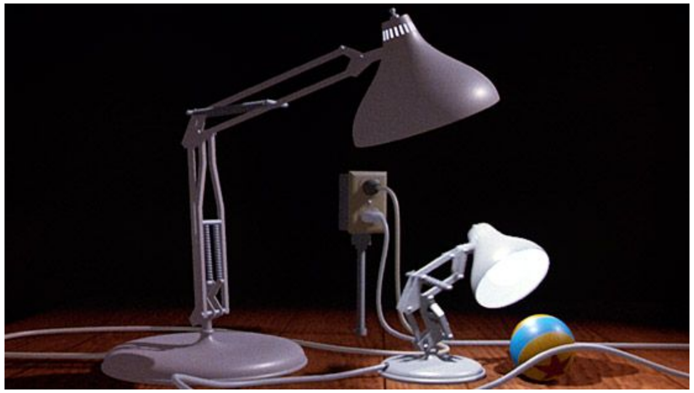
CSCI-4530/6530 Advanced Computer Graphics

http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S17/

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

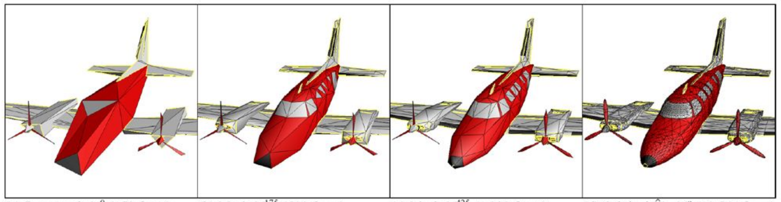
Luxo Jr.


Pixar Animation Studios, 1986

Topics for the Semester

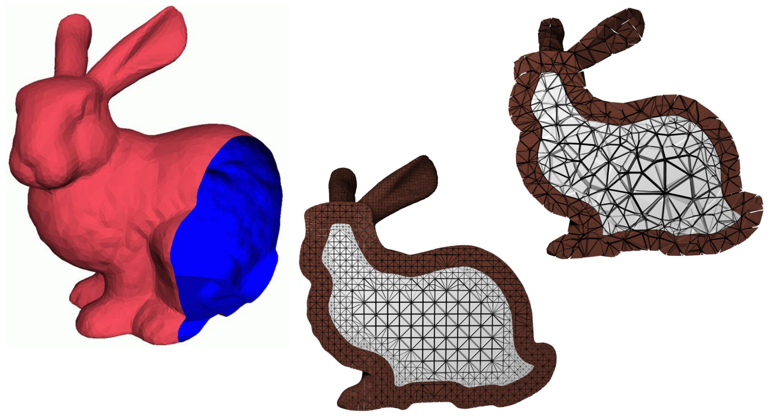
- Meshes
  - representation
  - simplification
  - subdivision surfaces
  - construction/generation
  - volumetric modeling
- Simulation
  - particle systems, cloth
  - rigid body, deformation
  - wind/water flows
  - collision detection
  - weathering
- Rendering
  - ray tracing, shadows
  - appearance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

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)

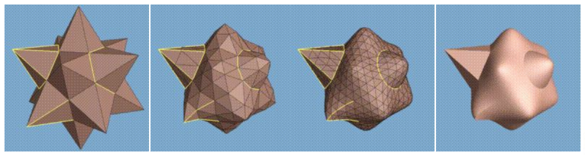
Hoppe "Progressive Meshes" SIGGRAPH 1996

Mesh Generation & Volumetric Modeling

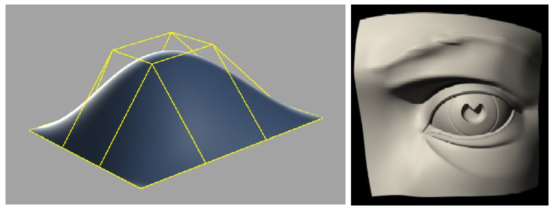


Cutler et al., "Simplification and Improvement of Tetrahedral Models for Simulation" 2004

