RenderMan Nuts N Bolts PART 2

Mark Hammel
&
Tal Lancaster
(Part 2 of 2)

Disclaimer

The contents of these documents were created for a class that Mark Hammel and Tal Lancaster gave at Disney Animation Studios in March 2007.

They are just two people's experiences with PRMan over the years. They are not official statements by or for Disney or Pixar. Mark and I do wish to thank Disney for letting us share these with the RenderMan Community.

Also at the time we only had PRMan-13. There have been many improvements since then.
-- Tal Lancaster March 11, 2008

Lots of topics

- There are a lot of topics to cover
  - Some won’t make it in this session
  - Others may be just a quick overview
    - Remember just “Nuts & Bolts”

Background Terms

- Multi-pass rendering
- MIP-map

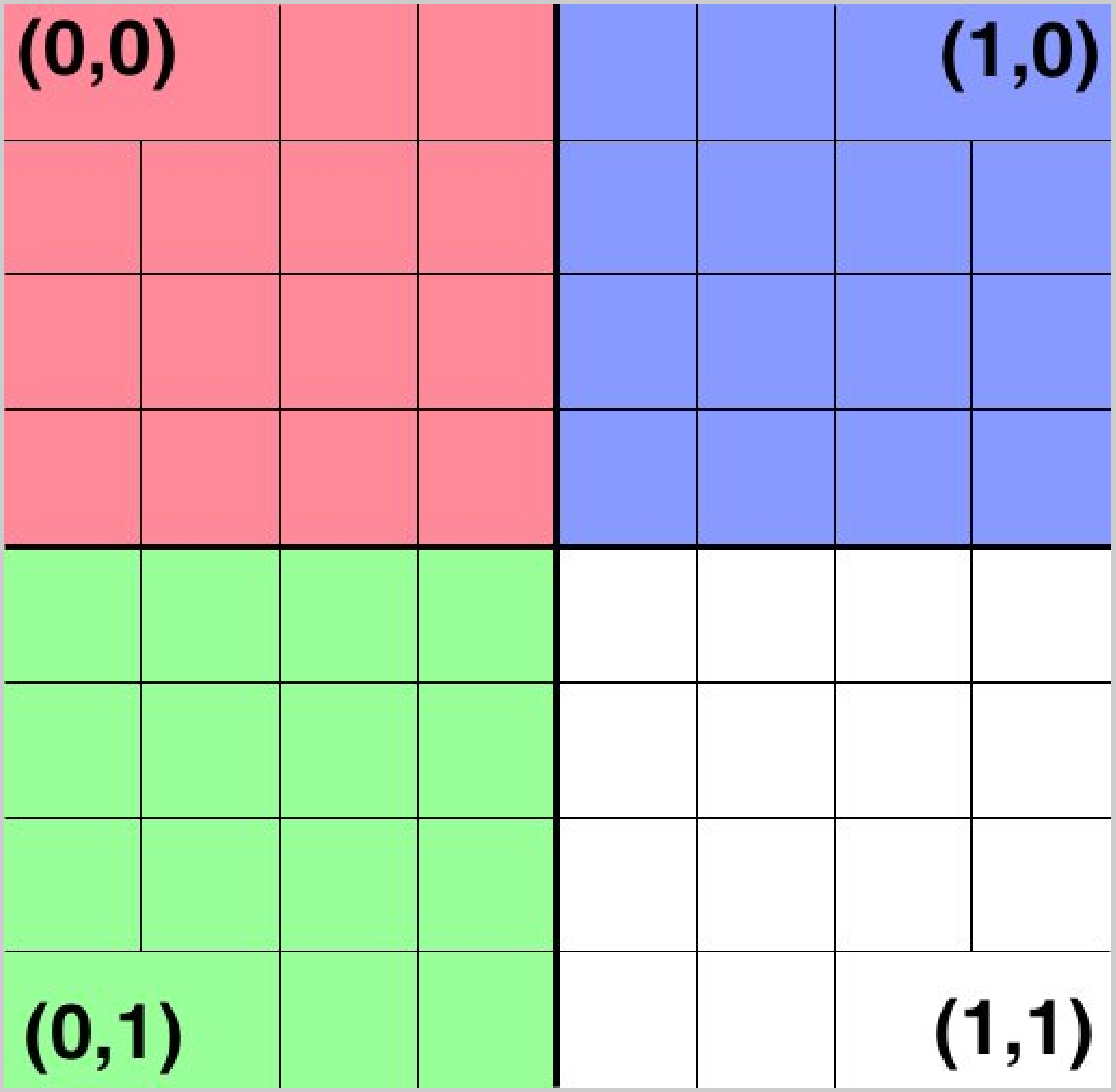
Multi-pass Rendering

- Run multiple render (passes)
  - Results (images) from one pass are fed into another pass.
  - Reflection, shadow generated before can be used by color pass.
- Nothing special about passes from render pipeline. It is just processing data and making images.

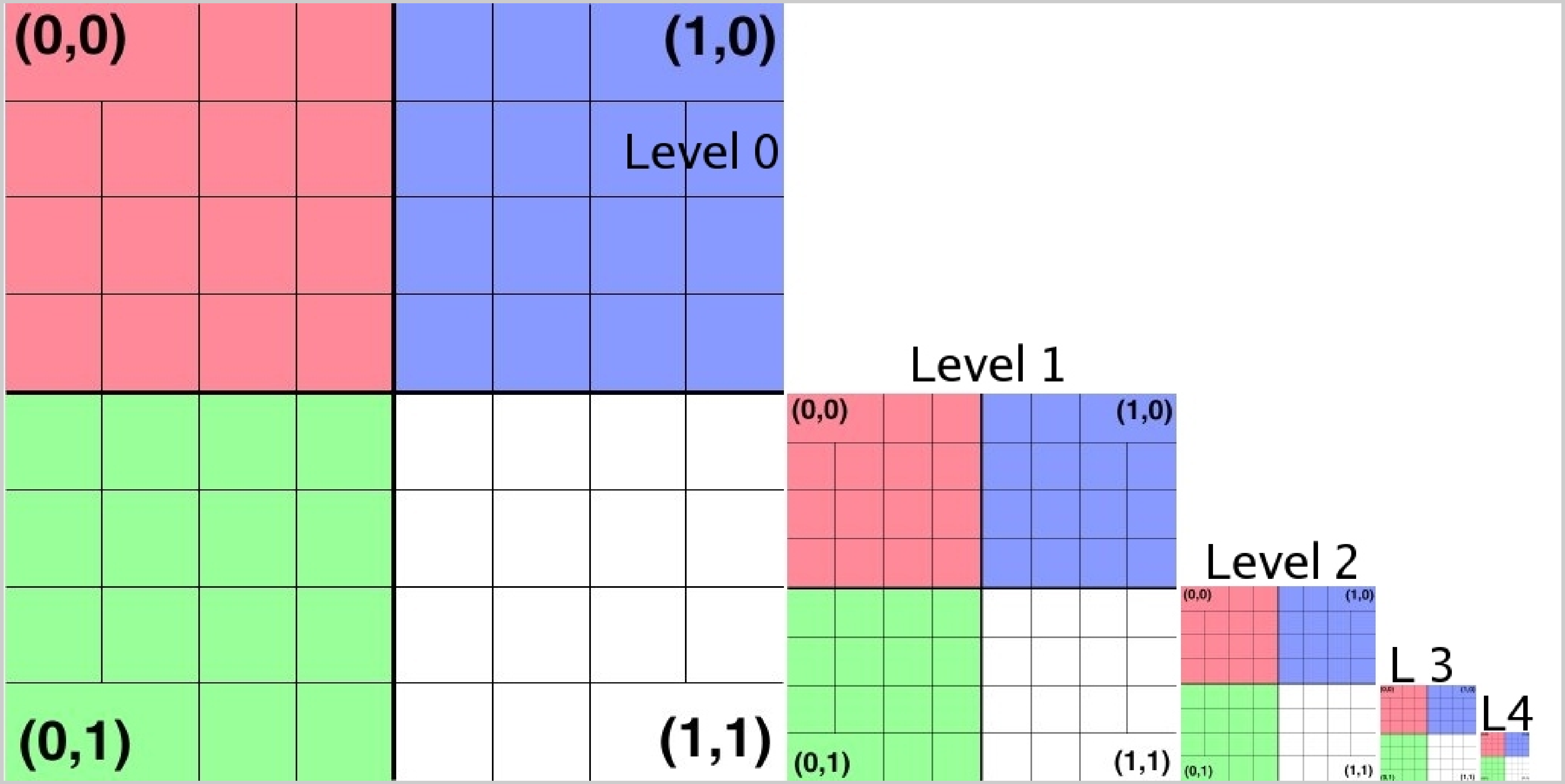
MIP-Mapping

- Some textures types in PRMan are MIP-maps
- Term Introduced Lance Williams 1983
  - multum in parvo (MIP acronym)
    Latin: much in little (space)
- Multiple images (levels) stored in a single file
  - Each level averaged down 1/4 size (power of two) previous
- Pixel area determines MIP-level
  - Allows smaller lo-res version when object distant.

Example Texture Map



MIP-map Levels 0-4



Topics

- Speed vs Memory
- Mapping
- AOVs
- Raytracing

Topics

- Speed vs Memory
- Mapping
- AOVs
- Raytracing

-woff

- If get lots of errors

```
S32004 {ERROR}   Transform from
"ramp1CoordSys_yucaAAShape" to "world"
in shader "s_looks" is undefined. [<Prehiding>
```

  - Can slow renders due to I/O
  - Always let someone know to get fixed
  - Meantime `-woff` the error/warning disables it.
  - No longer impact time; But image may be wrong

```
prman -woff S32004 ...
```

NetRenderMan

- PRman renders by buckets
  - Been available since mid 90s
  - Can farm out sets of buckets to different computers

```
prman -h HOST1 -h HOST2 ...
```

  - Uses prman license for each `-h`

Multi-proc RenderMan

- 11.5 (2003/2004)
- Make use of current machine’s procs only

```
prman -p:PROCS
```

- Uses `PROCS-1` license (ie, 1 processor free)

Multi-proc

- Each process
  - Independent memory
    - processes RIB; geometry; shaders; textures
  - Worse case PROCs * 1proc memory
    - ie. 1 proc 2Gig; 4 procs 4X memory (8GB) consumption!
    - Never actually seen full multiple
      - Sometimes 70-80%

-t:PROCS Multi-Threaded

- Memory for storing scene graph geometry/shaders
  - Shared execution memory location by the processors
  - Each thread gets own slice of data and executes shared code
    - Much smaller memory footprint than multi-proc
    - Theory: many procs   memory closer to 1 proc footprint

Multi-Threaded issues

- Every facet of rendering must be thread safe. Need to protect global/static data from getting trashed.
- Shader Plugins
- Procedural geometry

More Multi-Threaded issues

- In 13 multi-threading many times slower than multi-proc
  - 1 thread doing work other threads getting starved
  - Multi proc: no need to wait/starve as completely independent.
- In 13.5 multi-threading redone. Can be faster than multi-proc in many cases (post note)

Discrete buckets

- PRMan works discrete buckets
- Theory: when done can delete the process data.

Changing Memory footprint

- Change bucket size / gridsize
  - affect memory footprint
    16 x 16 / .5 = 512 MP
    4 x 4 / .5 = 32 MP
- Lower memory footprint
- Render less efficient (SIMD)
  - But if can’t render don't have choice

Tiling

- Doesn't always pan out. Some reason shrinking buckets not enough.
  - Render up to a certain point, then render dies
  - Tiling: break up frame into N CropWindow Renders
    - Each tile render
      - CropWindow (sub-frame) gets a set of buckets
      - subframe images glued together when finished

PRMan recovery

- Last known completed bucket stored in TIFF image while rendering
- Rerun `prman -recover 1`
  - Continue after last completed bucket
  - Re-render like original
    - Don’t change anything

MP Cache

- (PRMan 11)

  ```
  Hider “hidden” “mpcache” [1] “mpmemory”
  [6144]
  ```

- Limit memory usage when too many transient MPs that aren't needed by current bucket.
  - If more than 6MB MPs then cached to disk

MP Cache

- (guessing) Replacement for “Extreme displacements”
- Extreme displacement reduced memory footprint.
  - Wasteful
    - Redundant shading
    - Longer render times
- Probably why retired in 13.5 in favor of MP Cache.

MP Cache

- NOTE: As long as have the memory then turning off MP will render faster than on.
- When doing interactive renders I turn this off if I don't care about RAM usage.

Topics

- Speed vs Memory
- Mapping
- AOVs
- Raytracing

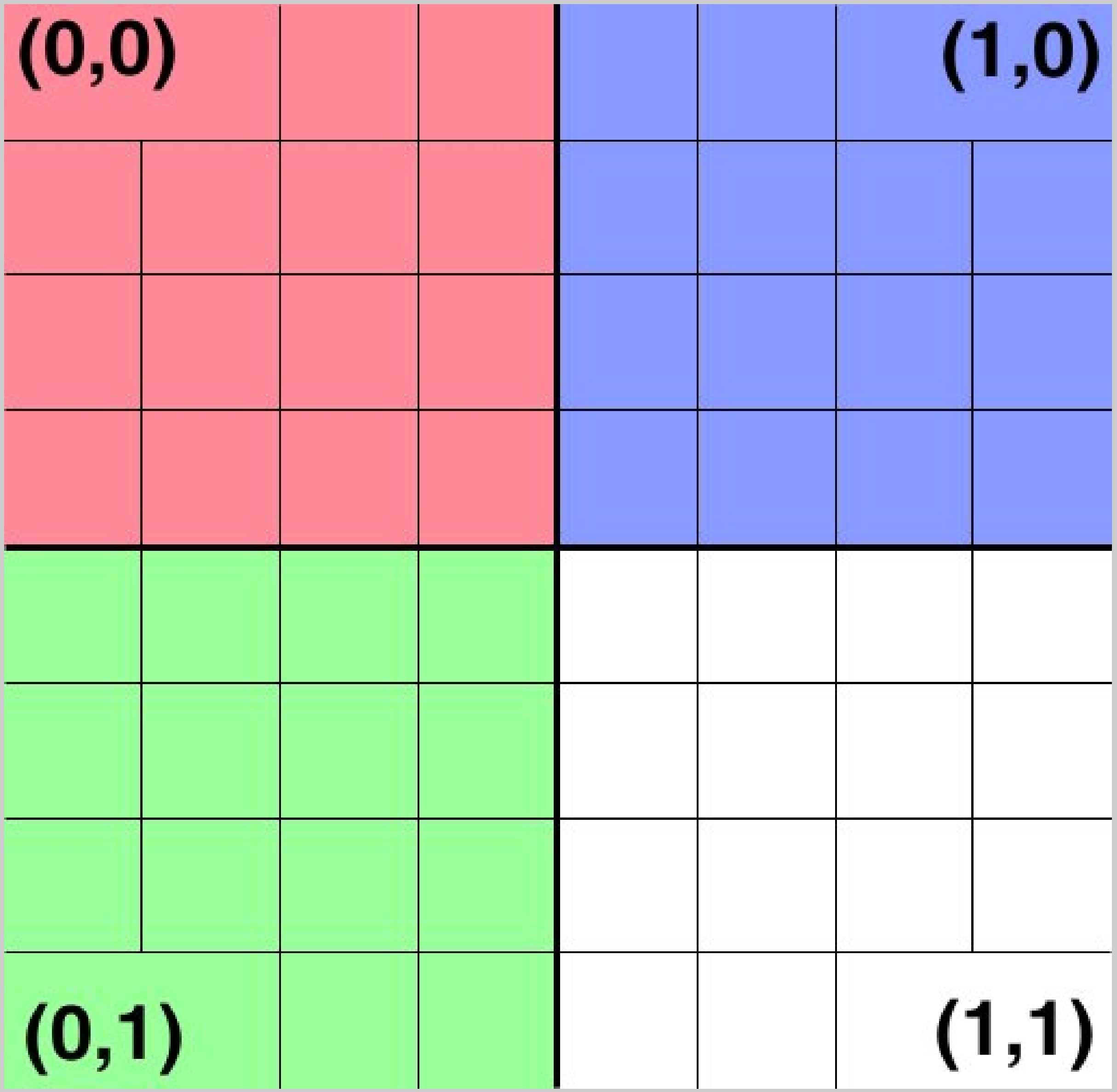
Mapping

- Texture maps
- Shadows
- Point clouds / brick maps
- SMSS
- PTEX
- Photon maps

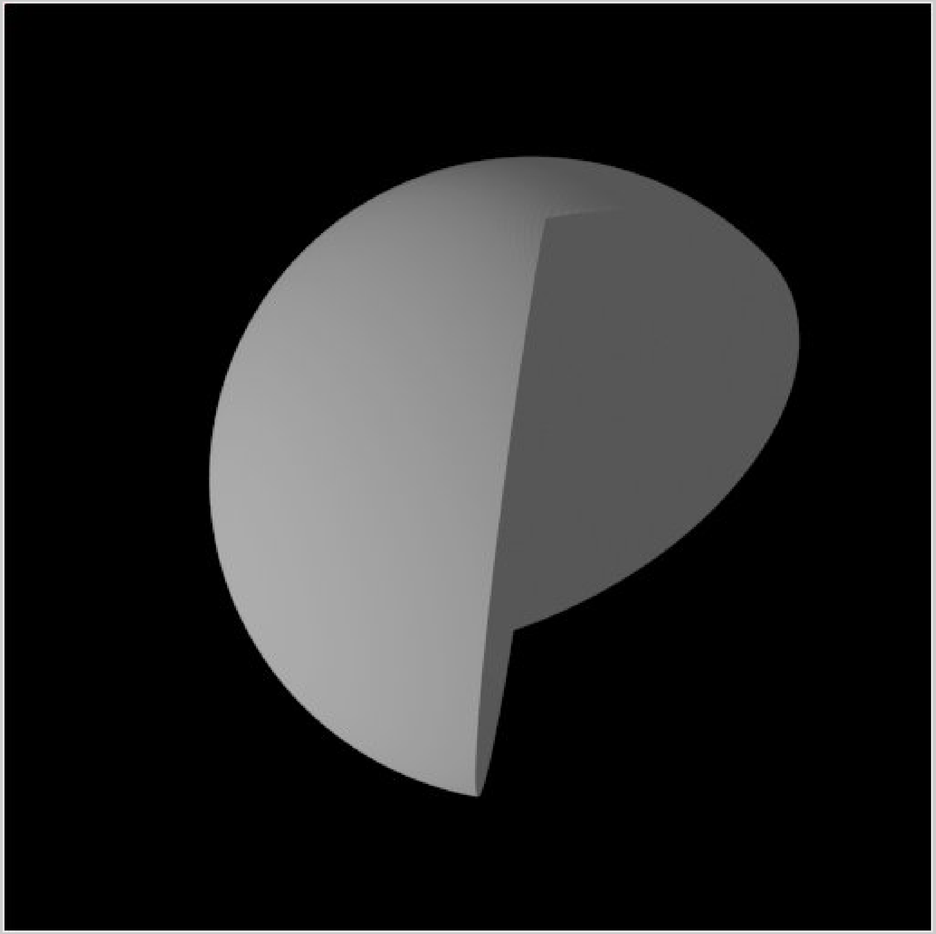
2D Texture Mapping

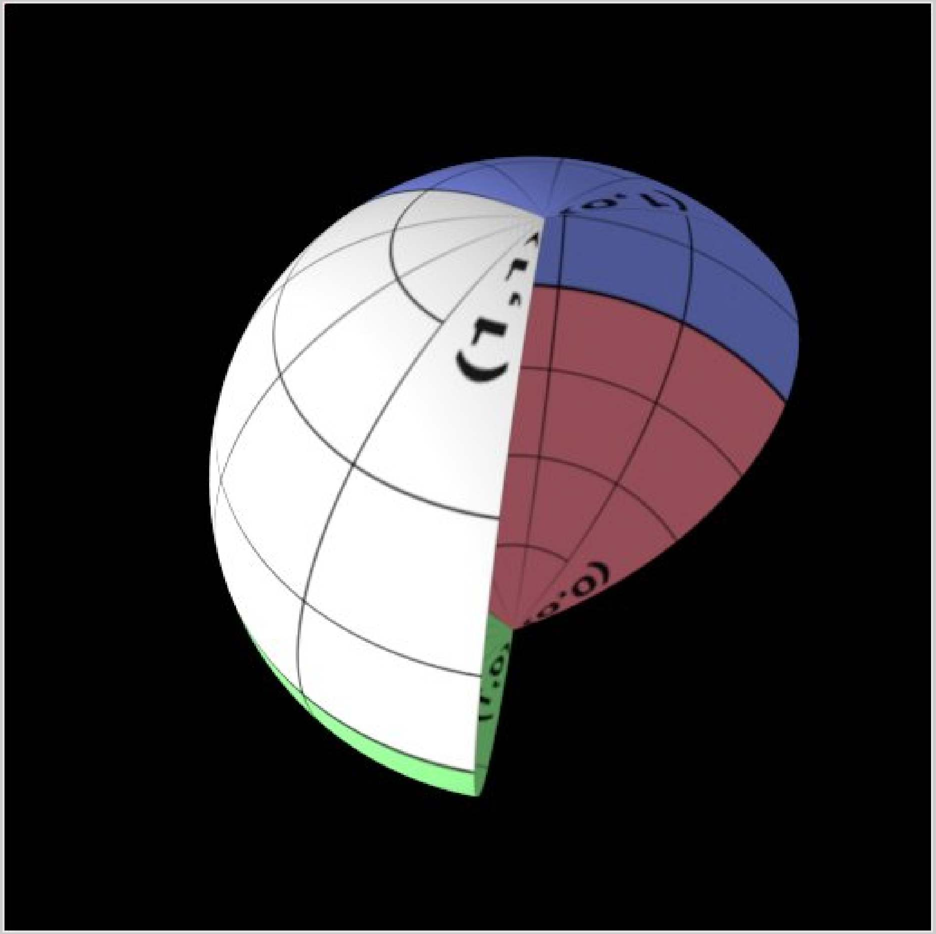
- s/t u/v parametric
- Projection
  - Camera, coordinate space
  - Cylindrical, Spherical
  - etc.
- Reflection
  - Coherent reflections on shiny surfaces (see other objects)

