

# CSCI-4530/6530 Advanced Computer Graphics

<http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S10/>

Barb Cutler  
cutler@cs.rpi.edu  
MRC 331A

1

## Luxo Jr.



Pixar Animation Studios, 1986

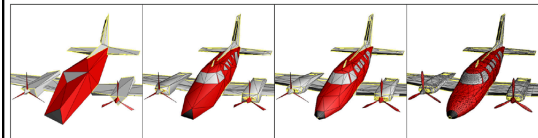
2

## Topics for the Semester

- Meshes
  - representation
  - simplification
  - subdivision surfaces
  - generation
  - volumetric modeling
- Simulation
  - particle systems
  - rigid body, deformation, cloth, wind/water flows
  - collision detection
  - weathering
- Rendering
  - ray tracing
  - appearance models
  - shadows
  - local vs. global illumination
  - radiosity, photon mapping, subsurface scattering, etc.
- procedural modeling
- texture synthesis
- hardware & more ...

3

## Mesh Simplification

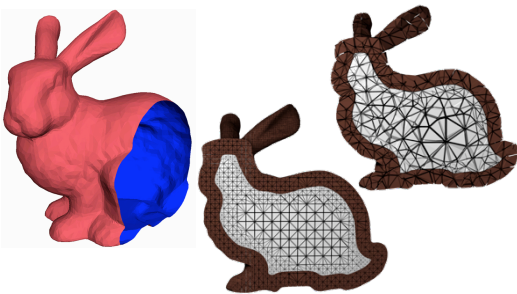


(a) Base mesh  $M^1$  (150 faces) (b) Mesh  $M^2$  (500 faces) (c) Mesh  $M^3$  (1,000 faces) (d) Original  $M=M^4$  (13,546 faces)

Hoppe "Progressive Meshes" SIGGRAPH 1996

4

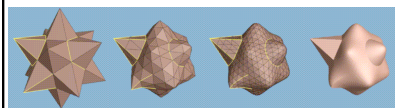
## Mesh Generation & Volumetric Modeling



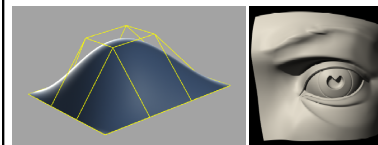
Cutler et al., "Simplification and Improvement of Tetrahedral Models for Simulation" 2004

5

## Modeling – Subdivision Surfaces



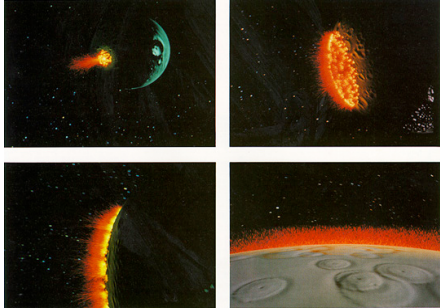
Hoppe et al., "Piecewise Smooth Surface Reconstruction" 1994



Geri's Game  
Pixar 1997

6

## Particle Systems

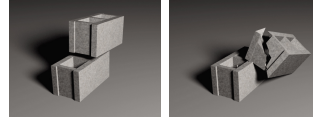
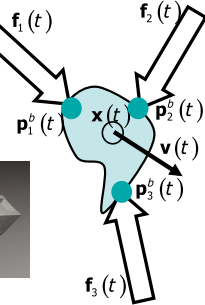


Star Trek: The Wrath of Khan 1982

7

## Physical Simulation

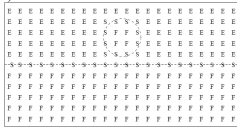
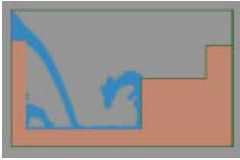
- Rigid Body Dynamics
- Collision Detection
- Fracture
- Deformation



Müller et al., "Stable Real-Time Deformations" 2002

8

## Fluid Dynamics



Foster & Matusik, 1996

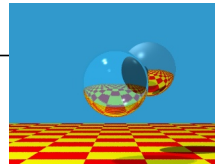


"Visual Simulation of Smoke"  
Fedkiw, Stam & Jensen  
SIGGRAPH 2001

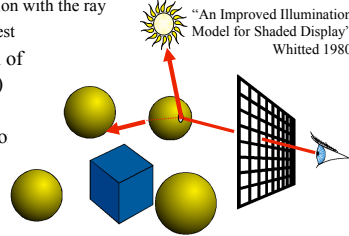
9

## Ray Casting/Tracing

- For every pixel  
construct a ray from the eye
  - For every object in the scene
    - Find intersection with the ray
    - Keep the closest
- Shade (interaction of light and material)
- Secondary rays (shadows, reflectio refraction)



