PROJECT REPORT

Group 2

HANGMAN GAME

दिल्ली विश्वविद्यालय
University of Delhi

62347628 : DISSERTATION / PROJECT WORK
Semester 6
B.A. Programme

Department of Computer Science
P. G. D. A. V. College
Nehru Nagar, New Delhi- 110065

Submitted by:

Abhinav Yadav	**20053501009**
Varenya Girish Hegde	**20053501017**
Yashvardhan Shukla	**20053501022**

Under the Guidance of:

Dr. Geeta Aggarwal
Assistant Professor

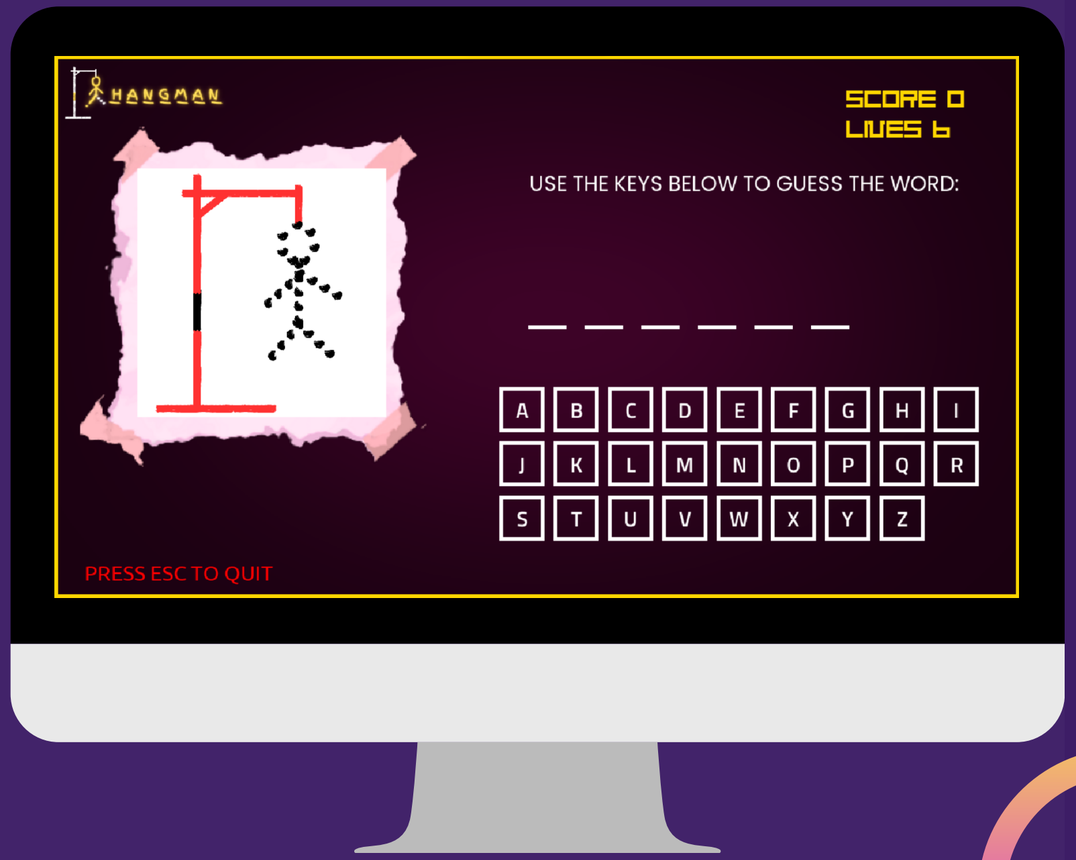
INTRODUCTION

Hangman is a **Word Guessing Game** for two or more players. One player thinks of a word, phrase or sentence and the other(s) tries to guess it by suggesting letters within a certain number of guesses. Originally a **Paper-and-Pencil** game, there are now **Electronic Versions**.

We have created one such electronic version of Hangman using python.

OVERVIEW:

•The game starts with, the word to guess is represented by a row of dashes, representing each letter of the word.
•If the suggested letter does not occur in the word, the player draws one element of a hanged man stick figure
•If a guess is correct , the points are assigned based on the guesses.
•The hangman has six body parts that are drawn. (for 6 letter guesses) in the order .
• The round ends either when the player has guessed all the word correctly, or after six incorrect guesses when the man has been hanged.



