

-=Illuminating=-

Anton Gerdelan
Trinity College Dublin

Refresh from Part I

- Shading models review
 - Q. Non-interpolating model?
 - Q. Gouraud's model?
 - Q. Phong's model?
- Reflection models review
 - Q. Difference between local and global illumination?
 - Q. What does Lambert model?
 - Q. How does Phong's reflection work?

Phong Lighting is the Sum

$$i = i_a + i_d + i_s$$

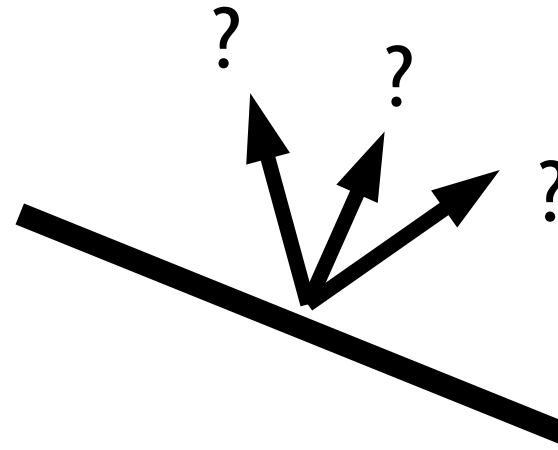
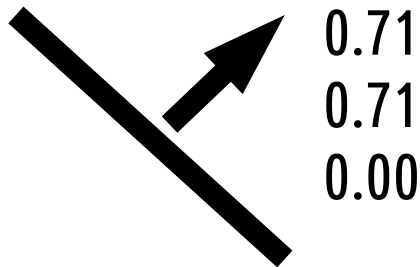


[somewhat open] Questions

- Q. How can we model a non-shiny surface?
- Q. Is any real surface completely matte?
- Q. What is physically inaccurate about Phong lighting?
- Q. Do any real surfaces have a non-white specular colour?
- Q. What is missing from this lighting model to make it convincing?

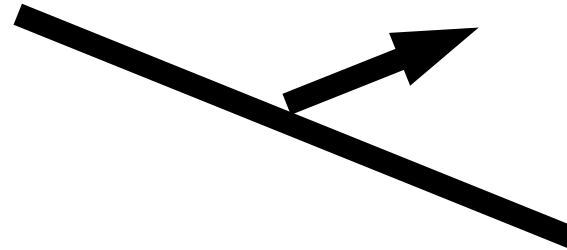
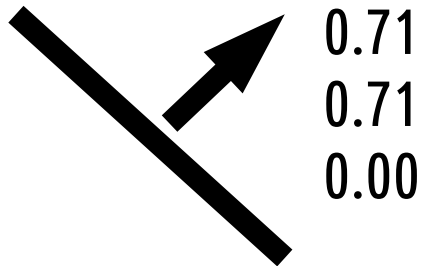
Warning: Corrupted Normals

- Q. Why should we never apply a non-uniform scaling to a normal?



- Q. Work out values of normal if we scale by (2.0, 1.0, 1.0)
- Q. Which way is our new normal pointing?

Warning: Corrupted Normals

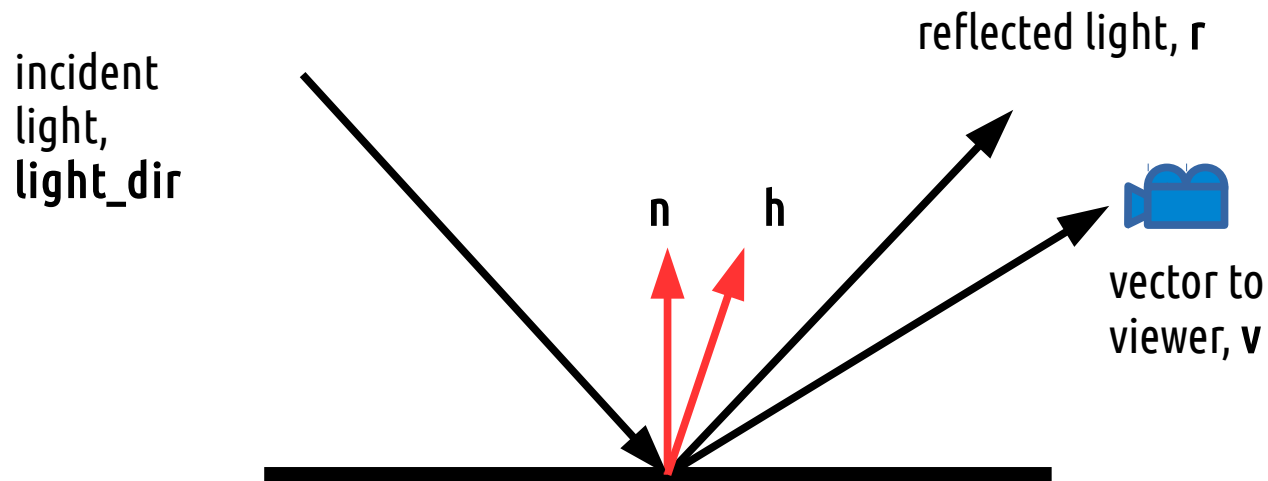


- How can we avoid this?
 - a) Create a separate model matrix with just the rotations “normal matrix”
 - b) Take inverse (transpose (model_matrix)) instead
 - c) Don't do lighting on things with uneven scaling
 - d) Don't ever do uneven scaling

