Notes for April 8
Urs Bischoff and Nguyen Truong

Occlusion
Why? \(\Rightarrow\) Speed up rendering and transmission (reduce time of transfer)

We discussed from-point visibility “now” (as opposed to pre-computation). Tests can tell us
- Certainly hidden (conservative)
- Probably hidden (errors)

Tests cannot tell us whether something is visible or not.

Would it be better to fetch images instead of geometry? (for example, in transmission: is it better to send an image or geometry?). Answer depends on two main criteria:
- Transmission delay
  - First Frame + Additional Frames
    - Image: 1 million pixels x 30 f/s = 30 MB/s; cost of rendering on server; can be supported by: compression, incremental, video, sliding, reuse of previous image.
    - 3D geometry: crude model + refinement; new geometry can suddenly become visible (\(\Rightarrow\) pre-fetching)
- Rendering cost
  - Image: very fast
  - 3D geometry: expensive, but improved with occlusion and LOD

Geometry is better when things remain almost the same, but NOT when a portion suddenly appears.

Pose (author with idea described below)
The red dot is my position.

Idea:
- Size of squares are fixed
- Construct poster of the scene on the boundary of “green” squares
- Compose these images in real-time when I move
- If I move a little bit in direction of red arrow:
  - Objects far away don’t change (i.e. the outer green squares)
  - But close images have to be recomputed (red square)
  - Re-compute more images when I move farther (if a moved square gets too close to another one, you have to re-compute it)
- ⇒ Inner images have to be replaced often; but the farther away the images, the less often you have to re-compute it.

This system solves the rendering problem of the server, but not the transmission problem. We cannot compress (low-res) images, because exactly those images that have to be represented often are very close to the viewpoint.

It might be possible to combine geometry + images:
- for short distances, use geometry
- images are better for things that are far away

Geometry: coherent for neighborhood
Images: coherent for background

**Another Approach (Levoy)**

Sample the scene around the viewpoint. Then, replace certain parts (complex geometry) with images / textures. ➔ combine images with depth
Some geometry (depth) + images
Try to reuse previous information and update only what is really necessary, refine until quality is good enough

**From-point visibility principle**

- Convex occluders: If A & B are occluded by convex occluder, then edge AB is too. Occludee doesn’t have to be convex, the convex hull of the occludee is covered. We only require occluders to be convex

**Occlusion Fusion**

- Connected set of triangles: If we have a lot of small occluders (e.g. triangle mesh), we want to use them together as one big occluder. Collection forms a cone
- Front to back shadow:
The following illustration only shows the green frame. All objects that are in front of this screen are projected onto it (you can see the red pixels that are rendered on this screen, they are occluded). Can we use this information to find out whether object A is occluded?

Is A occluded by “shadow screen”? \(\Rightarrow\) Render a simple shape containing A; if hidden: don’t render A. What’s the catch? It is waiting for an answer, which may take a while.

We can use the z-buffer. We render the front-objects in the buffer. Then, the question is whether one pixel of A is rendered if we also render it in this frame (using depth information). If one pixel of A is rendered, then A isn’t occluded.

This method could be used as a multi-layer approach. Sort front objects and compute this frame for several depth-ordered layers. Use first layer to build a frame and test the rest (for occlusion). Then, render other layers and do the same. Previously occluded layers don’t have to be rendered.

From last frame remember the big occluders
Render them first
(Render bounding boxes for big occluders with feedback to kill false occluders)
Render bounding boxes for all other objects and use hardware occlusion feedback
Render those that pass occlusion
Collect best occluders

**Next class:** What about pre-computing occlusion?