RAYTRACER PART II

Lighting & Shadows

Materials

REAL LIGHTING IS EXPENSIVE!

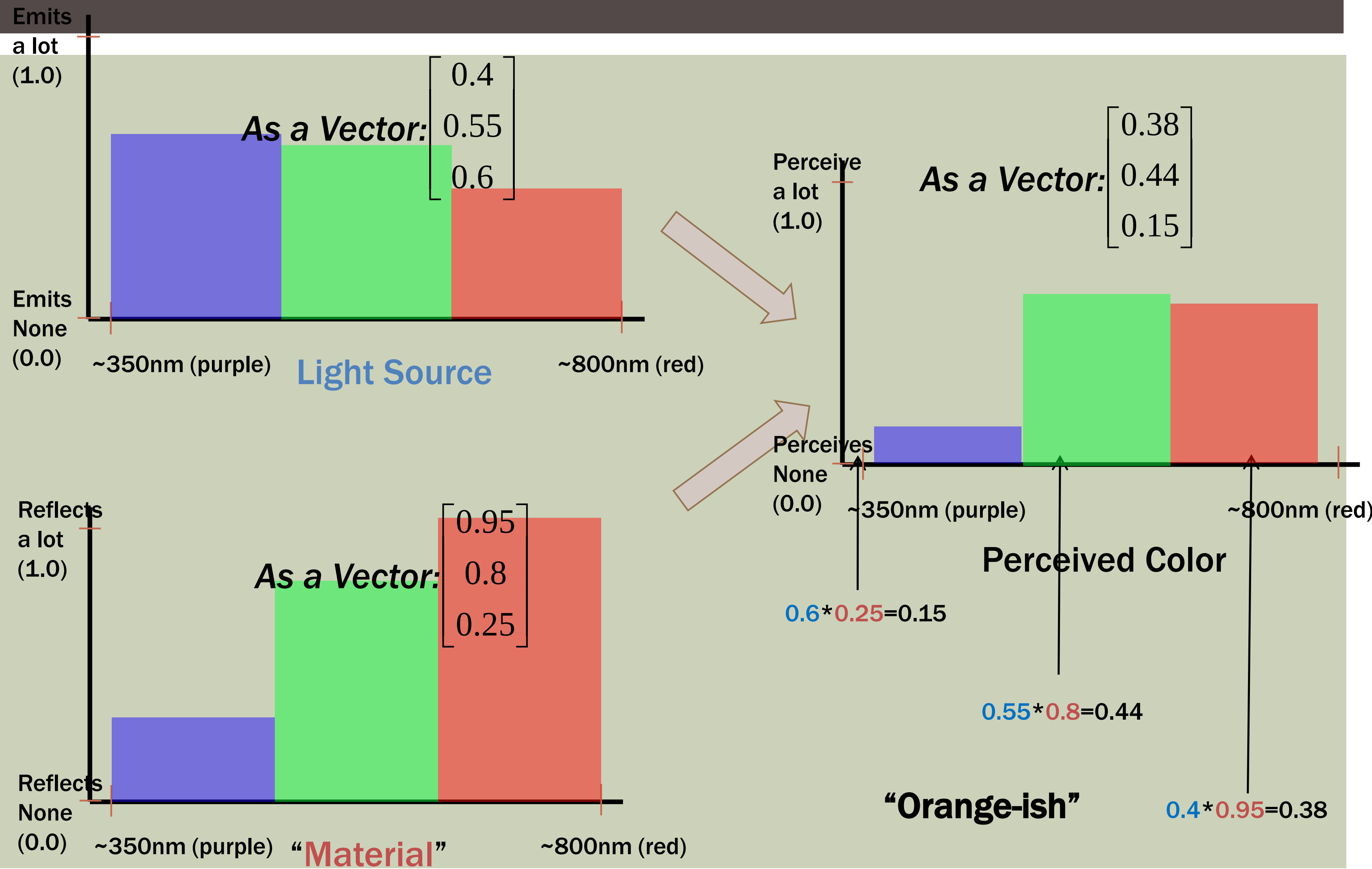
- Factors:
  - Lots of photons
  - Complex interaction with surfaces
  - Lots of bounces
- Solution:
  - <u>approximate</u> real-life lighting.
  - Not at all the way real-life lighting works.

REAL LIGHTING IS EXPENSIVE!

The most common approximations (in CG):

- Limit color values: [0…255 or 0…1] [discuss]
- **Diffuse:** (or **Lambertian** or "matte illumination")
- **Specular:** or **Blinn** or **Phong** or "hotspot" illumination
- **Ambient** or indirect illumination

VIRTUAL LIGHTING (REFLECTANCE, ABSORBANCE, AND PERCEPTION)



A NEW VECTORN FUNCTION: PAIR-WISE PRODUCT

- The symbol your book uses is $\otimes$
- Only useful for colors represented as Vector3's
  - x = red
  - y = green
  - z = blue
- Numeric interpretation (the only interpretation we'll use):

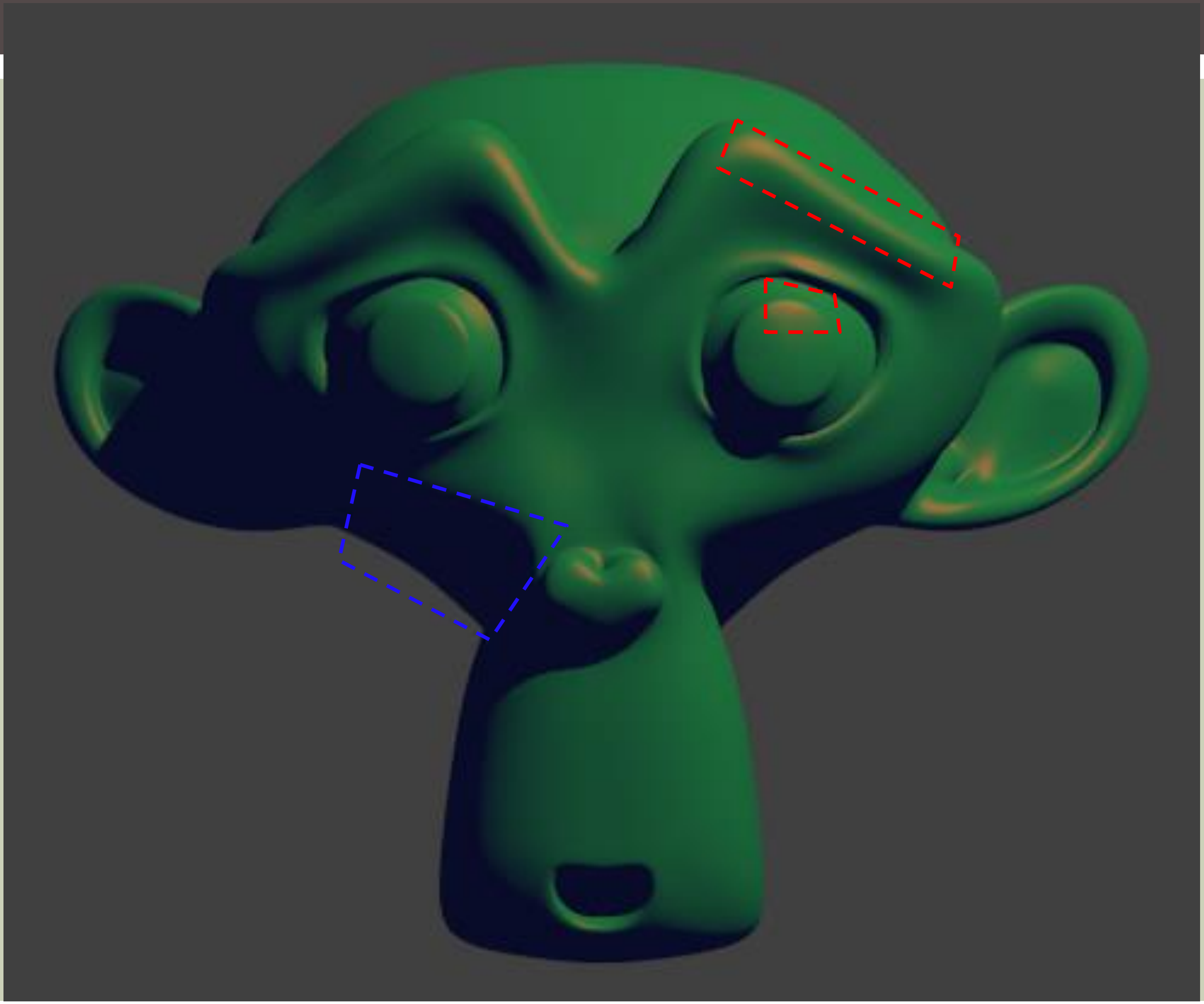
$$\vec{v} \otimes \vec{w} = \begin{bmatrix} \vec{v}_x \\ \vec{v}_y \\ \vec{v}_z \end{bmatrix} \otimes \begin{bmatrix} \vec{w}_x \\ \vec{w}_y \\ \vec{w}_z \end{bmatrix} = \begin{bmatrix} \vec{v}_x * \vec{w}_x \\ \vec{v}_y * \vec{w}_y \\ \vec{v}_z * \vec{w}_z \end{bmatrix}$$

