Document of Hangman Game

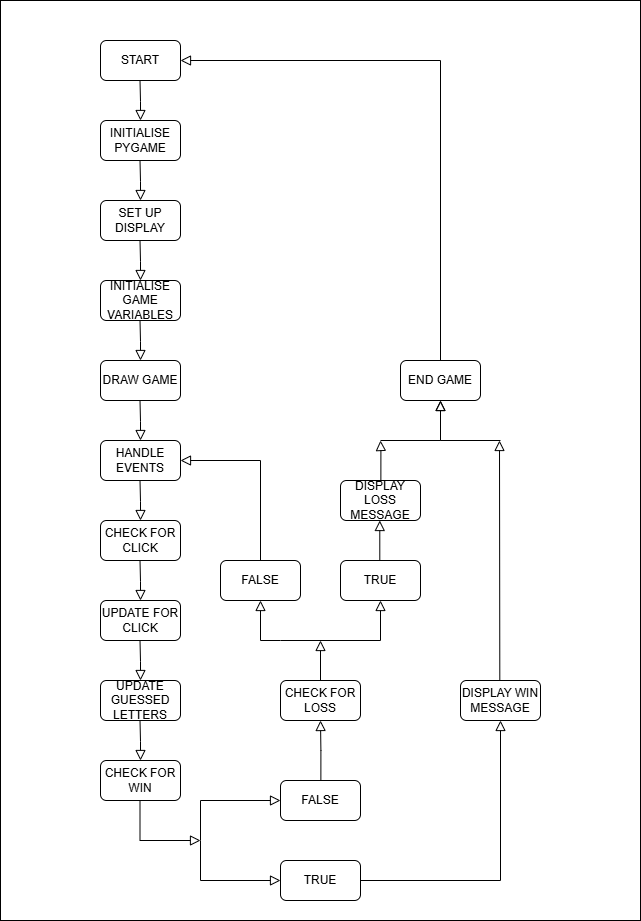
Gantt Chart Timeline

| Time | | WK1 | WK2 | WK3 | WK4 | WK5 | WK6 | WK7 | WK8 | WK9 | WK10 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Planning and Researching | |  |  |  |  |  |  |  |  |  |  |
| Sketching Ideas | |  |  |  |  |  |  |  |  |  |  |
| Programming  Using Python | |  |  |  |  |  |  |  |  |  |  |
| Modification of Code | |  |  |  |  |  |  |  |  |  |  |
| Documenting and Finalising | |  |  |  |  |  |  |  |  |  |  |

Programming

The programming language that I am using in this game is Python, with the extension of turning this code as a Pygame. The code of this game is created and modified via using the app Visual Studio Code. It is also where I have modified the issues with this game and made an improvement.

Flowchart



Pseudocode

# Initialize Pygame

initialize Pygame

# Set window dimensions

set window width and height

# Create the game window

create game window with title "Hangman Game"

# Button variables

set button radius

set gap between buttons

create a list of letters (A-Z) with their positions and visibility status

# Fonts

set fonts for letters, words, and the title

# Load images

load hangman images (0-6)

# Game variables

set initial hangman status

create a list of words to choose from

select a random word from the list

create an empty list for guessed letters

# Colors

define colors for white, black, and red

# --- Functions ---

# Draw function

fill the window with white

draw the title

create a string to display the word

for each letter in the word

if the letter is guessed, add it to the display string

else, add an underscore to the display string

draw the word on the screen

draw buttons for each letter

if the letter is visible

draw a circle for the button

draw the letter on the button

draw the current hangman image

update the display

# Display message function

pause for 1 second

fill the window with white

draw the message on the screen

update the display

pause for 2 seconds

# Set frames per second (FPS)

set FPS

# Create a clock object

create a clock

# Start the game loop

while the game is running

limit the frame rate to FPS

handle events

if the event is quit

set game running to false

if the event is mouse click

get mouse position

for each letter

if the letter is visible

calculate distance from mouse click to letter

if distance is less than button radius

set letter visibility to false

add letter to guessed letters

if the letter is not in the word

increase hangman status

draw the game elements

check if the player has won

set won to true

for each letter in the word

if the letter is not guessed

set won to false

break

if the player has won

display a "You WON!" message

break out of the loop

if the player has lost

display a "You LOST!" message

break out of the loop

# --- End of Game ---

# Quit Pygame

quit Pygame

Modification

I have modified various things in this Hangman Python game such as the window size, element spacing, added custom words and adding images into the game. For the window size, I increase the dimension of the window to 1920x720 as shown in win = pygame.display.set\_mode((WIDTH, HEIGHT)). For button spacing, I bring the buttons into the bottom side of the screen to create more space for the words as shown in GAP = 25

and y = starty + ((i // 13) \* (GAP + RADIUS \* 2)). I also added an additional color to help distinguish between word and letters in RED = (213, 73, 36). I have also increase the number of custom words in the game as shown in words = ["PYTHON", "PYGAME", "CODE", "GITHUB", "VISUALSTUDIOCODE", "HANGMAN"].

As for a new function I am adding, it is called displaying images. This function allow the game to display an image in the file if an event was to occur, in this case I put the image in the defeat screen when you lost in the game:

# function to display an image

def display\_image(image\_path, scale):

"""Loads and scales an image for display."""

try:

image = pygame.image.load(image\_path) # Load the image

image = pygame.transform.scale(image, (scale, scale)) # Scale the image

return image

except pygame.error as e:

print(f"Error loading image {image\_path}: {e}")

return None # Return None if image loading fails

# load images. Add trollege image

images = []

for i in range(7):

image = pygame.image.load("hangman" + str(i) + ".png")

images.append(image)

try:

trollege\_image = pygame.image.load("trollege.jfif")

except pygame.error as e:

print(f"Error loading trollege.jfif: {e}")

trollege\_image = None # Handle the case where the image is not found

For the defeat screen function I made the code into:

# Defeat screen

if hangman\_status == 6:

display\_message("You LOST!")

if trollege\_image:

# Create a list of rects for multiple images

trollege\_rects = []

for \_ in range(50): # Generate 50 images

x = random.randint(0, WIDTH - trollege\_image.get\_width())

y = random.randint(0, HEIGHT - trollege\_image.get\_height())

trollege\_rects.append(trollege\_image.get\_rect(topleft=(x, y)))

# Blit multiple images

for rect in trollege\_rects:

win.blit(trollege\_image, rect)

pygame.display.update()

pygame.time.delay(1000)

break