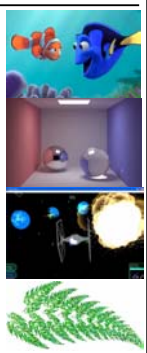
Luxo Jr

- Pixar Animation Studios, 1986
- Director: John Lasseter

Plan

- Introduction
- Overview of the semester
- Administrivia
- Iterated Function Systems (fractals)



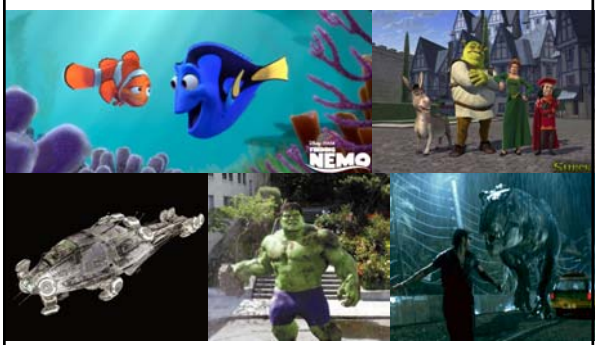
Team

- Lecturers
  - Frédo Durand
  - Barb Cutler
- TAs
  - Rob Jagnow
  - Patrick Nichols
- Course secretary
  - Bryt Bradley
- Staff email lists:
  - TAs only: 6.837-ta@graphics.csail.mit.edu
  - Staff: 6.837-staff@graphics.csail.mit.edu

Why Computer Graphics?

- Movies
- Games
- CAD-CAM
- Simulation
- Virtual reality
- Visualization
- Medical imaging

Movies



Digression

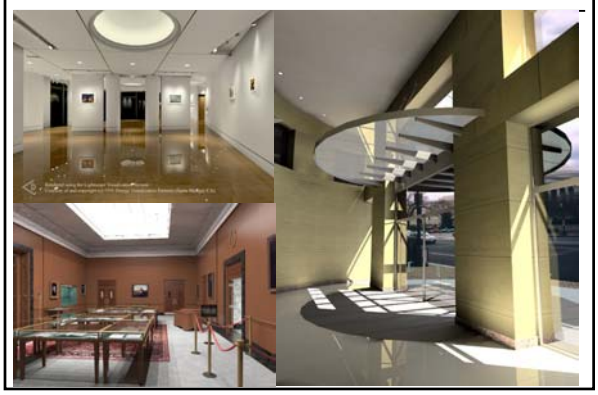
- George Borshukov, vfx technology supervisor , ESC entertainment (The Matrix)
  Will give a talk at MIT Nov 12

Games

Simulation

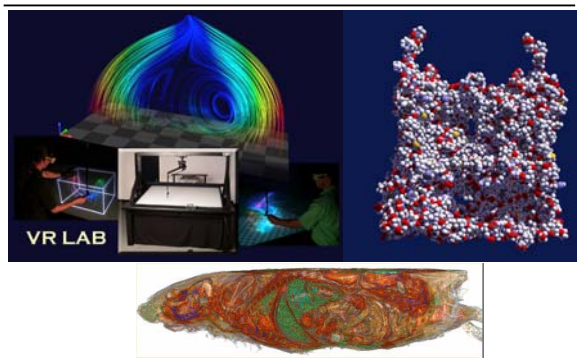
CAD-CAM & design

Architecture

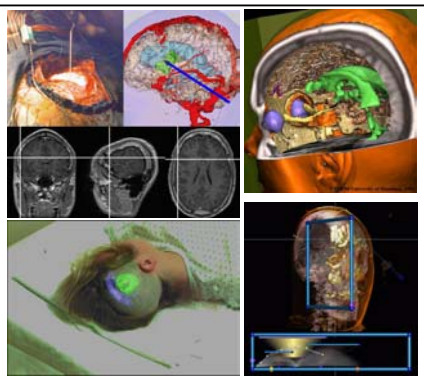


Virtual reality

Visualization



Medical imaging



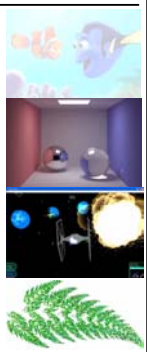
What you will learn in 6.837

- Fundamentals of computer graphics algorithms
- Able to implement most applications just shown
- Understand how graphics APIs and the graphics hardware work

