Computer Graphics - Week 7

Lighting and Shading

Dr Simon Lock

Mid Term Feedback

Thanks to those who submitted mid-term feedback
A couple of comments worth mentioning...

"More information on nature and format of exam"
Full briefing (plus past paper) later - after CW !

"Weekly briefings are too much intro & context"
"Better to dive a bit deeper into teaching material"
Let's try this out - see what style people prefer...

Lighting

Let's try making our lighting more sophisticated
To keep it simple, we make following assumptions:

- There is only a single source of light
- This light is a point-source (no width/height)

This will make our job a lot more straight-forward
But will still produce some nice looking light effects

(Maybe think about more complex lighting later on)

3 Types of Light !

It is "convenient" to consider 3 types of light:

- Diffuse: Direct illumination of the surface
- Ambient: All around, background light level
- Specular: Mirrored reflection of the light source

ThreeTypesOfLight

This concept (3 types) does not occur in nature
A quick "approximation" rather than a "simulation"
(But it does give us some plausible looking results)

Diffuse Light

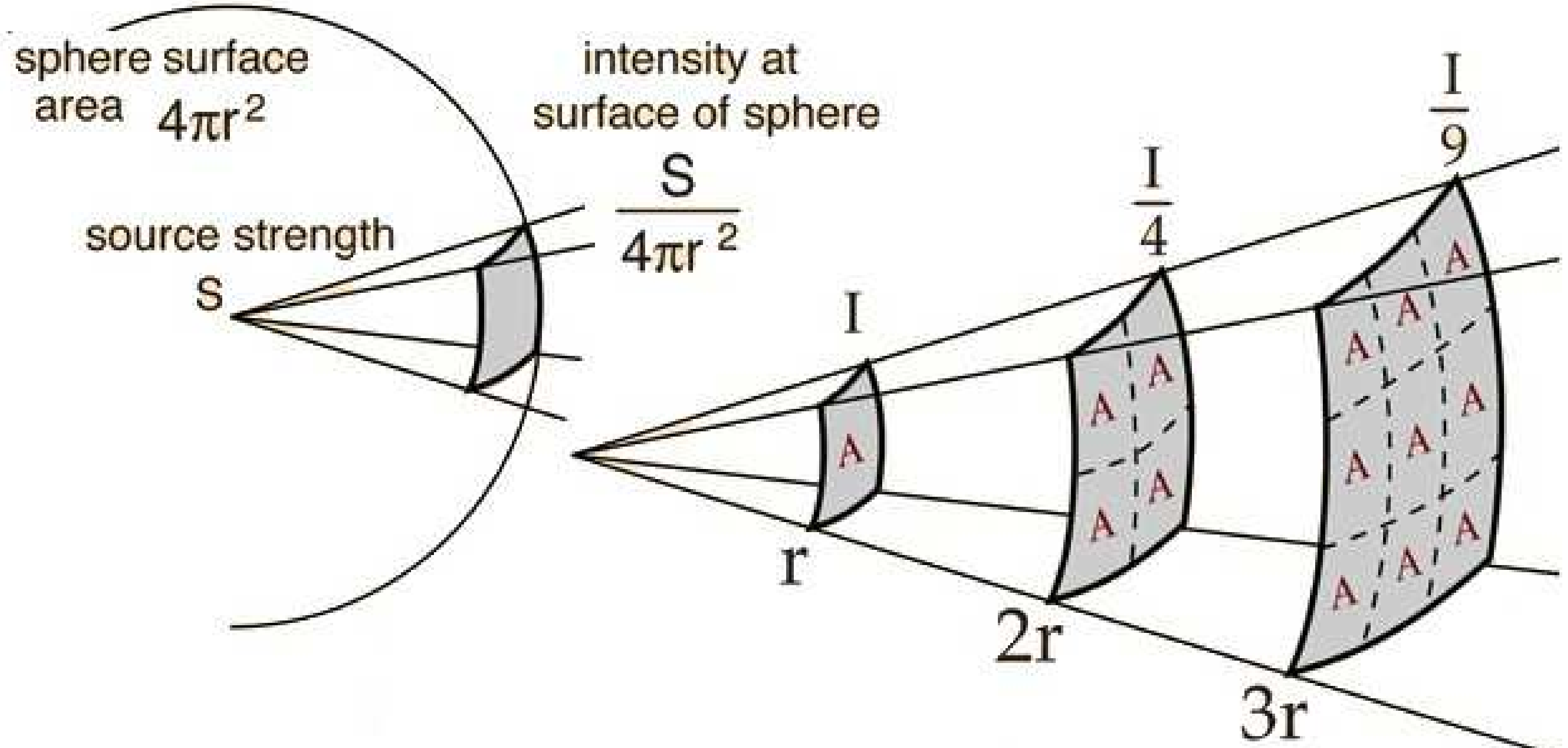
Think of this as "standard" or "ordinary" light effect

To calculate we must consider BOTH the following:

- The distance of the surface from light
- The angle the light falls onto the surface

Let's take a look at each of these in turn...

Light Intensity Drop-off (Dissipation)


sphere surface area $4\pi r^2$
source strength S
intensity at surface of sphere $\frac{S}{4\pi r^2}$
I
A
r
$\frac{I}{4}$
2r
$\frac{I}{9}$
3r

