VGP353 – Week 4

- Agenda:
  - More shadow maps:
    - Resolution matched shadow maps
    - Omni-directional lights

9-August-2009

Fixing Shadow Map Aliasing

- Recall from last week the two sources of aliasing:
  - Perspective aliasing – Comes from the relative orientation and distances of the light and camera
  - Projective aliasing – Comes from the relative orientation of surfaces, camera, and light
- PSMs and PSSMs only handle perspective aliasing
  - Projective aliasing would require expensive analysis of the scene

Fixing Shadow Map Aliasing

- Adaptive shadow maps (ASM) resolve both using a hierarchical data structure
  - Maps are stored in an adaptive quadtree
  - Tree is built iteratively

Adaptive Shadow Map Construction

```
render low-resolution shadow map
do
    for all camera pixels:
        Calculate (s, t, z, l) shadowmap coordinate and LOD
        Lookup shadow map texel
        If texel on shadow edge & page not in ASM:
            Convert (s, t, z, l) to page request
        Transfer page requests to CPU
        Remove invalid page requests
        Generate unique page requests
        Allocate new page in quadtree
        Bin requests into superpages
        Render shadow data into superpages
        Copy shadow data from superpage to quadtree
until page requests == 0
```

Adaptive Shadow Map Construction

- Problems:
  - The edge finding algorithm is EXPENSIVE!
  - The edge finding algorithm can miss some fine shadows

Resolution Matched Shadow Maps

- Store quadtree in a two part structure:
  - Store page table in a mipmap texture
    - LOD specifies the quad tree level
    - The s and t coordinates specify position in quadtree level
    - Value stored specifies location of page in second part of structure
  - Store pages in a single, large texture

RMSM Construction

```
for all pixels in image rendered from camera do
    calculate (s, t, z, l) shadowmap coordinates and LOD
    convert (s, t, z, l) to shadow page request
eliminate redundant requests via connected-components
eliminate invalid requests (compaction)
sort page requests
compact again to generate unique page requests
transfer unique page requests to CPU
allocate new page in quadtree
bin requests into superpages
render shadow data into superpages
copy shadow data from superpage to quadtree memory
```

RMSM Construction

- Phase one:
  - At each pixel calculate (s, t, z) for shadowmap lookup
  - Calculate LOD, l, as:

$$dX = \left(\frac{\partial s}{\partial x}, \frac{\partial t}{\partial x} \right)$$

$$dY = \left(\frac{\partial s}{\partial y}, \frac{\partial t}{\partial y} \right)$$

$$A = \left| dX \times dY \right|$$

$$l = \log_2 \left(\sqrt{A} \right)$$

RMSM Construction

- Eliminate redundant requests:
  - Mark only requests whose below and left neighbors request different pages
- Compact list:
  - Remove all unmarked page requests
  - See [Lefohn et al. 2006]
- Sort list of requests
  - See [Govindaraju et al. 2006]
- Remove all non-unique requests
- Transfer list to CPU

RMSM Construction

- Phase two:
  - Generate quadtree structure on the CPU
  - Merge (bin) requested pages into 1024x1024 “super pages”
  - Render super pages
  - Copy subsections of super pages into final data structure

References

Govindaraju, N. K., Gray, J., Kumar, R., and Manocha, D. 2006. GPUTeraSort: High performance graphics coprocessor sorting for large database management. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*. 325–336.

Lefohn, A. E., Kniss, J., Strzodka, R., Sengupta, S., and Owens, J. D. 2006. Glift: Generic, efficient, random-access GPU data structures. *ACM Transactions on Graphics*, vol. 26, no. 1, pages 60–99.

Lefohn, A. E., Sengupta, S., and Owens, J. D. 2006. "Resolution Matched Shadow Maps." *ACM Transactions on Graphics*, vol. 26, no. 4, pages 20:1--20:17. ACM, 2007. http://www.idav.ucdavis.edu/publications/print_pub?pub_id=919

Omni-directional Lights

- A single shadow map only works with single light frustum
  - Omni-directional lights inside the view frustum don't have a single light frustum
  - Ditto for hemispherical lights

Omni-directional Lights

- Obvious solution?
  - Render multiple views from the light to cover the whole scene
- Problems?
  - Many rendering passes → slow
  - Difficult to determine which shadow maps apply to which objects
- Solution?
  - Use a different parameterization to cover the scene

Omni-directional Lights

- Remind you of anything?
  - Environment maps
  - Same possible solutions:
    - Spherical
    - Cube
    - Dual-paraboloid
    - etc.

