To guarantee that "The Color Maker" performs as anticipated, provides accurate and pertinent feedback, and offers a positive user experience, validation testing is crucial. The following are some essential validation tests that need to be run:

Test of color matching

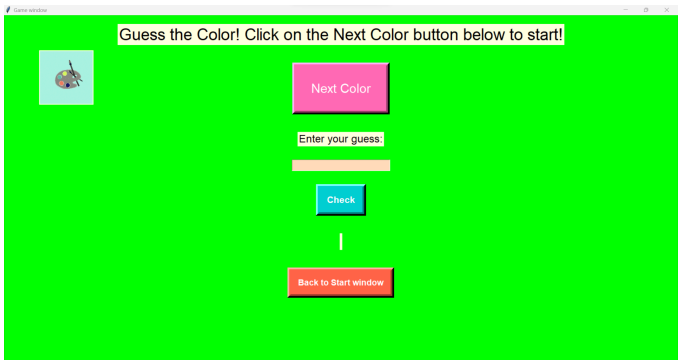
Verify whether the game correctly identifies the color the player guessed.

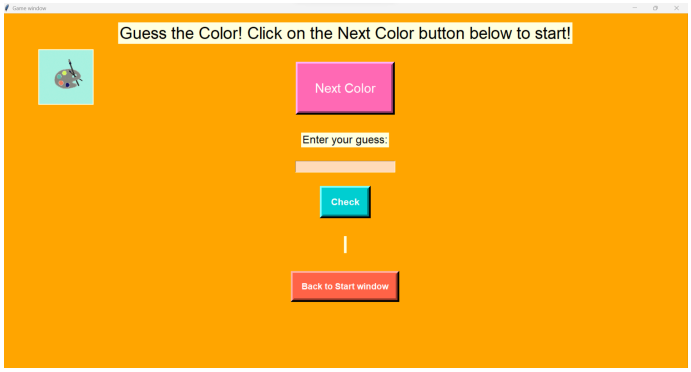
Procedure: Try inputting both the proper and erroneous color names, then watch the game's feedback.

Expected Result: When a player's guess is accurate, the game should accurately identify the color and display an appropriate message for both correct and incorrect guesses.

Result: The program correctly identifies the color the player guessed. It displays whether the player is correct or not.

Randomized color test
Verify that each time the "Next Color" button is pressed, a new random color is displayed in the game.
Follow this procedure to see if the color changes after each click on the "Next Color" button. The game should display several colors at random to provide players with a varied and engaging experience.

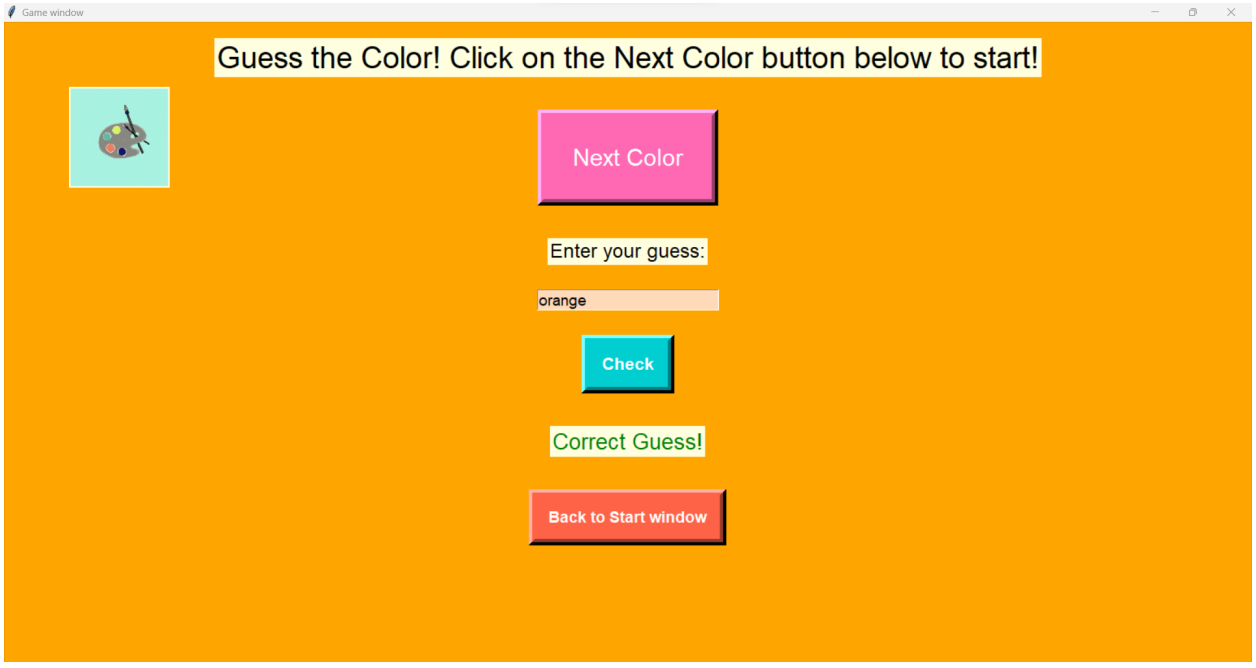




Result: The program works correctly and changes the background color to a random color each time the "Next Color" button is pressed.

User Interface Evaluation:
To evaluate the user interface's usability and clarity.

Expected Result: The user interface needs to be simple to use, with concise guidance and responsive buttons.



Result: passes the test, buttons work correctly and the program is also smooth. I played the game to make sure.

Input validation Test

Verify that the game offers suitable feedback and correctly handles user input.

Procedure: Fill in the name entry box and color guess field using a variety of input, including integers and special characters.
The game should check user input and elegantly handle erroneous or inaccurate input without crashing.
Result: Works as intended, passes the test. Please check the screenshots below. Works with integers and special characters for the name too and capitalization of color name.

I ran into the following problems during the testing process and fixed them:

An issue with Image Visibility:
Problem: Window 2's image was obscured.
Fix: To fix this problem, I loaded the appropriate GIF file and used a Label widget to display it in the window.

Resize the image:
Problem: Window 2's GIF image was too big.
Fix: To make the GIF picture on window2 smaller and better suit the window, I shrunk it using PhotoImage's subsample function.

Text Replacements for Images:
The Python environment used for the work does not permit installing extra packages, such as the PIL package needed for image processing, hence the user's request to add alternate text to the images was not able to be fulfilled.

Everything else worked out pretty well, the above issues were resolved except the Text replacements.