Rendering & Compositing

Jeremy Huddleston

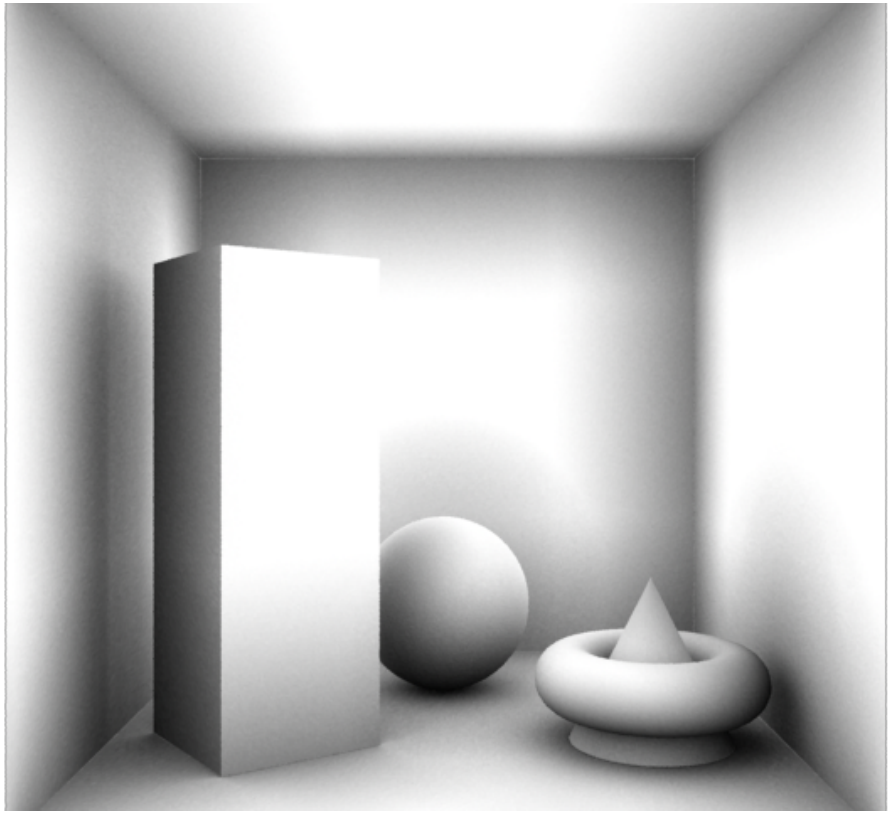
Companion Maya/Renderman/Shake Data:
http://cloud.cs.berkeley.edu/~cnm/lectures/S09

Helpful Resources

- Misc CG Tutorials with a CS-slant
  - http://www.fundza.com
- RMS 3.0 Documentation
  - http://cloud.cs.berkeley.edu/~cnm/pixar/docs/RMS_3
- The RenderMan Shading Language Guide
  - http://amzn.to/ecLmyp

Ambient Occlusion (1/4)

- Percentage of hemisphere around a point which is occluded by other objects.
- Usually rendered out as the percent not occluded for visualization:

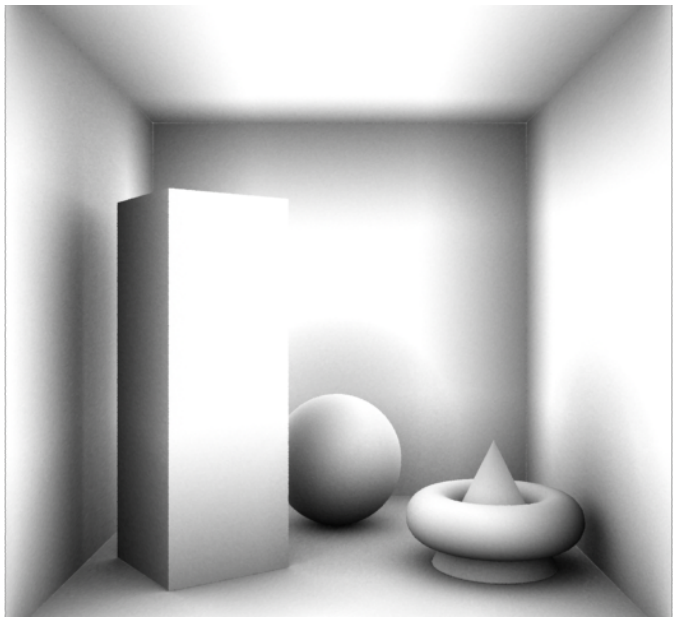


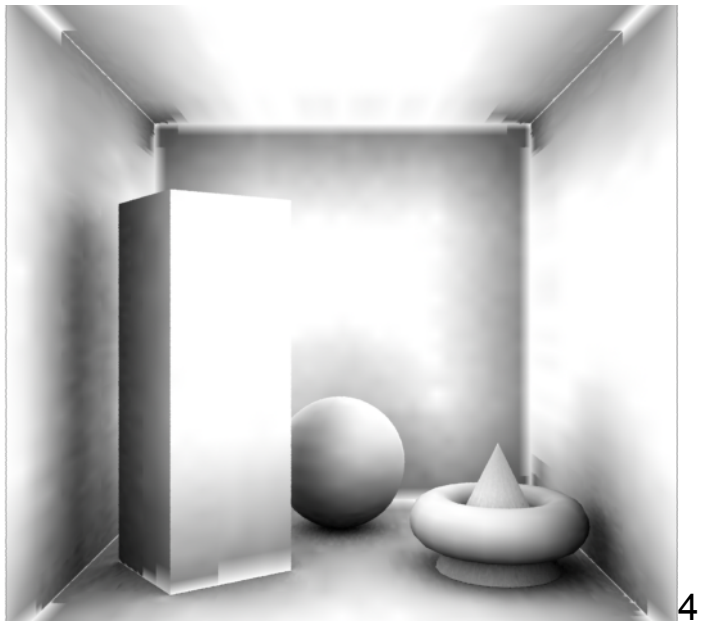
Ambient Occlusion (2/4)

- Enable Ray Tracing!
- ConeAngle:
  - Rays are cast out from the point being shaded through a cone with this characteristic angle (measured from the central axis to the edge).
  - Default = pi / 2 => Hemisphere
- Samples:
  - The number of rays to cast at each point.
- MaxDist:
  - Objects over this far away are considered to not occlude the point.

Ambient Occlusion (3/4)

- Method:
  - Choose between using occlusion() and gather().
  - gather() is computationally more expensive but offers better results.
  - occlusion() uses the knowledge that small changes in P result in small changes in occlusion to produce a good approximation.





Ambient Occlusion (4/4)

```
surface amboccl(uniform float samples = 32;
                uniform float useFastOccl = 0;
                uniform float maxdist = 1e30;
                uniform float coneangle = PI/2) {
   normal Nf = faceforward(normalize(N), I);
   float occ;

   if(useFastOccl != 0) {
      occ = occlusion(P, Nf, samples,
                      "maxdist", maxdist, "coneangle", coneangle,
                      "distribution", "cosine");
   } else {
      float hits = 0;
      gather("illuminance", P, Nf, coneangle, samples, "distribution”,
             "cosine", "maxdist", maxdist) {
         hits += 1;
      }
      occ = hits / samples;
   }
   Ci = 1 - occ;
   Oi = 1;
}
```

