Multi-Pass RenderMan

Marc Olano
SGI

What is RenderMan?

Interface for renderers

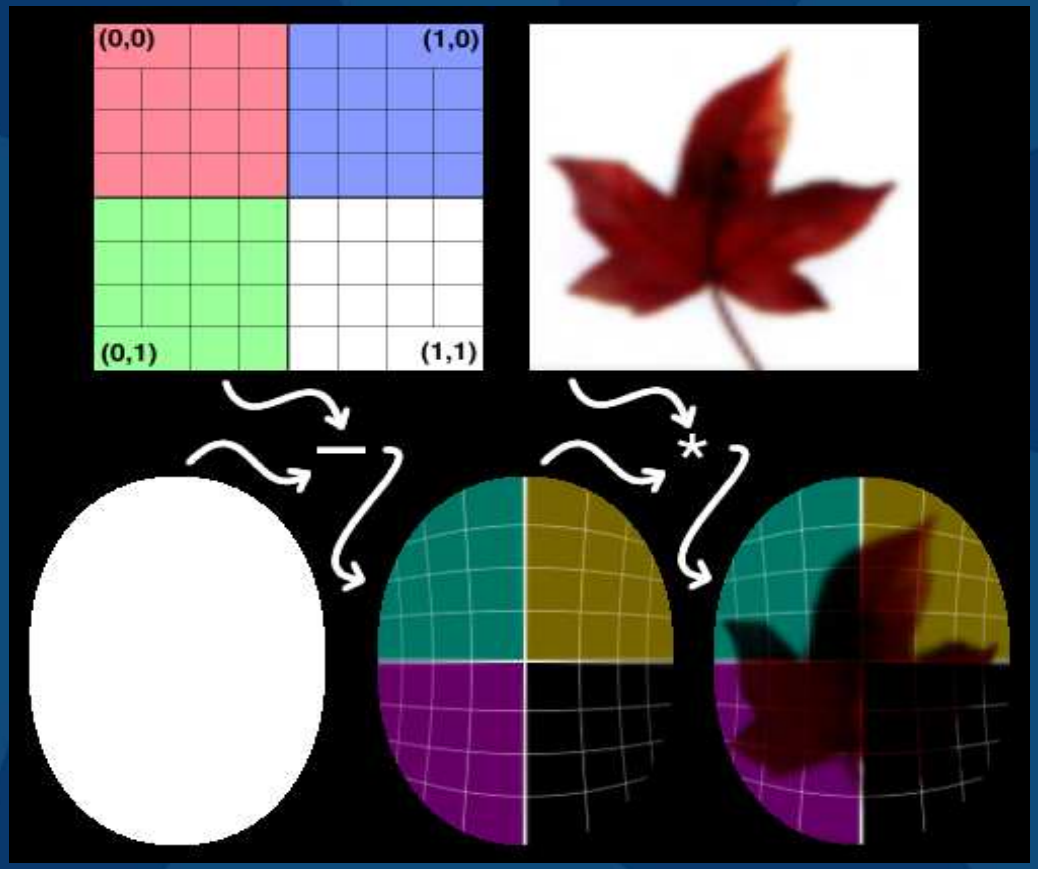
- Scene description & Shading language
- Created by Pixar
- Used by several software renderers

Why do we care?

- Powerful C-like shading language
- Widely used

Simple RenderMan Shader

```
surface simple() {
  Ci = (1 - texture("grid.tx")) * texture("leaf.tx");
}
```



Slightly Longer Shader

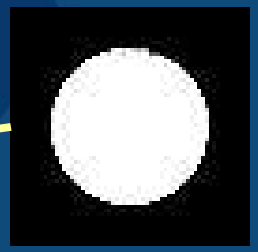

```
surface
beachball(
     uniform float Ka = 1, Kd = 1;
     uniform float Ks = .5, roughness = .1;
     uniform color starcolor = color (1,.5,0);
     uniform color bandcolor = color (1,.2,.2);
     uniform float rmin = .15, rmax = .4;
     uniform float npoints = 5;
      )
{
     color Ct;
     float angle, r, a, in_out;
     vector d1;
    ...
}
```

