Design Doc

CV2 Color Type + Chips

Currently, colors are set in CV2 using integer representations that are very opaque and require memorization to know which color corresponds to which integer.

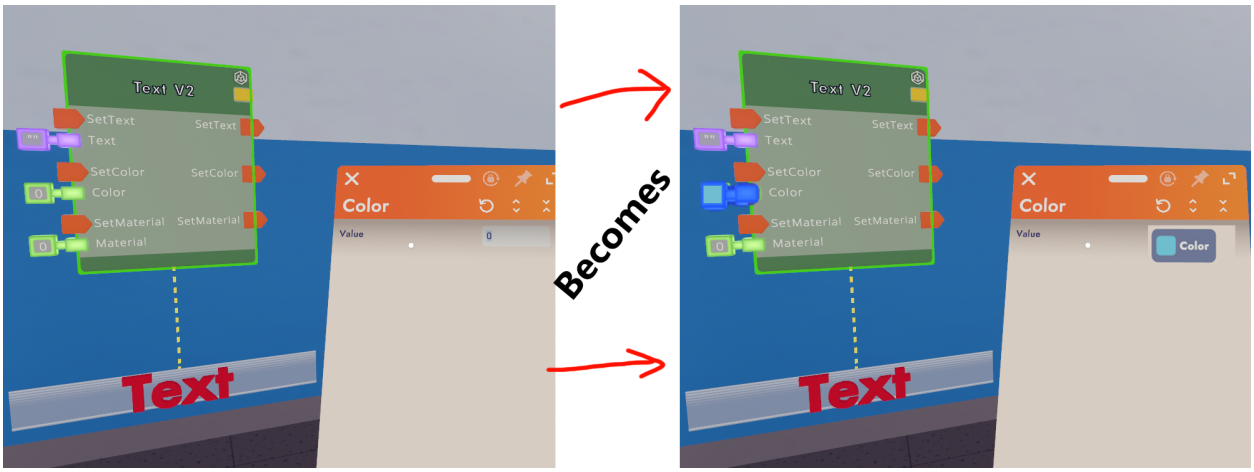
Goals

1. Deliver a Color Type that no longer requires creators to rely on memorization of arbitrary integer values
2. Color picking in CV2 is a simple UX experience with clear visual feedback.
3. Audit existing gadgets and CV2 nodes that use integers for color to accept Color Type without breaking existing rooms
4. Give creators greater flexibility for real-time expression with color in their rooms where possible
5. Push creators towards Rec Room style first, but lay groundwork that does not create debt if we give them greater power in the future.

Design

The Basic User Experience

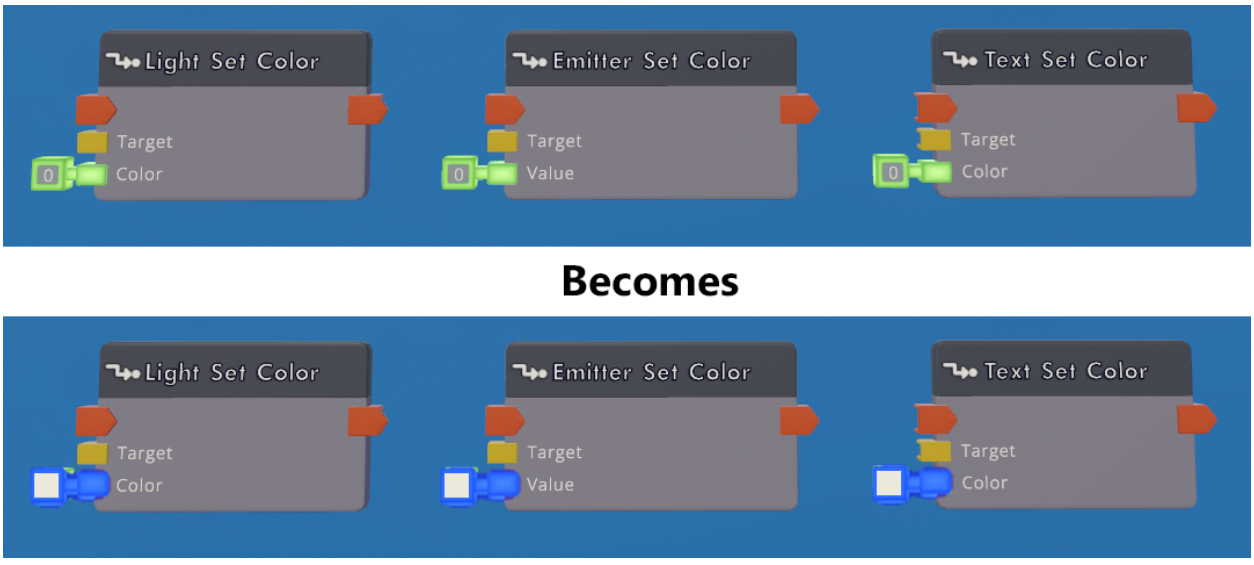
Currently colors are represented to the user as an integer. They should be represented to the user as a color.



