# CSCI-4530/6530
# Advanced Computer Graphics

http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S21/

Barb Cutler
cutler@cs.rpi.edu



## Luxo Jr.

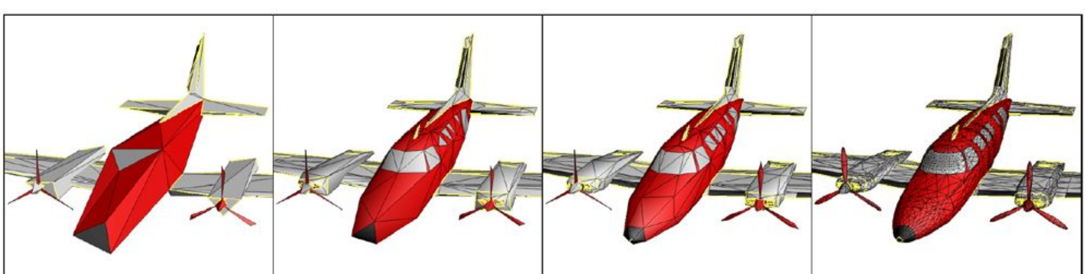
Pixar Animation Studios, 1986

# 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 …



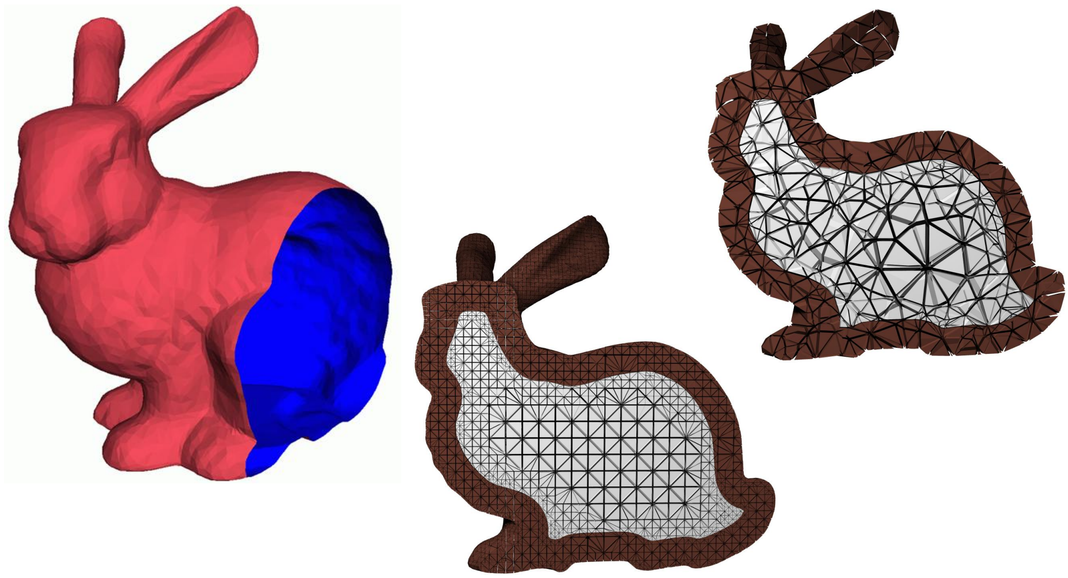
# Mesh Simplification



(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

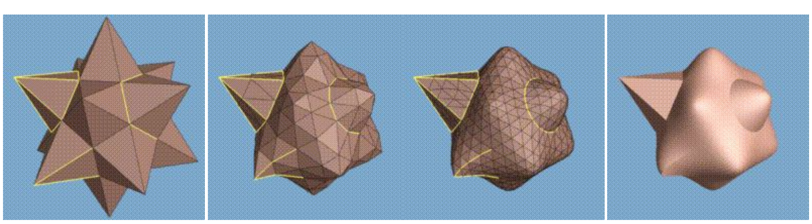
# Mesh Generation & Volumetric Modeling



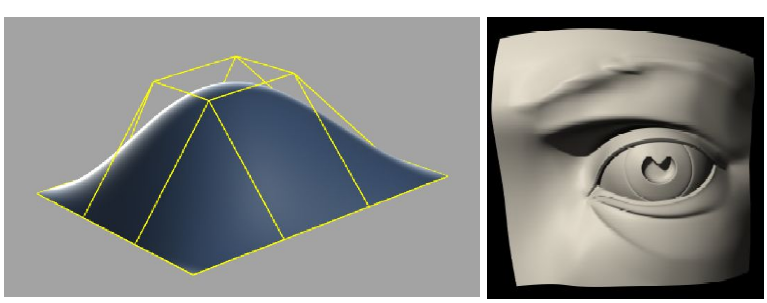
Cutler et al., “Simplification and Improvement of Tetrahedral Models for Simulation” 2004



