The Holodeck
Interactive Ray Cache

Greg Ward, Exponent

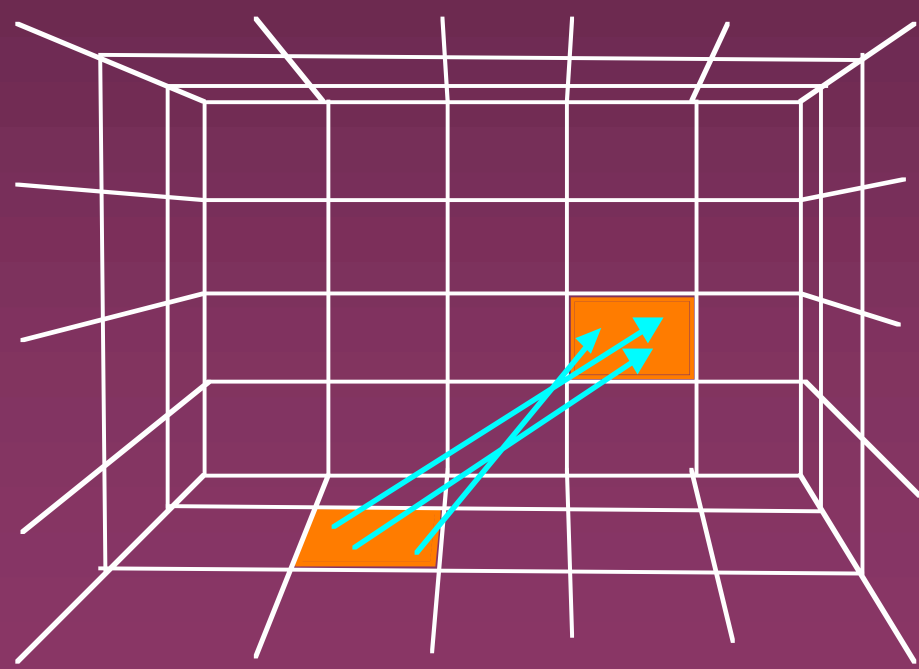
Maryann Simmons, UCB

The Driving Vision

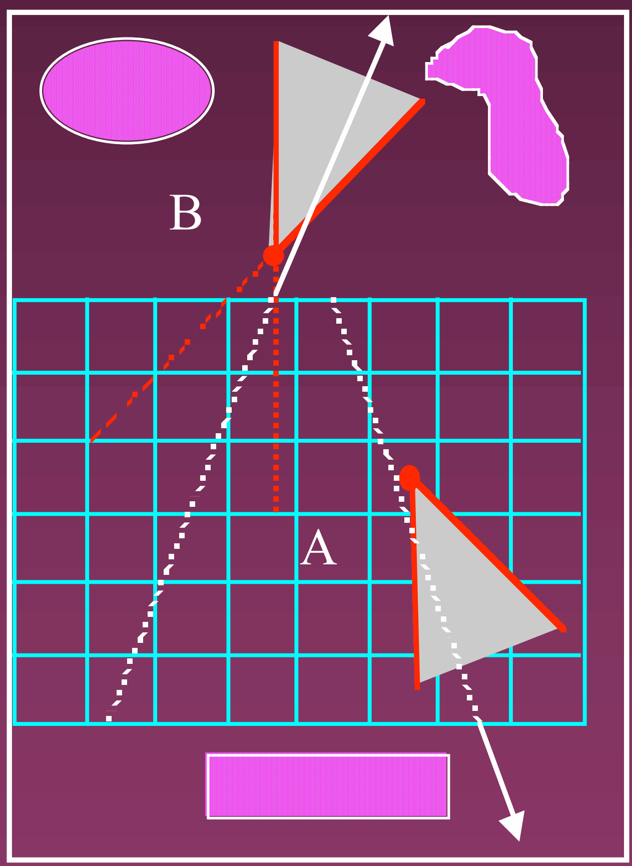
- Ray tracing with interactive display
- Accurate global and local illumination
- Interactive walk-throughs of non-diffuse environments
  - » display representation for motion feedback
- No wasted computation
  - » i.e., don't throw away what we'll want again

The Holodeck Ray Cache

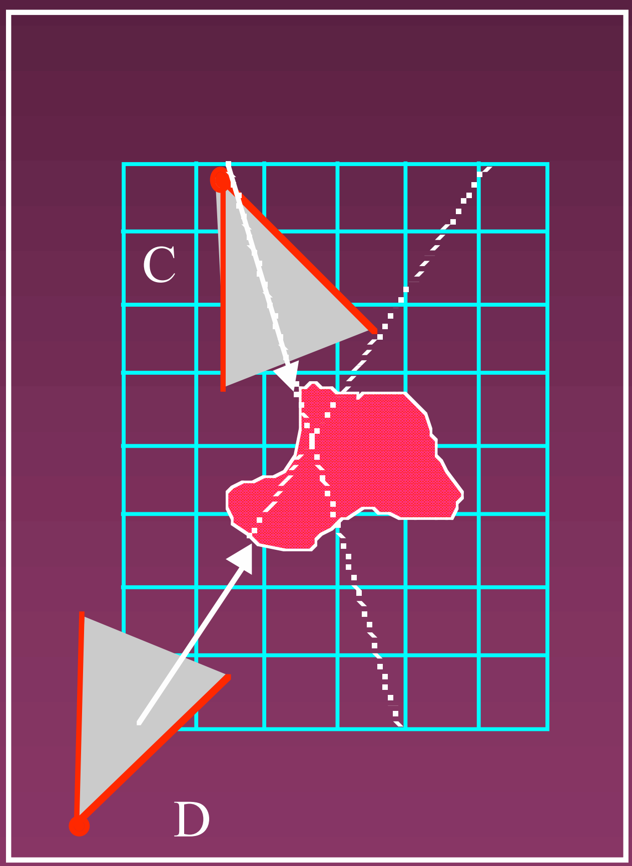
- Rays bundled in *beams* going in one *cell* and out another on a *section wall*