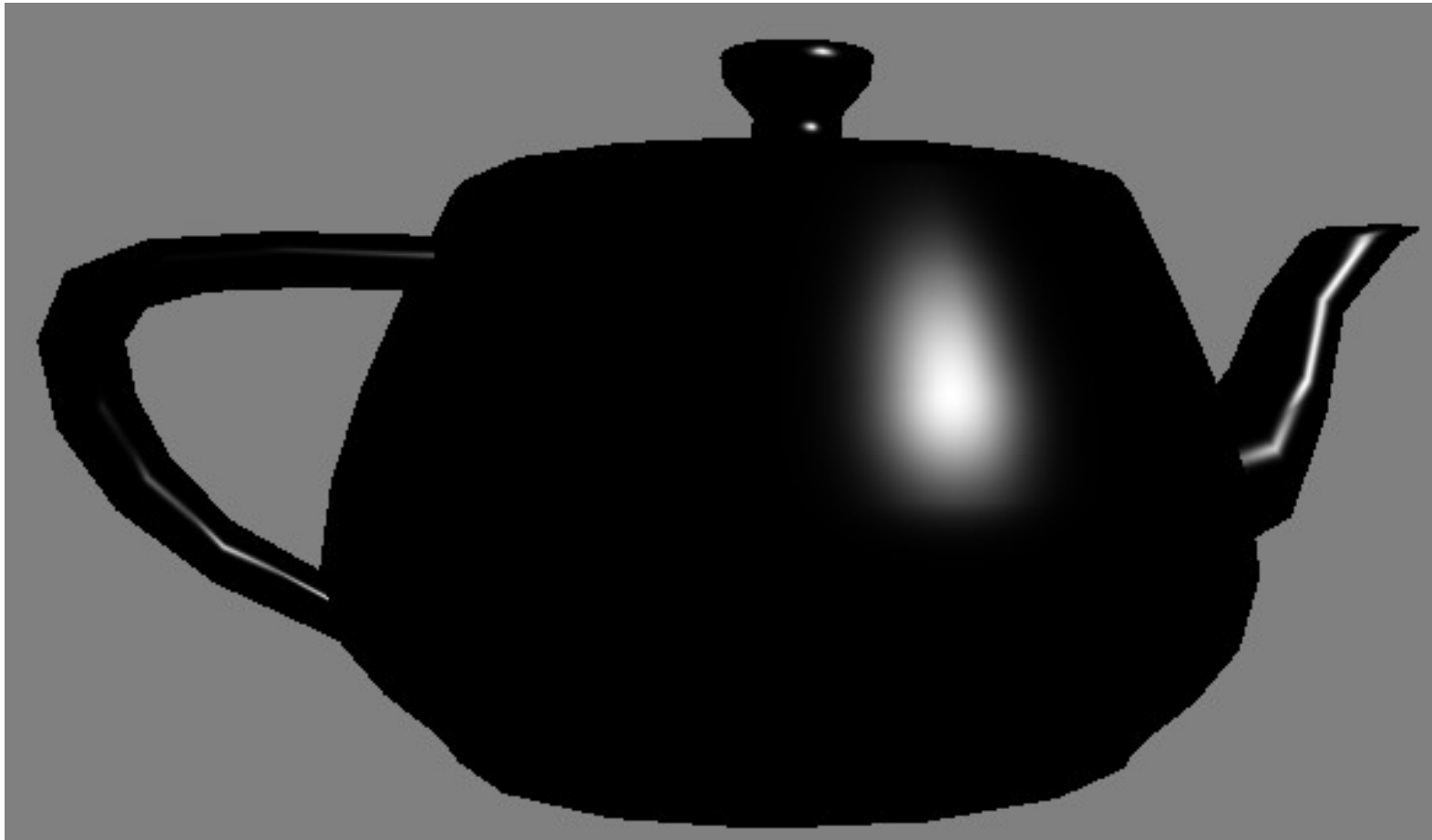
Blinn-Phong

- Lose a small amount of accuracy in specular equation
- Little bit cheaper to calculate
- Replace `reflect()` with a **half-way vector**:

```
vec3 h = normalize (v - light_dir);  
vec3 I_s = l_s * k_s * pow (dot (h, n), spec_exp);
```