MtoR Demo

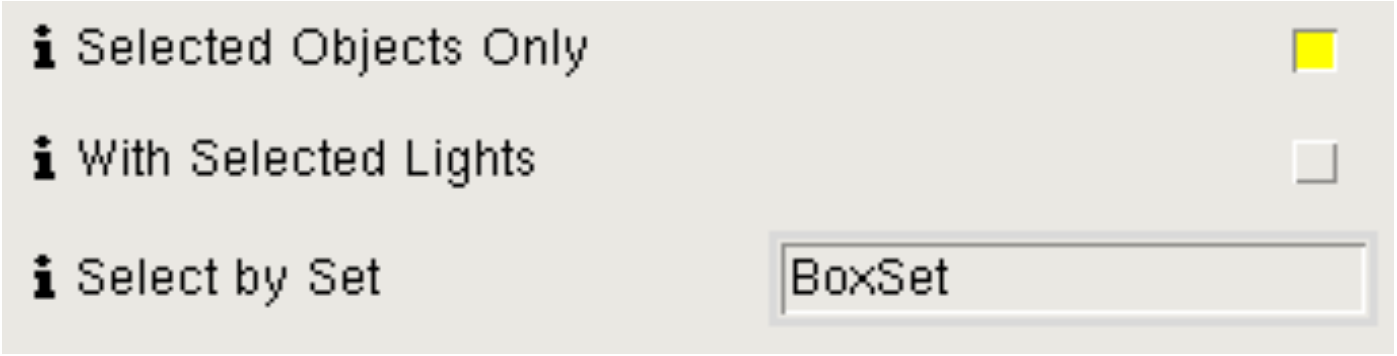
- Shader Compilation
- Shader -> Slim
- Remderman Globals

Scene File Organization

- Use references and versioning
  - Save versioned files as _###.ma suffix AND the latest one also as _latest.ma
  - Always reference _latest.ma
  - Shading/lighting files reference master lighting file
  - Master lighting file references animation file
  - Animation file references geometry file

Object Sets (1/3)

- MtoR can use Maya sets to determine which objects to render
  - Renderman Globals->Accel:



- Create sets early (in your geometry file), so you have consistency in later files

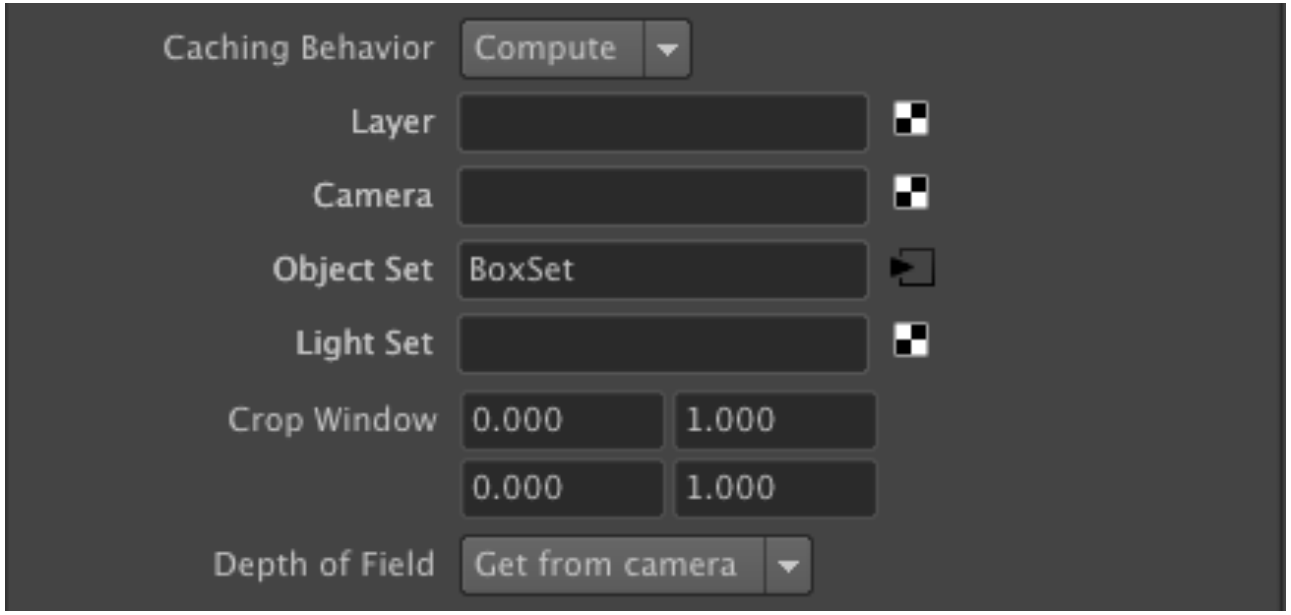
Object Sets (2/3)