Specular

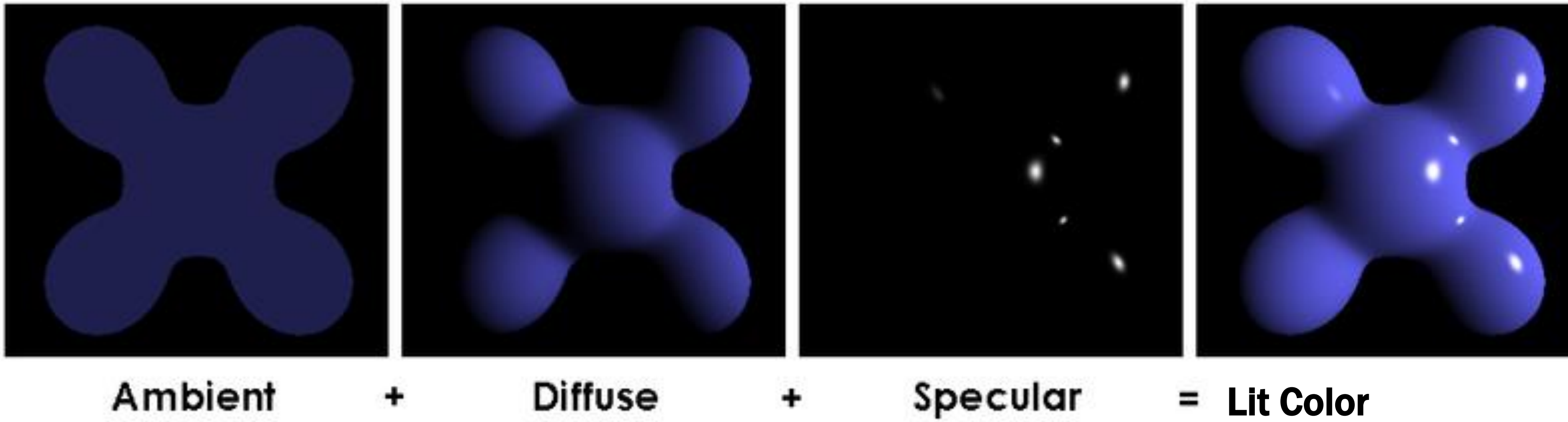
Ambient

Diffuse

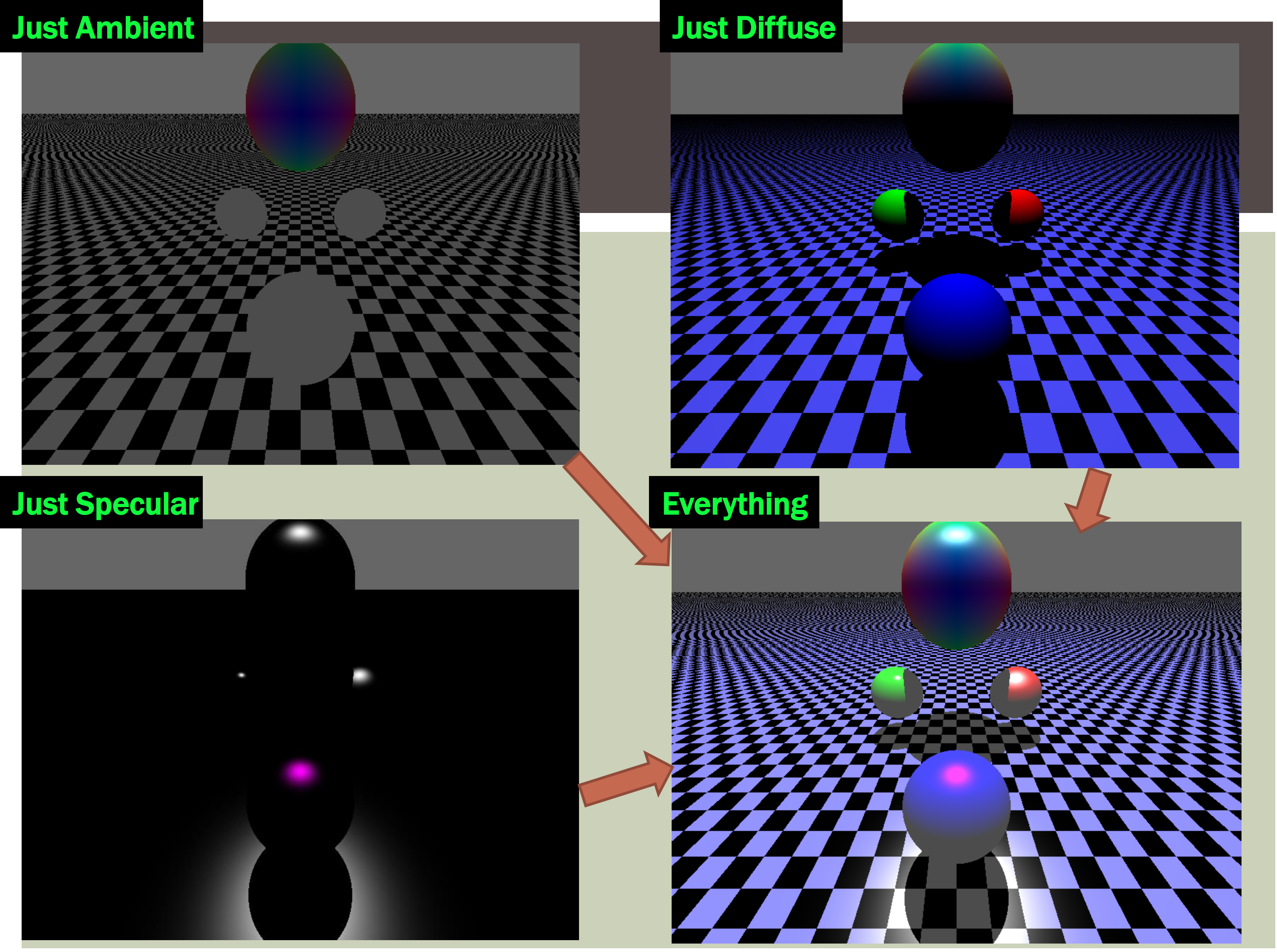
EXAMPLE



ANOTHER EXAMPLE.



