An Introduction to RenderMan Shaders
for all you GLSLers!

Oregon State University

Mike Bailey

mjb@cs.oregonstate.edu



RIB File
↓
Bounding Box Analysis
↓
Split
↓
Dice into Microfacets
↓
Call the Shaders
↓
Do Front-to-Back Compositing
↓
Assemble the Pixels
↓
Final Image

Why Are We Even Talking About This?

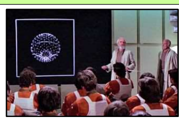
1. You never know – in the future you might be in a position to use RenderMan in the making of movies, TV commercials, etc.
2. You can get RenderMan for free for non-commercial use. It is fun to experiment with.
3. You will be surprised how close what you know now matches what you need to know to run RenderMan shaders. (Congratulations!)

You can get the non-commercial version of RenderMan by starting here:

https://renderman.pixar.com/intro

History of RenderMan, I

1977: Star Wars IV: A New Hope




1979: Ed Catmull, Alvy Ray Smith, and others leave NYIT to form the Computer Division of Lucasfilm

Image Processing

Digital Editing and Compositing

Effects

Image/Volume Rendering Hardware

1984: John Lassiter leaves Disney Animation to join Pixar

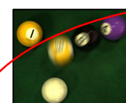
History of RenderMan, II

Image/Volume Rendering Hardware

Pixar Image Computer

Rendering Software


Star Trek II (1982)
Young Sherlock Holmes (1985)



1984 (1984)

REYES

RenderMan

Pixar Animation Studios

RIB

Shade Trees

prman

History of RenderMan, III

Pixar Animation Studios

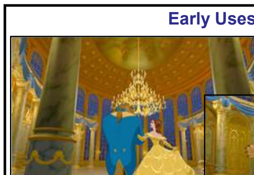
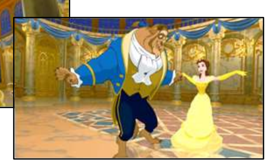
1986: *Luxo Jr.*– Nominated for an Academy Award

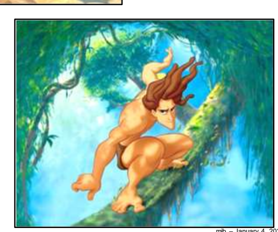
1988: *Tin Toy* – won Academy Award for Best Animated Short

1993: RenderMan wins a Technical Academy Award

1995: *Toy Story*

Early Uses of RenderMan in Movies

7

8

RenderMan Software Rendering Pipeline

9

RIB File
↓
Bounding Box Analysis
↓
Split
↓
Dice into Microfacets
↓
Call the Shaders
↓
Do Front-to-Back Compositing
↓
Assemble the Pixels
↓
Final Image

Oregon State University Computer Graphics

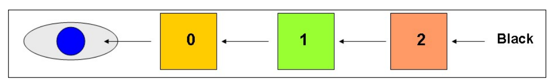
mjb – January 4, 2024

9

RenderMan Composites Starting at the Eye

10

First, let's think about it back-to-front:



$$color_{12} = \alpha_2 color_2 + (1-\alpha_2) black,$$

$$color_{01} = \alpha_1 color_1 + (1-\alpha_1) color_{12},$$

$$color^* = \alpha_0 color_0 + (1-\alpha_0) color_{01}.$$

Substituting gives us the front-to-back equation:

$$color^* = \alpha_0 color_0 + (1-\alpha_0)\alpha_1 color_1 + (1-\alpha_0)(1-\alpha_1)\alpha_2 color_2 + (1-\alpha_0)(1-\alpha_1)(1-\alpha_2) black.$$