Example Texture Map

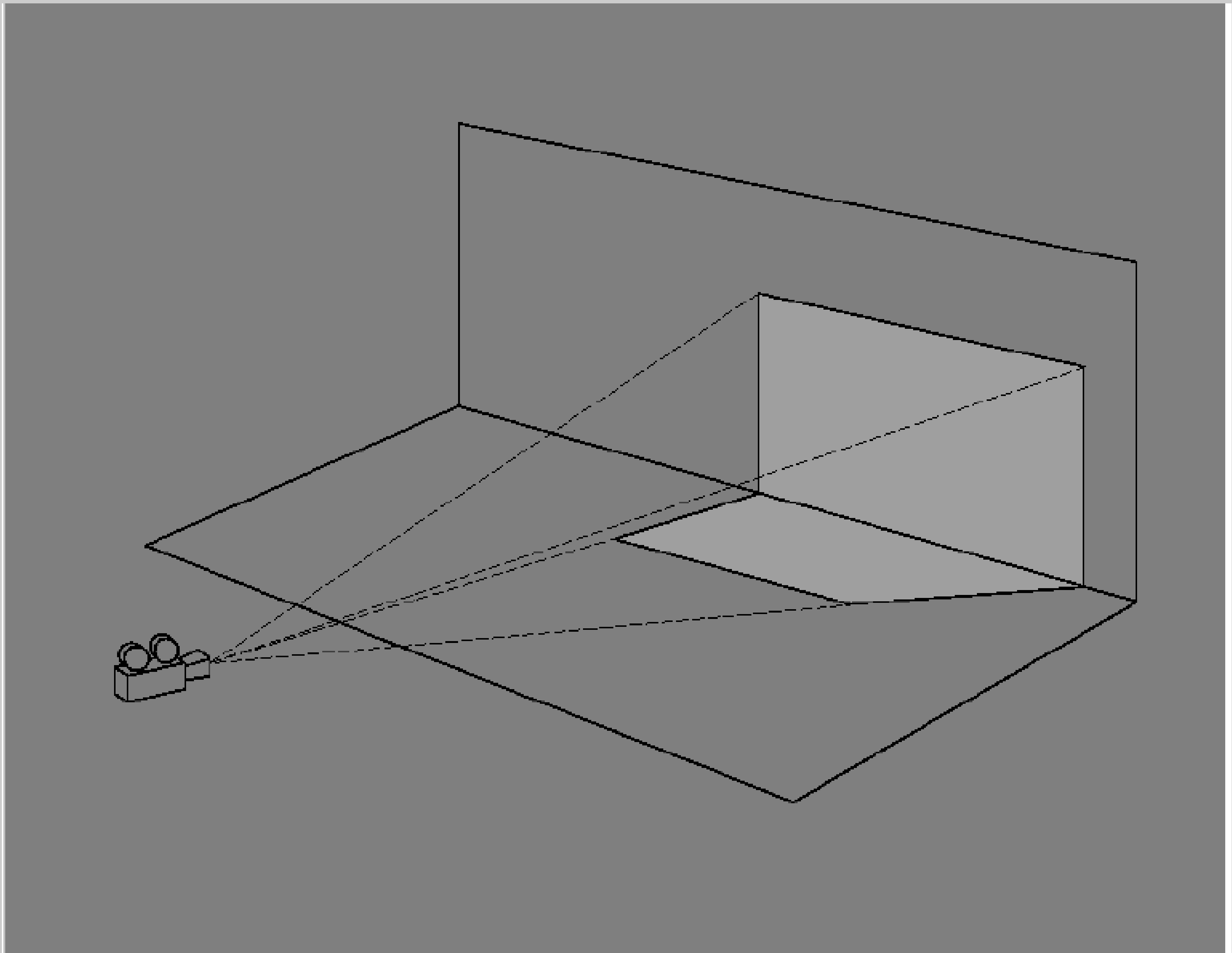


Parametric mapping

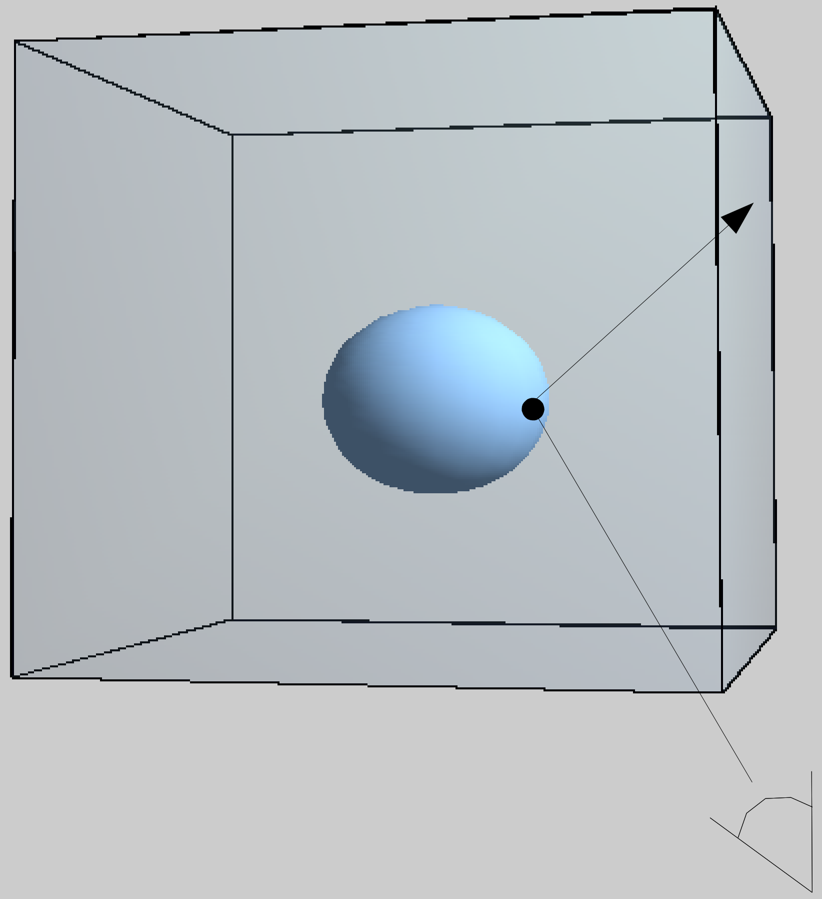




Perspective Projection



Reflection Mapping



- Lookup based on reflection vector.

Reflection Maps

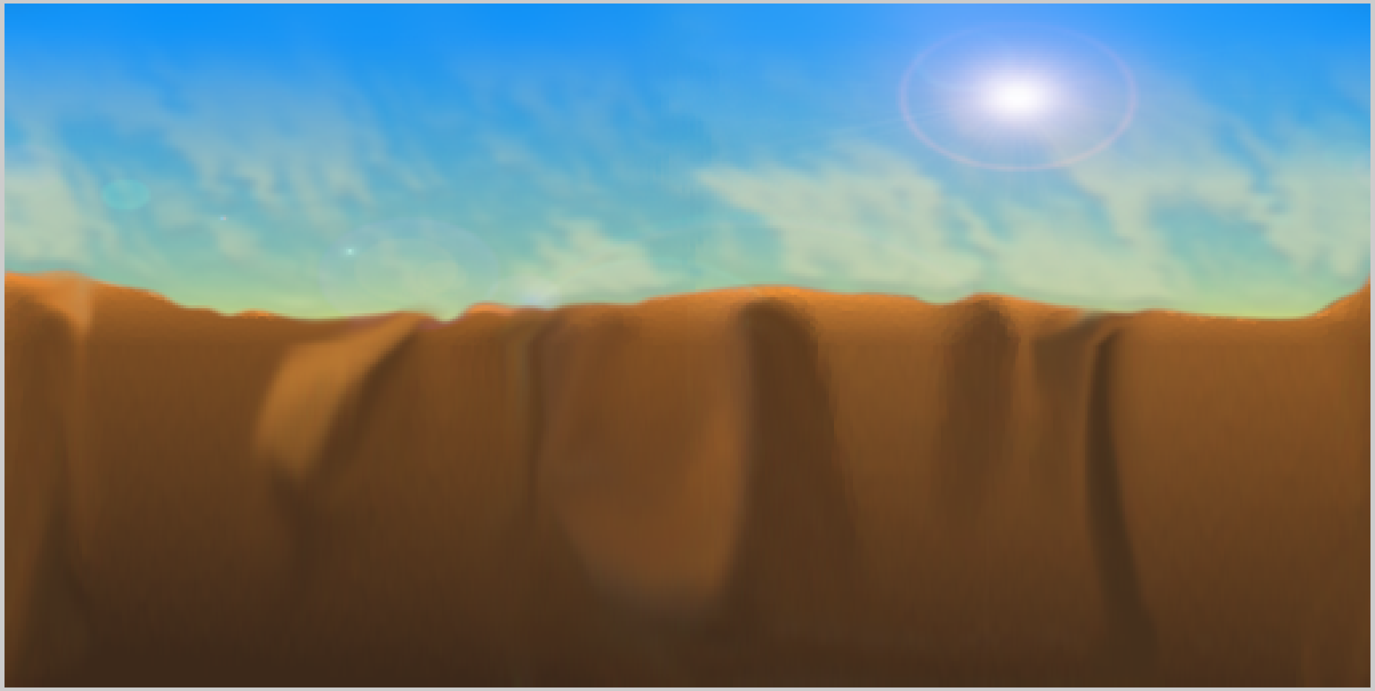
- Multipass (or painted)
  - latlong (for painting)
  - environment cube (for rendering)
- Limitations
  - All objects resolved out to same distant plane
    - No sense of depth / Hard to get sense of scale
    - Won’t get correct result when reflected object near reflecting object.
  - As camera moves (flat) reflections don’t easily move – direction vector

Latlong Map

- latitude/longitude map
  - 2d map
  - Convient for painting
  - Flat surfaces (ie. table top)
  - Camera view from back side of surface.

Latitude/Longitude Refl map

Lookup by reflection direction

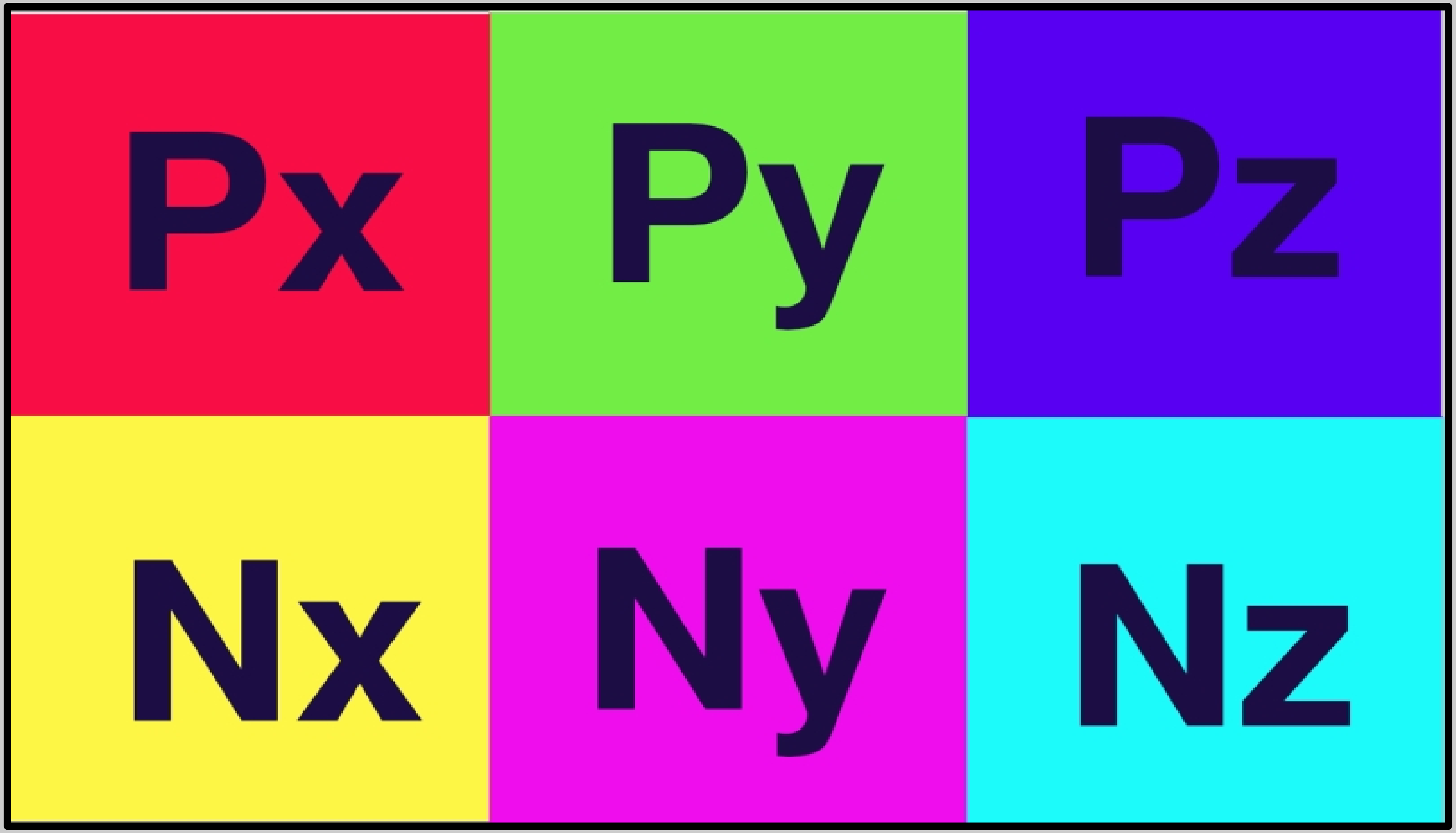


Environment cube

- Six sided cube (6 camera view) from center of object
  - More accurate representation of geometry in scene.
- Convient for rendering
  - However Disney tools to paint and generate cube.
- Blur happens only per side so see cube edges

Envrionment cube (newer 3x2 view)

Lookup by reflection direction



Shadow Pass

- Multi pass scene description
  - Version for each shadow
  - Version for final color

Shadow Pass

- What geometry is seen
- What shaders run

Shadow Pass

- What geometry is seen when?
  - Shadow receivers
    - Not in shadow pass (efficiency)
  - Shadow casters (blockers/phantom objects)
    - Might not be in color pass
- What shaders run
  - No shaders
  - Displacement
  - Surface

Shadows Maps

- Compromise between
  - Shadow quality
  - Rendering efficiency

Shadow types

- min
- max
- average
- midpoint
- deep
- single map soft shadow
- Pixar’s multi shadow (penubra- soft shadow)

Shadow Types

- min
- midpoint
- deep

Min Shadow type

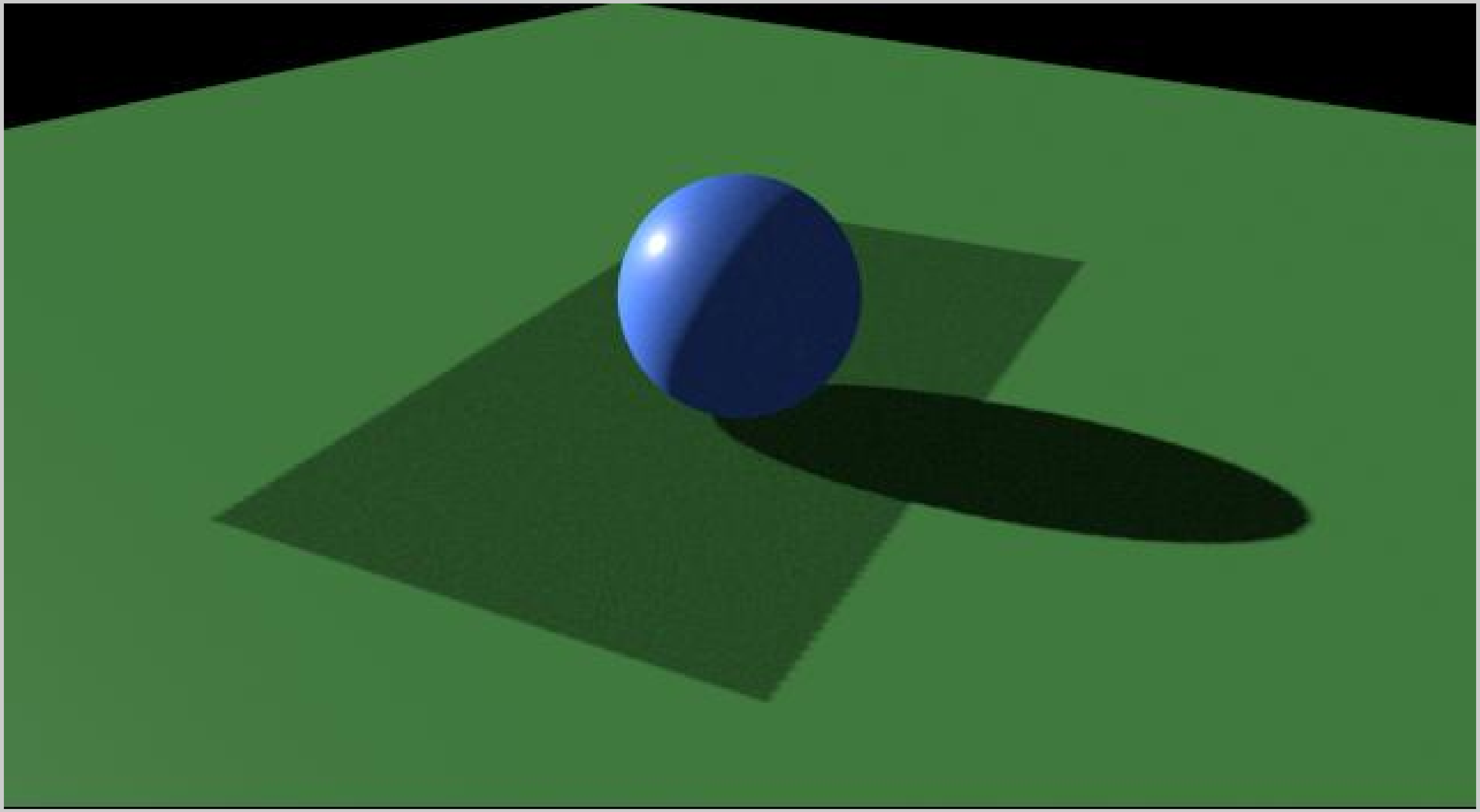
- Original shadowmap type
- Minimum of all depth values within current pixel
- Shadow bias issues

Shadow Bias

- Self shadowing
- Dirty / spots on surface
- Numerical inaccuracy
  - Color pass result:
    - Depth of object (in spots) slightly less, sometimes more, than the object when shadow was generated
- Not pretty or predictable.