Light Angle-of-Incidence

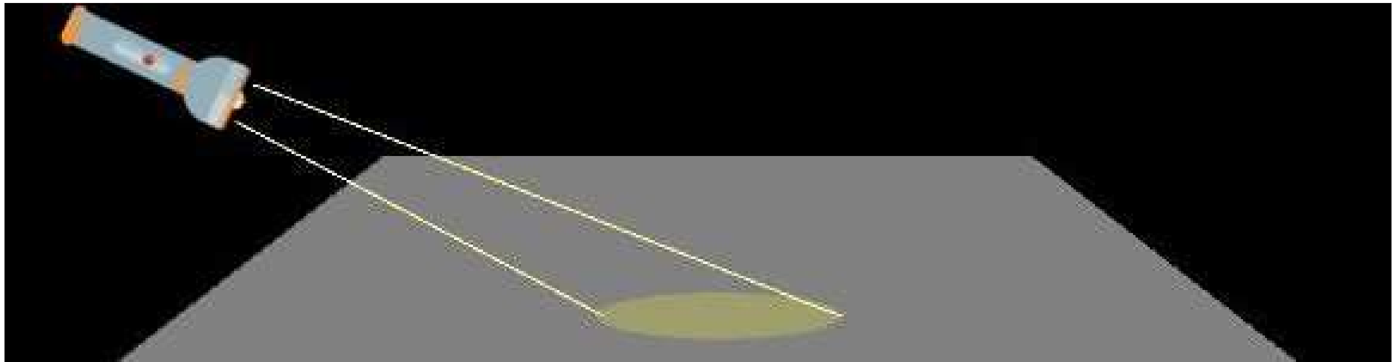
We also must take into account "angle of incidence"

The more oblique light strikes, further it is spread

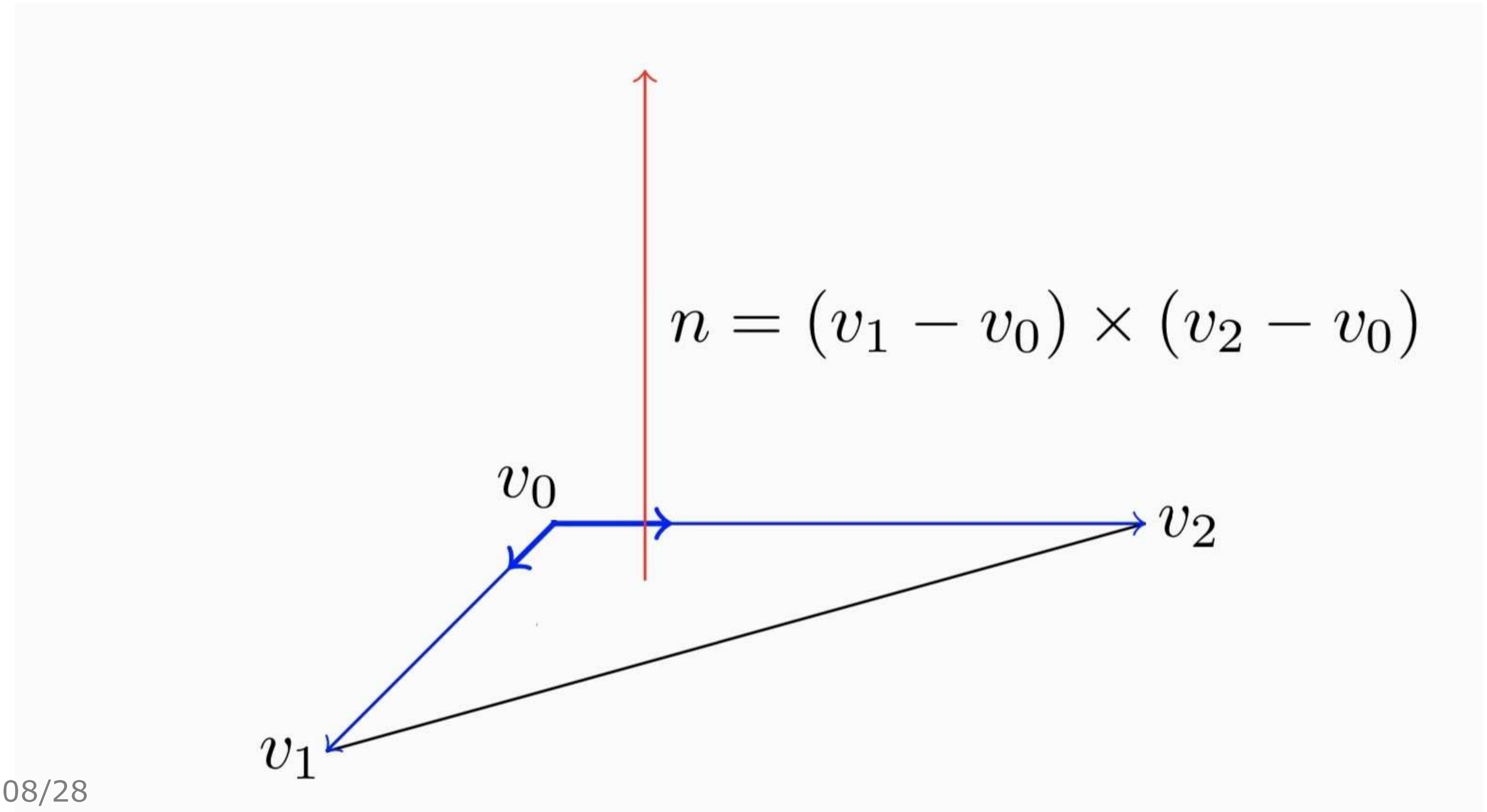
In order to calculate the angle of incidence

We need to know the position of the light

And also the orientation of the surface...



Calculate Surface Normal using Cross Product

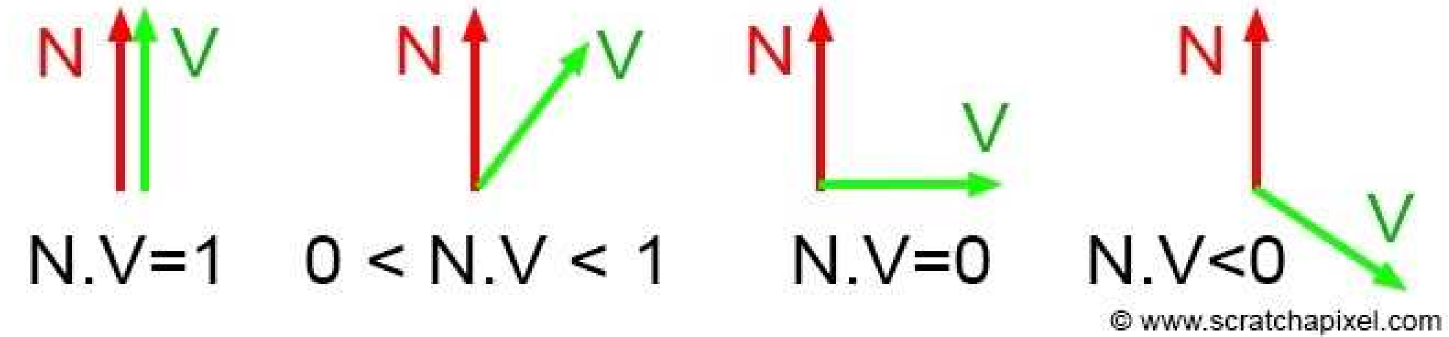


Calculating Angle-of-Incidence

dot product "surface normal" with "vector-to-light"

