Lecture 10: Shading Languages

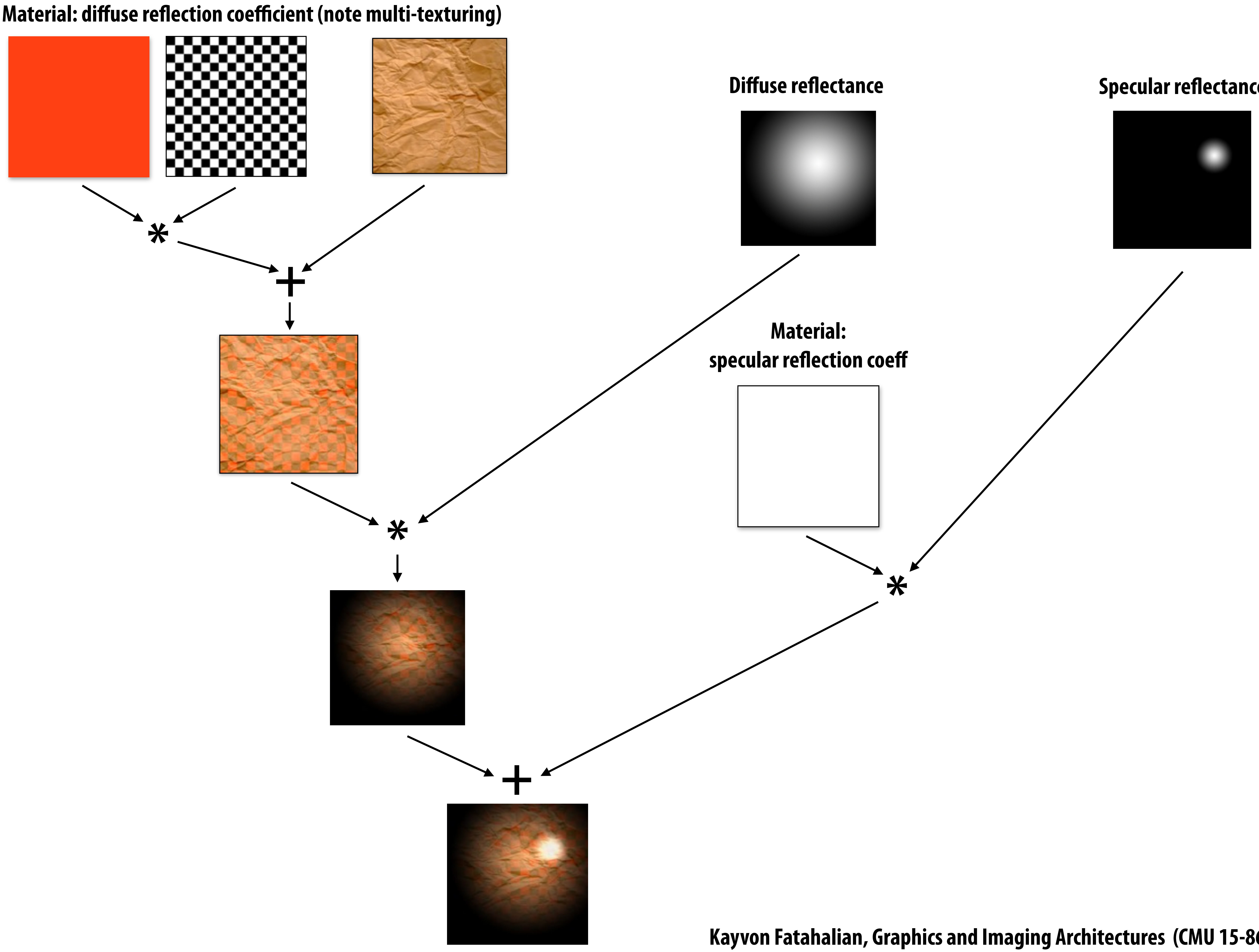
Kayvon Fatahalian
CMU 15-869: Graphics and Imaging Architectures (Fall 2011)

Review: role of shading languages

- Renderer handles surface visibility tasks
  - Examples: clip, cull, rasterizer, z-buffering
  - Highly optimized implementations on canonical data structures (triangles, fragments, and pixels)

- Impractical for rendering system to constrain application to use a single parametric model for surface definitions, lighting, and shading
  - Applications define these behaviors procedurally
  - Shading language is the interface between application-defined surface, lighting, material reflectance functions and the renderer

