CS 248 Assignment 1
Paint Program

Introduction to Computer Graphics Help Session
slides by Georg Petschnigg
Modified and presented by Rene Patnode
Stanford University
October 4, 2002

Session Overview

- **Getting Started**
- **Assignment Discussion**
  - Overpainting Brush
  - Tinting Brush
  - Brush Visualization
- **Grading Details**
- **Extra Credit**
- **Questions**

Getting Started

1. Read assignment carefully and pay attention to the details.
2. Go to review session
3. Familiarize yourself with Raptor/Firebird Lab Located in the Basement of Sweet Hall

Development Environment

- **Ways to work with TA Support**
  - Go to Sweet Hall (Best Way – 5 Minutes)
  - Work Remotely (Good Way – 5 Minutes)
- **Ways to work without TA support**
  - Reproduce Sweet Hall Lab development environment on you own Machine (1 Hour)
  - Your code still has to work on the Sweet Hall machines (more risk for you)

Sweet Hall

1. Pick a free computer, Log on
2. Copy assignment from
   `usr/class/cs248/assignments/assignment1/`
   to local directory
3. Run ‘`make`’
4. Run ‘`./paint.i386-linux`’

Working Remotely

1. ssh to firebird, raptor or leland
2. Export the display (using e.g. Exceed)
3. Follow instructions on previous slide

```
>ssh raptor.stanford.edu
>setenv DISPLAY yourIP:0.0
>xterm &
```

Assignment Discussion

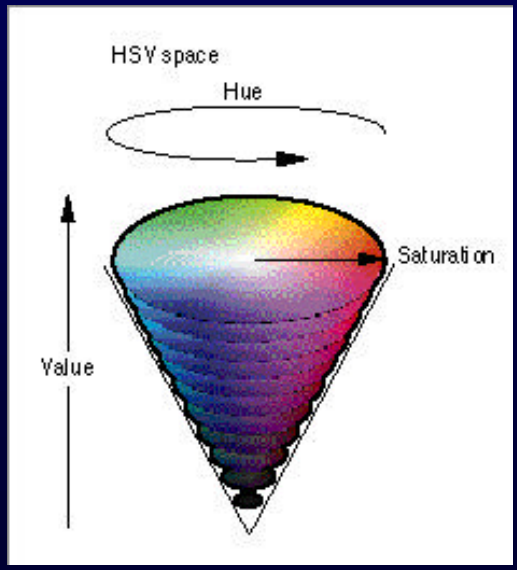
- **You are going to write a paint program**
  - Teaches you 2D Raster Graphics
  - Visualize concepts learned in Class (Brushes, HSV)
  - This assignment is a lot of fun
  - Be creative with extra credit

- **The next slides follow the Assignment (Handout #3) step by step**
  - Reminder: Read the assignment

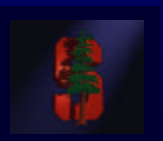
Part 1: Over Painting Brush

- **Rectangular Overpainting Brush**
  - Like Microsoft Paint or “Pencil Tool” in PhotoShop
- **Color Picker for RGB, HSV**
  - See http://www.ewertb.com/java/Java_ColorPicker.html or any commercial Paint Program
    - Value (1.0 bright, 0.0 black)
    - Saturation
      (1.0 strong hue, 0.0 faded hue)
- **Size Control for Brush**

Demo: Painting, Picking Colors in PhotoShop



Part 1: Basic Painting Loop

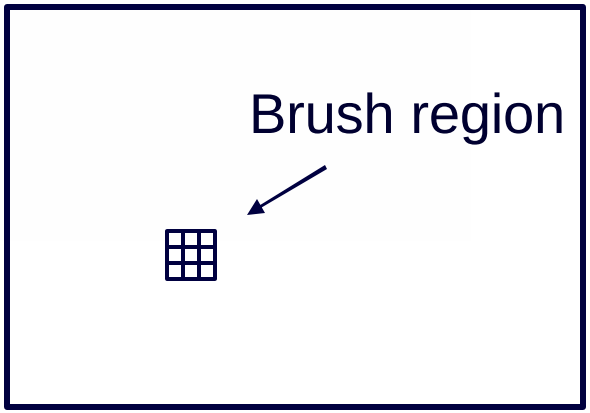


Basic painting loop:

```
do Forever
    readmouse (x, y, buttondown)
    if buttondown then
        do i = - brushwidth/2 to brushwidth/2
            do j = - brushwidth/2 to brushwidth/2

                modifypixel (x, y, i, j)

            enddo
        enddo
    endif
enddo
```



