CSCI-4972/6963
Advanced Computer Graphics

http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S07/

Professor Barb Cutler
cutler@cs.rpi.edu
MRC 309A

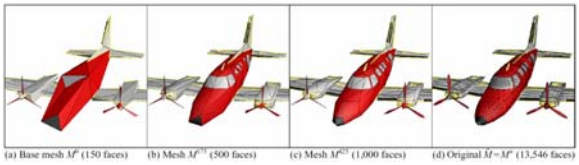
Luxo Jr.

Pixar Animation Studios, 1986

Topics for the Semester

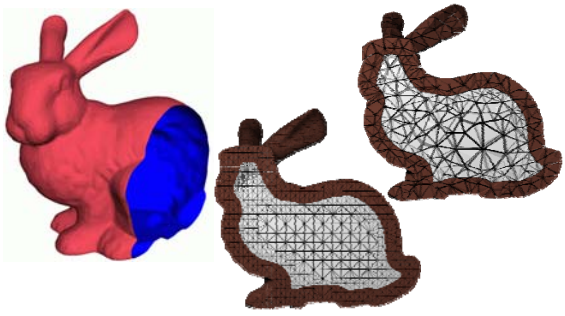
- Meshes
  - representation
  - simplification
  - subdivision surfaces
  - generation
  - volumetric modeling
- Simulation
  - particle systems
  - rigid body, deformation, cloth, wind/water flows
  - collision detection
  - weathering
- Rendering
  - ray tracing
  - appearance models
  - shadows
 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

Mesh Simplification



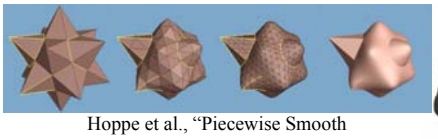
Hoppe "Progressive Meshes" SIGGRAPH 1996

Mesh Generation & Volumetric Modeling

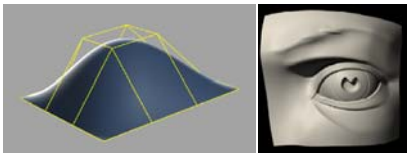


Cutler et al., "Simplification and Improvement of Tetrahedral Models for Simulation" 2004

Modeling – Subdivision Surfaces

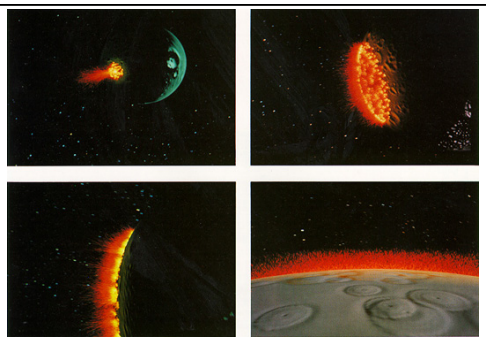


Hoppe et al., "Piecewise Smooth Surface Reconstruction" 1994



Geri's Game
Pixar 1997

Particle Systems

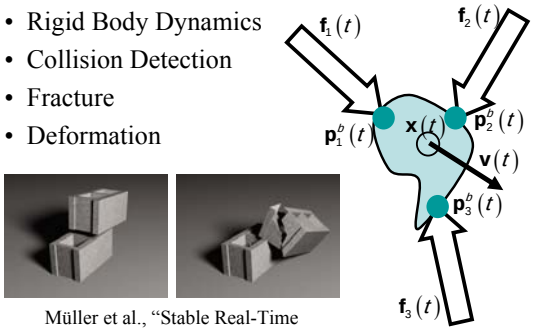


Star Trek: The Wrath of Khan 1982

7

Physical Simulation

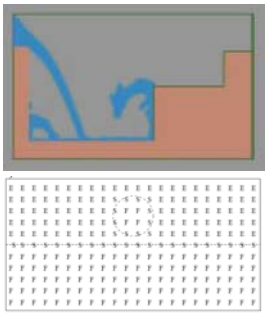
- Rigid Body Dynamics
- Collision Detection
- Fracture
- Deformation



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

8

Fluid Dynamics



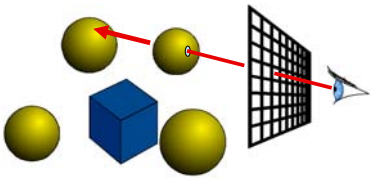
Foster & Mataxas, 1996

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

9

Ray Casting

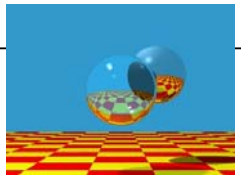
- For every pixel
  construct a ray from the eye
  - For every object in the scene
    - Find intersection with the ray
    - Keep if closest



