**RGB Matcher**

In this project, you'll create a game that challenges the user to match a randomly generated color using RGB color sliders. First, a bit of background on color (if necessary):

*Computer screens generally use an RGB (****Red****,* ***Green****,* ***Blue****)* ***additive color model****, which stores values for the red, green, and blue components for each pixel, each value ranging from either 0 to 255 (an 8-bit integer) or 0.0 to 1.0 (a floating-point value).*

*For example, you can make yellow by combining red and green (shown right). That probably sounds strange, but combining pixels isn't the same as mixing paint to make a color. The computer uses light to display color, not paint.*

*The RGB color model sometimes also stores an* ***alpha value*** *as well as the RGB values. The alpha value indicates how transparent or opaque the color is. A color that is transparent will let you see some of the color beneath it.*

**Requirements:**

1. [Here](https://www.loom.com/share/c8a10d2a17184888a03c8a1a04e13937) is how this app should work.
2. Two fields with black borders near the top for displaying the "target" color and the user's current generated color (controlled with the sliders).
3. Appropriate labels directly below the fields you made in the previous step.
4. Three sliders valued 0-1, for red/green/blue. When the user moves the sliders, it should update the current color field.
5. A timer that limits the user to a set amount of time to get the color as close as possible.
   1. Time remaining should be shown at the top.
6. An alert at the end of a game that shows the player their score:
   1. To calculate score, find the "linear distance" from the randomly generated color to the user-selected color.
      1. Use the distance formula to calculate the "linear distance" between the points in 3D space: diff = sqrt(rDiff2 + gDiff2 + bDiff2).
      2. To convert the above to a score, subtract the value you calculated from 1 then multiply by 100. Round to the nearest integer value.
7. Another round of the game should start after the user closes the score alert.

**Extension Ideas**

* Persistently store the fastest time and/or closest to the color.
* Make the game work in either portrait or landscape (and adapt to any display size).
* Add modular support for other color models (e.g. HSB or CMKY).