- For each set you render, change:
  - Renderman Globals->Display->Display Name
  - Renderman Globals->Accel->Select By Set
- This is easily scriptable in MEL:

```
mtor control getvalue -sync;
string $prefix="cornell_keyfill_";
string $sets[] = {"Cone", "Torus", "Cube", "Box"};

for($set in $sets) {
   string $dspyName = $prefix + $set;
   string $setName = $set + "Set";

   mtor control setvalue -rg dspyName -value $dspyName;
   mtor control setvalue -rg selectedSet -value $setName;
   mtor control renderspool;
}
```

Object Sets (3/3)

- RFM can do the same thing with the settings in RenderMan Controls->Final



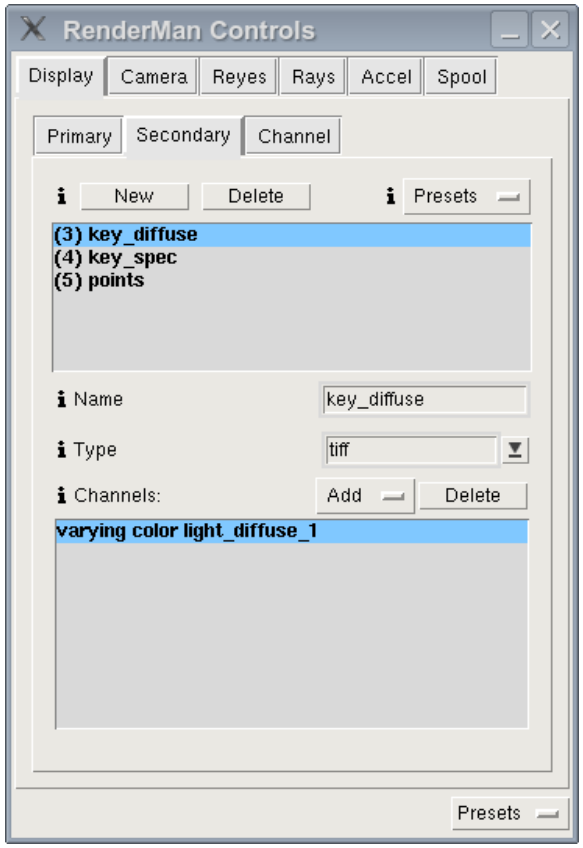
Shaders for Secondary Displays

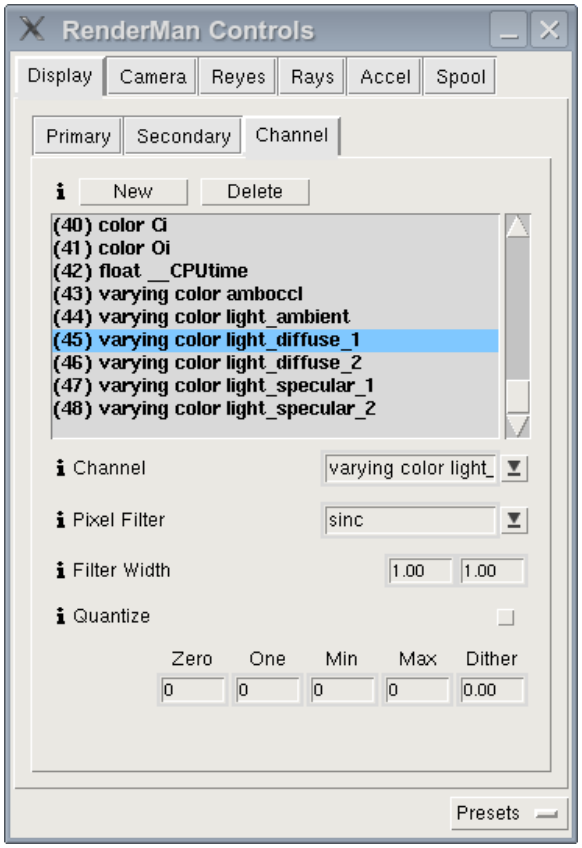
- Declare extra output channels by using the ‘varying’ keyword:

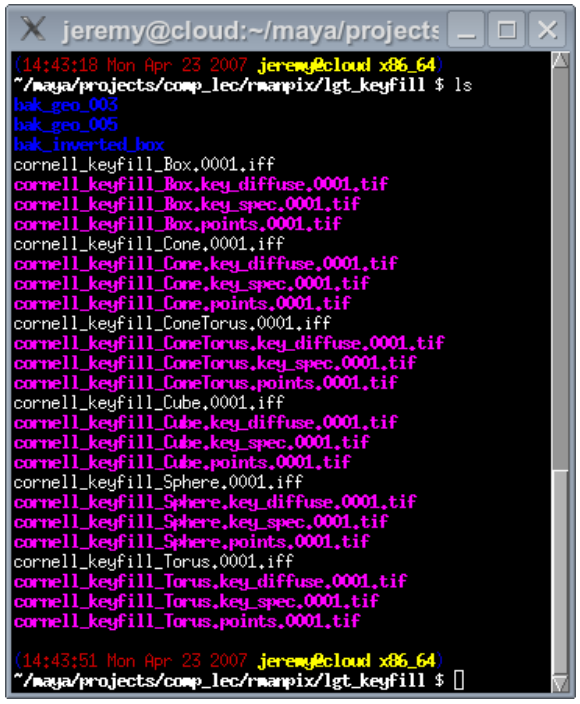
```
surface testsurf(output varying color half=0;) {
    half = Cs * 0.5;
    Ci = Cs;
    Oi = Os;
}
```

MtoR Secondary Displays

- You can select additional channels (like the one defined in the previous slide) to output with the Renderman Globals.



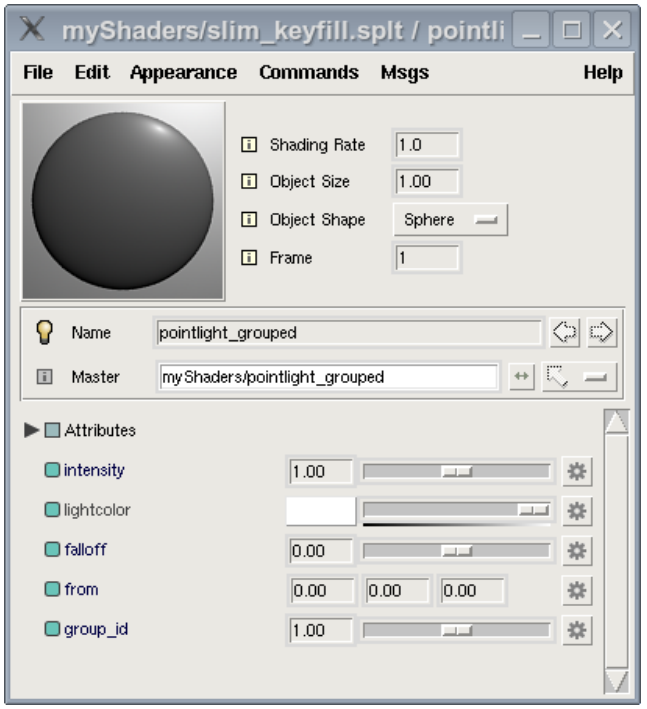




Light Groups (1/2)

- Save light contributions from different light sources to different files in a single pass.
- Write custom light shaders that take an extra parameter to set which group they belong to.

```
light pointlight_grouped(float intensity = 1;
    color lightcolor = 1;
    float falloff= 0;
    point from = point "shader" (0,0,0);
    uniform float group_id = 0;) {

    illuminate (from) {
        Cl = intensity * lightcolor;

        if(falloff == 1) {
            Cl *= 1.0 / length(L);
        } else if (falloff == 2) {
            Cl *= 1.0 / (L . L);
        }
    }
}
```