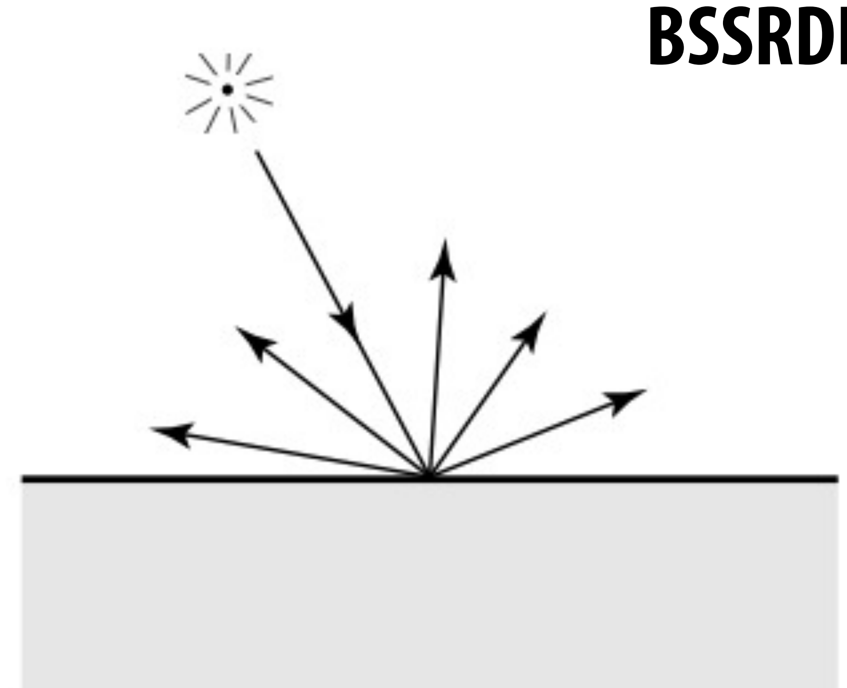
Some history: shade trees [Cook 84]

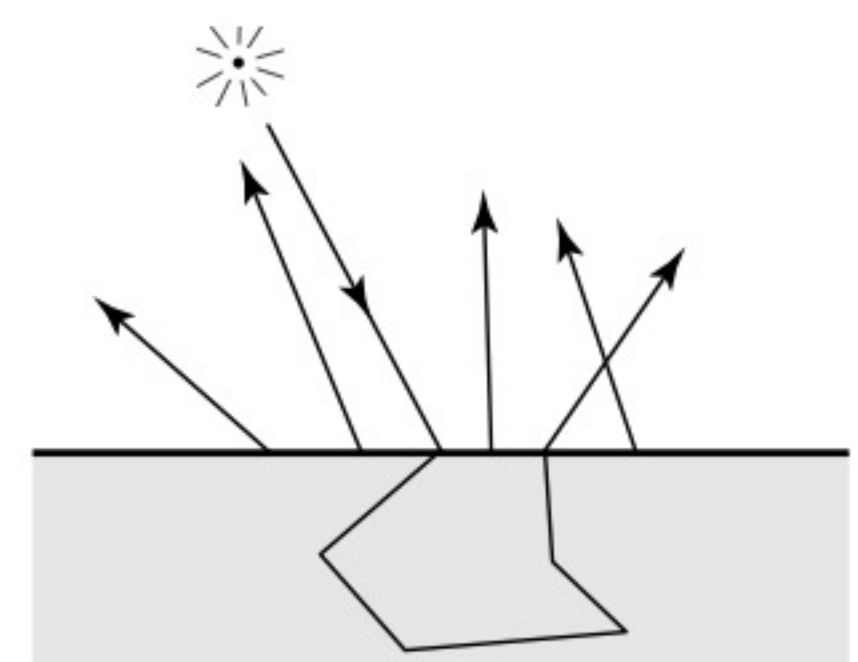


Aside: more advanced light/surface interaction
[Wann Jensen et al. 2001]

BRDF

BSSRDF





- Account for internal scattering
- Light exists surface from different location of location of incidence
  - Very important to matter translucent materials like skin, foliage, marble

Renderman shading language [Hanrahan and Lawson 90]

- **High-level, domain-specific language**
  - Domain: describing propagation of light through scene

What are the key RSL abstractions?

