CSCI-4530/6530 Advanced Computer Graphics

http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S13/

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

1

Luxo Jr.

Pixar Animation Studios, 1986

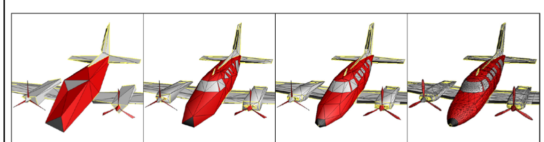
2

Topics for the Semester

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

3

Mesh Simplification

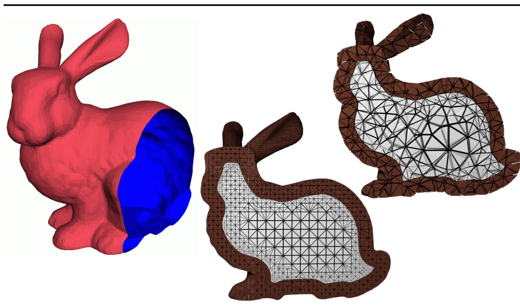


(a) Base mesh M^0 (150 faces) (b) Mesh M^{175} (500 faces) (c) Mesh M^{425} (1,000 faces) (d) Original $\hat{M}=M^n$ (13,546 faces)

Hoppe "Progressive Meshes" SIGGRAPH 1996

4

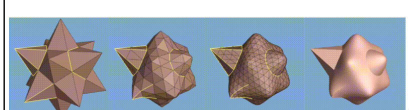
Mesh Generation & Volumetric Modeling



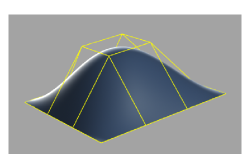
Cutler et al., "Simplification and Improvement of Tetrahedral Models for Simulation" 2004

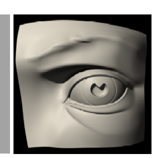
5

Modeling – Subdivision Surfaces



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

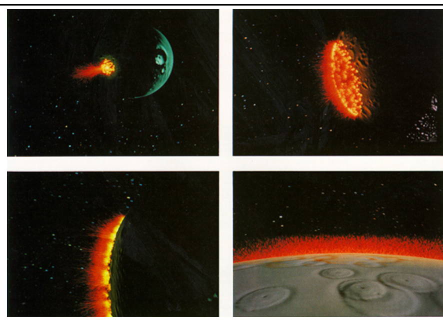




Geri's Game
Pixar 1997

6

Particle Systems

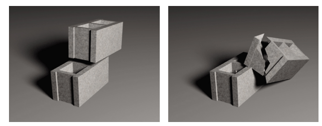


Star Trek: The Wrath of Khan 1982

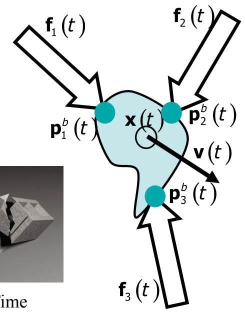
7

Physical Simulation

- Rigid Body Dynamics
- Collision Detection
- Fracture
- Deformation

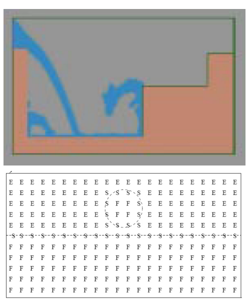


Müller et al., "Stable Real-Time Deformations" 2002



8

Fluid Dynamics



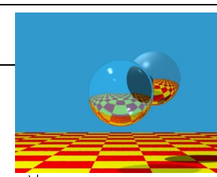
Foster & Mataxas, 1996

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

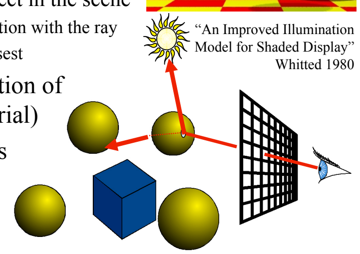
9

Ray Casting/Tracing

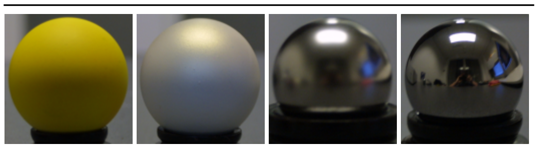
- For every pixel construct a ray from the eye
  - For every object in the scene
    - Find intersection with the ray
    - Keep the closest
- Shade (interaction of light and material)
- Secondary rays (shadows, reflection, refraction)