Source: *http://en.wikipedia.org/wiki/Phong_shading*

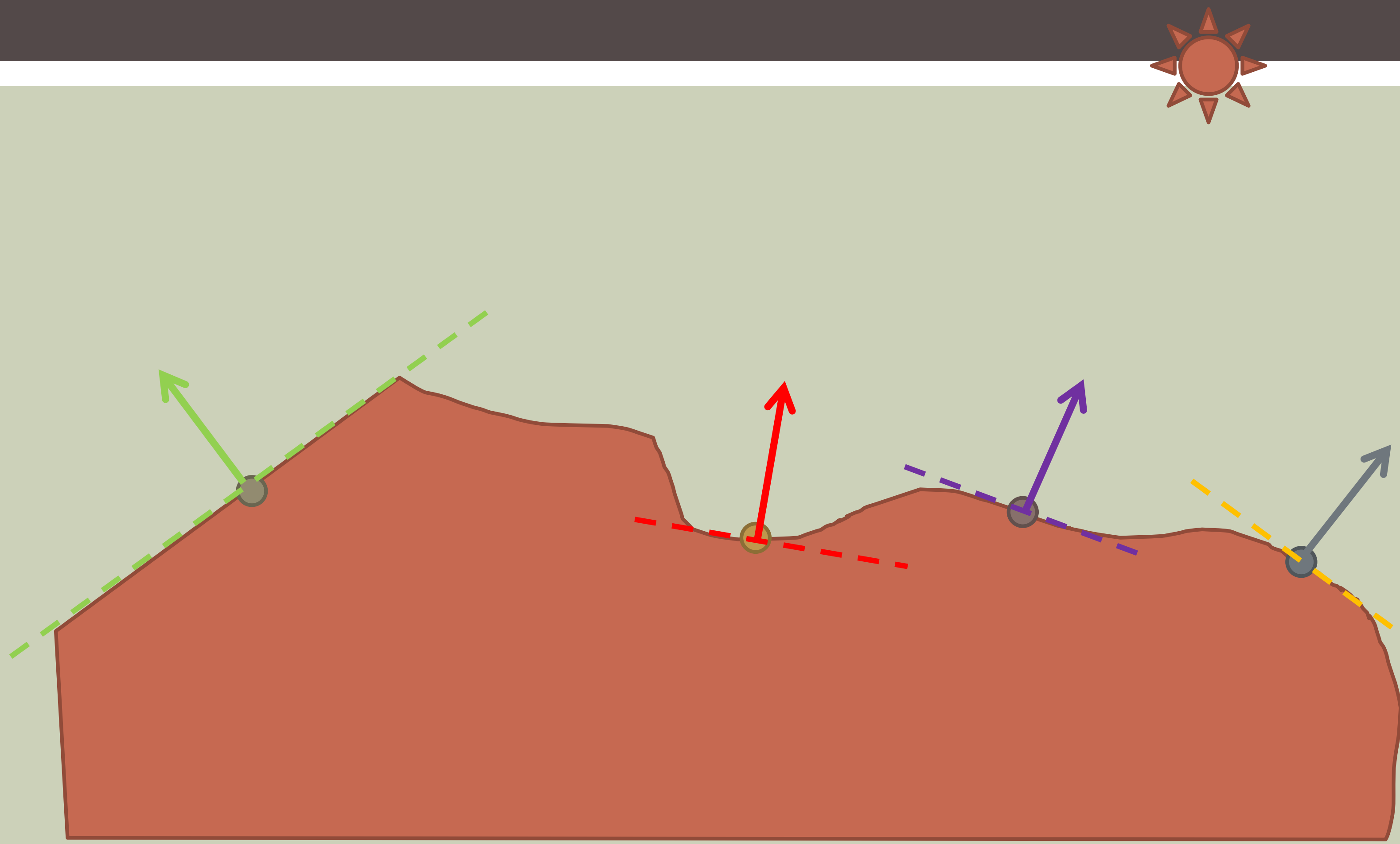

Just Ambient
Just Diffuse
Just Specular
Everything

VIRTUAL LIGHTING, FIRST STEP:

Lighting is dependent on:

- The material properties (diffuse, specular, specular-power, ambient color)
- The light source (position, diffuse and specular color, and attenuation)
  - Plus the number of lights.
  - [For spot-light, we'll also need a direction and angle(s)]
- (self-) obstructions to the light.
- The curvature at the point of illumination

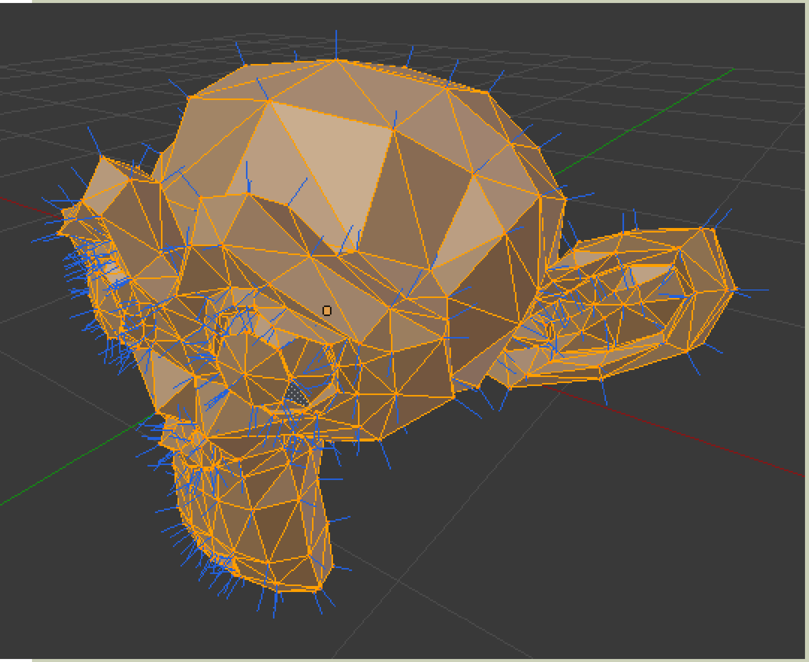
CURVATURE.



Imagine an infinitely small plane at the hit point (the dotted line) which lies on the ground. The normal vector is perpendicular to this plane.

NORMAL VECTORS, CONT.

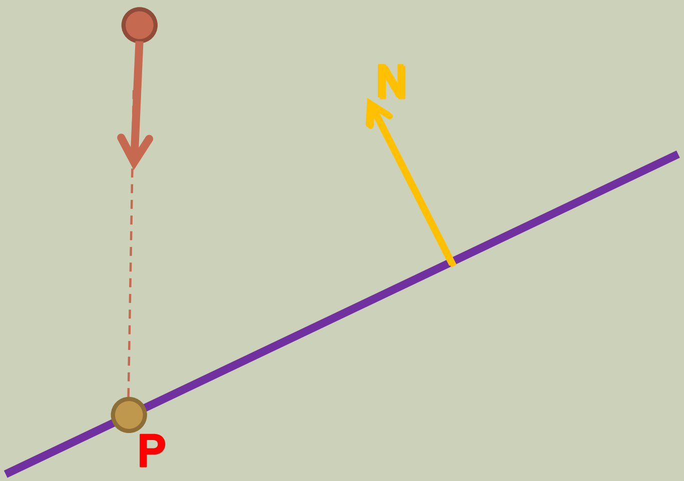
- One option: use a modelling program



```
# Blender v2.69 (sub 0) OBJ File: ''
# www.blender.org
mtllib monkey.mtl
o Suzanne
v 0.437500 0.164062 0.765625
v -0.437500 0.164062 0.765625
# 507 of these "v" lines
vn 0.671345 -0.197092 0.714459
vn -0.671345 -0.197092 0.714459
# 507 "vn" lines
usemtl None
s off
f 47//1 1//1 3//1
f 4//2 2//2 48//2
f 45//3 3//3 5//3
# 968 "F" lines
```

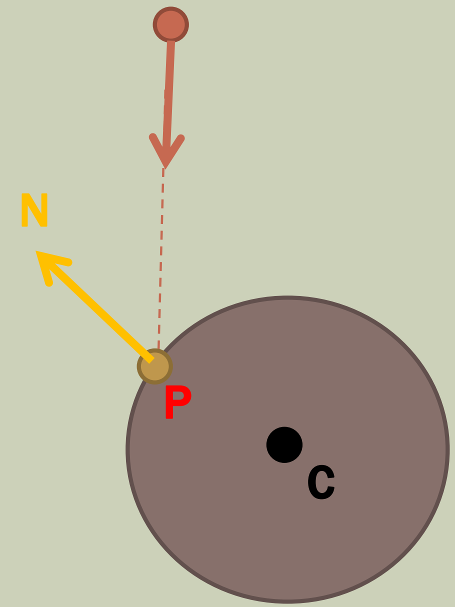
NORMAL VECTORS, CONT.

- Another option: calculate it as needed.
- The method depends on the renderable.



What’s the normal at P?

A: Anywhere on the plane, the normal will be the plane’s normal!



Not *quite* as easy here. The normal depends on where the hit point lies on the sphere.

Hint: Use C and P to calculate N.

