Today

• Finish image morphing
• Dynamic Range: Is 8 bits per channel enough?
• Trichromacy: Is RGB 3 bands enough?
• Start on 3D graphics

• Reading:
  – Shirley ch 3.1-3.4
  – Shirley ch 20.1-20.2
  – Shirley ch 6, except 6.2.2
Barycentric coordinates

- Idea: represent $P$ using $A_1, A_2, A_3$

\[
P - A_1 = \beta \cdot (A_2 - A_1) + \gamma \cdot (A_3 - A_1)
\]

\[
P = (1 - \beta - \gamma) \cdot A_1 + \beta \cdot A_2 + \gamma \cdot A_3
\]

\[
P = t_1 \cdot A_1 + t_2 \cdot A_2 + t_3 \cdot A_3
\]

\[
t_1 + t_2 + t_3 = 1
\]

\[
t_1 = \frac{\text{area}(PA_2A_3)}{\text{area}(A_1A_2A_3)}
\]

\[
t_2 = \frac{\text{area}(PA_3A_1)}{\text{area}(A_1A_2A_3)}
\]

\[
t_3 = \frac{\text{area}(PA_1A_2)}{\text{area}(A_1A_2A_3)}
\]
Non-parametric image warping

\[ P = w_A A + w_B B + w_C C \]

\[ P' = w_A A' + w_B B' + w_C C' \]

Barycentric coordinate

Turns out to be equivalent to affine transform
Non-parametric image warping

Gaussian: \( \rho(r) = e^{-\beta r^2} \)

\[ \Delta P = \frac{1}{K} \sum_i \rho(P' - X_i) \Delta X_i \]
Demo

• http://www.colonize.com/warp/warp04-2.php
• Warping is a useful operation for mosaics, video matching, view interpolation and so on.
Image morphing
Image morphing

• The goal is to synthesize a fluid transformation from one image to another.

• Cross dissolving is a common transition between cuts, but it is not good for morphing because of the ghosting effects.

\[(1-t) \cdot \text{Image}1 + t \cdot \text{Image}2\]
Image morphing

• Why ghosting?
• Morphing = warping + cross-dissolving

shape (geometric) color (photometric)
Image morphing

image #1       cross-dissolving       image #2

warp

morphing

warp
Morphing sequence
Image morphing

create a morphing sequence: for each time $t$

1. Create an intermediate warping field (by interpolation)
create a morphing sequence: for each time $t$

1. Create an intermediate warping field (by interpolation)
2. Warp both images towards it
Image morphing

create a morphing sequence: for each time $t$

1. Create an intermediate warping field (by interpolation)
2. Warp both images towards it
Image morphing

create a morphing sequence: for each time \( t \)

1. Create an intermediate warping field (by interpolation)

2. Warp both images towards it

3. Cross-dissolve the colors in the newly warped images
More complex morph

• Triangular Mesh
Results

Michael Jackson’s MTV “Black or White”
http://www.michaeljackson.com/quicktime_blackorwhite.html
Multi-source morphing
Multi-source morphing
The average face

- [http://www.uni-regensburg.de/Fakultaeten/phil_Fak_II/Psychologie/Psy_II/beautycheck/english/index.htm](http://www.uni-regensburg.de/Fakultaeten/phil_Fak_II/Psychologie/Psy_II/beautycheck/english/index.htm)

On the left: the “real” Miss Germany 2002 (= Miss Berlin) and on the right: the “virtual” Miss Germany, which was computed by blending together all contestants of the final round and was rated as being much more attractive.
3D Face morphing

http://www.youtube.com/watch?v=nice6NYb_WA
Blanz and Vetter, SIGGRAPH 1998
Dynamic Range

- The dynamic range is the ratio between the maximum and minimum values of a physical measurement. Its definition depends on what the dynamic range refers to.
  - For a Scene: ratio between the brightest and darkest parts of the scene.
  - For a Display: ratio between the maximum and minimum intensities emitted from the screen.
Dynamic Range

- Dynamic range is a ratio and as such a dimensionless quantity. In photography and imaging, the dynamic range represents the ratio of two luminance values, with the luminance expressed in candelas per square meter.

- A scene showing the interior of a room with a sunlit view outside the window, for instance, will have a dynamic range of approximately 100,000:1.

- The human eye can accommodate a dynamic range of approximately 10,000:1 in a single view.
Dynamic Range

- Dynamic range is a ratio and as such a dimensionless quantity. In photography and imaging, the dynamic range represents the ratio of two luminance values, with the luminance expressed in candelas per square meter.