View-Beam Correspondence

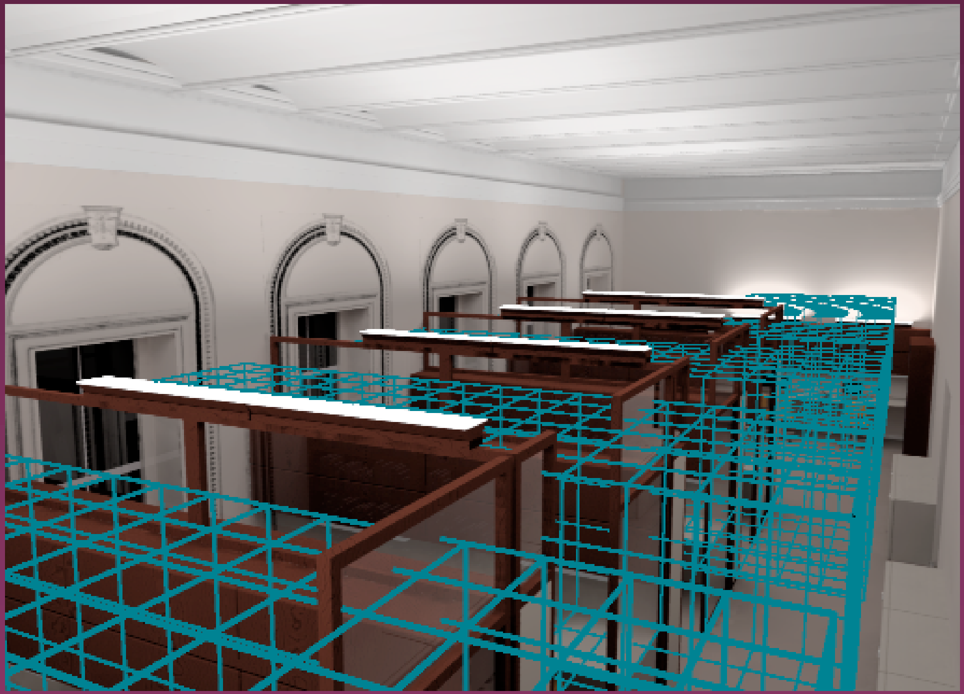


“Internal” Section



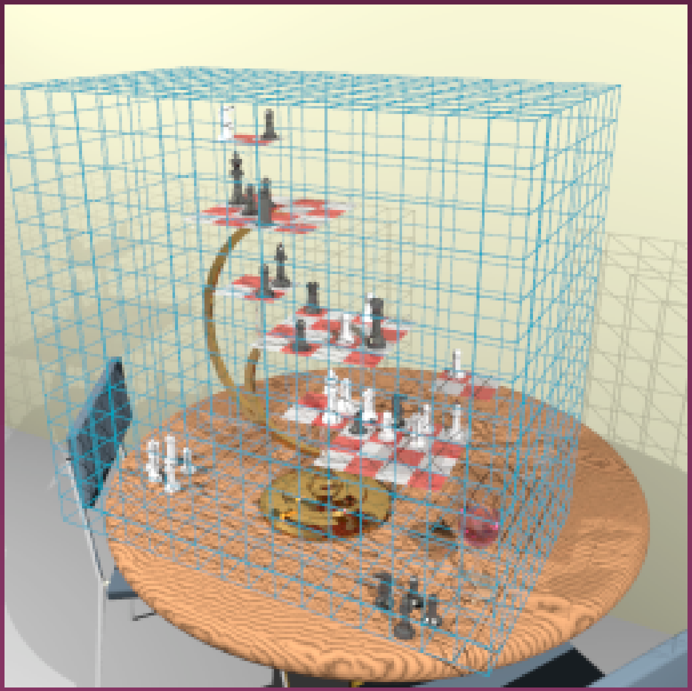
“External” Section

Example Holodecks



Internal Sections

(view from inside)



External Section

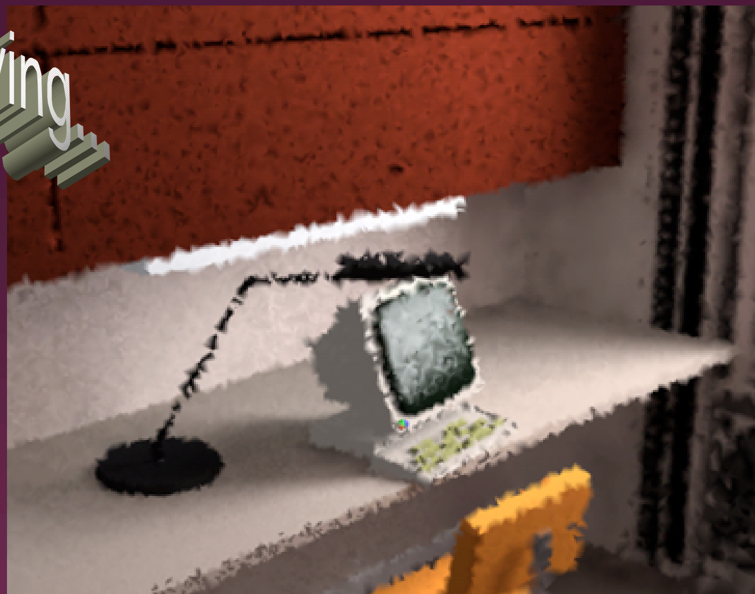
(view from outside)

Progressive Rendering

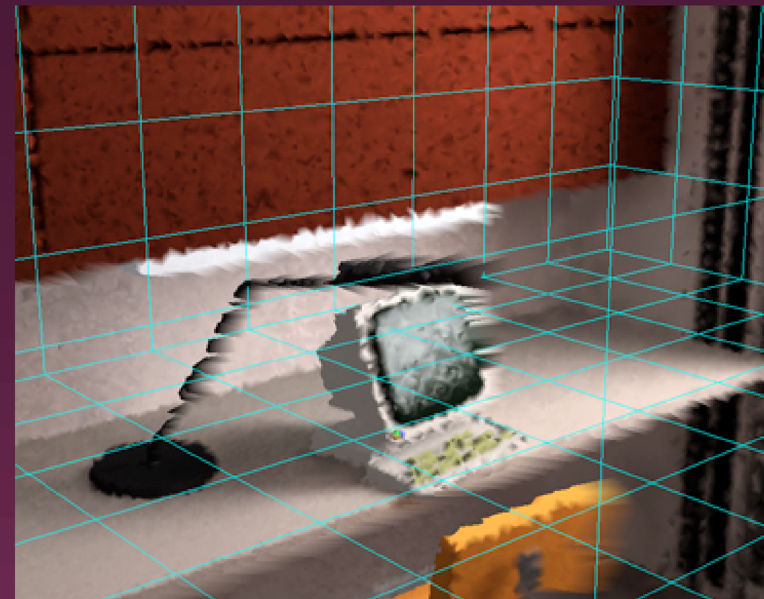
SP Octane after 10 seconds...

