CSE167: Introduction to Computer Graphics

Lecture #17: Shadow Volumes, Procedural Modeling

Jürgen Schulze
University of California, San Diego
Fall 2009
Announcements

• Second Exam: Thursday, December 3\textsuperscript{rd}, HSS 1330, 2pm-3:20pm
• Hand-written double sided index card permitted
• Jason has extra index cards
Lecture Overview

Advanced shader effects

- Shadow volumes
- Procedural modeling
Shadow volumes

- Light source
- Shadowing object
- Shadow volume (infinite extent)
- Partially shadowed object
- Surface outside shadow volume (illuminated)
- Surface inside shadow volume (shadowed)

Eye position (note that shadows are independent of the eye position)
Shading with shadow volumes

• Many variations

• Stencil shadow volumes
  - Classic algorithm
  - Hard shadows

• Here, two-pass algorithm for approximate soft shadows
  - Very simple and inaccurate, but often plausible enough

• Many more complicated and more accurate variations exist
Shadow volume construction

- Need to generate shadow polygons to bound shadow volume
- Extrude silhouette edges from light source

Extruded shadow volumes
Shadow volume construction

- Needs to be done on the CPU
- Silhouette edge detection
  - An edge is a silhouette if one adjacent triangle is front facing, the other back facing with respect to the light
- Extrude polygons from silhouette edges
Two-pass algorithm

First pass

• Render scene without shading, store depth buffer

Second pass

• Use depth image from first pass for z-buffering

• Render polygons of shadow volume only

• For each rasterized pixel, compute a fractional shadow value in a fragment shader
Two-pass algorithm

Approximation

- Shadow value depends on distance to center of occluder, distance to center of shadow volume

Second pass

- Construct a local coordinate system for each shadow volume
  - Origin is center of occluder
  - z-axis is center of shadow volume
- Transform location of each pixel to shadow volume coordinate system
- Use z-coordinate and distance to z-axis to look up shadow texture
Demo

• Shadow mapping demo from

http://www.paulsprojects.net/opengl/shadowmap/shadowmap.html
Resources

• Overview, lots of links
  http://www.realtimerendering.com/

• Basic shadow maps

• Avoiding sampling problems in shadow maps
  http://www.cg.tuwien.ac.at/research/vr/lispsm/

• Faking soft shadows with shadow maps
  http://people.csail.mit.edu/ericchan/papers/smoothie/

• Alternative: shadow volumes
  http://www.gamedev.net/reference/articles/article1873.asp
Lecture Overview

Advanced shader effects

• Shadow volumes

• Procedural modeling
Modeling

• Creating 3D objects/scenes and defining their appearance (texture, etc.)

• So far we saw
  - Triangle meshes
  - Bezier patches

• Interactive modeling
  - Place vertices, control points manually

• For realistic scenes, need extremely complex models containing millions or billions of primitives

• Modeling everything manually extremely tedious
Alternatives

• Data-driven modeling
  - Scan model geometry from real world examples
  - Use laser scanners or similar devices
  - Photographs as textures

• Procedural modeling
  - Construct 3D models and textures using algorithms

• Examples
  - [Levoy et al.]
  - [Levoy et al.]
  - http://www-graphics.stanford.edu/data/3Dscanrep/
  - http://www.tsi.enst.fr/3dmodels/
  - .ply file format reader
    - http://www.tecgraf.puc-rio.br/~diego/professional/rply/
Procedural modeling

- Wide variety of techniques for algorithmic model creation

- Used to create models that are too complex (or tedious) for a person to build
  - Trees, landscapes (mountains), clouds, etc.
  - Plants, ecosystems
  - Buildings, cities

- Usually defined by a small set of data, or rules, that describes the overall properties of the model
  - Tree defined by branching properties and leaf shapes

- Model is constructed by an algorithm (procedure)
  - Often including randomness to add variety
  - E.g., a single tree pattern can be used to model an entire forest

[Deussen et al.]
Randomness

- Use some sort of randomness to make models more interesting, natural, less uniform, clean

- *Pseudorandom* number generation algorithms
  - Produce a sequence of (apparently) random numbers based on some initial seed value

- Pseudorandom sequences are repeatable, as one can always reset the sequence
  - E.g., if tree is built using several random numbers, then the entire tree can be rebuilt by just resetting the seed to its initial value
  - If the seed is set to a different value, a different sequence of numbers will be generated, resulting in a slightly different tree
Recursion

• Repeatedly apply the same operation (set of operations) to an object

• Generate objects that are self-similar, fractals
  - Object that look the same when viewed at different scales

• For example, the shape of a coastline may appear as a jagged line when we view a map of California
  - As we zoom in closer and closer, we see that there is more and more detail at finer scales
  - We always see a jagged line no matter how close we look at the coastline
Height fields

- Landscapes are often constructed as *height fields*
- Regular grid in the ground plane (assume xz plane here)
- Store a height (y) value at each point
- Can store large terrain in memory
  - No need to store xz coordinates, connectivity
- Shaped terrain by operations that modify the y coordinates
- Can interpret height values as gray scale values
  - Apply image processing tools
Fractal landscapes

- Random midpoint displacement algorithm
  - Recursively subdivide triangles
  - Randomly displace edge midpoints
  - Reduce size of displacement as triangles get smaller
  - Similar for quadrilaterals
Fractal landscapes

• Add textures, nice rendering...
• Terragen, free software

http://www.planetside.co.uk/terragen/

[http://www.planetside.co.uk/gallery/f/tg09]
L-systems

- Developed by a biologist (Lindenmayer) in 1968 to study growth patterns of algae
- Defined by a grammar

\[ G = \{V, S, \omega, P\} \]

- \( V \) alphabet, set of symbols that can be replaced (variables)
- \( S \) set of symbols that remain fixed (constants)
- \( \omega \) string of symbols defining initial state
- \( P \) production rules
Sierpinski triangle

- Variables: A, B
  - Draw forward
- Constants: +, -
  - Turn left, right by 60 degrees
- Start: A
- Rules: (A→B-A-B), (B→A+B+A)

2 iterations

4 iterations

6 iterations

9 iterations
Fractal fern

• Variables: X, F
  – X: no drawing operation
  – F: move forward

• Constants: +, –
  – Turn left, right

• Start: X

• Rules:
  (X \rightarrow F-[X]+X]+F[+FX]-X), (F \rightarrow FF)

• Stochastic L-system
  - If there is more than one production rule for a symbol, randomly choose one
Fractal trees

- Recursive generation of trees in 3D
  [http://web.comhem.se/solgrop/3dtree.htm](http://web.comhem.se/solgrop/3dtree.htm)
- Model trunk, branches as cylinders
- Change color from brown to green at certain level of recursion

Fractal tree  
Sierpinski tree
Algorithmic beauty of plants

- Online book on algorithmic beauty of plants by Prusinkiewicz
  [http://algorithmicbotany.org/papers/#abop](http://algorithmicbotany.org/papers/#abop)

Buildings, cities

Pascal Mueller
[http://www.vision.ee.ethz.ch/~pmueller/publications.html]
Exam

• Thursday during lecture slot