Modeling – Subdivision Surfaces

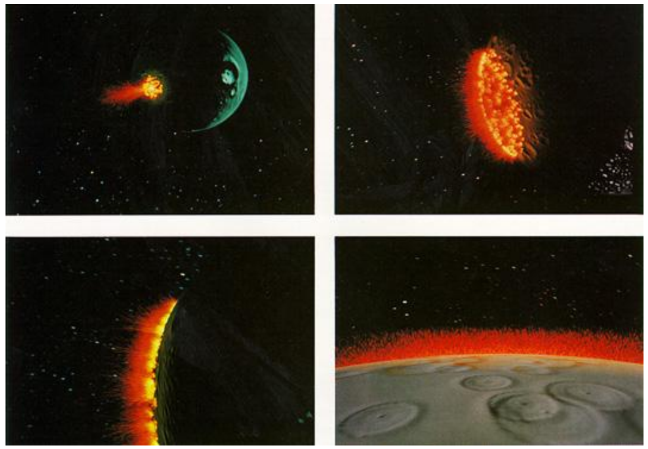


Hoppe et al., "Piecewise Smooth Surface Reconstruction" 1994



Geri's Game Pixar 1997

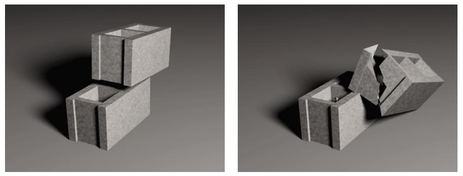
Particle Systems



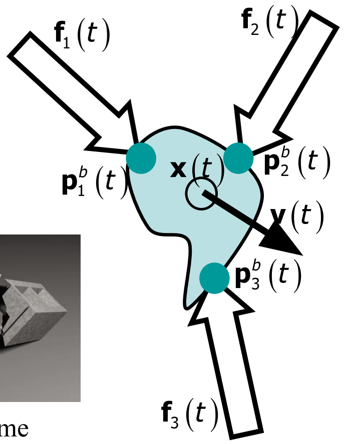
Star Trek: The Wrath of Khan 1982

Physical Simulation

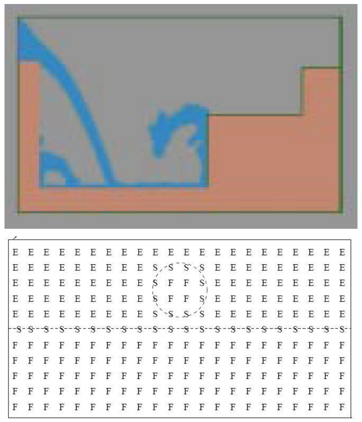
- Rigid Body Dynamics
- Collision Detection
- Fracture
- Deformation



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



Fluid Dynamics

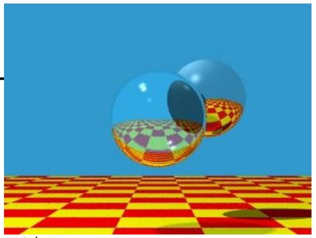


Foster & Mataxas, 1996

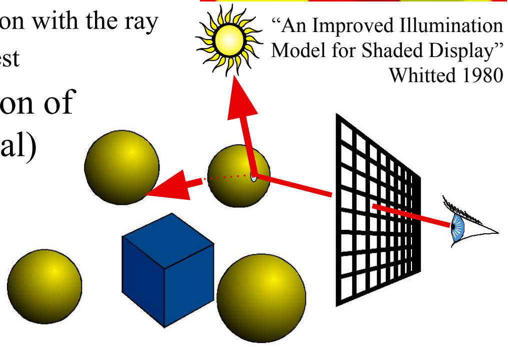
"Visual Simulation of Smoke"
Fedkiw, Stam & Jensen
SIGGRAPH 2001

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)

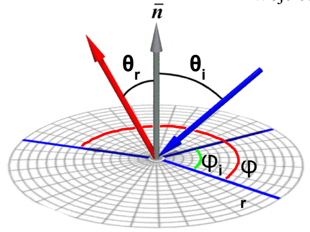


"An Improved Illumination Model for Shaded Display"
Whitted 1980



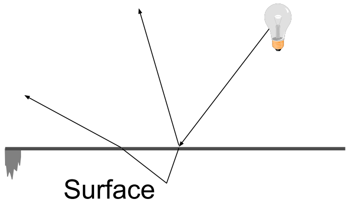
Appearance Models

Wojciech Matusik



Henrik Wann Jensen

Subsurface Scattering



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

Syllabus & Course Website

http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S17/

- Which version should I register for?
  - CSCI 6530 : 3 units of graduate credit
  - CSCI 4530 : 4 units of undergraduate credit

  (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.