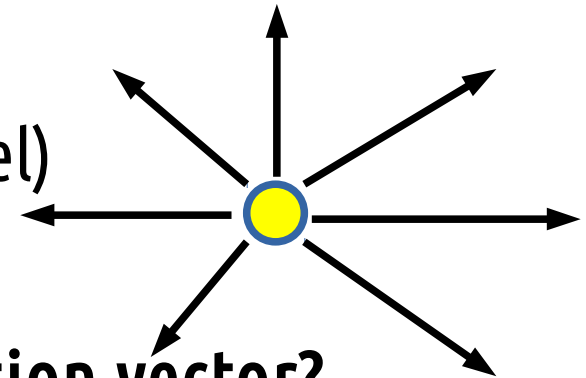
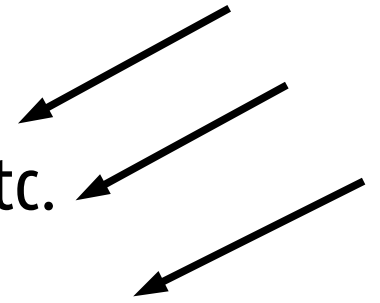
Blinn-Phong



- Reduces specular power by about half \rightarrow double the exponent

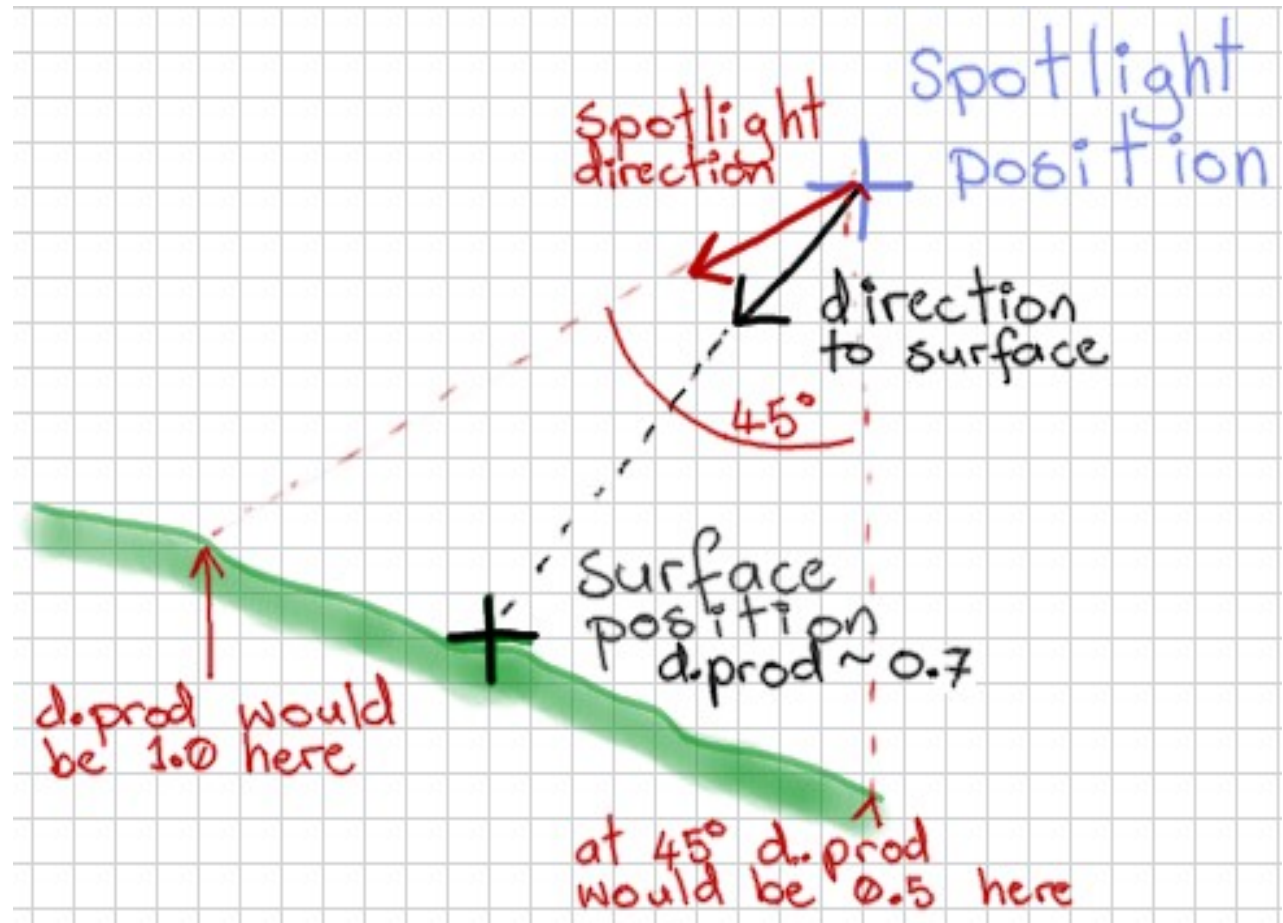
Point / Directional / Spot Lights

- Were built-in to fixed-function graphics libraries
- **Directional light** (our light vector)
 - Good for representing the sun, very distant lights etc.
 - Parallel
- **Point light**
 - Shines equally in all directions (not parallel)
 - Has a world position
 - **Q. how do we calculate the light direction vector?**



