## Goals

\* Practice using `Services` in angular

\* Practice implementing stateful services using `BehaviorSubject`

\* Practice displaying the observable values using `| async`

\* Practice using the various rxjs operators, especially:

- `map`

- `combineLatest`

\* Continue practicing angular components

## Overview

In this exercise we will write a game that lets the user practice his \*\*color composing skills\*\*. The application will present a color and allow the user to try and guess the RGB components of the color.

Shape, square

Description automatically generated

- The computer generates a random color. The user may ask for a new color by clicking the \*\*Random Color\*\* button.

- The user enters 3 numbers : Red, Green, and Blue components between 0 and 255

- The computer draws the color that matches the components that the user enters

- If the color of the user matches exactly to the color the compoter generated, a \*\*Success\*\* message appears on the screen

## Step 1 - Service

\* Create a `GameService` object that exposes the following methods

- GetRed(): Observable<number>

- GetGreen(): Observable<number>

- GetBlue(): Observable<number>

- SetRed(value: number): void

- SetGreen(value: number): void

- SetBlue(value: number): void

- GetComputerColor(): Observable<[number, number, number]>

- RandomizeColor(): void

## Step 2 - UI

\* Implement the UI that allows the user to play the game

- This time I will leave it to you to decide how many components you write and the relationship between them but please follow the \*\*SOLID\*\* principles and make sure that none of the components "know too much".

HINTS:

1. The operator `combineLatest` should interest you

2. Components may consume the service observables but they may want to present other data, in another form. They should use the service observables and apply operators to create the observables that best describe the data they want to present

3. The `Success` label should also be controled by an observable.