Lecture 22: Tuesday, 30 November 1999

Administrative:
- Final project reports due F 5pm; see posted requirements for report/presentation
- Schedule presentation with your TA ASAP
- HKN evaluation today (final 15 minutes)

Today:
- Visibility Computations:
  - List priority algorithms
  - Directional visibility
  - Hidden surface elimination
  - Visible surface identification
  - Additive/Subtractive visibility

Last Time

Acceleration of Ray Casting / Ray Tracing

Assumption: rendering based on directed sampling
- Thus, task was to answer sampling query AFAP

But ... other kinds of visibility queries arise:
- Is light source fully visible to a receiver?
- Is light source fully invisible from receiver?
- Can region of space be seen from viewpoint/region ?
- Must some object be drawn from viewpoint/region ?

Often, queries can be answered *conservatively*:
- In radiosity, one enumerates a superset of:
  - 1) Those source, receiver pairs that exchange energy
  - 2) The blockers between given source, receiver
- In visual simulation (architectural CAD, &c.):
  - If object visible, must answer "visible," but...
  - If invisible, can sometimes answer "visible" (why?)

Visibility approaches

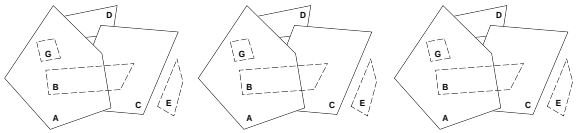
List priority orderings (BSP, painter's algorithm, Warnock)
- Visibility as back-to-front sort, or front identification

Directional culling (frustum culling)
- Spatial index, query mechanism for dynamic viewpt

Cell/portal culling ("additive" visibility)
- Augmented spatial index; sightline existence
- Useful in architectural interiors, biological vis., &c.

Occlusion culling ("subtractive" visibility)
- Finding (ideally small) proofs of invisibility
- Useful in urban exteriors, scientific visualization, &c.

Algorithm Classes



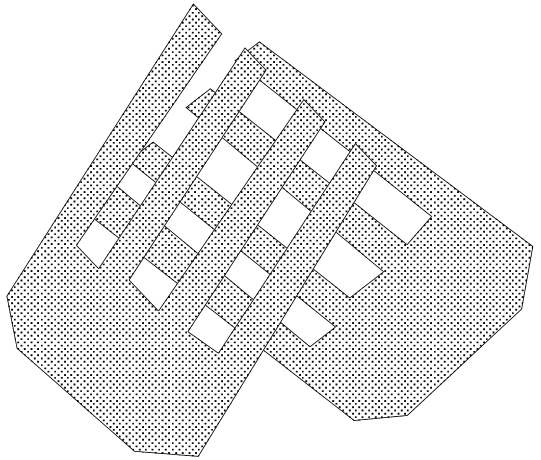
Analytic:
  Produces exact description of visible *fragments*
Ordering:
  Visible object is last one drawn at each pixel
Conservative / Superset algorithm:
  Produces upper bound on visible set
    Assumes z-buffer for per-pixel visibility
    Challenge: make upper bound usefully tight!
  Two families: additive and subtractive

Another Dimension of Variation

Object space: operates in continuous coordinates
  Often, can conclude visibility info about regions
Screen space: operates at screen resolution
  Typically, z-buffer based, requires viewpoint
Hybrid: some combination of both
Holy grail:
  Visibility algorithm for generic scenes
    Indoor scenes with evident structure
    Outdoor scenes with evident occluders
    Dense scenes with less obvious structure:
      Complex CAD model (engine compartment)
      Complex organic scene (forest, molecule)
      Statistical elements (fog, dust, snow, etc.)
  ... which is output sensitive, AND
  ... which allows predictive (region) visibility queries

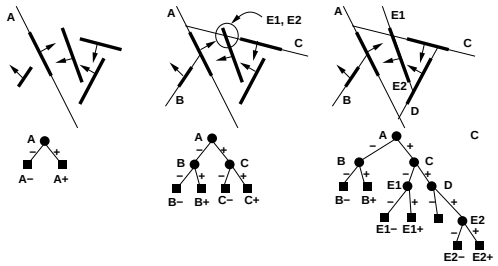
Analytic Algorithms

Analytic visibility:



Produces exact description of visible *fragments*
  Note: non-convex; descriptive complexity.
  Problems ?

