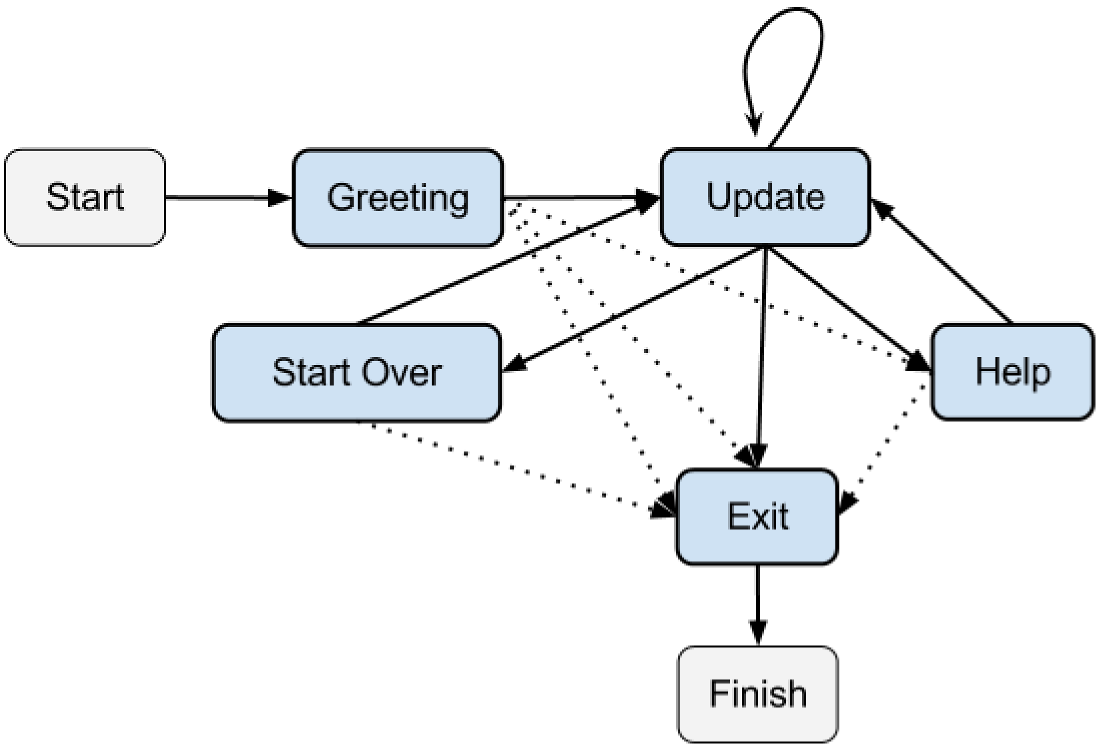
Overview of SDS for Talking Colors

Erik Ackermann
Richard Boyle
Kate Hansen

Domain Functionality

Our SDS system is a simple voice-controlled color-picker. The system displays a color window and allows the user to update the color's hue, saturation, and brightness. Additionally, the system can return the color's RGB or hexadecimal values. When the systems starts it begins in the Greeting state, which prompts the user to give an initial color description. Then the system enters the Update Color state, and stays in this state while the user keeps giving directions. In the Update Color state, the user can change any of the color's attributes, or transition to the Exit or Help states. Whenever the user gives a command in this state the system always replies with some sort of acknowledgement or confirmation, such as "Look good?" or "How is this?". At any point the user can say "Help" or "Exit," which will take the system into either the Help or Exit states. The Help state tells the user which color attributes he/she can change, as well as a list of commands. The Exit state prompts the user for an output format (RGB or Hexadecimal), and then returns the value of the color in this output format. Then the system asks the user if he/she would like to continue, at which point the system either terminates or re-enters the previous state.

System State Diagram



Changes Made to ASR and TTS

Since our domain became much larger, we had to re-record and expand the prompts from our original TTS and modify the grammar from our original ASR. Additionally, we added the visual component for user feedback. To handle all of the available commands, we had to create several functions for changing the attributes of a given color. We created an object called ColorState, which contains all of the information for the user's color. We keep a stack of these ColorState objects, which allows us to support undo. The colorutil.py includes methods for changing any of the attributes of a ColorState object. Additionally, the ColorState object contains definitions for the ten base colors in our system. These colors are: Red, Orange, Yellow, Green, Blue, Purple, Pink, Black, White, and Grey. The color definitions are stored in a dictionary that maps the color names (as strings) to color attributes (tuples of double values).

To handle variations of degrees for each attribute (e.g. *a lot more* orange vs. *a little less orange*) we mapped each degree modifier to an integer value. These mapped values are then used to determine the size of the change associated with the given attribute.

In the case of color mixing (e.g. "Make it more reddish"), the current color is blended with the target color. This is a three step process. First, the brightness of the current color is saved (so that this process doesn't alter *V* (brightness) and only alters hue). Then, the target color is determined from the dictionary of preset color values. This color's brightness is then shifted to the stored brightness from step one. Finally, the current color is averaged (component-wise) with the target color. The average is weighted by the degree modifier (if specified). If no degree modifier is given by the user, then the degree is assumed to be "a little".

The degree mappings are:

- "a lot" = 25
- "'much" = 16
- "a little" = 8
- "a tiny bit" = 4

The weighting function is

```
new R = current R + weight *(current R + target R)/2.0
```

where weight is defined as

```
weight = degree/35.0
```

and degree is the integer value defined above. We chose experimentally determined 35 as the best value for the weight divisor.

USAGE

To run Talking Colors, use the command "./color_picker.py". This will initialize the program; further prompts are explained by the program.

The ASR files are all contained in TalkingColors/ASR/, and the TTS files are contained in TalkingColors/TTS/. For each session of Talking Colors, a log directory is created in TalkingColors/logs/. These directories are labeled with the current time at the beginning of the session. Inside the directory are .wav files from every user input and system output.

AVAILABLE COMMANDS

Commands available to the user are selecting a starting color, adjusting the hue, saturation or brightness, or system commands for help, undo, output format, exit and start over.

Examples

"I would like a dark orange."
The user selects a starting color of dark orange. The system allows for a number of colors and they can give it an initial brightness, saturation or hue. This example gives it a initial brightness of dark.

"Can you make it a lot brighter?"
The user changes the brightness of the current color. They also have the added option of choosing how bright, (a lot, less, more, etc).

"Can you make it a little more red?"
The user asks the color picker to change the hue of the current color. The system allows the command to make the current color more or less of any of the other available colors.

"Can you make it less saturated?"
The user asks the system to change the current color's saturation. Again, like brightness, they have the option of choosing the degree of saturation or desaturation.

"Help me please."
The user can ask the system for help, and the system will give the user the available options.

"Can you make it RGB?"
The user selects the output format of RGB. They also have the available selection of hexadecimal output format.

"I'm done."
The user tells the system they are finished. The user can also say, "Exit" and the system will give the user the color in the desired format.

"Can I start over?"
The user tells the system to start over and the system will go back to the initial Greeting State. This user can then choose another starting color.

Tradeoffs

To improve the functionality and recognition of our system, we had to limit the complexity of our grammar. The user can only change one attribute at a time (e.g. “Make it more saturated” is accepted but “Make it brighter and more saturated” is not accepted). However, we decided this was an acceptable tradeoff to make because we wanted to user to see a visual feedback for each attribute change. This way, the user can undo the appropriate command. Additionally, we limit the number of colors the user can start from (mentioned above). Since users are most familiar with these common colors, and because the user can get to any color by specifying attributes, this is not a large limitation.