Grades

http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S17/

- This course counts as "communications intensive" for undergraduates. As such, you must satisfactorily complete all readings, presentations, project reports to pass the course.
- As this is an elective (not required) course, I expect to grade this course: "A", "A-", "B+", "B", "B-", or "F"
  - *Don't expect C or D level work to "pass"*
  - *I don't want to give any "F"s*

Participation/Laptops in Class Policy

http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S17/

- 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

Questions?

Outline

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

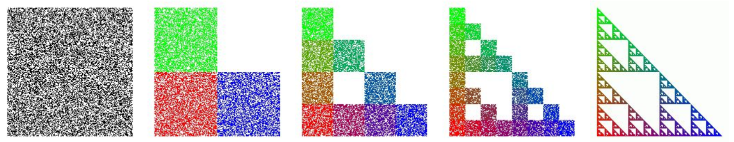
17

What is a Transformation?

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

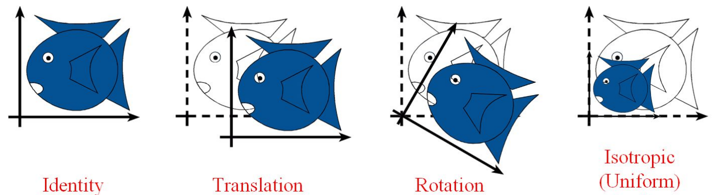
$$x' = ax + by + c$$
$$y' = dx + ey + f$$

- For example, Iterated Function System (IFS):



18

Simple Transformations

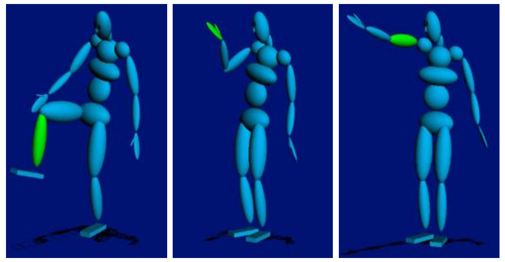


Yes, except scale = 0

19

Transformations are used to:

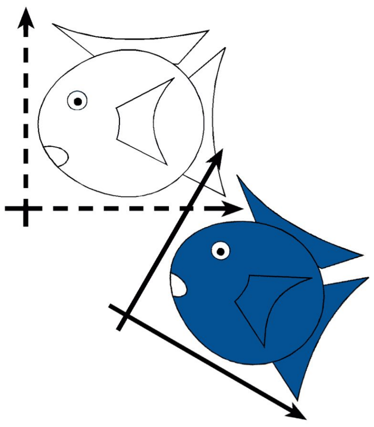
- Position objects in a scene
- Change the shape of objects
- Create multiple copies of objects
- Projection for virtual cameras
- Describe animations

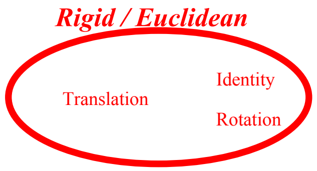


20

Rigid-Body / Euclidean Transforms

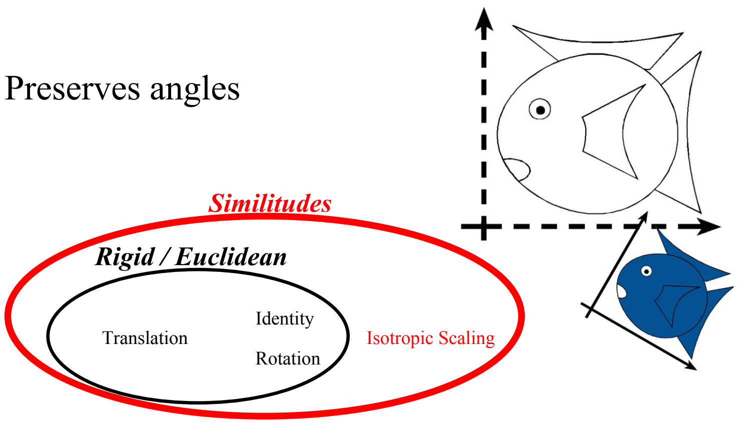
- Preserves distances
- Preserves angles