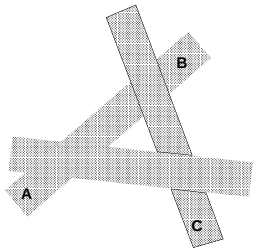
Ordering Algorithms



Ordering:

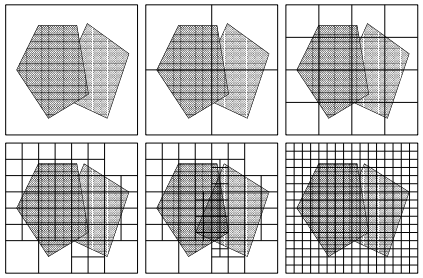
Visible object is last one drawn at each pixel

Problems ?



Classical Method: Warnock's Algorithm (1969)

Interesting object/screen-space hybrid



Attach all polygons to root viewport

Then, recursively:

Subdivide viewport, repopulate polygons

Termination conditions (0/1/other)?

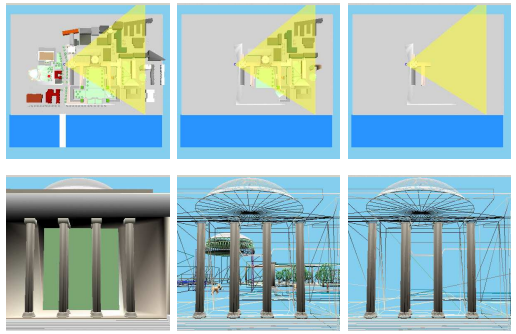
Conservative / Superset algorithm

Images courtesy Eric Brittain

Produces upper bound on visible set

Assumes z-buffer for per-pixel visibility

Problem: make upper bound usefully tight!



Classes of superset algorithm:

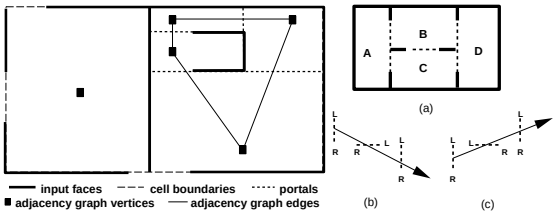
Additive: assume nothing visible; explore world

Useful (e.g.) in architectural interiors

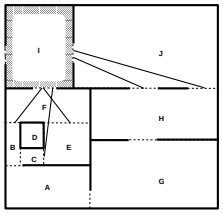
Subtractive: assume everything visible; remove occluded objects. Useful for exterior scenes

Additive Visibility: Cell/Portal Algorithms

Idea: subdivide along occluders into spatial cells



Search, opportunistically, for sightlines among cells

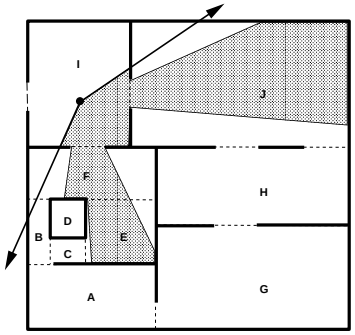


Determination of "cell visibility" can be:

- Done offline (as preprocessing)
- Done online, lazily (on demand by source cell), or ...
- Skipped entirely !

Cell/Portal Algorithms: Eye-based Visibility

Initialize constraint set to view frustum
Root a constrained, recursive search at cell



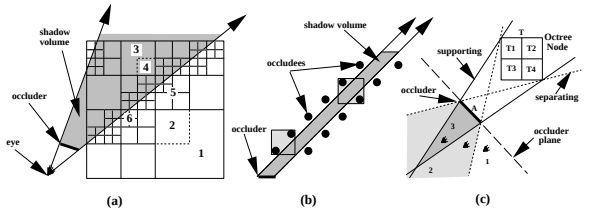
- Draw contents of current cell
- For each exiting portal:
  - Append associated constraints
  - If feasible, recurse to adjacent cell

Note treatment of "detail objects:"

- Ignored during index construction (how?)
- Culled hierarchically with active constraints

Subtractive Visibility: Occlusion Culling

First: can we use additive visibility here ?
Idea: identify large/effective occluders; use them to prove invisibility of detail objects



Note spatial, temporal coherence (how?)
Problem: detect compound occlusion/visibility!