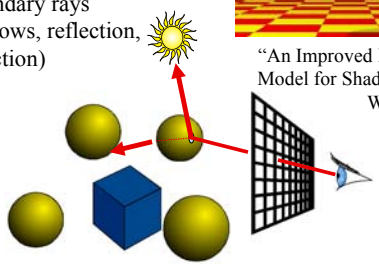
10

Ray Tracing

- Shade (interaction of light and material)
- Secondary rays (shadows, reflection, refraction)

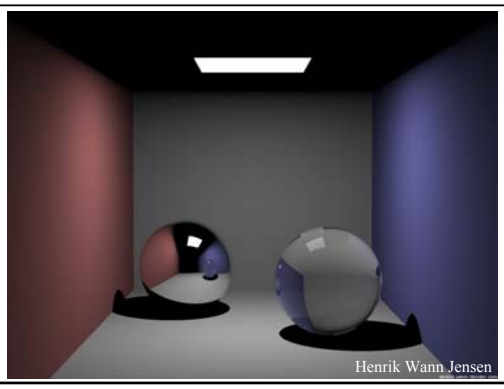


"An Improved Illumination Model for Shaded Display" Whitted 1980



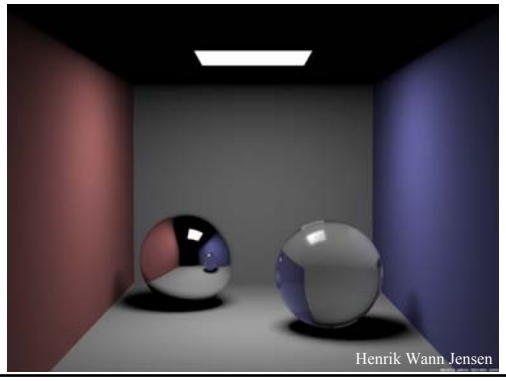
11

Traditional Ray Tracing



12

Ray Tracing + Soft Shadows



13

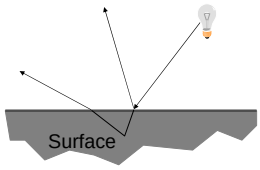
Ray Tracing + Caustics

14

Global Illumination

15

Subsurface Scattering

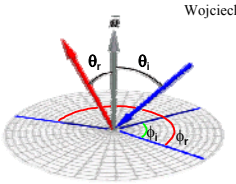


Jensen et al., "A Practical Model for Subsurface Light Transport" 2001

16

Appearance Models

Wojciech Matusik



Henrik Wann Jensen

17

Syllabus & Course Website

http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S07/

- Which version should I register for?
  - CSCI 6963
    - 3 units of graduate credit
  - CSCI 4972
    - 4 units of undergraduate credit

  same lectures, assignments, quizzes, & grading criteria
- Other Questions?

18

Introductions – Who are you?

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

19

Outline

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

20

What is a Transformation?

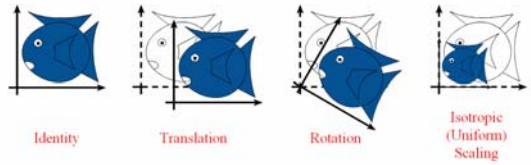
- Maps points (x, y) in one coordinate system to points (x', y') in another coordinate system

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

- For example, Iterated Function System (IFS):

21

Simple Transformations



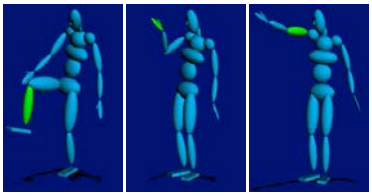
- Can be combined
- Are these operations invertible?