The result tells us the angle between the two:

- 1.0 if normal and vector-to-light are parallel
- 0.0 if normal and vector-to-light are perpendicular
- Something in-between if the angle is in-between !
- If less than zero, then normal is pointing away

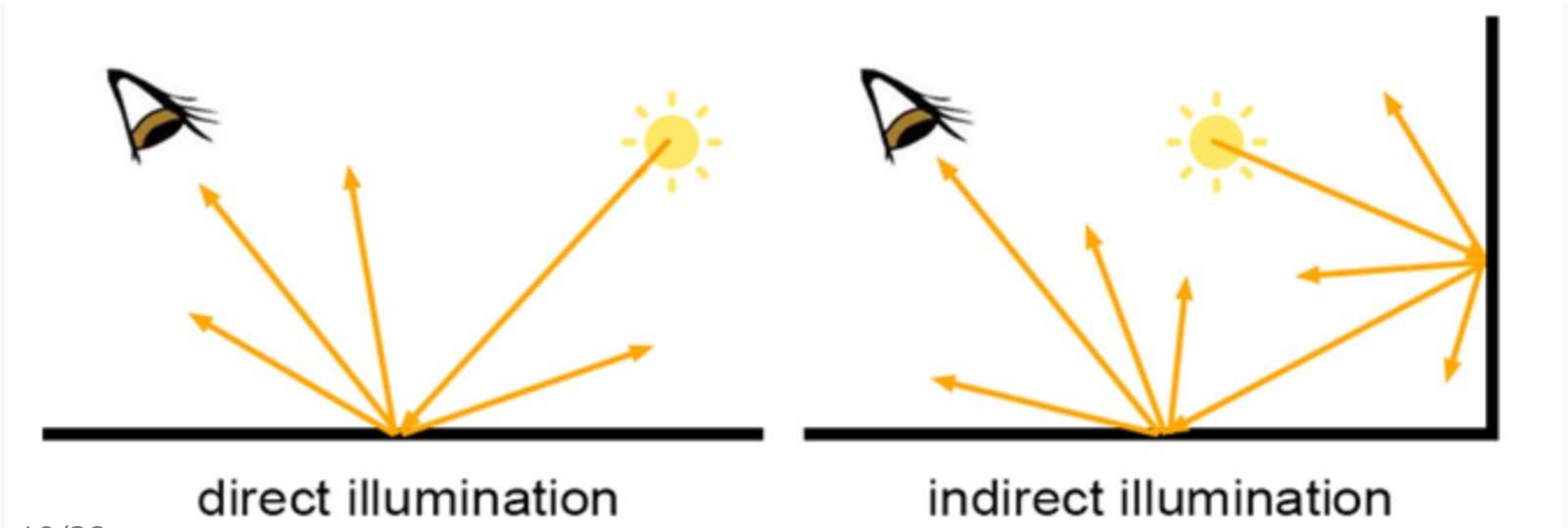


Indirect Illumination

In the real world, light bounces off other objects

Ray bounces many times before reaching a surface

Imagine trying to calculate this !



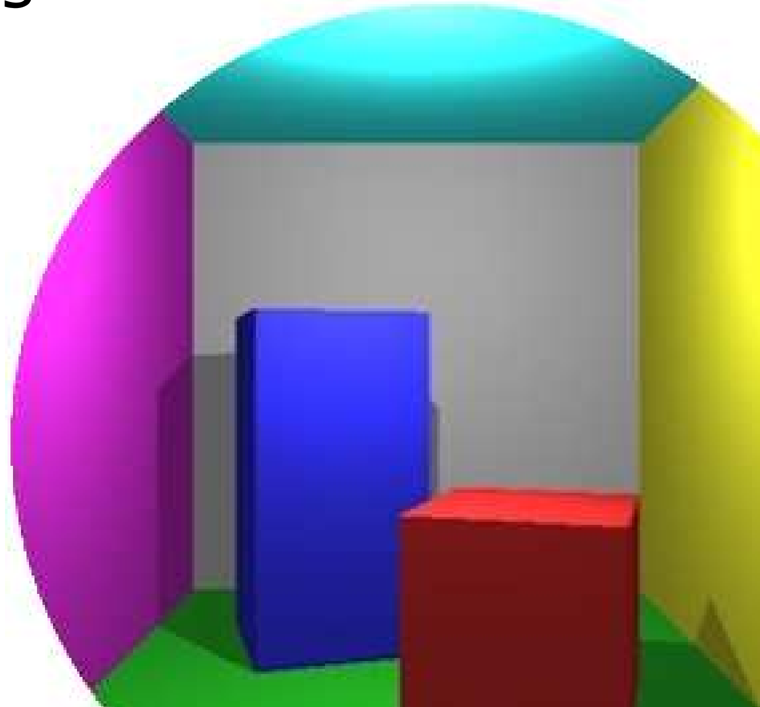
Faking Indirect Illumination

We could try to calculate fully bouncing light
But this can be VERY computationally expensive

"Ambient lighting"

A quick & dirty way to fake this
A "background" level of lighting
Min threshold for all surfaces

"don't let the brightness
drop below 0.2"



