Fixes for Final_Version.py

Oliver Simons – Quality Assurance

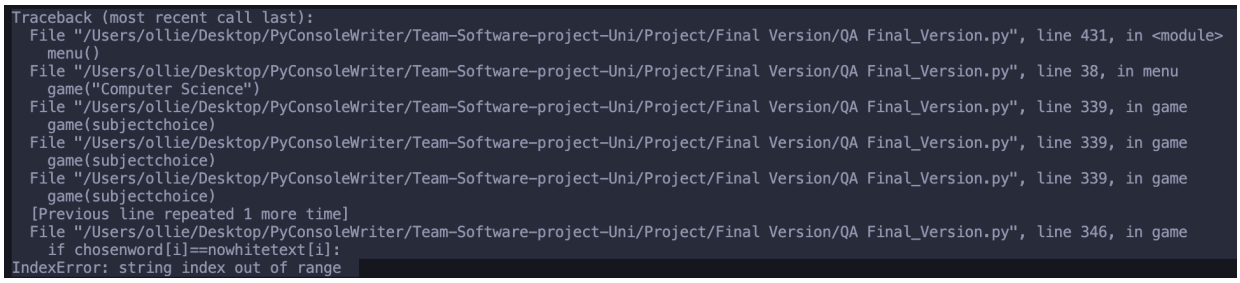
SMOL TEAM

To improve the code readability, maintainability, performance, and functionality. I have implemented a few improvements to the code to ensure all users have a good error free experience while using our software. These fixes help improve:

- Code quality to make the code easier to read, debug and extend.
- Performance to help reduce resource usage and redundant processes.
- Robustness to handle random errors to atop crashes.
- User experience to make the game easier to use and more fun.

I performed extensive testing on the software code, testingVL for different use cases to ensure all things worked as intended and to notice any potential issues that could lead to reduced functionality or crashes. Below are some of the errors or issues I found during testing which I later fixed.

1. An error related to python's 'IndexError' would cause the entire program to crash, this was due to the program comparing two variables and going out of range.



2. All images are loaded each time the game refreshed which caused stress on the processor due to constant image streaming.

```
running = True
CLICK = False

while running:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    screen.fill("white")

    bg=pygame.image.load("Wordie Logo.png").convert()
    largebg=pygame.transform.scale(bg,(460,75))
    screen.blit(largebg,(0,50))

    ENGLISH_BUTTON =pygame.image.load("English Button.png").convert()
    ENGLISH_BUTTON_LARGE =pygame.transform.scale(ENGLISH_BUTTON,(420,69))
    screen.blit(ENGLISH_BUTTON_LARGE,(20,280))

    MATHS_BUTTON =pygame.image.load("Maths Button.png").convert()
    MATHS_BUTTON_LARGE =pygame.transform.scale(MATHS_BUTTON,(420,69))
    screen.blit(MATHS_BUTTON_LARGE,(20,200))

    COMPSCI_BUTTON =pygame.image.load("CompSci Button.png").convert()
    COMPSCI_BUTTON_LARGE =pygame.transform.scale(COMPSCI_BUTTON,(420,69))
    screen.blit(COMPSCI_BUTTON_LARGE,(20,360))
```

3. The system used for detecting clicks was terribly slow and would cause the software to freeze for an extended period, this is related to the ‘CLICK=TRUE/FALSE’ variable.

```
for event in pygame.event.get():
    if event.type == pygame.MOUSEBUTTONDOWN :
        MENU_MOUSE_POS = pygame.mouse.get_pos()
        MENU_MOUSE_POS_X, MENU_MOUSE_POS_Y = MENU_MOUSE_POS
        CLICK = True
        if (CLICK == True) and (24 <= MENU_MOUSE_POS_X <= 438) and (207 <= MENU_MOUSE_POS_Y <= 265):
            print("Maths")
            subjectchoice="Maths"
            game(subjectchoice)
quote stuff  #wordList = maths
            #pygame.quit()
            #game("maths")
        elif (CLICK == True) and (24 <= MENU_MOUSE_POS_X <= 438) and (283 <= MENU_MOUSE_POS_Y <= 346):
            print("English")
            subjectchoice="English"
            game(subjectchoice)
            #wordList = english
            #pygame.quit()
            #game("english")
        elif (CLICK == True) and (24 <= MENU_MOUSE_POS_X <= 438) and (367 <= MENU_MOUSE_POS_Y <= 424):
            print("Computer Science")
            subjectchoice="Computer Science"
            game(subjectchoice)
```