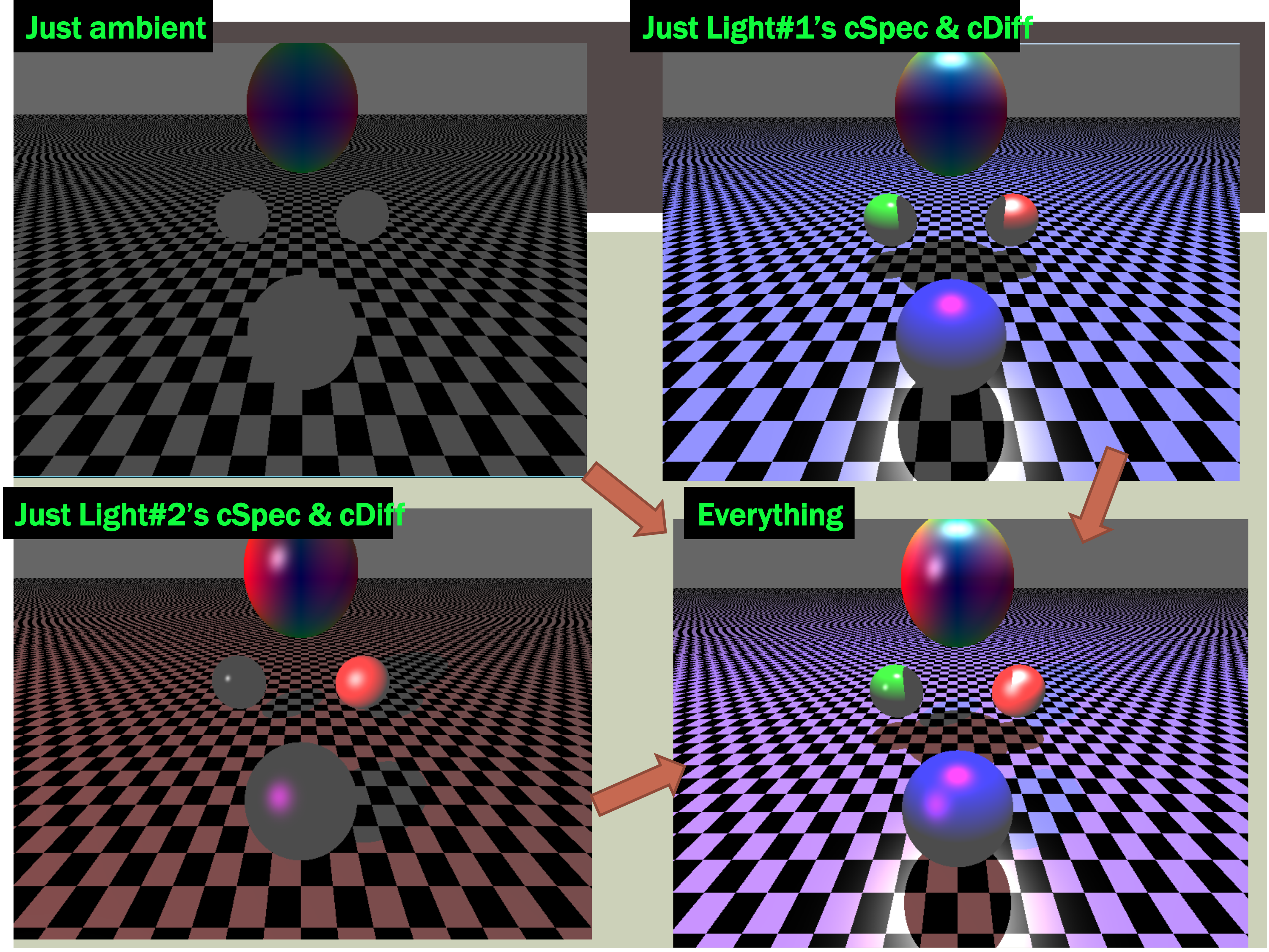
THE LIGHTING EQUATION!!! (10.6)

$$lit\_c = amb\_c + \sum_{i=0}^{numLights-1} (diff\_c_i + spec\_c_i)$$

Note1: We need to calculate a diffuse color and a specular color *for each light*.

Note2: We could have ambient "illumination" even with no light sources!

Note3: We could have lit colors over (1,1,1). So it is important that you clamp the result so that each component is in the range 0-1.


Just ambient
Just Light#1's cSpec & cDiff
Just Light#2's cSpec & cDiff
Everything

AMB_C (AMBIENT ILLUMINATION) (10.6.4)

- Remember: *amb_c* is based only upon:
  - The overall ambient light in the scene
  - The amount of ambient light the object reflects.
- Said mathematically:

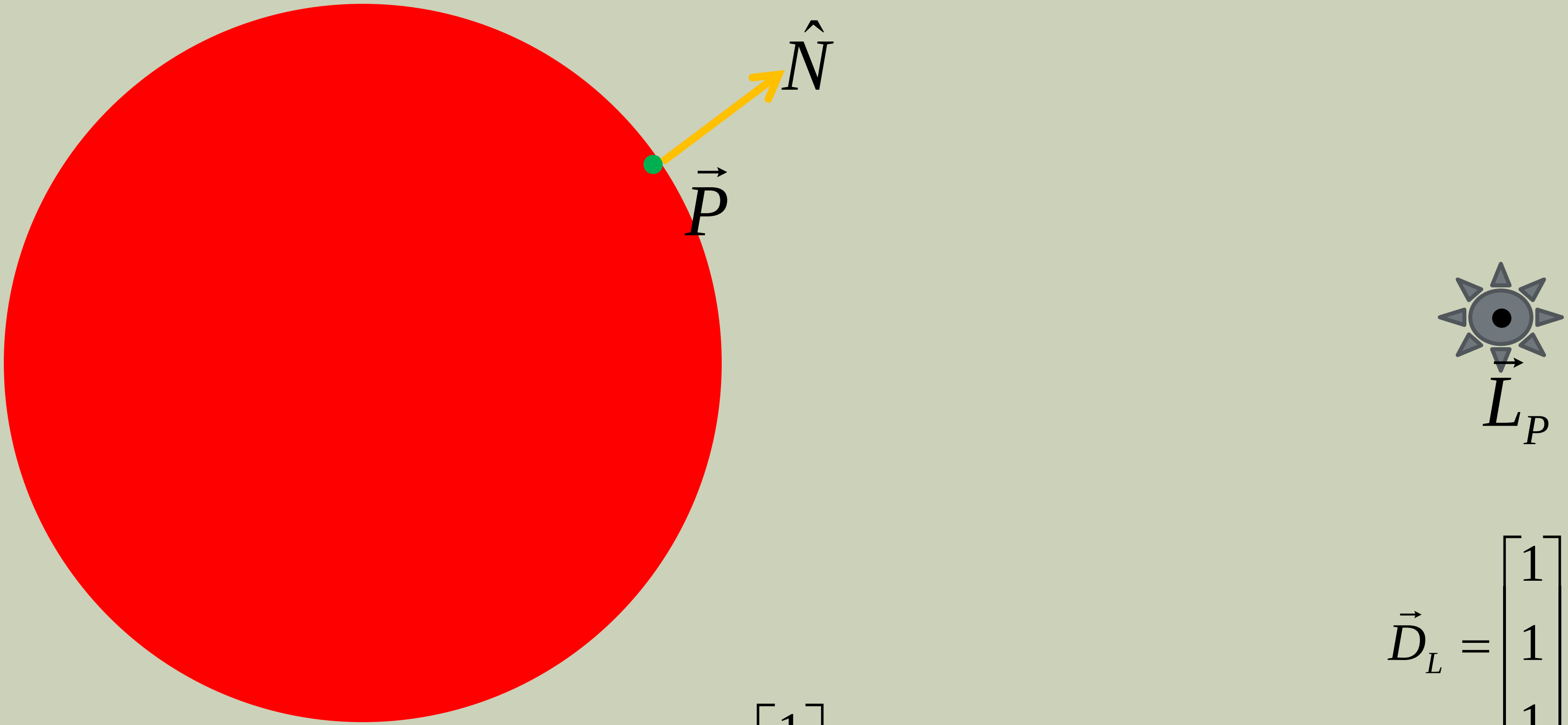
$$amb\_c = \vec{A}_L \otimes \vec{A}_M$$

  - $\vec{A}_L$ is the color vector of the (scene's) ambient light.
  - $\vec{A}_M$ is the color vector representing how much ambient light the material reflects.

DIFF_C (DIFFUSE ILLUMINATION)

- The angle the light hits a surface determines the Diffuse Illumination.
- To calculate Diffuse Illumination we'll need:
  - $\overrightarrow{L_p}$: The light position
  - $\vec{P}$: The point we're illuminating
  - $\widehat{N}$: The normal vector at the hit point
  - $\overrightarrow{D_L}$: The light's diffuse color
  - $\overrightarrow{D_M}$: The material's diffuse color