LANGUAGE USED:

Python: It is a high level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically- typed and garbage-collected language.

SYSTEM REQUIREMENTS:

- A computer with at least a 1GHz processor
- At least 512MB of RAM
- A graphics card capable of rendering 2D graphics
- Python 3.x installed

However, it is recommended to have a more powerful computer to ensure smooth performance and a better gaming experience.

Softwares used:

Visual Studio Code

It is streamlined code editor with support of development operations like debugging, task running etc.

Canva

A Graphic designing platform that is used to create graphics and presentations. In game background graphics and characters are designed using the app.

Python libraries used :

- **pygame:** a set of Python modules designed for writing video games.
- **random:** a module that implements pseudo-random number generators for various uses.
- **sys:** a module that provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.

FUNCTIONS USED IN CODE:

pygame.init()

This function initializes all the imported pygame modules.

pygame.display.set_mode()

This function sets the display mode and returns a surface object representing the game window.

pygame.time.Clock()

This function creates a Clock object that can be used to control the frame rate of the game.

pygame.font.Font()

This function creates a Font object that can be used to render text in a particular font and size.

pygame.mixer.Sound()

This function loads a sound file and returns a Sound object that can be used to play the sound.

pygame.image.load()

This function loads an image file and returns a Surface object that can be used to display the image.

random.choice()

This function chooses a random element from a sequence.

pygame.Surface()

This function creates a new Surface object with the specified dimensions.

pygame.Rect()

This function creates a new Rect object with the specified coordinates and dimensions.

pygame.sprite.Sprite()

This function creates a new Sprite object.

Font.render()

This function renders the specified text using the font and size of the Font object and returns a Surface object representing the rendered text.

Rect.collidepoint()

This function checks if the given point is inside the Rect object.

FUNCTIONS USED IN CODE:

pygame.draw.rect()

This function draws a rectangle on the surface object.

win.blit()

This function copies the contents of one surface to another surface.

Button.collision()

This function checks if the mouse cursor is over the button and returns True if the button is clicked.

Button.update()

This function updates the button by drawing it on the game window.

Button.reset()

This function resets the button to its default state.

FadeScreen.update()

This function updates the alpha value of the surface to create a fading effect.

FadeScreen.draw()

This function draws the surface on the game window.

Button()

This class is defined in a separate button module. It contains methods to draw and update the buttons and check for mouse clicks.

get_font()

This function takes in a font size and returns a pygame font object with the desired size.

play()

This function is called when the user clicks on the "START" button in the main menu. It imports and calls the testing_pop module, which contains the main logic of the game.

options()

This function is called when the user clicks on the "CREDITS" button in the main menu. It displays the group project name and a "BACK" button, which returns the user to the main menu when clicked.

main_menu()

This function is the main entry point of the game and is called when the script is run. It displays the game title, background image, and three buttons ("START", "CREDITS", and "QUIT"). It waits for the user to click on one of the buttons and calls the appropriate function depending on the user's input

