FINGER ART

Sean Liu
Grace Li
October 31, 2008

PROPOSAL

Introduction

In our project, we will simulate the act of painting. The user will wear a special glove with LEDs, which will enable a camera to track the motion and gestures of the user's hand. As the user "paints", the system will calculate the location, velocity, and angle of the hands, which translate into variations in virtual brushes. Additionally, the system also incorporates painting effects such as color mixing, paint splatter, and brush saturation. The goal is to simulate a real painting experience.

Our project is composed of two parts: the video interface and the paint canvas synthesis. The video interface takes movements from the user's hands and identifies the position of the virtual paintbrush, the velocity with which it is moving, and the orientation or angle of the user's hands. These three pieces of information are then passed to the paint canvas synthesis. The paint canvas synthesis system takes in the position, speed, and orientation data and selects a brush, color, saturation, and surrounding splatter around the brush. These selections are translated into pixel color assignments, which add to what the user has already painted.

Functionality

In order to understand the how the painting occurs; one must first understand how we model the paint, palette, and paintbrush. The paint is represented using three 5-bit binary numbers to encode RGB color, and an additional 3-bit number to encode the saturation level. The screen is divided into two sections – the canvas and palette. If the system identifies that the user's brush is over the palette section, then paint is absorbed onto the brush. If the user's brush is over the canvas section, then paint is deposited on the canvas. The palette contains a set of primary colors, which the user can absorb at various saturations to mix and match paints. Finally, the user's hands represent a paintbrush, which will be the instrument for depositing the paint onto the canvas and picking it up from the palette.

The paintbrush is represented as a line of blobs, and the line can have very different orientations. Each blob has a core zone of high saturation surrounded by a zone of lower saturation, which represents the spreading out of the paint. Where the paintbrush touches the canvas, the core blobs of the brush deposit color on the canvas at the user hand location. When the paintbrush moves, each place that is rests upon will have paint deposited in the area of the lines of blobs. The makeup of the paintbrush from different kinds of blobs comes from the velocity of the

brush. The faster the paintbrush, the narrower and more randomized the blobs, and the slower the brush, the finer and more well defined the blobs.

There will be a finite amount of paint on the paintbrush, and as the paintbrush is moved, paint is deposited onto the canvas, and less paint remains on the paintbrush. This simulates the real effect observed in painting, where long strokes gradually deposit reduced amounts of paint. We represent this effect by gradually decreasing the saturation of the color deposited. Eventually, when the paintbrush has deposited all of its paint on the canvas, there will no longer be any paint on the paintbrush, and paint will no longer be deposited onto the canvas.

The palette will serve as the location where the user can pick up more paint. There will be sections for the three primary colors, sections for both black and white so that the user can make their chosen color both darker and lighter, and a section where the user can clean off all color from their paintbrush. How much of a certain color is picked up by the paintbrush and the saturation of that color are directly proportional to the time that the paintbrush touches the color.

System Usage

In this section, we will describe the different user interactions that are possible with the system and how the different user actions translate to painting. The user should be able to do the following things:

- Determine the position and velocity of the paintbrush
- Determine the orientation of their paintbrush
- Touch their paintbrush to the canvas to start painting
- Lift their paintbrush from the canvas to stop painting
- Pick up different amounts of paint from different colors on the palette
- Mix colors
- Clean their paintbrush

The user will wear a glove which has two LEDs attached. One will be attached to the index finger and the other will be attached to the thumb. The midpoint of the two LEDs will determine the center of the paintbrush. Using the movement of their hand, the user will be able to determine the velocity of the paintbrush by changing the position of their hand.

Orientation of the paintbrush will be determined by the relative positions of the two LEDs. The line between the two LEDs on the user's index finger and thumb is used as the orientation of the paintbrush. In order to touch the paintbrush to the canvas, the user will blink the LED twice and the system will know to start painting. Then, when the user again blinks the LED twice, the system will know that it should stop painting. To acquire paint on the brush, the user will position their index finger over the patch of paint on the palette and move their finger in a circular motion. Then, to clean the paintbrush off, the user will move their index finger in a circular motion over the cleaning area of the palette.

Module Definitions

Here we will discuss overviews of the modules in the system. Figure 1 gives a block diagram identifying the modules in the system. Conceptually we can divide the system into two main parts. The first part consists of the Video Capture, Gesture recognizer, and Intention modules. The second part consists of the Painter, Saturation, Color Generator, and Display modules. The Game Pixel Generator and Scorer are areas for potential expansion.