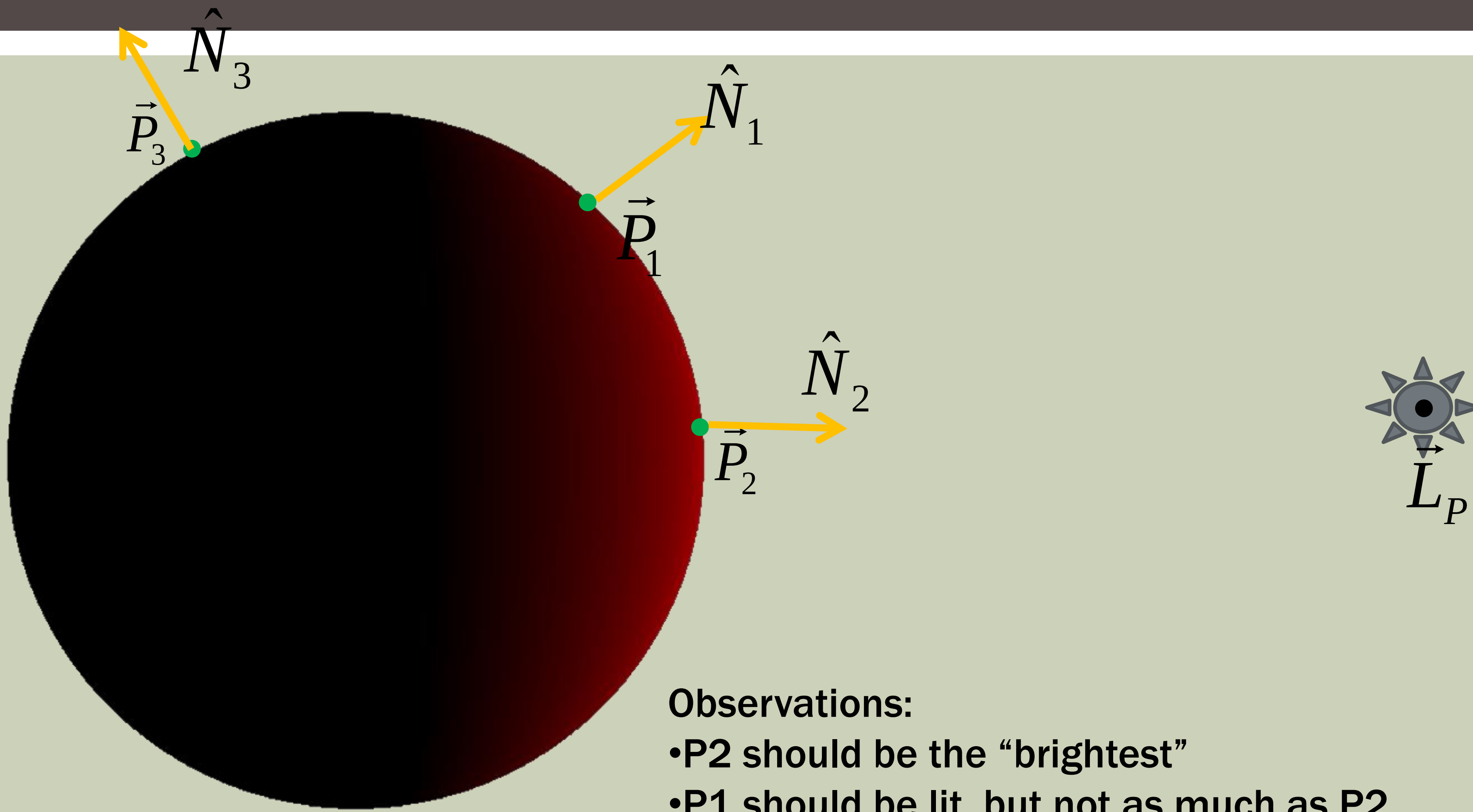
CDIFF (DIFFUSE ILLUMINATION)



$$\vec{D}_L = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\vec{D}_M = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

CDIFF (DIFFUSE ILLUMINATION)

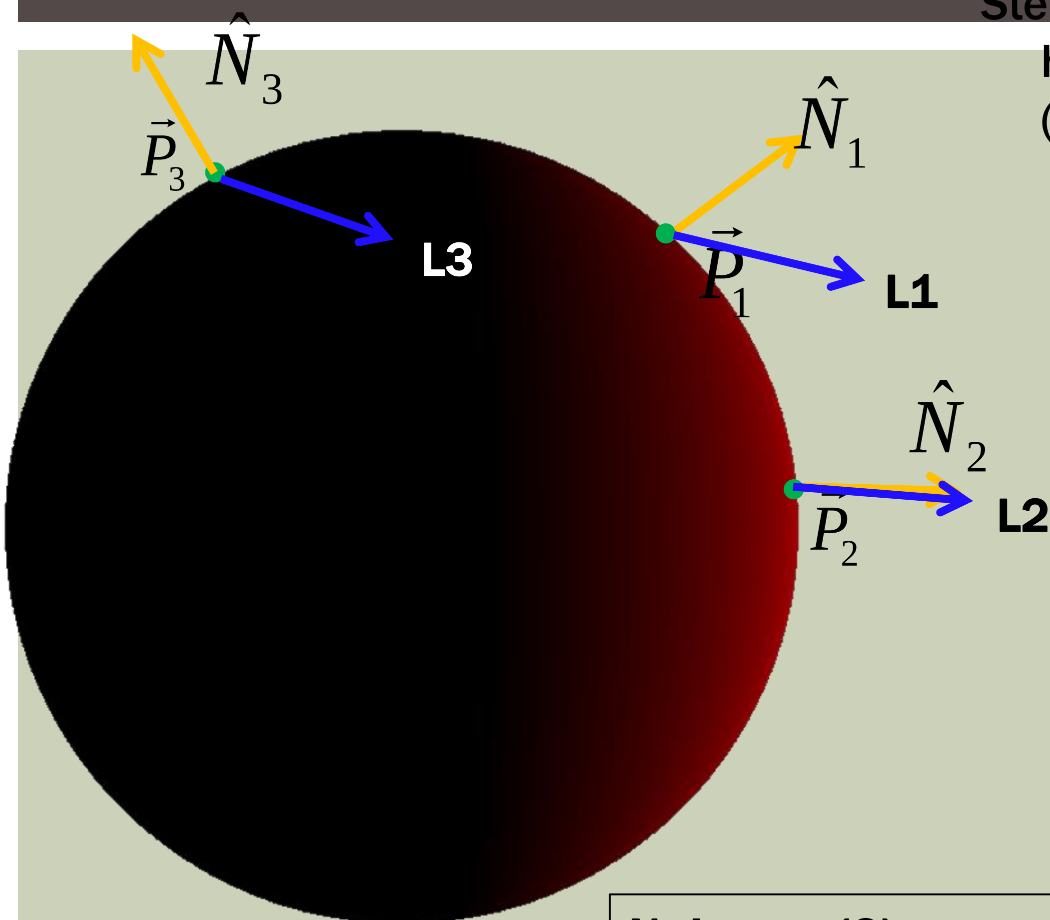


Observations:

- P2 should be the "brightest"
- P1 should be lit, but not as much as P2.
- P3 shouldn't be lit at all.

So how do we do this???

CDIFF (DIFFUSE ILLUMINATION)

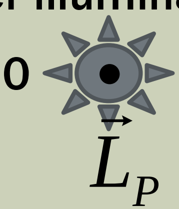


Step1: Calculate a direction from the hit point *towards* the light source (and normalize it)

Q: Notice anything about the angle made by the normal vector and the light direction vector?

A: Smaller angles == brighter illuminatio

A2: Cos(angle) is closer to 1.0



Q: What vector operation can we use to calculate this?

A: Dot product!

N•L = cos(Θ)

- This will be in the range -1…1 (since N and L are normalized)
- If N•L < 0, no diffuse illumination (so set it to 0)

CDIFF (DIFFUSE ILLUMINATION)

- So…

$$\vec{L}_{Dir} = \vec{L}_P - \vec{P}$$

$$\hat{L}_{Dir} = \frac{\vec{L}_{Dir}}{\left\| \vec{L}_{Dir} \right\|}$$

$$dStr = \hat{L}_{Dir} \bullet \hat{N}$$

$$cDiff = \begin{cases} \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \\ dStr * (\vec{D}_L \otimes \vec{D}_M) \end{cases}$$

If dStr <= 0

If dStr > 0

NOTE: You'll need to repeat this for each light. The <u>total</u> diffuse light is the sum of the dColor's for each light.

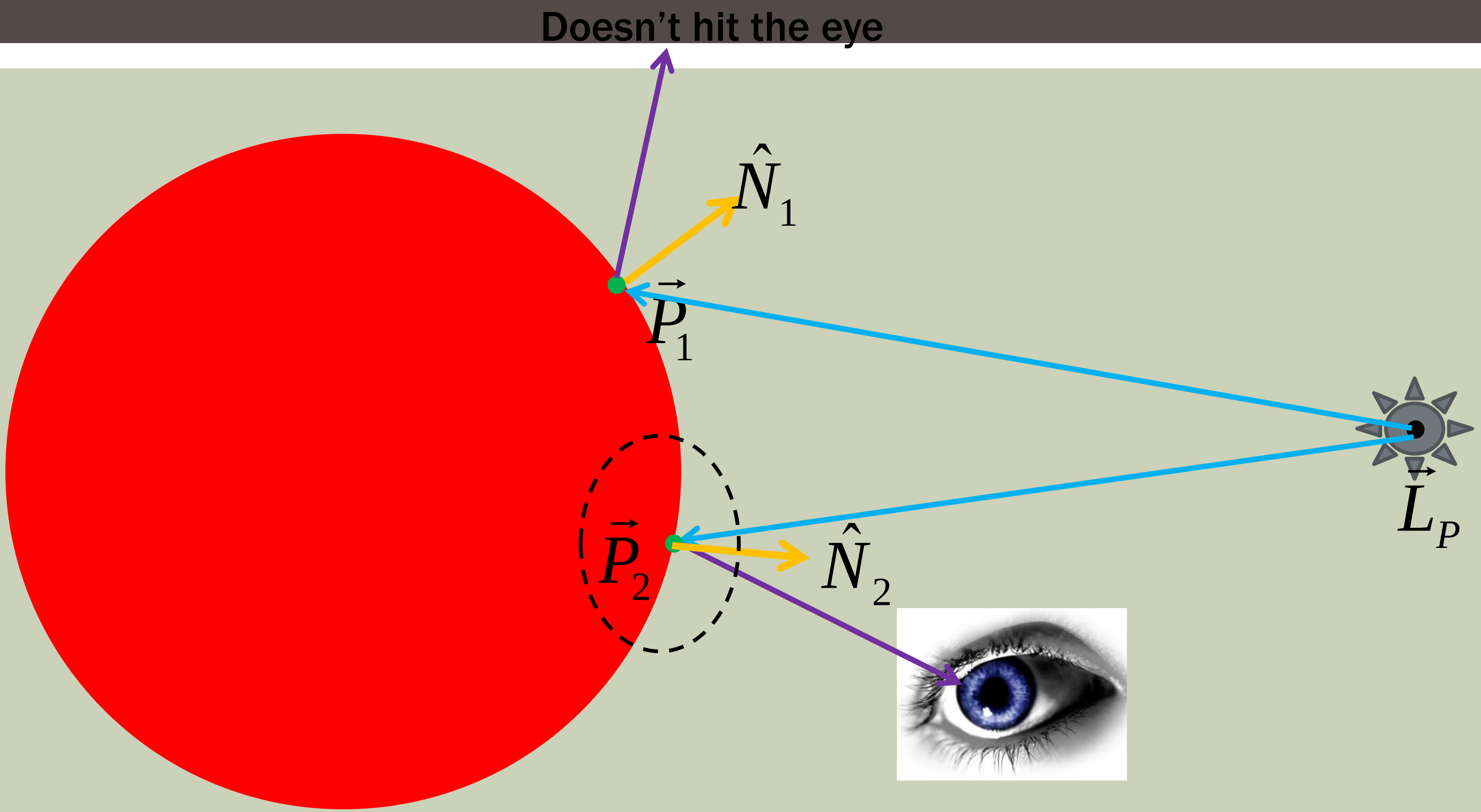
SPEC (SPECULAR ILLUMINATION) (10.6.2)

