Announcement

• Please rate the course:
  • https://my.cmu.edu/site/main/page.academics

• Please rate the TAs.
  Kristin Siu  http://www.surveymonkey.com/s.aspx?sm=eGK2BmF5UCv_2fcv47Fdjghw_3d_3d
  Frank Palermo  http://www.surveymonkey.com/s.aspx?sm=1o0WsX46xGrXWBCbJ_2foAgQ_3d_3d
  Eric Butler  http://www.surveymonkey.com/s.aspx?sm=1h0D60GotRz_2f3b0HEVQ7uQ_3d_3d

• Final Exam: HUB (WEH 7500)

• Late days - we will optimize for them

• P4 Due Tonight: Indicate P3 Features
<table>
<thead>
<tr>
<th><strong>Periodic Table of Everything You Need to Know to Ace the Graphics Exam</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DiffEQ Basics</strong></td>
</tr>
<tr>
<td>- General Def. of DiffEQ</td>
</tr>
<tr>
<td>- Understand the basic DiffEqs shown in class</td>
</tr>
<tr>
<td>- Stability and Instability of Systems</td>
</tr>
<tr>
<td><strong>Explicit Integration</strong></td>
</tr>
<tr>
<td>- Write Euler’s method</td>
</tr>
<tr>
<td>- Understand how it works</td>
</tr>
<tr>
<td>- Do simple 1D linear problems involving Euler’s method</td>
</tr>
<tr>
<td><strong>Implicit Integration</strong></td>
</tr>
<tr>
<td>- Write down the implicit method</td>
</tr>
<tr>
<td>- Understand how it works</td>
</tr>
<tr>
<td>- Why is it stable?</td>
</tr>
<tr>
<td>- Do simple 1D linear problems involving the implicit method</td>
</tr>
<tr>
<td><strong>Particle Dynamics</strong></td>
</tr>
<tr>
<td>- How does $F = MA$ become a DiffEQ?</td>
</tr>
<tr>
<td>- How do second order differential equations become first order differential equations?</td>
</tr>
<tr>
<td>- What about 3rd order differential equations?</td>
</tr>
<tr>
<td>- How springs work</td>
</tr>
<tr>
<td><strong>Pixel Filters</strong></td>
</tr>
<tr>
<td>- Understand what they do</td>
</tr>
<tr>
<td><strong>Neighborhood Filters</strong></td>
</tr>
<tr>
<td>- convolutions - understand how they work</td>
</tr>
<tr>
<td>- understand blur / sharpening / edge filters</td>
</tr>
<tr>
<td>- understand median filters</td>
</tr>
<tr>
<td><strong>Dithering</strong></td>
</tr>
<tr>
<td>- deterministic thresholding</td>
</tr>
<tr>
<td>- dithering matrices</td>
</tr>
<tr>
<td>- random thresholding</td>
</tr>
<tr>
<td>- Floyd-Steinberg Error Diffusion</td>
</tr>
<tr>
<td><strong>Basic Animation</strong></td>
</tr>
<tr>
<td>- squatch and stretch</td>
</tr>
<tr>
<td>- traditional cell animation</td>
</tr>
<tr>
<td>- uncanny valley</td>
</tr>
<tr>
<td><strong>Radiosity</strong></td>
</tr>
<tr>
<td>- subdivision</td>
</tr>
<tr>
<td>- 2 pass algorithms</td>
</tr>
<tr>
<td><strong>Keyframing</strong></td>
</tr>
<tr>
<td>- basic ideas</td>
</tr>
<tr>
<td><strong>Hair</strong></td>
</tr>
<tr>
<td>- understand basic spring model for hair</td>
</tr>
<tr>
<td>- what properties of real hair are captured, not captured</td>
</tr>
<tr>
<td><strong>Cloth</strong></td>
</tr>
<tr>
<td>- how to setup a spring network for cloth</td>
</tr>
<tr>
<td>- what properties of real hair are captured, not captured</td>
</tr>
<tr>
<td><strong>Diffusion</strong></td>
</tr>
<tr>
<td>- $q(x, y) = f(q, \frac{\partial q}{\partial x}, \frac{\partial q}{\partial y}, \ldots)$</td>
</tr>
<tr>
<td>- What is the difference between a partial and an ordinary differential equation</td>
</tr>
<tr>
<td>- What is the basic data structure used for storing a PDE. What operations are needed.</td>
</tr>
<tr>
<td><strong>Image Compression</strong></td>
</tr>
<tr>
<td>- JPG compression work</td>
</tr>
<tr>
<td><strong>Image-based Rendering</strong></td>
</tr>
<tr>
<td>- why do people do it</td>
</tr>
<tr>
<td>- what is the plenoptic function</td>
</tr>
</tbody>
</table>
DiffEQ Integration

Differential Equation Basics

Andrew Witkin

Pixar Animation Studios
A Canonical Differential Equation

\[ \dot{x} = f(x, t) \]