after 1 minute

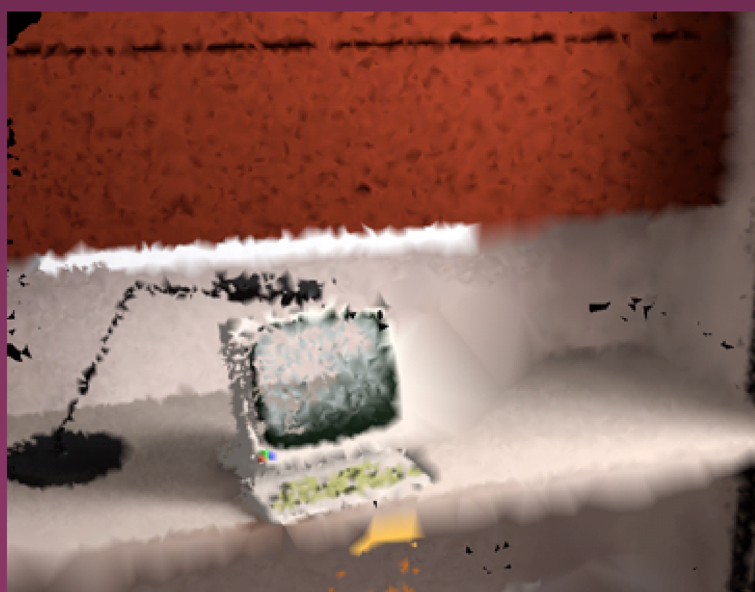
Moving



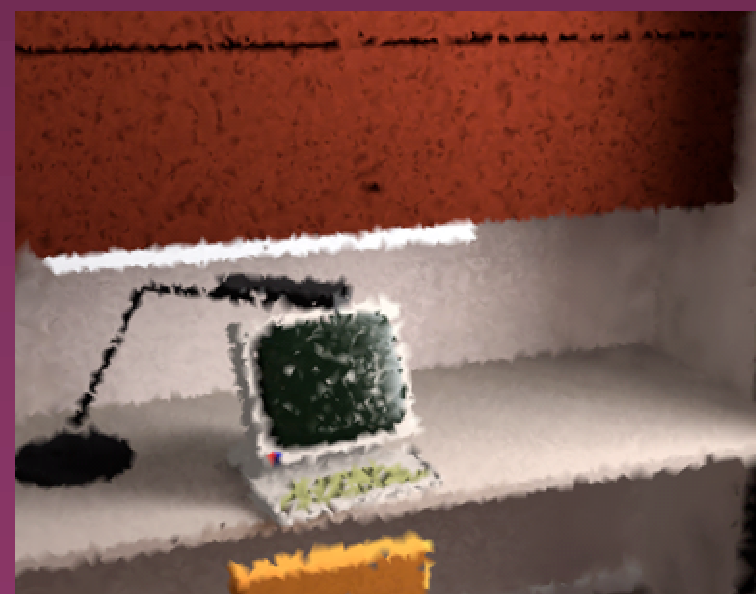
starting view



begin move...



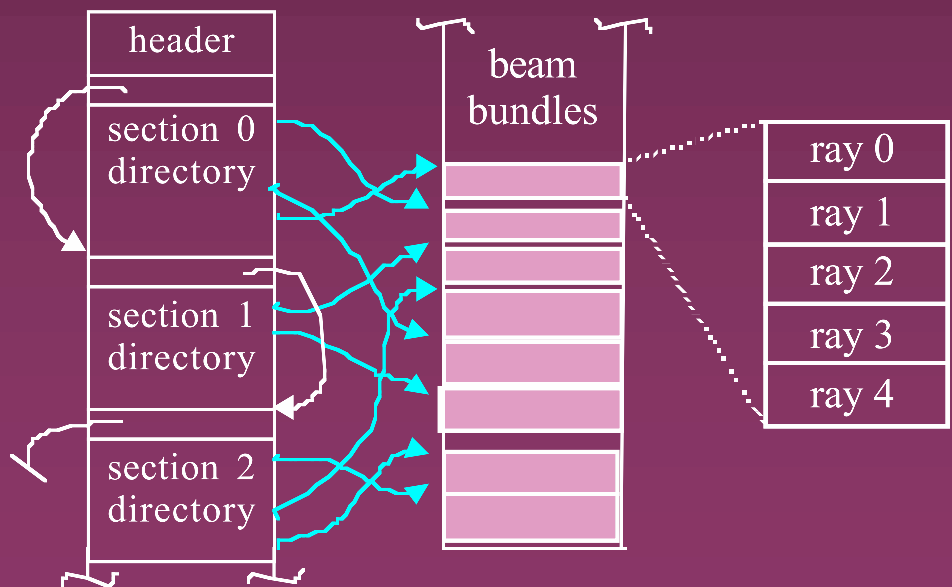
...end move



new samples come in

Holodeck File Structure

- Holodeck may have multiple sections
  - » each section has its own directory
  - » each beam sample (ray) takes 10 bytes