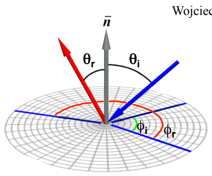
"An Improved Illumination Model for Shaded Display"
Whitted 1980

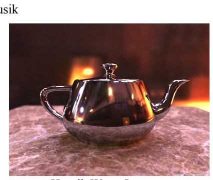


Appearance Models



Wojciech Matusik

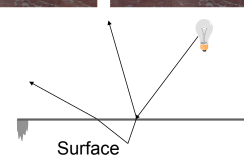




Henrik Wann Jensen

11

Subsurface Scattering



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

12

Syllabus & Course Website

`http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S13/`

- Which version should I register for?
  - CSCI 6530 : 3 units of graduate credit, class ends at 3:20
  - CSCI 4530 : 4 units of undergraduate credit, class ends at 3:50

  (same lectures, assignments, quizzes, & grading criteria)
- This is an intensive course aimed at graduate students and undergraduates interested in graphics research, involving significant reading & programming each week. Taking this course in a 5 course overload semester is discouraged.
- Other Questions?

13

Participation/Laptops in Class Policy

- Lecture is intended to be discussion-intensive
- Laptops, tablet computers, smart phones, and other internet-connected devices are not allowed
  - Except during the discussion of the day's assigned paper: students may use their laptop/tablet to view an electronic version of the paper
  - Other exceptions to this policy are negotiable; please see the instructor in office hours.

14

Introductions – Who are you?

- name
- year/degree
- graphics background (if any)
- research/job interests, future plans
- 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

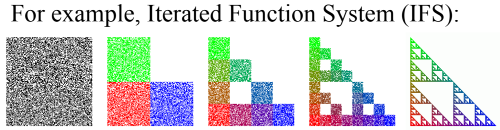
16

What is a Transformation?

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

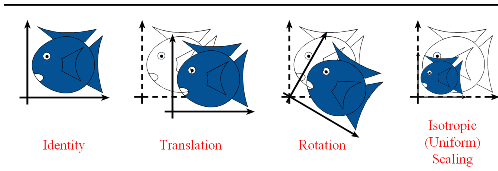
$$x' = ax + by + c$$
$$y' = dx + ey + f$$

- For example, Iterated Function System (IFS):



17

Simple Transformations



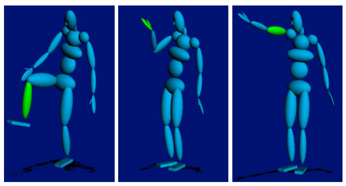
- Can be combined
- Are these operations invertible?

  Yes, except scale = 0

18

Transformations are used to:

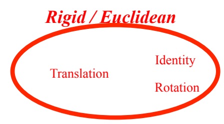
- Position objects in a scene
- Change the shape of objects
- Create multiple copies of objects
- Projection for virtual cameras
- Describe animations

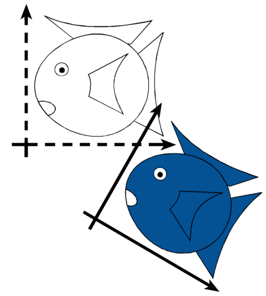


19

Rigid-Body / Euclidean Transforms

- Preserves distances
- Preserves angles

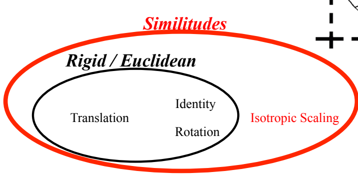


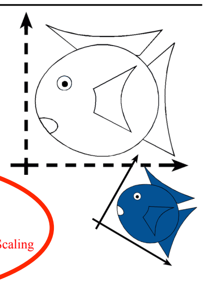


20

Similitudes / Similarity Transforms

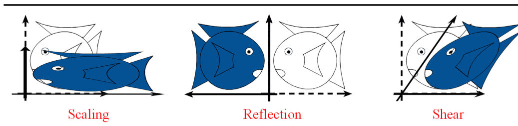
- Preserves angles

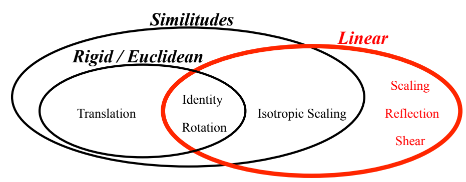




21

Linear Transformations



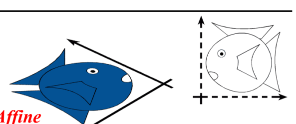


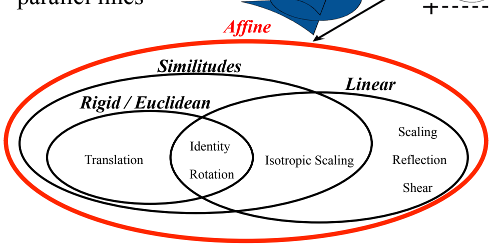
$L(p + q) = L(p) + L(q)$ $\qquad L(ap) = a\, L(p)$