- \( x(t) \): a moving point.
- \( f(x, t) \): \( x \)'s velocity.
The differential equation
\[ \dot{x} = f(x, t) \]
defines a vector field over \( x \).
Integral Curves

Start Here

Pick any starting point, and follow the vectors.
Given the starting point, follow the integral curve.
Euler’s Method

- Simplest numerical solution method
- Discrete time steps
- Bigger steps, bigger errors.

\[ x(t + \Delta t) = x(t) + \Delta t f(x, t) \]
Two Problems

- Accuracy
- Instability
Accuracy

Consider the equation:

\[
\dot{x} = \begin{bmatrix}
0 & 1 \\
-1 & 0
\end{bmatrix} x
\]

What do the integral curves look like?
Problem I: Inaccuracy

Error turns $x(t)$ from a circle into the spiral of your choice.
What is this a model for?

http://www.youtube.com/watch?v=3_fLO4xjTqq
Problem 2: Instability

- Consider the following system:

\[
\begin{align*}
\dot{x} &= -x \\
x(0) &= 1
\end{align*}
\]
Problem 2: Instability

Given the starting point, follow the integral curve.

Euler's Method

- Simplest numerical solution method
- Discrete time steps
- Bigger steps, bigger errors.

Problem I: Inaccuracy

Error turns $x(t)$ from a circle into the spiral of your choice.

Problem II: Instability to Neptune!
Accuracy of Euler Method

\[ \dot{x} = f(x) \]

Consider Taylor Expansion about \( x(t) \)...

\[
x(t + h) = x(t) + h f(x(t)) + O(h^2)
\]

Therefore, Euler’s method has error \( O(h^2) \)... it is first order.

How can we get to \( O(h^3) \) error?
“Give me Stability
or
Give me Death”

— Baraff’s other motto
If your step size is too big, your simulation blows up. It isn’t pretty.

Sometimes you have to make the step size so small that you never get anywhere.

Nasty cases: cloth, constrained systems.
A very simple equation

A 1-D particle governed by $\dot{x} = -kx$ where $k$ is a stiffness constant.

$E = \frac{1}{2} kx^2$

$\dot{x} = -kx$

$x$
Euler’s method has a speed limit

\[ x = -kx \quad \Delta x = -hkx \]

\[ h = 0.5(1/k) \quad h = 1(1/k) \quad h = 1.5(1/k) \quad h = 2(1/k) \quad h = 3(1/k) \]

\( h > 1/k: \) oscillate. \quad \( h > 2/k: \) explode!
In more complex systems, step size is limited by the largest $k$. One stiff spring can screw it up for everyone else.

Systems that have some big $k$’s mixed in are called **stiff** systems.
Accuracy

Consider the equation:

\[
\frac{dx}{dt} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x
\]

What do the integral curves look like?
Explicit Integration
Explicit Integration
(Explicit) Euler Method

\[ x(t_0 + h) = x(t_0) + h \dot{x}(t_0) \]
Implicit Euler Method

\[ x(t_0 + h) = x(t_0) + h \dot{x}(t_0) \]

\[ x(t_0 + h) = x(t_0) + h \dot{x}(t_0 + \Delta t) \]
Implicit Integration
Implicit Integration
Implicit Euler

\[ \begin{aligned}
\dot{x} &= -x \\
\dot{x}(0) &= 1
\end{aligned} \]
One Step: Implicit vs. Explicit

\[ \dot{x} = -x, \quad x(0) = 1 \]

**Correct Solution:** \( x(h) = e^{-hk} \)

**Implicit Euler Step:** \( x(h) = \frac{1}{1 + hk} \)

**Explicit Euler Step:** \( x(h) = 1 - hk \)
Particle Dynamics
Overview

- One lousy particle
- Particle systems
- Forces: gravity, springs
- Implementation
Newtonian particle

- Differential equations: \( f = ma \)
- Forces depend on:
- Position, velocity, time

\[
\ddot{x} = \frac{f(x, \dot{x})}{m}
\]
Second order equations

\[ \ddot{x} = \frac{f(x, \dot{x})}{m} \]  
Has 2\textsuperscript{nd} derivatives

\[ \dot{x} = v \]  
Add a new variable v to get

\[ \dot{v} = \frac{f(x, \dot{x})}{m} \]  
a pair of coupled 1\textsuperscript{st} order equations
Phase space

\[
\begin{bmatrix}
  x \\
  v
\end{bmatrix}
\]
Concatenate x and v to make a 6-vector:
position in phase space

\[
\begin{bmatrix}
  \dot{x} \\
  \dot{v}
\end{bmatrix}
\]
Velocity on Phase space:
Another 6-vector

\[
\begin{bmatrix}
  \ddot{x} \\
  \ddot{v}
\end{bmatrix} = \begin{bmatrix}
  v \\
  f / m
\end{bmatrix}
\]
A vanilla 1st-order differential equation
Second order equations

\[
\begin{align*}
\frac{d}{dt} x &= f(x, t) \\
\frac{d}{dt} v &= \frac{d}{dt} \frac{d}{dt} x = \frac{d}{dt} f(x, t)
\end{align*}
\]