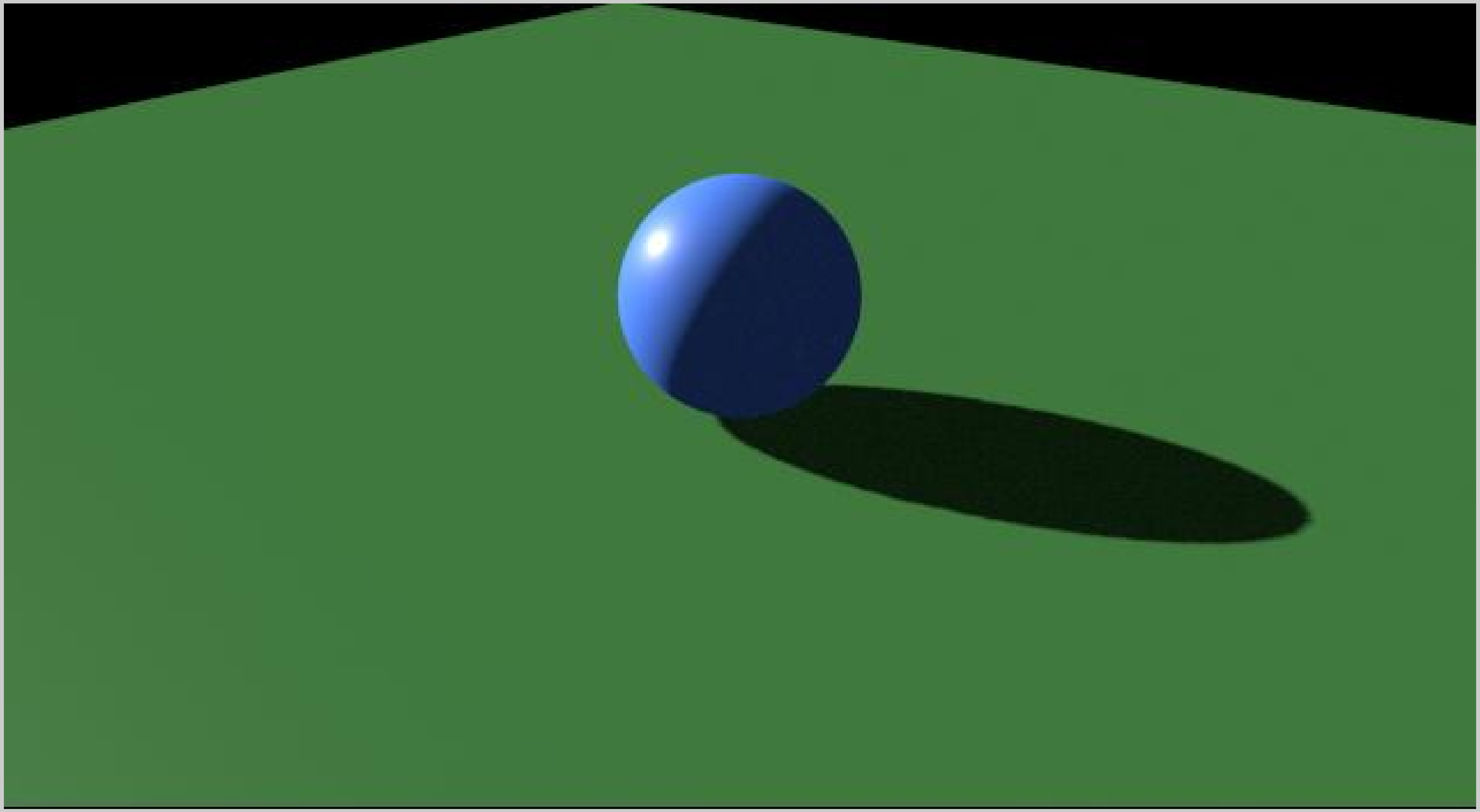
Shadow Bias

- Bias .01



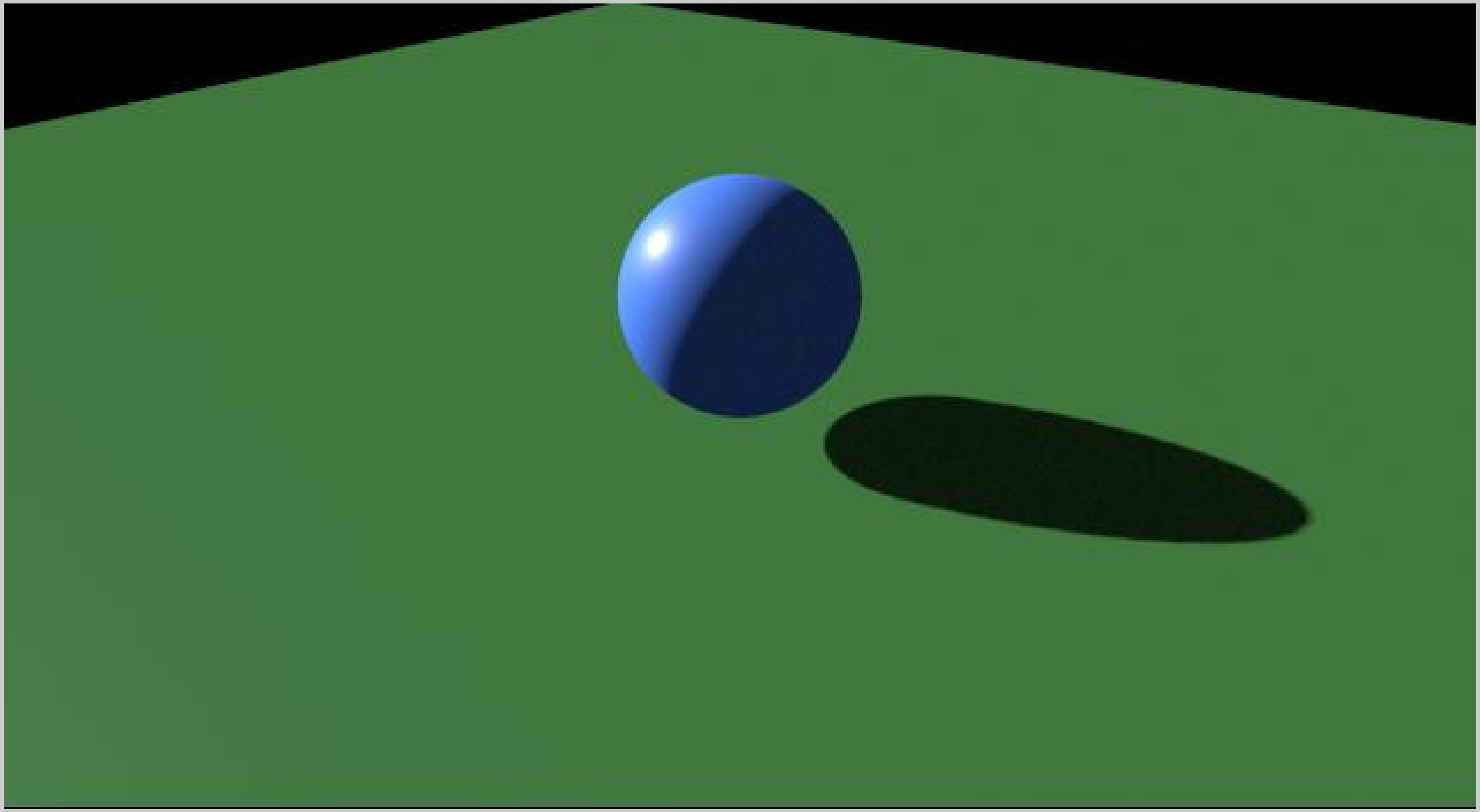
Shadow Bias

- Bias .1



Shadow Bias

- Bias 2



Shadow Bias

- 1 way
  - Render without shadows
  - Render with shadows
    - If areas not in shadow go darker: BIAS!
- Another way
  - Use a AOV(s) to track the shadow map(s) in question and see what they are affecting.

Midpoint Shadow type

- Midpoint between closest and second closest available
- Almost no bias issue
- Takes a little longer to generate
- If two objects too close
  - Gets confused
    - Precision issues
    - Thinks inner penetrating and produces incorrect map
    - Different kind of bias issue, but can’t correct via “bias”.

Deep Shadows

- Rather than 1 depth sample
  - Many samples and current opacity for said sample
  - Visibility function
    - fraction of original beam reaches that depth
- Semi-transparent shadows & color opacity/shadows
- Better anti-aliasing
- Great for fur/hair; volume illumination

More Deep shadows

- (some) Motion blur shadows
  - Only accurate when receiver is not moving relative to the shadow camera
- Resolution size smaller than standard maps; more precise
  - Standard 2k, 4k
  - Deep 512
- MIP-map support
  - Not forced to load entire map into memory

More More Deep Shadows

- Discrete
  - Samples preserved; assumes depth samples separated (discontinuous)
- Continuous
  - Samples interpolated through middle of discontinuous samples
  - When needing smooth depth
    - hair, volume

Deep Shadow down side

- Use a lot more disk space
- Slower to generate map (more geo that get's involved and tracked)
- Bias issues
- (MIP-map) However after 3 in a scene start seeing memory issues. Don't know why.

Motion blur shadows

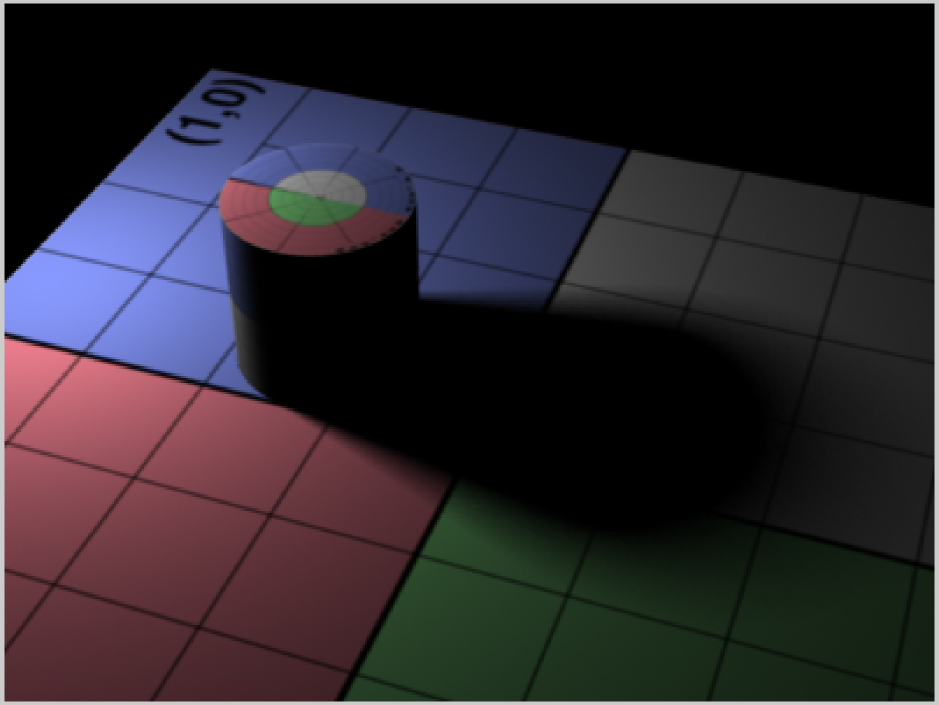
- Deep shadows (certain cases)
  - Practically free
  - Doing brunt of filtering work @ map generation
    - Random time for each shadow sample
    - Samples averaged for visibility function
      - Provides average coverage
        - Image plane
        - For time

Point clouds

- RSL bake3d()
- Shading Rate controls # of points generated (or map resolution size)
- Render settings to get full object
  - Don’t want culling of backside
  - Want non-raster oriented dicing

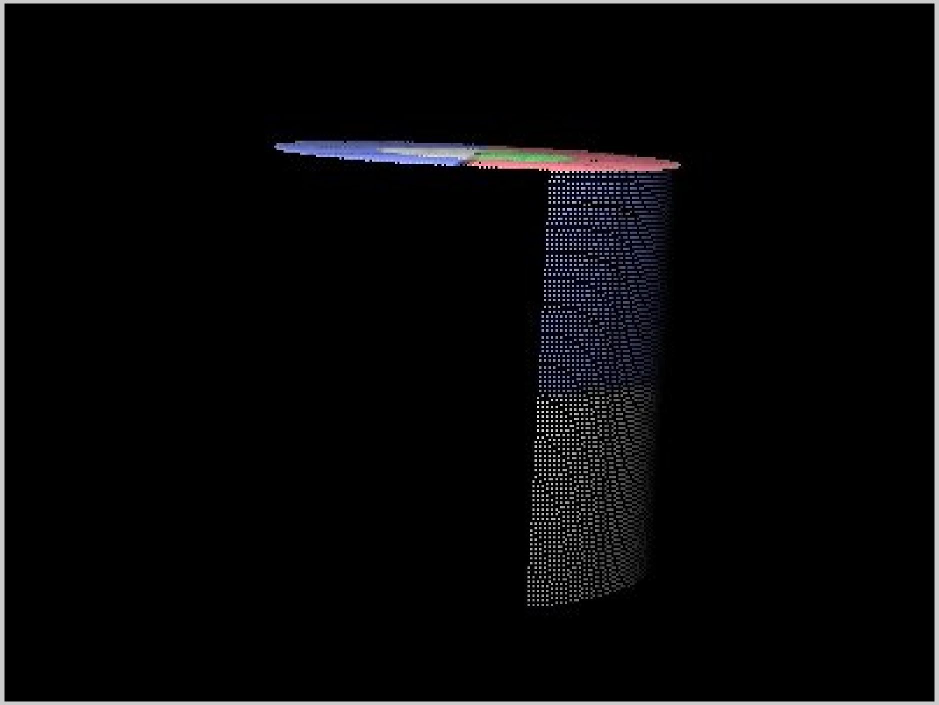
3D Texture baking

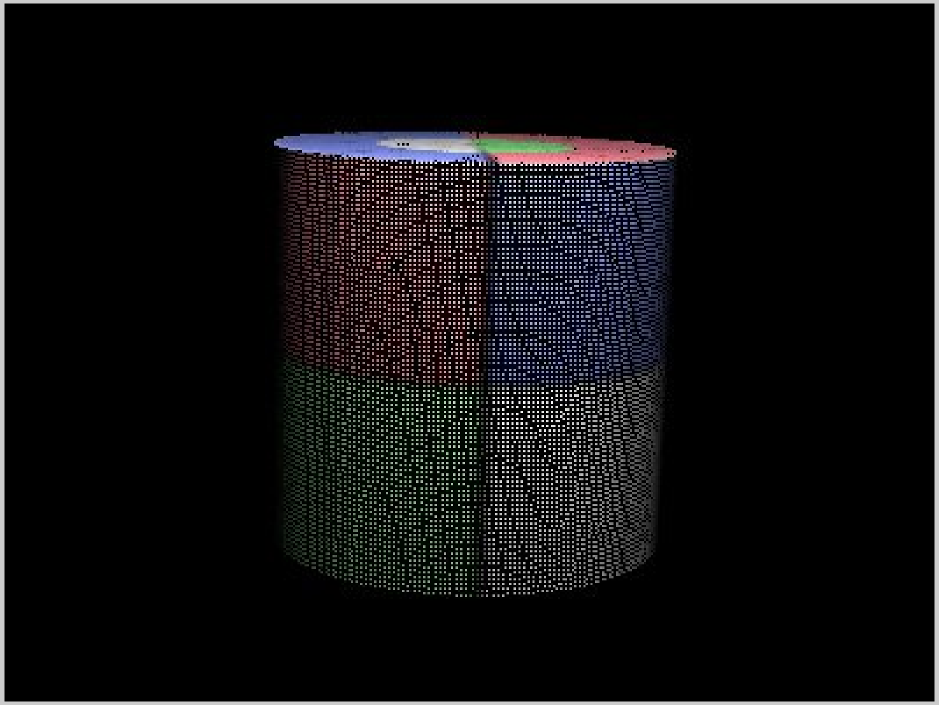
- Example scene
  - Baking out diffuse illumination
  - Into own point clouds
    - Cylinder
    - Ground plane



Culling on vs. off

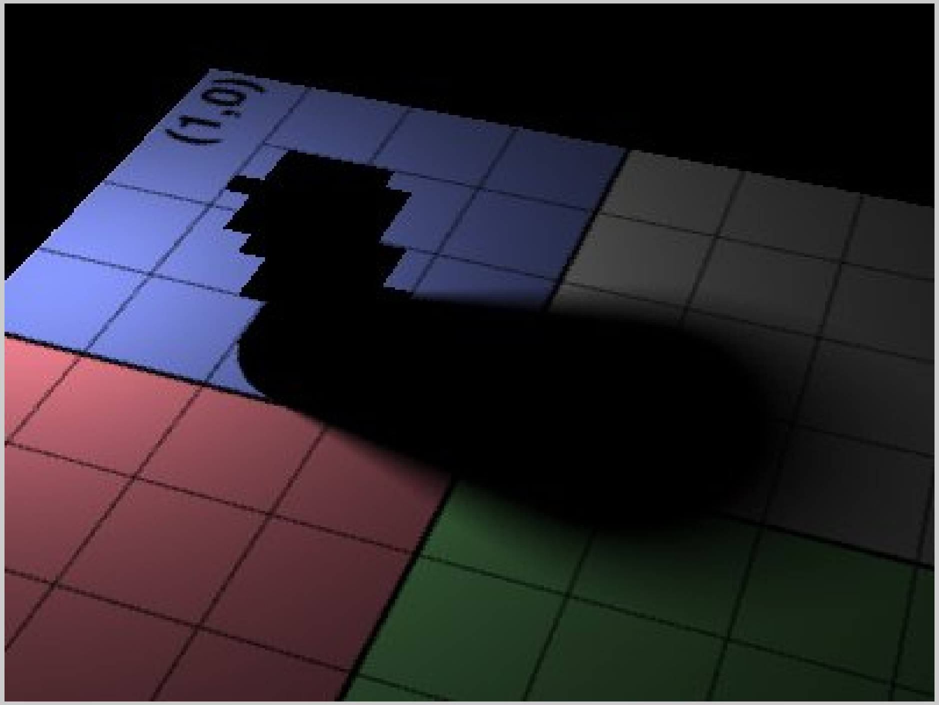
```
Attribute "cull" "hidden" 0   # don't cull hidden surfaces
Attribute "cull" "backfacing" 0   # don't cull backfacing surfaces
Attribute "dice" "rasterorient" 0   # view-independent dicing !
```