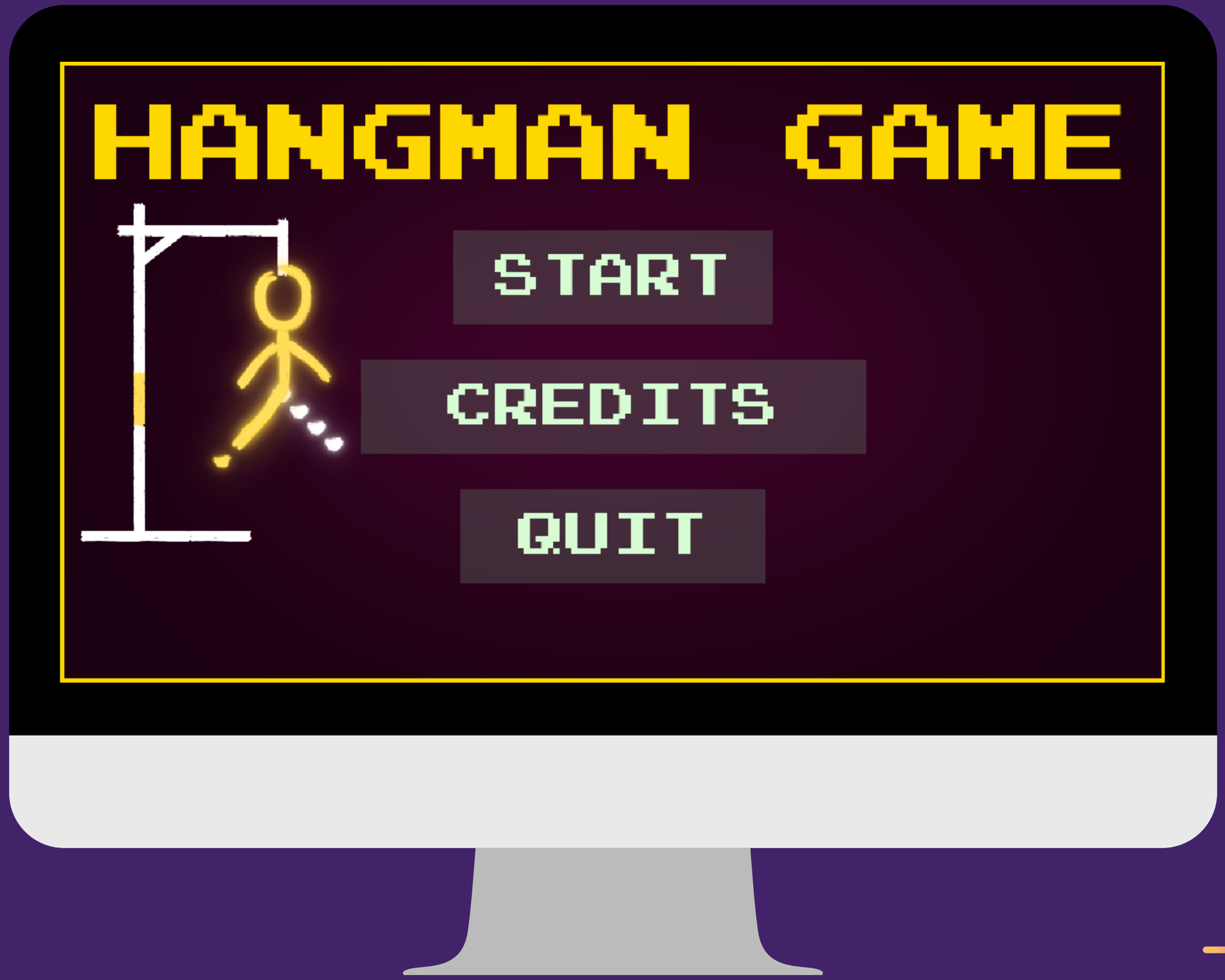
CODE

MAIN MENU

```
def main_menu():
    while True:
        SCREEN.blit(BG, (0, 0))


        MENU_MOUSE_POS = pygame.mouse.get_pos()
        pygame.draw.rect(SCREEN, (255,215,0), (0,0,screen_width,screen_height), 5)
        MENU_TEXT = get_font(100).render("HANGMAN GAME", True, "#ffd700")
        MENU_RECT = MENU_TEXT.get_rect(center=(640, 100))


        PLAY_BUTTON = Button(image=pygame.image.load("Assets/Play Rect.png"), pos=(640, 250),
                             text_input="START", font=get_font(55), base_color="#d7fcd4", hovering_color="White")
        OPTIONS_BUTTON = Button(image=pygame.image.load("Assets/Options Rect.png"), pos=(640, 400),
                             text_input="CREDITS", font=get_font(55), base_color="#d7fcd4", hovering_color="White")
        QUIT_BUTTON = Button(image=pygame.image.load("Assets/Quit Rect.png"), pos=(640, 550),
                             text_input="QUIT", font=get_font(55), base_color="#d7fcd4", hovering_color="White")


        SCREEN.blit(MENU_TEXT, MENU_RECT)


        for button in [PLAY_BUTTON, OPTIONS_BUTTON, QUIT_BUTTON]:
            button.changeColor(MENU_MOUSE_POS)
            button.update(SCREEN)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if PLAY_BUTTON.checkForInput(MENU_MOUSE_POS):
                    play()
                if OPTIONS_BUTTON.checkForInput(MENU_MOUSE_POS):
                    options()
                if QUIT_BUTTON.checkForInput(MENU_MOUSE_POS):
                    pygame.quit()
                    sys.exit()


        pygame.display.update()
```