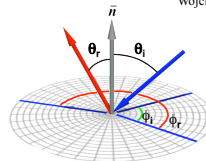
"An Improved Illumination Model for Shaded Display"  
Whitted 1980



## Appearance Models



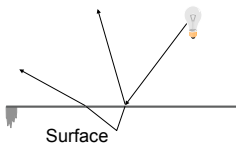
Wojciech Matusik



Henrik Wann Jensen

12

## Subsurface Scattering



Jensen et al., "A Practical Model for Subsurface Light Transport" 2001

11

## Syllabus & Course Website

<http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S10/>

- Which version should I register for?
  - CSCI 6530
    - 3 units of graduate credit
  - CSCI 4530
    - 4 units of undergraduate creditsame lectures, assignments, quizzes, & grading criteria
- Other Questions?

13

## Participation/Laptops in Class Policy

- Use of laptops for reference during paper discussion and general note-taking is encouraged.
- **Participation is 15% of your grade:** If your focus is mostly on your laptop *and* you rarely speak up in class, you will get a zero for participation

14

## Introductions – Who are you?

- name
- year/degree
- graphics background (if any)
- research/job interests
- why you are taking this class
- something fun, interesting, or unusual about yourself

15

## Outline

- Course Overview
- **Classes of Transformations**
- Representing Transformations
- Combining Transformations
- Orthographic & Perspective Projections
- Example: Iterated Function Systems (IFS)
- OpenGL Basics

16

## What is a Transformation?

- Maps points  $(x, y)$  in one coordinate system to points  $(x', y')$  in another coordinate system

$$x' = ax + by + c$$

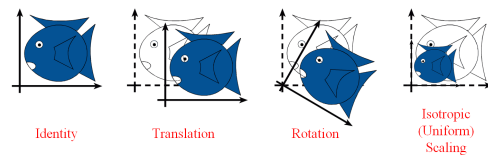
$$y' = dx + ey + f$$

- For example, Iterated Function System (IFS):



17

## Simple Transformations



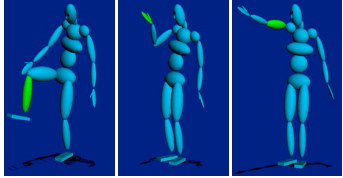
- Can be combined
- Are these operations invertible?

*Yes, except scale = 0*

18

## Transformations are used to:

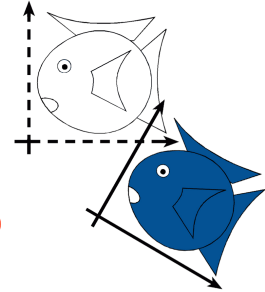
- Position objects in a scene
- Change the shape of objects
- Create multiple copies of objects
- Projection for virtual cameras
- Describe animations



19

## Rigid-Body / Euclidean Transforms

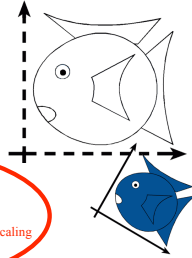
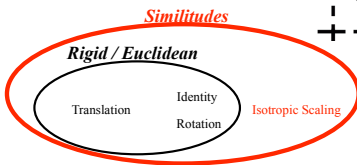
- Preserves distances
- Preserves angles



20

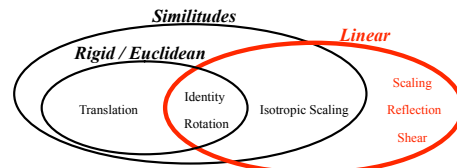
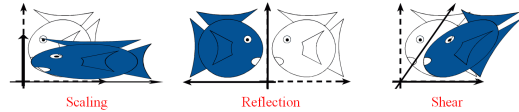
## Similitudes / Similarity Transforms

- Preserves angles



21

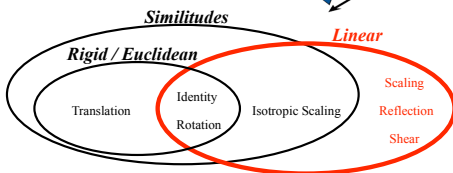
## Linear Transformations



22

## Linear Transformations

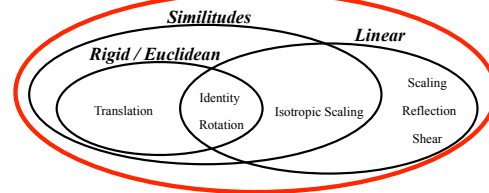
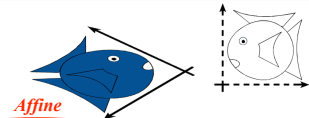
- $L(p + q) = L(p) + L(q)$
- $L(ap) = a L(p)$