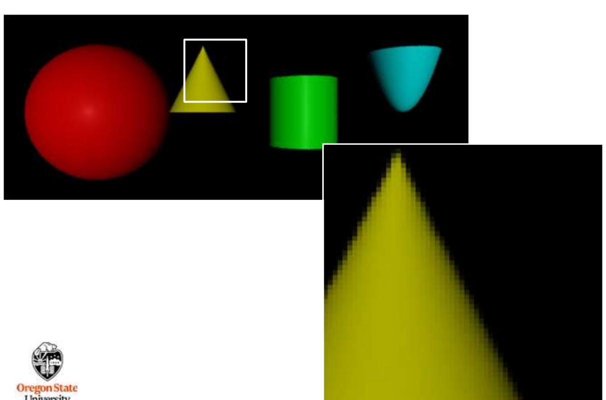
Oregon State University Computer Graphics

mjb – January 4, 2024

10

RenderMan Renders at Higher-than-Screen-Resolution

11



Oregon State University Computer Graphics

mjb – January 4, 2024

11

All Six RenderMan Shader Types

12

1. Displacement → ≈ GLSL Vertex Shader
2. Distortion / transformation → ≈ GLSL Vertex Shader
3. Surface → ≈ GLSL Fragment Shader
4. Lighting → ≈ GLSL Fragment Shader
5. Atmospheric / volumetric → ≈ GLSL Fragment Shader
6. Imaging → ≈ GLSL Fragment Shader

RenderMan ***Built-in Microfaceting*** ≈ Manual or GLSL Tessellation

Oregon State University Computer Graphics

mjb – January 4, 2024

12

Fundamental Differences Between RenderMan Shaders and GLSL Shaders

13

Topic	RenderMan	GLSL
Goals	1. Image quality, 2. Speed	1. Speed, 2. Image quality
Shader Types	Surface, Displacement (+4 others)	Vertex, Fragment, Geometry, Tessellation, Compute
Surface Preprocessing	Microfacets	None [± Tessellation shaders]
Recompute Normals	CalculateNormal	None
Getting Rid of Pixels	Oi = 0.;	discard;
Surface/Fragment shader sets	R, G, B, αr, αg, αb	R, G, B, A [,Z]
Shader Variables	Uniform, Varying	Attribute, Uniform, Out, In
Coordinate Systems	Shader (=Model), World	Model (=OC), Eye (≈WC)
Noise	Built-in	Somewhat built-in or use a Texture
Compile Shaders	Must do yourself	Driver does it for you
Compiler messages	Cryptic	Cryptic

Oregon State University
Computer Graphics

mjb – January 4, 2024

13

RIB Commands, I

14

```
Display          "outputimage.tiff"  "tiff"  "rgb"
Display          "outputimage.shd"  "shadow"  "z"
Imager           "background"  "color"  [r g b]
Clipping         znear  zfar
Format           1280  1024  1.0
PixelSamples     2  2
PixelFilter      "Gaussian"  4  4
Projection       "perspective"  "fov"  60.
ScreenWindow          xleft  xright  ybottom  ytop
Translate        tx  ty  tz
Rotate           deg  rx  ry  rz
Scale            sx sy sz

WorldBegin
WorldEnd

AttributeBegin
AttributeEnd

Color            [ r g b ]
Opacity          [ r g b ]

Surface          "Plastic"  "Ka"  ka  "Kd"  kd  "Ks"  ks  "roughness"  r  "specularcolor"  [r g b]
LightSource      "ambientlight"  num  "intensity"  i  "lightcolor"  [r g b]
LightSource      "distantlight"  num  "intensity"  I  "lightcolor"  [r g b]  "from"  [x y z]  "to"  [x y z]
```

14

RIB Commands, II

15

```
Polygon       "P" [x0 y0 z0  x1 y1 z1 …]  "st"  [s0 t0  s1 t1  …]

Points        "P" [x0 y0 z0  x1 y1 z1 …]  "constantwidth"  w
Points        "P" [x0 y0 z0  x1 y1 z1 …]  "width"  [w0  w1  …]

Sphere        radius  zmin  zmax  sweepAngle
Cylinder      radius  zmin  zmax  sweepAngle
Cone          height  radius  sweepDegrees
Torus         majorRadius  minorRadius  startAngle  endAngle  sweepAngle
Hyperboloid   x0 y0 z0  x1 y1 z1  sweep_Angle
Paraboloid    zmaxRadius  zmin  zmax  sweepAngle
Disk          xHeight  radius  sweepAngle

Basis         "Bezier"  3
Curves        "cubic"  [4]  "nonperiodic"  "P"  [x0 y0 z0  x1 y1 z1  …] "constantwidth"  [w]

TransformEnd
TransformBegin
```

