Computer Graphics

Dr. Scott Schaefer
Games
Movies
Visualization
Industrial Design
Mathematics and Graphics

*Graphics is mathematics made visible*

**Mathematics**
- Calculus
- Linear Algebra
- Differential Equations
- Real Analysis
- …

**Computer Science**
- Data Structures
- Searching
- Asymptotic Analysis
- Parallel Computing
- …
What are Fractals?

- Recursion made visible
What are Fractals?

- Recursion made visible
What are Fractals?

- Recursion made visible
What are Fractals?

- Recursion made visible
What are Fractals?

- Recursion made visible
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals
Rendering Fractals

\[ \text{Diagram showing the recursive nature of fractals with branches } T_1, T_2, T_3. \]
Rendering Fractals
Rendering Fractals

The diagram illustrates recursive fractal generation with transformations $T_1$, $T_2$, and $T_3$. The fractal structure is formed by applying these transformations repeatedly, as indicated by the arrows and recursive calls.
Fractal Tennis

Start with any point $x$

For ( $i=1; i<100; i++$ )

\[ x = T_{random}(x) \]

For ( $i=1; i<100000; i++$ )

\[ \text{draw}(x) \]

\[ x = T_{random}(x) \]
Fractal Tennis

Start with any point $x$

For ($i=1; i<100; i++$)

$x = T_{\text{random}}(x)$

For ($i=1; i<100000; i++$)

draw($x$)

$x = T_{\text{random}}(x)$

Gets a point on the fractal
Fractal Tennis

Start with any point $x$

For ( $i=1; i<100; i++$ )

\[ x = T_{random}(x) \]

For ( $i=1; i<100000; i++$ )

\[ \text{draw}(x) \]

\[ x = T_{random}(x) \]

Creates new points on the fractal
Fractal Tennis – Example

25,000 Points
Fractal Tennis – Example

50,000 Points
Fractal Tennis – Example

75,000 Points
Fractal Tennis – Example

100,000 Points
Fractal Tennis – Example

125,000 Points
Fractal Tennis – Example

150,000 Points
More Fractals
More Fractals
More Fractals
Parallel Processing Power

ATI’s Radeon HD 5870
1600 processors, 2.72 TFLOPS

IBM’s ASCI Red, 2.38 TFLOPS
Fastest Computer in the World 1999
Problem
Problem
Polygon Models

- Very fast to render
- Not smooth (faceted)
- High-resolution models require lots of band-width, computational resources, and memory
Goal: Higher Order Surfaces

- Eliminate faceting artifacts – no polygons
- Compact representation

Current  Future
A Little History

- **Xbox 360**
  - Launched Nov 22, 2005
  - GPU by ATI
  - Contained new feature called a “tessellator”
Tessellator Unit
Tessellator Unit
Subdivision Surfaces

- Used in movie and game industries
- Supported by most 3D modeling software

Toy Story © Disney / Pixar

Geri’s Game © Pixar Animation Studios
Subdivision Surfaces

- Used in movie and game industries
- Supported by most 3D modeling software
Approximate Catmull-Clark Patches

Fake

Real
Approximate Catmull-Clark Patches

Over 12X faster to evaluate!!!
Approximate Catmull-Clark Patches
Approximate Catmull-Clark Patches

Real

Fake
Approximate Catmull-Clark Patches
Approximate Catmull-Clark Patches

Real

Fake
DirectX 10 Pipeline

Input Assembler

Vertex Shader

Hull Shader

Tessellator

Domain Shader

Output Merger

Vertex Buffer

Index Buffer

Texture

Texture

Texture

Texture

Depth/Stencil

Render Target
Current Uses: GPU’s

DirectX 10.5 Nvidia Code Sample
Microsoft Demo part of DirectX 10 SDK
Current Uses: Video Games

- Better geometry
- Improved lighting effects
- More degrees of freedom for advanced geometry techniques
Current Uses: Movie Production
CSCE 441: Computer Graphics
CSCE 489: Computer Game Development

- Design and build a game over a semester
- Graphics, networking, AI, physics, software engineering, …
- Submit game to IGF competition
CSCE 645: Geometric Modeling
Graphics at TAMU

Jinxiang Chai
- Character Animation
- Data-driven graphics and vision
- Image-based rendering and modeling
- Image and video processing

John Keyser
- Robust Geometric Computation
- Geometric and solid modeling
- Physically-based simulation
- Scientific visualization

Scott Schaefer
- Geometric Modeling
- Deformation and Animation
- Surface reconstruction
- Scientific visualization