Part 1: Over Painting Brush

- Once you are done with Part 1 you should be able to draw some basic images

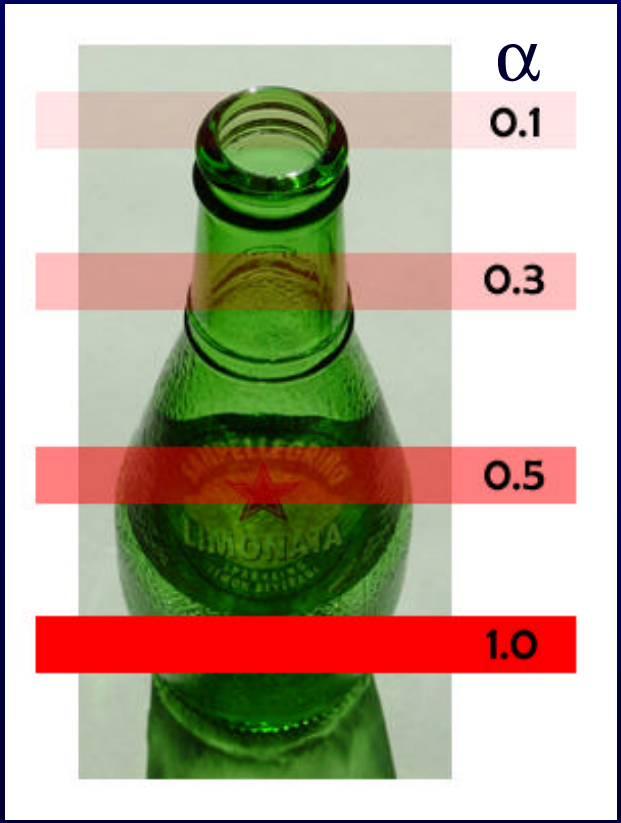
- Notice the hard edges and jaggies around the stroke… this is what Part 2 will fix

Part 2: Tinting Brush

- **Implement Weighted Mask Driven Brush as described in Handout #4**
  - Instead of a rectangular brush, have it gently “blend” to its surroundings. Use HSV interpolation
- **Checkboxes for interpolating along H,S,V axis**
  - Allow all permutations HSV, HS, HV, SV, H, S, V
- **Choose a mask function an give use control over it**
  - Make sure it gradually falls of to zero at the edge!

Part 2: Weighted Blending

Like painting with partially transparent paint.
Commonly referred to as “alpha” blending.

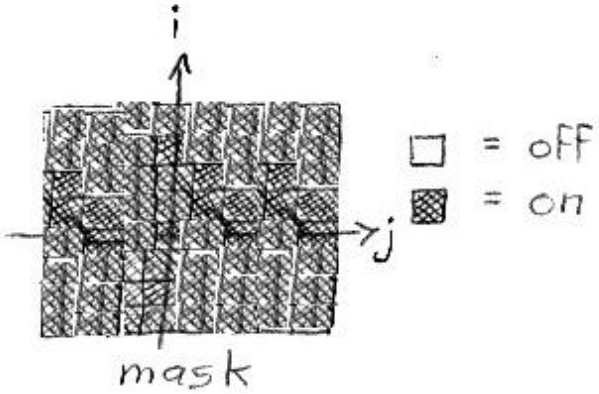


Compositing equation

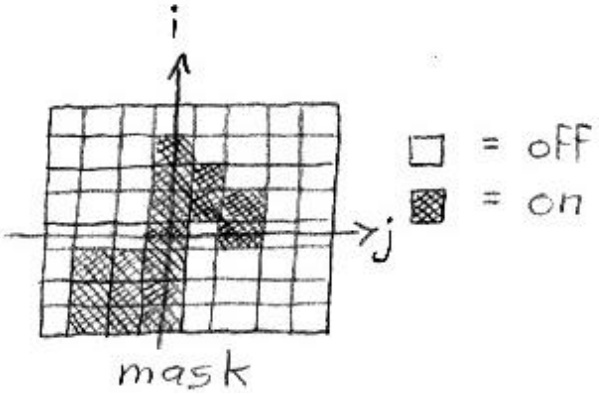
$$C_{new} = (1-\alpha)\, C_{old} + \alpha\, C_{paint}$$

Part 2: Mask driven painting

Lookup array determines how each pixel in the brush is affected.