Oregon State University
Computer Graphics

mjb – January 4, 2024

15

Cartesian (X) Stripes

16

stripes.rib

```
##RenderMan RIB
version 3.03
Declare "Prob"     "uniform float"

Display "stripes.tiff" "file" "rgb"
Format   512   512   -1
ShadingRate 1

LightSource  "ambientlight"  1  "intensity"  [0.25]
LightSource  "distantlight"    2  "intensity"  [0.75] "from"  [0  0  -10] "to"  [0  0  0]

Projection "perspective"  "fov"  [70]
WorldBegin
  Translate 0 0 8
  Surface "stripes"  "Prob" 0.8
  Color   [1  1  1]
  Opacity [1  1  1]
  Sphere 3  -3  3  360
  Translate 2.  0.  0.
WorldEnd
```

Tells RenderMan to start using a surface shader file called **stripes.slo**

Says that we want to set the **Prob** argument in that shader to **0.8**

Oregon State University
Computer Graphics

mjb – January 4, 2024

16

Cartesian (X) Stripes

stripes.sl

17

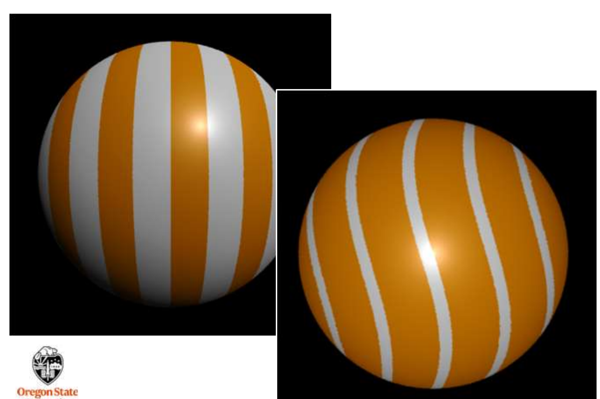
```
surface
stripes( float
            Prob = 0.4,  // probability of seeing orange
            Ks = 0.5,
            Kd = 0.5,
            Ka = .1,
            roughness = 0.1;
      color   specularColor = color( 1, 1, 1 )
)
{
            color stripeColor;

            float x = xcomp( P );
            //x = x + 0.30 * sin( 1. * ycomp(P) );                          // adding a wobble
            float xfrac = mod( x, 1. );

            if( xfrac < Prob )
                        stripeColor = color( 1., .5, 0. );        // beaver orange?
            else
                        stripeColor = Cs;

            varying vector Nf = faceforward( normalize( N ), I );
            vector V = normalize( -I );

            Oi = 1.;
            Ci = Oi * ( stripeColor * ( Ka * ambient( ) + Kd * diffuse(Nf) ) +
                                    specularColor * Ks * specular( Nf, V, roughness ) );
}
```

mjb – January 4, 2024

17

Cartesian (X) Stripes

18



Oregon State University
Computer Graphics

mjb – January 4, 2024

18

Rings

rings.rib

```
##RenderMan RIB
version 3.03
Declare "Prob"     "uniform float"

Display  "rings.tiff"  "file"  "rgb"
Format  500   500    -1
ShadingRate 1

LightSource  "ambientlight"  1  "intensity"   [0.25]
LightSource  "distantlight"     2  "intensity"  [0.75] "from"  [5  0  -10] "to"  [0  0  0]

Projection  "perspective"  "fov"  [70]
WorldBegin
  Translate 0 0 8
  Surface  "rings"    "Prob"  0.6
  Color     [1  1  1]
  Opacity [1  1  1]
  Sphere 3  -3  3  360
  Translate 2.  0.  0.
WorldEnd
```

Oregon State University Computer Graphics

mjb – January 4, 2024

19

Rings

rings.sl

```
surface
rings( float
              Prob = 0.4,
              Ks = 0.5,
              Kd = 0.5,
              Ka = .1,
              roughness = 0.1;
       color  specularcolor = color( 1, 1, 1 )
)
{
              varying vector Nf = faceforward( normalize( N ), I );
              vector V = normalize( -I );

              float x = xcomp( P );
              float y = ycomp( P );
              float r = sqrt( x*x + y*y );
              float rfrac = mod( r, 1. );

              if( rfrac < Prob )
                            Ci = color( 1., .5, 0. );
              else
                            Ci = Cs;

              Oi = 1.;
              Ci = Oi * ( Ci * ( Ka * ambient( ) + Kd * diffuse(Nf) ) +
                            specularcolor * Ks * specular( Nf, V, roughness )  );
}
```

mjb – January 4, 2024

20

Rings (= Polar Stripes)