What you will NOT learn

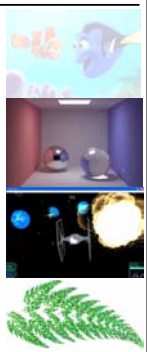
- Software packages
  - CAD-CAM
  - Photoshop and other painting tools
- Artistic skills
- Game design
- Graphics API
  - Although you will be exposed to OpenGL

Questions?



Plan

- Introduction
- Overview of the semester
- Administrivia
- Iterated Function Systems (fractals)

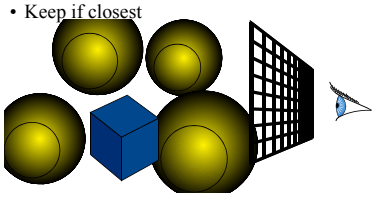


Overview of the semester

- Ray Tracing
  - Quiz 1
- Animation, modeling, IBMR
  - Choice of final project
- Rendering pipeline
  - Quiz 2
- Advanced topics

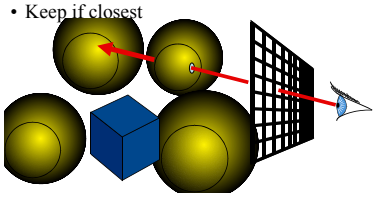
Ray Casting

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



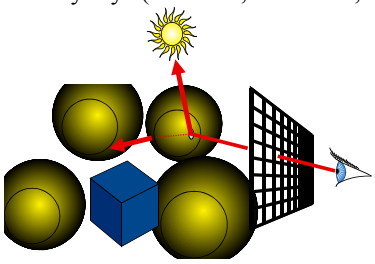
Ray Casting

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



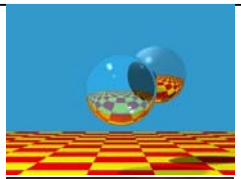
Ray Tracing

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



Ray Tracing

- Original Ray-traced image by Whitted

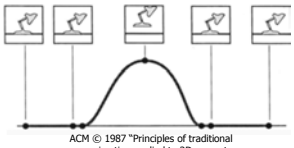


- Image computed using the Dali ray tracer by Henrik Wann Jensen
- Environment map by Paul Debevec

Overview of the semester

- Ray Tracing
  - Quiz 1
- Animation, modeling, IBMR
  - Choice of final project
- Rendering pipeline
  - Quiz 2
- Advanced topics

Animation: Keyframing



ACM © 1987 "Principles of traditional animation applied to 3D computer animation"

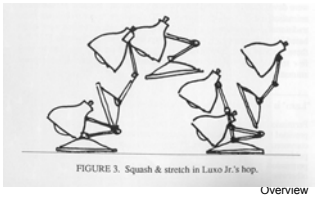
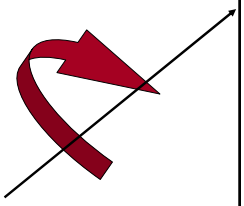
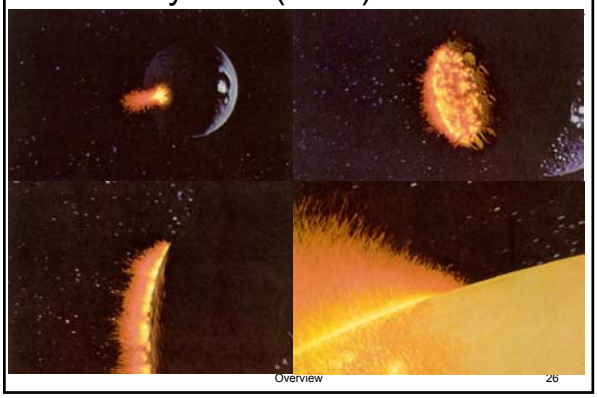


FIGURE 3. Squash & stretch in Luxo Jr.'s hop.