Specular Lighting

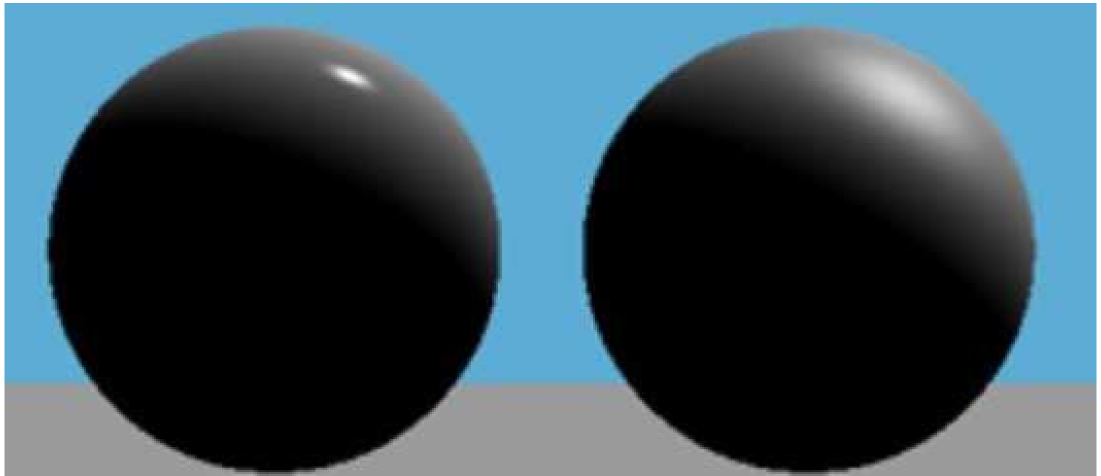
Specular varies depending on nature of material

The left-hand sphere is very glossy ("shiny")

The right-hand sphere is quite matt ("frosted")

Roughness / imperfections on surface scatter light

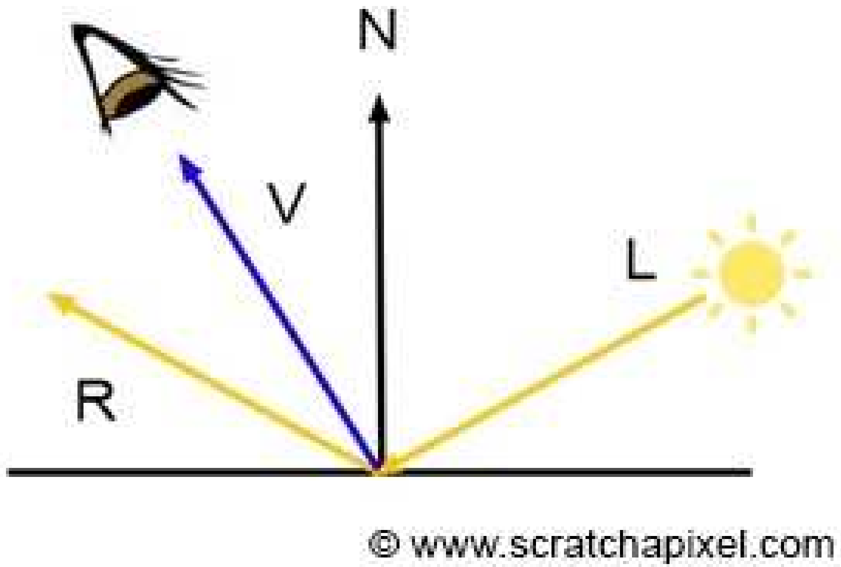
And cause the specular spot to "spread"



Specular Highlight - The Theory

Need to determine how far off the view angle "V" is from the "perfect mirror" light reflection vector "R"

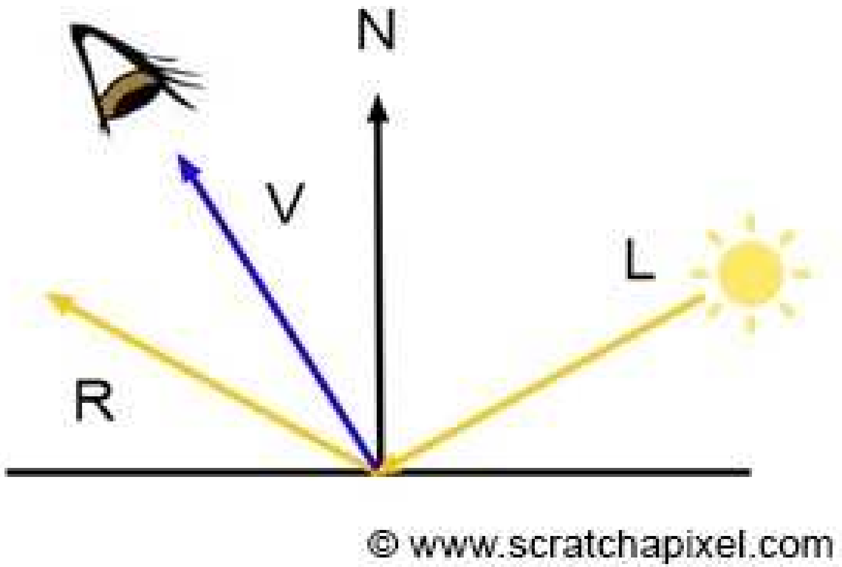
The closer the view angle is to reflection direction, the brighter you paint the pixel !



Specular Highlight - The Maths

To determine the difference in vector directions, we calculate the dot product of "Reflection" & "View"

Result ranges from 1.0 (when directions identical) and reaches 0 when the directions are perpendicular

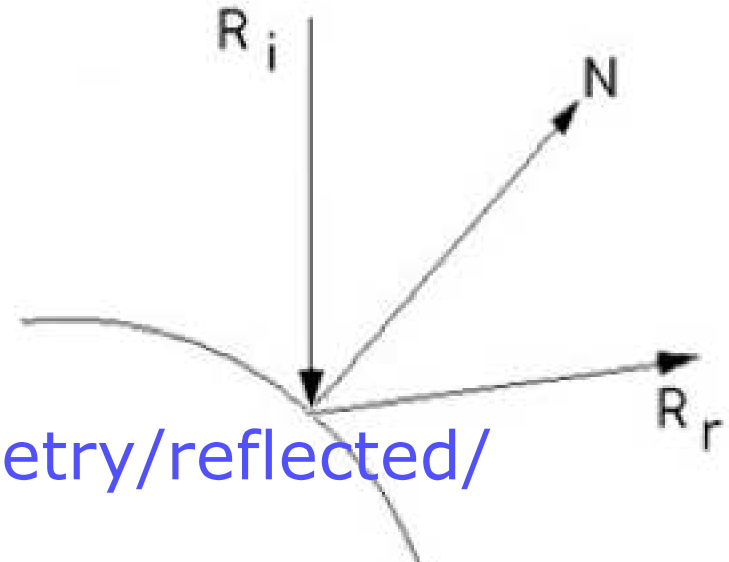


Vector of Reflection

Calculating the vector of Reflection (R_r)
Is slightly more complicated than you might think !