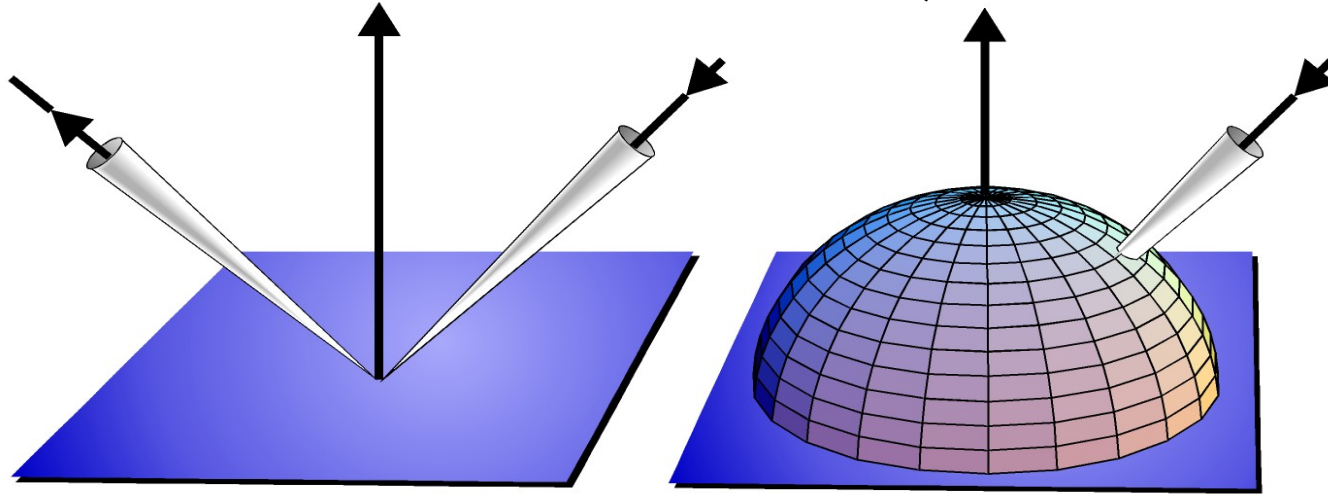
Spot Lights

- Cone of light
- Has **light direction** vector
- Has cone **angle**
- Centre of cone = brightest
- Edge of cone = no light

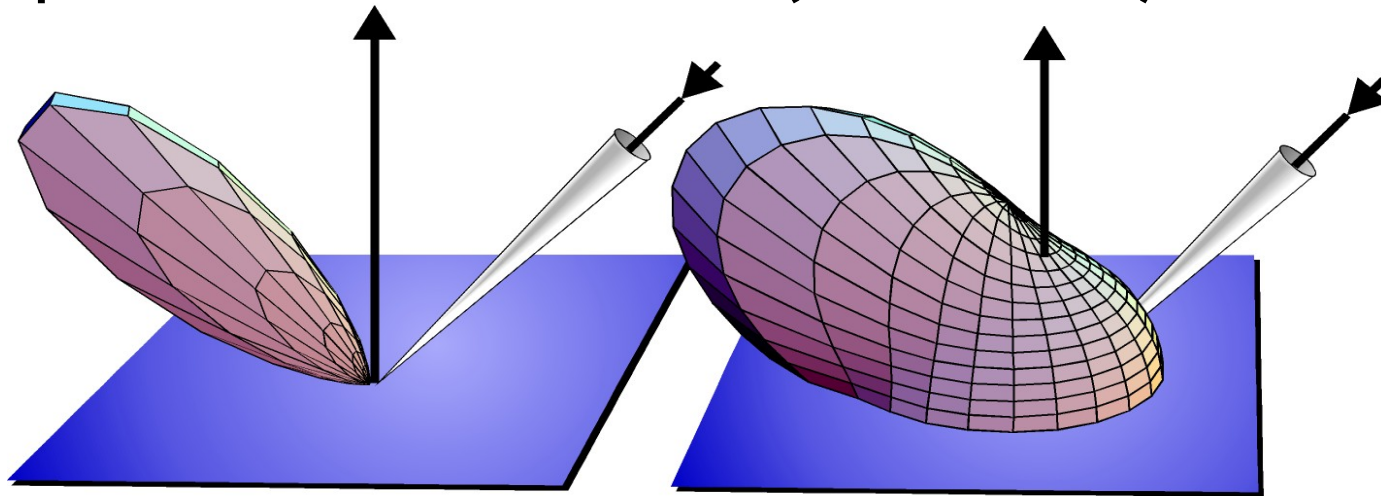


- Q. How can we work out:
 - If we our fragment is inside the cone or not
 - How close our fragment is to centre of the cone (0.0 to 1.0)

BRDF (Bi-directional reflectance distribution function) models



Ideal specular and ideal diffuse (Lambertian) reflections



Anisotropic specular (Ward, SIGGRAPH '92?), rough diffuse

Materials and Textures

- Q. How would we combine our texturing with lighting?
- Q. What if we want only **parts of a surface** to be shiny?