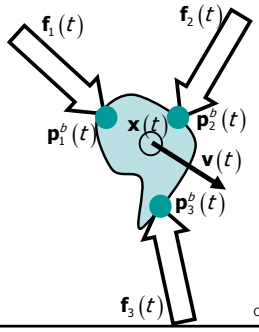


Particle system (PDE)



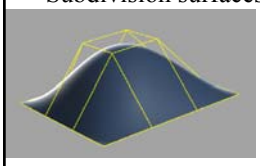
Rigid body dynamics

- Simulate all external forces and torques



Modeling

- Curved surfaces
- Subdivision surfaces



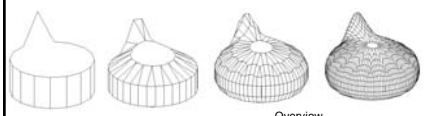


Image-based Rendering

- Use images as inputs and representation
- E.g. Image-based modeling and photo editing
  Boh, Chen, Dorsey and Durand 2001

Input image

New viewpoint

Relighting

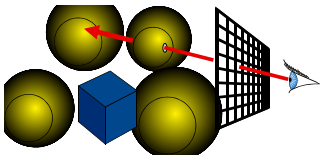
Overview of the semester

- Ray Tracing
  - Quiz 1
- Animation, modeling, IBMR
  - Choice of final project
- Rendering pipeline
  - Quiz 2
- Advanced topics

The Rendering Pipeline

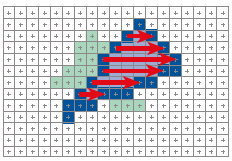
Ray Casting
- For each pixel
  - For each object

Send pixels to the scene



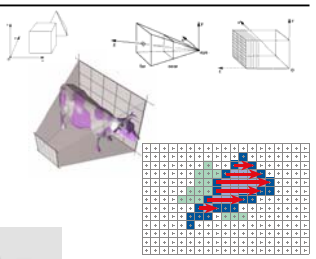
Rendering Pipeline
- For each triangle
  - For each projected pixel

Project scene to the pixels



The Rendering Pipeline

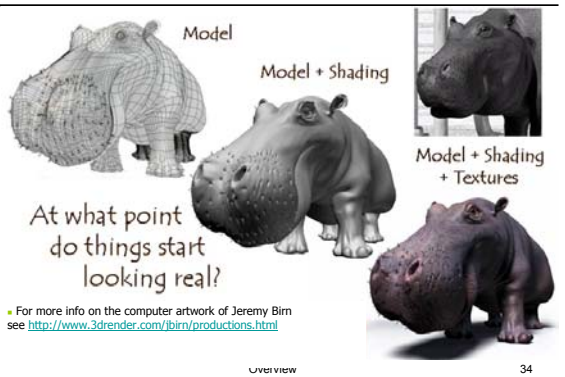
- Transformations
- Clipping
- Rasterization
- Visibility



Overview of the semester

- Ray Tracing
  - Quiz 1
- Animation, modeling, IBMR
  - Choice of final project
- Rendering pipeline
  - Quiz 2
- Advanced topics

Textures and shading



- For more info on the computer artwork of Jeremy Birn see http://www.3drender.com/jbirn/productions.html

shadows

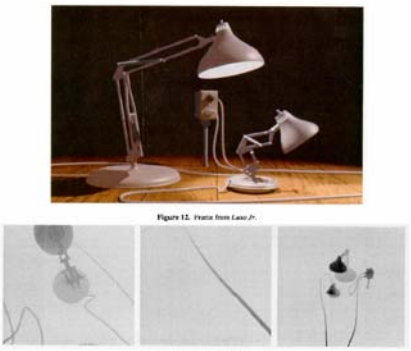
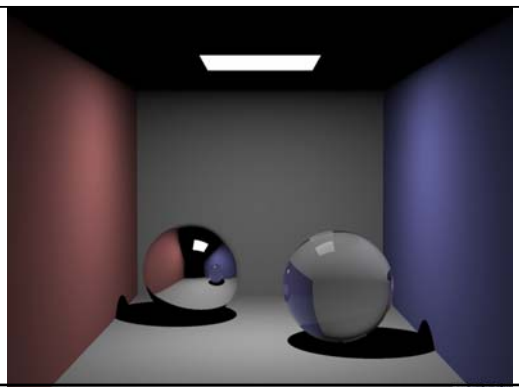