22

Affine Transformations

- preserves parallel lines

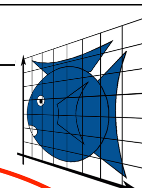


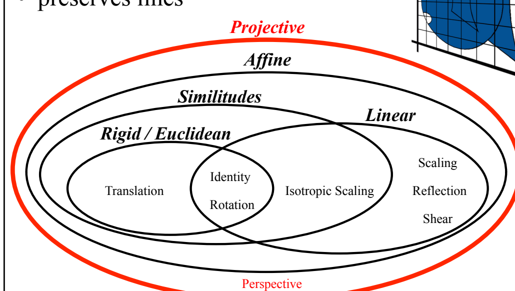


23

Projective Transformations

- preserves lines





24

General (Free-Form) Transformation

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

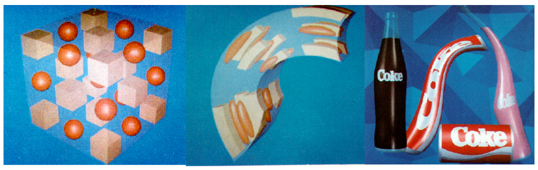


Fig 1. Undeformed Plastic

Fig 2. Deformed Plastic

Sederberg and Parry, Siggraph 1986

25

Outline

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

26

How are Transforms Represented?

$$x' = ax + by + c$$
$$y' = dx + ey + f$$

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

$$p' = Mp + t$$

27

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

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

$$p' = Mp$$

28

Translation in homogeneous coordinates

$$x' = ax + by + c$$
$$y' = dx + ey + f$$

Affine formulation

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

$$p' = Mp + t$$

Homogeneous formulation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$p' = Mp$$

29

Homogeneous Coordinates

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

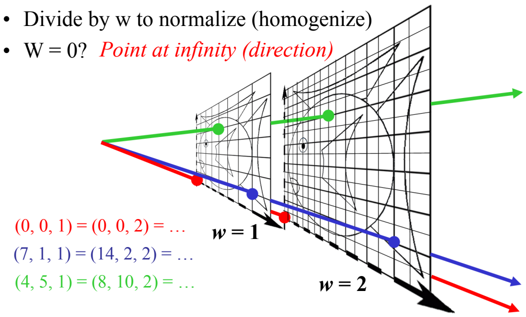
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

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

30

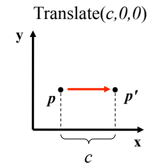
Homogeneous Visualization

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



31

Translate (tx, ty, tz)

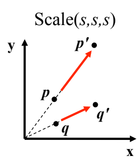


- Why bother with the extra dimension?
  Because now translations can be encoded in the matrix!

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

32

Scale (sx, sy, sz)

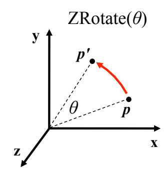


- Isotropic (uniform) scaling: $s_x = s_y = s_z$

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

33

Rotation

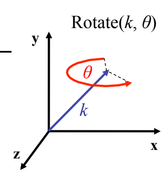


- About z axis

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

34

Rotation



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

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} k_xk_x(1-c)+c & k_zk_x(1-c)-k_zs & k_xk_z(1-c)+k_ys & 0 \\ k_yk_x(1-c)+k_zs & k_zk_x(1-c)+c & k_yk_z(1-c)-k_xs & 0 \\ k_zk_x(1-c)-k_ys & k_zk_x(1-c)-k_xs & k_zk_z(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

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

35

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

36

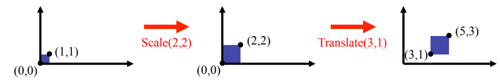
Outline

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

37

How are transforms combined?

Scale then Translate



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

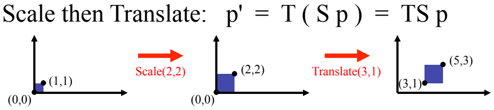
$$TS = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Caution: matrix multiplication is NOT commutative!