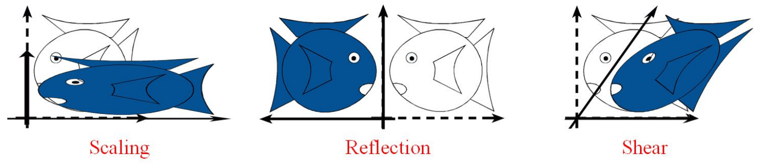


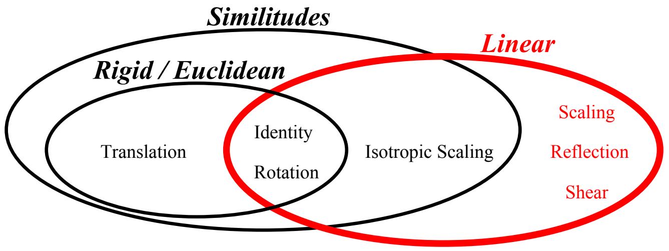
Similitudes / Similarity Transforms

- Preserves angles



Linear Transformations

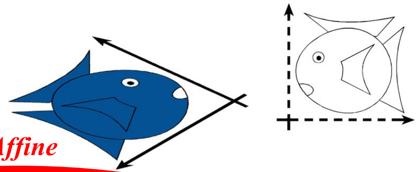


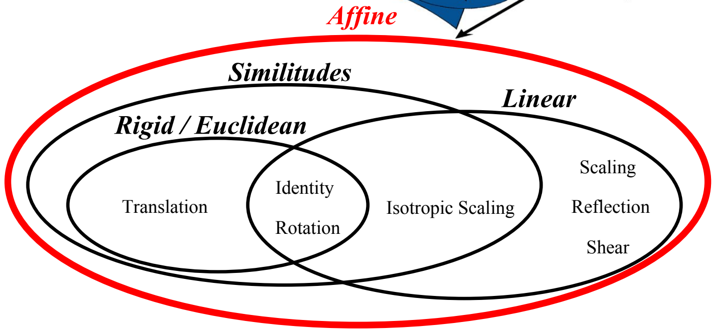


$L(p + q) = L(p) + L(q)$ $\qquad$ $L(ap) = a\,L(p)$

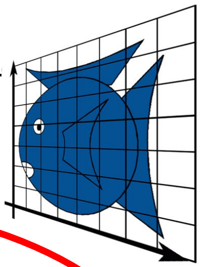
Affine Transformations

- preserves parallel lines

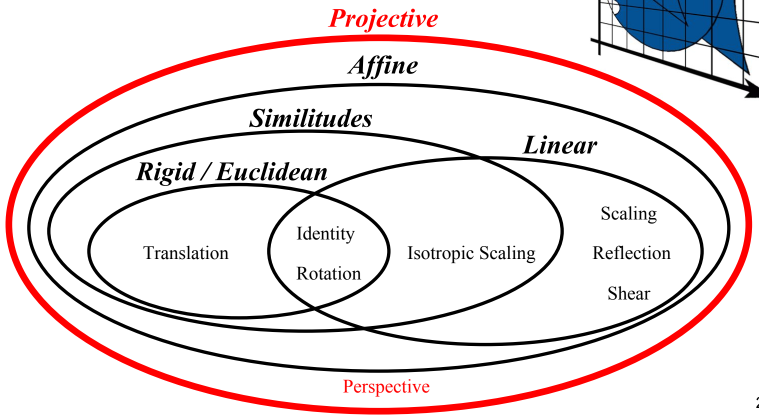




Projective Transformations



- preserves lines



General (Free-Form) Transformation

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

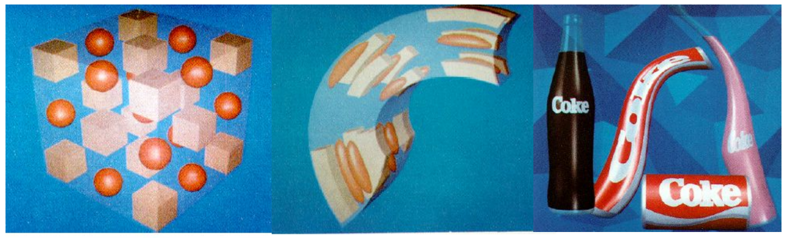


Fig 1. Undeformed Plastic

Fig 2. Deformed Plastic

Sederberg and Parry, Siggraph 1986

Outline

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

How are Transforms Represented?