# Modeling – Subdivision Surfaces

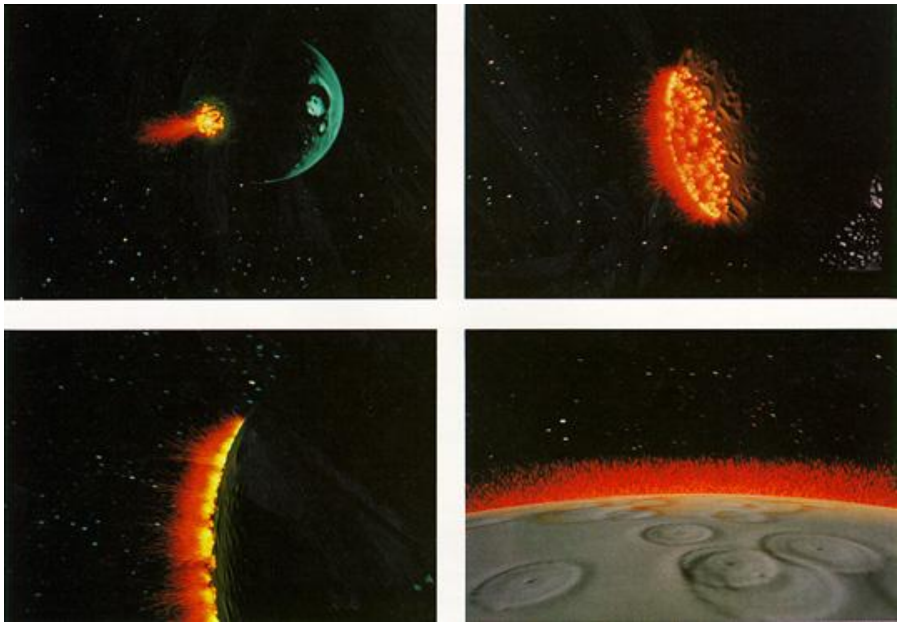


Hoppe et al., “Piecewise Smooth Surface Reconstruction” 1994



Geri’s Game
Pixar 1997

# Particle Systems

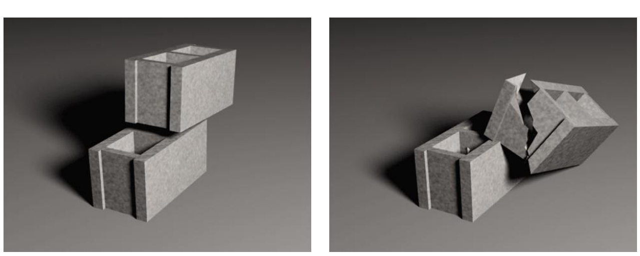


Star Trek: The Wrath of Khan 1982

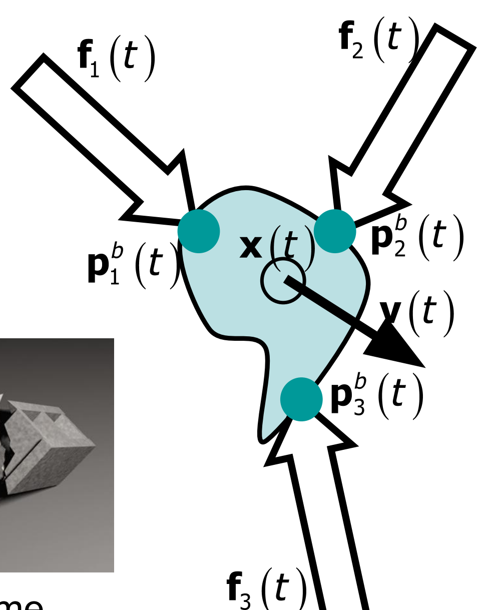


# Physical Simulation

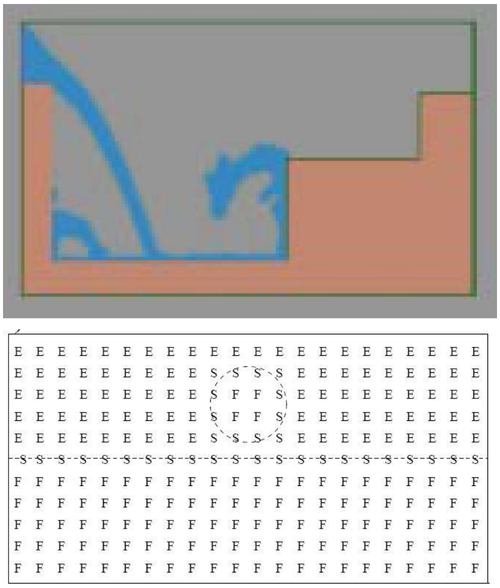
- Rigid Body Dynamics
- Collision Detection
- Fracture
- Deformation



Müller et al., “Stable Real-Time Deformations” 2002



# Fluid Dynamics



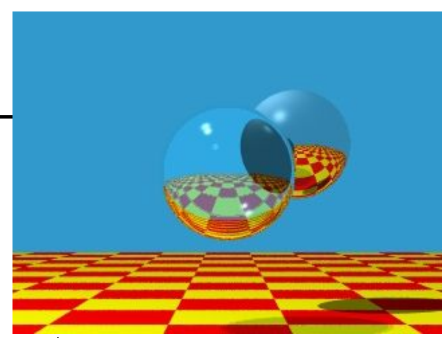

Foster & Mataxas, 1996

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

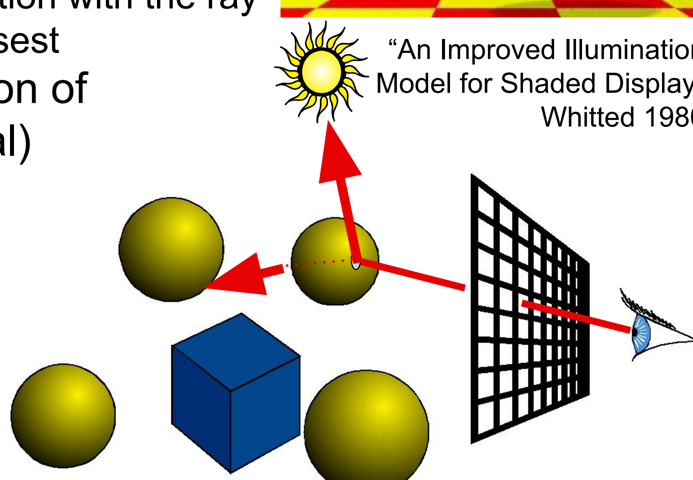


# Ray Casting/Tracing

- 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)

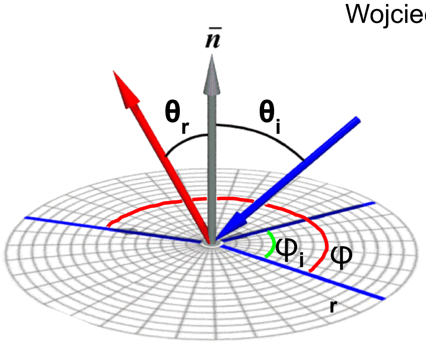


"An Improved Illumination Model for Shaded Display"
Whitted 1980



# Appearance Models

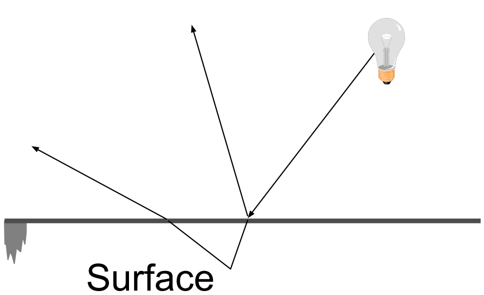
Wojciech Matusik




Henrik Wann Jensen



# Subsurface Scattering




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

# Syllabus & Course Website

http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S21/

- Which version should I register for?