Has 2nd derivatives

Add a new variable v to get a pair of coupled 1st order equations

Phase space

Concatenate x and v to make a 6-vector:

\[\begin{bmatrix} x \\ v \end{bmatrix}\]

Position in phase space

Another 6-vector

A vanilla 1st-order differential equation

\[
\begin{align*}
x &= v \\
v &= f(x, t)
\end{align*}
\]

Particle structure

- \[x\] \rightarrow position
- \[v\] \rightarrow velocity
- \[f\] \rightarrow force accumulator
- \[m\] \rightarrow mass

Solver interface
Second order equations

\[ f(x, \dot{x}, t) = a \]

Add a new variable \( v \) to get a pair of coupled first-order equations.

Phase space

Concatenate \( x \) and \( v \) to make a 6-vector:

- Position in phase space
- Velocity on phase space:

A vanilla first-order differential equation

\[ \begin{align*}
\dot{x} &= v \\
\dot{v} &= f/m
\end{align*} \]

Particle structure

Solver interface

\[ \begin{align*}
\text{getDim} &\rightarrow [6] \\
\text{getState} &\leftarrow \begin{bmatrix} x \\ v \end{bmatrix} \\
\text{setState} &\rightarrow \begin{bmatrix} x \\ v \end{bmatrix} \\
\text{derivEval} &\leftarrow \begin{bmatrix} v \\ f/m \end{bmatrix}
\end{align*} \]
Particle systems

\[
\begin{bmatrix}
    x & x & x & x \\
    v & v & v & v \\
    f & f & f & f \\
    m & m & m & m \\
\end{bmatrix}
\quad \cdots \quad
\begin{bmatrix}
    x \\
    v \\
    f \\
    m \\
\end{bmatrix}
\]
Solver interface

particles n time

get/setState

getDim

derivEval

6n

\begin{array}{ccccccc}
x_1 & v_1 & x_2 & v_2 & \cdots & x_n & v_n \\
v_1 & \frac{f}{m_1} & v_2 & \frac{f}{m_2} & \cdots & x_n & \frac{f}{m_n} \\
\end{array}

...
Differential equation solver

\[
\begin{bmatrix}
\dot{x} \\
\dot{v}
\end{bmatrix} = 
\begin{bmatrix}
v \\
f/m
\end{bmatrix}
\]

Euler method: \( x(t + h) = x(t) + h \cdot \&t \)

\[
x_{i+1} = x_i + Vt \cdot \dot{x}
\]

\[
v_{i+1} = v_i + Vt \cdot \dot{v}
\]

Gets very unstable for large \( Vt \)

Higher order solvers perform better: (e.g. Runge-Kutta)
derivEval loop

1. Clear forces
   - Loop over particles, zero force accumulators
2. Calculate forces
   - Sum all forces into accumulators
3. Gather
   - Loop over particles, copying v and f/m into destination array
Forces

- Constant (gravity)
- Position/time dependent (force fields)
- Velocity-dependent (drag)
- N-ary (springs)
Forces

- Constant (gravity)
- Position/time dependent (force fields)
- Velocity-dependent (drag)
- N-ary (springs)

Force structures

Force objects are black boxes that point to the particles they influence, and add in their contribution into the force accumulator.

Global force calculation:
- Loop, invoking force objects
Forces

- Constant (gravity)
- Position/time dependent (force fields)
- Velocity-dependent (drag)
- N-ary (springs)

Force structures

Force objects are black boxes that point to the particles they influence, and add in their contribution into the force accumulator.

Global force calculation:

- Loop, invoking force objects

Particle systems with forces

\[
\begin{align*}
\text{particles} & \quad n & \quad \text{time} & \quad \text{forces} & \quad \text{nf} \\
\begin{bmatrix}
\mathbf{x} \\
\mathbf{v} \\
\mathbf{f} \\
\mathbf{m}
\end{bmatrix} & \quad \begin{bmatrix}
\mathbf{x} \\
\mathbf{v} \\
\mathbf{f} \\
\mathbf{m}
\end{bmatrix} & \quad \begin{bmatrix}
\mathbf{x} \\
\mathbf{v} \\
\mathbf{f} \\
\mathbf{m}
\end{bmatrix} & \quad \begin{bmatrix}
\mathbf{x} \\
\mathbf{v} \\
\mathbf{f} \\
\mathbf{m}
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
p -> \mathbf{f} & += p -> \mathbf{m} \times F -> G
\end{align*}
\]
• Constant (gravity)
• Position/time dependent (force fields)
• Velocity-dependent (drag)
• N-ary (springs)

Force objects are black boxes that point to the particles they influence, and add in their contribution into the force accumulator.