OUTPUT

MAIN MENU



CODE

MENU OPTIONS

```
class Button():
    def __init__(self, image, pos, text_input, font, base_color, hovering_color):
        self.image = image
        self.x_pos = pos[0]
        self.y_pos = pos[1]
        self.font = font
        self.base_color, self.hovering_color = base_color, hovering_color
        self.text_input = text_input
        self.text = self.font.render(self.text_input, True, self.base_color)
        if self.image is None:
            self.image = self.text
        self.rect = self.image.get_rect(center=(self.x_pos, self.y_pos))
        self.text_rect = self.text.get_rect(center=(self.x_pos, self.y_pos))
```

```
for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if PLAY_BUTTON.checkForInput(MENU_MOUSE_POS):
                    play()
                if OPTIONS_BUTTON.checkForInput(MENU_MOUSE_POS):
                    options()
                if QUIT_BUTTON.checkForInput(MENU_MOUSE_POS):
                    pygame.quit()
                    sys.exit()
```

OUTPUT

MENU OPTIONS

CODE

FADING SCREEN

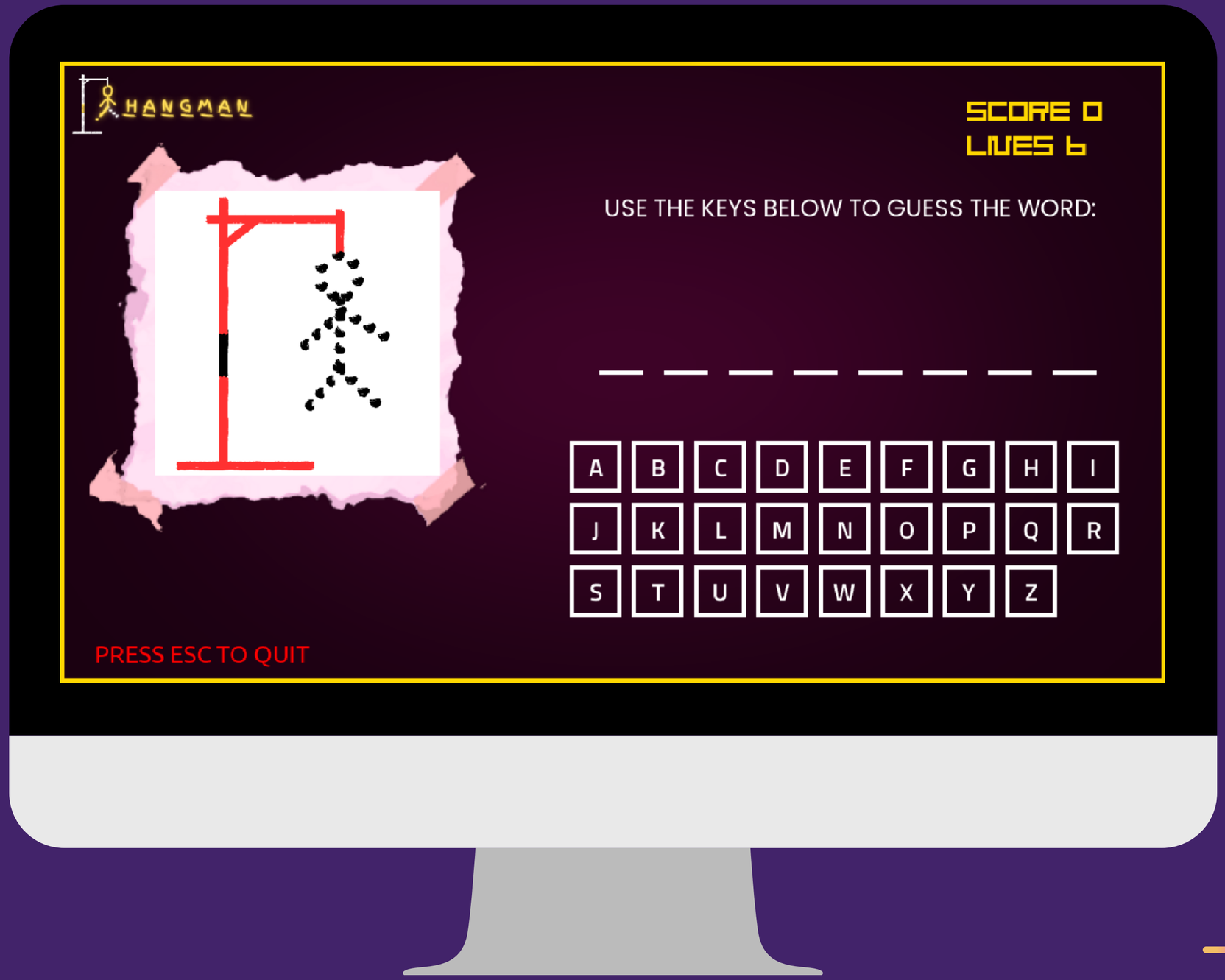
```
class FadeScreen:
    def __init__(self, w, h, colour):
        self.surface = pygame.Surface((w, h))
        self.surface.fill(colour)
        self.alpha = 255
        self.surface.set_alpha(self.alpha)

    def update(self):
        self.alpha -= 2
        self.surface.set_alpha(self.alpha)

    def draw(self, x, y):
        win.blit(self.surface, (x,y))


fadeScreen = FadeScreen(screen_width, screen_height, BLACK)



if homepage:
        if fadeScreen.alpha >= 0:
            fadeScreen.update()
            fadeScreen.draw(0,0)

            opening_animation.set_alpha(fadeScreen.alpha)
            win.blit(opening_animation, (screen_width - opening_animation.get_width() , screen_height -
opening_animation.get_height()))
```

