**Week 15 - Think Aloud – Hangman**

For the final week, I completed the ending screens for my hangman game.

I added in a timer that tells you how long it took for you to complete the game:

****

I found a few different options to complete this task, but I found this way to be the easiest to manage:

First, import default\_timer from timeit

from timeit import default\_timer

then create a variable for the time start wherever you want to timer to start counting

time\_start = default\_timer()

When you want the time to stop counting, create a variable for the end time.

time\_end = default\_timer()

Then simply take the end time minus the start time and print the value.

display\_text(f" Time taken: {round(time\_end - time\_start)} seconds ", x=160, y=170, font=btn\_font)

For the winning screen, I displayed a title stating that the player won, how long it took them to complete the game, and the word they guessed. At the bottom I gave the option to play the game again or quit:

A screenshot of a video game

Description automatically generated

The losing game screen is similar, but I also added the final Hangman screen as you don’t get to see it when you make the final incorrect guess:

A screenshot of a video game

Description automatically generated

**Week 13 - Think Aloud – Hangman**

This week was a successful week in the land of Hangman. I have added in the images of the hangman that updates as the user makes incorrect guesses.

A black background with white lines

Description automatically generated

I discovered that I could use a range to change the image, and just tally through the amount of time the user made a wrong guess:

A computer screen with text

Description automatically generated

After the game is either won or lost, I have a rough draft of an end screen created:

A screenshot of a computer

Description automatically generated

Next, I am going to try to add a bit to the end screen. I’m considering keeping some kind of score or time. My main goal is to show the word if it was incorrectly guessed, and add the final hanged man to the screen:

A cartoon of a stick figure with a black background

Description automatically generated

**Week 11 - Think Aloud – Hangman**

After many hours of struggling with displaying a keyboard on the screen that disappears after a letter is selected, I finally found something that works.

The first part - make\_alphabet() creates the alphabet and the boxes and appends them together.

The second pat – draw\_buttons() draws the buttons and alphabet to the screen

**A computer screen shot of a program code

Description automatically generated**

To make the buttons disappear, I just used the collide point of the button that was created earlier and remove the button that was selected from the screen:

A screen shot of a computer code

Description automatically generated

I also started on the displaying the word that is being guessed. To do that, I just used a collide point for the buttons for the difficulty and have them open a file and pass a random word from the file to the game

A computer screen with text

Description automatically generated

It starts with just having an underscore (\_) for every character in the word, as a correct letter in the alphabet is guessed, the letter is inserted within the underscores.

A screen shot of a computer

Description automatically generated

This is the final result:

A screenshot of a phone

Description automatically generated

Next, I will be adding in the hangman image and limiting the number of guesses as well as creating the win/lose screens.

I found a YouTube video that was extremely helpful for figuring a lot of this out:

<https://www.youtube.com/watch?v=qhGQPm_iju0>

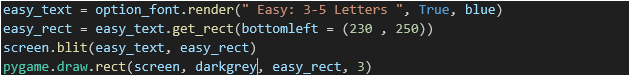
**Week 9 - Think Aloud – Hangman**

This week, I completed the option menu for selecting the difficulty in the game. I have also started the code for using the text files where the words are saved for the game.

A screenshot of a computer

Description automatically generated

I used a lot of what I learned for the previous screen when I created the difficulty options. I created my text, then my rectangles, and then I tied it all together using screen.blit(text, rectangle) so they would print to the screen in the same location.



The button selection is determined by where the rectangles are located on the screen to determine the difficulty.

A screen shot of a computer program

Description automatically generated

After that, those buttons open up the text files and choose a random word to pass onto play\_game.

A computer code on a black background

Description automatically generated

**Week 5 - Think Aloud – Hangman**

This week I have completed the beginning menu for my hangman game. The current working version is hangman\_v3.py.

I have added an icon to the Pygame window and renamed the window “Hangman!”:



* Code used to set the icon and caption:

# Set caption

pygame.display.set\_caption("Hangman!")

# Set icon for game

icon = pygame.image.load('icon.ico')

pygame.display.set\_icon(icon)

I also have a working menu screen with a background color and image:

**A neon light with a noose

Description automatically generated**

The “Play Game” button doesn’t do a whole lot so far – it just takes you to a window that says “Coming soon…”

A blue and white text

Description automatically generated

However, the “Quit Game” button functions as it should and exits the game.

**Week 4 - Think Aloud - Hangman**

My goal this semester is to create a game using pygame. I don’t have much experience with pygame, and I thought that this would be a good way to become more familiar with it. I will be using a Python that I created in my Python class – Hangman.

So far, I have added a few things to make the game a bit better before I start trying my hand at pygame:

* There are now 4 levels of difficulty: Easy, Medium, Hard, and Expert.
  + These are listed in a table using rich format

A screenshot of a computer game

Description automatically generated

* I have moved the word choice to 4 text files, rather than having a list in the main python program. I may eventually change this again – it is a work in progress. There are 199 words in each file for more variety.

A screenshot of a computer

Description automatically generated

The original word list I had paled in comparison. This was the list:

A black screen with text

Description automatically generated

* I also added in a while statement to force the player to make a correct game choice of 1-4.

A black background with white text

Description automatically generated

Right now, my hangman guy is a bit sad:

A black road with yellow faces

Description automatically generated

I would like to make the graphics a bit more fun. Right now, there are only 8 tries to guess the word. So, I want to be a bit more creative and maybe give the player a few more turns. If anyone has some suggestions, I would love to hear them!

In the current game, if you do not guess the word correctly, then you end the game never knowing what the word was. I think I may change this so the player will know what word they failed to guess.

The end goal will have a game menu where you will click a difficulty choice (easy – expert), then it will take you to the game. From there, it’s a bit up in the air! I suppose the whole point is to learn exactly how to create a fun game out of nothing!

Here is a link to the GitHub repository:

<https://github.com/h-m-harris/hangman.git>