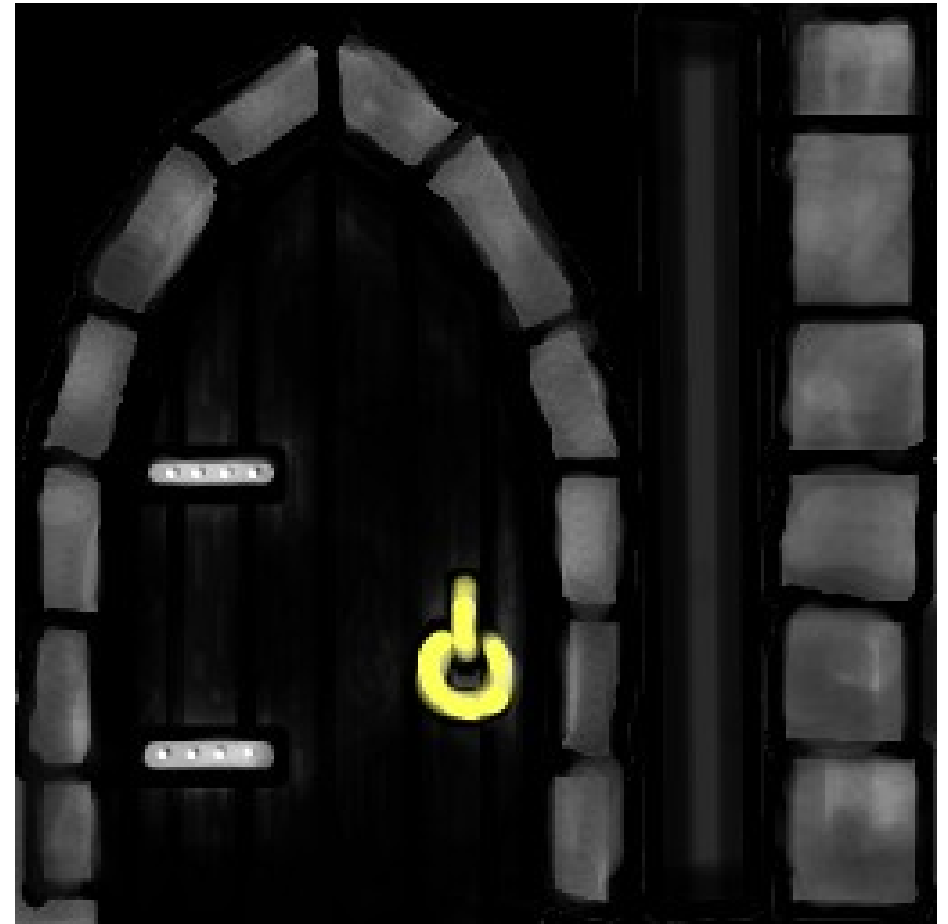
Attenuation (Roll-Off) with Distance

- Constant attenuation factor (reflection reduces energy)
- Linear attenuation factor (distance, easy to perceive)
- Quadratic attenuation factor (distance, hard to perceive)
- Realistic (for a spherical light source) – combine all 3

$$\text{atten_factor} = 1.0 /$$
$$(k_c + k_l * \text{distance} + k_q * \text{distance}^2))$$

$$k_c = 1 \quad k_l = 2 / \text{radius} \quad k_q = 1 / \text{radius}^2$$

Materials and Textures



Diffuse Map

Specular Map

DEMO TIME

Multi-Texturing

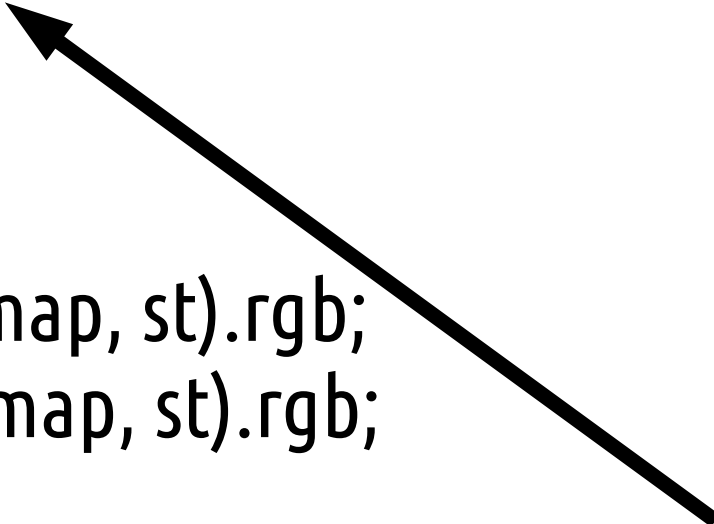
```
#version 420
```

```
layout (binding = 0) uniform sampler2D diff_map;  
layout (binding = 1) uniform sampler2D spec_map;
```