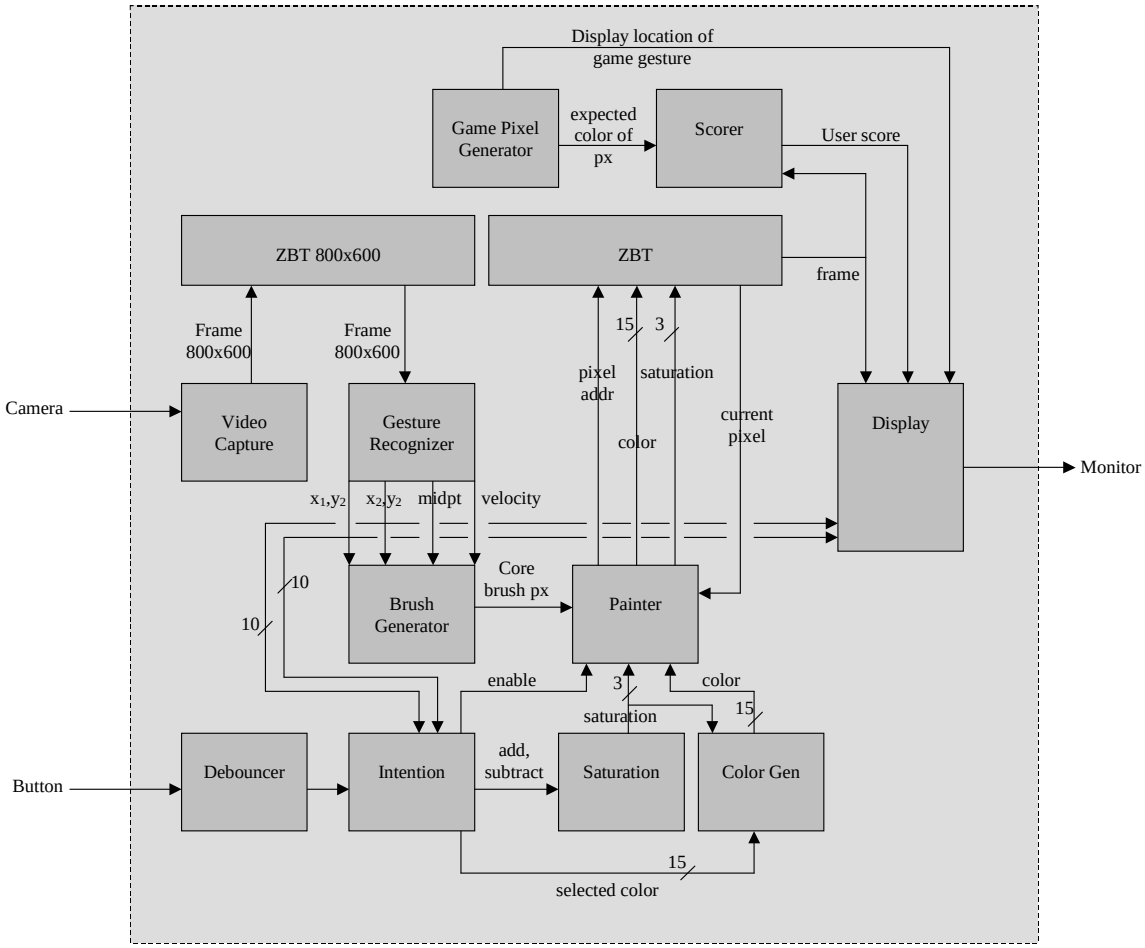


Figure 1: Modules of the Finger Paint system

Part 1: Video Capture, Gesture Recognition, and Intention Modules

The purpose of the video capture, gesture recognition, and intention modules is to identify the location, velocity, and orientation of the user's hands, which are then passed to the second part of the system for brush and paint rendering. Here we discuss each module in turn.

Video Capture

The video capture takes a frame every 1/60 of a second from the camera. It then filters the pixels, removing any pixels from the frame that are less than a given threshold. The threshold will be set in calibration when the device first powers on.

Gesture Recognizer

The purpose of the gesture recognizer is to take in a filtered frame captured from the video camera and identify the location, velocity, and orientation of the brush.

Location is calculated by identifying the midpoint of the two LEDs. This is performed by an nonzero-x and nonzero-y averaging calculation, where:

x_{avg} = sum(x_i)/(number of x)
y_{avg} = sum(y_i)/(number of y)

Velocity is actually a squared distance calculation. Here we remove the square root and division by time in order to save performance. Essentially, we can think of the velocity represented by the total distance traveled in the last ten screen captures. The last ten midpoints, and a current sum are stored every 1/60 seconds. For each new calculation, the new velocity is calculated by:

vel' = vel – squaredist(x[n-9],x[n-10]) + squaredist(x[n],x[n-1])

Orientation is represented by the location of the two LEDs. Averaging white pixels on either side of the overall average approximates the location of these two endpoints.