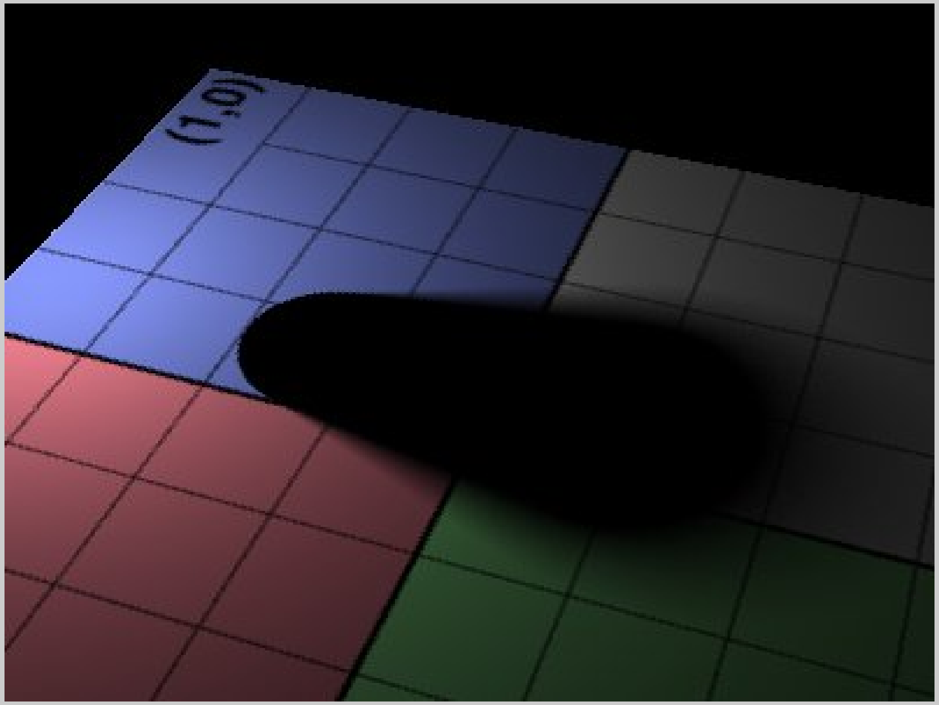




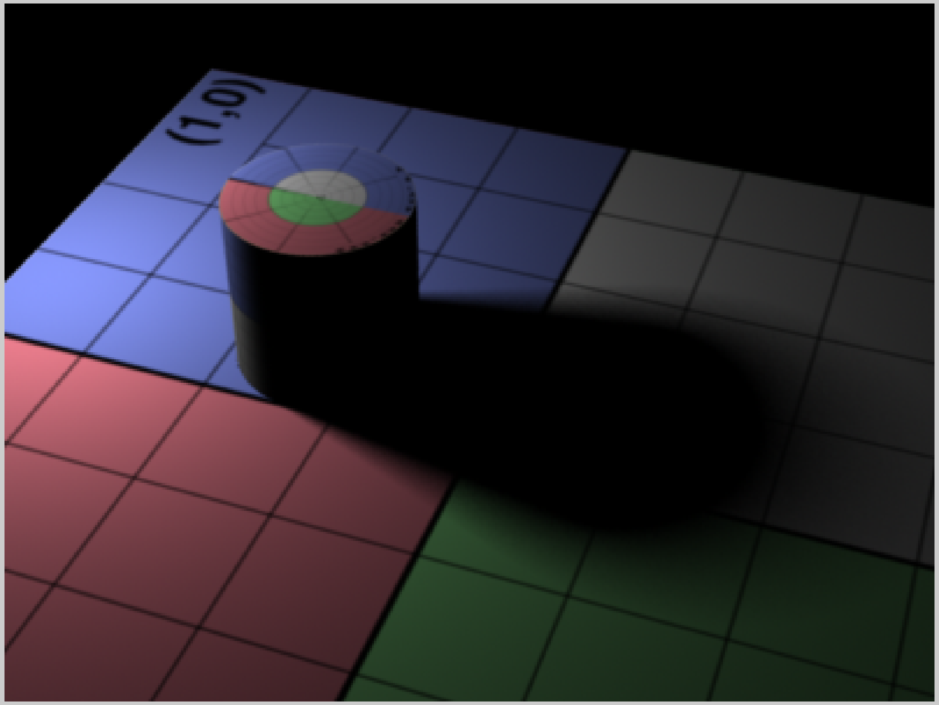
Culling on vs. off

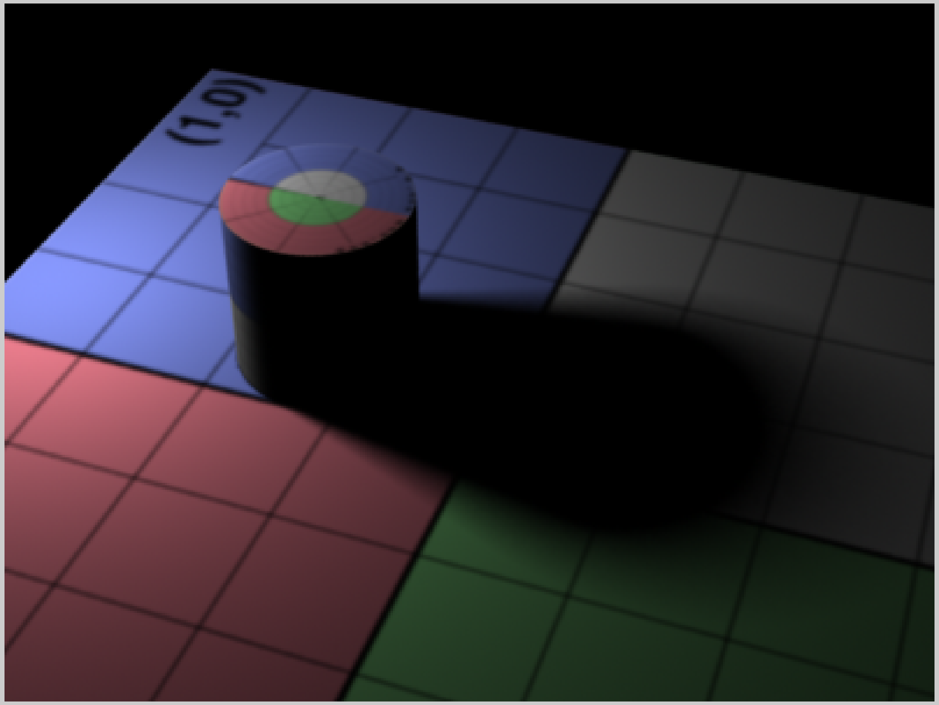
```
Attribute "cull" "hidden" 0   # don't cull hidden surfaces
Attribute "cull" "backfacing" 0   # don't cull backfacing surfaces
Attribute "dice" "rasterorient" 0   # view-independent dicing !
```





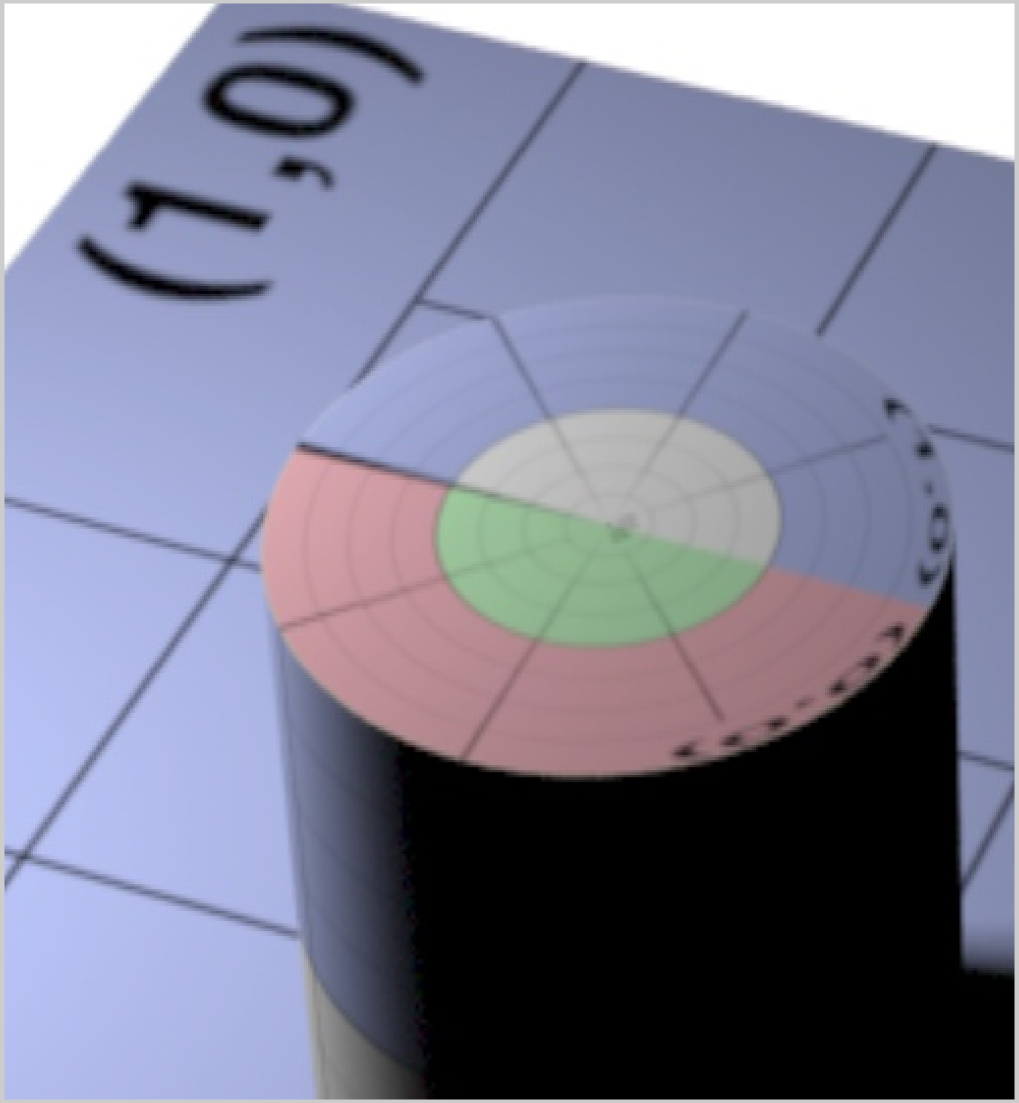
Point Cloud softer



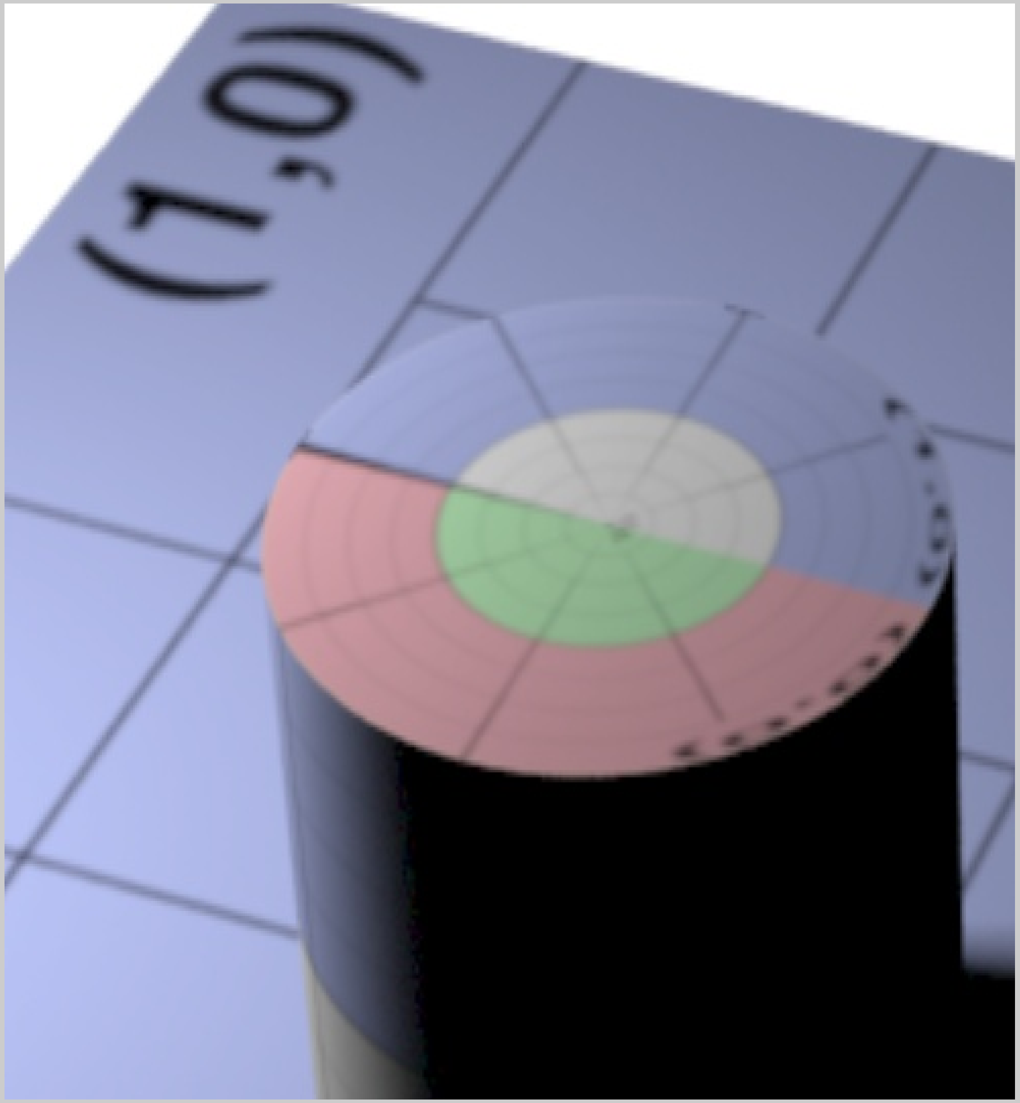


- Left image orig
- Right image re-rendered using point cloud

Point Cloud softer



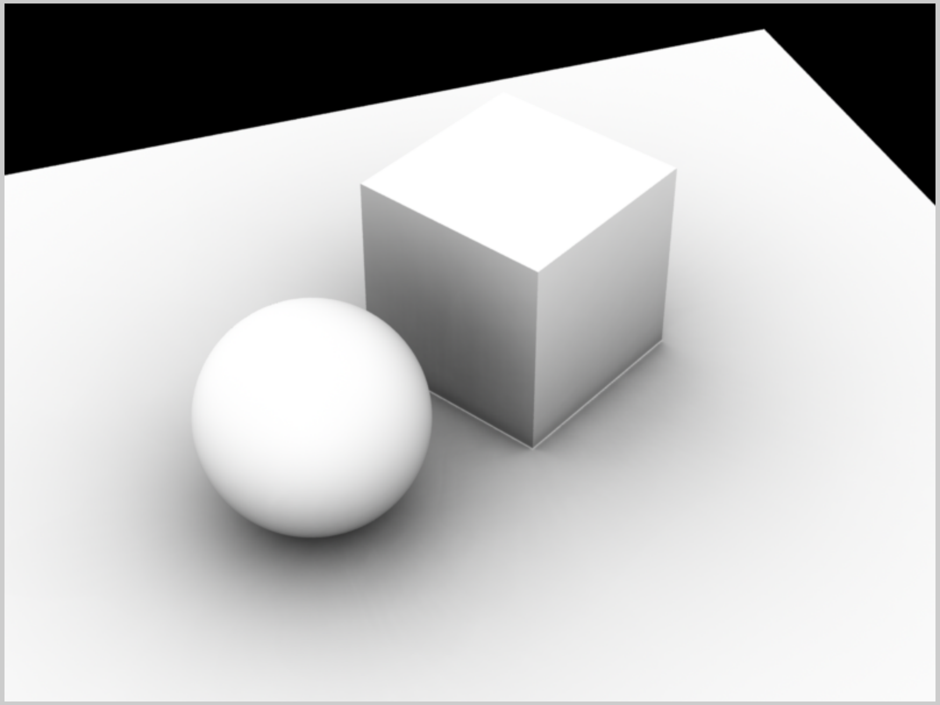
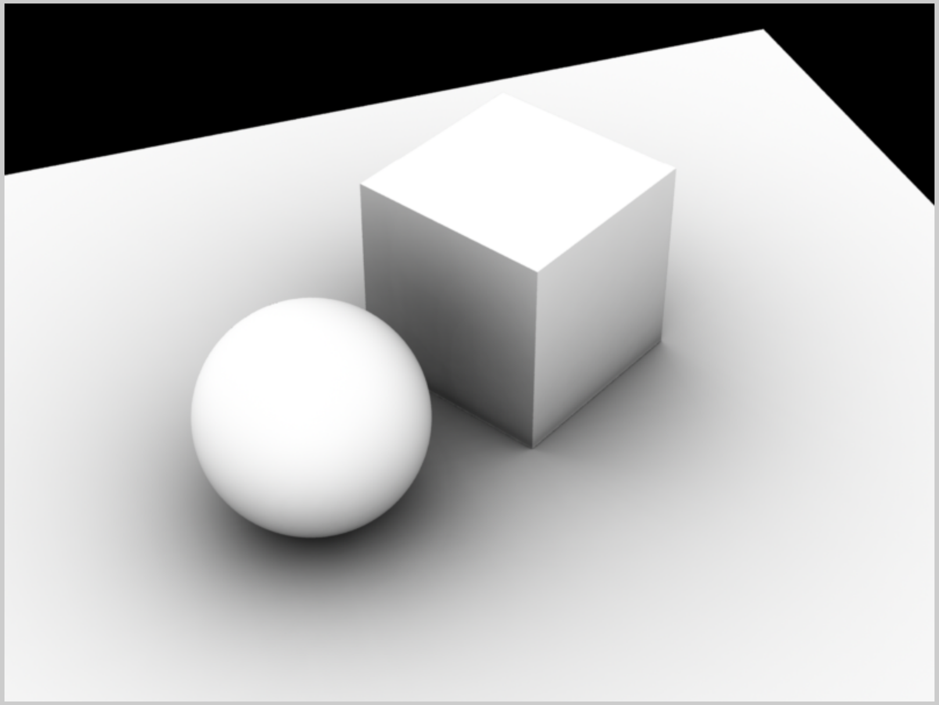
Left: Original



Right: Point cloud

Point Cloud Occlusion

No RT
Bad, Prior to PRMan-13.0.4

Manipulate PC

- Can join maps
  - exclusion and inclusion maps
- ptfilter
  - process/filter maps particular use
    - subsurface scattering