<table>
<thead>
<tr>
<th>Lum. (cd/m²)</th>
<th>0.00001</th>
<th>0.001</th>
<th>1</th>
<th>100</th>
<th>10,000</th>
<th>1,000,000</th>
<th>10^8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>starlight</td>
<td>moonlight</td>
<td>indoor lighting</td>
<td>outdoor shade</td>
<td>outdoor sunlit</td>
<td>sun</td>
<td></td>
</tr>
</tbody>
</table>

- A scene showing the interior of a room with a sunlit view outside the window, for instance, will have a dynamic range of approximately 100,000:1.

- The human eye can accommodate a dynamic range of approximately 10,000:1 in a single view.
  - Beyond that, use adaptation to adjust
Dynamic Range

- Image depth refers to the number of bits available.
- We can use those bits to represent
  - a large range at low resolution, or
  - a small range at high resolution

<table>
<thead>
<tr>
<th>Lum. (cd/m^2)</th>
<th>0.00001</th>
<th>0.001</th>
<th>1</th>
<th>100</th>
<th>10,000</th>
<th>1,000,000</th>
<th>10^8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>starlight</td>
<td>moonlight</td>
<td>indoor</td>
<td>lighting</td>
<td>outdoor</td>
<td>shade</td>
<td>outdoor</td>
</tr>
</tbody>
</table>

![Image of starlight setting]

![Image of sunlit setting]
High dynamic imaging

High Dynamic Range Image

Floating point pixel

Tone mapped image
8 bits, for standard monitors

http://www.hdrsoft.com/examples.html
Intensity Perception

• Humans are actually tuned to the ratio of intensities, not their absolute difference
  – So going from a 50 to 100 Watt light bulb looks the same as going from 100 to 200
  – So, if we only have 4 intensities, between 0 and 1, we should choose to use 0, 0.25, 0.5 and 1

• Ignoring this results in poorer perceptible intensity resolution at low light levels, and better resolution at high light levels
  – It would use 0, 0.33, 0.66, and 1
Display on a Monitor

• When images are created, a linear mapping between pixels and intensity is assumed
  – For example, if you double the pixel value, the displayed intensity should double
• Monitors, however, do not work that way
  – For analog monitors, the pixel value is converted to a voltage
  – The voltage is used to control the intensity of the monitor pixels
  – But the voltage to display intensity is not linear
  – Similar problem with other monitors, different causes
• The outcome: A linear intensity scale in memory does not look linear on a monitor
• Even worse, different monitors do different things
Gamma Control

• The mapping from voltage to display is usually an exponential function: \( I_{\text{display}} \propto I_{\text{to-monitor}}^{\gamma} \)

• To correct the problem, we pass the pixel values through a \textit{gamma function} before converting them to the monitor

\[ I_{\text{to-monitor}} \propto I_{\text{image}}^{1/\gamma} \]

• This process is called \textit{gamma correction}

• The parameter, \( \gamma \), is controlled by the user
  – It should be matched to a particular monitor
  – Typical values are between 2.2 and 2.5

• The mapping can be done in hardware or software
Color Vision

• We have seen that humans have three sensors for color vision

• What’s the implications for digital color representation...
Qualitative Response

[Graph showing sensitivity (ρ) and number of photons (E) for different wavelengths (400 to 700).] Multiply ρE and calculate the area under the curve. If the area is large, it results in a big response!
Qualitative Response

Multiply $\rho E$

Area under curve?

Tiny response!
Trichromacy means...

**Color Matching:**
People think these two spectra *look the same*

**Representing color:**
If you want people to “see” the continuous spectrum, you can just show the three primaries
For almost any given $E_{\text{target}}(\lambda)$, we can solve for $[r,g,b]$ so that the displayed color looks indistinguishable from the target color to our eyes.
Trichromacy

• Many colors can be represented as a mixture of R, G, B: M=rR + gG + bB (Additive matching)
• Gives a color description system - two people who agree on R, G, B need only supply (r, g, b) to describe a color
• Some colors can’t be matched like this, instead, write: M+rR=gG+bB (Subtractive matching)
  – Interpret this as (-r, g, b)
  – Problem for reproducing colors – you can’t subtract light using a monitor
Primaries are Spectra Too

• A primary can be a spectrum
  – Single wavelengths are just a special case

![Graph showing three primary wavelengths at 400, 500, and 600, followed by another graph showing a spectrum with peaks at the same wavelengths.](image-url)
Color Matching

• Given a spectrum, how do we determine how much each of R, G and B to use to match it?