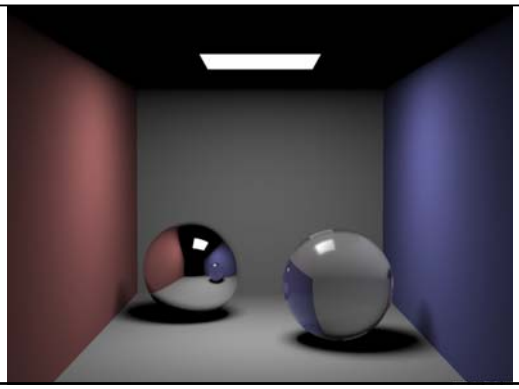
Figure 12. From Luxo Jr.

Figure 13. Shadow maps from Luxo Jr.

Traditional Ray Tracing

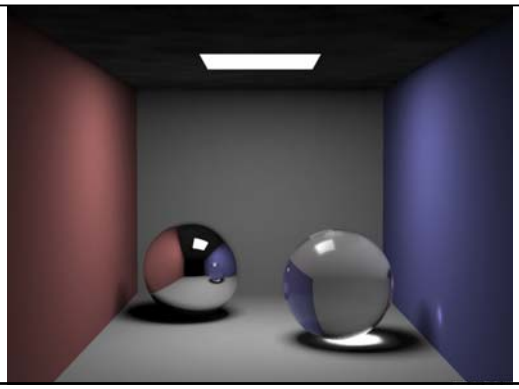


Ray Tracing+soft shadows



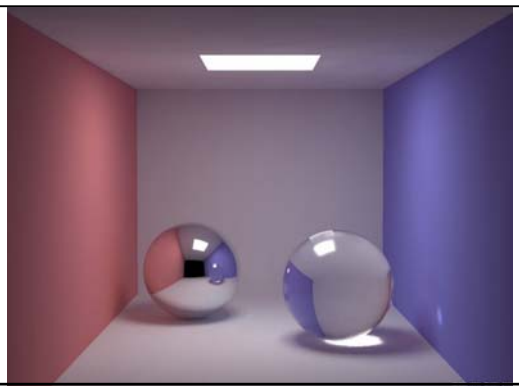
37

Ray Tracing+caustics



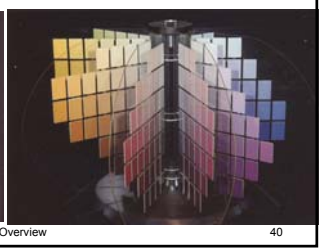
38

Global Illumination

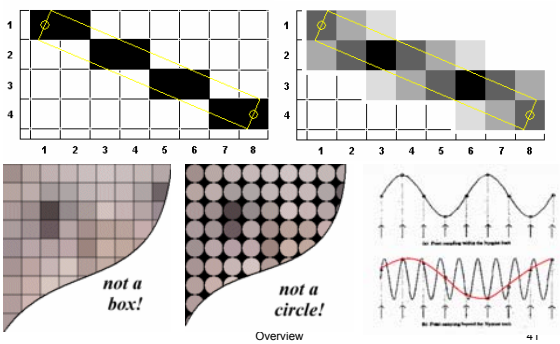


39

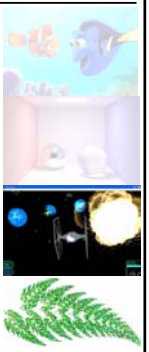
Color



Antialiasing

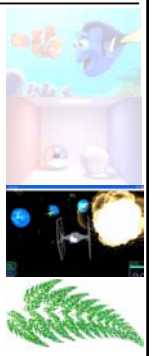


Questions?