- Specular illumination tries to approximate the material-light interaction that are **mirror-like**.
- In these, light that is bounced off the surface and hits the viewer's eye will make it look as if that point were illuminated.
- If the reflected light misses the viewer's eye, there is no specular illumination.

CSPEC, CONT.

- To Calculate Specular Illumination we'll need:
  - $\overrightarrow{L_P}$: The light position
  - $\vec{P}$: The point we're illuminating
  - $\vec{N}$: The normal vector at the hit point
  - $\overrightarrow{S_L}$: *The light's specular color*
  - $\overrightarrow{S_M}$: *The material's specular color*
  - *hardness: A scalar indicating the specular "focus"*
  - $\vec{C}$: **The viewer (camera)'s position**

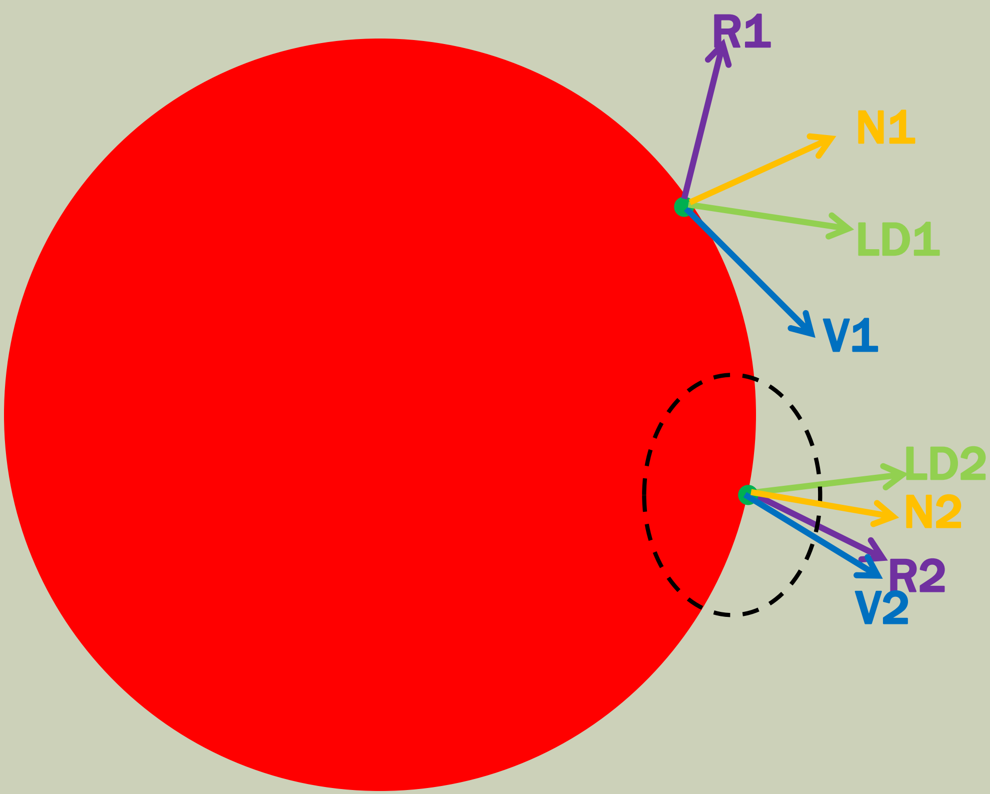
CSPEC, CONT.



Points in this dotted area would have specular illumination.
P2 would be the brightest illumination of these.

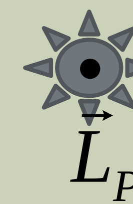
CSPEC, CONT.

Another way to look at the problem…



LD: The (unit-length) direction from the hitPt to the light.

R: The (unit-length) direction light would bounce off the surface (as in a mirror)

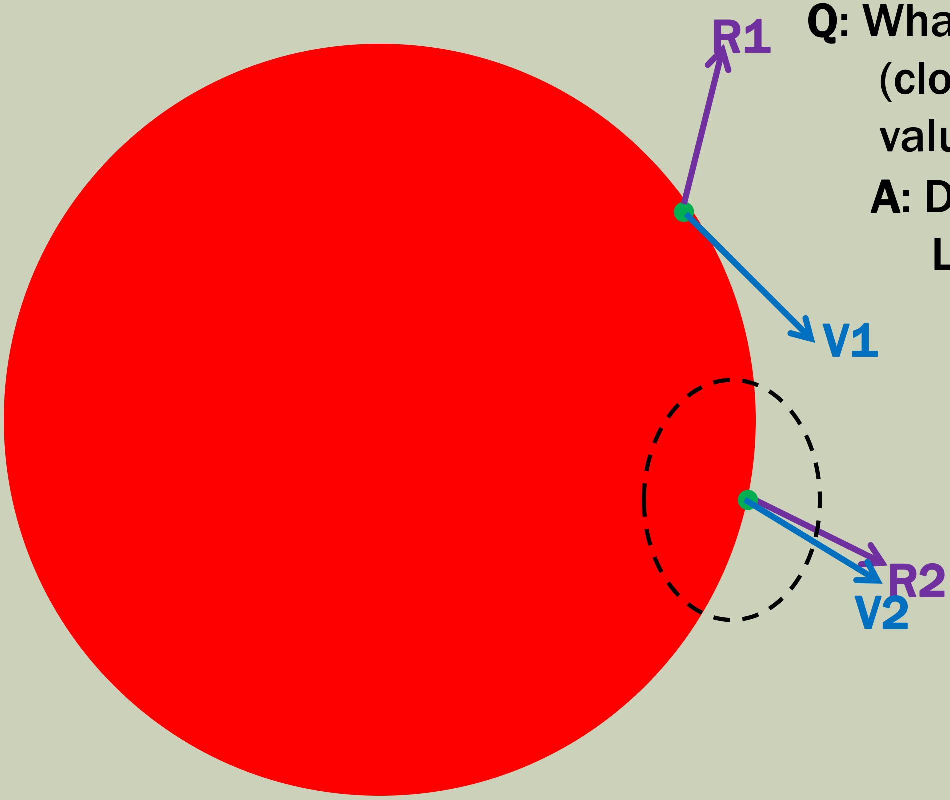


V: The (unit-length) vector from the hitPt to the camera.

Note: The angle between v1 and R1 is **big**…there should be **no** specular here.
The angle between v2 and R2 is **small**…there should be -of specular here.

CSPEC, CONT.

Now to calculate the specular strength…



Q: What vector operation will give us a high (close to 1.0) value for R2&V2, but a low value (0.0) for R1&V1?