Point Cloud downside

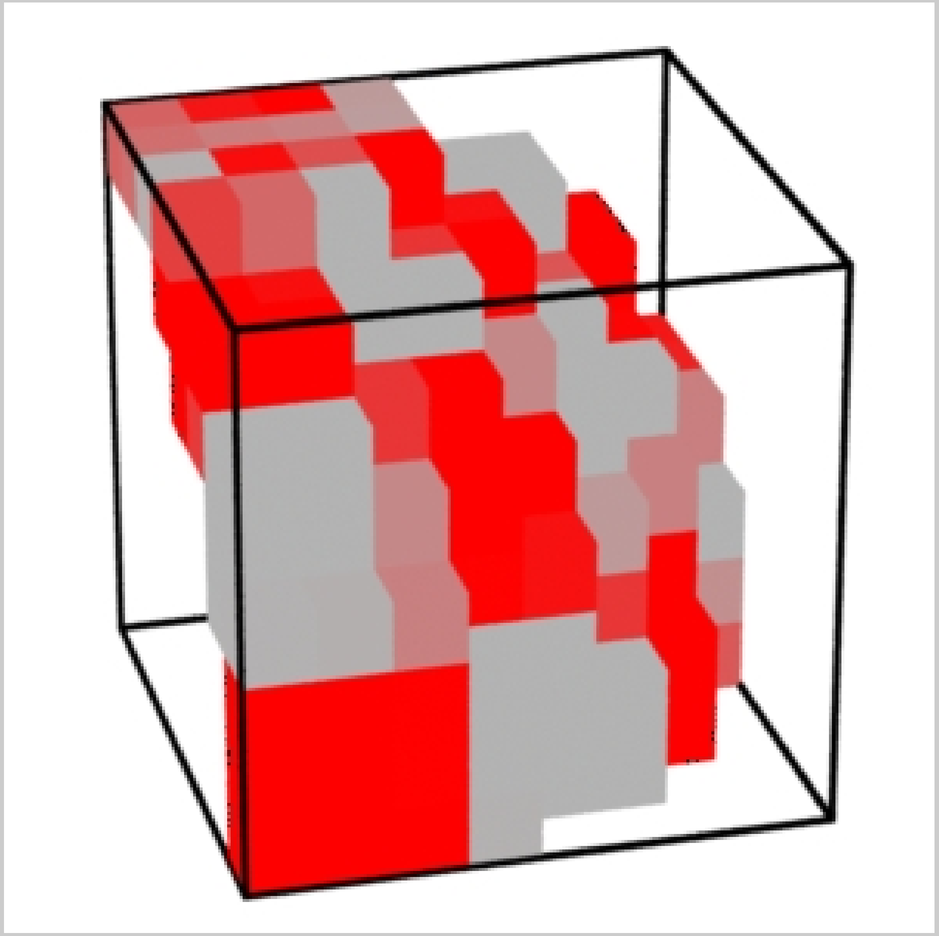
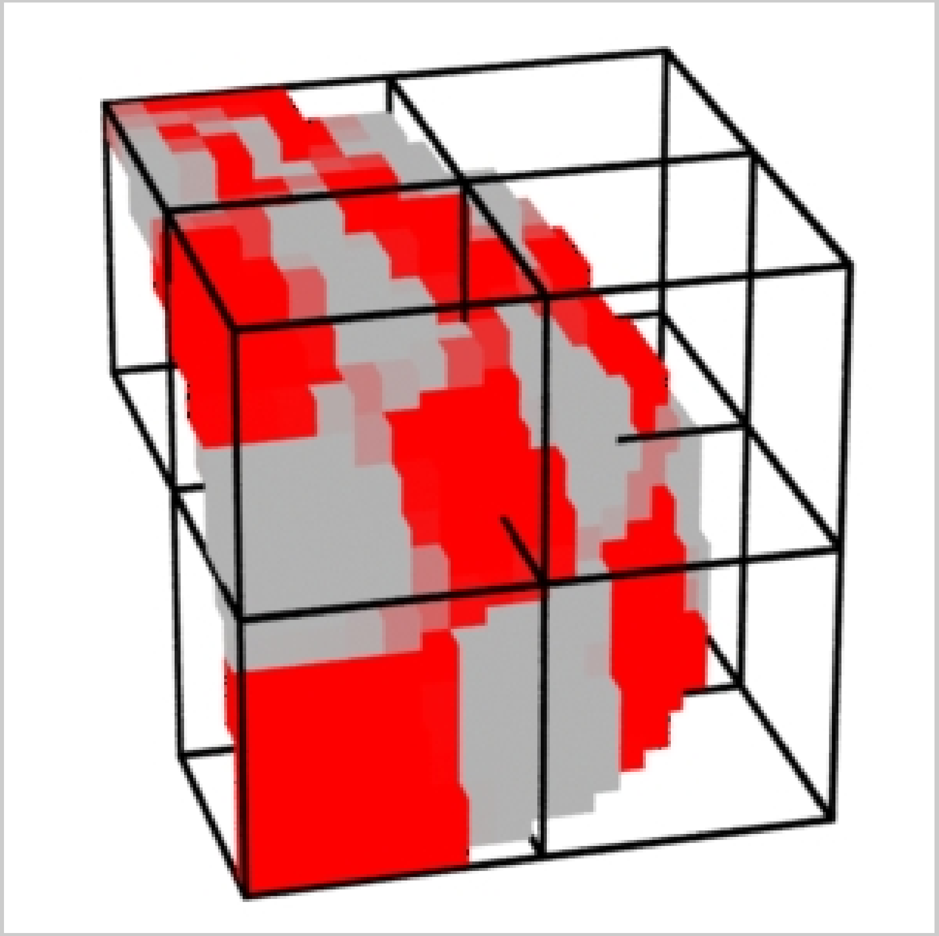
- Not as crisp as original source.
- Textures large (many factors larger. ie, 16x easy)
- Entire map must be loaded into memory to use; stays in memory
- Un-ordered and un-filtered must be filtered on fly. Slower

Brickmaps

- 3D MIP-mapped octree (brick = 8^3 voxels)
  - Only bring in portion needed
  - Memory efficient
  - Efficient filtering
  - Efficient caching (coherency)
  - Independent of surface representation or 2D parameterization
  - User specified accuracy – trade off accuracy with file size

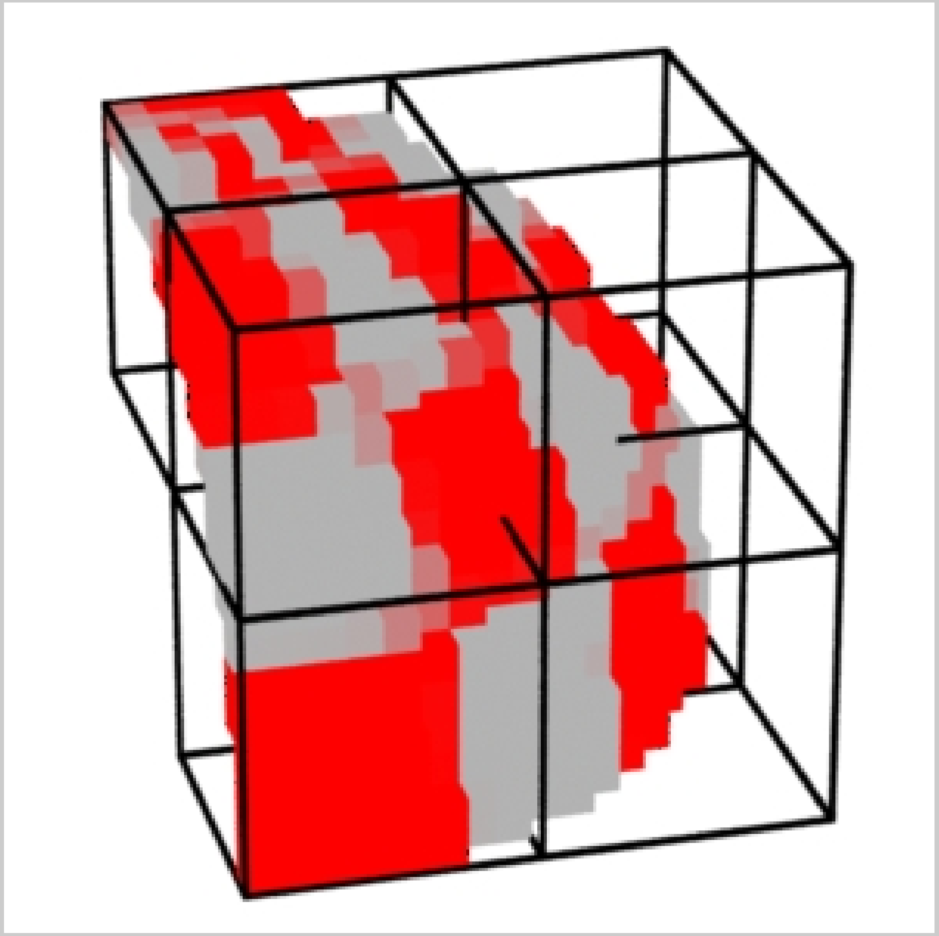
Brickmap L0 & L1

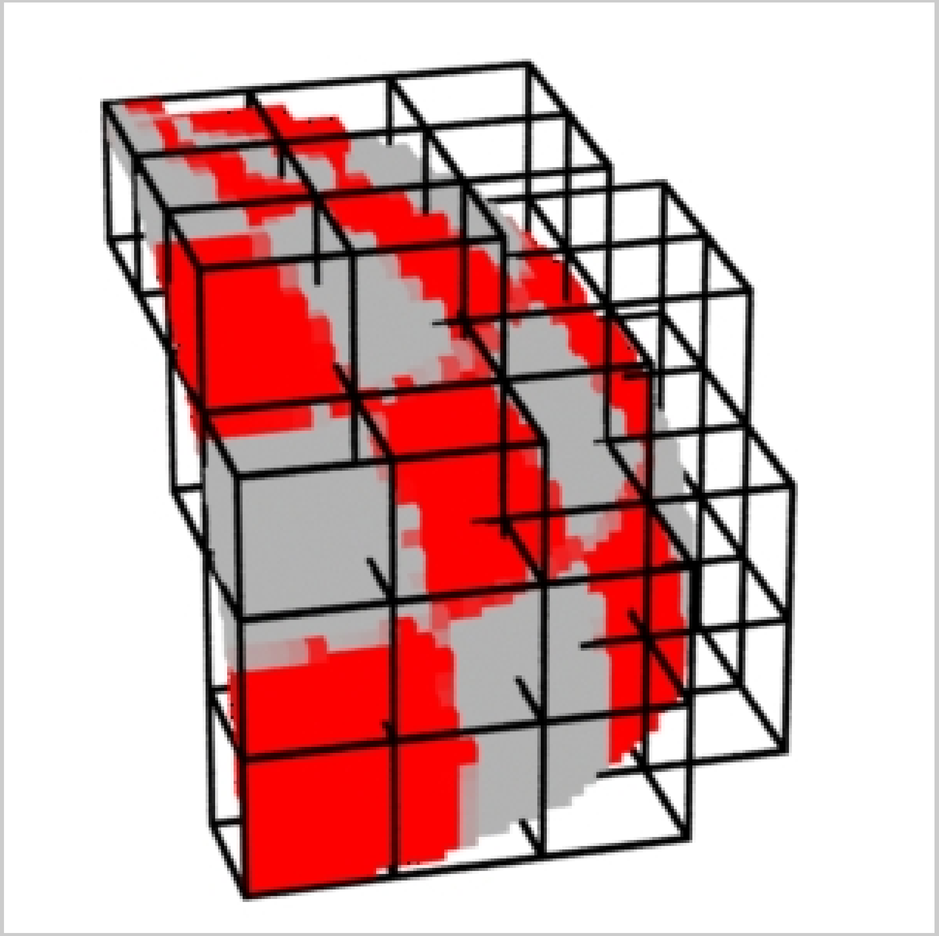
MIP Map refinement
Level 0 – least detail; Level 1 slightly more detail

Brickmap L1 & L2

MIP Map refinement
Each level down provides more detail





Brickmap uses

- Many places where point clouds can be used (but not everywhere)
  - PTC special case: occlusion, color bleeding
- Occlusion
- Color texturing
- Color bleeding
- Primitives (geometry / LOD)

Brick cache size

- Default 10MB cache. Similar to 2D texture cache

```
Option "limits" "brickmemory" 10240
```

- Not hard limit, still possible to use more than limit setting
- Multi-threaded
  - Each thread gets own cache.

Brickmap downside

- Filtering introduces artifacts
- Less accurate than point cloud
- Geometry chunky & gets larger

End Mapping: No time for

- SMSS – Single Map Soft Shadows (soft shadows) Brent Burley
- PTEX – new texture method Brent Burley
- Photon Mapping (and caustic maps)

Topics

- Speed vs Memory
- Mapping
- AOVs
- Raytracing

AOVs

- Arbitrary Output Variables
- Provide means to save out variables (data)
  - Own image (data) file
  - Independent of primary image
- Exposing certain internal calculations
  - Frequently same calc already doing
  - Or sometimes special calc for AOV
- Simultaneous with main image

AOV Examples

- Ambient
  - key fill bounce rim
  - other{1-5}
  - All
- CzDepthAlpha

```
point cP = transform("camera",P);
float avgAlpha = (Oi[0]+Oi[1]+Oi[2])/3;
CzDepthAlpha =
color(cP[2]*avgAlpha,avgAlpha,0);
```

AOVs Theory

- Multiple passes in time takes to do 1 (hopefully)
- Parsing scene graph once
- Processing geometry/shaders once

AOVS downside

- Nothing’s free
- Memory cost (fixed writing or not)
  - Track variables
- System I/O cost (related to # AOVs writing)
  - Writing simultaneous files (number of open files)
  - Could be 2-3x slowdown for all AOVs (above double digit)

Display statements

- Standard display

  ```
  Display "image.tif" "tiff" "rgba"
  ```

- AOV display

  ```
  Display "+myVar.tif" "tiff" "varying
  color myVar"
  ```

  - `"+"` must be first character in filename string
  - `"tiff"` file format to write out
  - `"myVar"` output variable name
    - From shader’s parameter list.

AOV display

```
Display "+myVar.tif" "tiff" "myVar"
```

- `"myVar"`
  - Any surface shader with matched name
    - Contributes to that image
- Convention have filename match var name
- RIB match to known variable
  - Write known filename

Minimal AOV display

```
Display "+OUTFILE" "IMAGE_FORMAT"
"SHADER_OUT_VAR"
```

- `Display "+N.tif" "tiff" "varying normal N"`
  - Image name
  - Image format
  - Output variable to track
- Produce 32bit image
- Share primary image
  - Filter and filter size

AOV display

```
Display "+fname" "ftype" "aovVariable"
"quantize" [zval oneval minval maxval] #int
"dither" [dval] #float
"filter" "filterName"
"filterwidth" [X Y] #float
```

- `"quantize" [0 255 0 255]  #8bit`
- `"quantize" [0 65535 0 65535] #16 bit`
- `"quantize" [0 0 0 0]  #float`

More AOV Display

- Possible to combine AOVs into 1 file
- Example adds alpha channel to AOV file
- NOTE: Requires use of DisplayChannel

```
DisplayChannel "varying color ambientAll"
DisplayChannel "varying float alpha"
Display “+ambientAll_alpha” “tiff”
“ambientAll,alpha”
```

Same filters

- AOVs take same filter as primary image
  - Gaussian
  - Box
  - Triangle
  - CatmulRom
  - Etc

5 Special AOV Filters

- Use where need absolute value
  - When blending doesn’t make sense
- 3 similar to depth map generation
  - Min
  - Max
  - Average
- 2 depth value used comparison first
  - Zmin
  - Zmax

TIPS

- Generally good idea use same filter/size
  - (Not applicable to 5 special AOV filters)
  - Otherwise not line up when compositing
- Use opacity threshold (zthreshold) 0
  - Get record of all semi-transparent objects
  - Otherwise may get black pixels

AOV per Light?

- AOVs come from surface or volume shaders
- Lights called via surface illuminance block

AOV per Light?

- AOVs come (only) from surface shaders
  - Not lights or displacement
  - Surface account for every known/possible light name
  - Have a reserved variable to match
  - A-priori
    - Known when shader built

Tracking lights

- Choices are
  - Customize shaders per scene
  - Set of shaders to account for some arbitrary light number combo.
  - Come up with max number of lights (100?)
  - Categorize lights
    - Use limited set of categories account for any number of lights

AOV per Light?

- Many illuminance loops in surface shader
  - Ambient
  - Diffuse
  - Specular
  - Reflection
  - Etc.
- Each light tracking
  - Only handled in illuminance

Tracking lights

- Customize shaders per scene
  - extra work; hard to track/maintain
- Set of shaders to account for some arbitrary light number combo.
  - Extra work; hard to track/maintain
- Come up with max number of lights (100?)
  - hard to track (how will remember which was light 73 when compositing)
- All three incur too high cost to system

Tracking Lights

- First three choices: Too high cost
  - 100 lights
    - 4 illum calls (ambient, diffuse, etc.)
      - 400 AOVs
    - LooksA + LooksB = X2
      - 800 AOVs!

AOV per Light?

- Compromise/balance
  - Group lights into pre-specified categories
  - Doesn’t matter how many lights still same amount
  - Track behavior against limited categories per illuminance type
    - 10 categories * 4 illum * 2 looks = 80 AOVs

Topics

- Speed vs Memory
- Mapping
- AOVs
- Raytracing

Ray Tracing (RT) Pipeline

- Not well documented
- Will attempt to cover in later talk

RT Pipeline

- PRMan Hybrid Renderer
  - Reyes Renderer 17+ years old
  - RT only 4 years old (prman-11)
    - Still evolving
- RT Impacts Reyes efficiencies
- Multiple Geo caches
  - 3 Ray classifications
    - Alternate tessellations

RT

- How impact Reyes efficiencies
- Control RT
  - lessen impact to Reyes

Reyes Review

- Covered Reyes pipeline
  - Work in discrete buckets
    - Process only what is needed when it is needed.
    - Memory efficient
    - Get rid of geometry data when done (bucket)

RT Pipeline

- Messes with Reyes efficiencies
  - Geometry not easily culled or freed
    - RT never culled/freed
      - Even off screen
    - Non RT not culled/freed
      - Even off screen
        - If used as reflection by RT surface
  - Code from one shader might call multiple shaders on other geos
  - Reason for memory spike

Shadow Pass vs. RT

- Shadow Pass
  - Multi-pass simplicity
  - Multiple alternate RIBs
    - What is/isn't there
    - What executes/not
    - Depending on pass context
- RT – not as convenient/simple
  - Single pass on the fly (usually with color)
  - Want/need similar controls

RT pipeline

- Need way of controlling scene description
- Control limits: speed & memory
  - What geometry is available
  - What shaders are run/not run
- Minimize impact of RT to only what is really needed
- RT objects not culled/freed!
  - Off screen reflected object never released (including non-RT reflected objects)

RT

- Visibility & trace sets
  - Control what is visible to what
- Shader hit mode
  - Control if shaders should run or not

RT Visibility Controls

```
Attribute "visibility"
"int camera" [1]
"int diffuse" [0]
"int specular" [0]
"int transmission" [0]
"int photon" [0]
"int midpoint" [0]
```

- Turning on middle 4
  - Tells renderer can't culled or freed

RT Visibility Controls

```
Attribute "visibility" "int camera" [1]
```

- Primary ray (camera)
- Original Reyes behavior
- Set to 0 if invisible to camera

RT Visibility Controls

```
Attribute "visibility" "int diffuse" [0]
```

- Diffuse rays
  (not diffuse illumination, ie. Kd)
  - PRMan distinguishes 2 types of rays for efficiency reasons
- Emitted by RSL function calls
  - occlusion()
  - indirectdiffuse()
  - Certain cases of gather()

RT Visibility Controls

```
Attribute "visibility" "int specular" [0]
```

- Specular rays
  (not specular illumination ie, Ks)
  - PRMan distinguishes 2 types of rays for efficiency reasons
- Emitted by RSL function calls
  - trace()
  - environment()
  - Certain cases of gather()
- Different ray types use different tessellations

RT Visibility Controls

```
Attribute "visibility"
"int transmission" [0]
```

- (RT) Shadow rays
- RSL transmission()

RT Visibility Controls

```
Attribute "visibility" "int photon" [0]
```

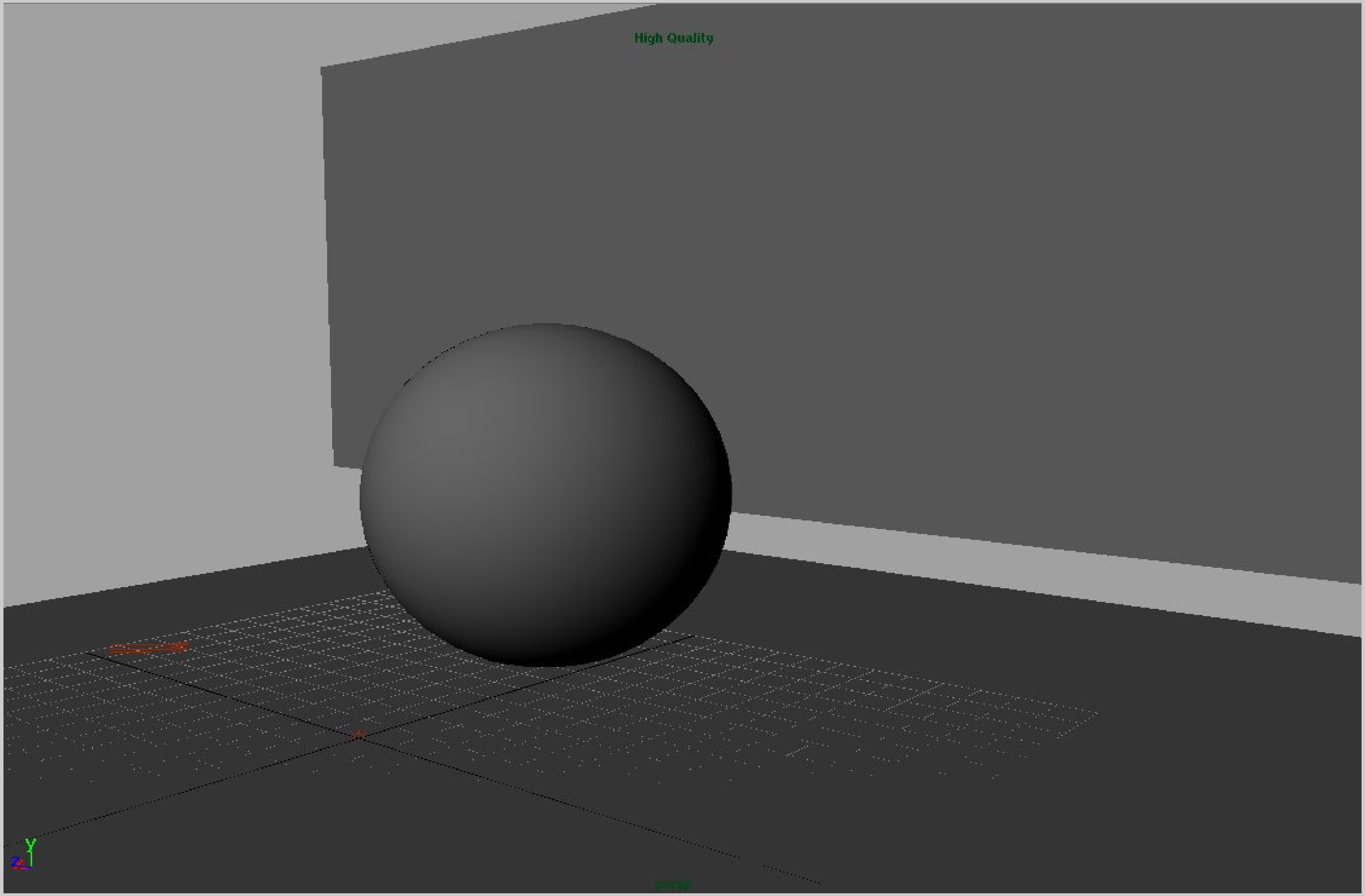
- Photon map creation
  - Separate pass
  - Support RIB archive multi-use

Visibility Controls

```
Attribute "visibility" "int midpoint" [0]
```

- Control what visible for midpoint shadow generation
- Separate pass
  - Multi-use RIB archive
- Not RT control!