It can be calculated using the following equation:

$$R_r = R_i - 2 N (R_i \cdot N)$$

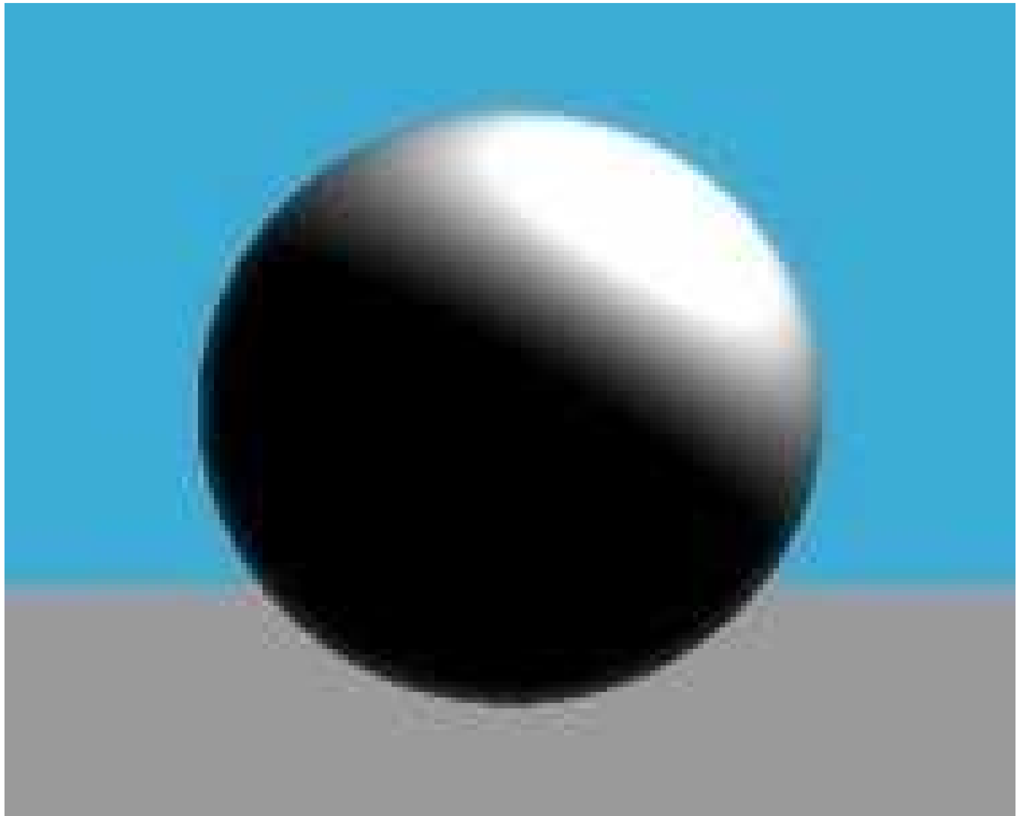


For the derivation, see:
http://paulbourke.net/geometry/reflected/

Applying the Result

We can use the dot products to set pixel brightnesses

Problem is that this results in a very broad spread:



Specular Exponent

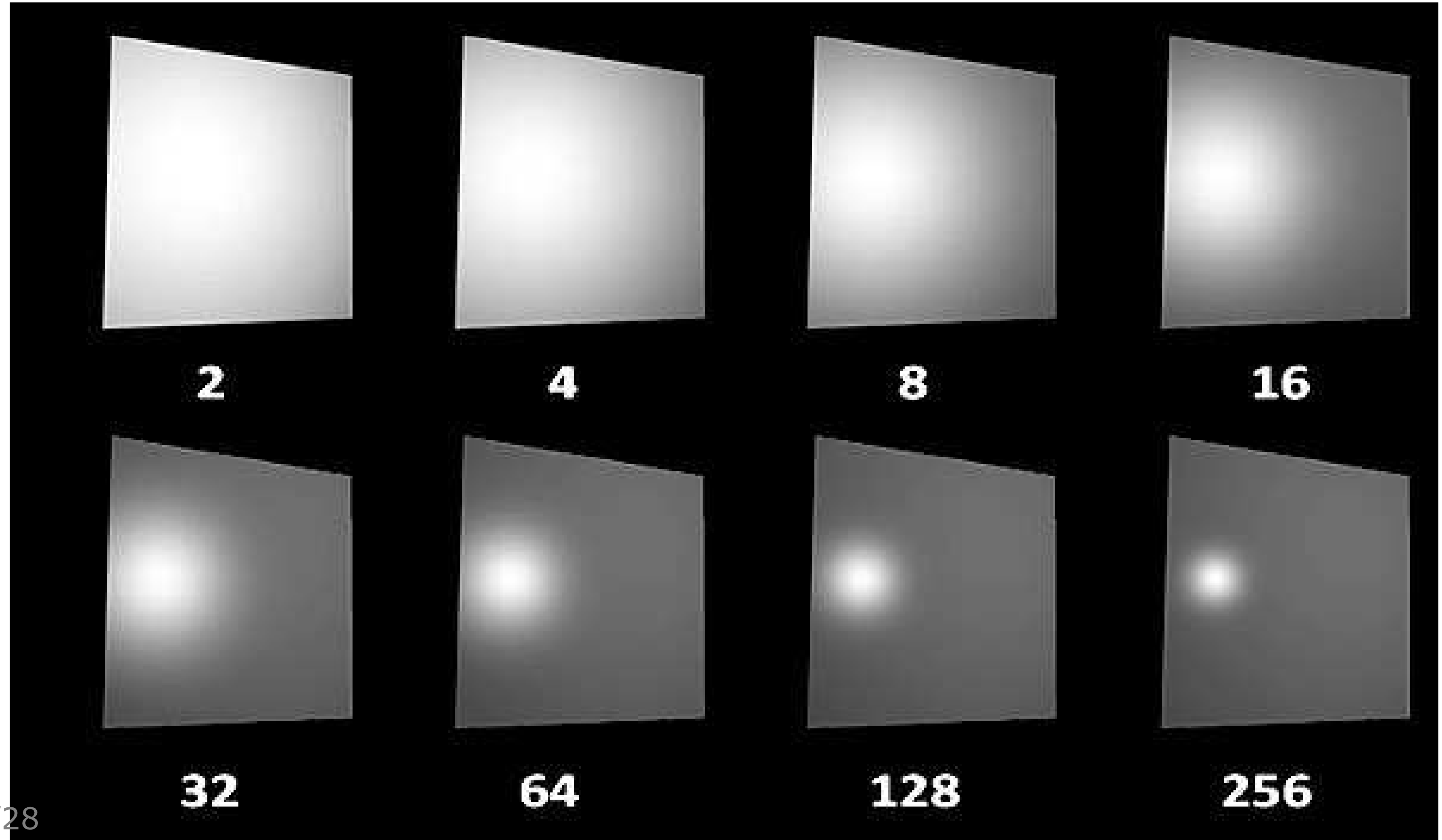
We can adjust spread by raising result to power `n`