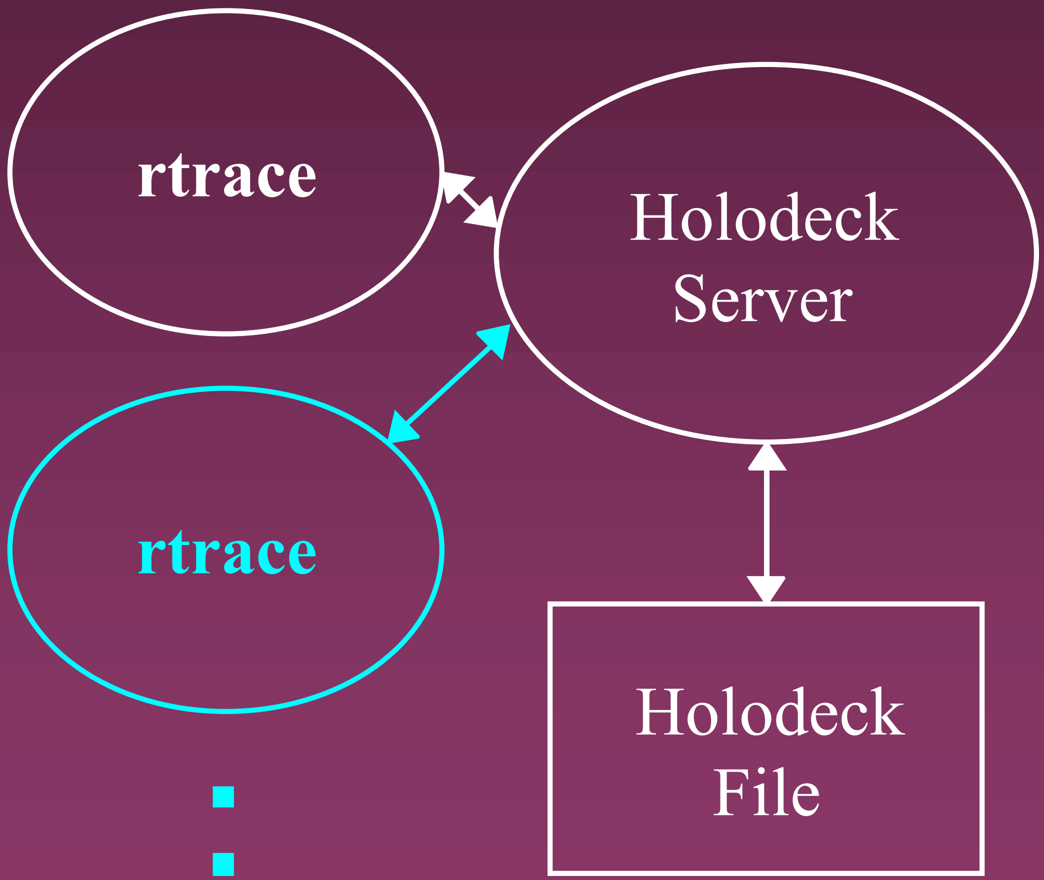


Encoded value	Size
floating point color	4 bytes
position in starting cell	2 bytes
position in ending cell	2 bytes
ray distance	2 bytes

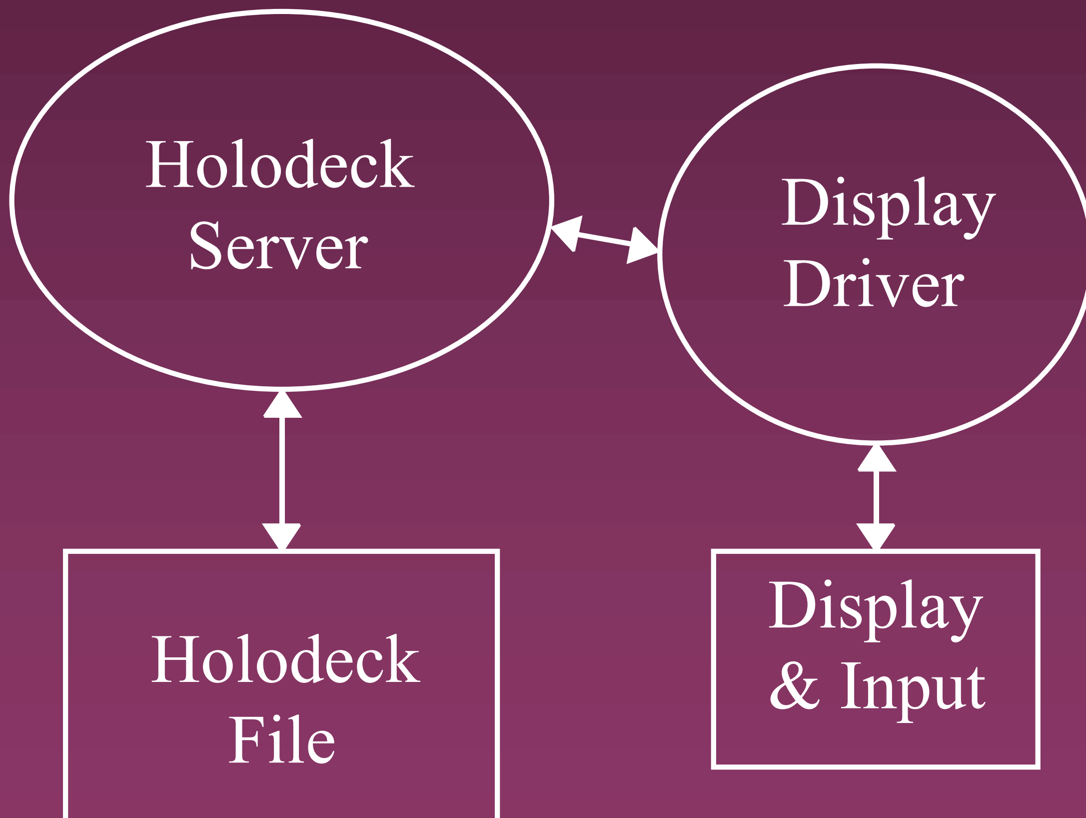
Holodeck Operation

- Holodeck server manages holodeck file
- Display driver manages what the user sees and does
- Sample generator(s) compute new ray samples to use in holodeck