Light Groups (2/2)

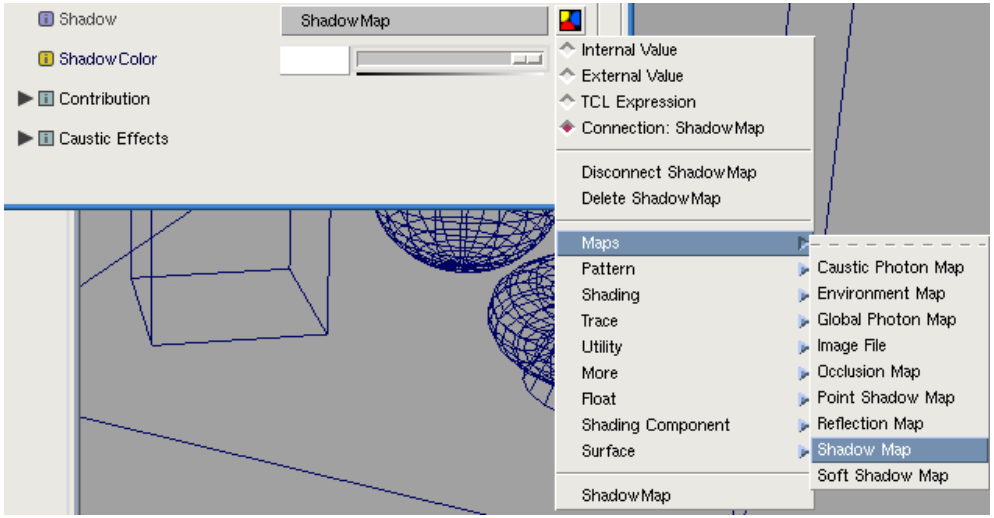
- Write custom surface shaders that take advantage of the light's group_id parameter (this is a simplified version of the swissPhong.sl shader):

```
color diffuse_indexed(
  normal Nn;
  output color lt_ar[4];) {
  color C = 0;
  extern point P;

  illuminance (P, Nn, PI/2) {
    uniform float id=0;
    lightsource ("group_id", id);
    float scale=(normalize(L).Nn);
    color curCol = Cl * scale;
    C += curCol;
    lt_ar[id] += curCol;
  }
  return C;
}
```

```
surface matte_indexed (float Kd = 1;
  float Ka = 1;
  output varying color light_amb = 0;
  output varying color light_0 = 0;
  output varying color light_1 = 0;) {

  color lt_ar[2] = {0, 0};

  normal Nf = faceforward (normalize(N),I);

  light_amb = ambient();

  Ci  = Cs * Ka * light_amb;
  Ci += Cs * Kd * diffuse_indexed(Nf, lt_ar);
  Oi  = Os;  Ci *= Oi;

  light_0 = lt_ar[0];
  light_1 = lt_ar[1];
}
```

