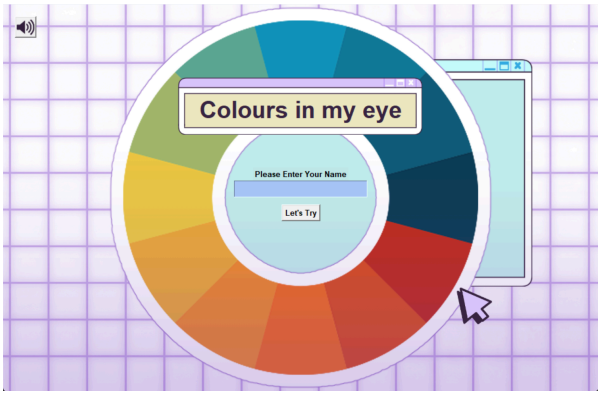
Introduction:

The app was designed to let more people learn about the world of color blind people and more about color theories. I got inspired by such an idea since I always wonder how a world could be visually different from different perspectives. Later, I decided to add in game function and color mixing tools to teach about the misconception of the three primary colors and the difference between additive and subtractive colors as people generally mix them up quite a lot of times.

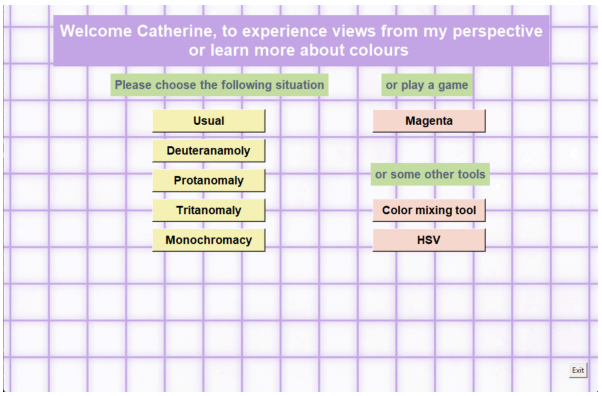
Methodology & Design:

Users would be greeted by this welcome page when they run the app.py.

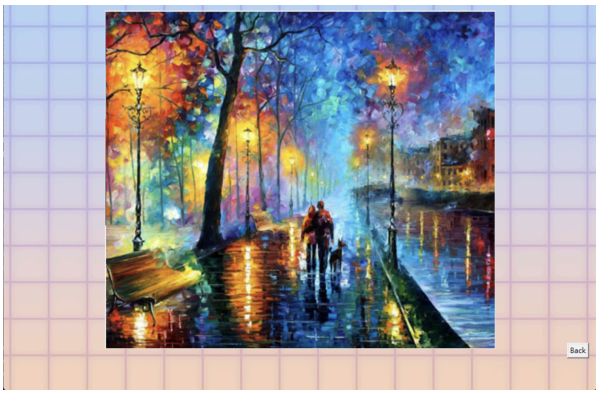


They can enter their name and press the “Let’s Try” button or press Enter on the keyboard to the next page.

Users can also choose to mute the background music on the button shown on the top left.



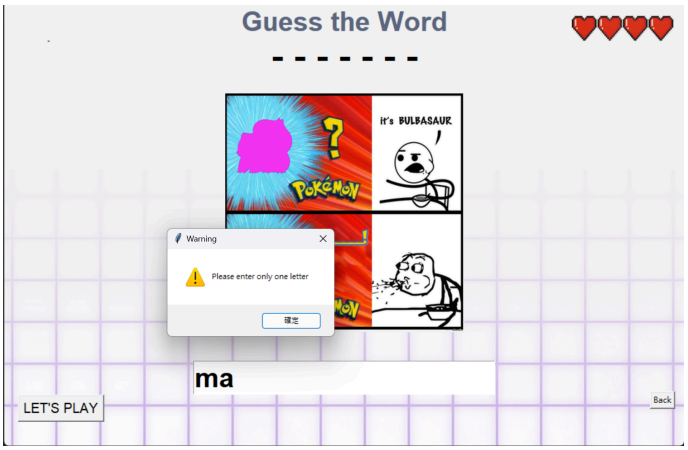
Users would arrive in the menu page where they can choose to experience different situations or play a game or try out some tools.



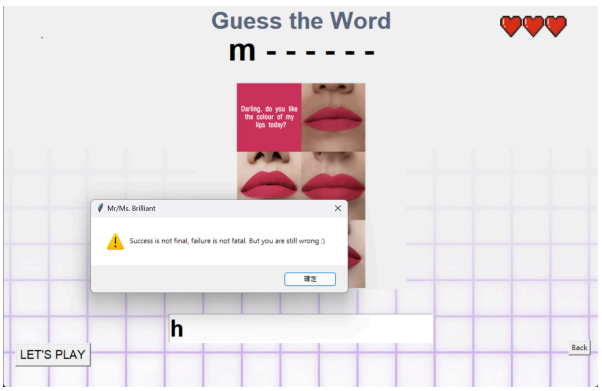
Users can press on the photo and a message that explains the situation would appear. Then they can press the “back” button to go back to the menu page.