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

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

$$p' = M p + t$$

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

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

$$p' = \color{red}{M}\, p$$

Translation in homogeneous coordinates

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

Affine formulation

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

$$p' = \color{red}{M}\, p + \color{red}{t}$$

Homogeneous formulation

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

$$p' = \color{red}{M}\, p$$

Homogeneous Coordinates

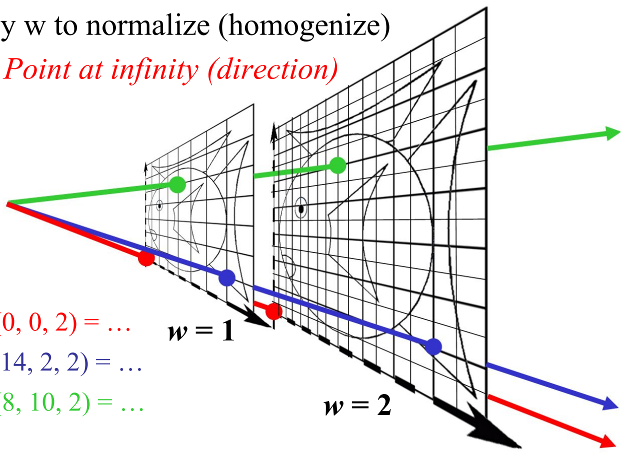
- Most of the time w = 1, and we can ignore it

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

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

Homogeneous Visualization

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

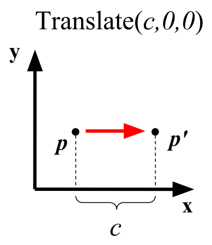


(0, 0, 1) = (0, 0, 2) = …

(7, 1, 1) = (14, 2, 2) = …

(4, 5, 1) = (8, 10, 2) = …

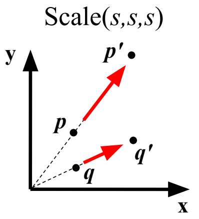
Translate (t_x, t_y, t_z)



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

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

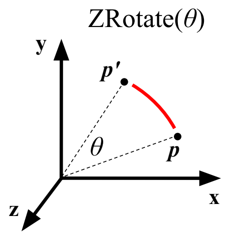
Scale (s_x, s_y, s_z)



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

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

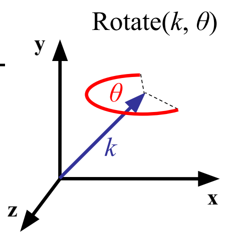
Rotation



- 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

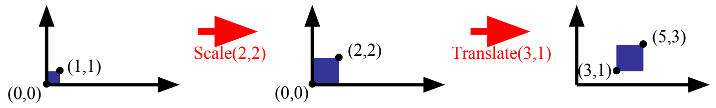
- Often, w is not stored (always 1)
- Needs careful handling of direction vs. point
  - Mathematically, the simplest is to encode directions with $w = 0$
  - In terms of storage, using a 3-component array for both direction and points is more efficient
  - Which requires to have special operation routines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

Outline

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

How are transforms combined?

Scale then Translate



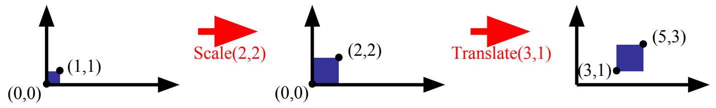
Use matrix multiplication: p' = T (S p) = TS p

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

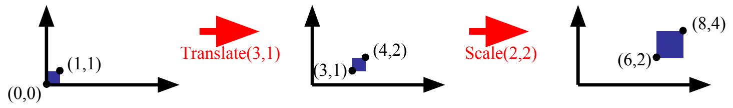
Caution: matrix multiplication is NOT commutative!

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

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

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

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

Exercise!

Form teams of 2. Use 1 piece of paper. Put both names on the top. Work together. Both people should write. Hand in to TA Jeramey Tyler after we discuss.

Write down the 3x3 matrix that transforms this set of 4 points:

A: (0,0) B: (1,0) C: (1,1) D: (0,1)

to these new positions:

A': (-1, 1) B': (-1, 0) C': (0, 0) D': (0, 1)

Show your work.

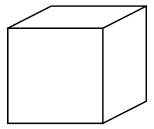
If you finish early…
Solve the problem using a different technique.