One that's really easy to pick. We already have a color picker, port it here.

So gadgets like *Text V2* and *Point Light V2* change their input type from **Int** to **Color**

We do this by auditing the existing Set Color chips and creating new versions that accept the color type and replacing the integer versions in the palette and in the control panels of Lights, Text V2, and Emitter V2



! To avoid breaking existing rooms, we will keep the integer version in the game, rename them i.e *Light Set Color (Int)*, and remove them from the palette.

MVP

The MVP implementation of a color type is really just a UX change in how we present the `ShapeColor enum` from **Shape.cs** to the user. With this implementation we can enable creators to use colors in CV2 without needing to know arbitrary integer to color values.

Color Chips

A few new chips so creators can have fun with colors in CV2

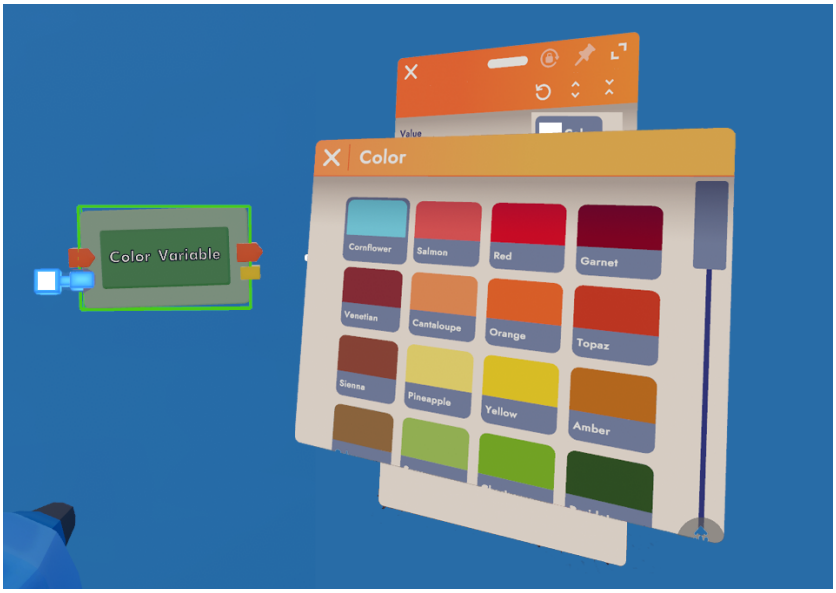
Color Variable

A building block that holds color information and can be changed at runtime.

- Input:
  - Exec
  - Color
- Output:
  - Exec
  - Color

Notes:

- The color of the Color Type needs a sign off from art. Dark blue from UE4 or a dark purple seem like good options.



List<Color> Variable

A building block that holds color information and can be changed at runtime.

- Input:
    - Exec
    - List<Color>
- Output:
    - Exec
    - List<Color>

Notes:

- This is required to give creators an alternative to integer addition in the cases where they want to cycle through a lot of colors on an object. An example is using *List Create* to have a reference to many colors, then using *Random From List* or doing arithmetic with the index.

FadeLightColor

A node to change between two colors over a period of time.

- Input:
    - Run <Exec>
    - Reverse <Exec>
    - Target <Light>
    - From <Color>
    - To <Color>
    - Time <Float>
- Output:
    - Run <Exec>
    - Reverse <Exec>
    - Finished <Exec>

Notes:

- Color for lights is actually fed as RGB to Unity's Light game objects under the hood so lerping between Vector values is totally doable even if we do not necessarily give players access to every possible RGB color to feed into the chip

Recolor Prop

Let's have fun with the color type and use it to recolor things dynamically in CV2!