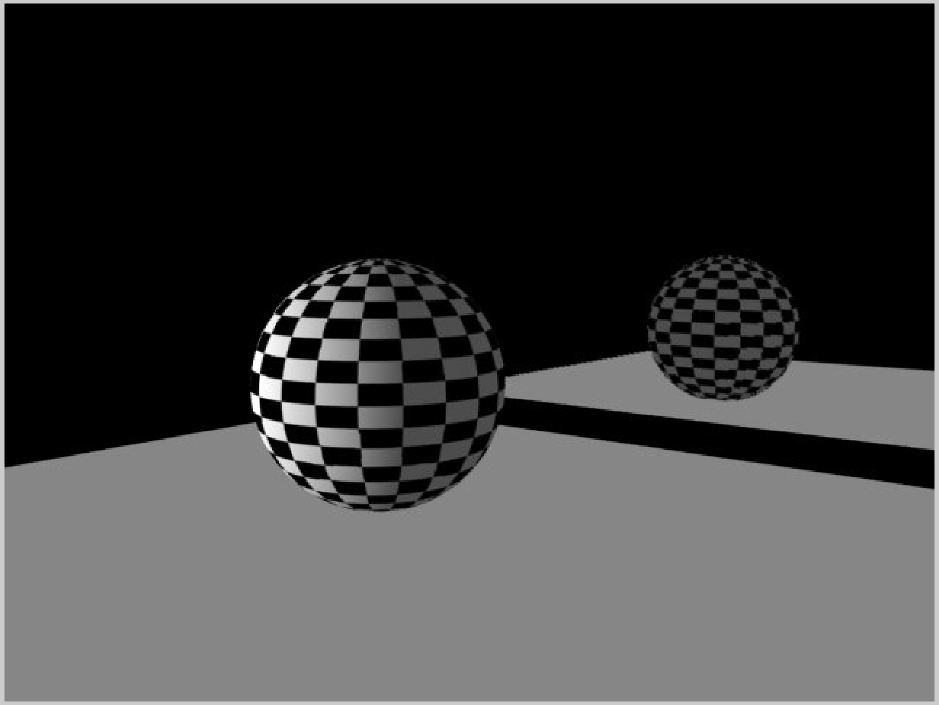
Mirror reflecting sphere and ground



RT visibility example 1

- Floor: RT
  Sphere: RT

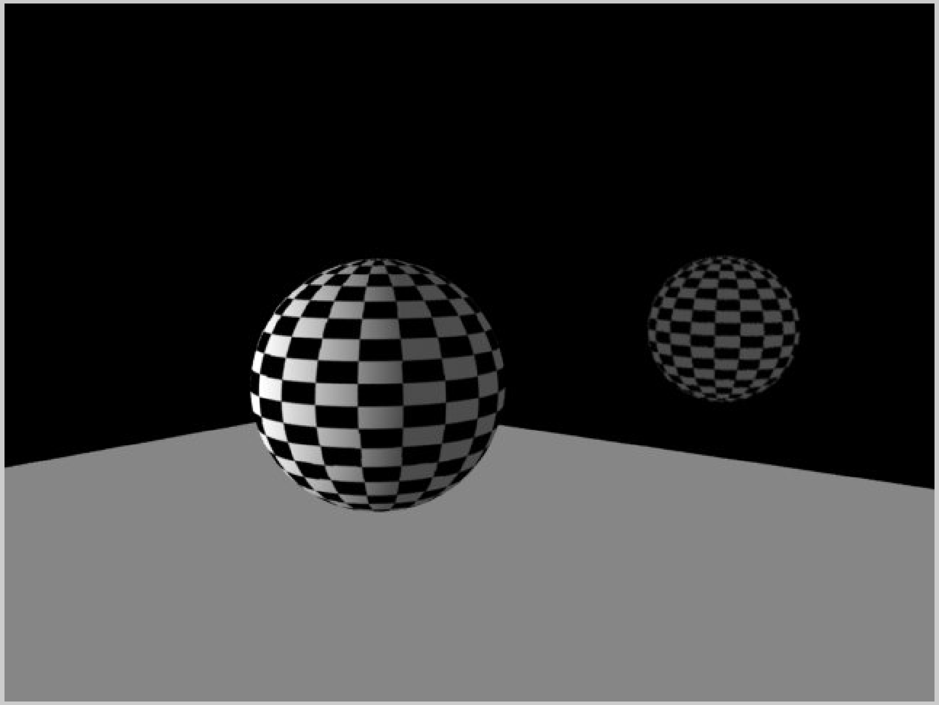
```
Attribute "visibility"
"camera" 1
"specular" 1
```



RT visibility example 1

- Floor: no RT

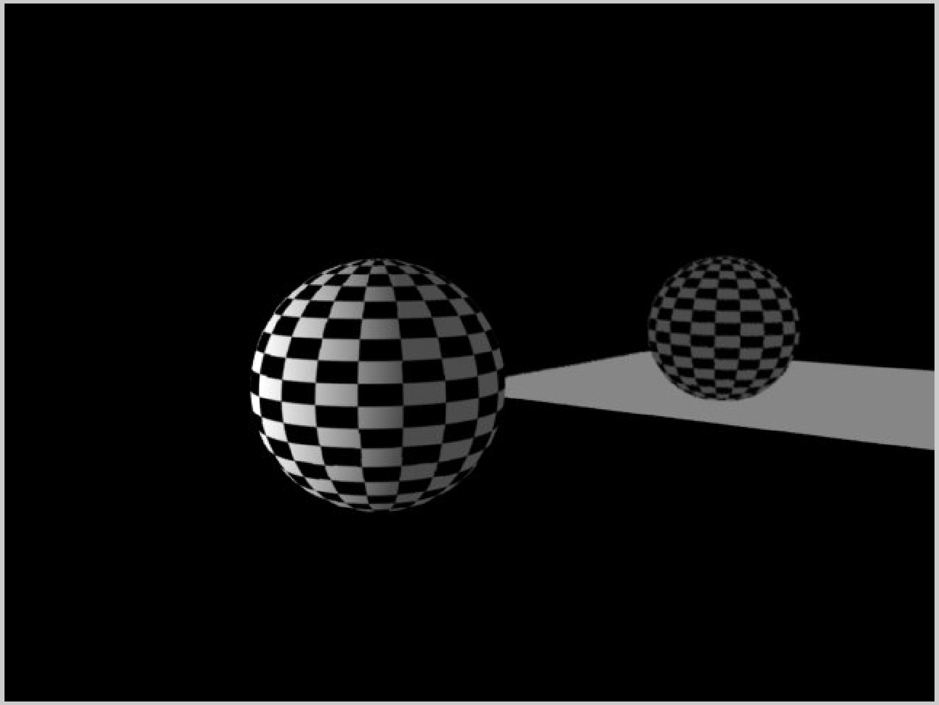
```
Attribute "visibility"
"camera" 1
"specular" 0
```



RT visibility example 2

- Floor: hide from camera

```
Attribute "visibility"
"camera" 0
"specular" 1
```



visibility trace?

- Pixar stopped using this in prman-12
  - When renderer started todistinguish between different ray types.
- Doesn't offer the fine level of control as the other visibility statements provide.
- It no longer makes sense to use anymore
  - It is obsolete and sometime the renderer won't support it anymore.

RT Hitmode

- Once hit by a given ray type control what happens

```
Attribute "shade"
"string diffusehitmode" ["primitive"]
"string specularhitmode" ["shader"]
"string transmissionhitmode" ["shader"]
"string camerahitmode" ["shader"]
```

- Controls shader execution on hits
  - Primitive – Cs/Os Color/Opacity attribute (cheap)
  - Shader – run shader to return Ci/Oi (expensive)

RT Hitmode

- Controls what to do when object hit by diffuse ray.

```
Attribute "shade"
"string diffusehitmode" ["primitive"]
```

  - occlusion()
  - indirectdiffuse()
  - Certain cases of gather()

RT Hitmode

- Controls what to do when object hit by specular ray.

```
Attribute "shade"
"string specularhitmode" ["shader"]
```

  - trace()
  - environment()
  - Certain cases of gather()

RT Hitmode

```
Attribute "shade"
"string transmissionhitmode" ["shader"]
```

- transmission()
  - (RT) Shadow rays

RT Hitmode

```
Attribute "shade"
"string camerahitmode" ["shader"]
```

- Primary rays (camera)
- Not RT

RT Options

- Maxdepth
- Specular threshold

Maximum Depth

```
Option "trace" "int maxdepth" [10]
```

- Controls recursive depths before stopping
- Number of bounces/surfaces ray can hit
- Black if reached
  - Unless have shader smarts (map lookup)

Specular Threshold

```
Option "trace"
"float specularthreshold" [10]
```

- Distinguishing between diffuse & specular rays
- Many RT calls have “coneangle” parameter
  - gives area of influence (hemisphere)
- Angular threshold
- Values greater than will be considered diffuse rays
- gather() result might be specular or diffuse
- indirectdiffuse() & occulsion() respect this metric
  - Possible to fire specular rays

RT Control Attributes

- Maxdiffusedepth
- Maxspeculardepth
- Displacements
- Bias
- samplemotion

Maximum Diffuse Depth

```
Attribute "trace"
"int maxdiffusedepth" [1]
```

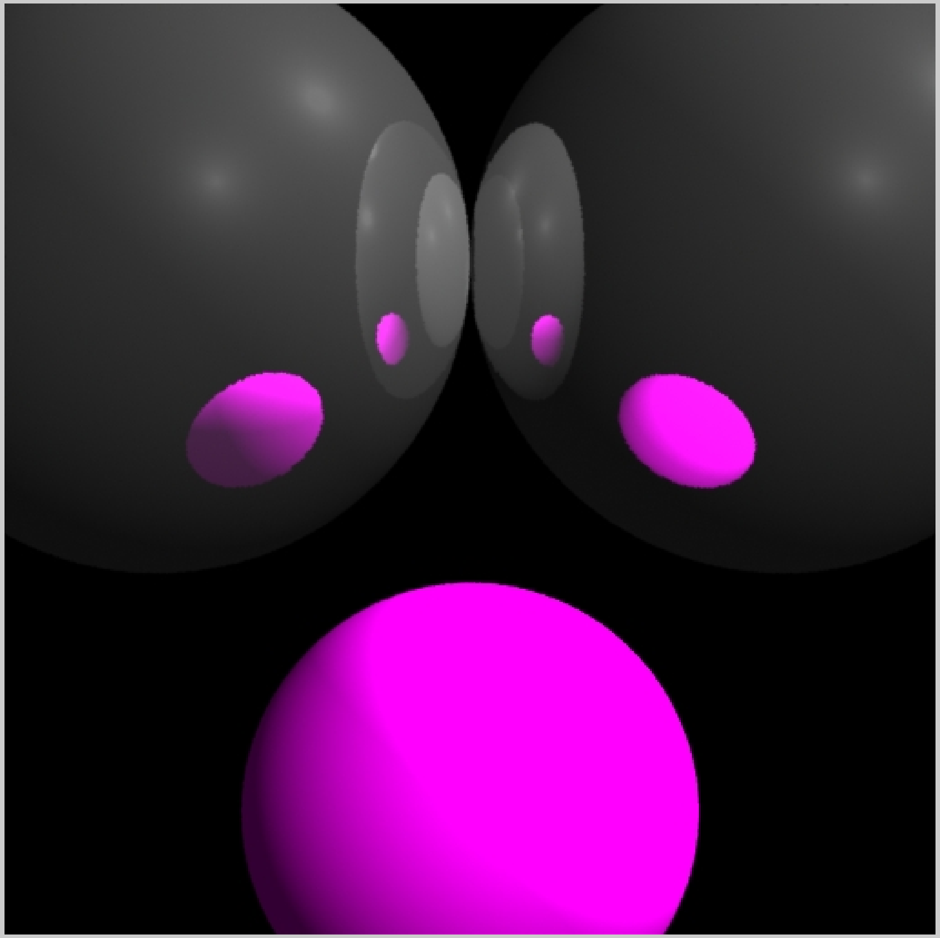
- Limit number of bounces for diffuse rays
- <= maxdepth option

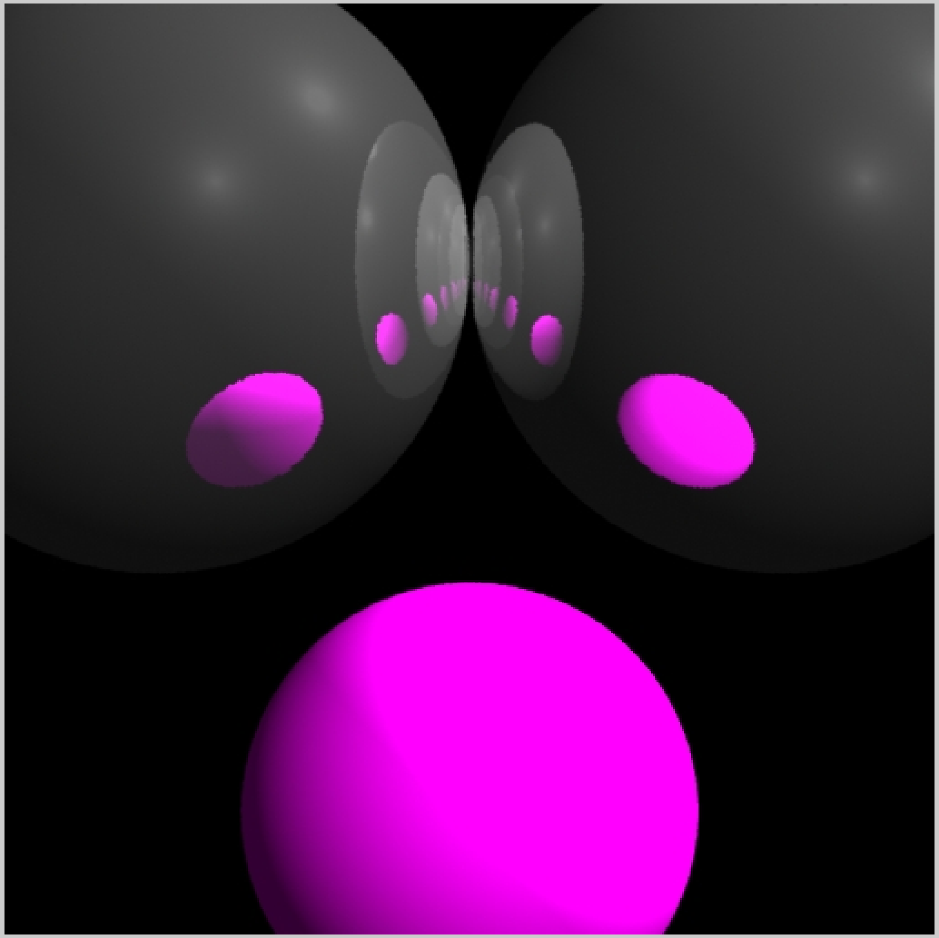
Maximum Specular Depth

```
Attribute "trace"
"int maxspeculardepth" [2]
```

- Limit number of bounces for specular rays
- <= maxdepth option

Attribute “trace” “int maxspeculardepth [7]





Default maxspeculardepth = 2

Trace Displacements

```
Attribute "trace" "int displacements" [0]
```

- Controls ray-primitive intersection
- If off
  - Bump mapped (less expensive)
    - Displacements run for shading purposes
    - Displacement ignored for ray-primitive intersection tests
- When on true displacements appear in RT results
  - More expensive; more intensive calc; more geo

- Bump results often acceptable

Trace Bias

```
Attribute "trace" "float bias" [.01]
```

- Prevents blotchy artifacts
  - If ray intersects itself (surface just left)
- Offset applied to ray origin (slightly off surface)
- Smaller numbers / longer renders
  - Higher precision = less prims trivially rejected
- Rough starting values .01 - .001 (I tend to use these; sometimes even .0001)

RT Motion Blur

```
Attribute "trace" "int samplemotion" [0]
```

- If motion blur is taken into account

Stitch Trace Enable

```
Attribute "stitch" "int traceenable" [1]
```

- Internal crack elimination within a given RT primitive with displacement
- Like stitching Mark described earlier. (Doesn't help edges between two pieces of geometry)
- Memory hit

Trace sets

- Visibility refinement through categories/sets
- Tracing relationship among objects

  ```
  Attribute "grouping" "string membership"
  ["NAME_OF_GROUP"]
  ```

  - RT RSL functions label rays (categories)
    - Which member(s) to hit (send rays to)
  - For example not do SSS on hair
  - Purely renderer grouping via RIB box attributes
    - Not a maya set

Setting Trace Sets

- Not typical attribute. Has relative mode!

  ```
  Attribute "grouping" "string membership"
  ["NAME_OF_GROUP"]
  ```

