Frontend Developer Interview Code Challenge

Requirements

- Please use React to build this web application.
- TypeScript is preferred.

Project overview

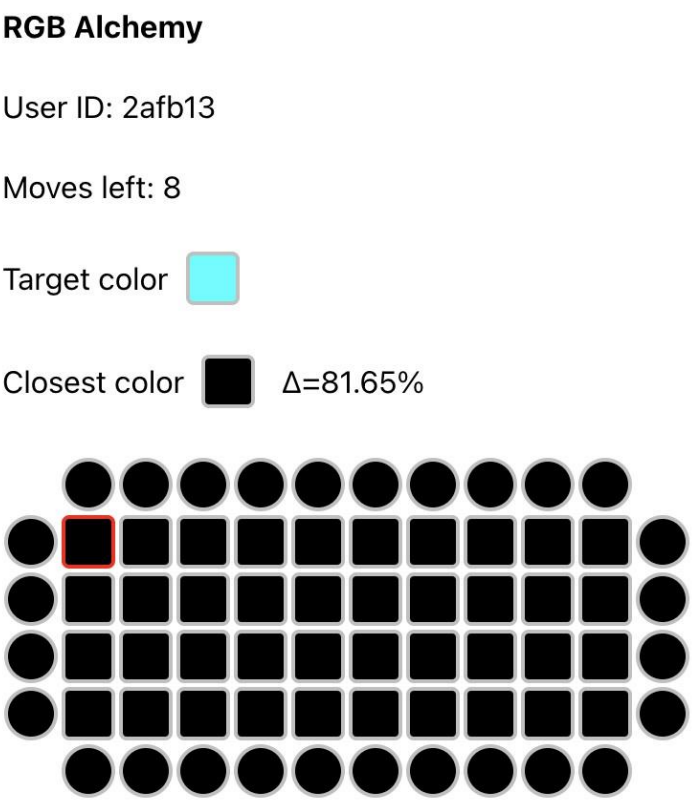


Figure 1: Basic UI

RGB Alchemy is a small game application in which the player can try to mix colors to create new colors. The game goal is to get a color which “matches” the target color in limited moves.

Preliminary concepts

- We use CSS syntax `rgb(red, green, blue)` to describe the color. `red` , `green` and `blue` are numbers between 0 and 255.

- The difference of target color ($rgb(r_1,g_1,b_1)$) and obtained color ($rgb(r_2,g_2,b_2)$) is defined as

$$= \frac{1}{255} \times \frac{1}{\sqrt{3}} \times \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}$$

  If it is less than 10%, the player wins the game.

UI components

- Information lines: game name, user ID, moves left, target color, the closest color the player has obtained, and the color difference Δ between the closest color and the target (see Figure 1).

- 2-dimensional array of squares (“tile”) surrounded by circles (“source”). The size of the array is $w \times h$. In the Figure 1, $w = 10$ and $h = 4$. The initial colors of all tiles and sources are black (see Figure 1).

- Additional details on tiles/sources:

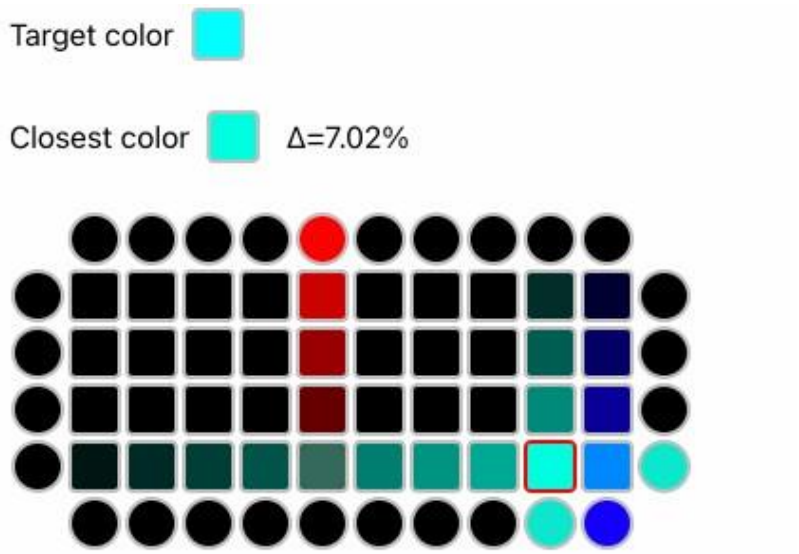


Figure 2: tooltip and highlighting of tiles

  . Each tile has a tooltip showing its color, including the tiles in the information lines, i.e., “target color” and “closest color” (see Figure 2).

    . The closest one in the array of tiles is highlighted with red border (see Figure 2).

  . `cursor: pointer` only applies to the tiles/sources that are clickable or draggable.

- A popup dialog allows the player to restart the game. You can use `window.confirm()` or dialog components provided by any UI libraries.