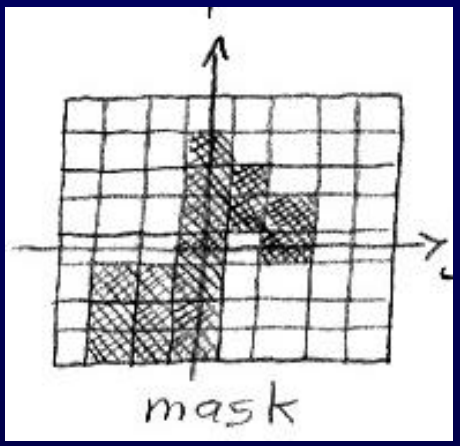
Paint every pixel in the brush region

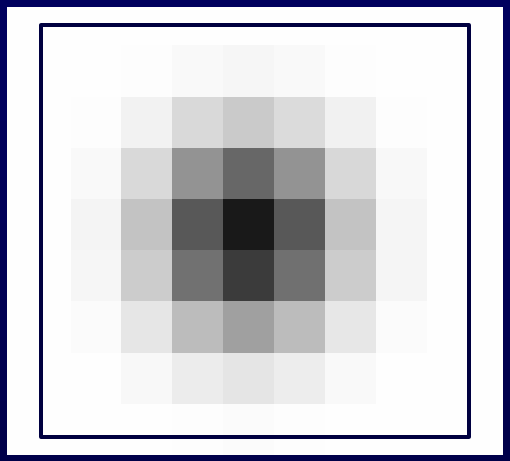


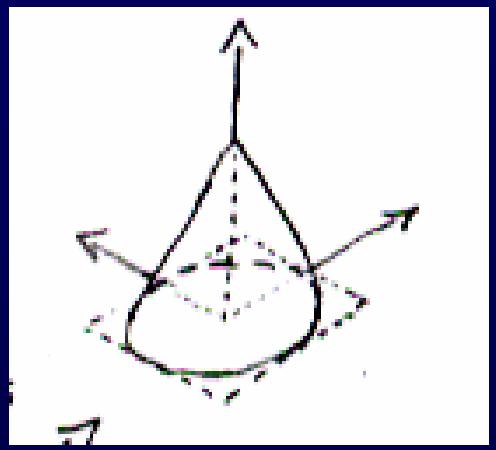
Paint only some of the pixels

Part 2: Weighted mask driven painting

Mask contains alpha/weight for each pixel in brush







- $0 \leq \alpha \leq 1$ everywhere
- α is highest in the middle of the mask, ≤ 1
- α is smooth ($\geq C^0$) except (optionally) at the center
- α falls off to zero at the edges of the mask