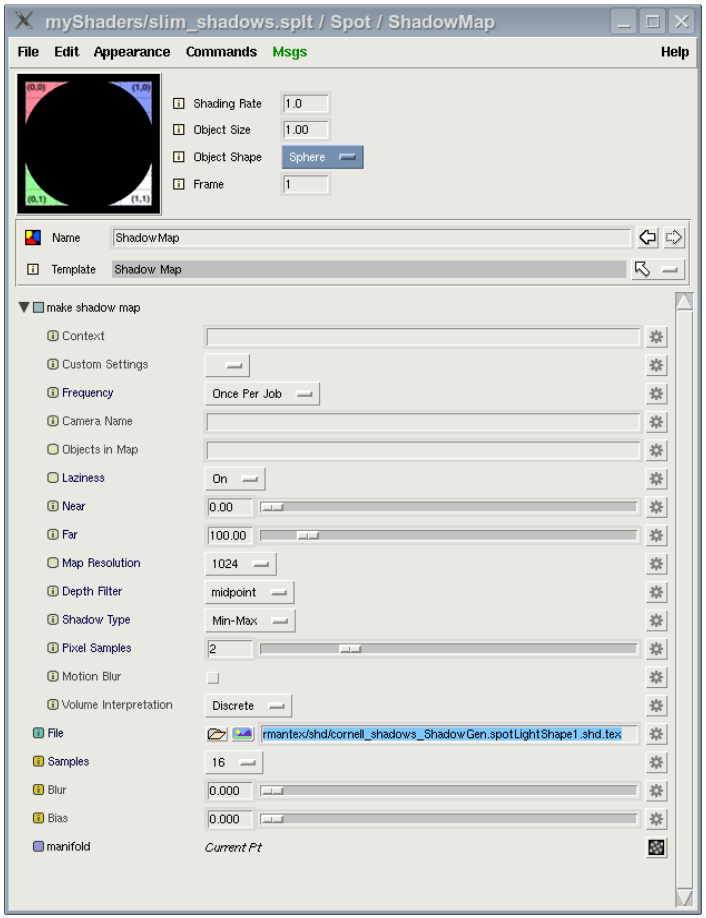
Shadows (1/4)

- Step 1: Replace point lights with spot lights.
  - Point lights require 6 shadow maps!
- Step 2: Create your light shaders.
- Step 3: Apply a shadowmap to your light shader:



Shadows (2/4)

- Step 4: Turn off laziness to force shadowmap to be built. If you choose ‘Use Global’, you can control laziness from Renderman Globals->Accel.
- Step 5: Explicitly set the file parameter, so the same shadow map is used for each object set.
- Step 6: Turn off file cleanup, so the shadow map doesn’t get deleted:
  - RMGlobals->Spool->Job Setup
  - Under Cleanup, make sure ‘map’ is disabled.



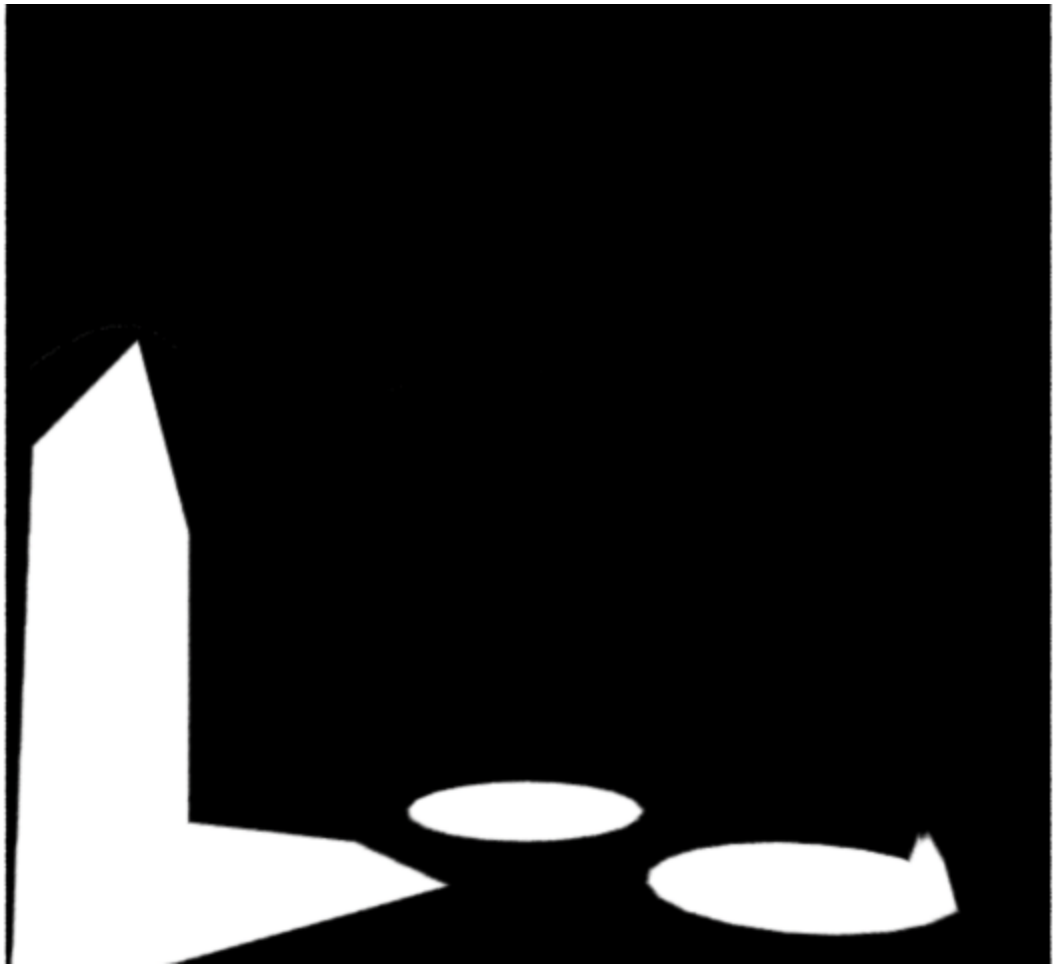
Shadows (3/4)