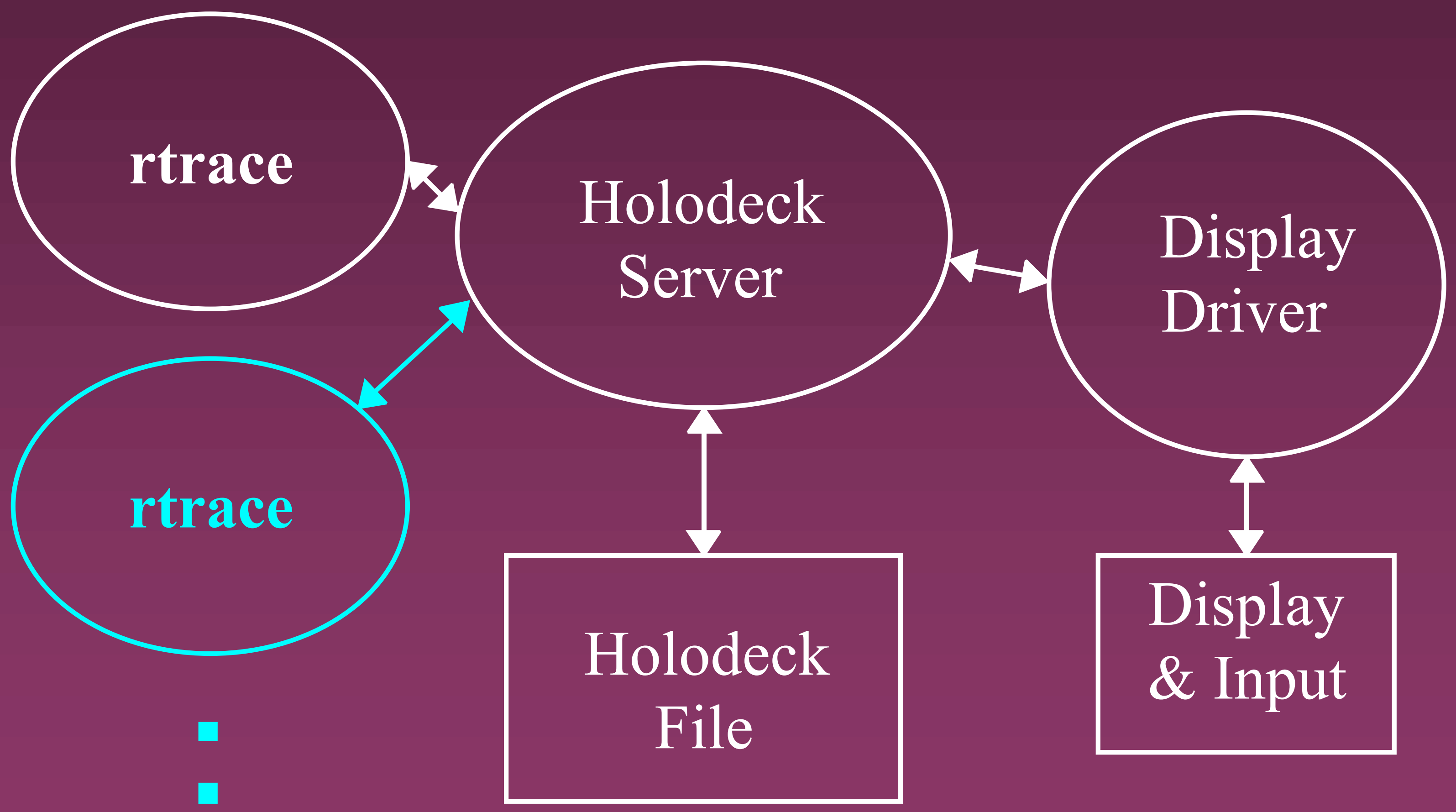
Batch Rendering Mode



Display Only Mode



Interactive Rendering



Subproblems to Solve

- Holodeck cache management
  - » managing beam requests
  - » LRU beam replacement scheme
- Parallel rendering
  - » process synchronization and data sharing
- User interaction and display
  - » determining which beams to request
  - » display representation and tone mapping

Holodeck Cache Management

- Sort and maintain beam request list
  - Nrequested/(Ncomputed+1) priority
- Load data from holodeck into memory
- When memory cache limit is reached, use LRU scheme to free space
- As beam sizes grow, maintain file fragment list to optimize disk usage
- Recover from system errors

Parallel Rendering

- Coarse-grained parallelization
  - » multiple ray tracing processes sharing memory and data as much as possible
- Beam packets assigned to maintain maximum queue size on each process
- Packets returned to server, which caches them and passes them on to display driver

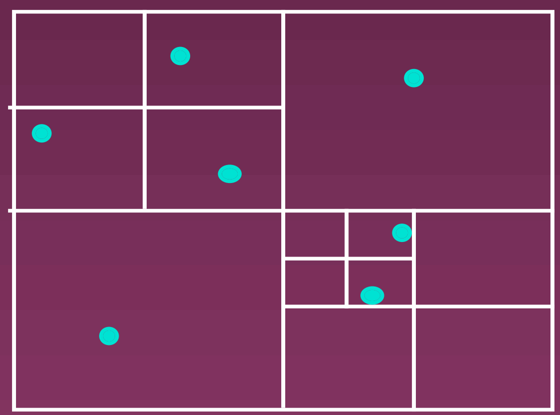
User Interaction and Display

- Get user input
  - » view-motion and control commands
- Determine beams corresponding to view
  - » sparse, jittered view sampling
  - » should be stable for small motions
- Display beam sample data
  - » need 2.5-D intermediate representation
  - » fill in gaps & resolve multidepth samples