OUTPUT

FADING SCREEN

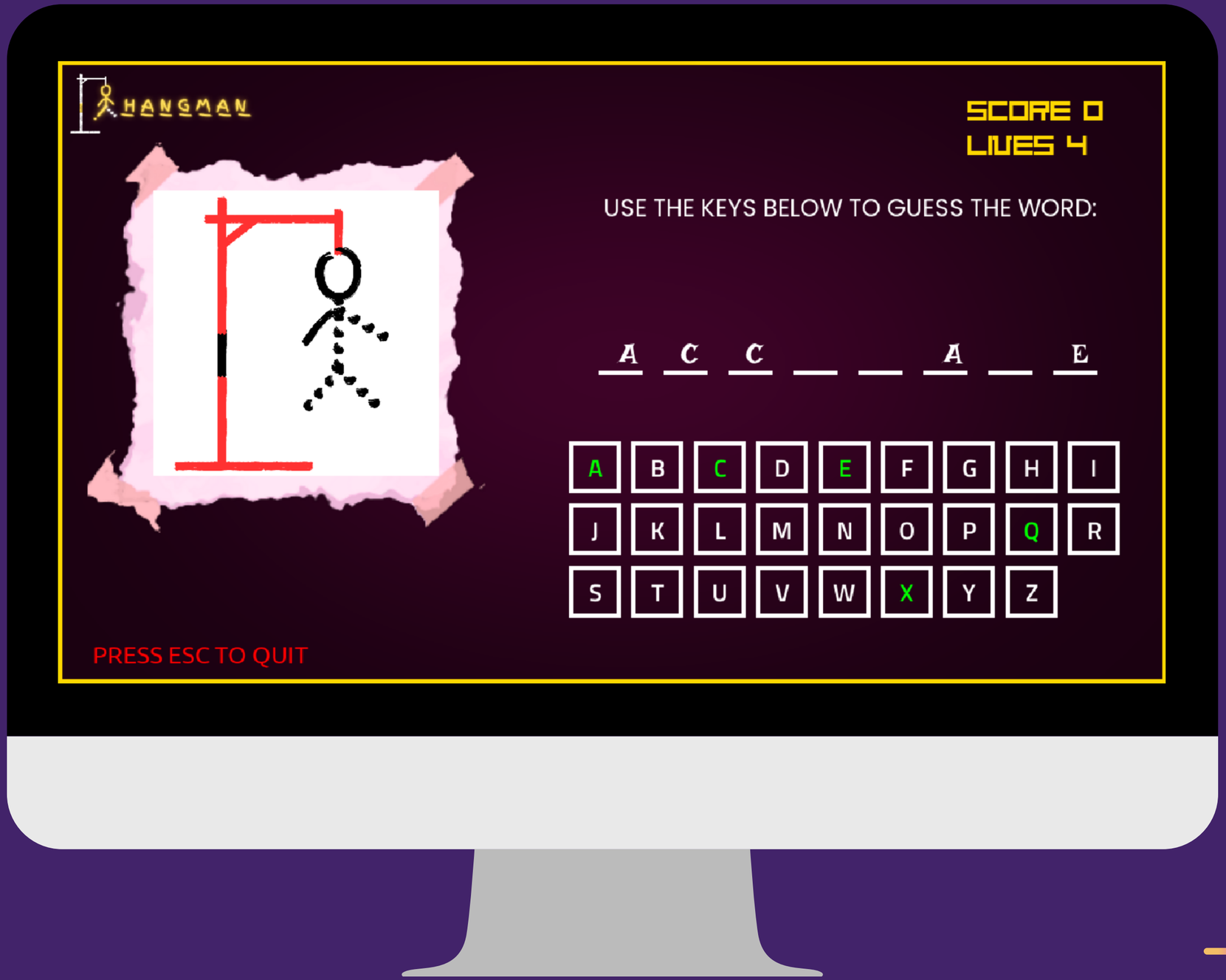
OUTPUT

BASIC GAME SCREEN