- Step 7: Apply the shadow shader to all objects:

```
surface shadow_surface() {
    illuminance (P) {
       Ci = Cl;
       Oi = 1;
   }
}
```

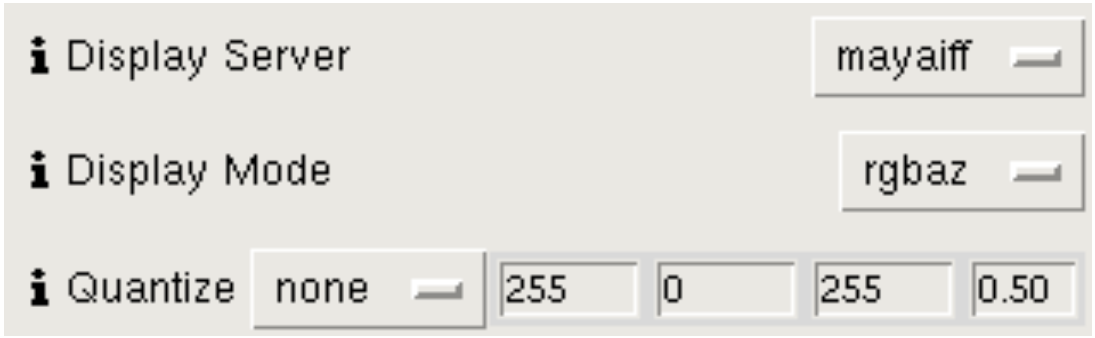
- Step 8: Render to produce your shadow map.
- Step 9: Turn on laziness, so the map won't get recomputed.
- Step 10: Render object sets as normal

Shadows (4/4)



General Render Settings (1/2)

- Do image-space processing in comp
  - Avoid quantize, dither, blur etc.
  - If you want to anti-alias, increase resolution and store the data for your compositor
- For storing the z-buffer, do one of:
  - Use “mayaiff” or “openexr” file type witn “rgbaz”
  - Use “rgba” for one pass and “rgbz” for others
  - Do standalone depth layers (use a varying output)



General Render Settings (2/2)

- Set RM Globals->Reyes->Shading Rate below 1.0 for final render and around 10.0 for testing.
- Short on disk space?
  - Spool->Job Setup->RIB Generation->Deferred
  - Spool->Job Setup->RIB Format->gzipped-ascii

Misc Tips of the Trads

- Keyframe or lock EVERYTHING
- Don’t Autokeyframe
- Always save maya scenes as ascii.
  - It’s MEL
  - Easier to recover corruption or backport
  - Text processors: sed, grep, etc.

Shake Demo

- Dither
- Viewing Z
- Shot Compositing