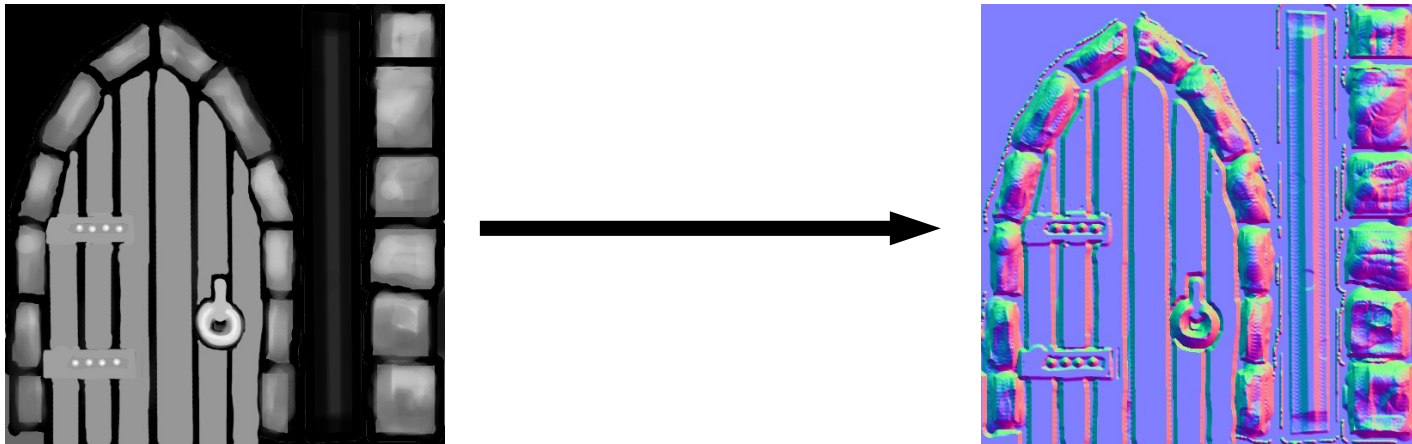
```
...
```

```
k_d = texture (diff_map, st).rgb;  
k_s = texture (spec_map, st).rgb;
```

reads from
`glActiveTexture(GL_TEXTURE1)`



Other Light-Texture Combos

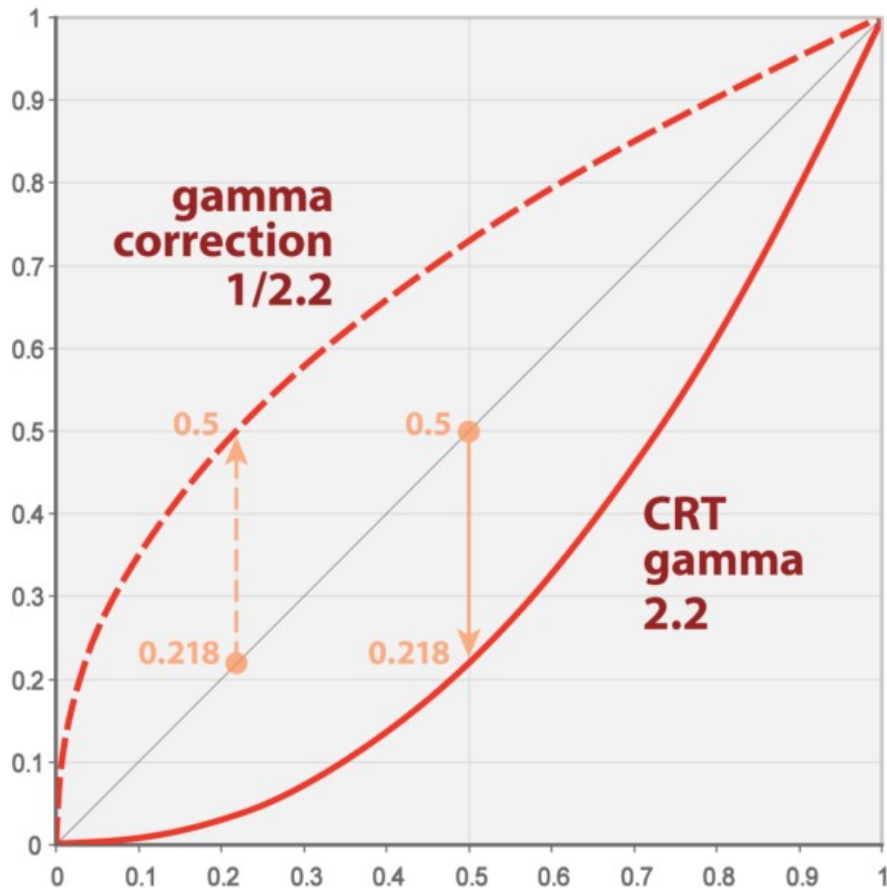


- Ambient Occlusion maps – bake in Blender
- Normal maps (quite tricky to set up)
 - rgb colours == xyz normals per-fragment
 - generate from a height map
 - create height-map from mesh in Blender