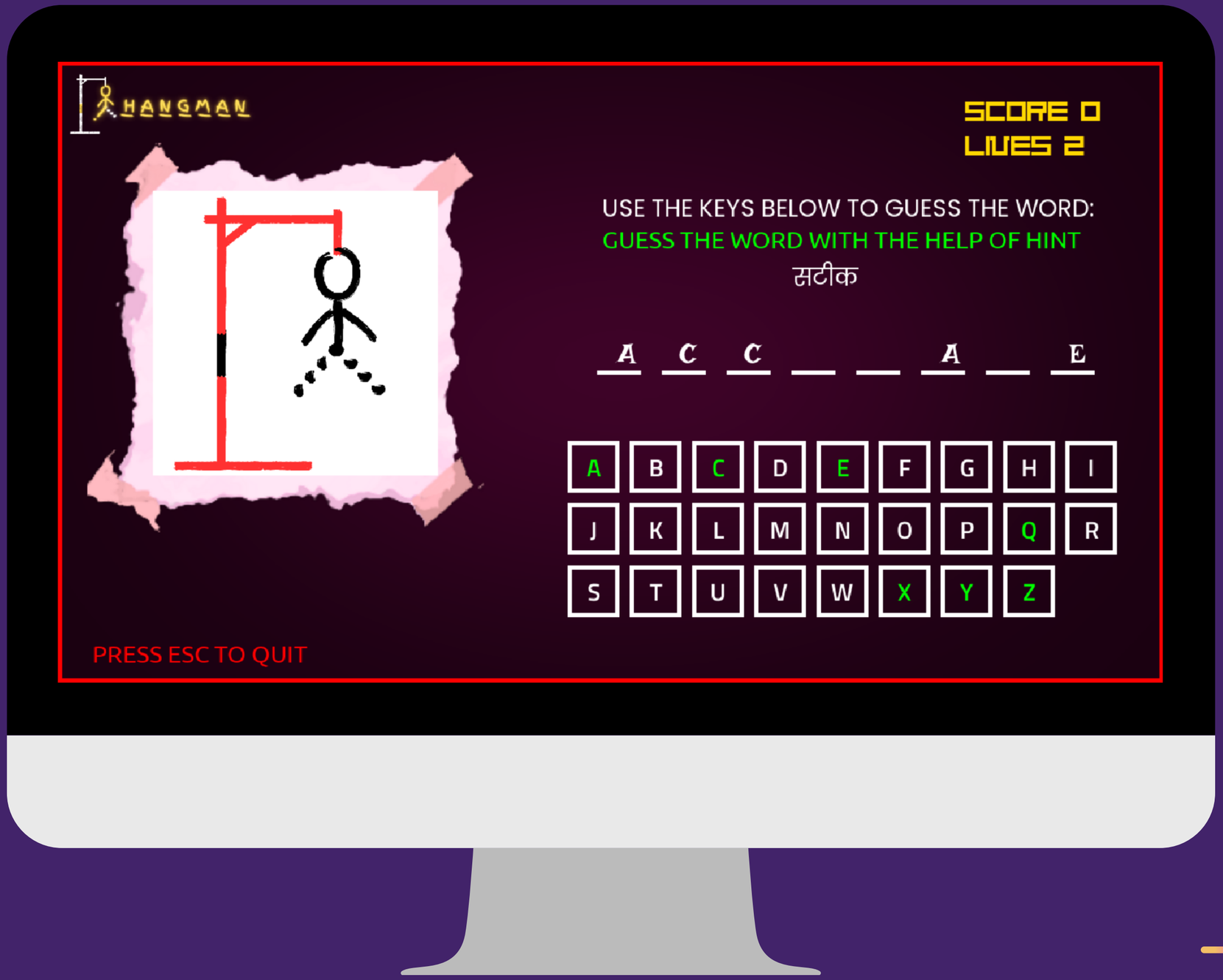
OUTPUT

WHILE PLAYING



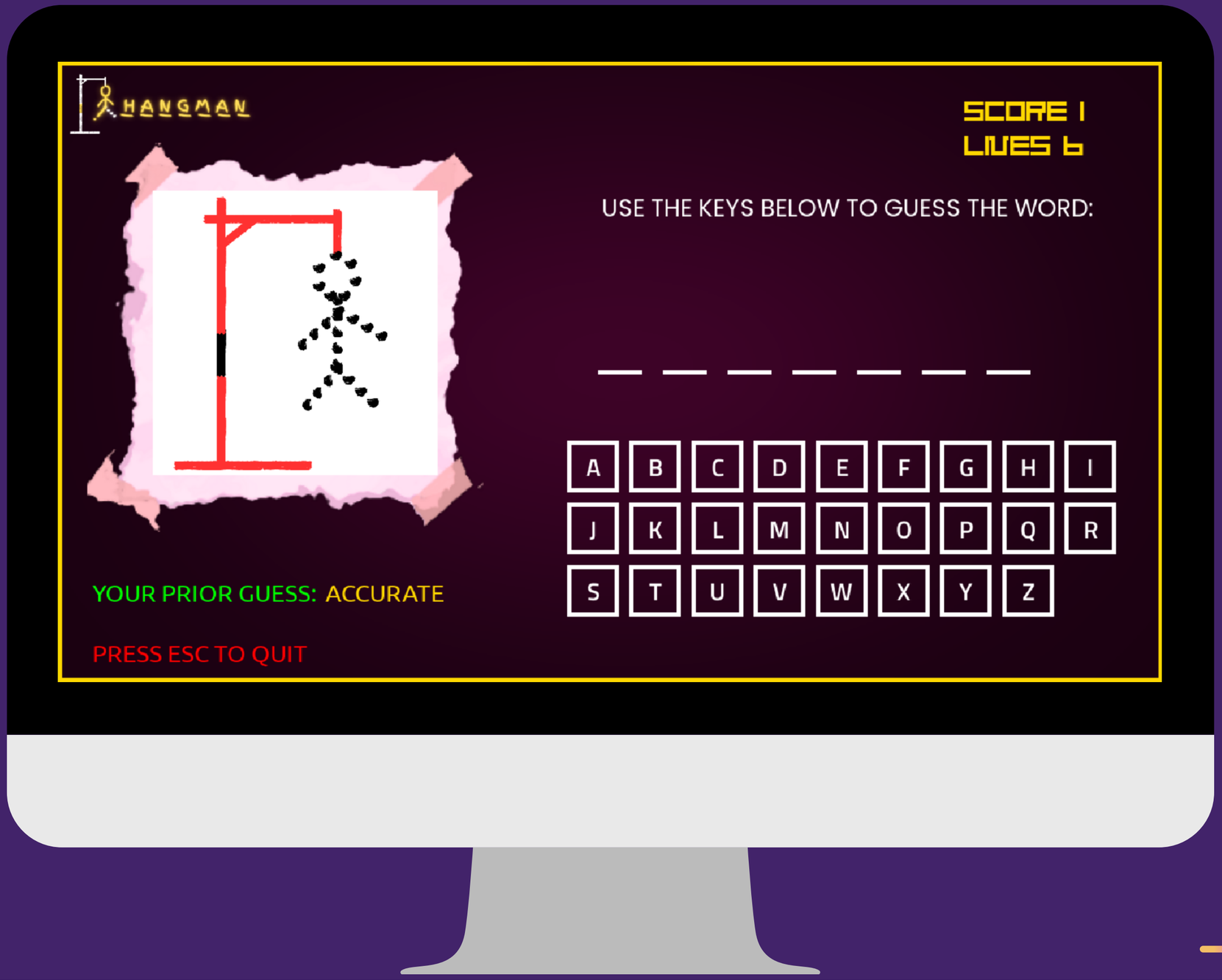
OUTPUT

ALERT AND HINT @ LIVES = 2



OUTPUT

AFTER A RIGHT ANSWER



OUTPUT

UNABLE TO GUESS

THANK YOU