### **1. Resolution and Window Size:**

* **First Code**: The resolution is set to WIDTH, HEIGHT = 800, 500.
* **Second Code**: The resolution is set to WIDTH, HEIGHT = 1480, 720.
* **Reason**: Increasing the window size improves the game's visual experience by providing more space to display elements such as the word, buttons, and images. It makes the game look more modern and visually engaging, especially for larger screens.

### **2. Image File Path:**

* **First Code**: The images are loaded directly with a relative path: image = pygame.image.load("hangman" + str(i) + ".png").
* **Second Code**: The images are loaded using a path with a specific directory for images: image = pygame.image.load("images/hangman" + str(i) + ".png") and image\_path = os.path.join(os.path.dirname(\_\_file\_\_), "images", f"hangman{i}.png").
* **Reason**: The change ensures that the images are loaded correctly even if the script is run from a different location. By using os.path.join, you ensure the game can find the image files in the images folder relative to the script’s location, making it more portable.

### **3. Font Size Adjustments:**

* **First Code**: The font size for letters is set to 40, and the word font is set to 60.
* **Second Code**: The font size for letters is reduced to 30, and the word font remains at 60.
* **Reason**: Reducing the font size for the letters allows for a cleaner and less cluttered display, especially in the larger window, where you have more space to work with. This ensures that all the letters fit well on the screen.

### **4. Game State (Wins and Losses) Tracking:**

* **First Code**: There is no tracking of wins and losses.
* **Second Code**: You added variables to track wins and losses (wins, losses), and these are displayed on the menu screen.
* **Reason**: Tracking wins and losses improves the game experience by providing feedback to the player about their performance. It also adds a competitive element that can motivate players to keep playing.

### **5. Menu Screen:**

* **First Code**: The game starts immediately when run, with no main menu or option to quit.
* **Second Code**: You added a menu screen that displays the current win/loss statistics and allows players to either start a new game (via pressing ENTER) or quit (via pressing ESC).
* **Reason**: Adding a menu screen improves the overall user experience by giving players a chance to see their progress (wins and losses) and decide whether they want to play again or exit the game.

### **6. Key Handling:**

* **First Code**: The game only processes mouse clicks to select letters.
* **Second Code**: The game now allows key presses as well (pygame.KEYDOWN), where players can press a key to guess a letter instead of just clicking the buttons.
* **Reason**: This change improves user experience, as players can use the keyboard for quicker interaction, especially on devices where using a mouse might be less intuitive or desirable.

### **7. Game Over and Message Display:**

* **First Code**: When the game ends (win/loss), a message is displayed for 3 seconds, then the game resets.
* **Second Code**: The game now displays a message indicating the correct word when the player loses and tracks wins and losses.
* **Reason**: Showing the correct word when the player loses helps with learning, and keeping track of wins/losses makes the game more competitive and fun.

### **8. Function Structure:**

* **First Code**: The main() function runs indefinitely until the game is quit.
* **Second Code**: The game\_loop() function first shows the menu, then starts the game upon pressing ENTER and allows quitting via ESC.
* **Reason**: Separating the menu and game loop into distinct functions improves the code structure and readability. It also separates concerns menu management vs. game logic making the code easier to maintain and extend in the future.

### **9. Improved Word List:**

* **First Code**: The word list is small, with only a few options.
* **Second Code**: The word list is significantly larger, with a variety of interesting and challenging words.
* **Reason**: Expanding the word list increases replayability, providing more variety and challenges for players.

**Gantt Chart Timeline**

| Task/Weeks | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Week 5 |  | Design of game | Continuing development of game | Continuing development of game | Testing game for bugs |  |  |
| Week 6 |  | Testing game for bugs and fixing bugs | Researching solutions to bugs | Fixing code to remove bugs | Fixing code to remove bugs |  |  |
| Week 7 | Play testing with friends | Designing new features in game | Add new features to game | Testing game for bugs | Testing game and fixing bugs | Researching solutions to bugs |  |
| Week 8 | Play testing with friends | Adding code suggested from friends | Checking for file errors | Fixing files  and doing flowchart | Add files and other documents to github | Fixing file structure to upload to github |  |
| Week 9 | Finishing structure | Checking for more bugs/errors | Play testing | Fixing final errors | Commenting  And doing pseudo code | Writing README file |  |
| Week 10 | Play testing with friends and uploading | Uploaded |  |  |  |  |  |

**Pseudocode**

START

INITIALIZE Pygame library

SET up game display (WIDTH, HEIGHT)

CREATE variables for game settings (letters, word list, images, etc.)

LOAD fonts and images (for hangman status)

SET initial game variables (hangman\_status, wins, losses, etc.)

SET up the starting word randomly from the word list

FUNCTION draw():

CLEAR the screen

DISPLAY the title ("DON'T LET BRO DIE!")

DISPLAY the word, replacing unguessed letters with underscores ("\_")

DISPLAY available letters as buttons

DISPLAY hangman image based on current hangman status

UPDATE the screen

FUNCTION display\_message(message):

DISPLAY a message (win/lose condition)

WAIT for a few seconds

CLEAR the screen

FUNCTION show\_menu():

DISPLAY the game menu (title, wins, losses, play option, quit option)

WAIT for user input (press ENTER to play, ESC to quit)

FUNCTION main\_game():

RESET game variables (hangman\_status, guessed letters, new word)

RESET letter visibility for the alphabet buttons

WHILE the game is running:

HANDLE user events (mouse click or keyboard press)

IF the user clicks on a letter or presses a key:

MARK that letter as guessed

IF the letter is in the word:

CONTINUE to check for win condition

ELSE:

UPDATE hangman status (increment failure count)

CHECK if all letters are guessed (win condition):

CALL display\_message("You Saved Bro!")

INCREMENT win count

BREAK out of the game loop

CHECK if hangman\_status reaches 6 (lose condition):

CALL display\_message("You Let Bro Die! The word was: " + word)

INCREMENT loss count

BREAK out of the game loop

FUNCTION game\_loop():

WHILE the game is running:

CALL show\_menu() to display the game menu

HANDLE user input (ENTER to start game, ESC to quit)

IF the user presses ENTER:

CALL main\_game() to start a new game

IF the user presses ESC:

EXIT the game

START the game loop

END

**FlowChart**