I. Color Spaces

- Definition: A mapping of color components onto a Cartesian coordinate system in three or more dimensions.

- RGB, CMY, XYZ, HSV, HLS, Lab, UVW, YUV, YCrCb, Luv, L'* u'* v'*

- Different Purposes: display, editing, computation, compression,

- Equally distant colors may not be equally perceivable
Additive Model: (RGB System)

- R, G, B normalized on orthogonal axes
- All representable colors inside the unit cube
- Color Monitors mix R, G and B
- Video cameras pick up R, G and B
- 3 fixed components acting alone can’t generate all spectrum colors.

Problems with RGB

- Only a small range of potential perceivable colors (particularly for monitor RGB)
- It isn’t easy for humans to say how much of RGB to use to get a given color
  - How much R, G and B is there in “brown”?
- Perceptually non-linear
  - Two points, a certain distance apart, may be perceptually different in one part of the space, but could be same in another part of the space.

Subtractive model (CMY System)

- Color results from removal of light from the illumination source
- Pigments absorb R, G or B and so give C, M or Y
- Used in deskjet/inkjet printers.
- No ink (pigment) = white
Specifying Color

- Color perception usually involves three quantities:
  - Hue: Distinguishes between colors like red, green, blue, etc
  - Saturation: How far the color is from a gray of equal intensity
  - Lightness: The perceived intensity of a reflecting object
- Sometimes lightness is called brightness if the object is emitting light instead of reflecting it.

How Do Artists Do It?

- Artists often specify color as tints, shades, and tones of saturated (pure) pigments
- Tint: Gotten by adding white to a pure pigment, decreasing saturation
- Shade: Gotten by adding black to a pure pigment, decreasing lightness
- Tone: Gotten by adding white and black to a pure pigment
Computer scientists frequently use an intuitive color space that corresponds to tint, shade, and tone:

- **Hue** - The color we see (red, green, purple)
- **Saturation** - How far is the color from gray (pink is less saturated than red, sky blue is less saturated than royal blue)
- **Brightness (Luminance)** - How bright is the color (how bright are the lights illuminating the object?)

### HSV Color Space

- **H** = Hue
- **S** = Saturation
- **V** = Value (or brightness)

http://www.cs.rit.edu/~ncs/color/a_spaces.html
HSV System

- Normally represented as a cone or hexcone
- Hue is the angle around the circle or the regular hexagon; $0 \leq H \leq 360$
- Saturation is the distance from the center; $0 \leq S \leq 1$
- Value is the position along the axis of the cone or hexcone; $0 \leq V \leq 1$
- Value is not perceptually-based, so colors of the same value may have slightly different brightness
- Main axis is grey scale

```java
if ( S == 0 )                       //HSV values = From 0 to 1
    { R = V * 255                      //RGB results = From 0 to 255
      G = V * 255
      B = V * 255
    }
else
    { var_h = H * 6
      var_i = floor( var_h )           //Or ... var_i = floor( var_h )
      var_1 = V * ( 1 - S )
      var_2 = V * ( 1 - S * ( var_h - var_i ) )
      var_3 = V * ( 1 - S * ( 1 - ( var_h - var_i ) ) )
      if ( var_i == 0 ) { var_r = V        ; var_g = var_3 ; var_b = var_1 }
      else if ( var_i == 1 ) { var_r = var_2 ; var_g = V        ; var_b = var_1 }
      else if ( var_i == 2 ) { var_r = var_1 ; var_g = V        ; var_b = var_3 }
      else if ( var_i == 3 ) { var_r = var_1 ; var_g = var_2 ; var_b = V    }
      else                          { var_r = V       ; var_g = var_1 ; var_b = var_2  }
      R = var_r * 255                  //RGB results = From 0 to 255
      G = var_g * 255
      B = var_b * 255
    }
```

II. Accelerometer

- Micro-electro-mechanical system that measures force
  - $F = ma$ (I. Newton)
  - Measured as change in capacitance between moving plates
  - Designed for a maximum g-force (e.g., 2-10g)
  - 2-axis and 3-axis versions
  - Used in airbags, laptop disk drives, etc.
**Analog Devices ADXL202**

- 2-axis accelerometer
  - Set 0g at 50% duty-cycle
  - Positive acceleration increases duty cycle
  - Negative acceleration decreases duty cycle
  - 12.5% per g in either direction

**Typical measurement for ADXL202**

- Noisy data – all forces are aggregated by accelerometer
- Sample trace at 250Hz

**Typical signal from ADXL202**

- Cause interrupts at Ta, Tb, and Tc from X-axis output
  1. Look for rising edge, reset counter: Ta = 0
  2. Look for falling edge, record timer: Tb = positive duty cycle
  3. Look for rising edge, record timer, reset counter: Tc = period
  Repeat from 2
- Same for Y-axis output (T2 is the same for both axes)
What to do about noise/jitter?

- Average over time – smoothing
  - Software filter – like switch debouncing
- Take several readings
  - Use average for Tb and Tc or their ratio
- Running average so that a reading is available at all times
  - E.g., update running average of 4 readings
  - Current average = \( \frac{3}{4} \) current average + \( \frac{1}{4} \) new reading
- Take readings of both Tb and Tc to be extra careful
  - Tc changes with temperature
  - Usually can do Tc just once

Built-in filter

- Filter capacitors limited noise frequency
  - Bandwidth limiting

<table>
<thead>
<tr>
<th>Bandwidth (Hz)</th>
<th>Capacitor Value (μF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.47</td>
</tr>
<tr>
<td>50</td>
<td>0.10</td>
</tr>
<tr>
<td>100</td>
<td>0.05</td>
</tr>
<tr>
<td>200</td>
<td>0.027</td>
</tr>
<tr>
<td>500</td>
<td>0.01</td>
</tr>
<tr>
<td>5 kHz</td>
<td>0.001</td>
</tr>
</tbody>
</table>

ADXL202 Output

- Accelerometer duty cycle varies with force
  - 12.5% for each g
  - \( R_{SET} \) determines duration of period
- At 1g duty-cycle will be 62.5% (37.5%)

ADXL202 Orientation

- Sensitivity (maximum duty cycle change per degree) is highest when accelerometer is perpendicular to gravity
PWM Calculations

- How big a counter do you need?
- Assume 7.37MHz clock
- 1ms period yields a count of 7370
  - This fits in a 16-bit timer/counter
- Should you use a prescaler for the counter?
- Bit precision issues

```c
unsigned int positive;
unsigned int period;
unsigned int pos_duty_cycle;

BAD:
    pos_duty_cycle = positive/period;
BAD:
    pos_duty_cycle = ( positive * 1000 ) / period;
OKAY:
    pos_duty_cycle = ( (long) positive * 1000 ) / period;
```