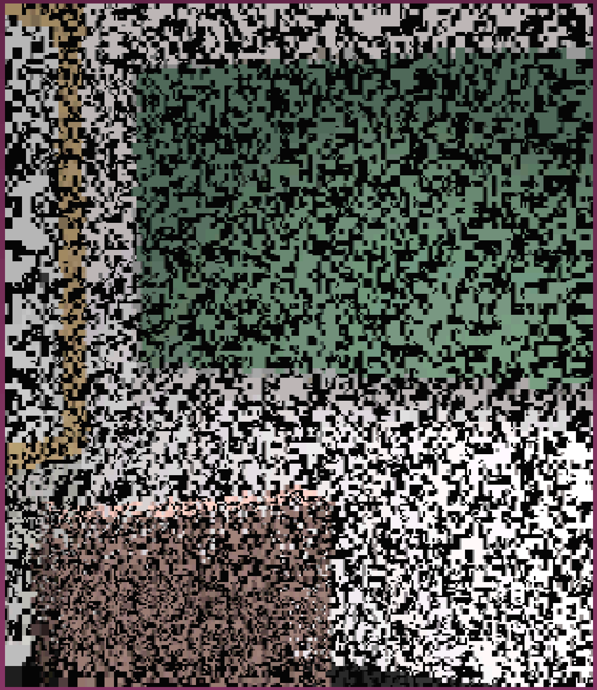
Display Representation

- Quadtree representation w/ X11
  - » simple, fast, ugly
- Vornoi cell representation w/ OpenGL
  - » simple, almost as fast, not as ugly
- Spherical Delaunay mesh w/ OpenGL
  - » not simple, pretty fast, less ugly

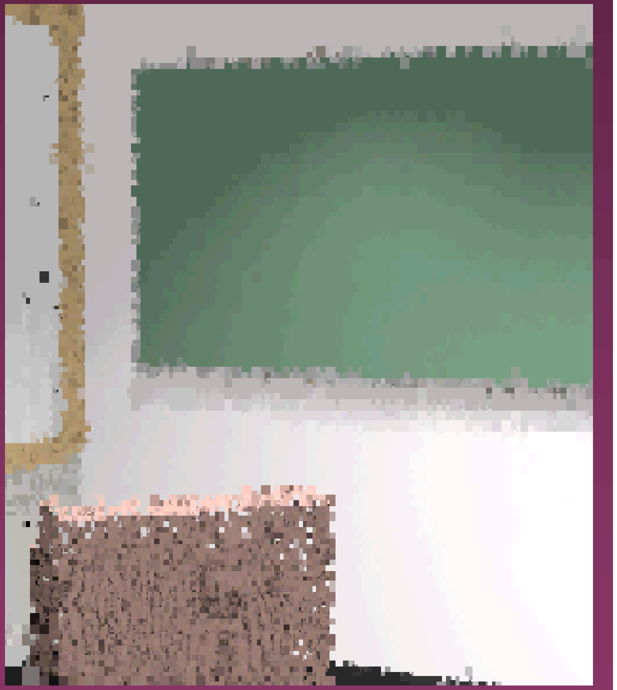
Quadtree Representation



One sample per leaf

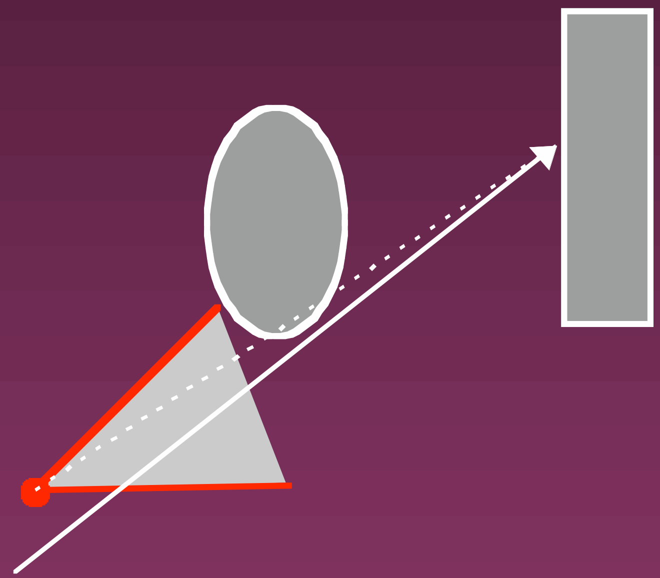


No filling



Average filling

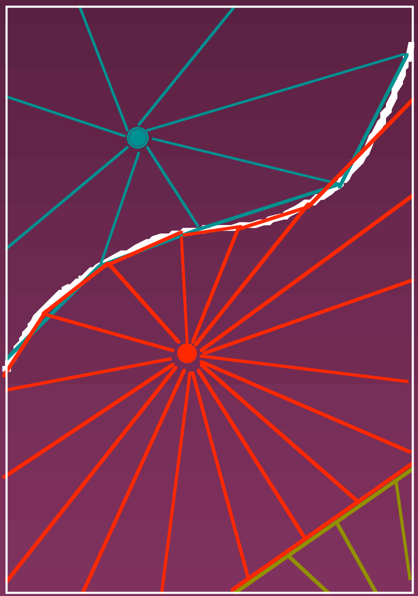
Multidepth Samples



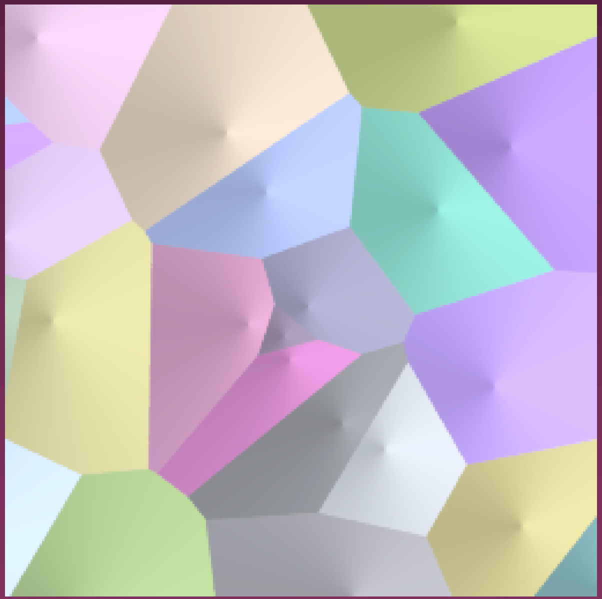
Because rays do not pass exactly through view point, multidepth samples result after reprojection to avoid loss of image focus. Our display method must attend to this.