A game screen like this would show up when the user presses the “Magenta” game button.

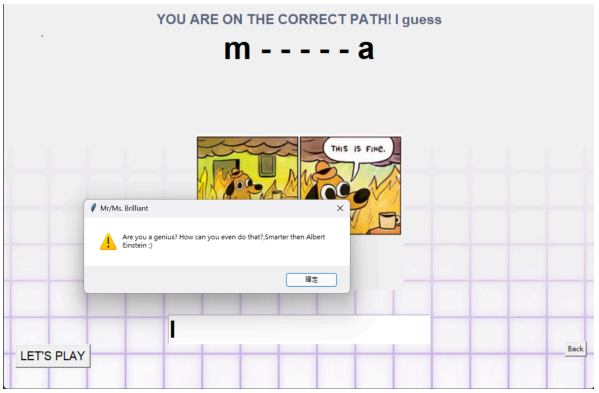
Users can type in a letter of their guess in the entry box and press the “LET’S PLAY” or Enter on keyboard to check if the guess is correct.



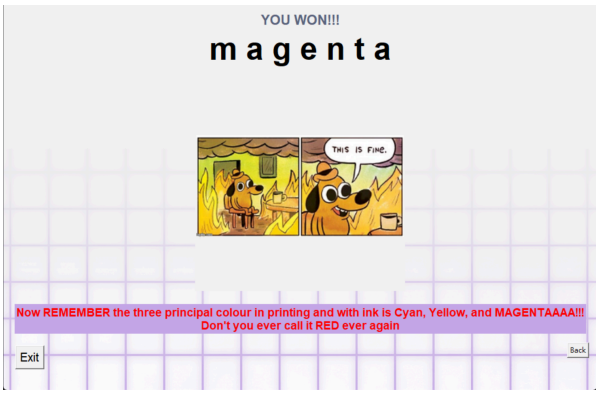
A warning box would appear when more than one letter is being typed in the entry box.



A random mockery would appear when the user guessed wrong. Users will loss a heart on the HP bar and the clue picture behind will change.

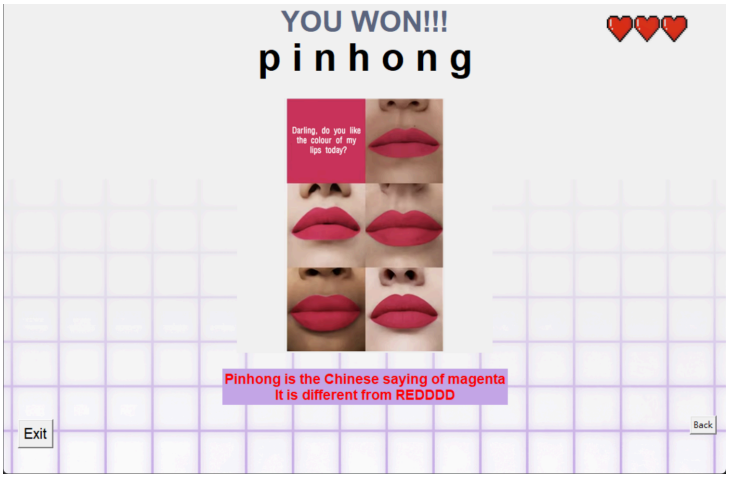


After the fourth wrong guess, no matter what the user types in, it will automatically change to one of the correct answer and a mockery box would appear.



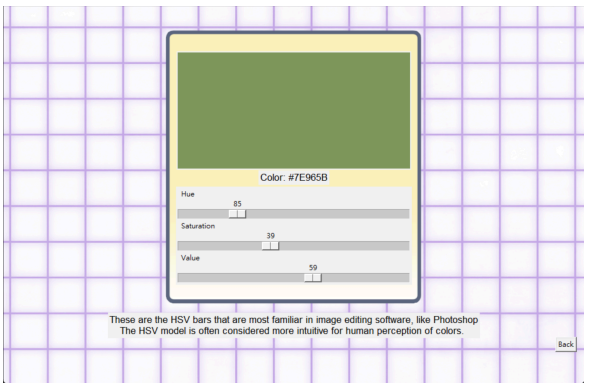
A message would be shown after the game.

If the user win the game without any wrong Guessing, it will leads to this page where a cat gif and a message shows.



If the user had pressed the secret button at the very first of the game (a dot on the top left of the game screen), the secret word would change to “Pinhong” and a different message would show when the user won the game.

This is supposed to be an ARG (Alternate Reality Game) design as I had left a clue hinting the secret button in my “image” file.



Users can experience the color mixing tool and the HSV tool and a message would be shown underneath the tool to explain the tool. Messages would be changed when pressed in the page of the color mixing tool.

Limitation:

I would love to present the difference between principal colors on digital media and reality printing media in a more relatable and practical way. However, the application is literally digital, which makes it only possible to explain the difference theoretically by words. If it is possible to hand in something that is non-digital, I would love to attach transparent color paper to let my user experience subtractive color, and attach color light bulbs for additive colors.