Global force calculation:
• Loop, invoking force objects

Gravity

Force law:
\[ f_{\text{grav}} = mg \]

\[ p->f += p->m \times F->G \]
Viscous drag

Force law:
\[ \mathbf{f}_{\text{drag}} = -k_{\text{drag}} \mathbf{v} \]

\[ \mathbf{p} \rightarrow \mathbf{f} = \mathbf{F} \rightarrow k \ast \mathbf{p} \rightarrow \mathbf{v} \]
Damped spring

Force law:

\[ f_1 = -k_s(|\Delta x| - r) + k_d \left( \frac{\Delta v \Delta x}{|\Delta x|} \right) \frac{\Delta x}{|\Delta x|} \]

\[ f_2 = -f_1 \]

\( r = \text{rest length} \)

\( \Delta x = x_1 - x_2 \)
Viscous drag force law:

\[ \text{drag} = k \cdot v \]

Damped spring force law:

\[ \text{spring} = -k \cdot \nabla \cdot f \]

DerivEval Loop:

1. Clear force accumulators
2. Apply forces to particles
3. Return \([v,f/m,\ldots]\) to solver
Solver interface

- **particles**: List of particles
- **n**: Number of particles
- **time**: Current time

**get/setState**

**getDim**

**derivEval**

- **6n**: Total number of variables (3n positions + 3n velocities)
- **x₁, v₁, x₂, v₂, L, xₙ, vₙ**: Positions and velocities of particles
- **f₁, f₂, L, fₙ**: Forces acting on particles
- **m₁, m₂, L, mₙ**: Masses of particles
Differential equation solver

\[
\begin{bmatrix}
\dot{x} \\
\dot{v}
\end{bmatrix} = \begin{bmatrix}
v \\
f / m
\end{bmatrix}
\]

Euler method:

\[
\begin{bmatrix}
x_1^{i+1} \\
v_1^{i+1} \\
\vdots \\
x_n^{i+1} \\
v_n^{i+1}
\end{bmatrix} = \begin{bmatrix}
x_1^i \\
v_1^i \\
\vdots \\
x_n^i \\
v_n^i
\end{bmatrix} + \begin{bmatrix}
v_1^i \\
f_1^i / m_1 \\
\vdots \\
v_n^i \\
f_n^i / m_n
\end{bmatrix} \Delta t
\]

Bouncing off the walls

• Add-on for a particle simulator
• For now, just simple point-plane collisions

Normal and tangential components

\[
N \cdot V = 0
\]

Collision Detection

Within \( e \) of the wall
Pixel Operations
Point Processing

**Original**
- $x$

**Darken**
- $x - 128$

**Lower Contrast**
- $x / 2$

**Nonlinear Lower Contrast**
- $((x / 255.0) ** 0.33) * 255.0$

**Invert**
- $255 - x$

**Lighten**
- $x + 128$

**Raise Contrast**
- $x * 2$

**Nonlinear Raise Contrast**
- $((x / 255.0) ** 2) * 255.0$
Neighborhood Filters
Neighborhood Operations
Convolution

\[ F = \begin{bmatrix}
  0.2 & 0.1 & -1.0 \\
  0.3 & 0.0 & 0.9 \\
  0.1 & 0.3 & -1.0
\end{bmatrix} \quad I' = F \ast I \]
Convolutions are Linear

\[ F \ast I + G \ast I = (F + G) \ast I \]

\[ 2F \ast I = F \ast 2I = 2(F \ast I) \]

(We will use this fact when we talk about sharpening filters.)
Original Image
Shifted Image
X-Edge Detection

\[
\begin{bmatrix}
0 & 0 & 0 \\
-1 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}
\]
Y-Edge Detection

\[
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 0 \\
0 & -1 & 0
\end{bmatrix}
\]
General Edge Detection

Can this be described as a convolution?
Blurred Image
Blurring Filters

• A simple blurring effect can be achieved with a 3x3 filter centered around a pixel,
• More blurring is achieved with a wider $n \times n$ filter:

Original Image  
Blur 3x3 mask  
Blur 7x7 mask
Image Filtering:  Blurring

original, 64x64 pixels  
3x3 blur  
5x5 blur
Blurred Image
Sharpened Image

$$2 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$
Noise
Blurred Noise
Median Filter

Can this be described as a convolution?
Example: Noise Reduction

- Image with noise
- Median filter (5x5)
Example: Noise Reduction

Original image

Image with noise

Median filter (5x5)
Dithering
Dithering

- Compensates for lack of color resolution
- Eye does spatial averaging
- Black/white dithering to achieve gray scale
  - Each pixel is black or white
  - From far away, color determined by fraction of white
  - For 3x3 block, 10 levels of gray scale
Dithering takes advantage of the human eye's tendency to "mix" two colors in close proximity to one another.
Dithering takes advantage of the human eye's tendency to "mix" two colors in close proximity to one another.

- Original: Colors = $2^{24}$
- No dithering: Colors = $2^8$
- With dithering: Colors = $2^8$
Dithering takes advantage of the human eye's tendency to "mix" two colors in close proximity to one another.

How could we do this?
How Could We Do This?