Plan

- Introduction
- Overview of the semester
- Administrivia
- Iterated Function Systems (fractals)



Administrivia

- Web: http://graphics.csail.mit.edu/classes/6.837/F03/
- Lectures
  - Slides will be online
- Office hours
  - Posted on the web
- Review sessions
  - C++, linear algebra

Prerequisites

- Not enforced
- 18.06 Linear Algebra
  - Simple linear algebra, vectors, matrices, basis, solving systems of equations, inversion
- 6.046J Algorithms
  - Orders of growth, bounds, sorting, trees
- C++
  - All assignments are in C++
  - Review/introductory session Monday

Grading policy

- Assignments: 40%
  - Must be completed individually
  - No late policy. Stamped by stellar.
- 2 Quizzes: 20%
  - 1 hour in class
- Final project: 40%
  - Groups of 3, single grade for the group
  - Initial proposal: 3-5 pages
  - Steady weekly progress
  - Final report & presentation
  - Overall technical merit

Assignments

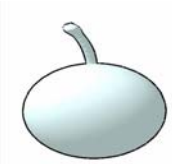
- Turn in code AND executable
- We will watch code style
- Platform
  - Windows
  - Linux
- Collaboration policy:
  - You can chat, but code on your own
- No late policy

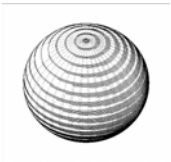
Project

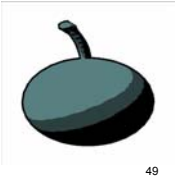
- Groups of 3
- Brainstorming
  - Middle of the semester
- Proposal
- Weekly meeting with TAs
- Report & presentation

Non-Photorealistic Rendering

- Eric Ho
- Philip Lee
- Vivienne Lee

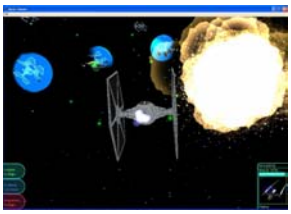






Simulating a Starship Battle

- Christopher Taylor
- Benji Sterling
- Jon Wolk



Bubble Trouble bobble 3D Challenge 3000

- Roger Hanna
- Ryan Williams
- Chris Elledge
- Paul Elliott

Dino

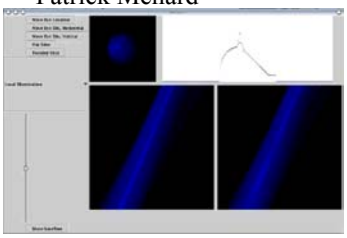
Windup

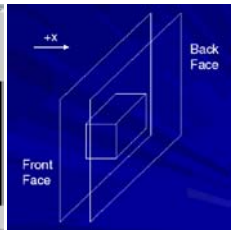
Alien

Whitemage

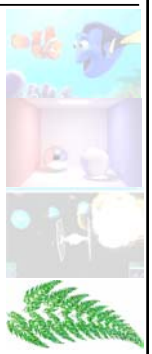
Selective Ray Recast For Optimized Rendering of Animation

- Bradley Lassey
- Ying Li
- Patrick Menard



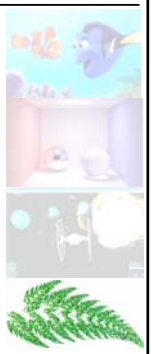


Questions



Plan

- Introduction
- Overview of the semester
- Administrivia
- Iterated Function Systems (fractals)



IFS: self-similar fractals

- Described by a set of *n* transformations *fi*
  - Capture the self-similarity
  - Affine transformations
  - Contractions (reduce distances)
- An attractor is a fixed point

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

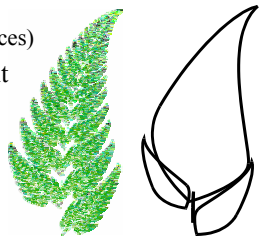
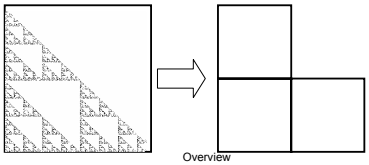


Image from http://spanky.triumf.ca/www/fractal-info/ifs-type.htm

Example: Sierpinsky triangle

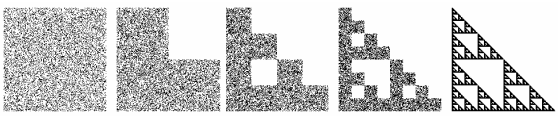
- 3 transforms
- Translation and scale by 0.5