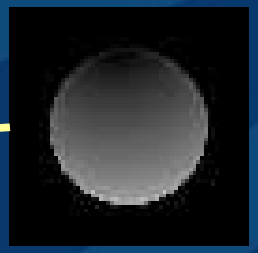
Slightly Longer Shader

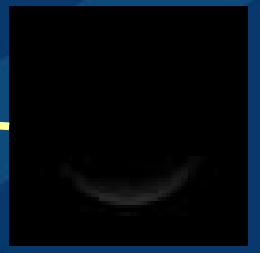

```
uniform float starangle = 2*PI/npoints;
uniform point p0 = rmax*point(cos(0),sin(0),0);
uniform point p1 = rmin*
    point(cos(starangle/2),sin(starangle/2),0);
uniform vector d0 = p1 - p0;

angle = 2*PI * s;
r = .5-abs(t-.5);
a = mod(angle, starangle)/starangle;

if (a >= 0.5)
     a = 1 - a;
d1 = r*(cos(a), sin(a),0) - p0;
in_out = step(0, zcomp(d0^d1));
Ct = mix(mix(Cs, starcolor, in_out), bandcolor, step(rmax,r));

normal Nf = normalize(faceforward(N,I));
Oi = Os;
Ci = Os * (Ct * (Ka * ambient() + Kd * diffuse(Nf)) +
           Ks * specular(Nf,-normalize(I),roughness));
```

Beachball passes

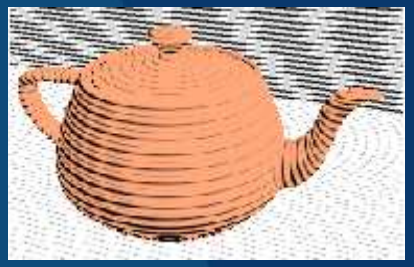
```
angle = 2*PI * s

angle = 2*PI * s

r = .5-abs(t-.5)

r = .5-abs(t-.5)

r = .5-abs(t-.5)

r = .5-abs(t-.5)
```







prman vs. Multi-pass



Doing Better

Optimizations we did

- Fold constants
- Reuse textures
- Avoid redundant copies
- Remove dead code
- Use hardware features

An Optimization Tool

iburg based tree matching tool [Fraser92]

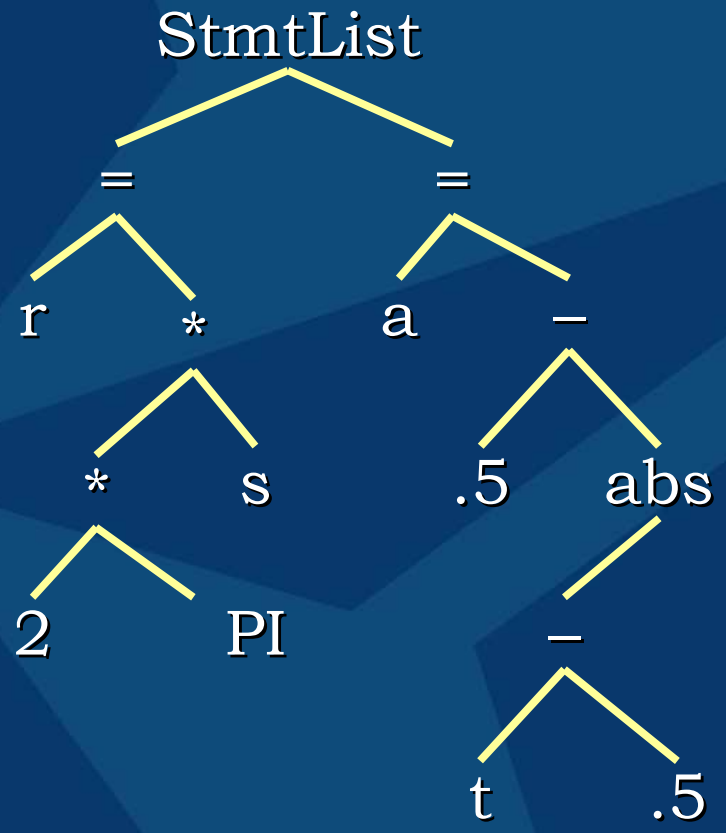
- Set of rules and costs
- Cover tree with least cost