- Deterministic Thresholding
- Random Thresholding
- Threshold Patterns
  - Dithering Matrices
- Diffusion
Deterministic Thresholding

Random Thresholding
Dithering Matrices

How do we select a good set of patterns? Regular patterns create some artifacts. Example of good 3x3 dithering matrix.
Floyd-Steinberg Error Diffusion

- Diffuse the quantization error of a pixel to its neighboring pixels
- Scan in raster order
- At each pixel, draw least error output value
- Add the error fractions into adjacent, unwritten pixels

- If a number of pixels have been rounded downwards, it becomes more likely that the next pixel is rounded upwards

```c
for each y
    for each x
        oldpixel := pixel[x][y]
        newpixel := find_closest_palette_color(oldpixel)
        pixel[x][y] := newpixel
        quant_error := oldpixel - newpixel
        pixel[x][y+1] := pixel[x][y+1] + 7/16 * quant_error
        pixel[x-1][y+1] := pixel[x-1][y+1] + 3/16 * quant_error
        pixel[x][y+1] := pixel[x][y+1] + 5/16 * quant_error
        pixel[x+1][y+1] := pixel[x+1][y+1] + 1/16 * quant_error
```
Floyd-Steinberg Error Diffusion
Floyd-Steinberg Error Diffusion

Enhances edges
Retains high frequency
Some checkerboard

How could this be improved?

Color Dithering

• Example: 8 bit framebuffer
  – Set color map by dividing 8 bits into 3,3,2 for RGB
  – Blue is deemphasized because we see it less well

• Dither RGB separately
  – Works well with Floyd-Steinberg

• Generally looks good
Basic Animation
Traditional Cel Animation

• Each frame is drawn by hand

• Film runs at 24 frames per second (fps)
  – That’s 1440 pictures to draw per minute

• Artistic issues:
  – Artistic vision has to be converted into a sequence of still frames
  – Not enough to get the stills right--must look right at full speed
    » Hard to “see” the motion given the stills
    » Hard to “see” the motion at the wrong frame rate
Traditional Animation: The Process

- **Key Frames**
  - Draw a few important frames in pencil
    - beginning of jump, end of jump and a frame in the air

- **Inbetweens**
  - Draw the rest of the frames

- **Painting**
  - Redraw onto clear sheet of plastic called a *cel*, color them in

- Use one layer for background, one for object
  - Draw each separately
  - Stack them together on a copy stand
  - Transfer onto film by taking a photograph of the stack

- Can have multiple animators working simultaneously on different layers, avoid re-drawing and flickering
Example
Principles of Traditional Animation
[Lasseter, SIGGRAPH 1987]

• Stylistic conventions followed by Disney’s animators and others

• From experience built up over many years
  – Squash and stretch -- use distortions to convey flexibility
  – Timing -- speed conveys mass, personality
  – Anticipation -- prepare the audience for an action
  – Followthrough and overlapping action -- continuity with next action
  – Slow in and out -- speed of transitions conveys subtleties
  – Arcs -- motion is usually curved
  – Exaggeration -- emphasize emotional content
  – Secondary Action -- motion occurring as a consequence
  – Appeal -- audience must enjoy watching it
Squash and Stretch

Use distortions to convey flexibility
Principles of Traditional Animation

SQUASHED & STRETCHED & TWISTED
DEJECTED  JOY  TANTRUM  CURIOUS

COCKY  LAUGHTER  BELLIQUEMENT  MORE LAUGHTER

The famous half-filled flour sack, guide to maintaining volume in any animatable shape, and proof that attitudes can be achieved with the simplest of shapes.
Squash and Stretch
Use distortions to convey flexibility

Defines the rigidity of the material

Gives the sense that the object is made out of a soft, pliable material.

Elongating the drawings before and after the bounce increases the sense of speed, makes it easier to follow and gives more snap to the action.
Example

The ball on the left moves at a constant speed with no squash/stretch.
The ball in the center does slow in and out with a squash/stretch.
The ball on the right moves at a constant speed with squash/stretch.
Timing & Motion

Speed conveys mass, personality

A heavier object takes a greater force and a longer time to accelerate and decelerate.

A larger object moves more slowly than a smaller object and has greater inertia.

Motion also can give the illusion of weight.
For example, consider a ball hitting a box.

http://www.siggraph.org/education/materials/HyperGraph(animation/character_animate/character_animate/principles/timig.htm)
Anticipation
Prepare the audience for an action

Don’t surprise the audience
Direct their attention to what’s important
Follow Through and Overlapping Action

The termination of an action and establishing its relationship to the next action

Audience likes to see resolution of action
Discontinuities are unsettling
Secondary Action

Motion occurring as a consequence
Uncanny Valley
Radiosity
Doing it Right

The real solution is to solve simultaneously for incoming and outgoing light at all surface points. This is a massive integral equation.
Key Idea

• Model **diffuse** interaction only!
Far Too Many Points

- Concentrate on patches instead.
- Want to have as few as possible.
Our new equation gives the radiosity ($B$) of a single patch, so to specify the radiosity of all $n$ patches we need $n$ radiosity equations, one for each patch.