This would be applicable to any object that has the SandboxColorable component (recolorable props) and in the future be able to support recoloring shape containers/OM groups.

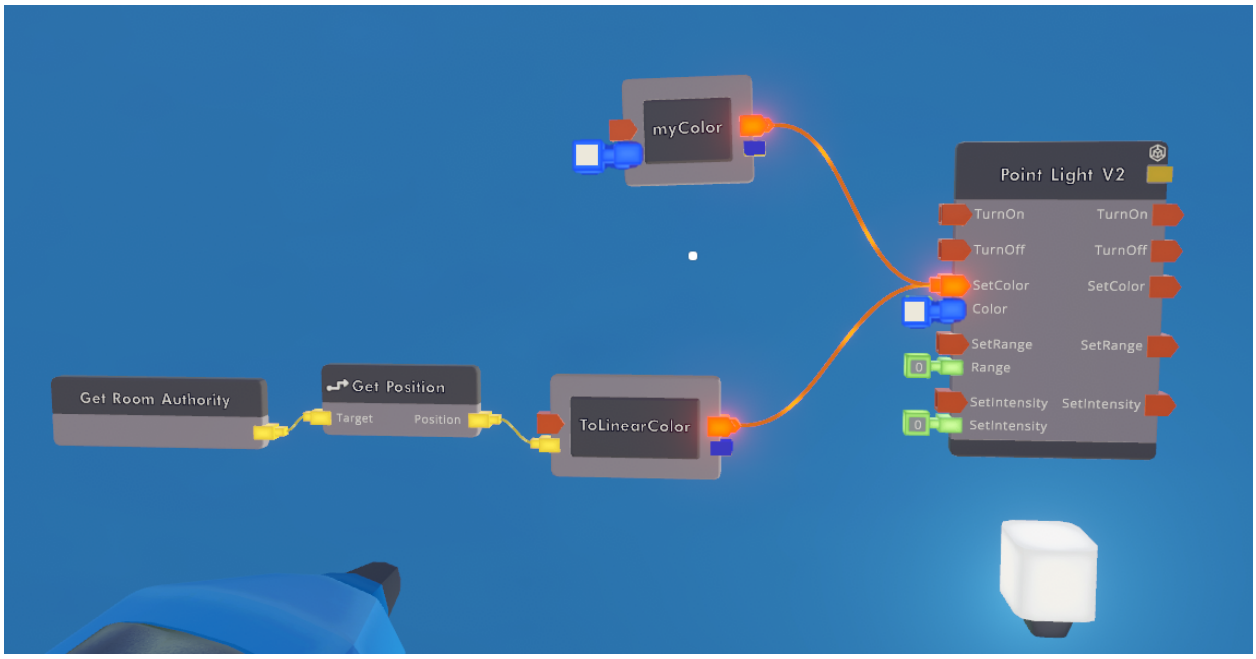
- Input:
  - Exec
  - Target <RecRoomObject>
  - Color <Color>
- Output:
  - Exec

Notes:

- The recolored state DOES NOT SAVE. Creators should be able to reset the room to restore to the objects' saved color.
- If an object can't be recolored, send a warning to the console.
- According to [REDACTED], recoloring shapes would require new tech to add a color on top of the shapes' default color so the meshes do not need to be recreated in real-time. It is solvable, but OM renderer work is blocking this work at the time of writing.
- "Recolor Object" would be a rebrand for the same chip in the future.

Future-Proofing

I predict we will give our creators access to more colors in the near future. If we continue to gate it behind N colors identified in the `ShapeColor enum`, then this system will work indefinitely. However, if there is a future in which we give creators full RGB, then the implementation of this color type should not block that freedom. There is a ton of power we could unlock for creators if they were enabled to do Vector arithmetic for colors and that is a future I believe in, I just don't necessarily think this is the time to set the precedent. **Let's just not create future debt now but rather set a tech foundation for this to be an inexpensive pivot later.**



We shouldn’t block a future where these are both valid inputs