Questions

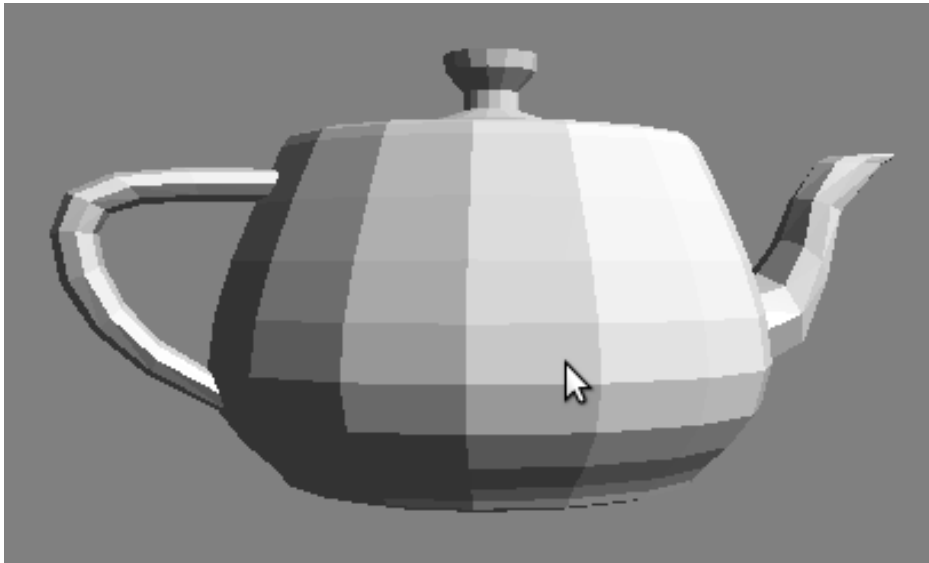
- Q. More than one light? How?
- Q. Problems with that?

Gamma Correction



- Output linear colours = voltages
- Display not linear response
- = $\text{our_colour}^{2.2}$
- = dark
- Add inverse of response curve to make colour linear
- Don't correct textures unless
- sRGB format colour palette

Gamma Correction

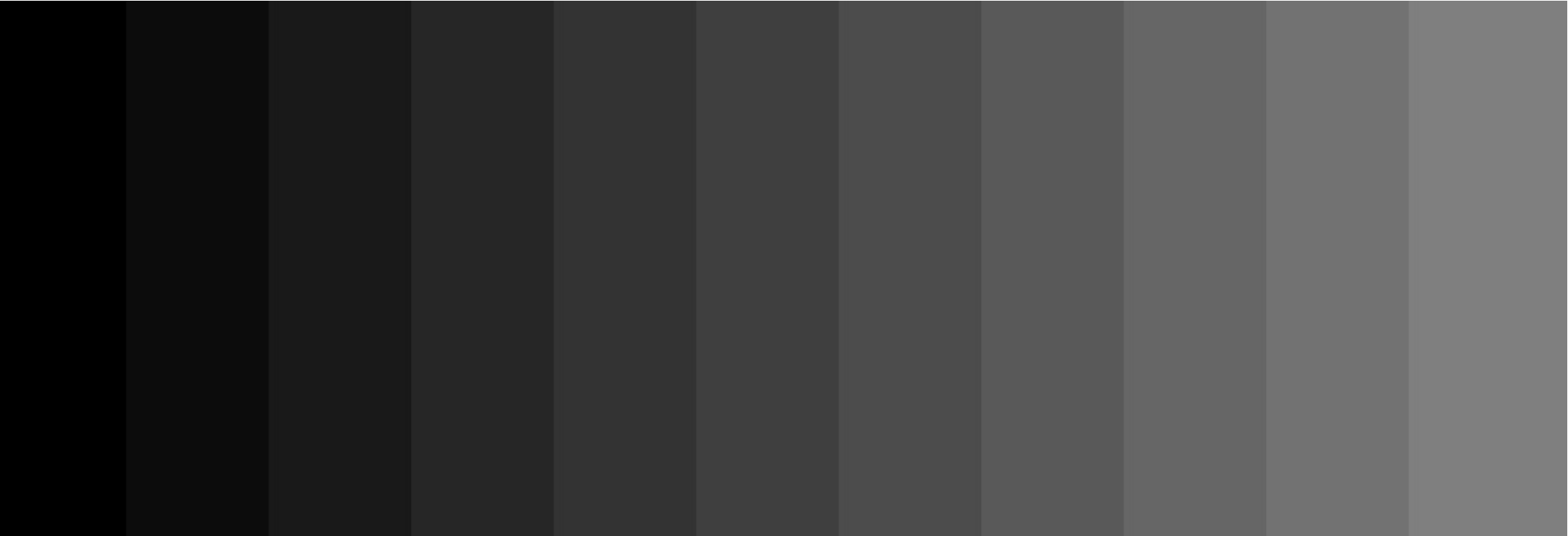


```
colour = pow (colour, vec3 (2.2, 2.2, 2.2));
```