- Shaders
  - Surfaces
  - Lights
  - A few more types (but will not address them today)

- Light shader `illuminate` construct

- Surface shader `illuminance` loop (integrate light)

- Texturing primitives

Renderman shading language [Hanrahan and Lawson 90]

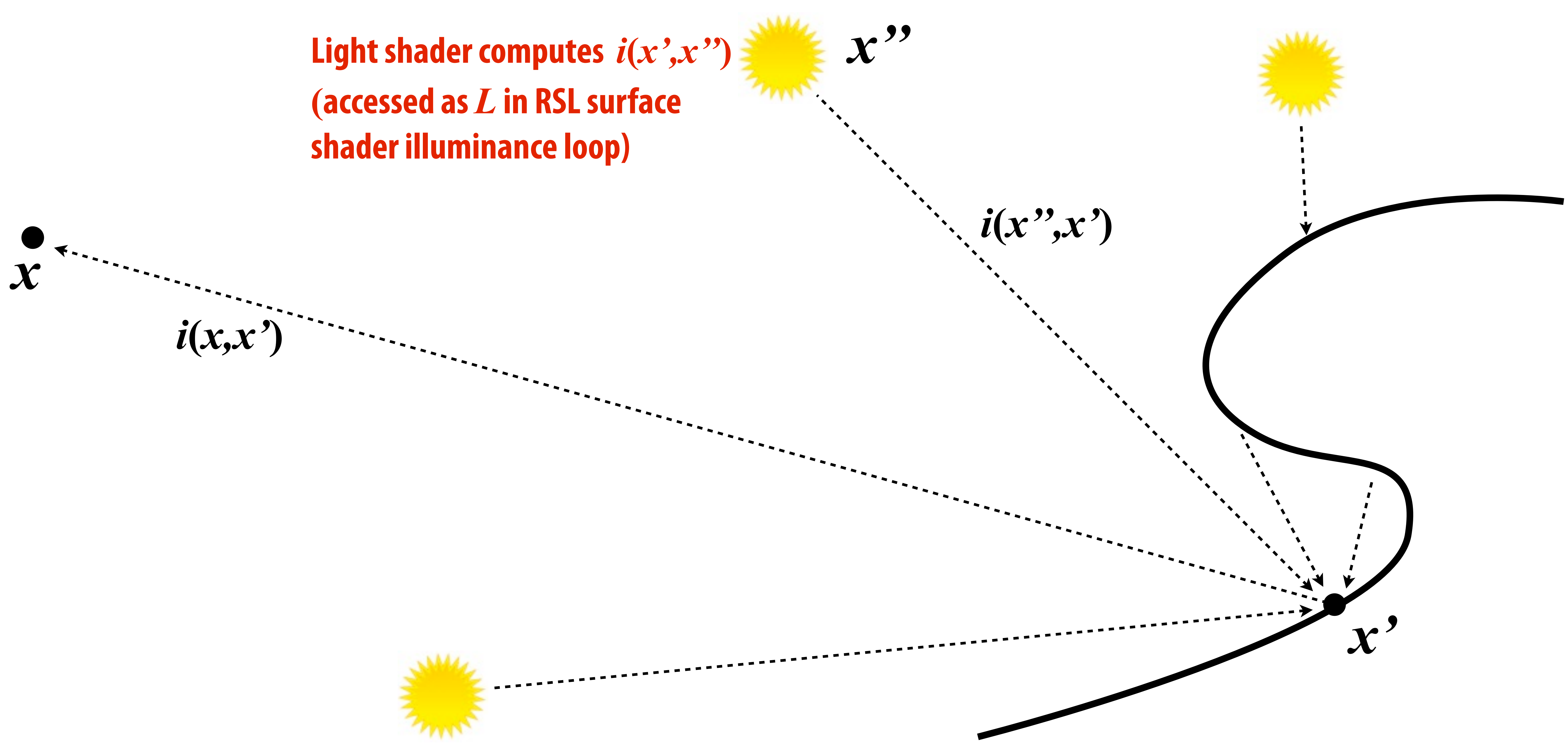
- Separate surface shaders from light source shaders
  - Light source shaders describe distribution of energy from a light
  - Surface shaders
    - Define surface reflectance distribution function (BRDF)
    - Integrate light from light sources

Recall: rendering equation

[Kajiya 86]

$$i(x,x') = v(x,x')\boxed{\left[l(x,x') + \int r(x,x',x'')\,i(x',x'')\,dx''\right]}$$