Known values:
- $E$ (given), $R$ (given), $F$ (computable)

Unknown: $B$

$n$ equations, $n$ unknowns
Linear System

Restate as a matrix equation...and solve

\[
\begin{bmatrix}
1 - R_1 F_{11} & - R_1 F_{12} & \cdots & R_1 F_{1n} \\
- R_2 F_{21} & 1 - R_2 F_{22} & \cdots & R_2 F_{2n} \\
\cdots & \cdots & \cdots & \cdots \\
- R_n F_{n1} & R_n F_{n2} & \cdots & 1 - R_n F_{nn}
\end{bmatrix}
\begin{bmatrix}
B_1 \\
B_2 \\
\vdots \\
B_n
\end{bmatrix}
= 
\begin{bmatrix}
E_1 \\
E_2 \\
\vdots \\
E_n
\end{bmatrix}
\]

Each of our \( n \) linear equations contains \( n \) double integrals, one for each form factor.
The Radiosity “Pipeline”

1. Input Scene Geometry
2. Meshing (division into patches)
3. Form Factor Calculations
4. Input Reflectance/Emission Factors
5. Solve Radiosity Equation

Texture Geometry with Radiosity Solution
Antialiasing (on hemicube)
Classical, Resolution 300
Classical, Resolution 1200
Classical, Resolution 2500
Classical, Resolution 2500, Interpolated
Supersampled, Res 100, Interpolated
Adaptive Subdivision

Introduce a patch substructure—divide each patch into smaller elements.

Keep distinction between patches and elements in order to avoid efficiency problems.
Adaptive Subdivision

Determine light transport one-way from patches onto elements, not analyzing element-to-element interaction

$O(mn)$ for $m$ elements and $n$ patches. More expensive than the original $n^2$ approach, since $m \gg n$, but much better than $O(m^2)$. 
Adaptive Subdivision

Subdivide elements adaptively:

Begin with elements identical to patches.
Determine radiosity of an element, then compare to neighbors to obtain an error value. If within some error threshold, assign constant radiosity (or optionally interpolate).
Otherwise, subdivide the element and recurse until the error threshold or a minimum element size is reached.
Adaptive Subdivision

Results in very smooth-looking results for a relatively small amount of extra work.

Shadows, areas near lights, and edges in general look much better.

Not an idea specific to radiosity! Adaptive subdivision is a general tool used in many areas of graphics and other fields as well.
Adaptive Subdivision Examples

http://www.acm.org/jgt/papers/TeleaVanOverveld97/
http://aig.cs.man.ac.uk/gallery/vrad.html
Problems

- Blocky (Aliasing)
  ➞ More Patches
  ➞ Adaptive Subdivision

- No Specular Effects
  ➞ 2-pass Algorithms
Problems

• Blocky (Aliasing)
  ➟ Adaptive Subdivision

• No Specular Effects
  ➟ 2-pass Algorithms
Patch Size

Wireframe

300

1200

2500

2500 (interpolated)

100 (interpolated, supersampled)
Adaptive Subdivision

Subdivide elements adaptively:

Begin with elements identical to patches.

Determine radiosity of an element, then compare to neighbors to obtain an error value. If within some error threshold, assign constant radiosity (or optionally interpolate).

Otherwise, subdivide the element and recurse until the error threshold or a minimum element size is reached.
Adaptive Subdivision Examples

http://www.acm.org/jgt/papers/TeleaVanOverveld97/
Problems

- Blocky (Aliasing) ➞ Adaptive Subdivision
- No Specular Effects ➞ 2-pass Algorithms
Problems

- Blocky (Aliasing)
  ➟ Adaptive Subdivision

- No Specular Effects
  ➟ 2-pass Algorithms
Radiosity vs. Raytracing
Can we inject specular effects into radiosity?
2-pass Algorithms

1st Pass: Radiosity

2nd Pass: Raytracing (using Radiosity result)

Result

http://www.cg.tuwien.ac.at/research/rendering/rays-radio/
Example
Problems

- Blocky (Aliasing)
- No Specular Effects

\[\rightarrow\] Adaptive Subdivision

\[\rightarrow\] 2-pass Algorithms
Problems

- Blocky (Aliasing)
  ➞ Adaptive Subdivision

- No Specular Effects
  ➞ 2-pass Algorithms
Overview

Keyframing
Figure 10.5 Inbetweening with linear interpolation. Linear interpolation creates inbetween frames at equal intervals along straight lines. The ball moves at a constant speed. Ticks indicate the locations of inbetween frames at regular time intervals (determined by the number of frames per second chosen by the user).
How Do You Interpolate Between Keys?