Part 2: RGB vs. HSV interpolation

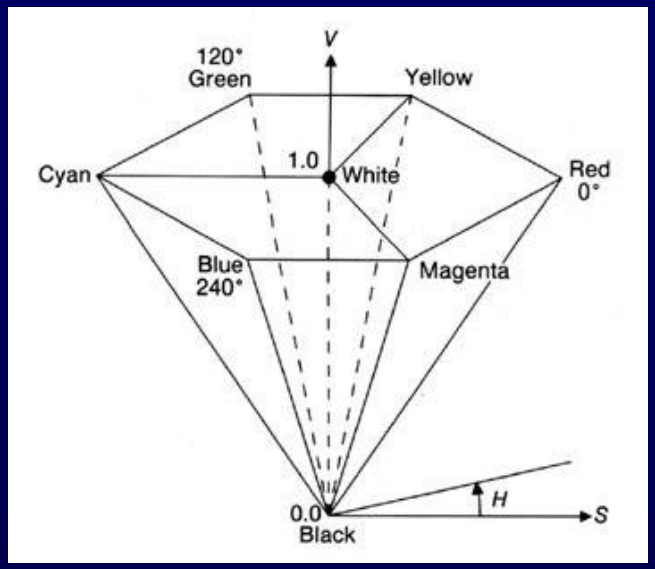
RGB interpolation

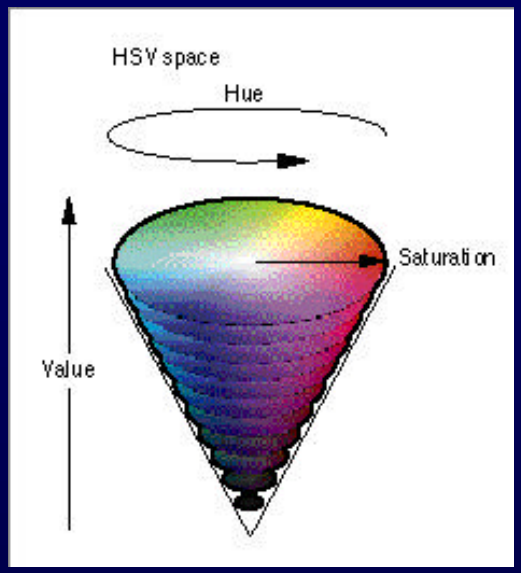
$$NewR = (1-a)\,CanvasR + a\,PaintR$$
$$NewG = (1-a)\,CanvasG + a\,PaintG$$
$$NewB = (1-a)\,CanvasB + a\,PaintB$$

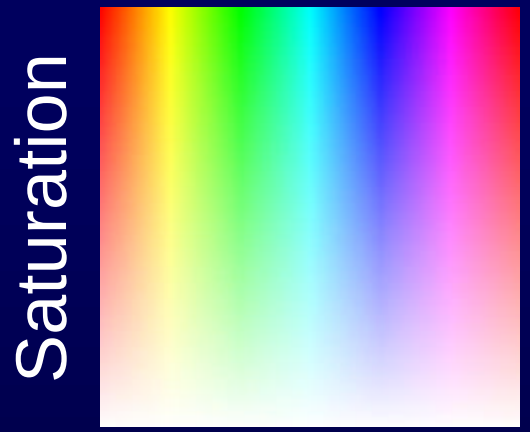
HSV interpolation

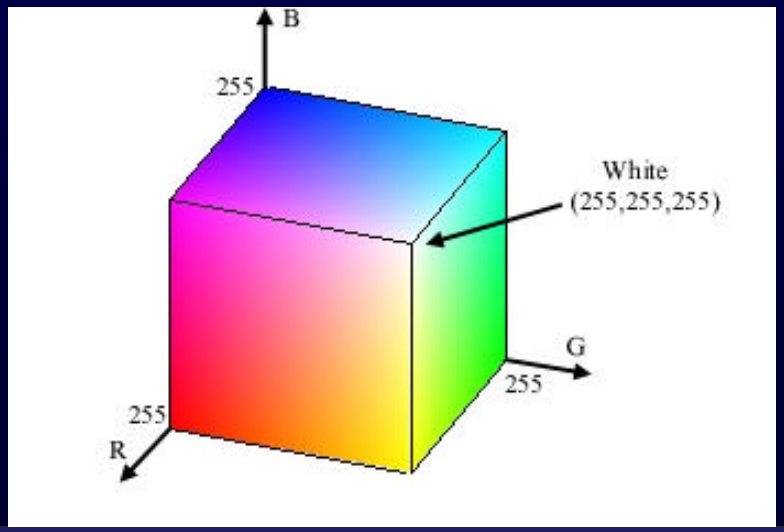
$$NewH = (1-a)\,CanvasH + a\,PaintH$$
$$NewS = (1-a)\,CanvasS + a\,PaintS$$
$$NewV = (1-a)\,CanvasV + a\,PaintV$$

