**Wall Colouring Prototype**

The aim of this prototype is to answer the technical question of whether we will be able to implement a way for the end users to virtually change the paint colour on the walls of their space. It will also serve as research into the best methods for doing so.

**Similar software researched**

Dulux visualizer – an app distributed by Dulux which allows users to virtually test out their range of paint by applying it to their walls in an augmented reality environment.

The core functionality is strong; if we could manage similar results with some minor tweaks it would be ideal.

Drawbacks:

* Inability to paint connecting walls separate colours.
* No differentiation between ceiling and walls.
* Objects too similar in colour to the walls get misinterpreted as being part of the wall and change colour along with it.

Thoughts on their implementation:

* The software must utilise colour detection and then recolour anything within a certain “colour distance” to match the desired colour. This leads to both drawbacks, but allows for a very intuitive user-experience of simply tapping where you want to paint and it filling in the rest.
* While the errors in detection are frustrating, when working it manages to look very smooth. I believe it isn’t going through pixel by pixel and recolouring each one.

**Initial prototype**

The first attempt at prototyping this was done on Processing 3 using the Ketai [reference] library to access the android camera. The way it works is it allows the user to tap an item on the live camera feed, which the software then draws the RGB values from by selecting the specific pixel. Once it has a “colourGrabbed” it loops through all other pixels currently in the camera feed and if they are similar it will reassign them to red. “Similar” in this case is adjustable.

Wall colouring functional prototype v1

As displayed in the image above, the software at a basic level is beginning to take shape. It manages to detect similar colours (though also picks up unintended objects too close in colour), while avoiding objects which are clearly different. Unfortunately inconsistent lighting is heavily affecting the outcome, with some sections of the wall classed as “too different” despite being the same colour. The performance when calculating colour distance on each pixel also starts to become an issue. Reducing calculations by grouping pixels together may be a solution to that.

**Thoughts going forwards**

We need to reduce the number of calculations performed; more research into image segmentation is required. Consider having more user interaction to aid the software, e.g. have the user draw a rough outline of where the wall should be filled in, and then the software need only detect windows and objects that aren’t part of the wall.

We need to conduct more research into “colour difference” and ways of dealing with several light sources impacting colours.

MORE STUFF. INCOMPLETE.

TA:L ABOUT HSB/ HSL vs RGB. I think HSL is better for colour matching.

In processing at least though you can only return RGB value with get()