Surface shader



Shading objects in RSL

compiled code
(plastic material)

current transforms

bound parameters
kd = 0.5
ks = 0.3

Surface shader object

compiled code
(spotlight)

current transforms

bound parameters
intensity =0.75
color = (1.0, 1.0, 0.5)
position = (5,5,10)
axis = (1.0, 1.0, 0)
angle = 35

compiled code
(point light)

current transforms

bound parameters
position = (5,5,5)
intensity =0.75
color = (1.0, 1.0, 0.5)

compiled code
(point light)

current transforms

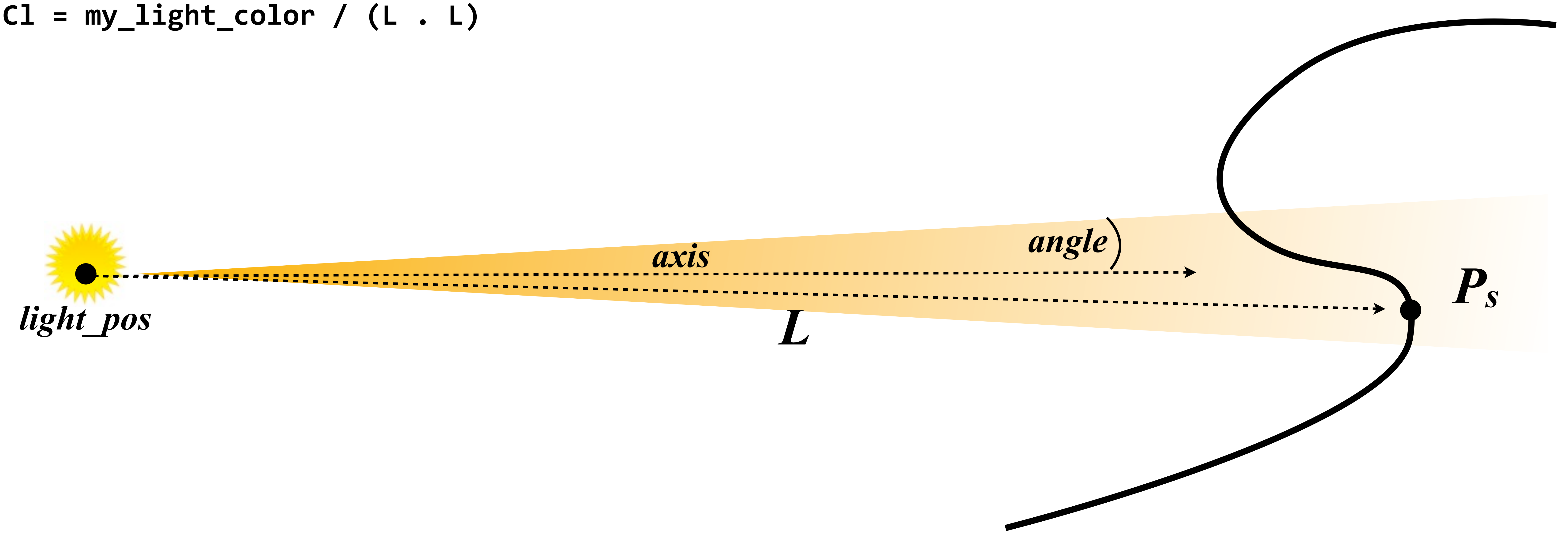
bound parameters
position = (20,20,100)
intensity =0.5
color = (0.0, 0.0, 1.0)

Light shader objects
(bound to surface)

Light shaders

Example: Attenuating spot-light (no area fall off)

```
illuminate (light_pos, axis, angle)
{
  Cl = my_light_color / (L . L)
}
```



Surface shaders

```
illuminance (position, axis, angle)
{

}
```

Example: Computing diffuse reflectance

```
surface diffuseMaterial(color Kd)
{
  Ci = 0;

  // integrate light over hemisphere
  illuminance (P, Nn, PI/2)
  {
     Ci += Kd * Cl * (Nn . normalize(L));
  }
}
```

`Cl` = Value computed by light shader

`L` = Vector from light position (recall `light_pos` argument to light shader's `illuminate`) to surface position being shaded (see `P` argument to `illuminance`)

Surface shader computes `Ci`

RSL design retrospective

(switching to notes by Pat Hanrahan)

Cg

(Class discussion)