  Yes, except scale = 0

22

Transformations are used to:

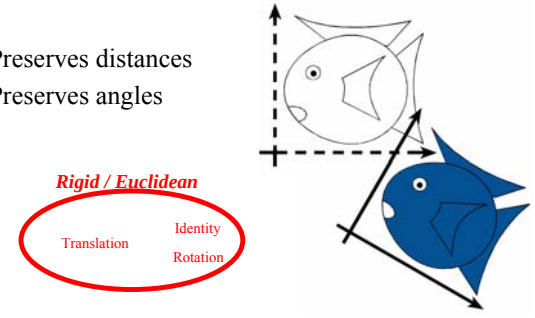
- Position objects in a scene
- Change the shape of objects
- Create multiple copies of objects
- Projection for virtual cameras
- Describe animations



23

Rigid-Body / Euclidean Transforms

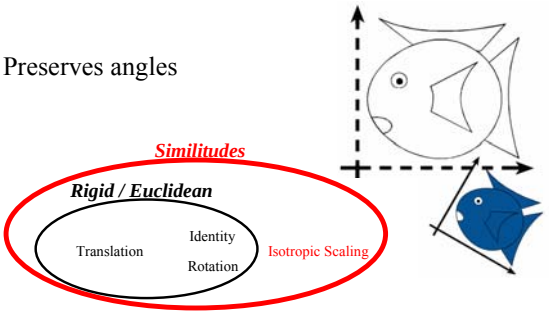
- Preserves distances
- Preserves angles



24

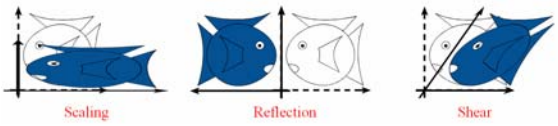
Similitudes / Similarity Transforms

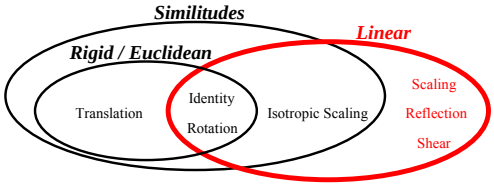
- Preserves angles



25

Linear Transformations

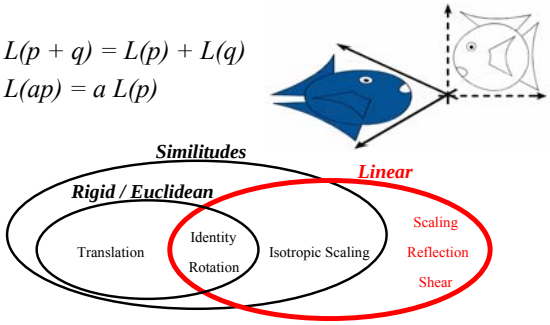




26

Linear Transformations

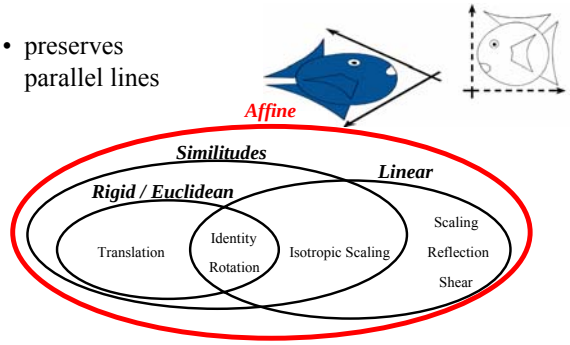
- $L(p + q) = L(p) + L(q)$
- $L(ap) = a\,L(p)$



27

Affine Transformations

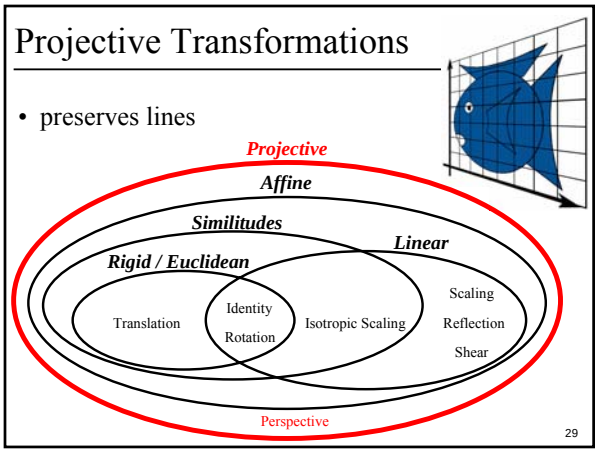
- preserves parallel lines



28

Projective Transformations

- preserves lines



29

General (Free-Form) Transformation

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

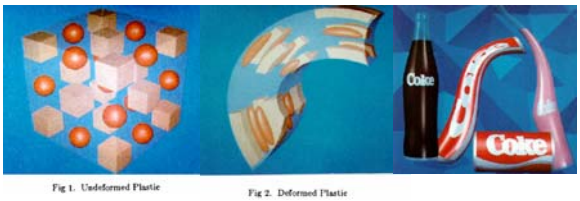


Fig 1. Undeformed Plastic Fig 2. Deformed Plastic

Sederberg and Parry, Siggraph 1986

30

Outline

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

31

How are Transforms Represented?