• Procedure:
  – For a light of unit intensity *at each wavelength*, ask people to match it using some combination of R, G and B primaries
  – Gives you, r(\(\lambda\)), g(\(\lambda\)) and b(\(\lambda\)), the amount of each primary used for wavelength \(\lambda\)
  – Defined for all visible wavelengths, r(\(\lambda\)), g(\(\lambda\)) and b(\(\lambda\)) are the RGB *color matching functions*
The RGB Color Matching Functions

4.13 THE COLOR-MATCHING FUNCTIONS ARE THE ROWS OF THE COLOR-MATCHING SYSTEM MATRIX. The functions measured by Stiles and Burch (1959) using a 10-degree bipartite field and primary lights at the wavelengths 645.2 nm, 525.3 nm, and 444.4 nm with unit radiant power are shown. The three functions in this figure are called $\tilde{r}_{10}(\lambda)$, $\tilde{g}_{10}(\lambda)$, and $\tilde{b}_{10}(\lambda)$. 
Computing the Matching

• Given a spectrum $E(\lambda)$, how do we determine how much each of R, G and B to use to match it?
• The spectrum function that we are trying to match, $E(\lambda)$, gives the amount of energy at each wavelength
• The RGB matching functions describe how much of each primary is needed to give one energy unit’s worth of response at each wavelength

$$E = rR + gG + bB \quad r = \int r(\lambda)E(\lambda)d\lambda$$

$$\quad g = \int g(\lambda)E(\lambda)d\lambda$$

$$\quad b = \int b(\lambda)E(\lambda)d\lambda$$
Color Spaces

• The principle of trichromacy means that the colors displayable are all the linear combination of primaries
• Taking linear combinations of R, G and B defines the *RGB color space*
• If R, G and B correspond to a monitor’s phosphors (monitor RGB), then the space is the range of colors displayable on the monitor
RGB Cube

Cyan (0,1,1)

White (1,1,1)

Green (0,1,0)

Yellow (1,1,0)

Blue (0,0,1)

Magenta (0,1,1)

Black (0,0,0)

Red (1,0,0)

Demo
Problems with RGB

• It isn’t easy for humans to say how much of RGB to use to make a given color
  – How much R, G and B is there in “brown”? (Answer: .64,.16, .16)
HSV Color Space  (Alvy Ray Smith, 1978)

• Hue: the color family: red, yellow, blue...
• Saturation: The purity of a color: white is totally unsaturated
• Value: The intensity of a color: white is intense, black isn’t
• Space looks like a cone
  – Parts of the cone can be mapped to RGB space
• Not a linear space, so no linear transform to take RGB to HSV
  – But there is an algorithmic transform
#define RETURN_HSV(h, s, v) {HSV.H = h; HSV.S = s; HSV.V = v; return HSV;}
#define RETURN_RGB(r, g, b) {RGB.R = r; RGB.G = g; RGB.B = b; return RGB;}
#define UNDEFINED -1

// Theoretically, hue 0 (pure red) is identical to hue 6 in these transforms. Pure red always maps to 6 in this implementation. Therefore UNDEFINED can be defined as 0 in situations where only unsigned numbers are desired.

typedef struct {float R, G, B;} RGBType;
typedef struct {float H, S, V;} HSVType;

HSVType RGB_to_HSV( RGBType RGB ) {
    // RGB are each on [0, 1]. S and V are returned on [0, 1] and H is returned on [0, 6]. Exception: H is returned UNDEFINED if S==0.
    float R = RGB.R, G = RGB.G, B = RGB.B, v, x, f;
    int i;
    HSVType HSV;
    x = min(R, G, B);
    v = max(R, G, B);
    if(v == x) RETURN_HSV(UNDEFINED, 0, v);
    f = (R == x) ? G - B : ((G == x) ? B - R : R - G);
    i = (R == x) ? 3 : ((G == x) ? 5 : 1);
    RETURN_HSV(i - f/(v - x), (v - x)/v, v);
}

http://alvyray.com/Papers/hsv2rgb.htm
HSV to RGB

```c
RGBType HSV_to_RGB( HSVType HSV ) {

    // H is given on [0, 6] or UNDEFINED. S and V are given on [0, 1].
    // RGB are each returned on [0, 1].
    float h = HSV.H, s = HSV.S, v = HSV.V, m, n, f;
    int i;
    RGBType RGB;
    if (h == UNDEFINED) RETURN_RGB(v, v, v);
    i = floor(h);
    f = h - i;
    if ( !(i&1) ) f = 1 - f; // if i is even
    m = v * (1 - s);
    n = v * (1 - s * f);
    switch (i) {
    case 6:
        RETURN_RGB(v, n, m);
    case 1: RETURN_RGB(n, v, m);
    case 2: RETURN_RGB(m, v, n)
    case 3: RETURN_RGB(m, n, v);
    case 4: RETURN_RGB(n, m, v);
    case 5: RETURN_RGB(v, m, n);
    }
}
```

http://alvyray.com/Papers/hsv2rgb.htm
http://www.mandelbrot-dazibao.com/HSV/HSV.htm