Voronoi Cell Representation

Draw local geometry with OpenGL



Constrain cones along depth discontinuities



Cones seen from above create Voronoi cells

Driver Comparison

Quadtree representation

Voronoi representation

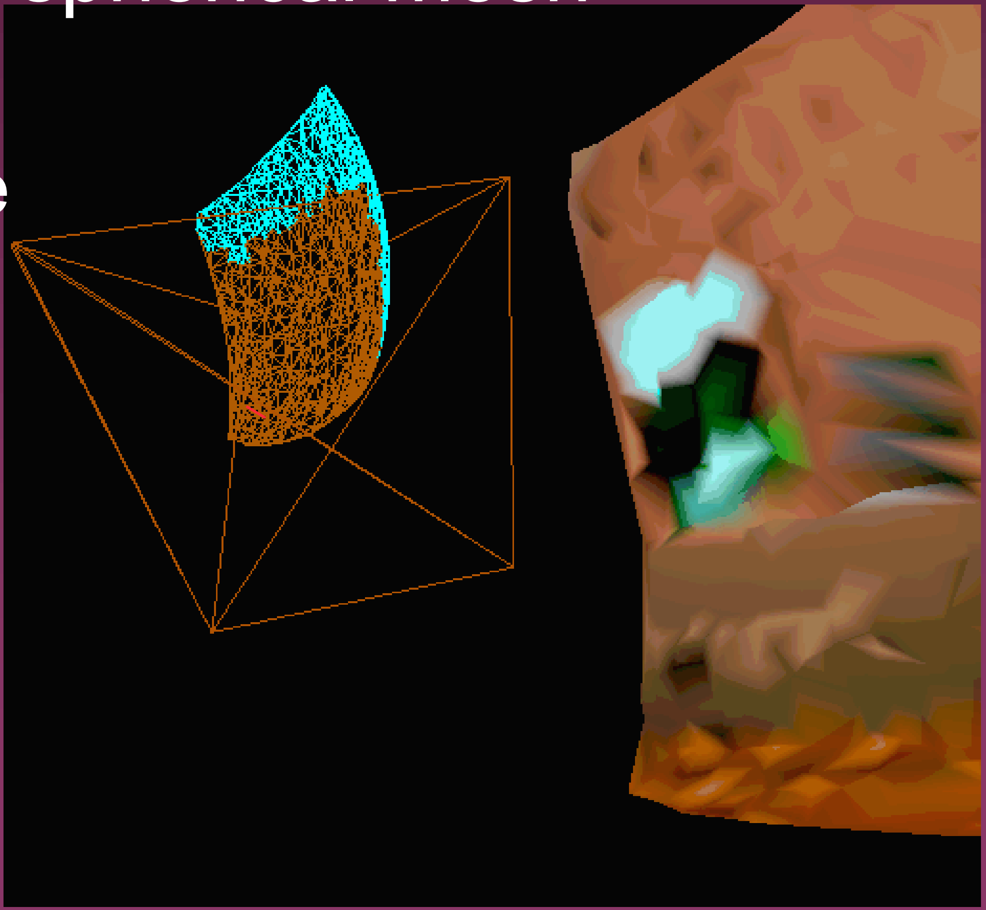
Mesh representation

Mesh Representation

- Maintain Delaunay triangulation of sphere centered on current eye point
  - » each mesh vertex is a ray sample
  - » vertex samples are added dynamically
  - » view rotation uses the same mesh
  - » viewpoint change may result in new mesh
- Mesh triangles are Gouraud shaded
  - » new triangles are drawn over old ones

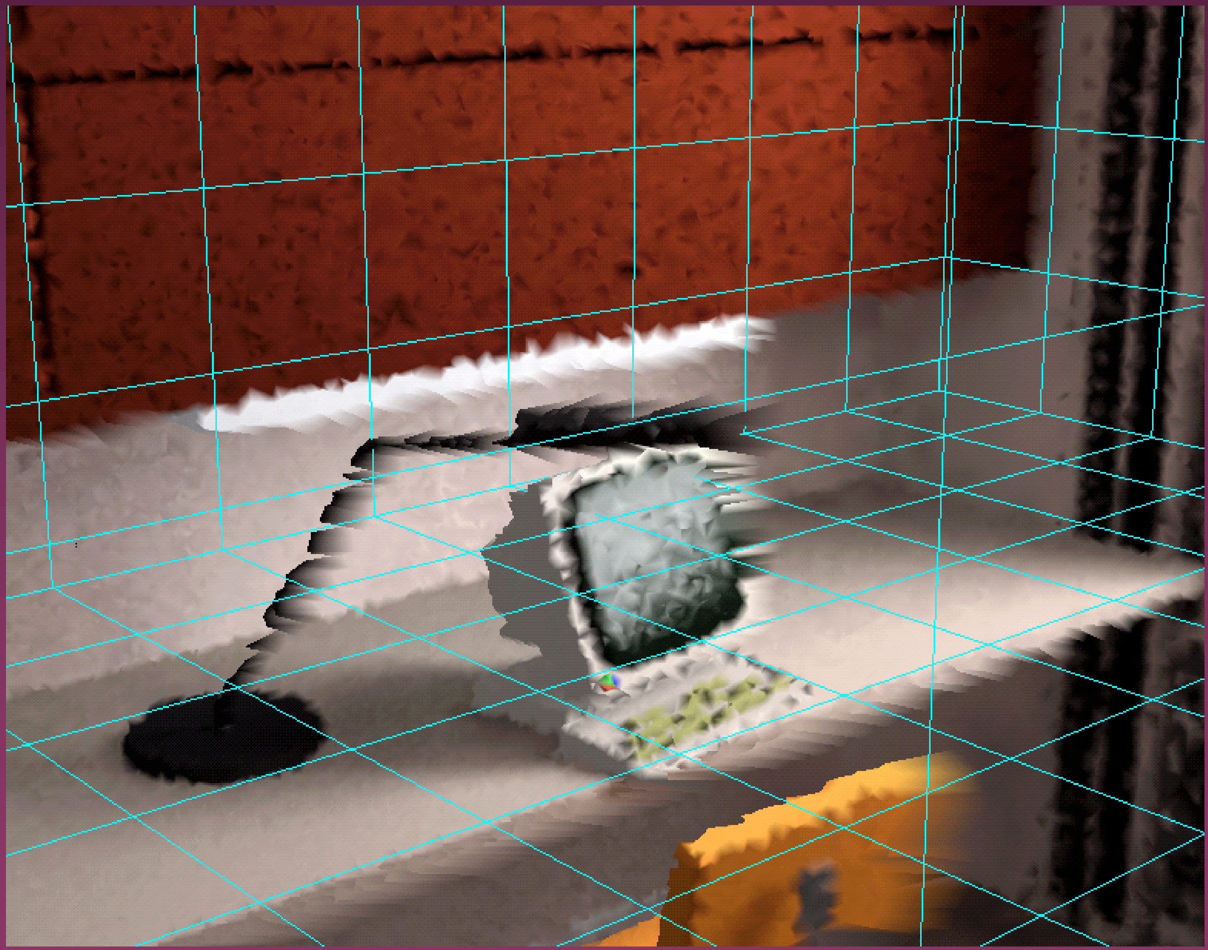
Mesh Representation (2)