Dual-Paraboloid Shadow Maps

- View of environment as reflected by a convex parabolic mirror
  - The *outside* of a satellite dish
  - Reflects 180° of the environment
    - Capture 360° by using two maps
    - Known as dual paraboloid
  - Fairly similar to a hemispherical reflection map

Dual-Paraboloid Shadow Maps

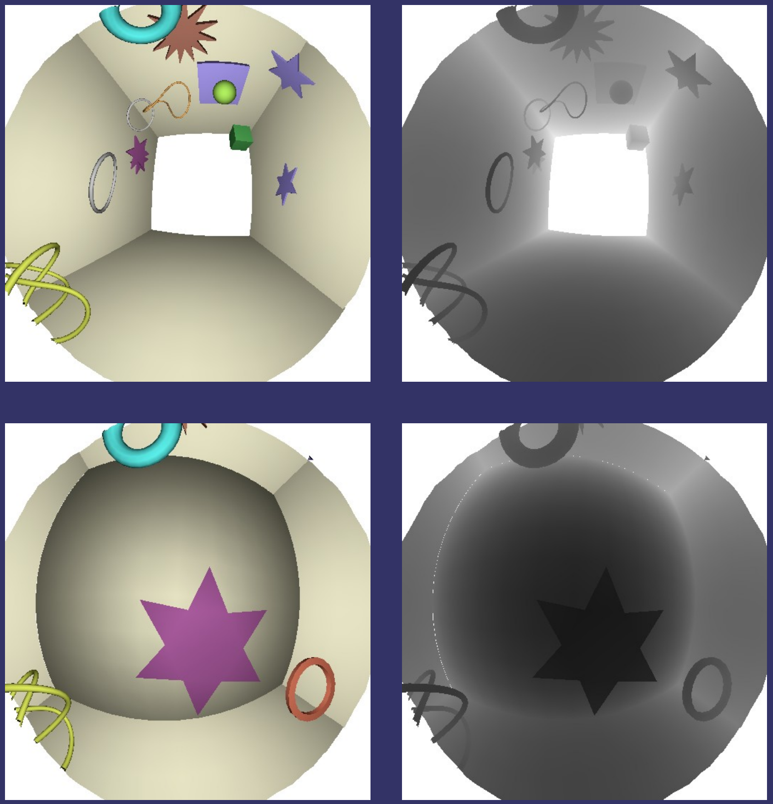
- Easily convert reflection vector to 2D texture coordinate for paraboloid map:

$$\begin{pmatrix} s \\ t \\ 1 \\ 1 \end{pmatrix} = A \cdot P \cdot S \cdot M_n^T \cdot R^T$$

$$A = \begin{pmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, S = \begin{pmatrix} -1 & 0 & 0 & d_x \\ 0 & -1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - d is the view direction vector
    - { 0 0 1 } or { 0 0 -1 } depending on the viewing direction
  - M_n is the transformation matrix for normals

Dual-Paraboloid Shadow Maps



Original image from http://www.mpi-inf.mpg.de/~brabec/doc/talk_cgi02.ppt

Dual-Paraboloid Shadow Maps

- From view point of reflector:
  - Draw two images
  - Transfrom vertices as usual but:
    - Divide X, Y, and Z by W
      - Call the magnitude of this vector L
    - Normalize and divide X and Y by (Z + 1)
    - Set Z to L remapped to view volume
      - Usual [0, 1] mapping based on near / far
    - Set W to 1.0

References

Stefan Brabec, Thomas Annen, and Hans-Peter Seidel, “Shadow Mapping for Hemispherical and Omnidirectional Light Sources”, *Computer Graphics International (CGI) 2002* proceedings, Bradford, UK. http://www.mpi-inf.mpg.de/~brabec/

http://opengl.org/resources/code/samples/sig99/advanced99/notes/node184.html

Jason Zink. “Dual Paraboloid Mapping in the Vertex Shader.” GameDev.net, 1996. http://www.gamedev.net/reference/articles/article2308.asp

Wolfgang Heidrich and Hans-Peter Seidel. “View-independent environment maps.” In *Proceedings of the SIGGRAPH/Eurographics Worksjhop on Graphics Hardware*, 1998. http://www.cs.ubc.ca/~heidrich/Papers/GH.98.pdf

Next week...

- Stencil-volume shadows
- Quiz #2

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of Intel or the Art Institute of Portland.

OpenGL is a trademark of Silicon Graphics, Inc. in the United States, other countries, or both.

Khronos and OpenGL ES are trademarks of the Khronos Group.

Other company, product, and service names may be trademarks or service marks of others.