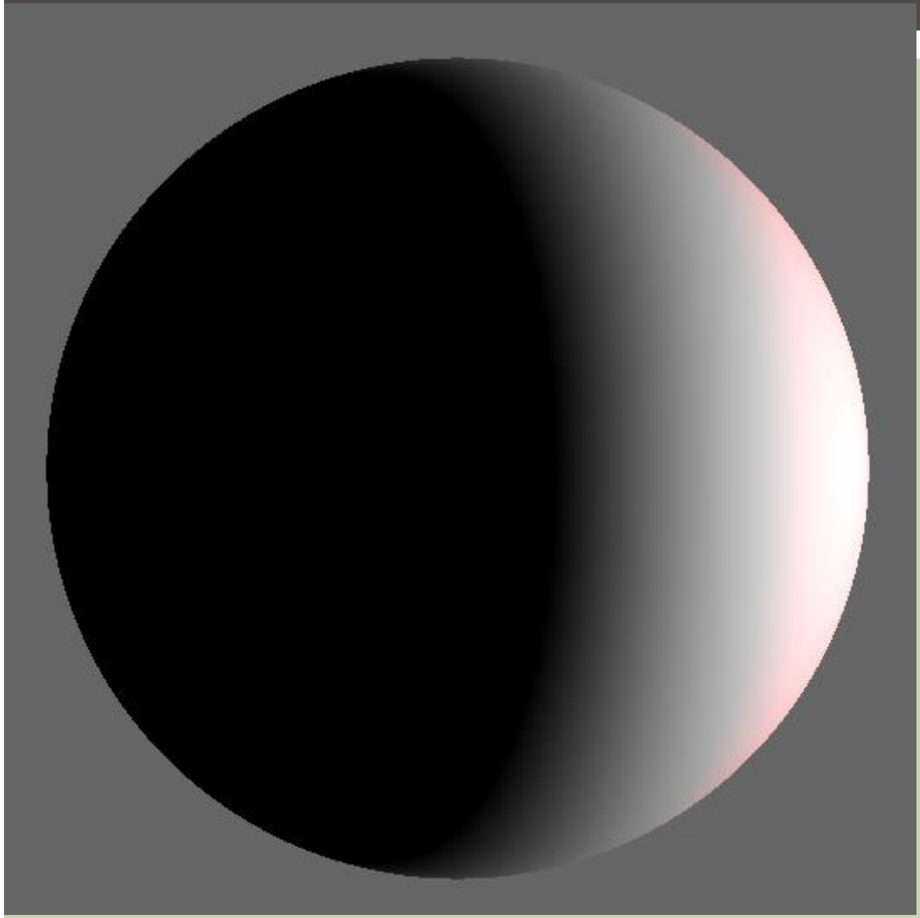
A: Dot-product (just as we used N-dot-LD for the Diffuse calculations)!

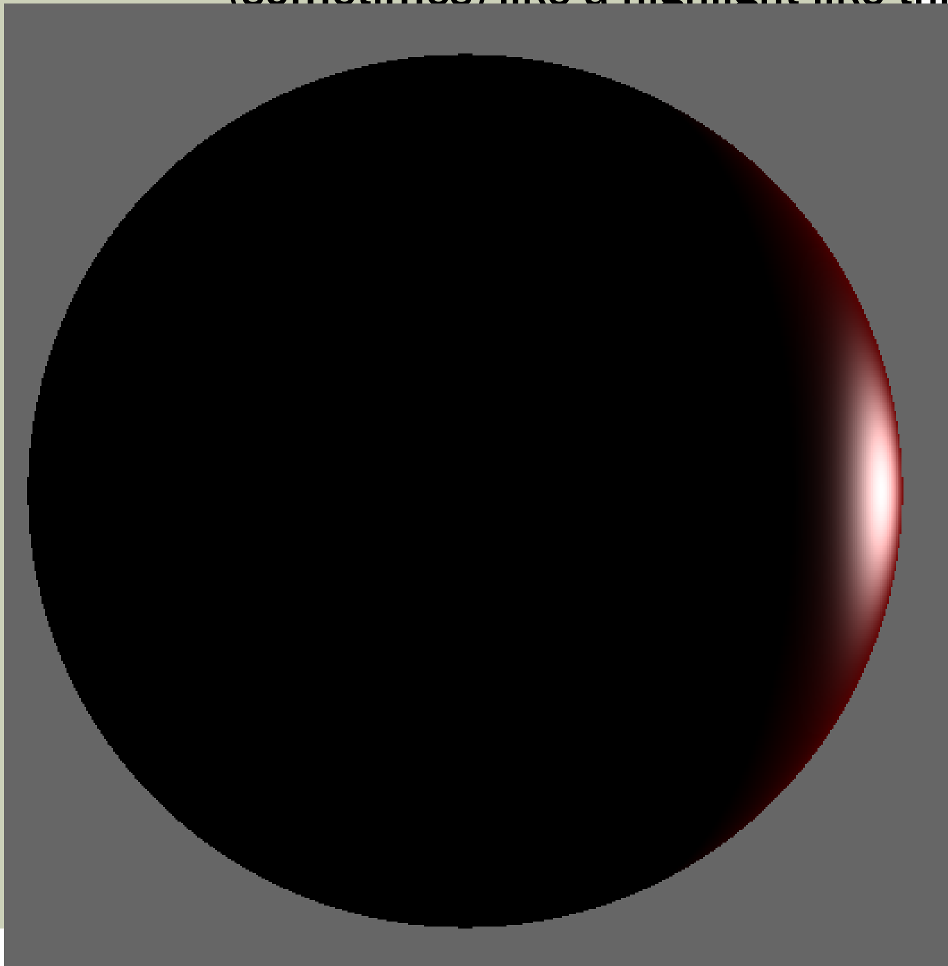
$$sStr = \max(\vec{R} \bullet \vec{V}, 0)$$

CSPEC, CONT.

Camera at (-5,0,15). White light. Sphere (pos=(0,0,0), diffuse=(1,0,0), spec=(1,1,1))

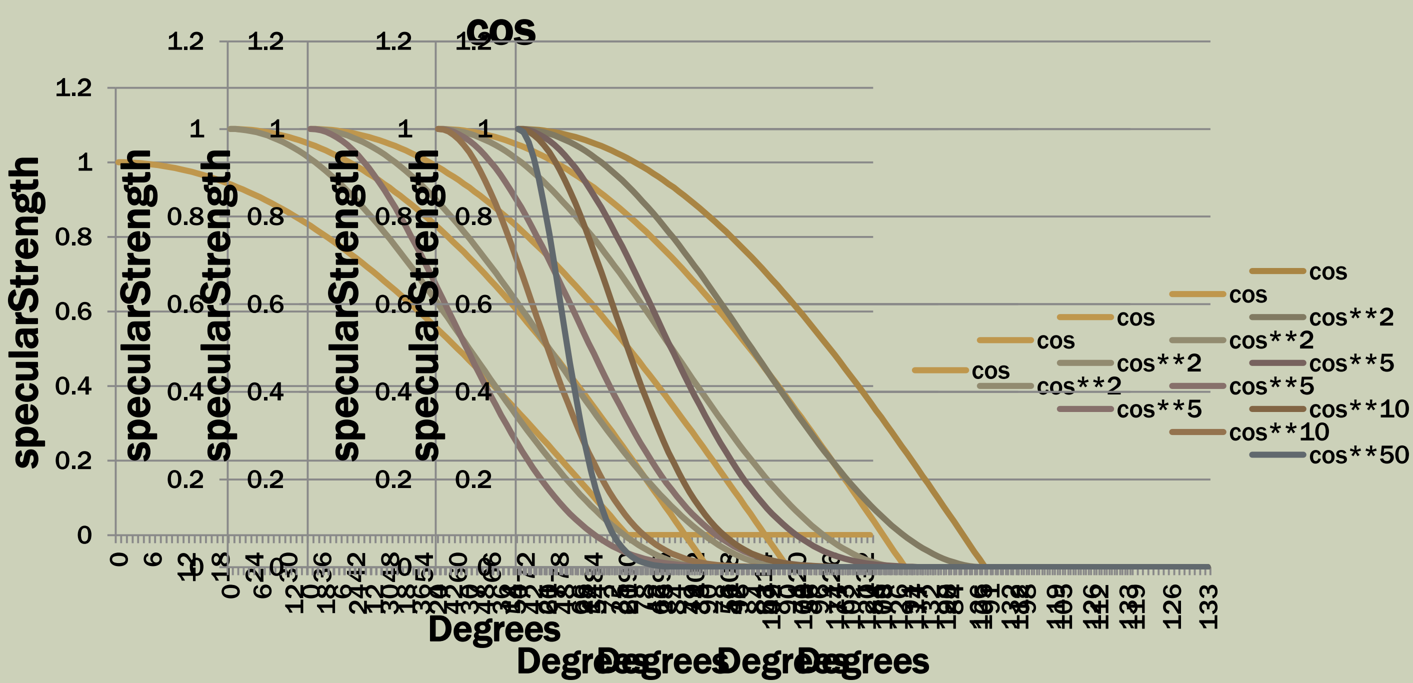
Problem: Notice how the specular hilight is very “spread out” – it’s almost lik diffuse shading. This might be fine sometimes, but we’d also (sometimes) like a highlight like thi





CSPEC, CONT.