Part 2: RGB vs. HSV interpolation

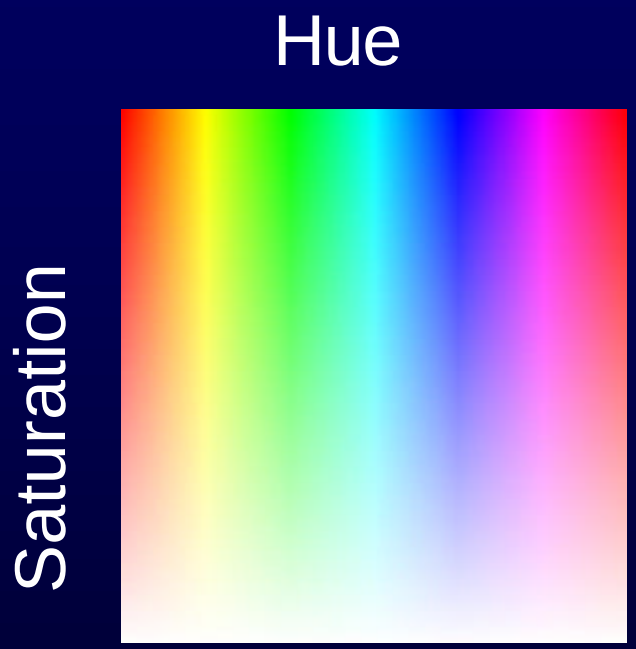


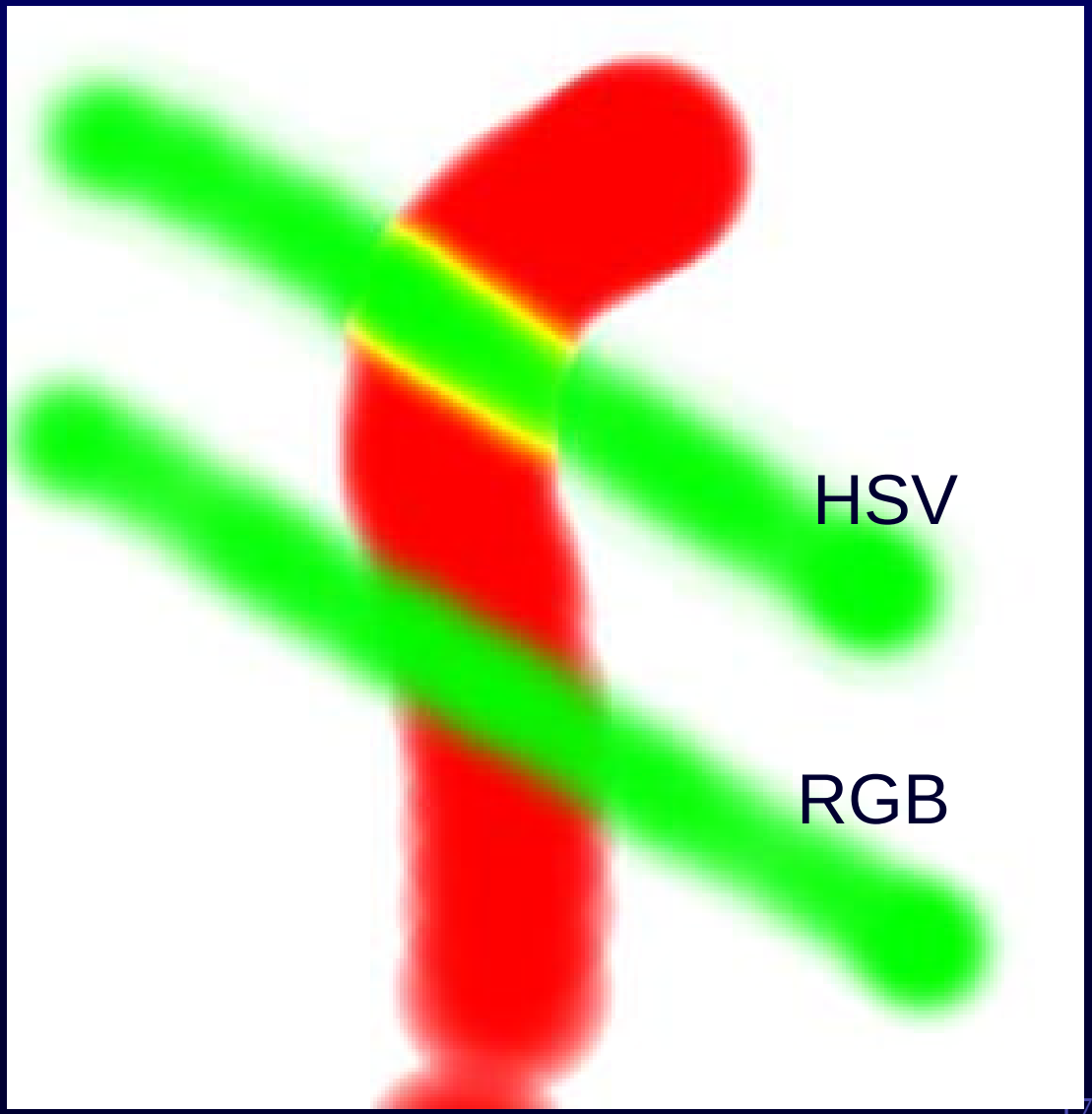






Part 2: RGB vs. HSV interpolation





Part 2: Math Example

- Interpolating half way between Red and Cyan ($a = 0.5$)
- NewColor = 0.5 Cyan + 0.5 Red

	R	G	B	H	S	V
Cyan	0.0	1.0	1.0	180	1.0	1.0
Red	1.0	0.0	0.0	0	1.0	1.0
Interpolation	0.5	0.5	0.5	90	1.0	1.0

50% Gray

Greenish

Part 2: HSV Checkboxes

- **Choose which HSV components to affect.**
- **Allow for any combination.**

if (H_check) { NewH = (1-α) CH + α PaintH }
if (S_check) { NewS = (1-α) CS + α PaintS }
if (V_check) { NewV = (1-α) CV + α PaintV }