23

## Affine Transformations

- preserves parallel lines



24

## Projective Transformations

- preserves lines

25

## General (Free-Form) Transformation

- Does not preserve lines
- Not as pervasive, computationally more involved

Sederberg and Parry, Siggraph 1986

26

## Outline

- Course Overview
- Classes of Transformations
- **Representing Transformations**
- Combining Transformations
- Orthographic & Perspective Projections
- Example: Iterated Function Systems (IFS)
- OpenGL Basics

27

## How are Transforms Represented?

$$x' = ax + by + c$$

$$y' = dx + ey + f$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$$

$$p' = Mp + t$$

28

## Homogeneous Coordinates

- Add an extra dimension
  - in 2D, we use 3 x 3 matrices
  - In 3D, we use 4 x 4 matrices
- Each point has an extra value, w

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

$$p' = Mp$$

29

## Translation in homogeneous coordinates

$$x' = ax + by + c$$

$$y' = dx + ey + f$$

Affine formulation	Homogeneous formulation
$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$	$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
$p' = Mp + t$	$p' = Mp$

30

## Homogeneous Coordinates

- Most of the time  $w = 1$ , and we can ignore it

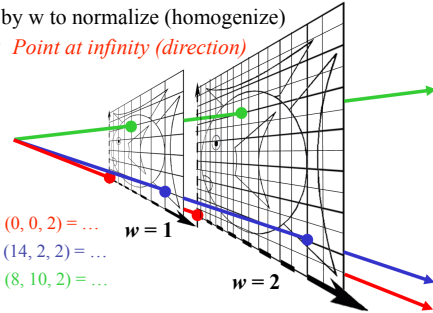
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- If we multiply a homogeneous coordinate by an *affine matrix*,  $w$  is unchanged

31

## Homogeneous Visualization

- Divide by  $w$  to normalize (homogenize)
- $w = 0$ ? *Point at infinity (direction)*

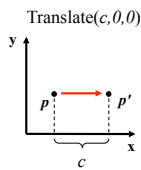


$$\begin{aligned} (0, 0, 1) &= (0, 0, 2) = \dots \\ (7, 1, 1) &= (14, 2, 2) = \dots \\ (4, 5, 1) &= (8, 10, 2) = \dots \end{aligned}$$

32

## Translate ( $t_x, t_y, t_z$ )

- Why bother with the extra dimension?  
Because now translations can be encoded in the matrix!

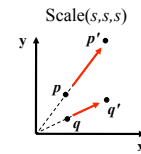


$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

33

## Scale ( $s_x, s_y, s_z$ )

- Isotropic (uniform) scaling:  $s_x = s_y = s_z$

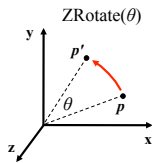


$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

34

## Rotation

- About z axis

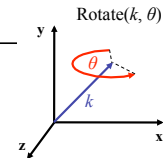


$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

35

## Rotation

- About  $(k_x, k_y, k_z)$ , a unit vector on an arbitrary axis (Rodrigues Formula)



$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} k_x k_x (1-c) + c & k_x k_y (1-c) - k_z s & k_x k_z (1-c) + k_y s & 0 \\ k_y k_x (1-c) + k_z s & k_y k_y (1-c) + c & k_y k_z (1-c) - k_x s & 0 \\ k_z k_x (1-c) - k_y s & k_z k_y (1-c) - k_x s & k_z k_z (1-c) + c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

where  $c = \cos \theta$  &  $s = \sin \theta$

36

## Storage

- Often,  $w$  is not stored (always 1)
- Needs careful handling of direction vs. point
  - Mathematically, the simplest is to encode directions with  $w = 0$
  - In terms of storage, using a 3-component array for both direction and points is more efficient
  - Which requires to have special operation routines for points vs. directions

37

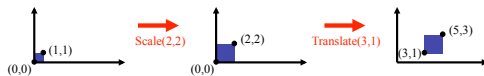
## Outline

- Course Overview
- Classes of Transformations
- Representing Transformations
- **Combining Transformations**
- Orthographic & Perspective Projections
- Example: Iterated Function Systems (IFS)
- OpenGL Basics

38

## How are transforms combined?

Scale then Translate



Use matrix multiplication:  $p' = T(S p) = TS p$

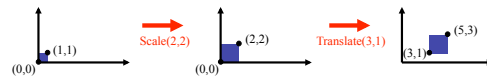
$$TS = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Caution: matrix multiplication is NOT commutative!