Our version runs C++ code

- To find rule cost
- Before processing children
- After processing children

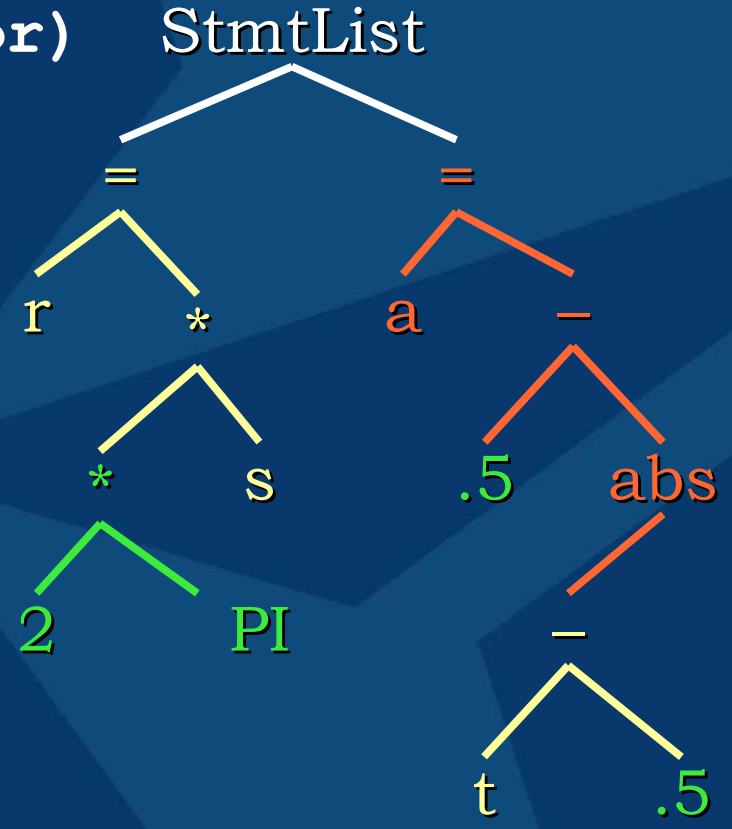
Simple Parse Tree

```
angle = 2*PI * s;
r = .5-abs(t-.5);
```



Simple Parse Tree

```
stmtlist: StmtList(expr,expr)
expr: linearST
    | const
    | Sub(expr,expr)
    | Abs(expr)
    | Assign(Var,expr)
linearST : s | t
    | const
    | Mul(const,linearST)
    | Sub(linearST,linearST)
    | Assign(Var,linearST)
const: ConstFloat
    | Mul(const,const)
```



Mapping Options

By pass

- No restrictions on order
- Hard to map to operations

By simple operation

- Complex order restrictions
- Simple to map to operations

More complex matching framework

- See Chan, et al., Graphics Hardware 2002

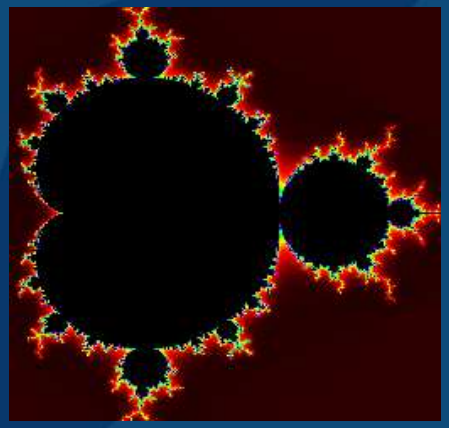
OpenGL Requirements

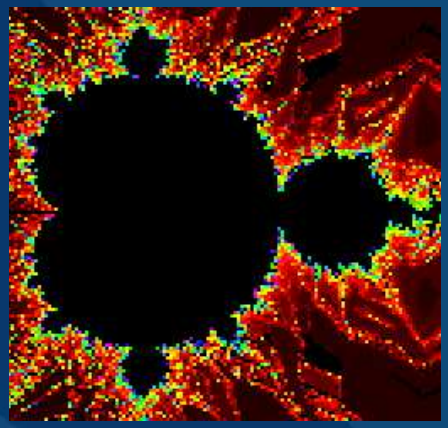
What do we need?

- Extended range and precision
- Pixel texture / Dependent texture
- Color swizzle (e.g. color matrix)
- Feedback for ending loops (e.g. min/max)

Extended Range and Precision


```
surface mandelbrot(float maxIter=64) {
  varying float zs = 0, zt = 0, ss = 0, tt = 0;
  varying float iter;
  for(iter=0; iter < maxIter && ss + tt < 4); iter += 1){
    ss = zs*zs;
    tt = zt*zt;
    zt = 2.0*zs*zt + t;
    zs = ss - tt + s;
  }
  Ci = color spline(iter/maxIter, /*...*/);
}
```





Real-Time RenderMan?

Can we get there?

- YES (for some shaders)

Real-Time “Toy Story”?

- No (at least not yet)
- BIG shaders, BIG scenes

Do we want it? Maybe, maybe not

- Real-time targeted shaders
- Real-time targeted languages
- Learn and adapt!

SAN ANTONIO

SIGGRAPH 2002