User interactions

- In the first 3 moves, the player has to click sources. The clicked source will be painted as red/green/blue, i.e., `rgb(255, 0, 0)` / `rgb(0, 255, 0)` / `rgb(0, 0, 255)` . The tiles in the same row or column will be painted as well. If the distance between the tile and the sourece is d, the color of the tile depends on the "relative distance" from the colored source, $(w + 1 - d)/(w + 1)$ or $(h + 1 - d)/(h + 1)$. For example, in Figure 3, the 4 red tiles from the top to bottom have colors: `rgb(255*4/5, 0, 0)` , `rgb(255*3/5, 0, 0)` , `rgb(255*2/5, 0, 0)` , `rgb(255*1/5, 0, 0)` .

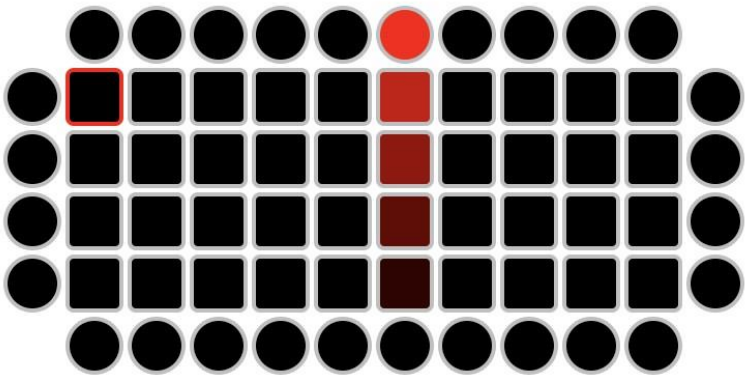


Figure 3: One colored source

- If the tile is "shined" by multiple sources, the resulting color can be calculated by the following equations

$$r = r_1 + r_2 + r_3 + r_4 \; g = g_1$$

$$+ g_2 + g_3 + g_4 \; b = b_1 + b_2$$

$$+ b_3 + b_4$$

$$f = 255/\max(r,g,b,255)$$

$$\text{Result} = \text{rgb}(r \times f, g \times f, b \times f)$$

Where $\text{rgb}(r_i,g_i,b_i), i = 1,2,3,4$ are the contributions from each source. f is a normalization factor to make sure that all RGB elements of the resulting color does not exceed 255. In Figure 4, the color of highlighted tile can be calculated as the sum of the contributions of three sources: `rgb(255*4/11, 0, 0)` , `rgb(0, 255*7/11, 0)` and `rgb(0, 0, 255*3/5)` . The result is `rgb(93, 162, 153)` .

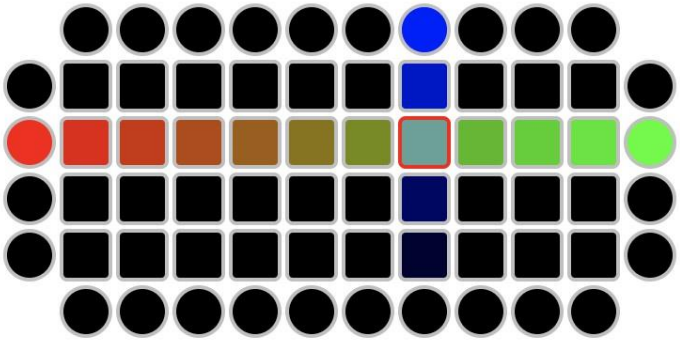


Figure 4: Multiple colored sources

- After the first 3 steps, the player cannot click sources any more, but they can drag any tile onto any source, including the black tile, to replace the source color with the tile color. By this way, the player can create more colors in the array of tiles.

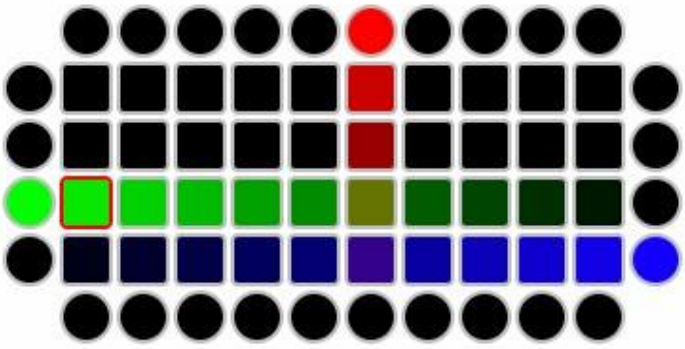


Figure 5: Dragging the tile to the source

- Once the target is reached, or there is 0 step left, a pop up window is displayed to notify the player the result (success or failure) and ask them if they want to play the game again. If the player chooses to “play the game again”, the app will fetch the data from the server and initialize the UI, but keep the user ID unchanged.