  CSCI 6530 : 4 units of graduate credit

  CSCI 4530 : 4 units of undergraduate credit

- 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/S21/

- 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*

# Lecture Attendance/Participation

http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S21/

- Lecture will be discussion-intensive
  - We will discuss research papers
  - We will do worksheets in groups of 2 or 3
- You are expected to regularly attend and participate in the live lecture
  - Lecture will be recorded & posted on Mediasite
  - If time zones or technical problems force you to miss more than a couple lectures, please contact me ASAP



# Questions?

# Outline

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

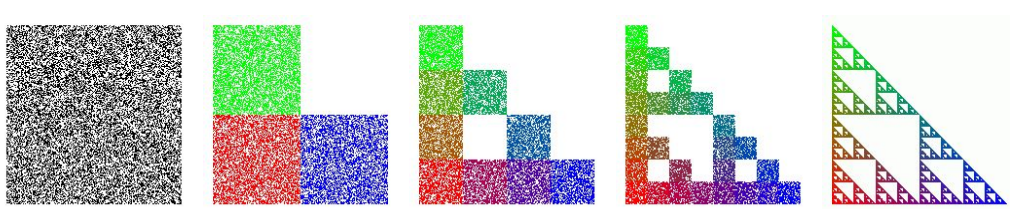


# What is a Transformation?

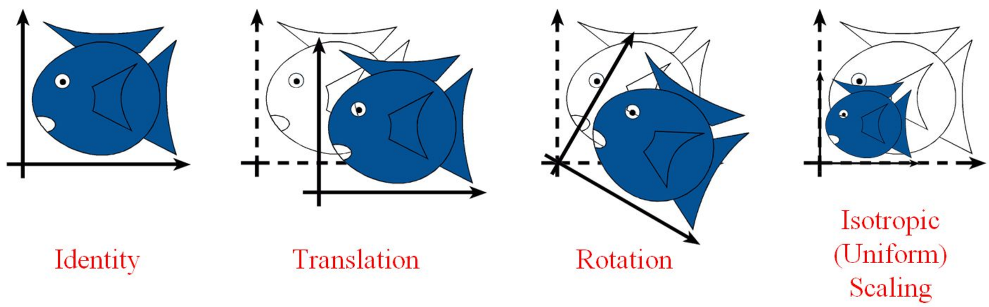
- Maps points ($x$, $y$) in one coordinate system to points ($x'$, $y'$) in another coordinate system

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

- For example, Iterated Function System (IFS):



# Simple Transformations



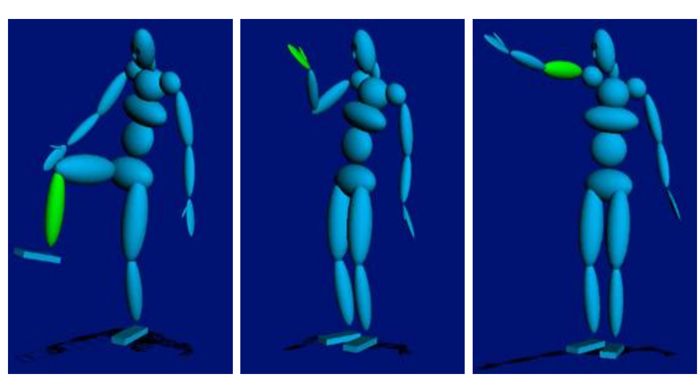

- Can be combined
- Are these operations invertible?

  *Yes, except scale = 0*



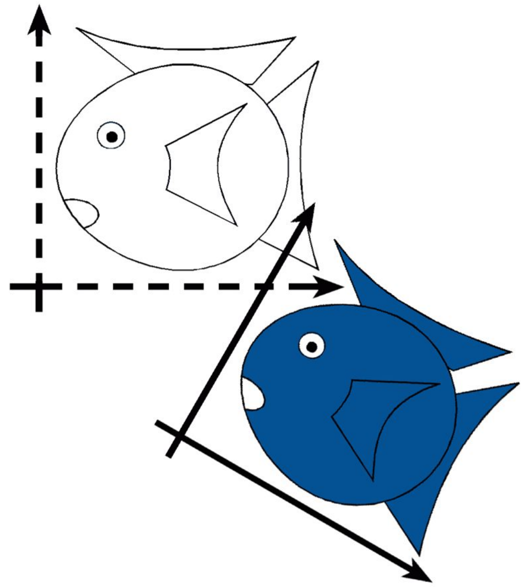
# Transformations are used to:

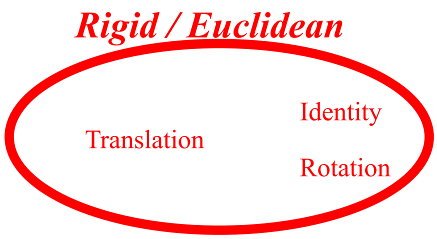
- Position objects in a scene
- Change the shape of objects
- Create multiple copies of objects
- Projection for virtual cameras
- Describe animations