Rendering

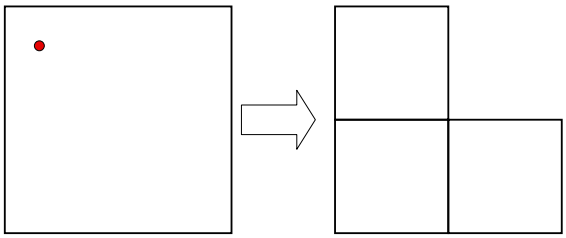
```
For a number of random input points (x0, y0)
    For j=0 to big number
        Pick transformation i
            (xk+1, yk+1) = fi (xk, yk)
    Display (xk, yk)
```

- Probabilistic application of one transformation



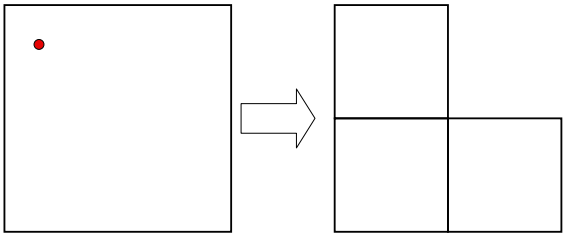
Example: Sierpinsky triangle

```
For a number of random input points (x0, y0)
    For j=0 to big number
        Pick transformation i
            (xk+1, yk+1) = fi (xk, yk)
    Display (xk, yk)
```



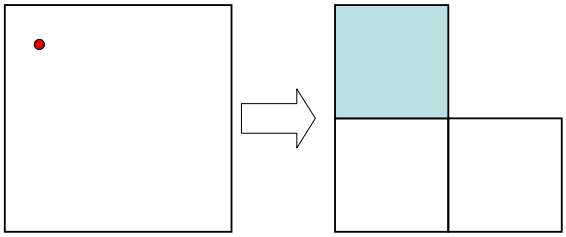
Example: Sierpinsky triangle

```
For a number of random input points (x0, y0)
    For j=0 to big number
        Pick transformation i
            (xk+1, yk+1) = fi (xk, yk)
    Display (xk, yk)
```



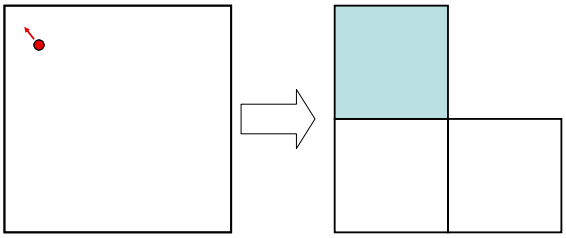
Example: Sierpinsky triangle

```
For a number of random input points (x0, y0)
    For j=0 to big number
        Pick transformation i
            (xk+1, yk+1) = fi (xk, yk)
    Display (xk, yk)
```



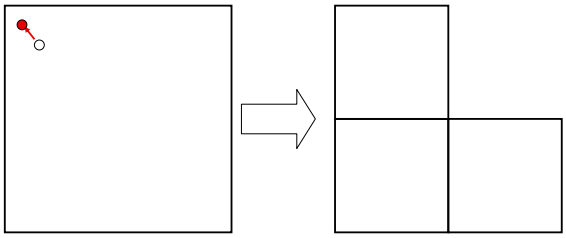
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



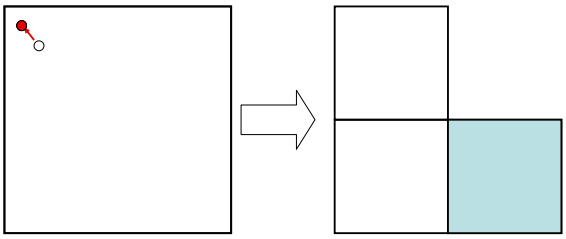
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