$$(V \cdot R)^n$$

The larger the number `n` the tighter the "spot"
(and the glossier/shinier the surface will appear)
Remember that result of dot product is 0.0 to 1.0

This might seem pretty hacky !
Welcome to the world of Computer Graphics ;o)

Specular Exponent Examples

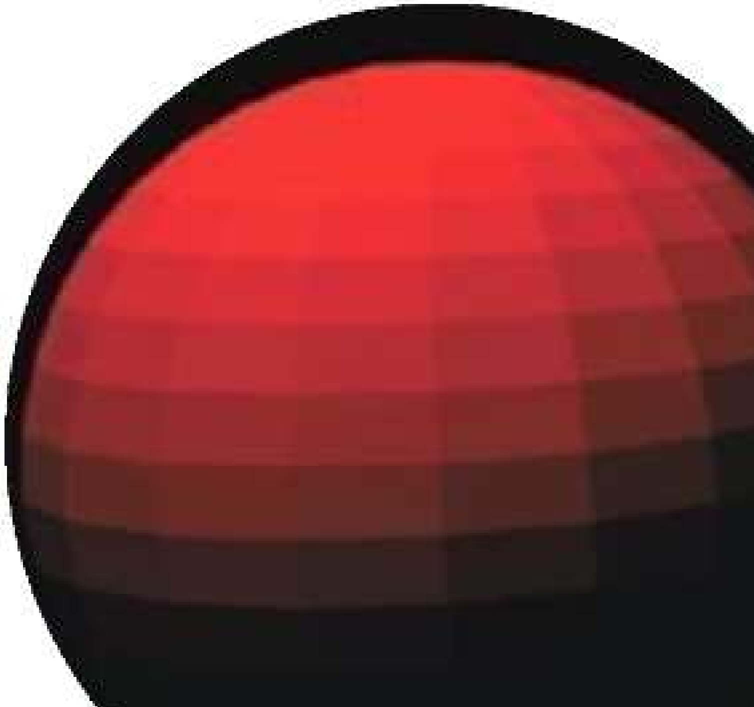


Blocky Model Problem

All our models are composed of a set of triangles
When shading these triangle to illustrate lighting, they are bound to look like flat surfaces
(because that is what they are !)

This can make the model look quite "low-res" and "blocky"

Particularly if there aren't that many triangles !

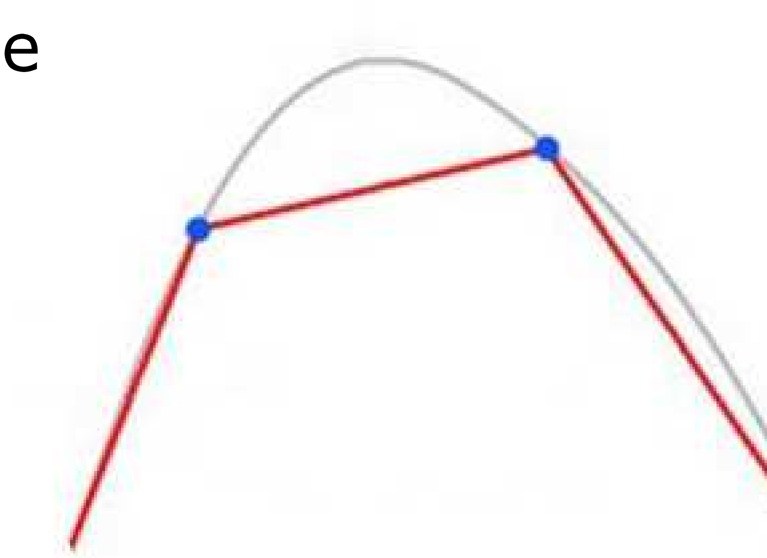


Intelligent Shading ?

Could we shade triangles more "intelligently" ?

Not just looking at each triangle in isolation
But also at the *surrounding* triangles

Detect the "trend" of the surface
Smooth off those sharp corners
"Blend" the triangles together
Avoid those ugly flat faces



Gouraud Shading

Gouraud shading in one such intelligent approach !