# Rigid-Body / Euclidean Transforms

- Preserves distances
- Preserves angles

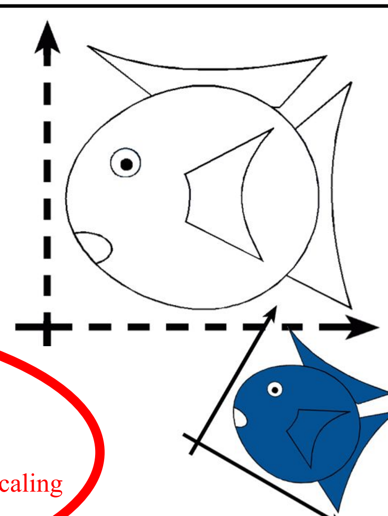


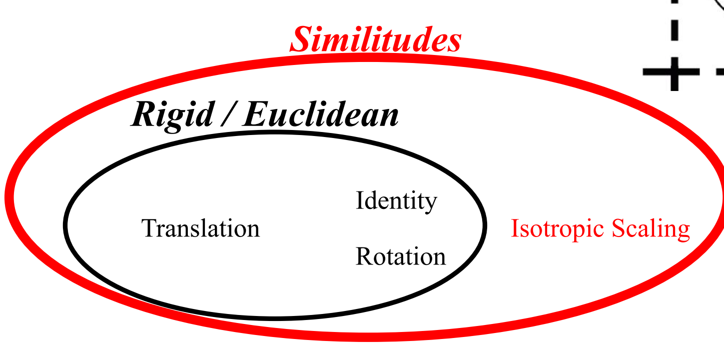




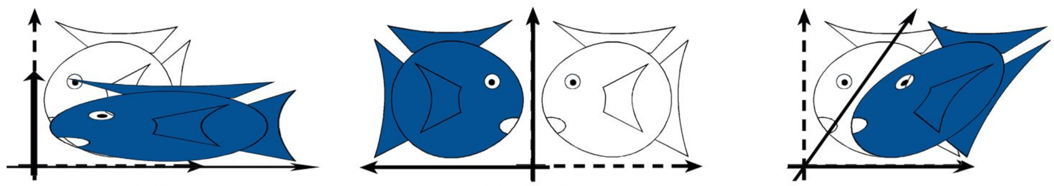

# Similitudes / Similarity Transforms

- Preserves angles

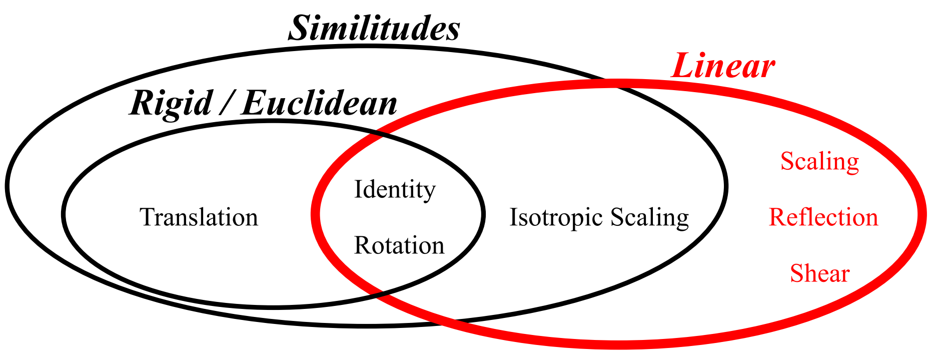




# Linear Transformations



Scaling Reflection Shear

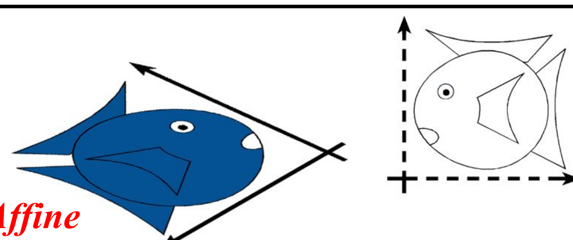


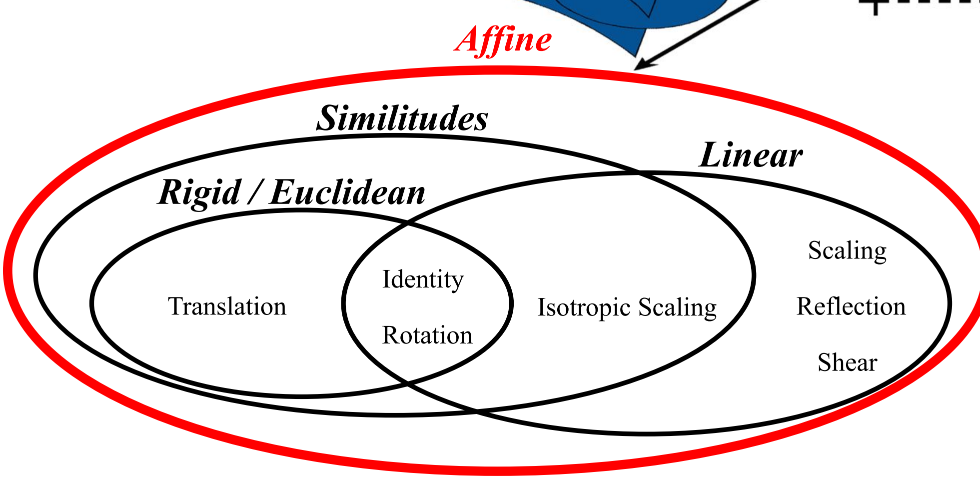

$$L(p + q) = L(p) + L(q) \qquad L(ap) = a\,L(p)$$



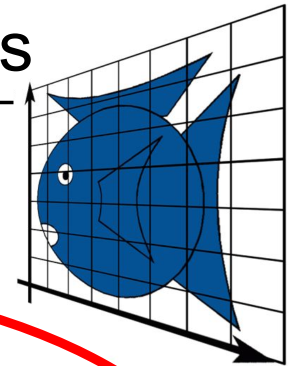
# Affine Transformations

- preserves parallel lines

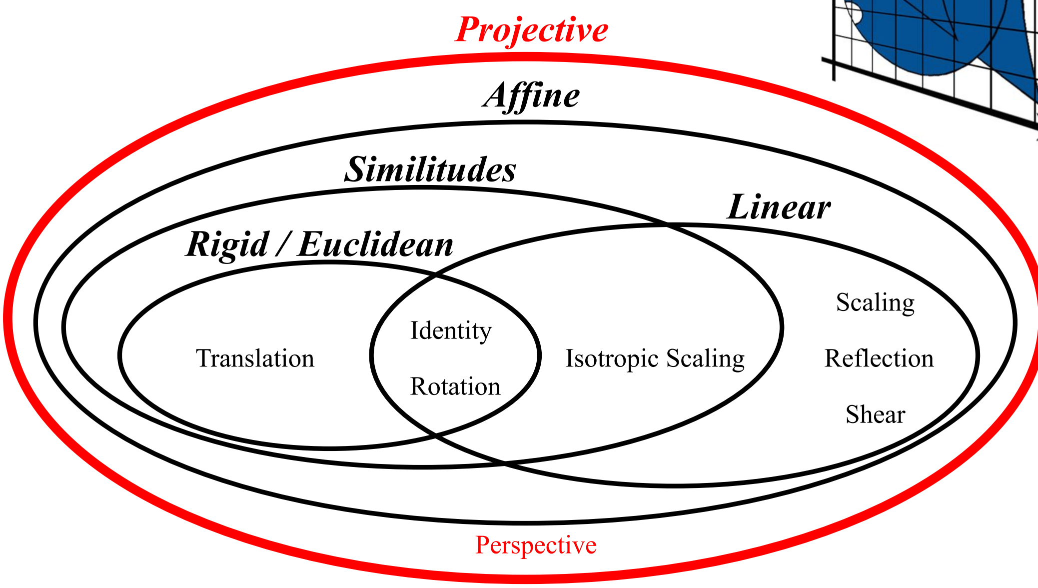




# Projective Transformations



- preserves lines






# General (Free-Form) Transformation

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

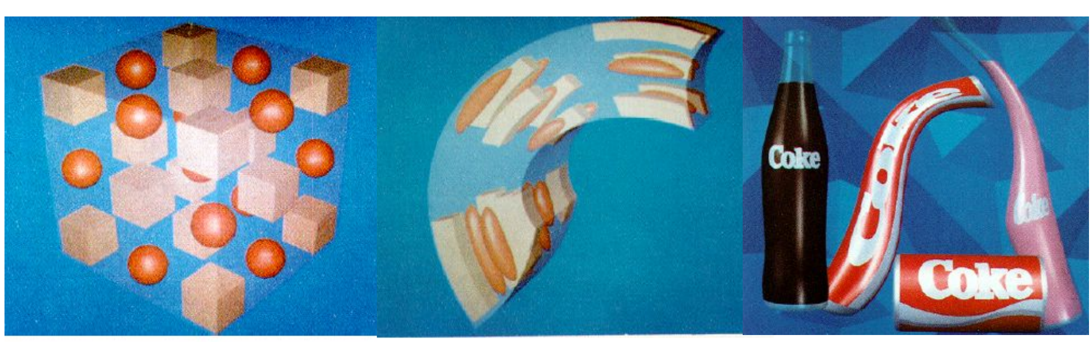



Fig 1. Undeformed Plastic

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?

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

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