Part 2: Sample Images

Overpainting #1 vs. Weighted Mask driven painting #2

Image showing H, S, and V Tinting

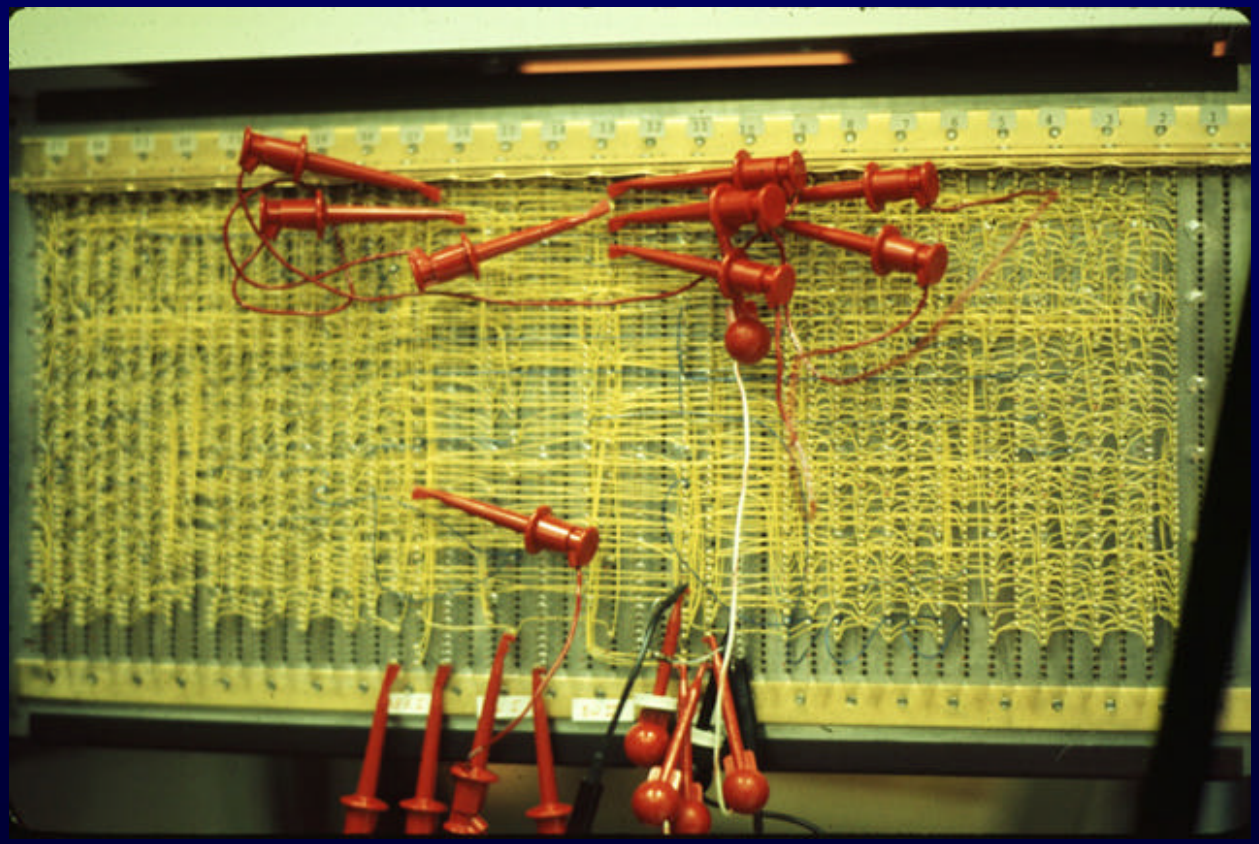
Part 3: Brush Visualization

- **Brush Visualization should tell user what its color, falloff and size is**
  - Brush should always be visible regardless of color
  - Draw 1x (actual size) and 4x (four times larger in x and y) versions of the brush
  - Make the larger version discretized – that is it should be a choppy/chunky/pixel replicated version of the actual brush (think xmag, snoop)
  - Make sure this visualization will help you explain to user, TAs, Professor and yourself how the brush weights affect drawing

Requirements

- **Correctness (40%)**
  - Don’t crash
  - Implement all required features
    - (Read the directions like a lawyer)