**AUTO**
- $y'_{i-1}$
- $y_{i-1}$
- $t_{i-1}$
- $y_i$
- $t_i$
- $y_{i+1}$
- $t_{i+1}$

**MANUAL**
- $y'_{i-1}$
- $y_{i-1}$
- $t_{i-1}$
- $y'_{i+1}$
- $y_{i+1}$
- $t_{i+1}$

**SPLIT**
- $y'_{i-1}$
- $y_{i-1}$
- spline$(i-1,i)$
- $y'_{i+1}$
- $y_{i+1}$
- spline$(i,i+1)$
Keyframing Basics

**Figure 10.9** Inbetweening with nonlinear interpolation. Nonlinear interpolation can create equally spaced inbetween frames along curved paths. The ball still moves at a constant speed. (Note that the three keyframes used here and in Fig. 10.10 are the same as in Fig. 10.4.)
**Figure 10.10** Inbetweening with nonlinear interpolation and easing. The ball changes speed as it approaches and leaves keyframes, so the dots indicating calculations made at equal time intervals are no longer equidistant along the path.
Keyframe basics

• For each variable, specify its value at the “important” frames. Not all variables need agree about which frames are important.

• Hence, key values rather than key frames.

• Create path for each parameter by interpolating key values.
Problems with Interpolation

- Splines don’t always do the right thing

- Classic problems
  - Important constraints may break between keyframes
    - feet sink through the floor
    - hands pass through walls
  - 3D rotations
    - Euler angles don’t always interpolate in a natural way

- Classic solutions:
  - More keyframes!
  - Quaternions help fix rotation problems
Overview

Hair
Real Hair: Curly

Short curly hair
Real Hair: Straight

Long smooth hair
Real Hair

- Typical human head has 150k-200k individual strands.
- Dynamics not well understood.
- Subject still open to debate.
Recall...

Cloth and Fur Energy Functions

Michael Kass
Limp hair: Just a set of springs.
Hair Model

Add body: Angular Springs

\[ E = \frac{1}{2} \sum_{i} k_i \theta_i^2 \]
Hair Model

Alternative: More Linear Springs

Difficulty:
Each spring constant affects both bending and stretching
Problems

The linear spring model is very simple but has several problems:

- Not length preserving.
- No torsion forces (twist).
What is cloth?

Two basic types...

Woven

Knit
What is cloth?

- 2 basic types: woven and knit
- We’ll restrict to woven
  - Warp vs. weft

Figure 1.8. The weaving process.

House, Breen [2000]
Warp and Weft

Cloth and Fur Energy Functions

Michael Kass
Stretch (Continuum Version)

\( (u, v) \) \hspace{1cm} \bar{x}

\[ S_u = \left\| \frac{\partial \bar{x}}{\partial u} \right\| - 1 \]

\[ E = \frac{1}{2} k \int (S_u^2 + S_v^2) du dv \]
Shear (Continuum Version)

\[ \theta = \cos^{-1} \left( \frac{\partial \hat{x}}{\partial u} \cdot \frac{\partial \hat{x}}{\partial v} \right) \]

\[ E = \frac{1}{2} k \int \theta^2 du dv \]
Bend (Continuum Version)

\[ E = \frac{1}{2} k \int (\kappa_u^2 + \kappa_v^2) du \, dv \]
Resitence To...

- Stretching
- Shearing
- Bending
Basic Model
Warp Strings
Weft Springs
Shear Springs
Bend Springs
Parameters

- Given stretch, shear, and bending constants...
- How would you make a wrinkly t-shirt, thick cloth, or non-uniform cloth?
Springs vs. Constraints

Before Simulation

Only Springs

Stretch Constraints

Stretch+Shear Constraints

Source: Xavier Provot
Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior
What is a PDE?

- Ordinary Differential Equation:

\[ \dot{q} = f(q) \]

- Partial Differential Equation:

\[ \dot{q}(x, y) = f\left(q, \frac{\partial q}{\partial x}, \frac{\partial q}{\partial y}, \ldots \right) \]
class Function1D {
    public:
        int len() const;
        double set(int index, double value);
        double get(int index) const;
        double getDeriv(int index) const;
};
Compression
Image Sizes

- 1024*1024 at 24 bits uses 3 MB

- Encyclopedia Britannica at 300 pixels/inch and 1 bit/pixel requires 25 gigabytes (25K pages)

- 90 minute movie at 640x480, 24 bits per pixel, 24 frames per second requires 120 gigabytes

- Applications: HDTV, DVD, satellite image transmission, medical image processing, fax, ...
Types of Compression

- Coding Redundancy
  - Huffman Coding (lossless)
- Spatial Coherence
  - Run Length Encoding (lossless)
- Psycho visual
  - JPEG Encoding (lossy)
Types of Compression

- Coding Redundancy
  - Huffman Coding (lossless)

- Spatial Coherence
  - Run Length Encoding (lossless)

- Psycho visual
  - JPEG Encoding (lossy)
Suppose we have the following 4 colors:

```
00  01  10  11
```

As used in this image:

```
00  00  10  10  10  10
10  10  10  10  01  10
10  10  11  10  11  00
```

Binary String (36 bits):
```
0000101010101010
1010100110101010
11101100
```

Switch to this encoding:

```
000  001  1  010
```

Which is equivalent to:

```
000  000  1  1  1  1  1
1  1  1  1  1  001  1
1  1  010  1  010  000
```

Binary String (28 bits):
```
000111111111000
11110101010000
```
Huffman Coding

Suppose we have the following 4 colors:

00 01 10 11

As used in this image:

00 00 10 10 10 10 10
10 10 10 10 10 10 10
00 10 10
10 10
10

Switch: 000 001 1 010
Which: 000 001 1 010

Huffman Codes provide the optimal answer to encoding such a representation.