$$p' = Mp + t$$

# 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

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

$$p' = Mp$$



# 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

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

$$p' = Mp$$

# Homogeneous Coordinates

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

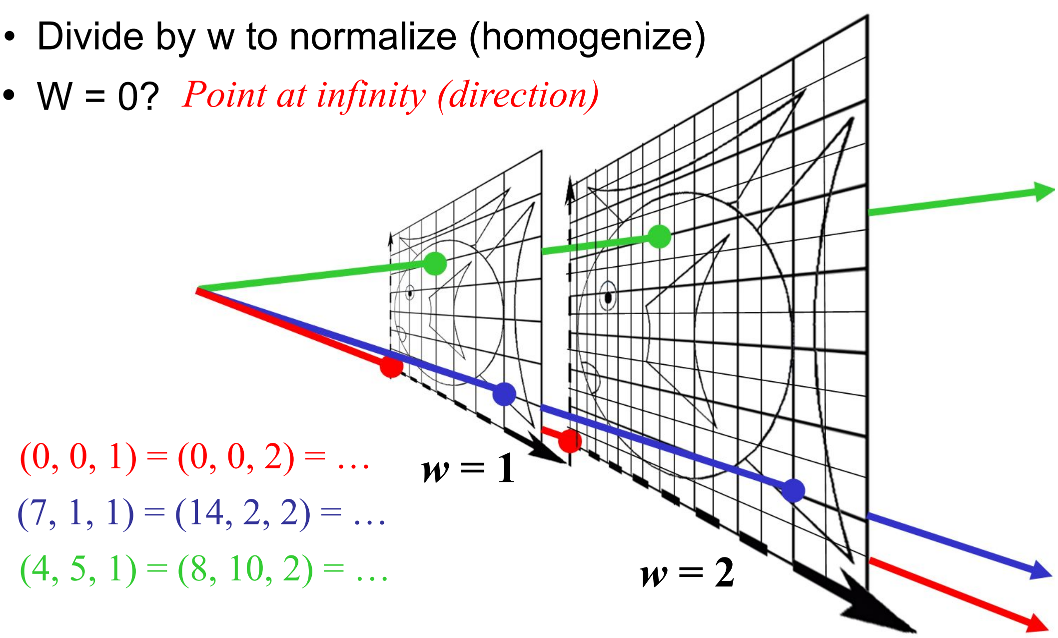
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

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



# Homogeneous Visualization

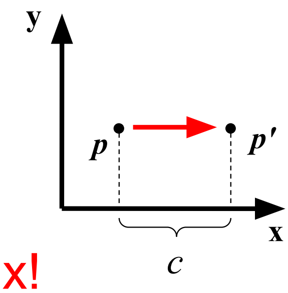
- Divide by w to normalize (homogenize)
- W = 0? *Point at infinity (direction)*



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

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

Translate($c$,0,0)



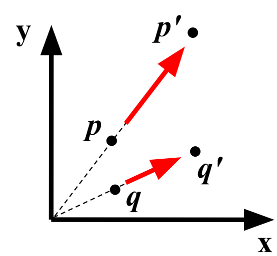

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



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

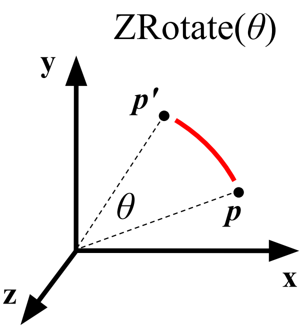
- Isotropic (uniform) scaling: $s_x = s_y = s_z$