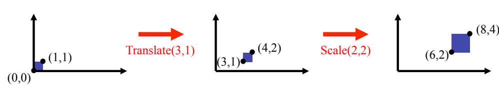
38

Non-commutative Composition

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



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



39

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

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

40

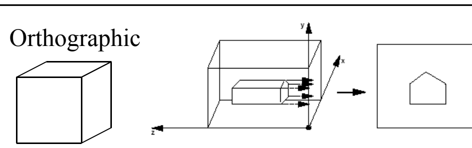
Outline

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

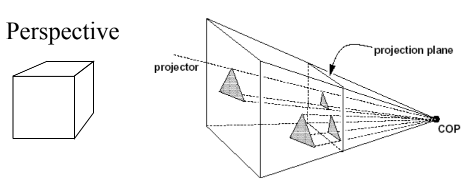
41

Orthographic vs. Perspective

- Orthographic



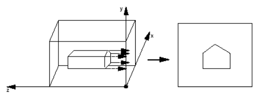
- Perspective



42

Simple Orthographic Projection

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

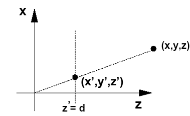


$$\begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Simple Perspective Projection

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

By similar triangles:
x'/x = d/z
x' = (x*d)/z



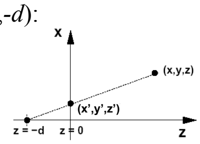
homogenize

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

Alternate Perspective Projection

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

By similar triangles:
x'/x = d/(z+d)
x' = (x*d)/(z+d)



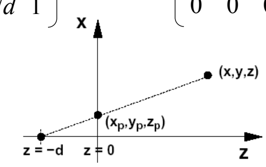
homogenize

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

In the limit, as $d \to \infty$

this perspective projection matrix... → ...is simply an orthographic projection

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



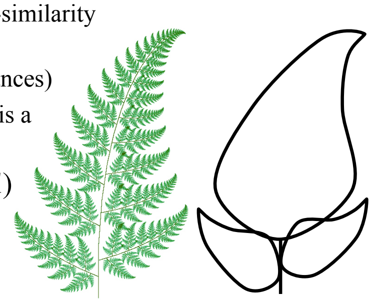
Outline

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

Iterated Function Systems (IFS)

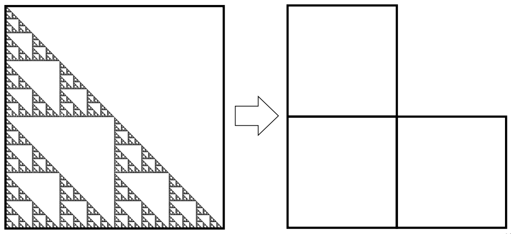
- Capture self-similarity
- Contraction (reduce distances)
- An attractor is a fixed point

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



Example: Sierpinski Triangle

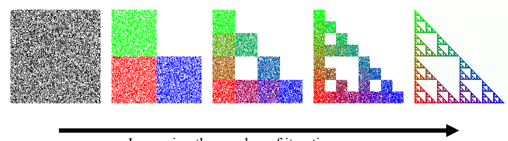
- Described by a set of n affine transformations
- In this case, $n = 3$
  - translate & scale by 0.5



49

Example: Sierpinski Triangle

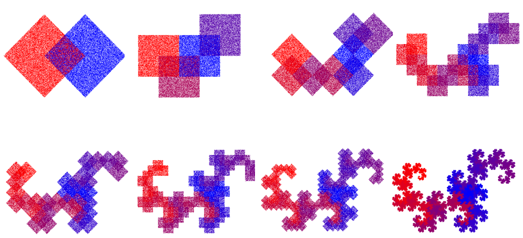
```
for "lots" of random input points (x_0, y_0)
   for j=0 to num_iters
      randomly pick one of the transformations
      (x_k+1, y_k+1) = f_i (x_k, y_k)
   display (x_k, y_k)
```