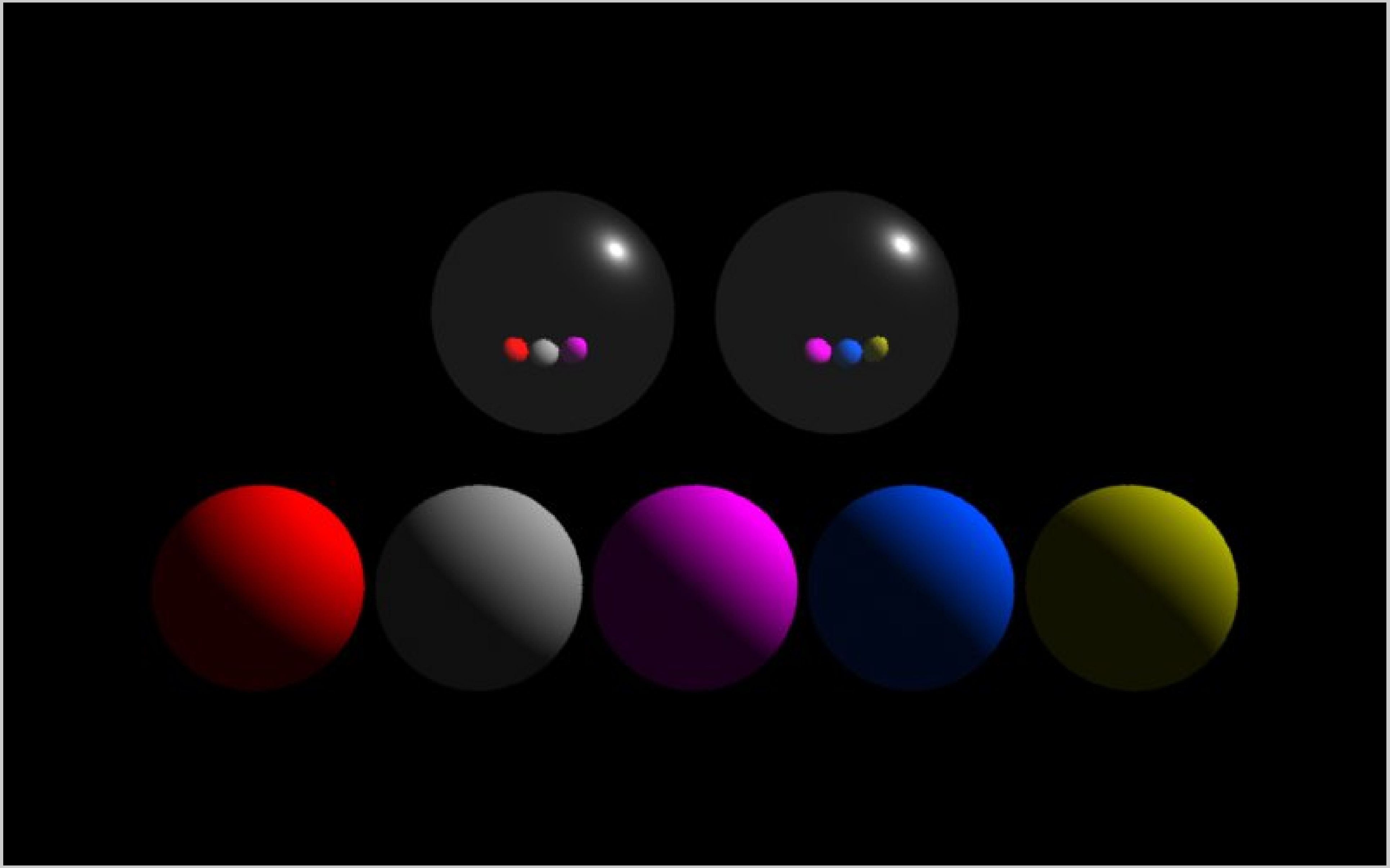
  - `NAME_OF_GROUP`
    - Absolute
      - `"name"`
      - `"list,of,names" "list of names"`
    - Relative
      - `"+additional,names"`
      - `"-without,names"`

Trace sets -- shader

- In addition to creating TS RIBbox attributes
  - Shaders need to be told what sets they are to distinguish.
  - Shader parameter to control some **subset** of the available traceable objects:

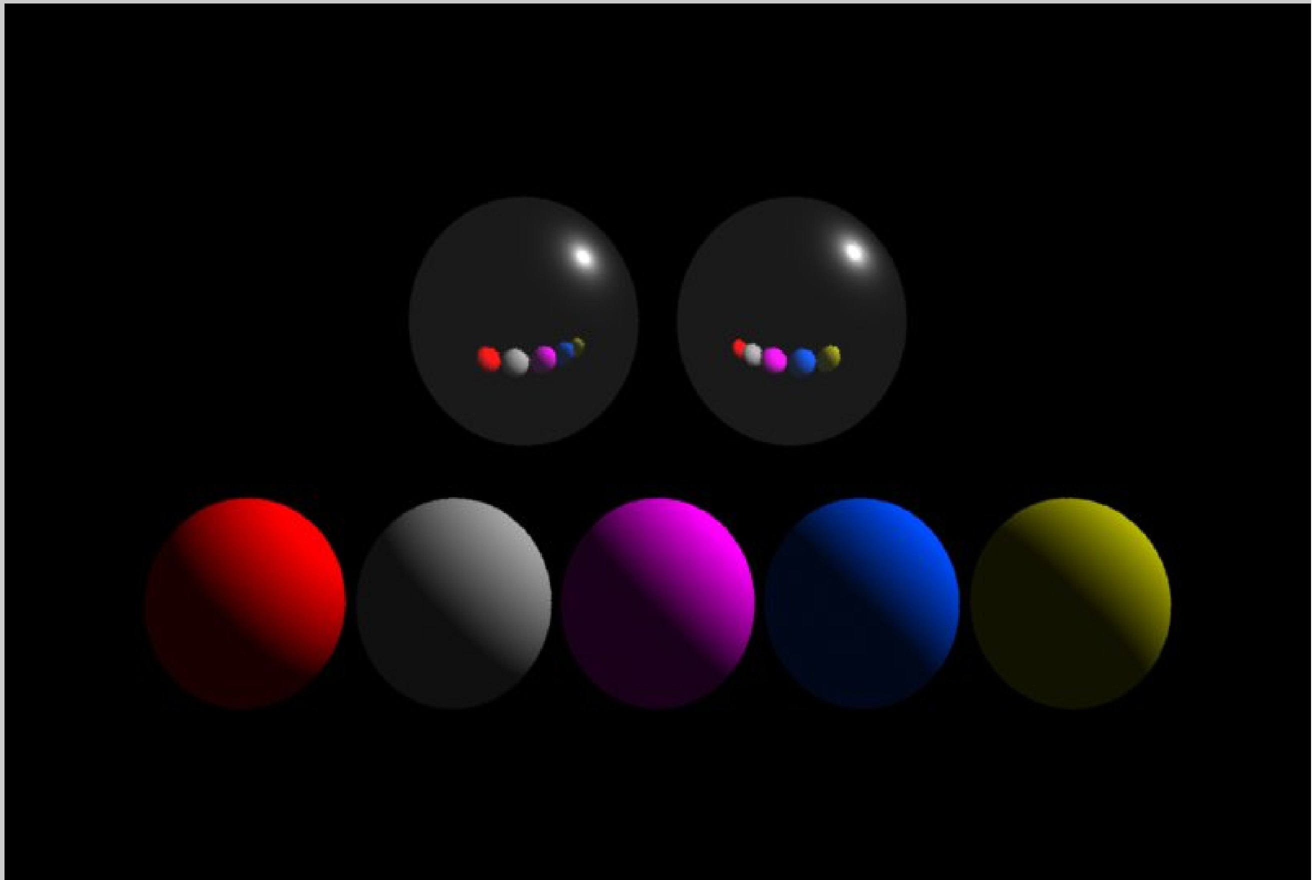
```
*TraceSet = "";
```

Trace Set Examples



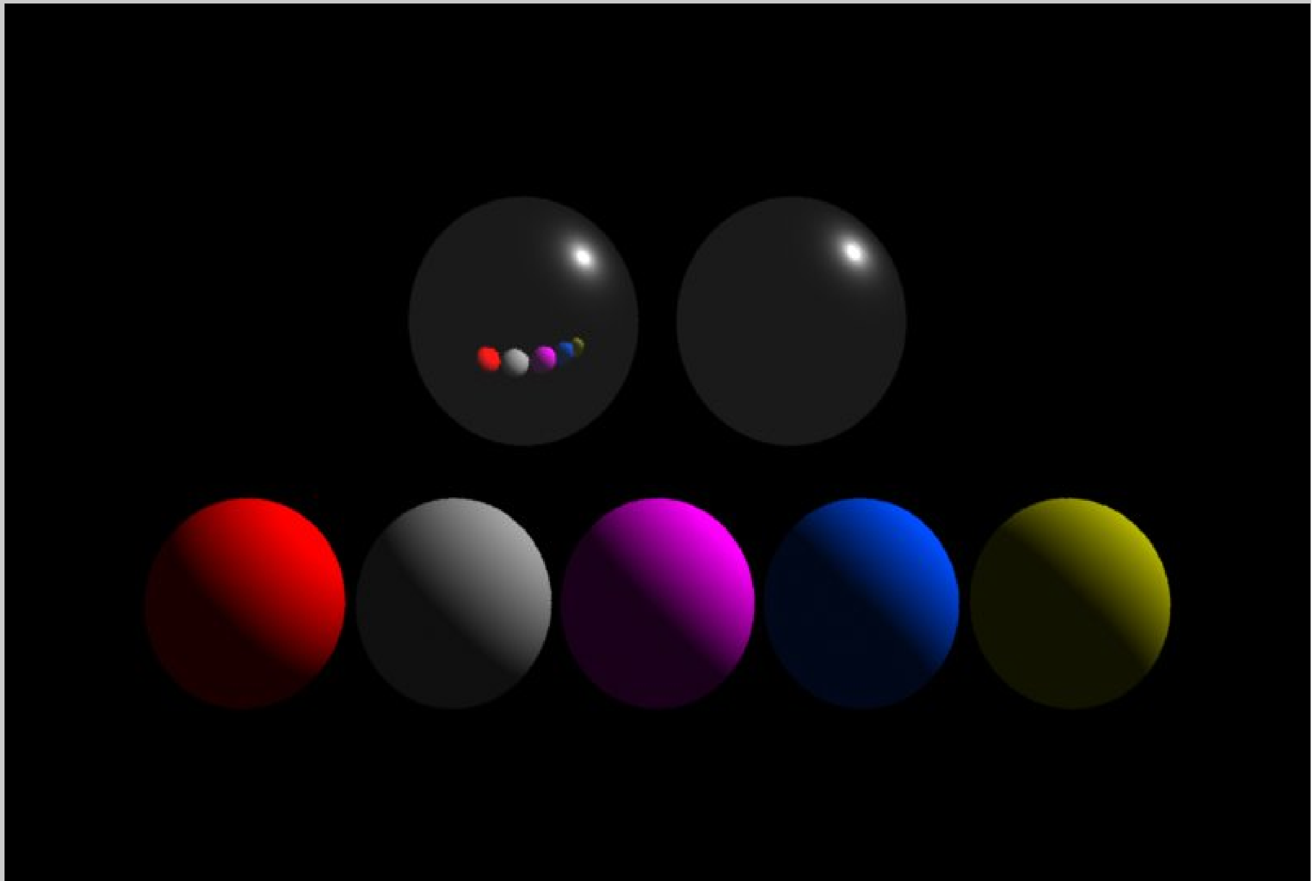
TS – Example 1

```
#Calling shader parameter “subset” instead of *TraceSet
#Right RT sphere
Surface "shinymetal2" "string subset" [""]

#Left RT sphere
Surface "shinymetal2" "string subset" [""]

Sphere 1
Sphere 2
Sphere 3
Sphere 4
Sphere 5
```



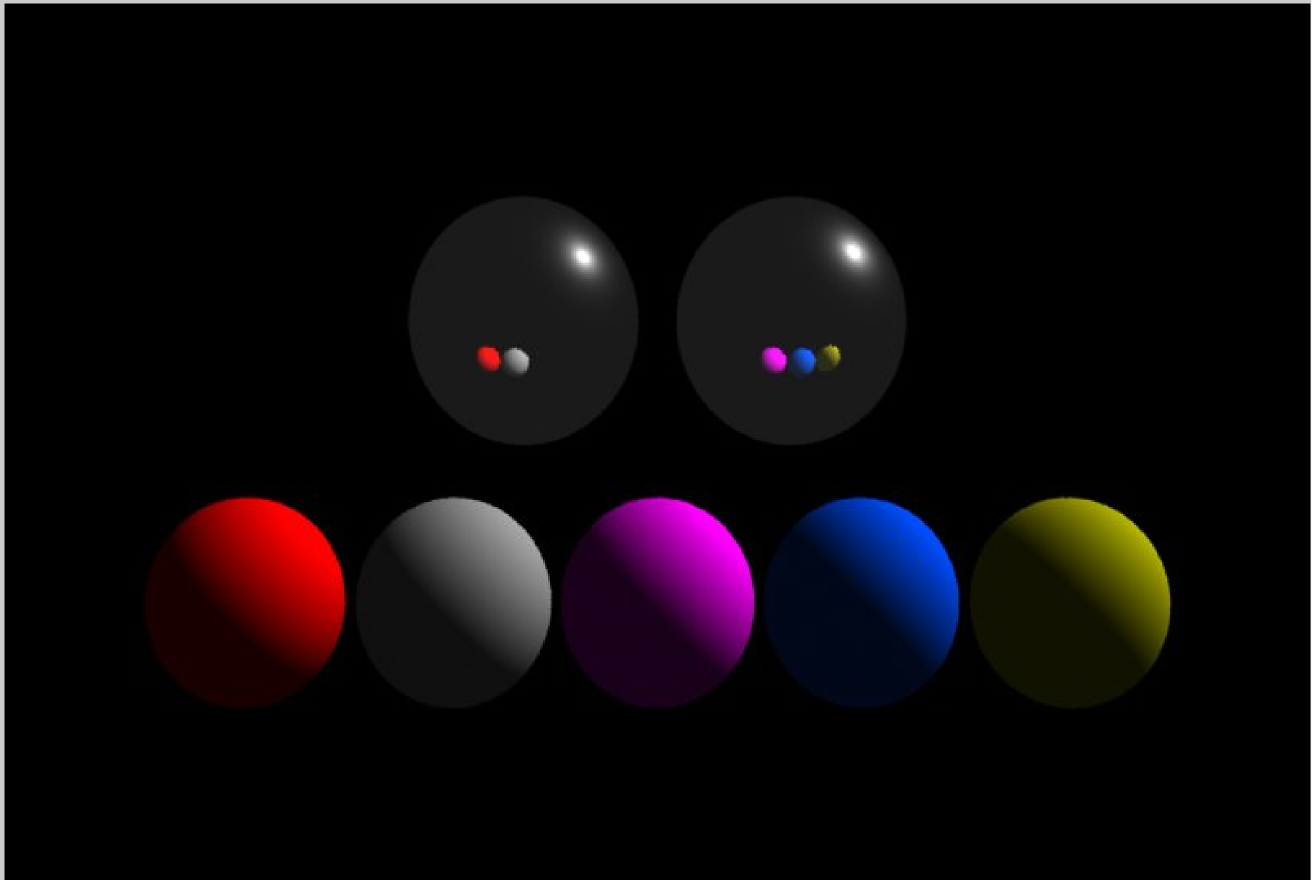
TS – “left” example

```
#Right RT sphere
Surface "shinymetal2" "string subset" ["right"]

#Left RT sphere
Surface "shinymetal2" "string subset" ["left"]

Attribute "grouping" "membership" ["left"]
Sphere 1
Sphere 2
Sphere 3
Sphere 4
Sphere 5
```



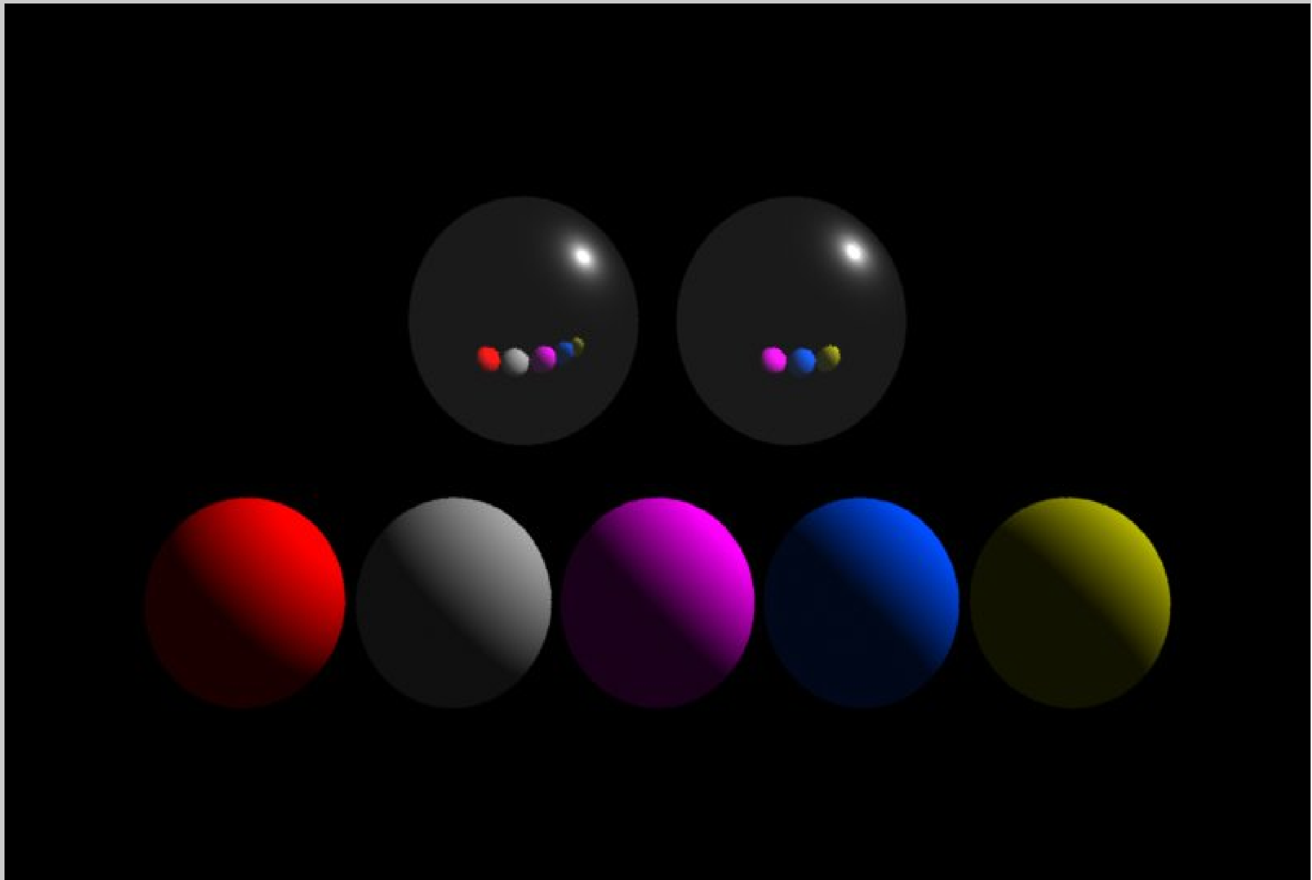
TS: Example 3 left/right

```
#Right RT sphere
Surface "shinymetal2" "string subset" ["right"]

#Left RT sphere
Surface "shinymetal2" "string subset" ["left"]

Attribute "grouping" "membership" ["left"]
Sphere 1
Sphere 2

Attribute "grouping"
   "membership" ["right"]
Sphere 3
Sphere 4
Sphere 5
```



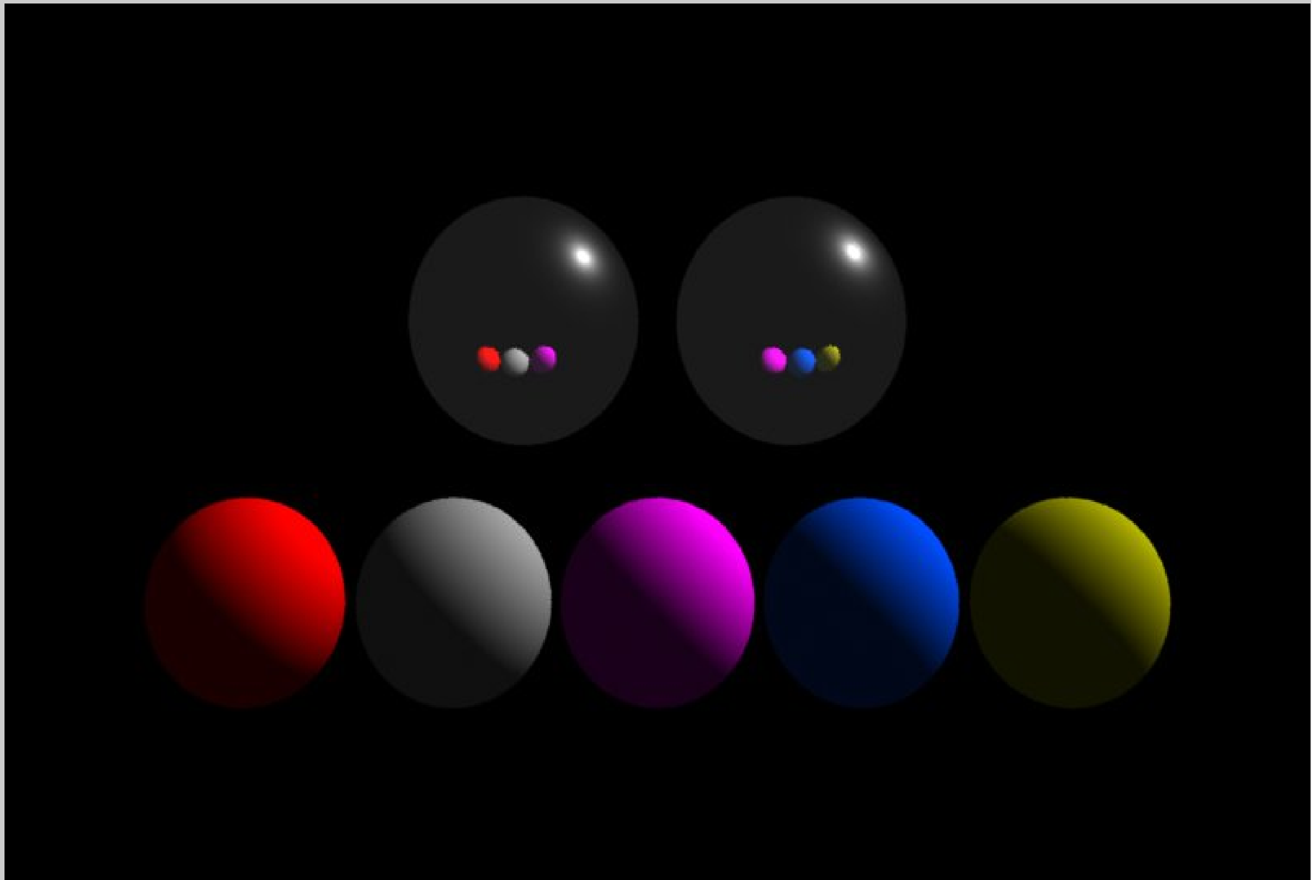
TS: Example4 left/+right

```
#Right RT sphere
Surface "shinymetal2" "string subset" ["right"]

#Left RT sphere
Surface "shinymetal2" "string subset" ["left"]

Attribute "grouping" "membership" ["left"]
Sphere 1
Sphere 2

Attribute "grouping"
   "membership" ["+right"]
Sphere 3
Sphere 4
Sphere 5
```