Scale($s$,$s$,$s$)




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

# Rotation

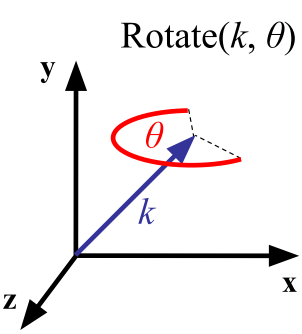



- About z axis

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



# Rotation




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

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

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

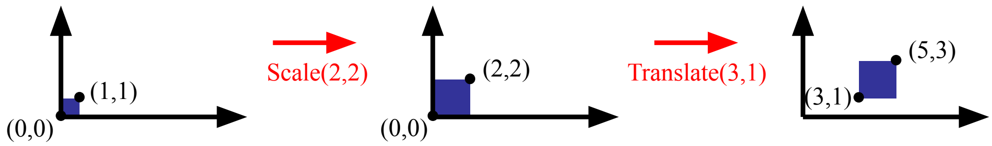


# Outline

- 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

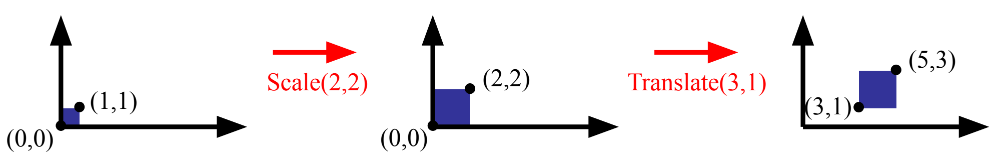
$$TS = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Caution: matrix multiplication is NOT commutative!

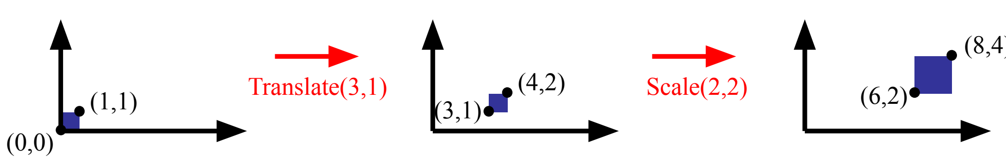


# Non-commutative Composition

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

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



# Worksheet!

**WebEx Breakout Sessions**
**(teams of 2 or 3)**
**Team upload to Submitty**

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

to th

*NOTE: We'll be doing pair worksheets throughout the term. We'll randomize the groups so you work with lots of different partners.*

Show your work.

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

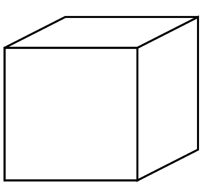
# Outline

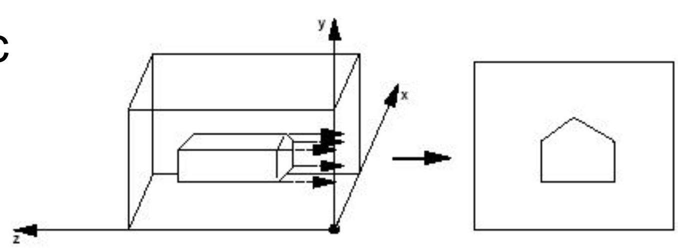
- Course Overview
- Classes of Transformations
- Representing Transformations
- Combining Transformations
- Orthographic & Perspective Projections
- Example: Iterated Function Systems (IFS)



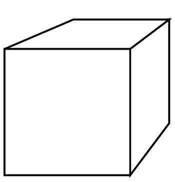
# Orthographic vs. Perspective

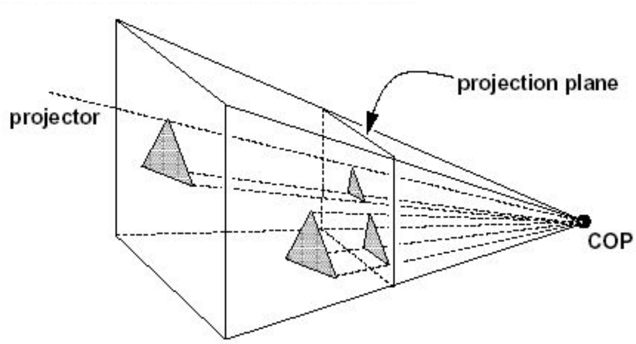
- Orthographic





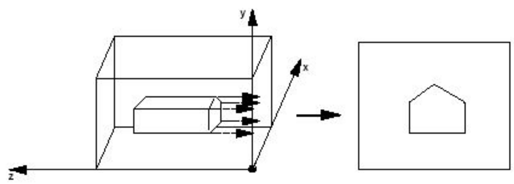

- Perspective





# Simple Orthographic Projection

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



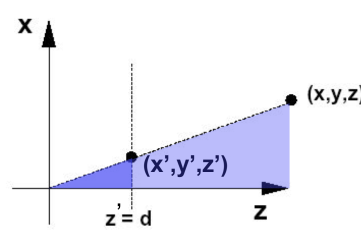

$$\begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



# 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



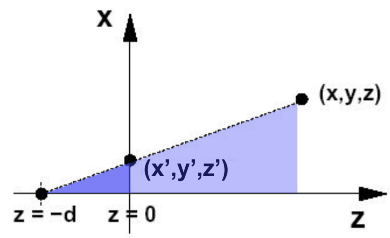

*homogenize*

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

# 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*

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

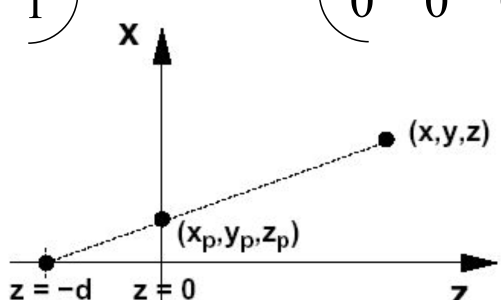


# In the limit, as $d \rightarrow \infty$

this perspective projection matrix...