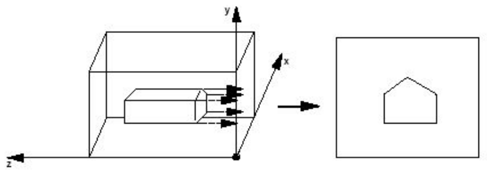
Outline

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

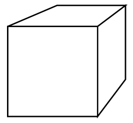
Orthographic vs. Perspective

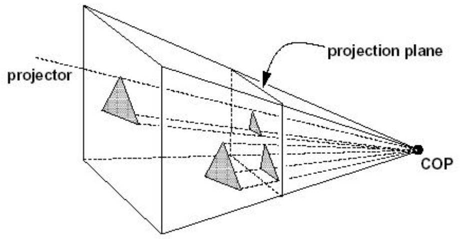
- Orthographic





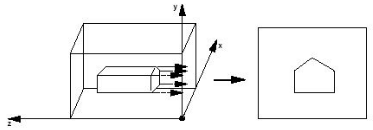
- Perspective





Simple Orthographic Projection

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

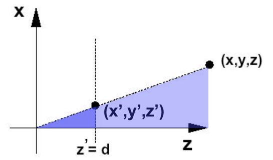


$$\begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

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



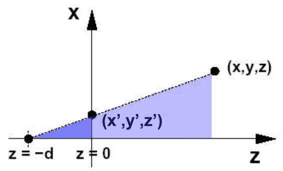
homogenize

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

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)



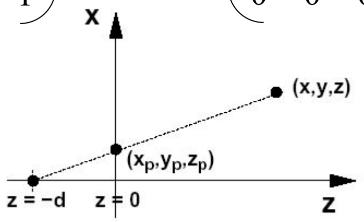
homogenize

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

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

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

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



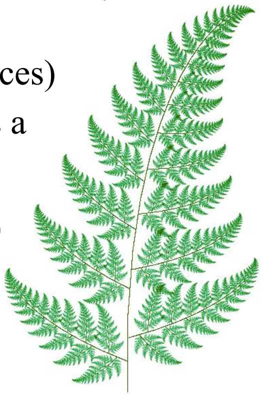
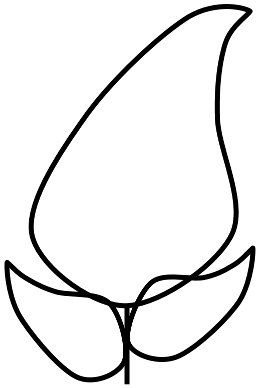
Outline

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

Iterated Function Systems (IFS)

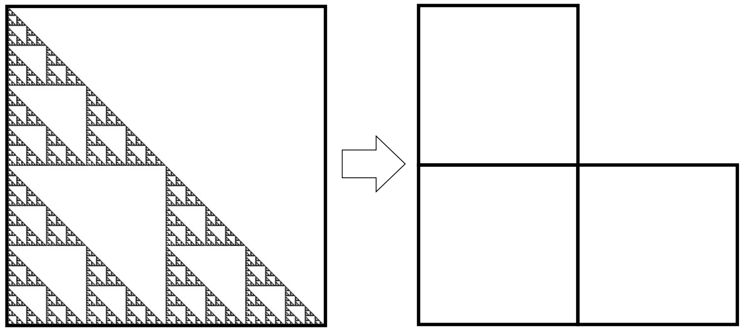
- Capture self-similarity
- Contraction (reduce distances)
- An attractor is a fixed point