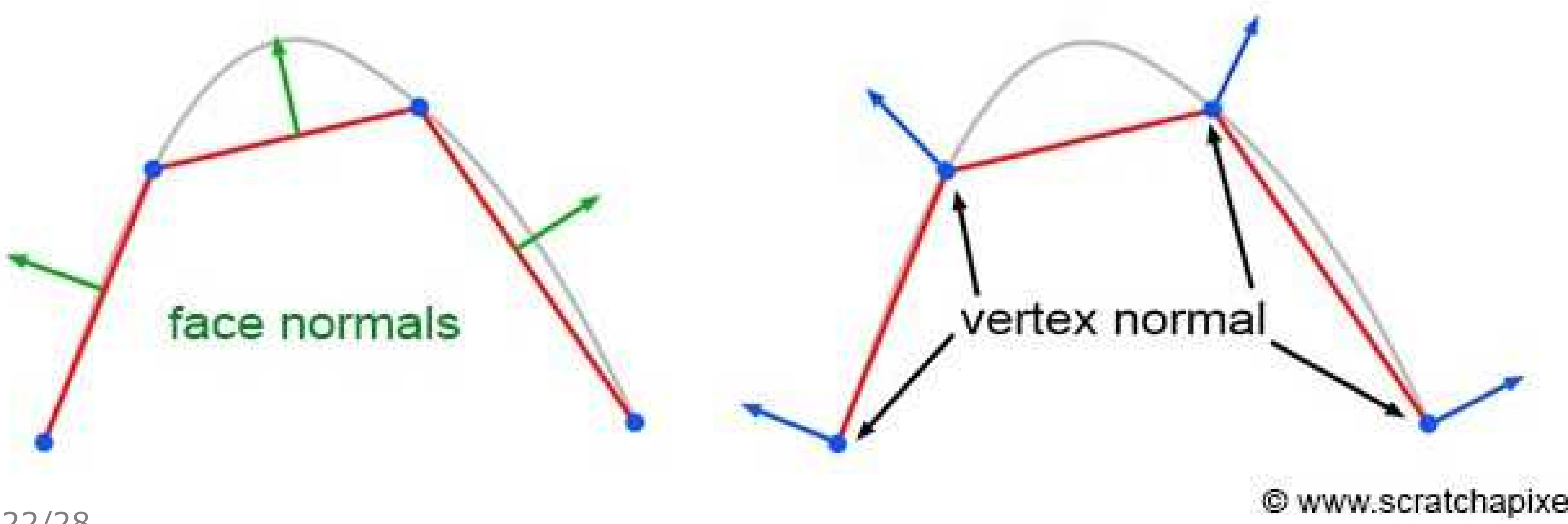
Rather than shading whole triangle with a flat colour
(based on the triangle's single surface normal)

We shade every single pixel on the triangle uniquely
(Depending on where on the triangle the pixel sits)

Our calculation will be based on "Vertex Normals"...

Vertex Normals

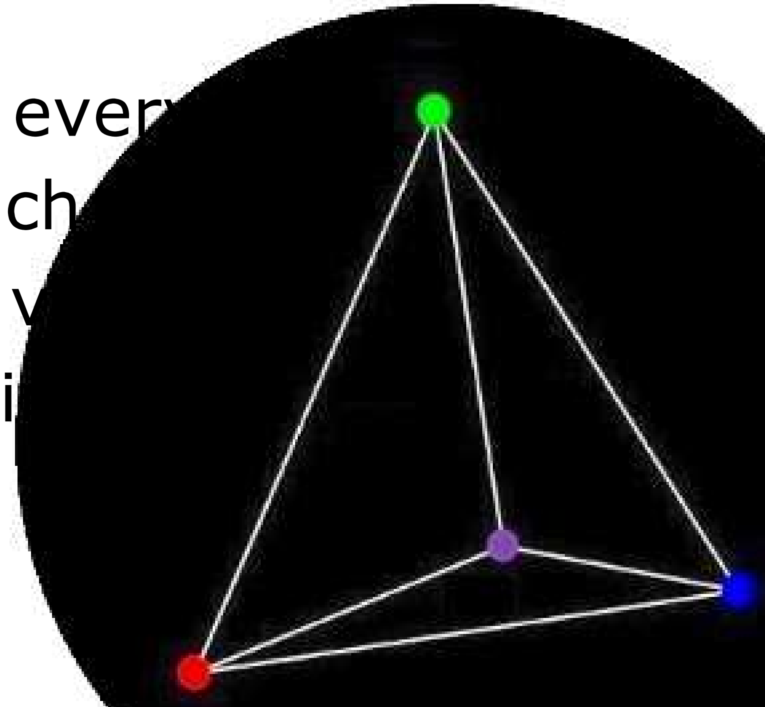
A Vertex Normal is calculated by taking the average of all Face Normals that "involve" that vertex