...is simply an orthographic projection

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



# Outline

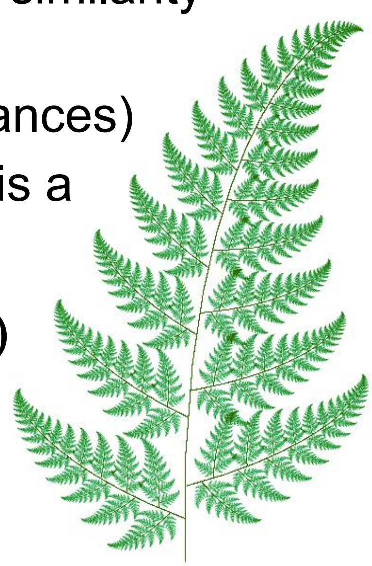
- Course Overview
- Classes of Transformations
- Representing Transformations
- Combining Transformations
- Orthographic & Perspective Projections
- Example: Iterated Function Systems (IFS)

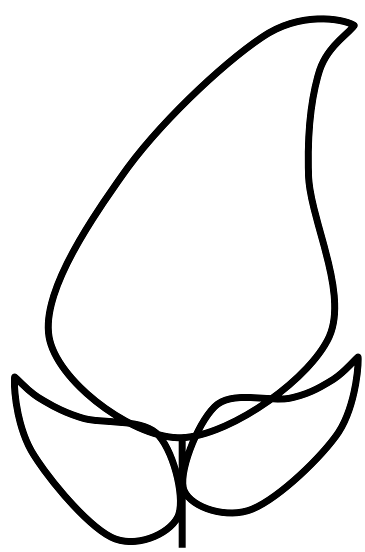


# Iterated Function Systems (IFS)

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

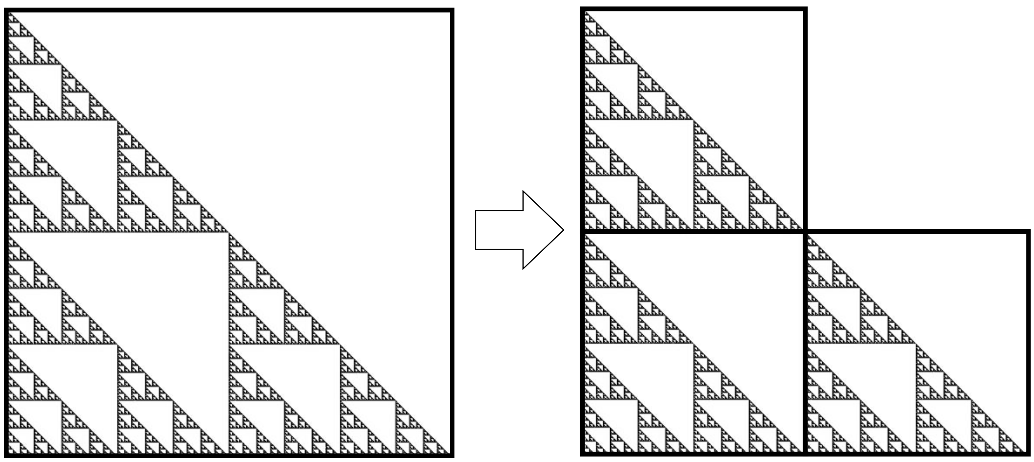
$$A = \bigcup f_i(A)$$





# Example: Sierpinski Triangle

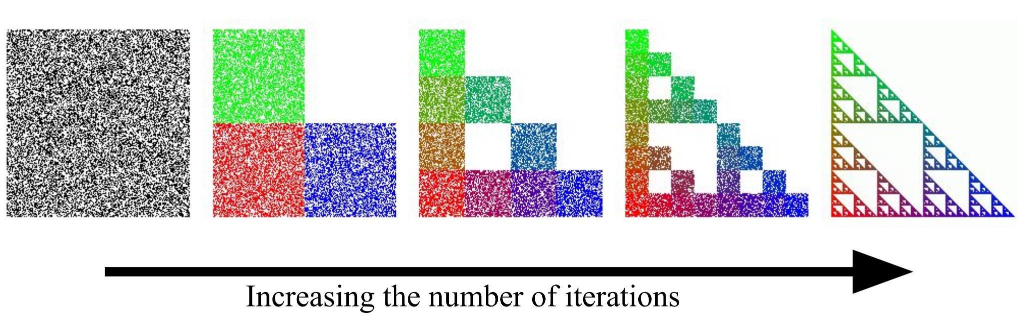
- Described by a set of *n* affine transformations
- In this case, *n* = 3
  - translate & scale by 0.5