$A = \square f_i(A)$

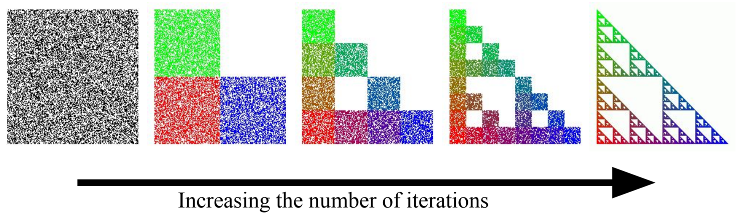
Example: Sierpinski Triangle

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

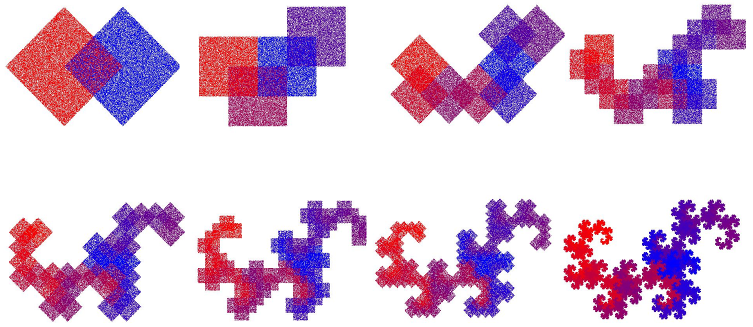


Example: Sierpinski Triangle

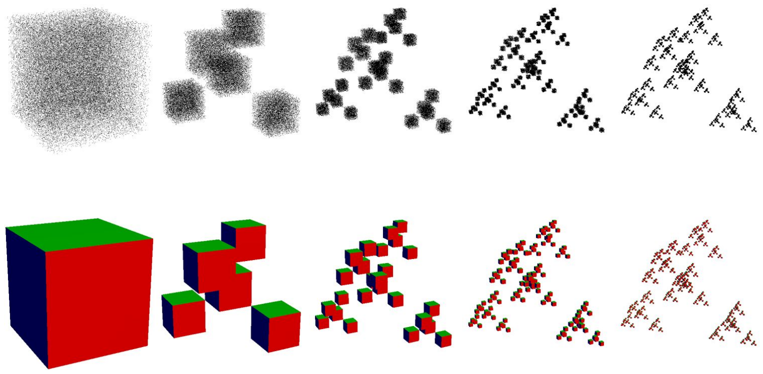
```
for "lots" of random input points ($x_0$, $y_0$)
   for j=0 to num_iters
      randomly pick one of the transformations
      ($x_{k+1}$, $y_{k+1}$) = $f_i$ ($x_k$, $y_k$)
   display ($x_k$, $y_k$)
```



Another IFS: The Dragon

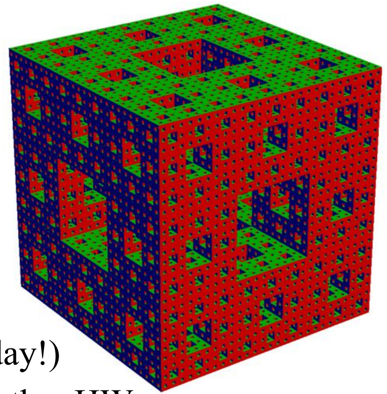


3D IFS in OpenGL



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!*)



Questions?

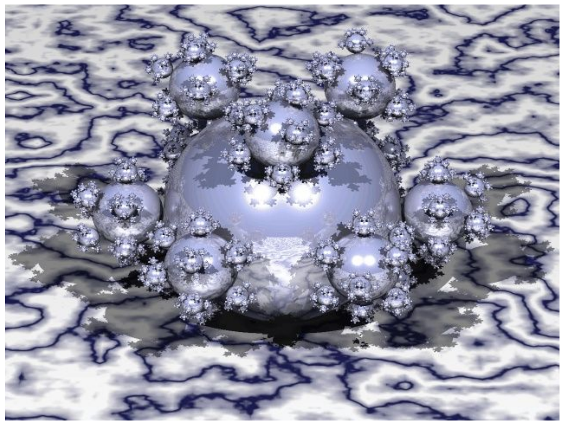
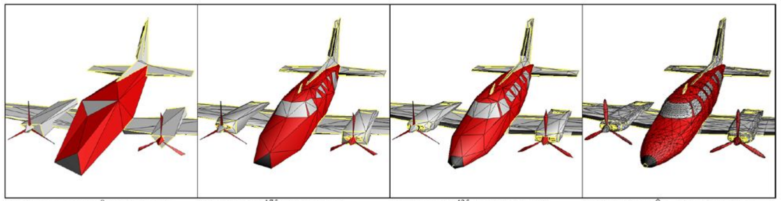


Image by Henrik Wann Jensen

For Next Time:

Volunteer to be "Discussant"?
Note: This is not a "presentation".
Be sure to read blurb (& linked webpage) on course webpage about Assigned Readings & Discussants.

- 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)

Questions to think about:

- How do we represent meshes?
- How to automatically decide what parts of the mesh are important / worth preserving?
- Algorithm performance: memory, speed?
- What were the original target applications?
  Are those applications still valid?
  Are there other modern applications that can leverage this technique?