Increasing the number of iterations

50

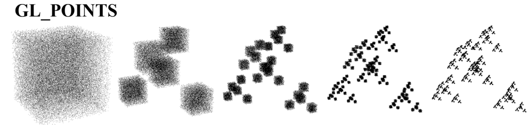
Another IFS: The Dragon



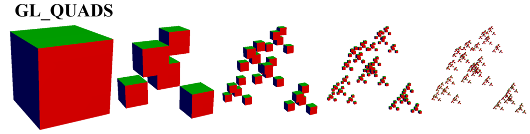
51

3D IFS in OpenGL

GL_POINTS



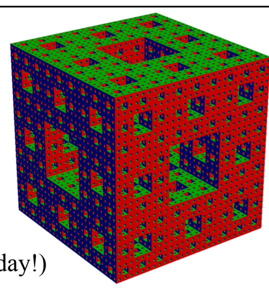
GL_QUADS



52

Assignment 0: OpenGL Warmup

- Get familiar with:
  - C++ environment
  - OpenGL
  - Transformations
  - simple Vector & Matrix classes
- Have Fun!
- Due ASAP (start it today!)
- ¼ of the points of the other HWs *(but you should still do it and submit it!)*



53

Outline

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

54

OpenGL

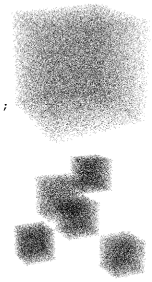
- OpenGL is a “state machine”
- OpenGL has lots of finicky setup & execution function calls
  - omitting a function call, swapping the order of 2 function calls, or passing the “wrong” argument, can result in a blank screen, nothing happens/changes, craziness happens, bus error, seg fault, etc.
- Often usually more than one way to do things
  - often one way is much faster than another
- What is possible depends on your hardware

55

OpenGL Basics: Array Buffer

- **Some useful commands:**

```
/* store data in points array */
glGenBuffers(1, &points_VBO);
glBindBuffer(GL_ARRAY_BUFFER,points_VBO);
glBufferData(GL_ARRAY_BUFFER, ..., points);
glColor3f(0,0,0);
glPointSize(1);
glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(...);
glEnableVertexAttribArray(...);
glVertexAttribPointer(...);
glDrawArrays(GL_POINTS, ...);
glDisableClientState(GL_VERTEX_ARRAY);
glDisableVertexAttribArray(...);
```

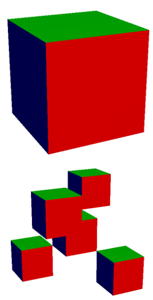


56

OpenGL Basics: Index Vertex Buffers

- **Some useful commands:**

```
/* store data in verts & faces arrays */
glBindBuffer(GL_ARRAY_BUFFER,cube_verts_VBO);
glBufferData(GL_ARRAY_BUFFER, cube_verts, ...);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,
   cube_face_indices_VBO);
glBufferData(GL_ELEMENT_ARRAY_BUFFER,
   cube_face_indices, GL_STATIC_DRAW);
glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(... BUFFER_OFFSET(0));
glEnableClientState(GL_NORMAL_ARRAY);
glNormalPointer(..., BUFFER_OFFSET(12));
glEnableClientState(GL_COLOR_ARRAY);
glColorPointer(..., BUFFER_OFFSET(24));
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,
   cube_face_indices_VBO);
glDrawElements(GL_QUADS, ...);
glDisableClientState(GL_NORMAL_ARRAY);
glDisableClientState(GL_COLOR_ARRAY);
glDisableClientState(GL_VERTEX_ARRAY);
```

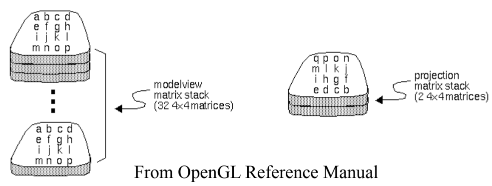


57

OpenGL Basics: Transformations

- Useful commands:

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



From OpenGL Reference Manual

58

Questions?

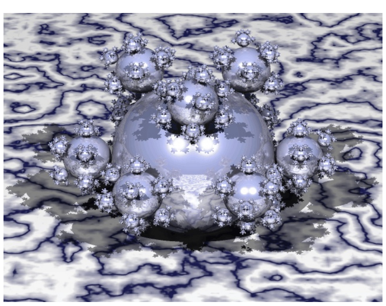
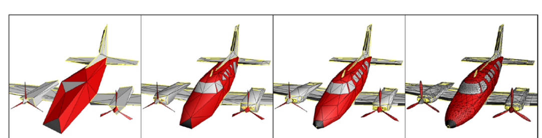


Image by Henrik Wann Jensen

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



(a) Base mesh M^0 (150 faces) (b) Mesh M^{175} (500 faces) (c) Mesh M^{425} (1,000 faces) (d) Original $\hat{M}=M^n$ (13,546 faces)

60