- Samples stored in spherical mesh around eye point
  - » spherical quadtree used for vertex location and frustum culling
  - » fast vertex insertion is key



Example Mesh Rendering

From earlier motion sequence:



Dynamic Tone Mapping

- Quickly map dynamic range of scene samples into dynamic range of display
  - » map every sample; update on redraw
- Optionally match human visibility
  - » make visible on display what is visible in real life
- Use [Larson et al] from TVCG '97
  - » histogram adjustment method

Tone Mapping Example

Linear

Camera model

Human model

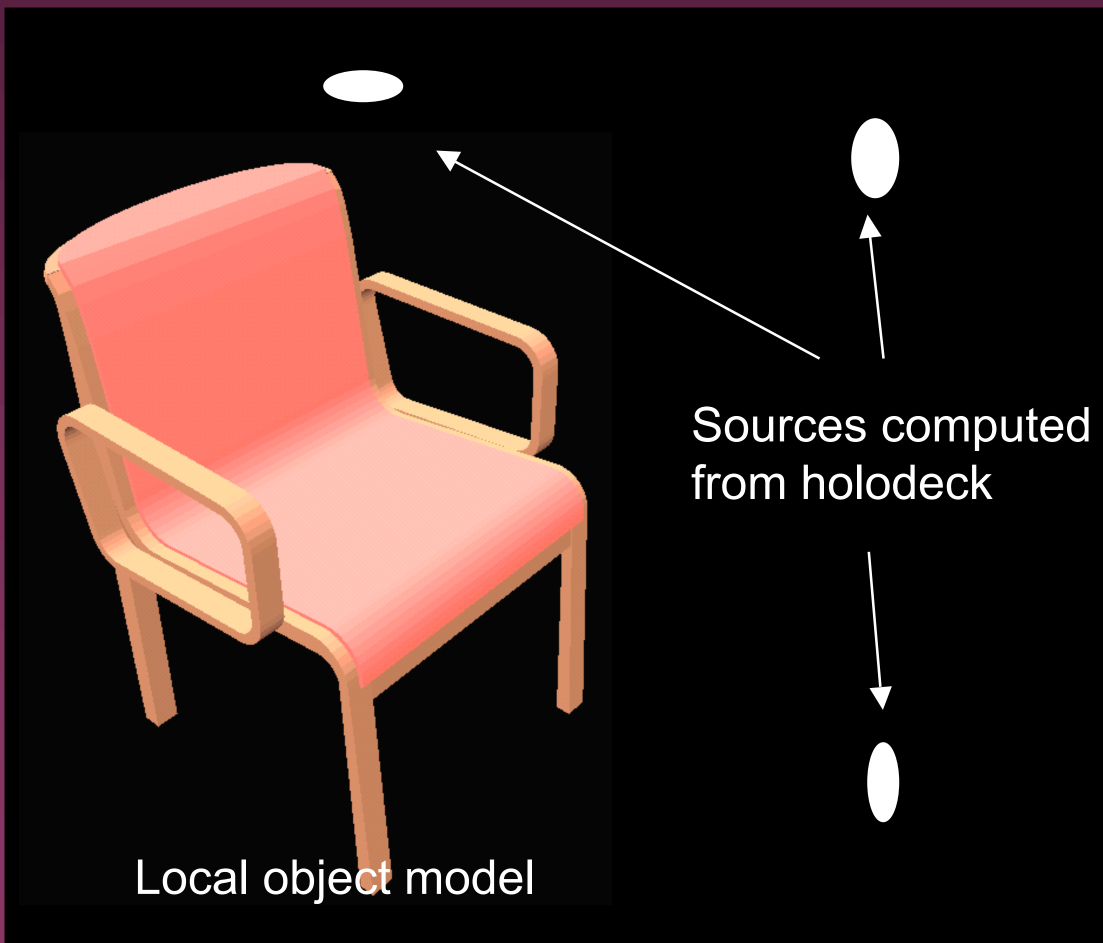
Stereo Display Driver

- Use “full screen” stereo
   - » supported on most platforms
   - » reduced vertical resolution unimportant
- Draw left and right buffer alternately
   - » twin representation in Voronoi driver
   - » single representation in mesh driver
- Stereo effect is very strong
   - » tends to show triangles a bit too much

Local Dynamic Objects

- Compute lighting for local objects
  - » uses holodeck samples a la [Walter97]
  - » OpenGL local shading is approximate
  - » small motions require rerendering only
- Currently implemented in trial form
  - » manual object placement
  - » no animation
  - » demonstrates principles, feasibility

Local Dynamic Objects (2)



Wastebasket lit by OpenGL

Roll Video

- Example interactive session
- Multiple processor performance
- Local object illumination and rendering

Conclusion

- New method for interactive ray tracing
- Multiprocessor ray calculation
- Dynamic ray caching
- Intermediate representation for display
- Local objects with approximate shading
- Applications in visual simulation, architecture, virtual reality

Software Availability

Radiance 3.1.20 from LBNL:

http://radsite.lbl.gov/radiance/

Holodeck software from UCB:

http://positron.cs.berkeley.edu/gwlarson/hd/

- Y1K Compliant!