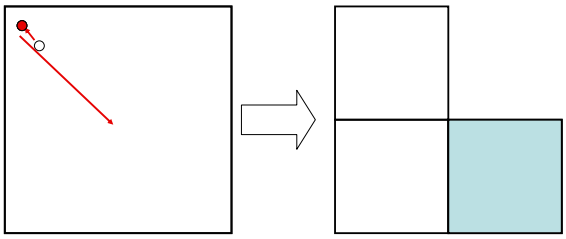
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



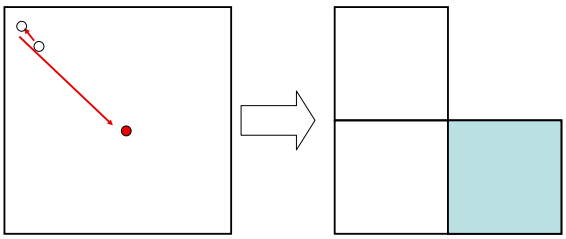
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



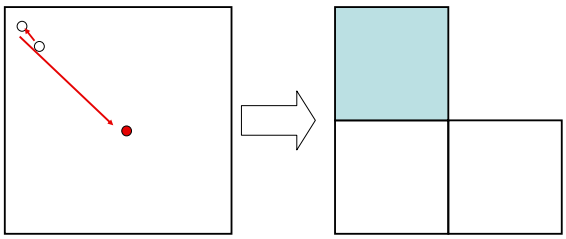
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



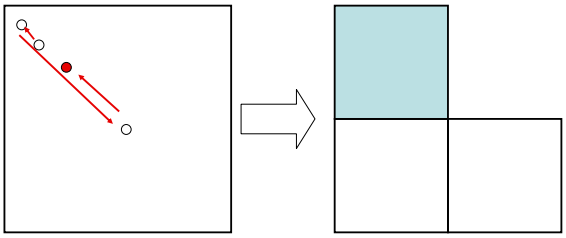
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



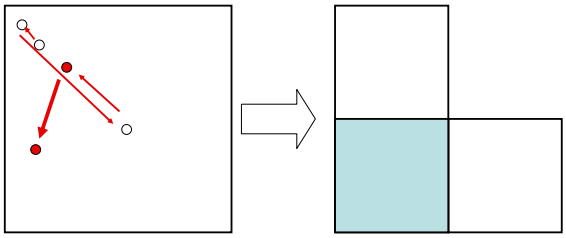
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



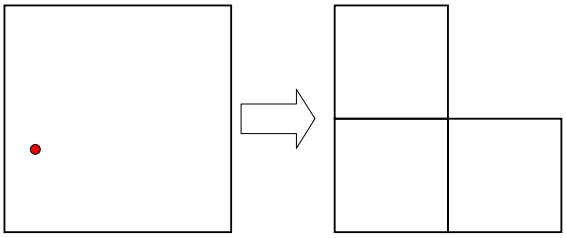
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



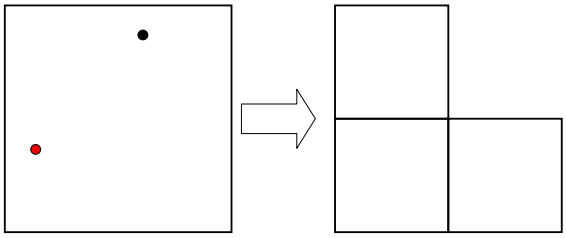
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



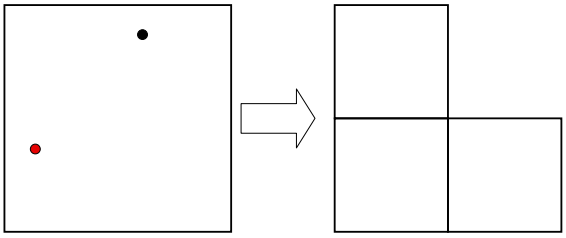
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



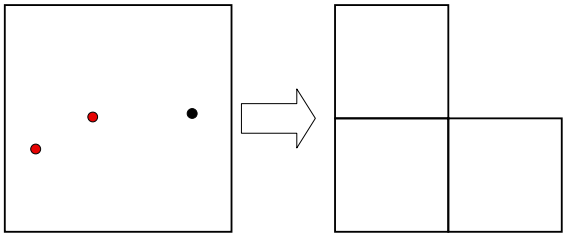
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