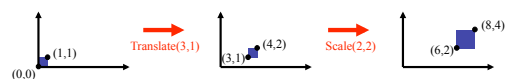
39

## Non-commutative Composition

Scale then Translate:  $p' = T(S p) = TS p$



Translate then Scale:  $p' = S(T p) = ST p$



40

## Non-commutative Composition

Scale then Translate:  $p' = T(S p) = TS p$

$$TS = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Translate then Scale:  $p' = S(T p) = ST p$

$$ST = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 6 \\ 0 & 2 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$

41

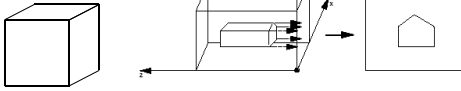
## Outline

- Course Overview
- Classes of Transformations
- Representing Transformations
- Combining Transformations
- **Orthographic & Perspective Projections**
- Example: Iterated Function Systems (IFS)
- OpenGL Basics

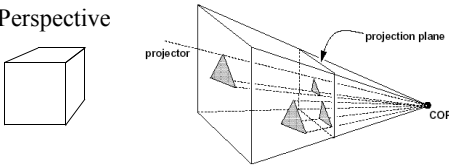
42

### Orthographic vs. Perspective

- Orthographic



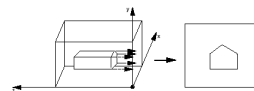
- Perspective



43

### Simple Orthographic Projection

- Project all points along the z axis to the z = 0 plane



$$\begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

44

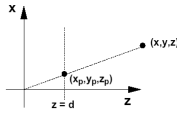
### Simple Perspective Projection

- Project all points along the z axis to the z = d plane, eyepoint at the origin:

$$x_p = \frac{d \cdot x}{z} = \frac{x}{z/d}$$

$$y_p = \frac{d \cdot y}{z} = \frac{y}{z/d}$$

$$z_p = d$$



homogenize

$$\begin{pmatrix} x * d / z \\ y * d / z \\ d \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z/d \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

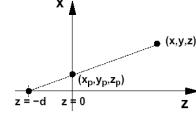
45

### Alternate Perspective Projection

- Project all points along the z axis to the z = 0 plane, eyepoint at the (0,0,-d):

$$x_p = \frac{d \cdot x}{z + d} = \frac{x}{(z/d) + 1}$$

$$y_p = \frac{d \cdot y}{z + d} = \frac{y}{(z/d) + 1}$$



homogenize

$$\begin{pmatrix} x * d / (z + d) \\ y * d / (z + d) \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \\ (z + d)/d \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

46

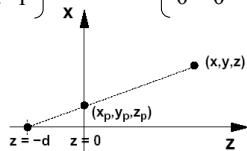
### In the limit, as d → ∞

this perspective projection matrix...

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix}$$

...is simply an orthographic projection

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



47

### Outline

- Course Overview
- Classes of Transformations
- Representing Transformations
- Combining Transformations
- Orthographic & Perspective Projections
- Example: Iterated Function Systems (IFS)
- OpenGL Basics

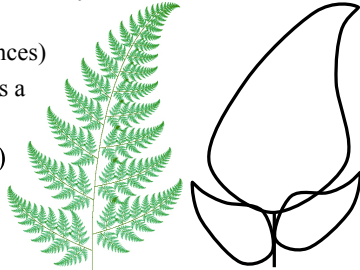
48



## Iterated Function Systems (IFS)

- Capture self-similarity
- Contraction (reduce distances)
- An attractor is a fixed point

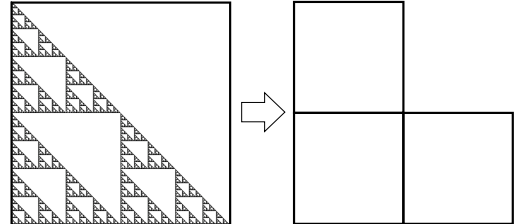
$$A = \bigcup f_i(A)$$



49

## Example: Sierpinski Triangle

- Described by a set of  $n$  affine transformations
- In this case,  $n = 3$ 
  - translate & scale by 0.5



