if you use microphone and the system could read the voice, then it could play the game. However, usually, the computer cannot understand what you are talking about. To command or control the signal, we use keyboard, mouse, camera, scanner, joystick, or the microphone; these are referred to as the input to the computer.



In this assignment, we will use ‘**on\_key\_down**’ and ‘**on\_key\_up**’ built-in functions in Pygame Zero. Through these two functions, we can move the character or modify the system while the game is implementing. In other words, we can command or control the signal to the computer by using the keyboard.

The syntax for ‘on\_key\_down’ function is:

***on\_key\_down(key, mod, unicode)***

This function is called when a key is depressed. It takes three parameters, but we will use only key parameter for this assignment.

The syntax for ‘on\_key\_up’ function is:

***on\_key\_up(key, mod)***

This function is called when a key is released. It takes two parameters, but we will use only key parameter for this assignment.

There are many keys for the functions, but I want to introduce 6 keys which are mainly used.

|  |  |  |  |
| --- | --- | --- | --- |
| Key | Object | Key | Object |
| Right Arrow(←) | keys.RIGHT | Left Arrow(→) | keys.LEFT |
| Down Arrow(↓) | keys.DOWN | Up Arrow(↑) | keys.Up |
| Enter | keys.RETURN | Spacebar | keys.SPACE |

In this assignment, we will use **update()** function which is built-in function in Pygame Zero. Just like the **draw** function, the **update** function is automatically called (executed) 60 times per second by Pygame Zero engine. You might see the empty **update()** function from the previous assignments:



For this Assignment, we will use the **update()** function to make the right or left movement of the character.

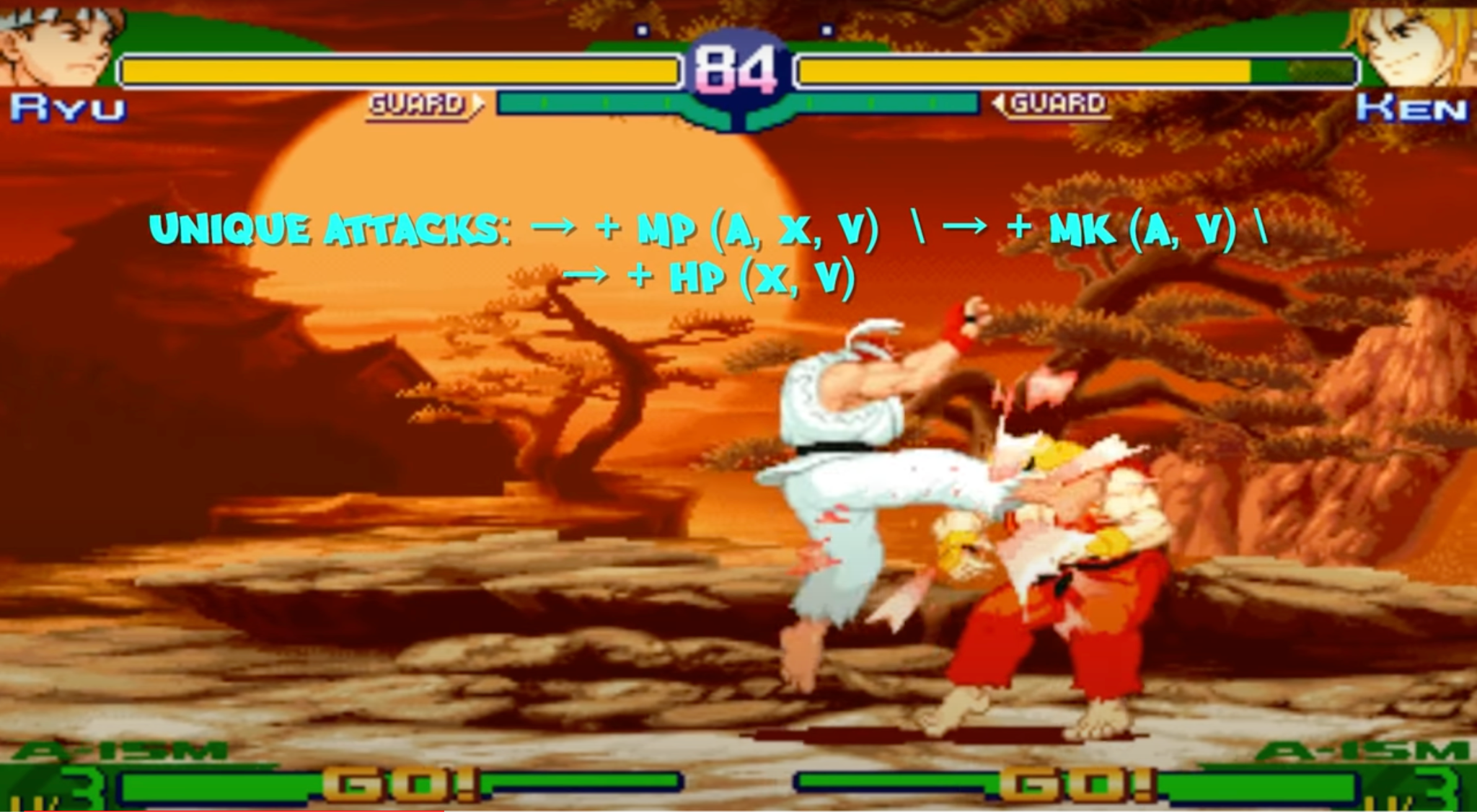
## 2. Assignment Description:

In assignment 4, we are going to continue adding more features to our game, Mortal Kombat. We are going to allow our character to have animation and perform special moves. Characters in fighting games have some special moves which can be invoked in certain situations or conditions. Fighting games start with a default picture of the character, and when a special move occurs an animation occurs then the character returns back to the default picture. Below are examples of special moves being performed in popular games:

**Samurai Warrior**



**Street Fighter**



Here is another example, a video of a battle round of the game Mortal Kombat. We can see how Sub-Zero is able to perform different kicks and punches when different combinations of buttons are pressed on the keyboard. We are looking forward to incorporating similar features in our assignment.

[Mortal Kombat 1 Moves List](https://youtu.be/8wpleRmHF6A?si=9VqDisSaDN8uGpdx)

## 3. ALGORITHMS FOR THE ASSIGNMENT:

In Assignment 4, we are going to make the player1 move smoothly as we command. Basically, we will use the keyboard to command to player1. We will use ‘on\_key\_down()’ and ‘on\_key\_up()’ functions to let the keyboard make the command.

In both functions, we are going to use various blocks of if-elif statements. If-elif statements in programming allow for conditional checks: if a condition is met, a specific block of code executes. If not, subsequent conditions are evaluated until one is true, or an optional `else` block provides a default action. This construct enables programs to make decisions based on changing conditions, improving flexibility and adaptability to different scenarios.

In our assignment, our conditional checks are going to be either different keys on the keyboard or features of our player. Depending on what the condition is, we modify the properties of our player.

For example, if the key pressed is “LEFT”, we modify the position of the player and the property that stores the movement status of the player.

Also, it is important to understand how the images are composed in this assignment. When you look through the ‘images’ folder, you can see the images and the name of them. You will see ‘**p1hp0~4**: player1’s punching images’, ‘**p1idle0~6**: player1’s basic images’, ‘**p1lk0~4**: player1’s kicking images’, and ‘**p1walk0~8**: player1’s walking images’.

Can you see the patterns of the name? All of them are for player1, so it starts with ‘p1’. Accordingly, the movement of player1 is followed by like ‘hp’ for punching, ‘idle’ for basic moving, ‘lk’ for kicking, and ‘walk’ for walking. The number is followed by at the end. Each movement has many images because we can make the movement smooth if we subdivide the movement in many.

In short, you can change the movement by changing the word in the middle of the name between ‘hp’, ‘idle’, ‘lk’, and ‘walk’ by using if-elif statements. Also, you can make the movement smooth by making the number in the end of the name go through in the order.

## Assignment Questions:

1. Declare four more properties for **player1**.

- ***imgNum***: declare with **0** integer value. This variable indicates the number at the end of the image’s name.

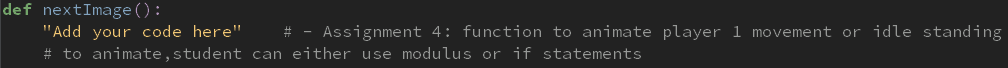
- ***move***: declare with “**idle**” string value. This variable indicates the middle of the image’s name which means the movement that the character will take; ‘idle’, ‘hp’ for punching, ‘lk’ for kicking, and ‘walk’.