4. Code related to drawing boxes was the same line consistently repeated only with a change to location, this could deal with a refactor to try and avoid future repeated code.

```
def drawcolorboxes(boxcolors):
    pygame.draw.rect(screen, boxcolors[0], pygame.Rect(32, 82, 56, 56))
    pygame.draw.rect(screen, boxcolors[1], pygame.Rect(112, 82, 56, 56))
    pygame.draw.rect(screen, boxcolors[2], pygame.Rect(192, 82, 56, 56))
    pygame.draw.rect(screen, boxcolors[3], pygame.Rect(272, 82, 56, 56))
    pygame.draw.rect(screen, boxcolors[4], pygame.Rect(352, 82, 56, 56))
    pygame.draw.rect(screen, boxcolors[5], pygame.Rect(32, 157, 56, 56))
    pygame.draw.rect(screen, boxcolors[6], pygame.Rect(112, 157, 56, 56))
    pygame.draw.rect(screen, boxcolors[7], pygame.Rect(192, 157, 56, 56))
    pygame.draw.rect(screen, boxcolors[8], pygame.Rect(272, 157, 56, 56))
    pygame.draw.rect(screen, boxcolors[9], pygame.Rect(352, 157, 56, 56))
    pygame.draw.rect(screen, boxcolors[10], pygame.Rect(32, 232, 56, 56))
    pygame.draw.rect(screen, boxcolors[11], pygame.Rect(112, 232, 56, 56))
    pygame.draw.rect(screen, boxcolors[12], pygame.Rect(192, 232, 56, 56))
    pygame.draw.rect(screen, boxcolors[13], pygame.Rect(272, 232, 56, 56))
    pygame.draw.rect(screen, boxcolors[14], pygame.Rect(352, 232, 56, 56))
    pygame.draw.rect(screen, boxcolors[15], pygame.Rect(32, 307, 56, 56))
    pygame.draw.rect(screen, boxcolors[16], pygame.Rect(112, 307, 56, 56))
    pygame.draw.rect(screen, boxcolors[17], pygame.Rect(192, 307, 56, 56))
    pygame.draw.rect(screen, boxcolors[18], pygame.Rect(272, 307, 56, 56))
    pygame.draw.rect(screen, boxcolors[19], pygame.Rect(352, 307, 56, 56))
    pygame.draw.rect(screen, boxcolors[20], pygame.Rect(32, 382, 56, 56))
    pygame.draw.rect(screen, boxcolors[21], pygame.Rect(112, 382, 56, 56))
    pygame.draw.rect(screen, boxcolors[22], pygame.Rect(192, 382, 56, 56))
    pygame.draw.rect(screen, boxcolors[23], pygame.Rect(272, 382, 56, 56))
    pygame.draw.rect(screen, boxcolors[24], pygame.Rect(352, 382, 56, 56))
    pygame.draw.rect(screen, boxcolors[25], pygame.Rect(32, 457, 56, 56))
    pygame.draw.rect(screen, boxcolors[26], pygame.Rect(112, 457, 56, 56))
    pygame.draw.rect(screen, boxcolors[27], pygame.Rect(192, 457, 56, 56))
    pygame.draw.rect(screen, boxcolors[28], pygame.Rect(272, 457, 56, 56))
    pygame.draw.rect(screen, boxcolors[29], pygame.Rect(352, 457, 56, 56))
```

Below is the list of improvements, the improved QA code can be found on the GitHub Repository, alongside the original Final_Version.py.

1. Consolidated Image loading - By moving image loading and scaling outside of loops it helps avoid wasting resources as the software doesn't need to repeat tasks more than once.
2. Refactored repeated code – I created reusable functions including 'drawcolorboxes' and 'drawboxes' to try and reduce repeated code for drawing boxes for the characters on guessing screen.
3. Improved variable names – I changed commonly used variable names to be more descriptive to improve readability of the code. These included 'MENU_MOUSE_POS_X' and "MENU_MOUSE_POS_Y'.
4. Optimised event handling – I improved event handling logic to avoid repeating redundant checks which improved performance and readability of the code.
5. Removed unused variables – I removed unused variables including 'CLICK=TRUE/FALSE' used to check if the mouse was clicked before I changed to a new system.
6. Organised code – I organised code using vscode's built in code formatting tools to ensure the code was split into sections for better readability and maintainability.
7. Fixed scoring logic – I rewrote the code which oversees the logic for scoring and updating box colours based on the guessed word. It's now quicker and easier to read and maintain.
8. Fixed 'IndexError' – I fixed an error related to IndexError in the word comparison when comparing 'chosenword' and 'nowhitetext' to only iterate up to the length of

the shorter string using ‘min(len(chosenword), len(nowhitetext))’. This makes sure the program doesn’t attempt to access indices that are out of range preventing the software from crashing.