50

## Example: Sierpinski Triangle

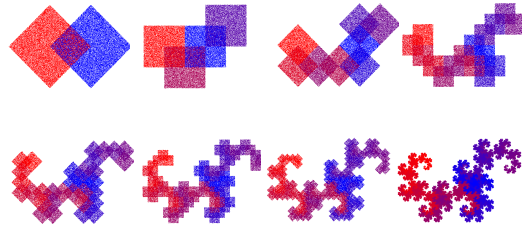
```
for "lots" of random input points (x0, y0)
  for j=0 to num_iters
    randomly pick one of the transformations
    (xk+1, yk+1) = fi (xk, yk)
  display (xk, yk)
```



Increasing the number of iterations

51

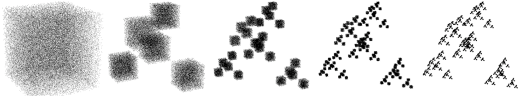
## Another IFS: The Dragon



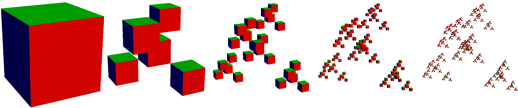
52

## 3D IFS in OpenGL

GL\_POINTS



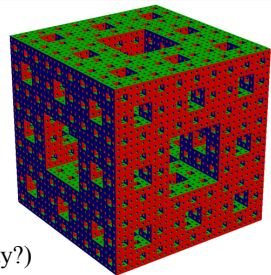
GL\_QUADS



53

## Assignment 0: OpenGL Warmup

- Get familiar with:
  - C++ environment
  - OpenGL
  - Transformations
  - simple Vector & Matrix classes



- Have Fun!
- Due ASAP (start today?)
- ¼ of the points of the other HWs  
(*but you should still do it and submit it!*)

54

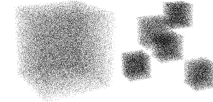
## Outline

- Course Overview
- Classes of Transformations
- Representing Transformations
- Combining Transformations
- Orthographic & Perspective Projections
- Example: Iterated Function Systems (IFS)
- **OpenGL Basics**

55

## OpenGL Basics: GL\_POINTS

```
glDisable(GL_LIGHTING);
glBegin(GL_POINTS);
glColor3f(0.0,0.0,0.0);
glVertex3f(...);
glEnd();
```

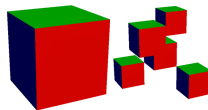


- **lighting should be disabled...**

56

## OpenGL Basics: GL\_QUADS

```
glEnable(GL_LIGHTING);
glBegin(GL_QUADS);
glNormal3f(...);
glColor3f(1.0,0.0,0.0);
glVertex3f(...);
glVertex3f(...);
glVertex3f(...);
glVertex3f(...);
glEnd();
```



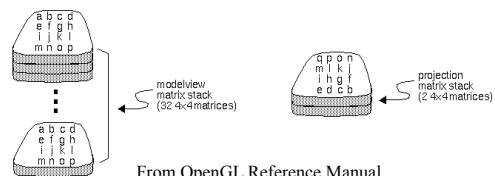
- **lighting should be enabled...**
- **an appropriate normal should be specified**

57

## OpenGL Basics: Transformations

- Useful commands:

```
glMatrixMode(GL_MODELVIEW);
glPushMatrix();
glPopMatrix();
glMultMatrixf(...);
```



58

## Questions?

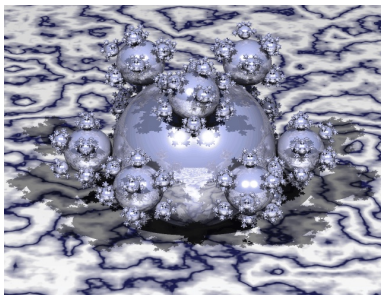
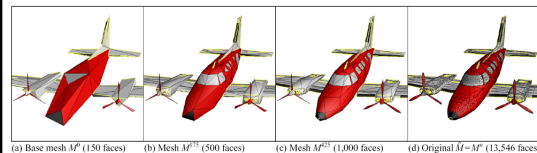


Image by Henrik Wann Jensen

59

## For Next Time:

- Read Hugues Hoppe "Progressive Meshes" SIGGRAPH 1996
- Post a comment or question on the course WebCT/LMS discussion by 10am on Friday 1/29



(a) Base mesh  $M^0$  (150 faces) (b) Mesh  $M^1$  (500 faces) (c) Mesh  $M^2$  (1,000 faces) (d) Original  $M=M^3$  (13,546 faces)

60