- ***animCounter***: declare with **8** integer value.

- ***dx***: declare with **0** integer value. This variable indicates the direction on the x-axis and the speed of player1’s movement.

1. Add the code for **nextImage()** function & draw **nextImage()** function.

: This function will animate player 1movement or idle standing. To do so, we will modify the variables of ‘**player1.animCounter**’, ‘**player1.move**’, ‘**player1.imgNum**’, and ‘**player1.image**’ in this function. You will see the nextImage() function like below and and you should add the code under it.

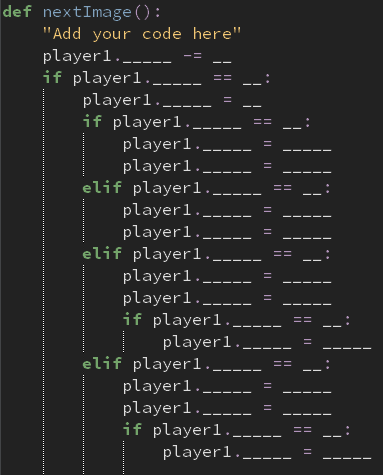


You should fill the blank for the given example.

You should write the code by following the pseudo-code.

\* Pseudo-code is a methodology that allows the programmer to represent the implementation of an algorithm.

2.a (usual way with the blank)



2.b (pseudo code)

|  |
| --- |
| def nextImage():  decrease player1.animCounter by 1  if player1.animCounter is 0:  set player1.animCounter equal to 8  if player1.move is “idle”:  player1.imgNum \_\_need to fix\_\_  player1.image \_\_need to fix\_\_  elif player1.move is “walk”:  player1.imgNum \_\_need to fix\_\_  player1.image \_\_need to fix\_\_  elif player1.move is “hp”:  player1.imgNum \_\_need to fix\_\_  player1.image \_\_need to fix\_\_  if player1.imgNum is 0:  set player1.move to “idle”  elif player1.move is “lk”:  player1.imgNum \_\_need to fix\_\_  player1.image \_\_need to fix\_\_  if player1.imgNum is 0:  set player1.move to “idle” |

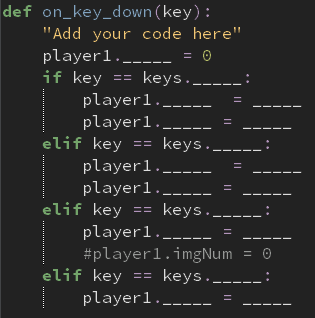
After adding the code, you should add this function under the draw function to animate.

1. Add the code for **on\_key\_down()** function.

: This function is called if the key is pressed, and we will modify the movement through this function. You will see the on\_key\_down() function like the picture below, and you should add the code under it.



3.a (usual way with the blank)



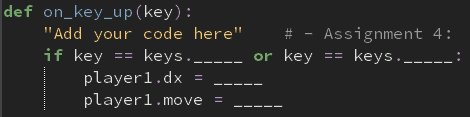
3.b (pseudo code)

|  |
| --- |
| def on\_key\_down(key):  set player1.imgNum equal to 0  if key is LEFT  set player1.dx equal to -2  set player1.move equal to “walk”  elif key is RIGHT  set player1.dx equal to 2  set player1.move equal to “walk”  elif key is UP  set player1.move equal to “hp”  elif key is DOWN  set player1.move equal to “lk” |

1. Add the code for **on\_key\_up()** function.

: This function is called when the key is released, and we will make the movement back to the ‘idle’ movement.

4.a (usual way with the blank)



4.b (pseudo code)

|  |
| --- |
| def on\_key\_up(key):  if key is LEFT or key is RIGHT:  set player1.dx equal to 0  set player1.move equal to “idle” |

1. Add the code for update() function.

Through update function, we will add continuous character movement. As the draw function working, the update function is automatically called by Pygame Zero engine 60 times per second. This function will make the character move back and forth.

You can implement this just by adding **player1.dx** to **player1.x** under this function.

## What to hand in:

Submit your project electronically through D2L by attaching and submitting your Python program file (assignment4.py).