Reference:

1) Inputting the gif animation (line 268 - 288), referenced from chatgpt

```
import tkinter as tk
from PIL import Image, ImageTk

def update_gif(frame):
   global gif_frames, gif_index, gif_after_id

   canvas.itemconfig(gif_display, image=gif_frames[gif_index])
   gif_index = (gif_index + 1) % len(gif_frames)

   gif_after_id = canvas.after(100, update_gif, frame + 1)

# Create the Tkinter window
root = tk.Tk()
root.title("Display Animated GIF")

# Create a Canvas widget
canvas = tk.Canvas(root, width=300, height=300, bg="white")
canvas.pack()

# Load frames of the animated GIF
gif_frames = []
gif = Image.open("animated.gif")
for frame in ImageSequence.Iterator(gif):
   gif_frames.append(ImageTk.PhotoImage(frame))

# Display the first frame initially
gif_index = 0
gif_display = canvas.create_image(150, 150, anchor=tk.CENTER,
image=gif_frames[gif_index])

# Start the animation loop
update_gif(0)

# Start the Tkinter event loop
root.mainloop()
```

2) The function to change incorrect guessing to revealing random correct letter (line 326-342), referenced from chatgpt

```
# Add this part to replace the incorrect letter with a random letter
          incorrect_letter = guessed_letter.get()
          correct_letter_indices = [i for i, letter in
enumerate(secret_word) if letter == incorrect_letter]
          if correct_letter_indices:
              random_index = random.choice(correct_letter_indices)
              display_word[random_index] = secret_word[random_index]
              display.configure(text=display_word)
```

```
            # Update the display_word for the last trial
            if trial == 4:
                display_word = list(secret_word)
                display.configure(text=display_word)
```

3) The color mixing tool (line 520-584), referenced from chatgpt

```
import tkinter as tk

def update_color():
   # Get the RGB values from sliders
   red_value = red_slider.get()
   green_value = green_slider.get()
   blue_value = blue_slider.get()

   # Create a hexadecimal color string
   color = f"#{red_value:02X}{green_value:02X}{blue_value:02X}"

   # Update the color on the canvas
   color_canvas.config(bg=color)

   # Update the color label
   color_label.config(text=f"Color: {color}")

# Create the Tkinter window
root = tk.Tk()
root.title("Color Mixer")

# Create sliders for Red, Green, and Blue components
red_slider = tk.Scale(root, from_=0, to=255, orient=tk.HORIZONTAL,
label="Red", command=update_color)
green_slider = tk.Scale(root, from_=0, to=255, orient=tk.HORIZONTAL,
label="Green", command=update_color)
blue_slider = tk.Scale(root, from_=0, to=255, orient=tk.HORIZONTAL,
label="Blue", command=update_color)

# Create a canvas to display the mixed color
color_canvas = tk.Canvas(root, width=200, height=100, bg="#000000")

# Create a label to display the hexadecimal color code
color_label = tk.Label(root, text="Color: #000000", font=("Helvetica", 12))

# Pack widgets
red_slider.pack(pady=10)
green_slider.pack(pady=10)
blue_slider.pack(pady=10)
color_canvas.pack(pady=10)
color_label.pack(pady=10)

# Initialize color
update_color()

# Start the Tkinter event loop
root.mainloop()
```

4) TheHSV tool (line 604-686), referenced from chatgpt

```
import tkinter as tk
import colorsys

def update_color():
   # Get the HSV values from the sliders
   hue = hue_slider.get() / 360.0  # Convert hue to a value between 0 and 1
   saturation = saturation_slider.get() / 100.0
   value = value_slider.get() / 100.0

   # Convert HSV to RGB
   red, green, blue = colorsys.hsv_to_rgb(hue, saturation, value)
   red = int(red * 255)
   green = int(green * 255)
   blue = int(blue * 255)

   # Create a hexadecimal color code and update the canvas background
   color_code = f'#{red:02X}{green:02X}{blue:02X}'
   color_canvas.config(bg=color_code)
   color_label.config(text=f'Color: {color_code}')

# Create the main window
root = tk.Tk()
root.title("HSV Color Picker")

# Initialize HSV values
hue_value = tk.IntVar(value=0)  # Initial hue (0-360)
saturation_value = tk.IntVar(value=100)  # Initial saturation (0-100)
value_value = tk.IntVar(value=100)  # Initial value (brightness) (0-100)

# Create sliders for each HSV component
hue_slider = tk.Scale(root, label="Hue", from_=0, to=360, variable=hue_value,
orient="horizontal", command=update_color)
saturation_slider = tk.Scale(root, label="Saturation", from_=0, to=100,
variable=saturation_value, orient="horizontal", command=update_color)
value_slider = tk.Scale(root, label="Value", from_=0, to=100,
variable=value_value, orient="horizontal", command=update_color)

# Create a canvas to display the selected color
color_canvas = tk.Canvas(root, width=200, height=100)
color_canvas.pack()

# Create a label to display the color code
color_label = tk.Label(root, text="Color: #000000")
color_label.pack()

# Place the sliders on the window
hue_slider.pack()
saturation_slider.pack()
value_slider.pack()

# Initial color update
update_color()
```

```
# Start the Tkinter event loop
root.mainloop()
```