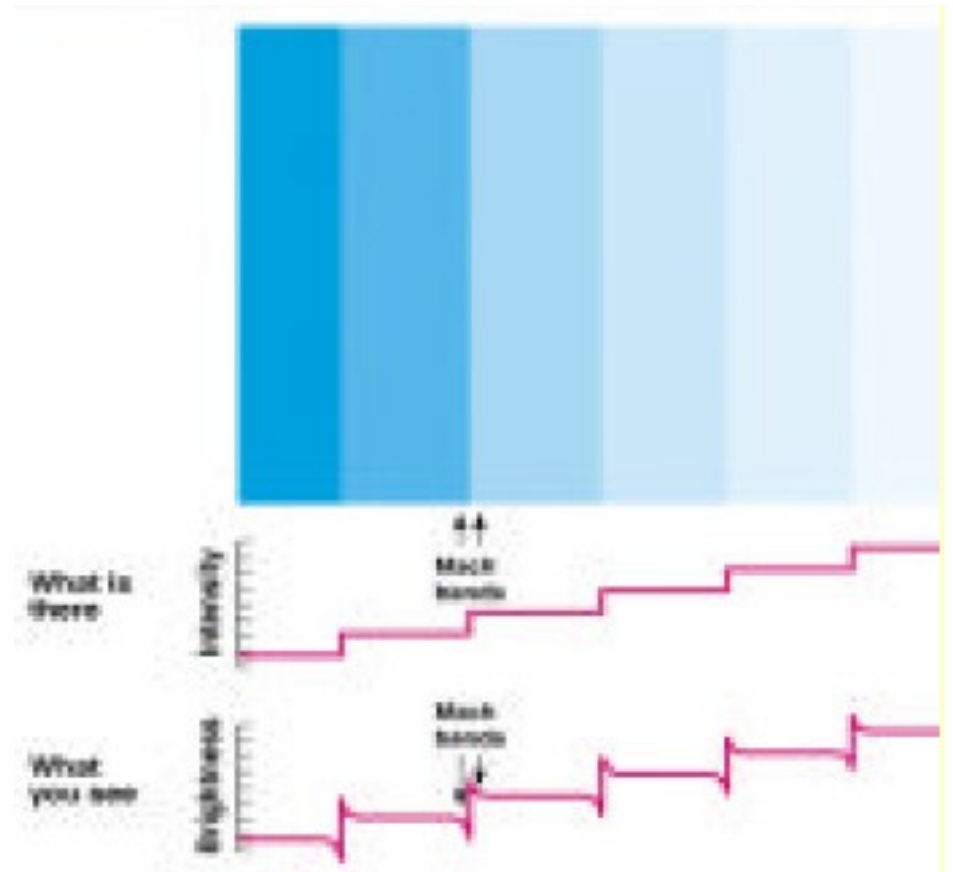
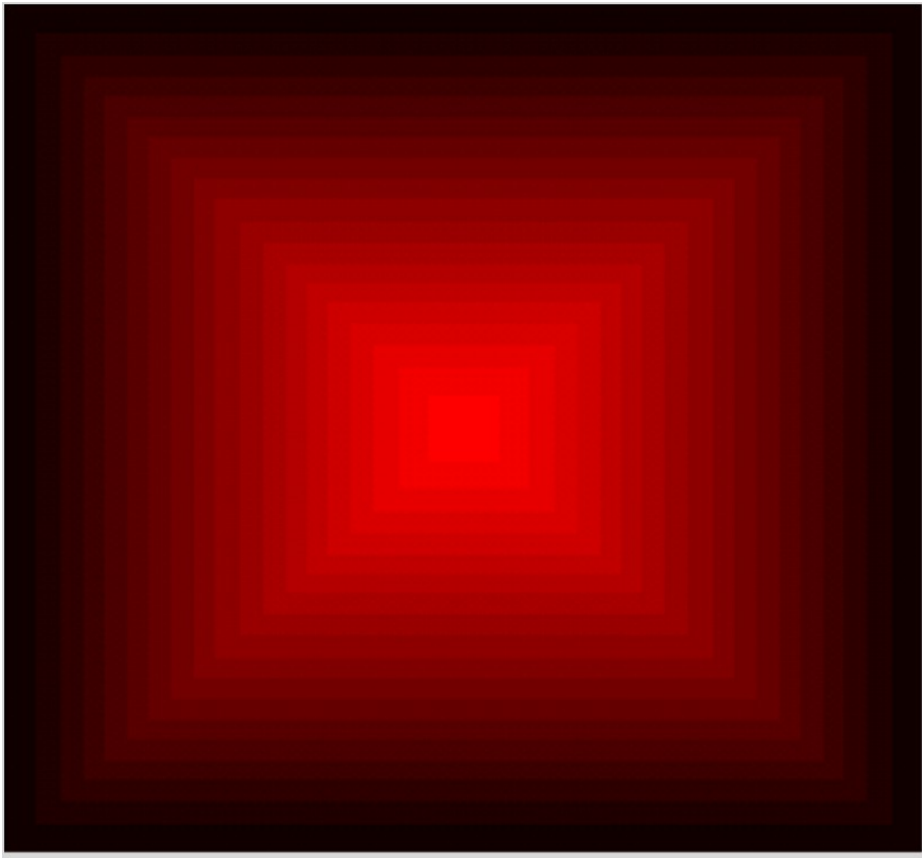
Gives you the full range of colour intensities on your display

Mach Banding

- Optical illusion named after physicist Ernst Mach
- Perceived contrast at edges - exaggerated by human vision system
 - We have built-in edge-detection



Mach Banding



- The bright diagonal lines do not exist
- Cause: excite/inhibit behaviour in neural processing