$x1_{avg}$ = sum(x_i left of x_{avg})/(number of x_i left of x_{avg})
$x2_{avg}$ = sum(x_i right of x_{avg})/(number of x_i right of x_{avg})
yA_{avg} = sum(y_i above y_{avg})/(number of y_i above y_{avg})
yB_{avg} = sum(y_i below y_{avg})/(number of y_i below y_{avg})

The endpoint combination of x and y is determined by whether or not a white pixel exists at location: ($x1_{avg}$, yA_{avg}) or ($x2_{avg}$, yB_{avg}).

Intention

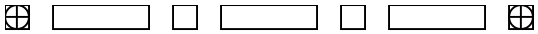
The idea of the intention is to identify if the user is over the color palette or over the canvas, and whether the user is attempting to pick up paint or deposit paint. While the end goal is to recognize a circular motion of absorbing paints, and flashing LEDs to toggle between painting and moving, for now we will simply use a button which toggles whether or not the user is moving the brush or actually painting/absorbing paint.

Part 2: Painter, Saturation, Color Generator, and Display Modules

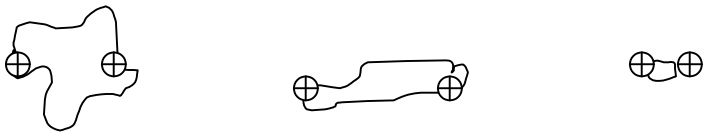
Brush Generator

The brush generator takes as inputs the endpoints of the brushes and the velocity of the brush. The velocity of the brush is used to select between two different brushes – one of which is a splatter brush (for high velocities), and the other is a fine brush (for low velocities). If the squared distance traveled is greater than some threshold T, then the splatter brush is selected, else the fine brush. Each brush is defined by it's "core" pixels. These core pixels are passed to the painter, which will then have color bleeding effects from these "core" pixels. Note that each brush is anchored by its endpoints (noted as crosses in the figures below).

The fine brush will look something as follows:



The splatter brush will actually be a pseudo-random alternation between the following:



Each brush is stored as a bitmap collection of pixels in memory, and is angled appropriately to be anchored by the endpoints.

Painter

The painter module takes the core pixels for the brush. These core pixels get the color and saturation passed as inputs to the module. Pixels surrounding the core pixels are given values with saturation that decreases with distance from the core pixels.

This color is then alpha blended with the current pixel on the screen.

ZBT

Overall we store two different frames:

- Frame for video capture: 800x600x18 bits (18 bits color; 800x600 resolution)
- Frame for painting: 800x600x18 bits (15 bits color; 3 bits saturation; 800x600 resolution)

The ZBT memory provided in the system is more than enough.

Saturation

The saturation module takes as input a one-bit signal that determines whether the saturation module should add or subtract from its current saturation. Saturation is stored as a 3-bit number. The saturation module should subtract if the user is painting over the canvas. The saturation should add if the user is picking up color from the color palette.

Color Generator

The color generator module takes input three bits from the saturation module, which is how saturated the brush is with the color it currently holds, which we will denote beta, as well as fifteen bits from the intention detector, which represents the color that the user is currently trying to pick up. The color generator looks at the saturation of the previous color and then the color that the user is trying to pick up to make the new, resulting color. The color that the user is picking up from the palette is fully saturated, and we make a weighted average of the old color

and the color picked up from the palette using as weights beta, and 8 – beta, respectively. The color generator module then passes the color to the painter module.

Display

The display module interfaces with the ZBT which stores the painting, and it receives as input from the gesture recognizer module the x and y positions of the LEDs. Given x1, y1, x2, and y2, the display module will form crosshairs to enable the user to see where the LED is. Additionally, the display module will interface with the ZBT to get the painting out, and then it will aggregate the crosshairs and the painting to send to the monitor.

Expansion: Game Logic

If there is time, we will add in functionality for a game. The game would display the outline of a shape on the screen, as well as the required color, and the player would be required to first create the color and then draw the shape over the provided outline. The game functionality will be implemented by two modules, the Game Pixel Generator and the Scorer.

Game Pixel Generator

The game pixel generator module will probably just create circles or squares somewhere on the screen. If more complicated shapes were to be implemented, we'd probably have to prefabricate them and store then in a ZBT. All the game pixel generator does is decide upon a color and a shape, which it will pass to both the display and the Scorer module.

Scorer

The scorer module will take the desired shape and the color from the Game Pixel Generator module as well as the frame that the player produced from the ZBT. It will iterate over the frame passed in by the Game Pixel Generator while looking at the frame produced the player, comparing the two. It will look at all the pixels where the user was supposed to have painted a certain color to make sure that it is indeed that color, while keeping track of the total number of target pixels. Then, when the scorer module has looked at the entire frame, it has a final score, which it will send to the display module to display back to the player.