- Here's a graph of specularStrength as a function of θ, the angle made by **V** and **R**:



Cos(θ) (i.e. V-dot-R) falls off "gradually" as the angle increases (until it finally reaches 0 at θ=90).

We want a "sharper" fall-off.

How do we get it?

A trick: Watch what happens if we square cos(θ)…
…or raise it to larger powers…

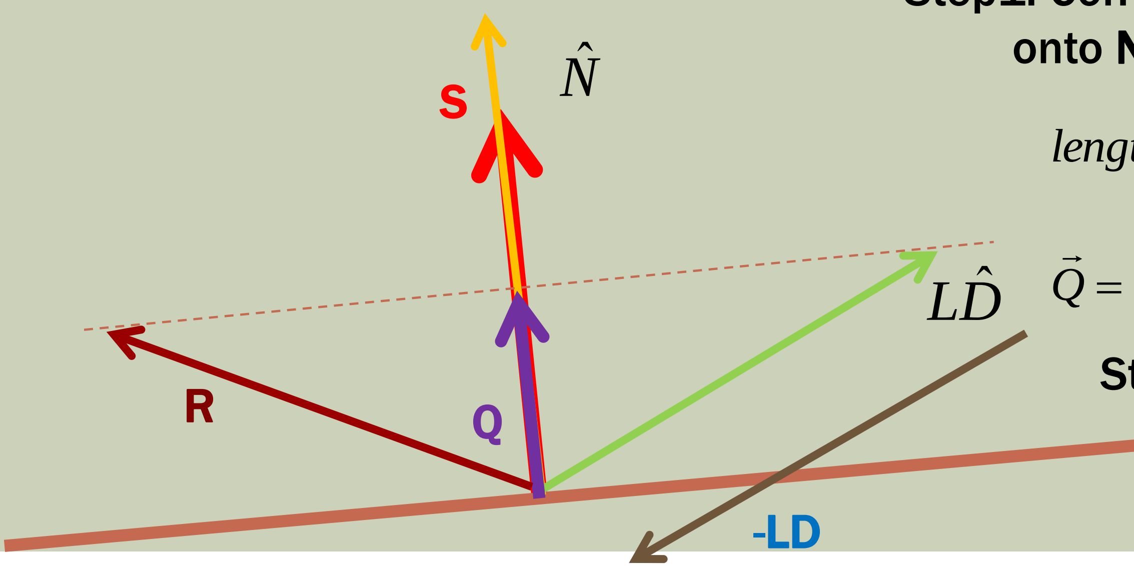
CSPEC, CONT.

- One final problem: How do we compute R (the reflected light direction)?
  - We want to “mirror” LD (the direction towards the light) about N (the normal at the hit point).
  - Note: This is "Phong" reflection. There are others (the book also has Blinn)

Step1: Compute Q, the projection of LD onto N.

$$length-of-Q=\frac{\hat{LD}\bullet\hat{N}}{\|\hat{N}\|}=\hat{LD}\bullet\hat{N}$$

$$\vec{Q}=(\hat{LD}\bullet\hat{N})*\hat{N}$$



Step2: Compute S, which is 2*Q

Step3: R is now S – LD (which is the same as S + (-LD))

SPEC, CONT.

- So here's our complete algorithm for calculating the specular illumination:

$$\vec{L}_{Dir} = \vec{L}_P - \vec{P}$$

$$\hat{L}_{Dir} = \frac{\vec{L}_{Dir}}{\left\|\vec{L}_{Dir}\right\|}$$

$$\vec{R} = 2*(\hat{L}_{Dir} \bullet \hat{N})*\hat{N} - \hat{L}_{Dir}$$

$$\vec{V} = \vec{C} - \vec{P}$$

$$\hat{V} = \frac{\vec{V}}{\left\|\vec{V}\right\|}$$

$$sStr = \hat{V} \bullet \vec{R}$$

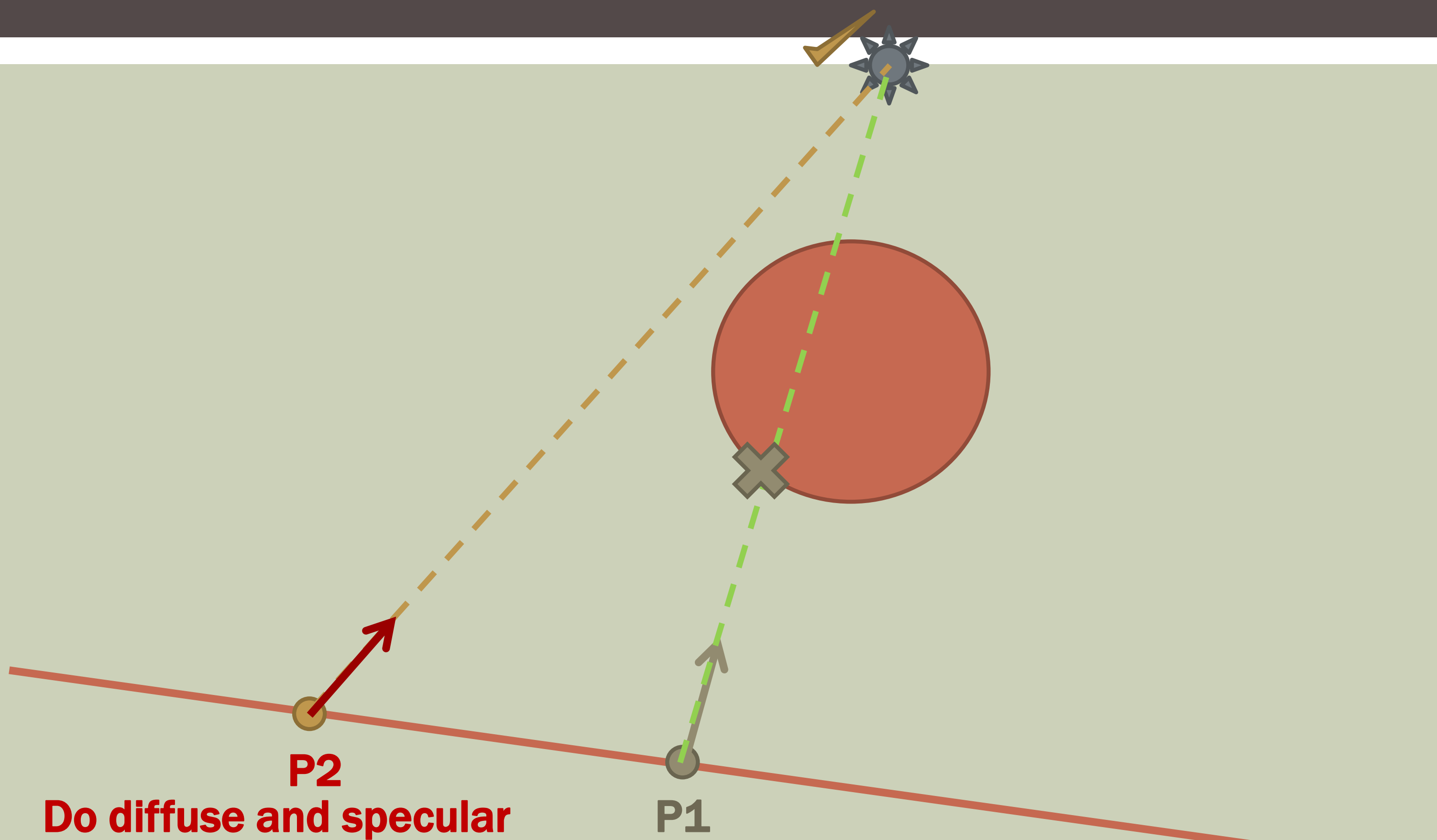
$$cSpec = \begin{cases} \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & \textit{If sStr <= 0} \\ sStr^{shiny} * (\vec{S}_L \otimes \vec{S}_M) & \textit{If sStr > 0} \end{cases}$$

NOTE: You'll need to repeat this for each light. The <u>total</u> specular light is the sum of the sColor's for each light.

SHADOWS

- Raytracer shadows are quite easy.
- Algorithm:
  - When lighting a point P…
  - …If the light is blocked by another renderable, don't add diffuse or specular illumination.
  - …If it's not blocked, illuminate it as normal.

SHADOWS, CONT.



Do diffuse and specular as normal.

Don’t do diffuse or specular for this point (for this light)

Does this look familiar?

SHADOWS, CONT.

- Sure it does!
- We're just *casting a ray*:
  - Here the origin's not the pixel plane, it's a point we're illuminating.
  - The direction is towards the light.
  - If we hit anything but the object the origin-point is on before hitting the light, don't illuminate!
- Hint: If you calculate the distance between the light and origin-point, the t-values for any hits must be less than this value.