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

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

$$p' = M p + t$$

32

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

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

$$p' = M p$$

33

Translation in homogeneous coordinates

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

Affine formulation

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

$$p' = M p + t$$

Homogeneous formulation

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

$$p' = M p$$

34

Homogeneous Coordinates

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

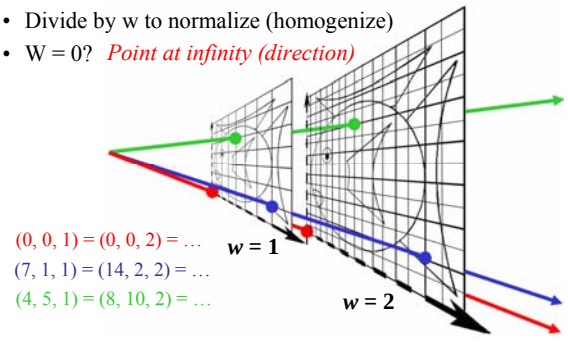
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

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

35

Homogeneous Visualization

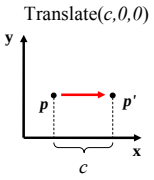
- Divide by w to normalize (homogenize)
- W = 0? *Point at infinity (direction)*



36

Translate (tx, ty, tz)

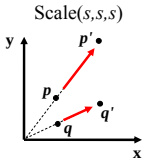
- Why bother with the extra dimension?
  Because now translations can be encoded in the matrix!



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

Scale (sx, sy, sz)

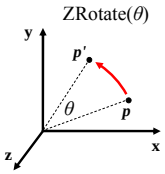
- Isotropic (uniform) scaling: $s_x = s_y = s_z$



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

Rotation

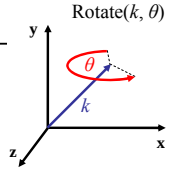
- About z axis



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

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

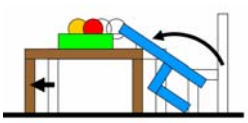
Storage

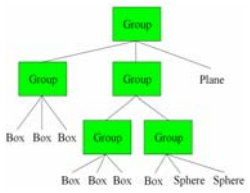
- Often, w is not stored (always 1)
- Needs careful handling of direction vs. point
  - Mathematically, the simplest is to encode directions with $w = 0$
  - In terms of storage, using a 3-component array for both direction and points is more efficient
  - Which requires to have special operation routines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

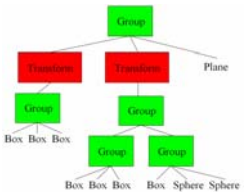
Outline

Adding Transformations

- To position the logical groupings of objects within the scene

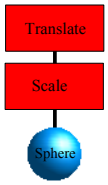




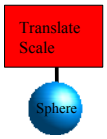


Nested Transforms

$$p' = T(S\,p) = TS\,p$$



same as

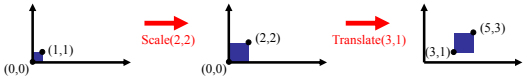



```
Transform {
    Translate { 1 0.5 0 }
    Transform {
        Scale { 2 2 2 }
        Sphere {
            center 0 0 0
            radius 1 } } }
```

```
Transform {
    Translate { 1 0.5 0 }
    Scale { 2 2 2 }
    Sphere {
        center 0 0 0
        radius 1 } }
```


How are transforms combined?

Scale then Translate



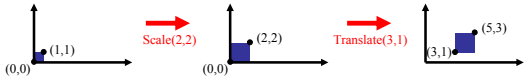
Use matrix multiplication: $p' = T(S\,p) = TS\,p$

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

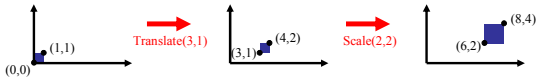
Caution: matrix multiplication is NOT commutative!