- **Efficiency (20 %)**
  - No noticeable lag while using your application
- **User Interface (20%)**
- **Programming Style (20%)**
  - Copying code (Don’t do it)
- **Submitting with ‘/usr/class/cs248/bin/submit’**

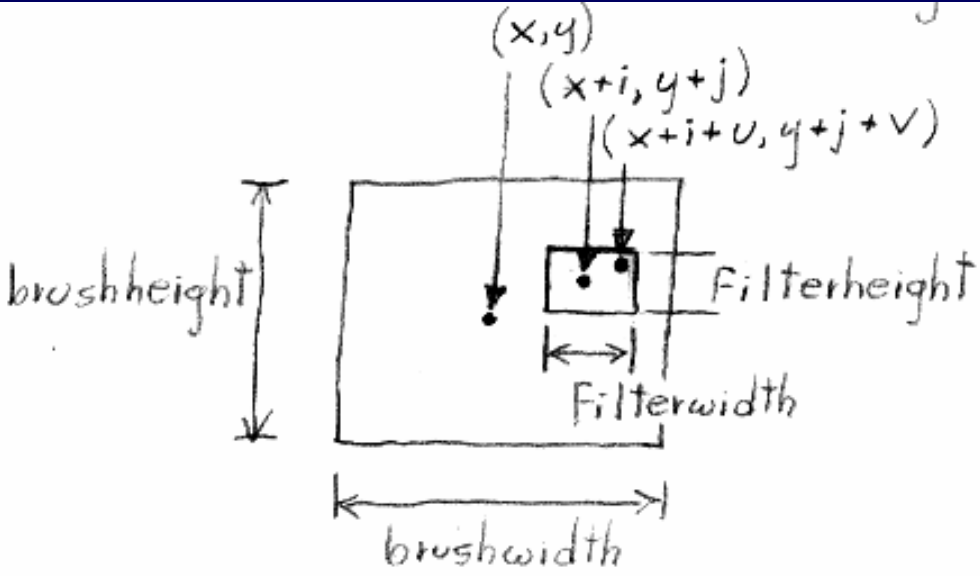
Paint Program 1973



Source: Dick Shoup “SuperPaint: An Early Frame Buffer Graphics System” IEEE Annals of the History of Computing, Vol 23, No 2, Apr-Jun 2001

Extra credit example

Blurring the image under the brush.



and $\sum \omega = \sum_{u,v \in Filter} \omega(u,v)$

Typical Filters are:

1	1	1
1	1	1
1	1	1

or

0	-1	0
-1	4	-1
0	-1	0

Questions?

- Ask now
- Come to Office Hours
- Email: cs248-aut0304-tas@lists.stanford.edu

- Remember: Computer Graphics is fun - if you are not having fun ask TAs for help