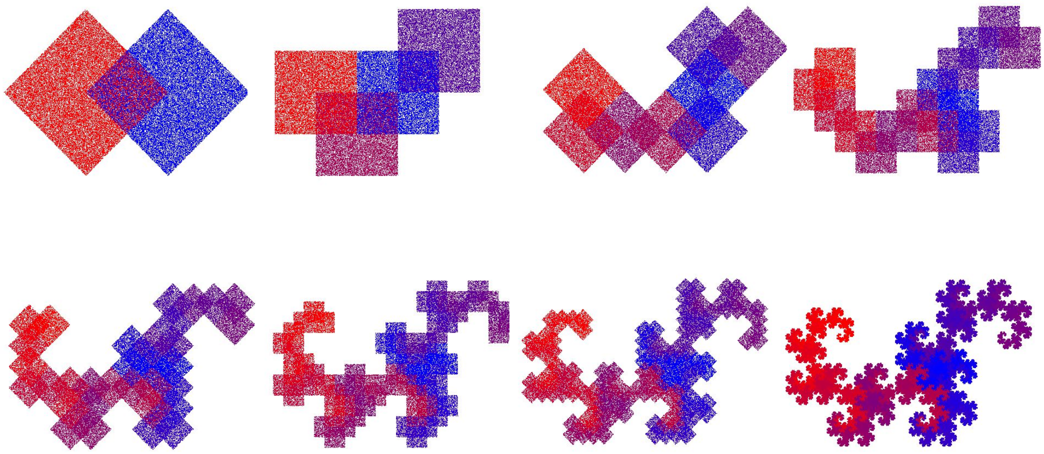


# Example: Sierpinski Triangle

```
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)
```

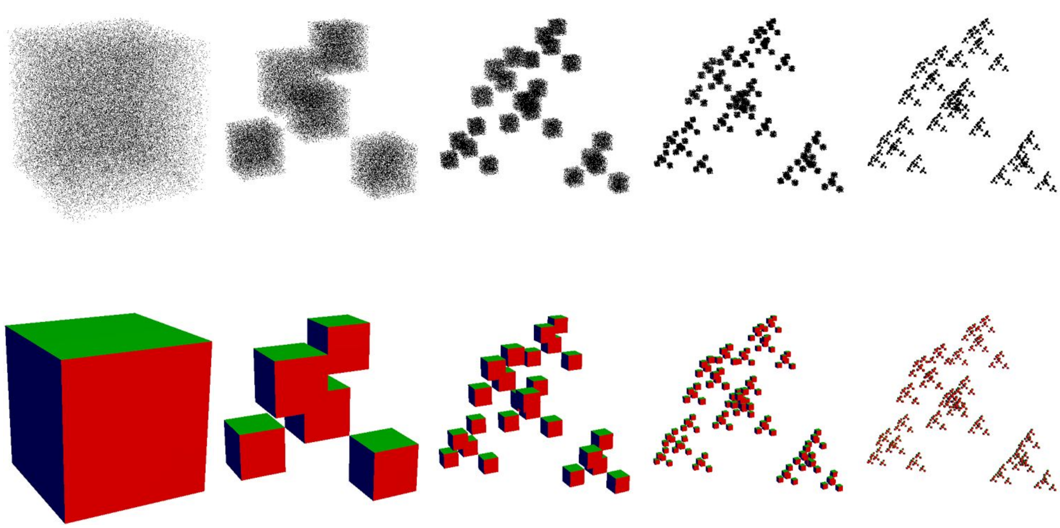


# Another IFS: The Dragon

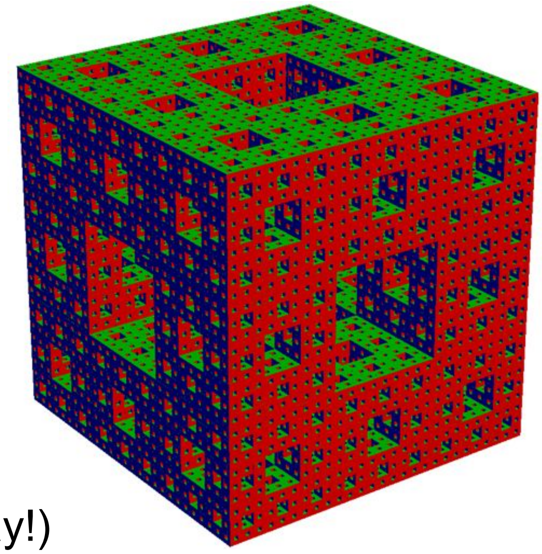




# 3D IFS in OpenGL / Apple Metal



# Assignment 0: OpenGL/Metal Warmup



- Get familiar with:
  - C++ environment
  - OpenGL / Metal
  - 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?

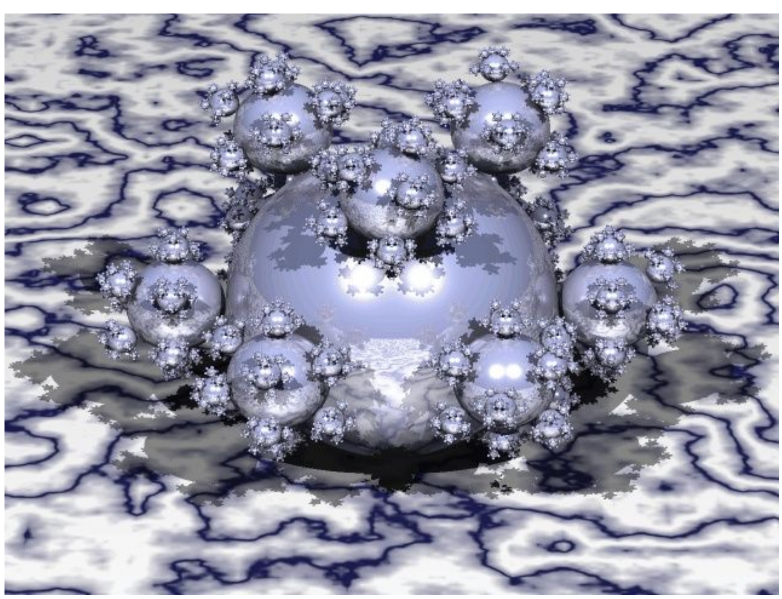
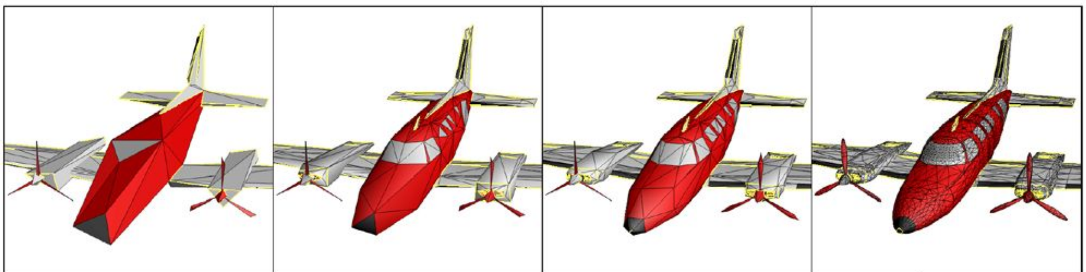


Image by Henrik Wann Jensen

# For Next Time:

We need 4 volunteers to be “Discussants”
*Note: This is not a “presentation”. Don’t make slides!*
*Be sure to read blurb (& linked webpage) on course webpage about Assigned Readings & Discussants.*

- Read Hugues Hoppe “Progressive Meshes” SIGGRAPH 1996
- Everyone will a comment or question on the course Submitty discussion forum before 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?