Non-commutative Composition

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



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



Non-commutative Composition

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

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

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

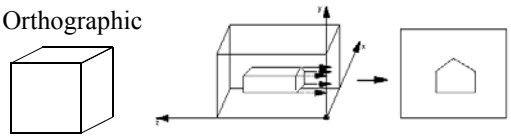
$$ST = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 6 \\ 0 & 2 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

Outline

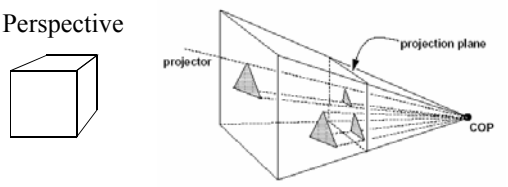
- Course Overview
- Classes of Transformations
- Representing Transformations
- Transformations in Modelling
- Combining Transformations
- Orthographic & Perspective Projections
- Example: Iterated Function Systems (IFS)

Orthographic vs. Perspective

- Orthographic



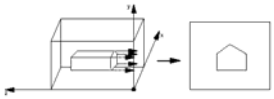
- Perspective



49

Simple Orthographic Projection

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



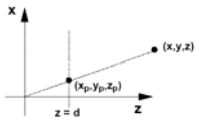
$$\begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

50

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

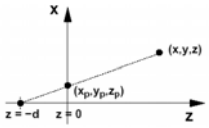
$$\begin{pmatrix} x * d / z \\ y * d / z \\ d \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z / d \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

51

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

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

52

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

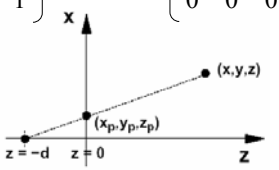
this perspective projection matrix...

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

$\rightarrow$

...is simply an orthographic projection

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



53

Outline

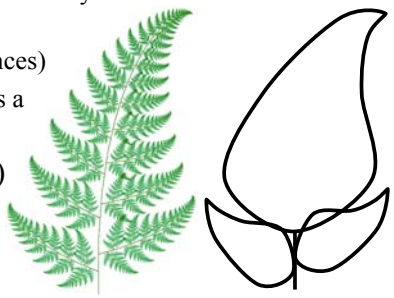
- Course Overview
- Classes of Transformations
- Representing Transformations
- Transformations in Modelling
- Combining Transformations
- Orthographic & Perspective Projections
- Example: Iterated Function Systems (IFS)

54

Iterated Function Systems (IFS)

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

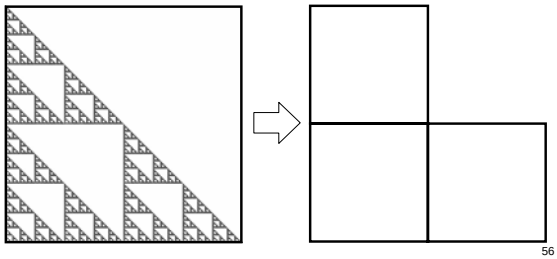
$$A = \bigcup f_i(A)$$



55

Example: Sierpinski Triangle

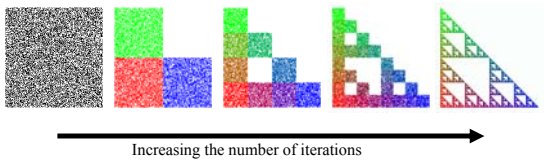
- Described by a set of *n* affine transformations
- In this case, $n = 3$
  - translate & scale by 0.5



56

Example: Sierpinski Triangle

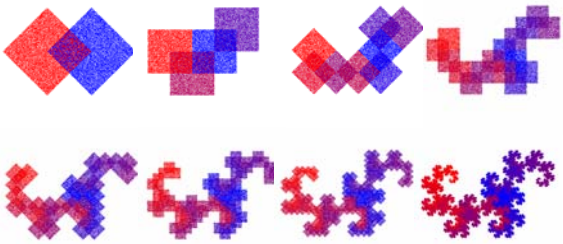
```
for "lots" of random input points (x0, y0)
   for j=0 to num_iters
      randomly pick one of the transformations
      (xk+1, yk+1) = fi (xk, yk)
   display (xk, yk)
```



Increasing the number of iterations

57

Another IFS: The Dragon



58

3D IFS in OpenGL

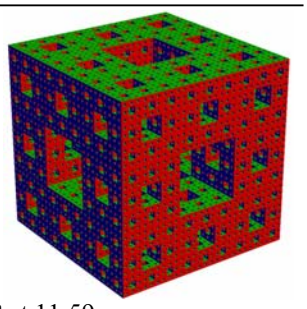
GL_POINTS

GL_QUADS

59

Assignment 1: OpenGL Warmup

- Get familiar with:
  - C++ environment
  - OpenGL
  - Transformations
  - simple Vector, Matrix & Image classes
- Have Fun!



- Due Thursday Jan 25th at 11:59pm

60