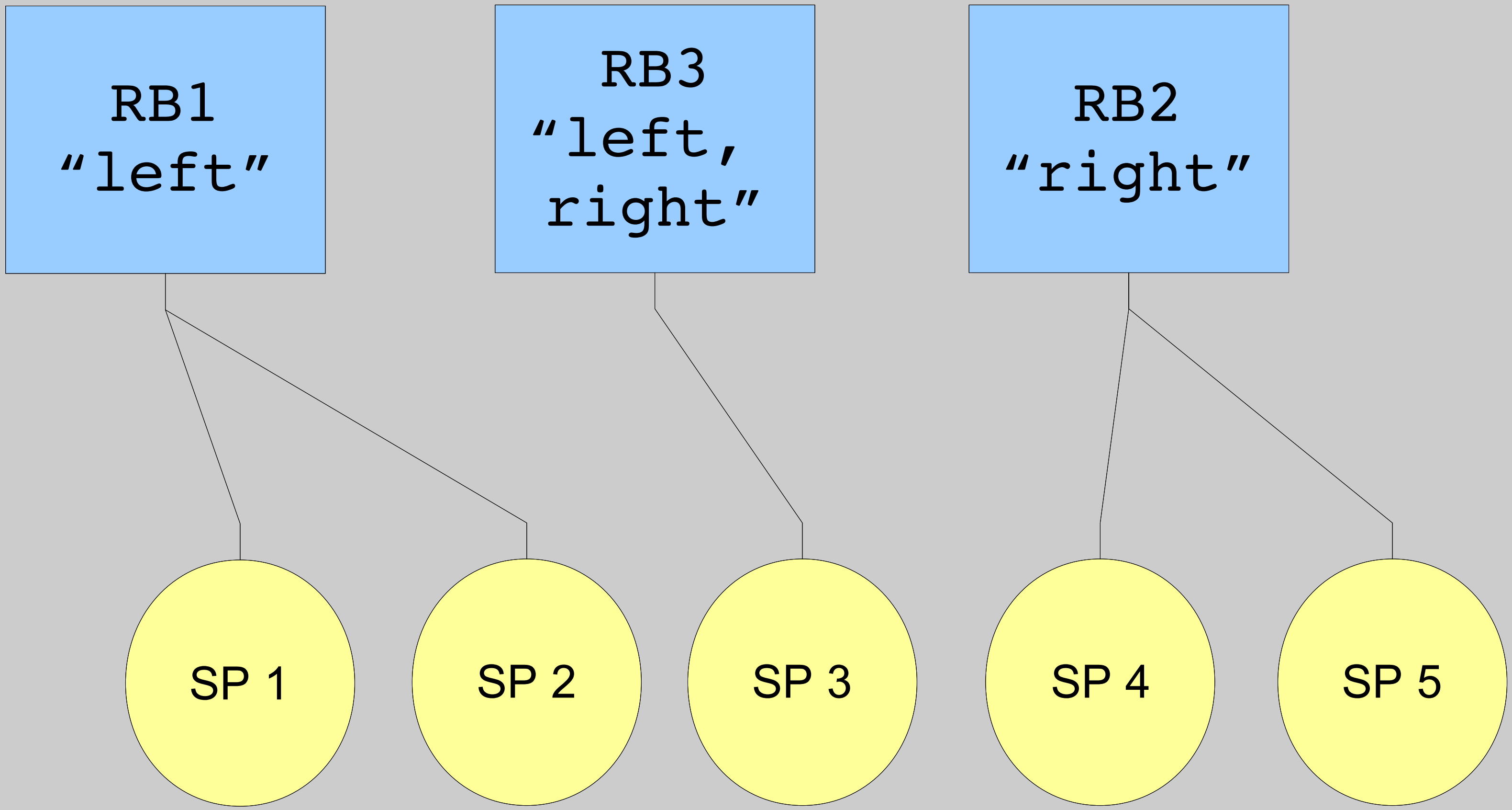
TS: Example 5 +right/-left

```
#Right RT sphere
Surface "shinymetal2" "string subset" ["right"]

#Left RT sphere
Surface "shinymetal2" "string subset" ["left"]

Attribute "grouping" "membership" ["left"]
Sphere 1
Sphere 2

Attribute "grouping"
   "membership" ["+right"]
Sphere 3
Attribute "grouping"
   "membership" ["-left"]
Sphere 4
Sphere 5
```



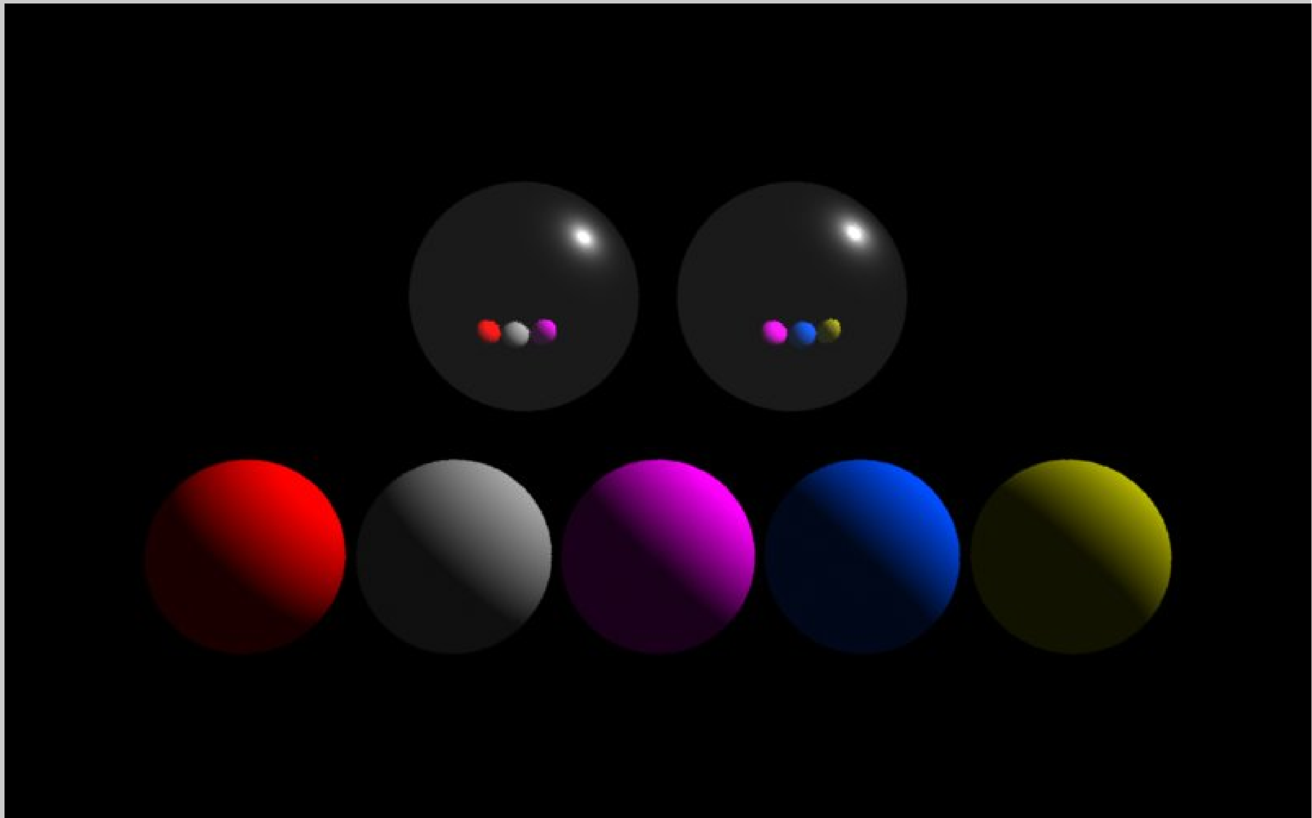
TS: direct approach

```
#RIBbox membership for “left” “left,right”
“right”
Attribute “grouping” “string membership”
[“”]
```



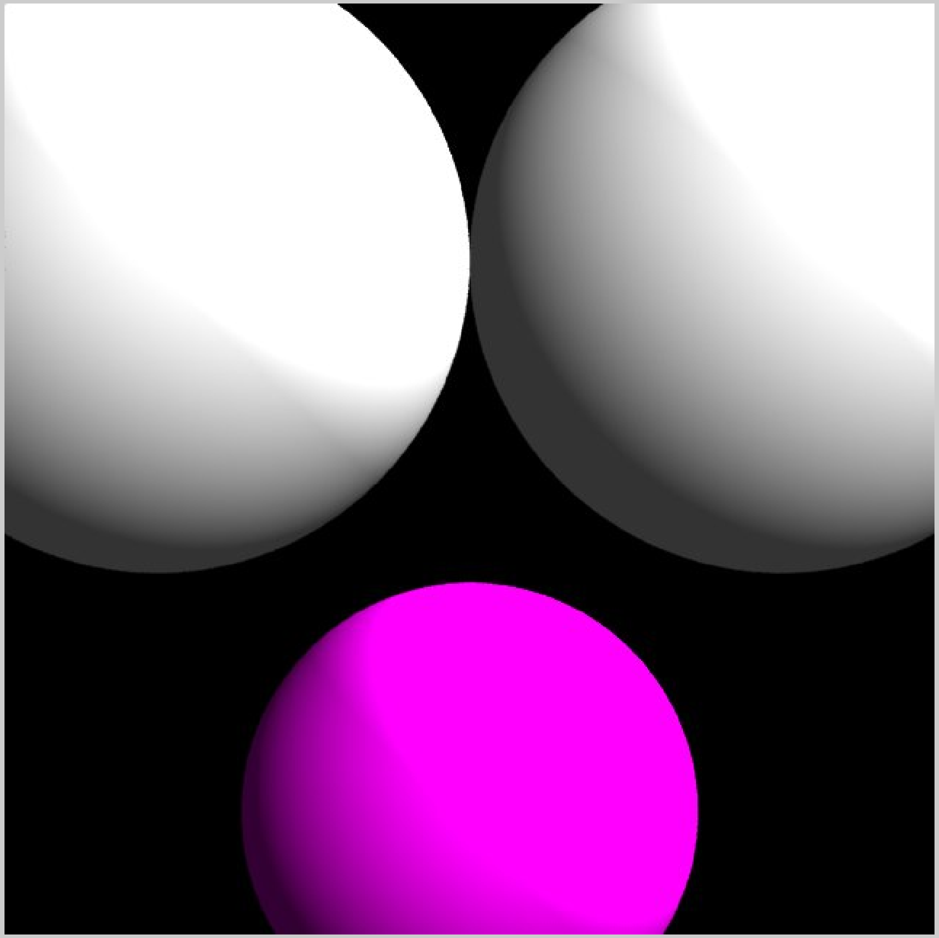
MS: Example 6 Direct approach

```
Surface "shinymetal2" "string subset" ["right"]
Surface "shinymetal2" "string subset" ["left"]
AB
#RIB Box RB1
Attribute "grouping" "membership" ["left"]
Sphere 1 AE
AB
#RIB Box RB1
Attribute "grouping" "membership" ["left"]
Sphere 2 AE
AB
#RIB Box RB3
Attribute "grouping"
   "membership" ["left,right"]
Sphere 3 AE
AB
#RIB Box RB2
Attribute "grouping"
   "membership" ["right”]
Sphere 4 AE
AB
#RIB box RB2
Attribute "grouping"
   "membership" ["right”]
Sphere 5 AE
```



RT control: shader behavior

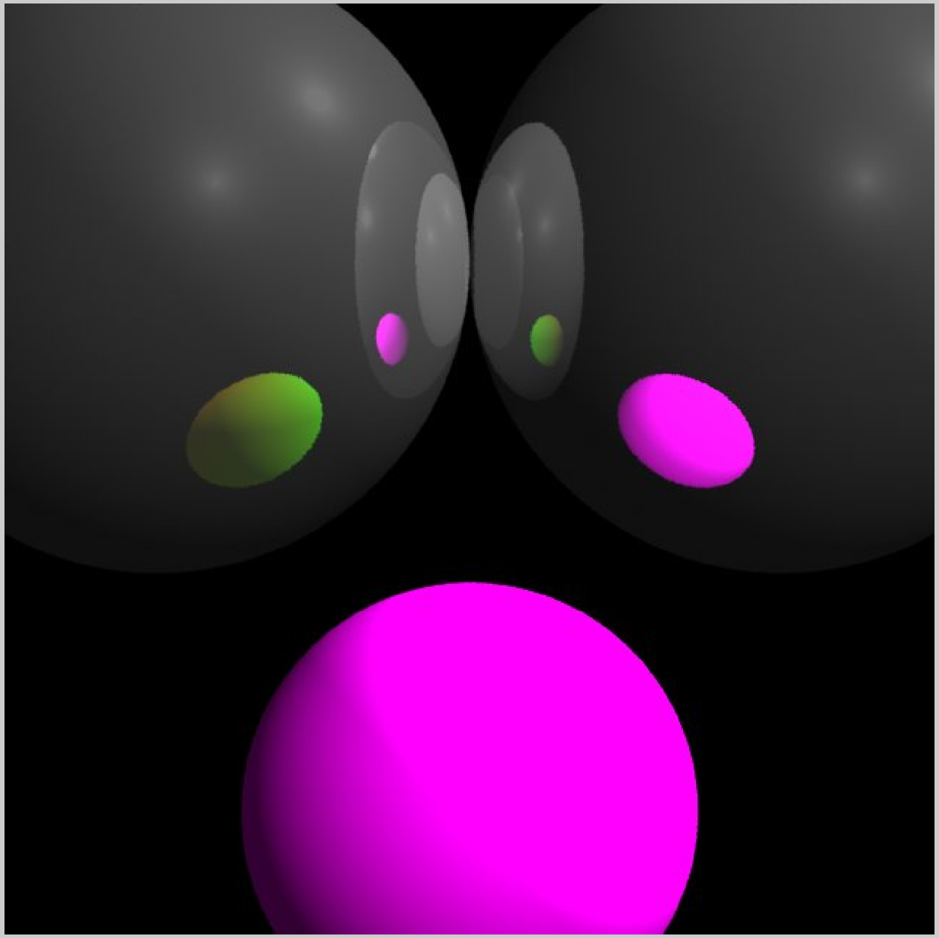
- 2 shiny (RT) spheres
- 1 non RT object
  - Behave differently
    - Depending on which RT object is looking at it



RT control: shader behavior

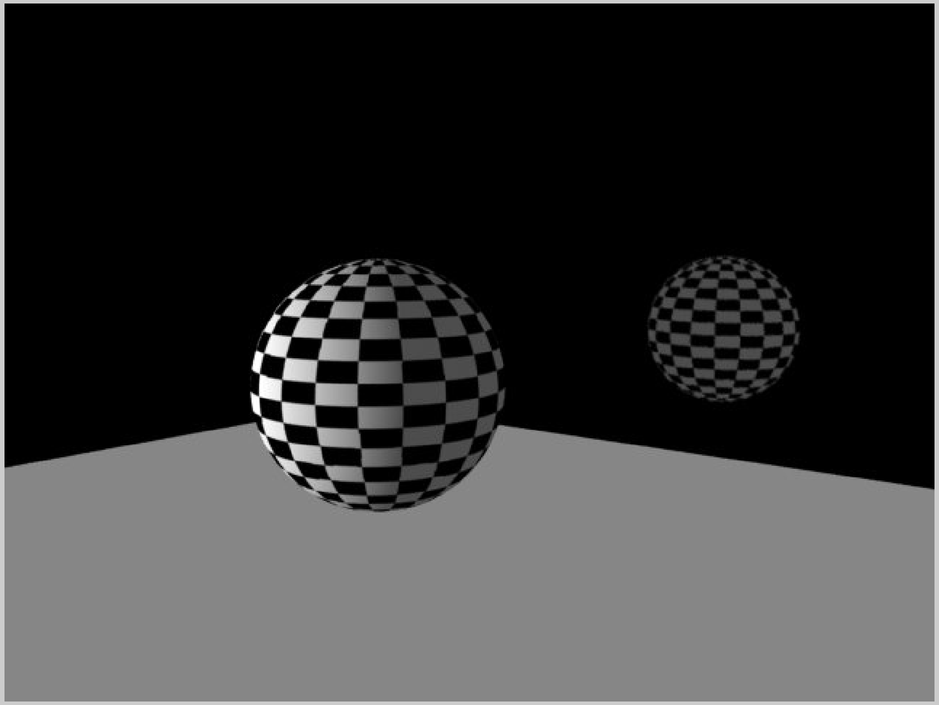
- Magenta shader

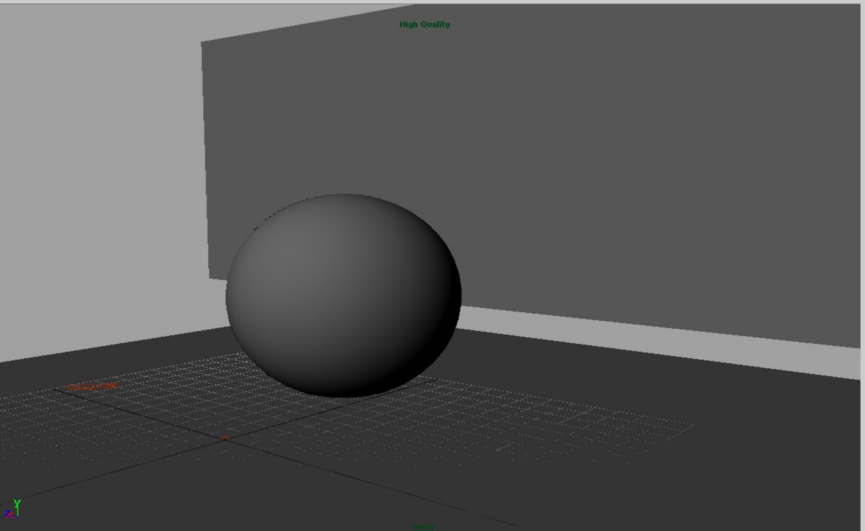
```
rayinfo("label", lbl);
if (lbl == "leftsphere"
   Ci = color(s,t,0);
else
   Ci = Cs;
```



Shader “hitsides”, “back”

- Floor sides 1
- RT Shader only looking at backfacing surfaces.
- Yes, looks like floor invisible to trace rays





END!