Binary String (28 bits):
00011111111000
11110101010000
Exploiting Coding Redundancy

- Not limited to images (text, other digital info)
- Exploit nonuniform probabilities of symbols
- Entropy as measure of information content
  - \( H = -\sum_i \text{Prob}(s_i) \log_2 (\text{Prob}(s_i)) \)
  - Low entropy \( \rightarrow \) non uniform probability
  - High entropy \( \rightarrow \) uniform probability
  - If source is independent random variable need \( H \) bits
Types of Compression

- Coding Redundancy
  - Huffman Coding (lossless)

- Spatial Coherence
  - Run Length Encoding (lossless)

- Psycho visual
  - JPEG Encoding (lossy)
Types of Compression

- Coding Redundancy
  - Huffman Coding (lossless)

- Spatial Coherence
  - Run Length Encoding (lossless)

- Psycho visual
  - JPEG Encoding (lossy)
Run Length Encoding

Same Image As Before:

```
00 00 10 10 10 10
10 10 10 10 01 10
10 10 11 10 11 00
```

Scan Convert:

```
00 00 10 10 10 10 10 10 10 10 10 10 01 10 10 10 10 11 10 11 00
```

Encode:

```
2× 00 8× 10 1× 01 3× 10 1× 11 1× 10 1× 11 1× 00
```
Run Length Encoding

Same Image As Before:

00 00 10 10 10 10
10 10 10 10 10 10
10 10 10 10 10 10
10 10 10 10 10 10
10 10 10 10 10 10
00 00

Scan Convert:

Encode:

2 × 00 8 × 10 1 × 01 3 × 10 1 × 11 1 × 10 1 × 11 1 × 00

Related Ideas

• **Quadtrees:** Recursively subdivide until cells are constant color.

• **Region Encoding:** Represent boundary curves of constant-color regions.
Types of Compression

- Coding Redundancy
  - Huffman Coding (lossless)

- Spatial Coherence
  - Run Length Encoding (lossless)

- Psycho visual
  - JPEG Encoding (lossy)
Types of Compression

- Coding Redundancy
  - Huffman Coding (lossless)

- Spatial Coherence
  - Run Length Encoding (lossless)

- Psycho visual
  - JPEG Encoding (lossy)
JPEG Compression

Divide image into 8x8 blocks.
JPEG Compression

• Express each block as a linear combination of 8x8 basis blocks made of cosines.
• This is called the *discrete cosine transform*. 
Key Insight!

- Upper left blocks have higher values than lower right? (*They are more important.*)
- Why?

[Image showing discrete cosine bases and coefficients]

How can we exploit this insight?
Scaled Coefficients

\[ \begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} \div \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]

\[
\text{round} \left( \frac{-415}{16} \right) = \text{round} \left( -25.9375 \right) = -26
\]

• What can we see about the quantization matrix?
• How can we compress the scaled coefficients?

Answer:
Run Length + Huffman Coding
Types of Compression

- Coding Redundancy
  - Huffman Coding (lossless)

- Spatial Coherence
  - Run Length Encoding (lossless)

- Psycho visual
  - JPEG Encoding (lossy)
Image Based Rendering
Computer Graphics

Geometry + Material attributes
Computer Vision

Output

Model

Real Scene

Real Cameras
Combined

Output

Image

Synthetic Camera

Model

Real Scene

Real Cameras
But, vision technology falls short

Hard to re-create much of the complex geometry and lighting effects found in real world
… and so does graphics.

Output
Image
Synthetic Camera
Model
Real Scene
Real Cameras

Hard to render world illumination
What is Image-Based Rendering?

All we usually care about in rendering is generating images from new viewpoints.
Image-Based Rendering

Geometry based
  • Geometry + Material attributes

Skip traditional modeling/rendering process

Image based rendering seeks to replace geometry and surface properties with images
Quicktime VR

Skip traditional modeling/rendering process

Capture environment maps from given locations

Look around from a fixed point

Show Demo
Lightfields and Lumigraphs
Modeling light

Capture flow of light in region of environment

Described by plenoptic function
Plenoptic Function

Describes the intensity of light:
- passing through a given point, \( x \)
- in a given direction, \( (\theta, \phi) \)

5D
- 3D position
- 2D direction
All Rays

Plenoptic Function:
• all possible rays
Plenoptic Function

Many image-based rendering approaches can be cast as sampling from and reconstructing the plenoptic function

Note, function is generally constant along segments of a line (assuming vacuum)