Development environment

- We provides a simple Node.js program to start a local server listening at port 9876. You can start the server by running the following command

```
npm install
npm start
```

This program was tested in Node version 14.17.3.

API docs

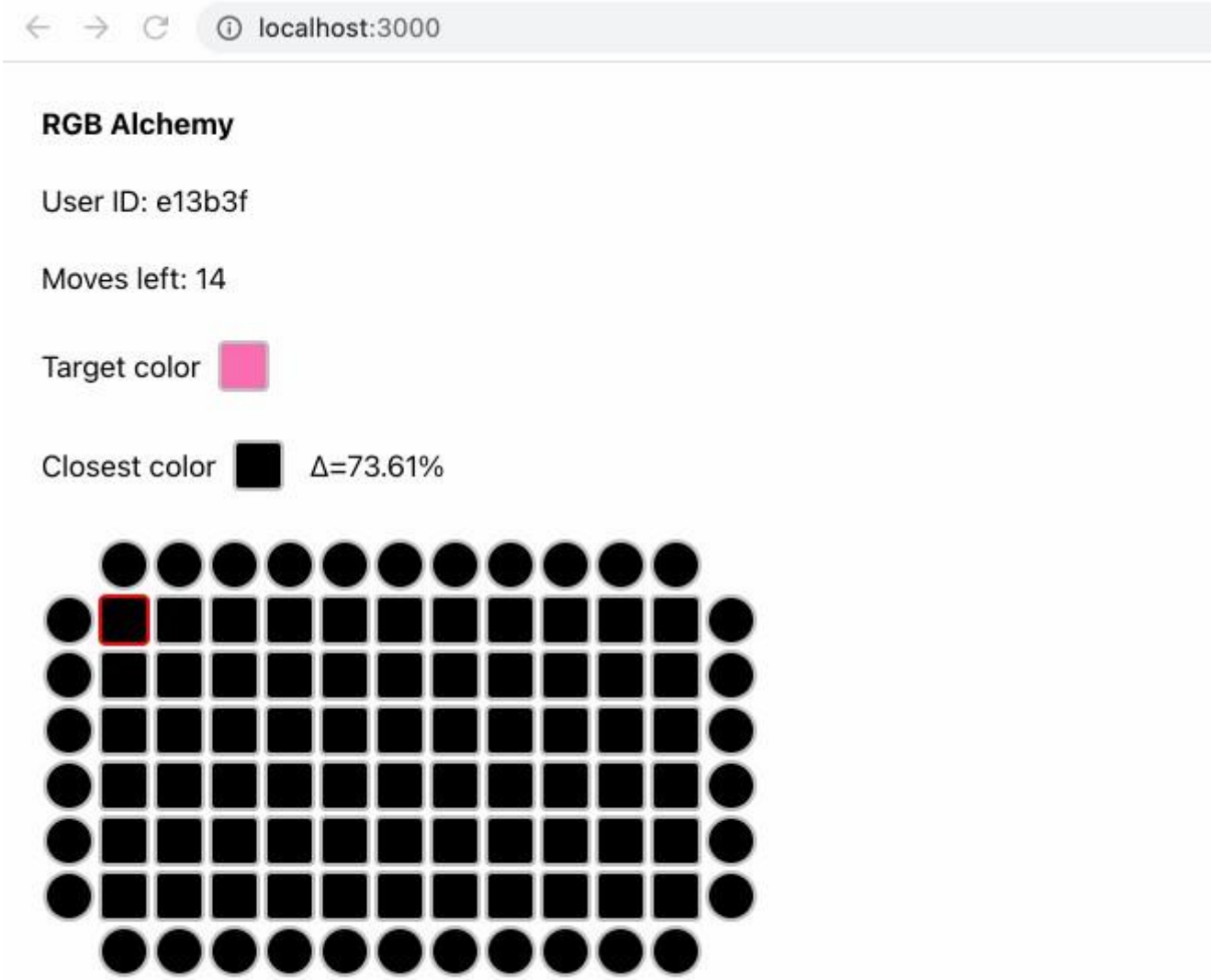
- The web application fetches the basic information from the server to initialize the game.

```
// Request
curl http://localhost:9876/init

// Response
{
    "userId":"774b31",      // Random string
    "width":10,             // Random integer min=10 max=20
    "height":4,             // Random integer min=4 max=10
"maxMoves":8,           // Random integer min=8 max=20
    "target":[0,255,255]    // Random RGB Color
}
```

- If the user chooses to play the game again, the current user ID should be used in the request.

```
// Request curl http://localhost:9876/init/user/<my-
user-id>

// Respons[](https://)e
{"userId":"<my-user-id>","width":10,"height":4,"maxMoves":10,"target":[0,255,255]}
```

Demo GIF


localhost:3000
RGB Alchemy
User ID: e13b3f
Moves left: 14
Target color
Closest color
Δ=73.61%