Oregon State University Computer Graphics

mjb – January 4, 2024

21

Dots

dots.rib

```
##RenderMan RIB
version 3.03
Declare "Diam"     "uniform float"

Display "dots.tiff" "file" "rgb"
Format 512 512 -1
ShadingRate 1

LightSource  "ambientlight"  1  "intensity"  [0.25]
LightSource  "distantlight"     2  "intensity" [0.75]  "from"  [5  8  -10]  "to"  [0  0  0]

Projection  "perspective"  "fov"  [70]
WorldBegin
  Translate  0  0  6
  Surface  "dots"    "Diam"  0.10
  Color     [1 1 1]
  Opacity [1 1 1]
  TransformBegin
     Rotate 90  1. 0. 0.
     Sphere 3 -3 3 360
  TransformEnd
WorldEnd
```

University Computer Graphics

mjb – January 4, 2024

22

Dots

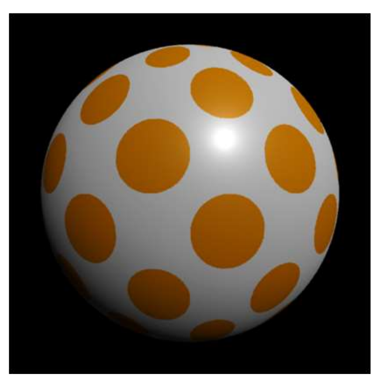
dots.sl

```
surface
dots( float
              Diam = 0.10,  // dot diameter
              Ks = 0.5,
              Kd = 0.5,
              Ka = .1,
              roughness = 0.1;
       color  specularColor = color( 1, 1, 1 )
)
{
              float up = 2. * u;
              float vp = v;
              float numinu = floor( up / Diam );
              float numinv = floor( vp / Diam );

              color dotColor = Cs;
              if( mod( numinu+numinv, 2 ) == 0 )
              {
                            float uc = numinu*Diam + Diam/2.;
                            float vc = numinv*Diam + Diam/2.;
                            up = up - uc;
                            vp = vp - vc;
                            point upvp =  point( up, vp, 0. );
                            point cntr   =  point( 0., 0., 0. );
                            if( distance( upvp, cntr ) < Diam/2. )
                            {
                                          dotColor = color( 1., .5, 0. );      // beaver orange?
                            }
              }

              varying vector Nf = faceforward( normalize( N ), I );
              vector V = normalize( -I );

              Oi = 1.;
              Ci = Oi * ( dotColor * ( Ka * ambient( ) + Kd * diffuse(Nf) ) +
                                          specularColor * Ks * specular( Nf, V, roughness )  );
}
```

mjb – January 4, 2024

23

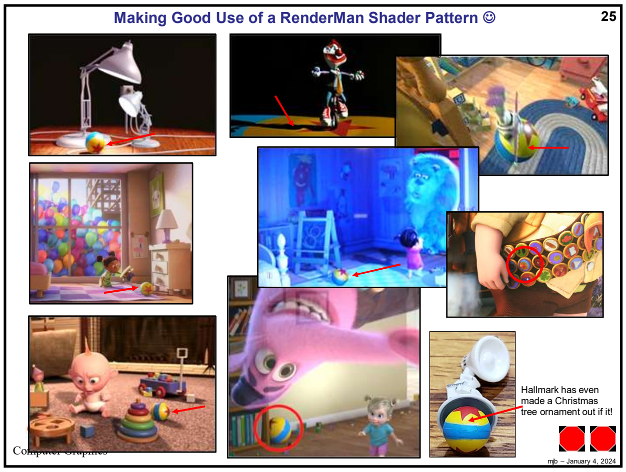
Dots



Oregon State University Computer Graphics

mjb – January 4, 2024

24


Making Good Use of a RenderMan Shader Pattern ☺
25
Hallmark has even made a Christmas tree ornament out if it!
Computer Graphics
mjb – January 4, 2024

25