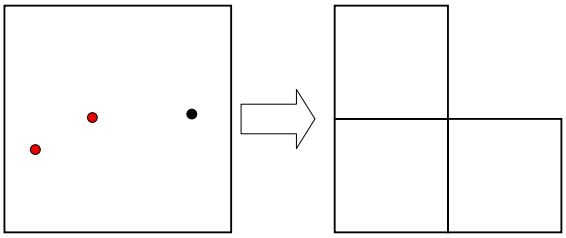
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



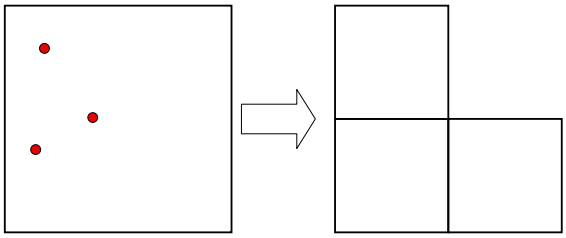
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



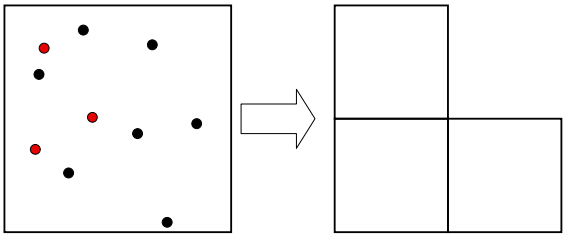
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



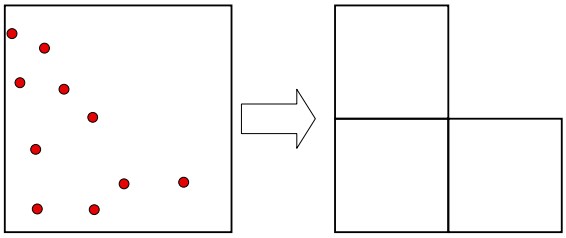
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



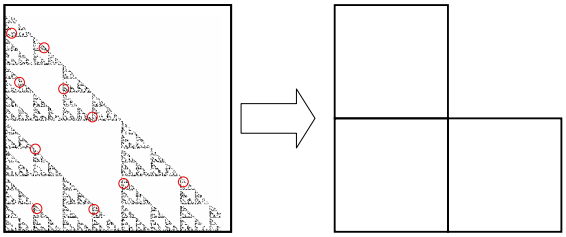
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



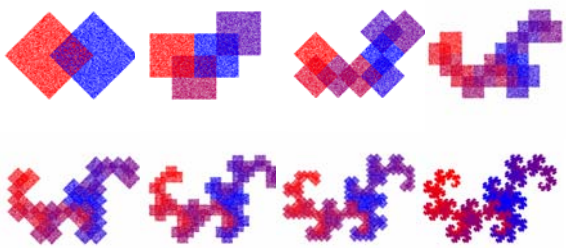
Example: Sierpinsky triangle

For a number of random input points (x_0, y_0)
  For j=0 to big number
    Pick transformation i
      $(x_{k+1}, y_{k+1}) = f_i\ (x_k, y_k)$
  Display (x_k, y_k)



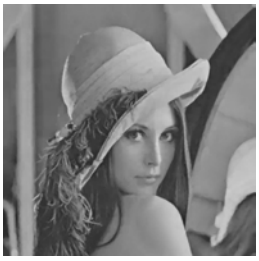
Other IFS

- The Dragon



Application: fractal compression

- Exploit the self-similarity in an image
- E.g. http://fractales.inria.fr/index.php?page=img_compression



Compressed using Fractal Photo Lab

Assignment: IFS

- Write a C++ IFS class
- Get familiar with
  - vector and matrix library
  - Image library
- Due Wednesday at 11:59pm

- Check on the web page

http://graphics.lcs.mit.edu/classes/6.837/F03/

Review/introduction session: C++

- Monday 7:30-9

Questions?