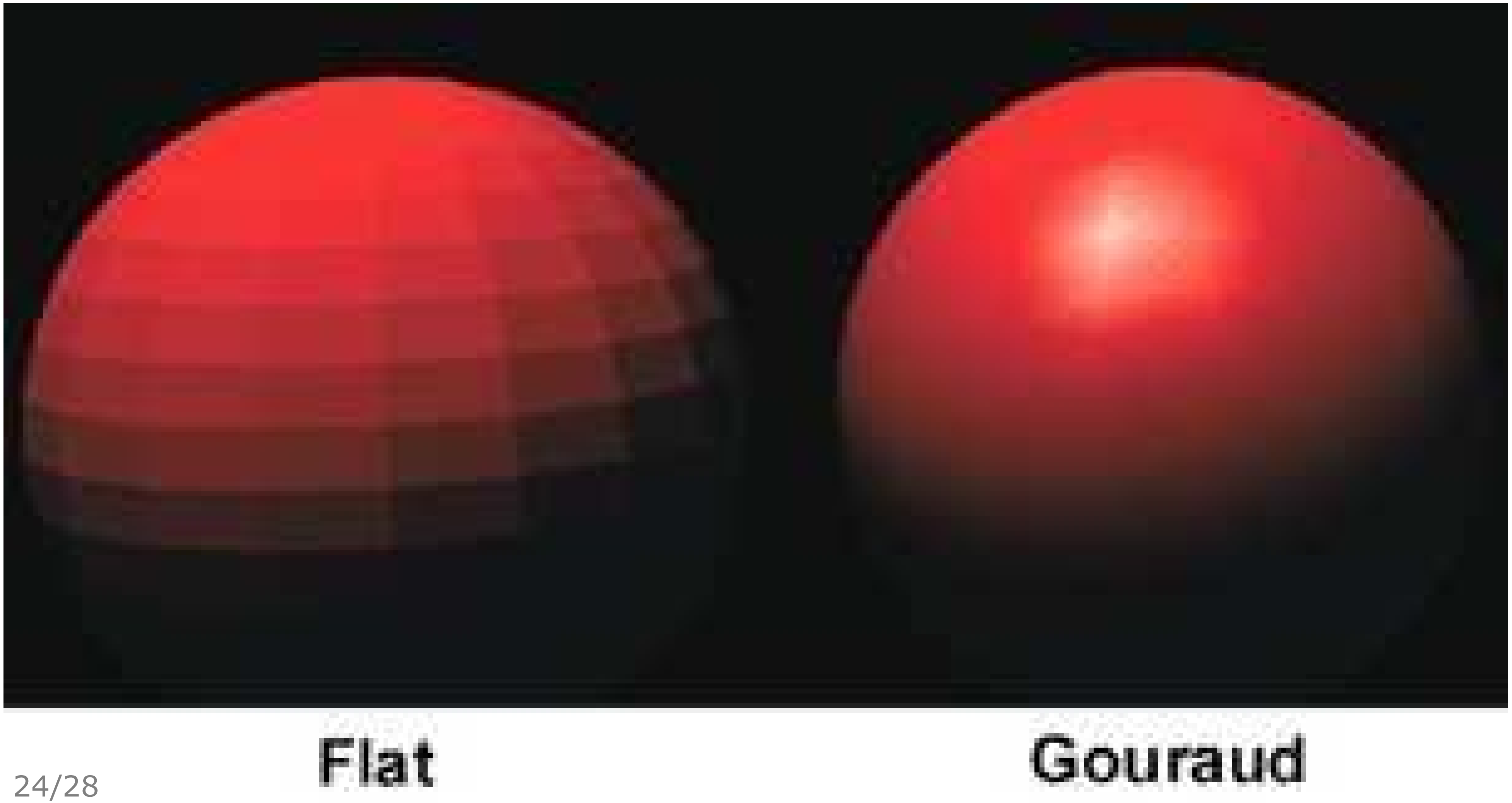
Lighting Calculation

Each triangle now has 3 normals (not just 1) !
Each of which can be used to calculate lighting
(angle-of-incidence, specular highlight etc.)

Calculate unique brightness for ever
Weighted average of light at each
Based on distance/proximity to v
Makes use of Barycentric Coordi
(see animation in workbook !)

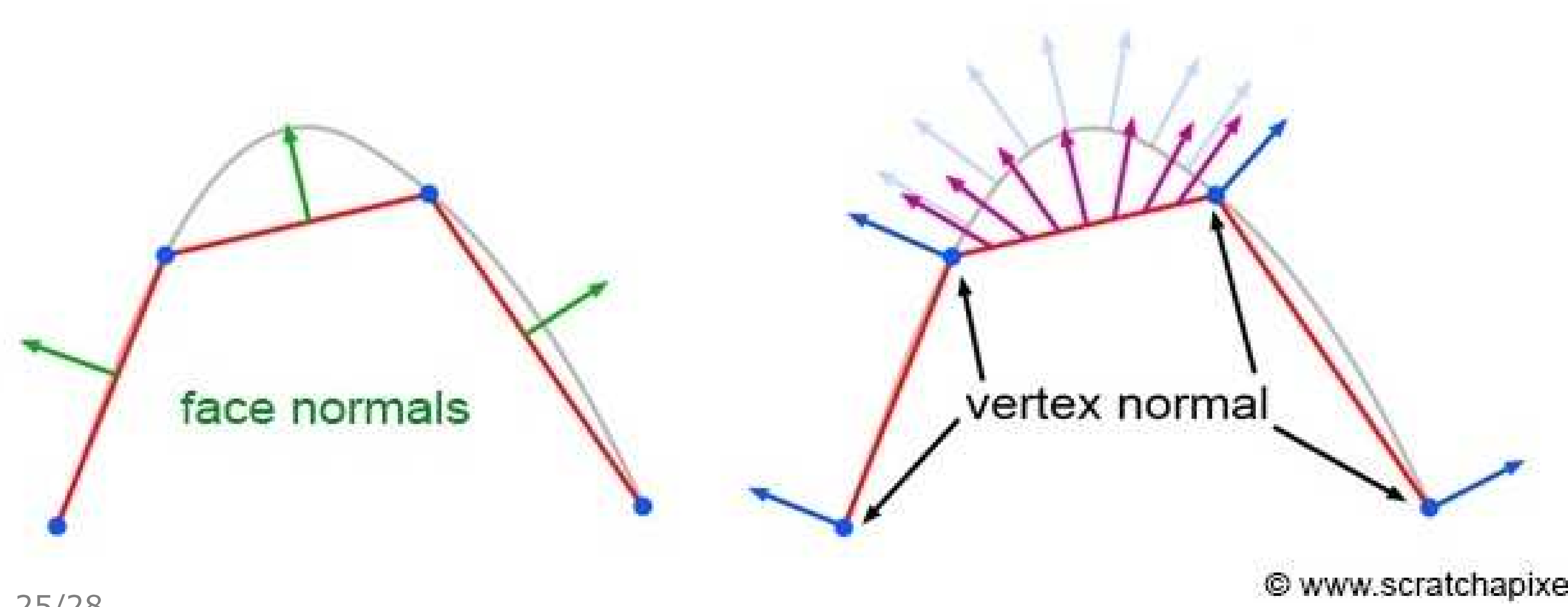


Same Geometry - Different Shading



Phong Shading - Even Smoother !

Interpolate the vertex normals across the surface (again using Barycentric Coordinates)

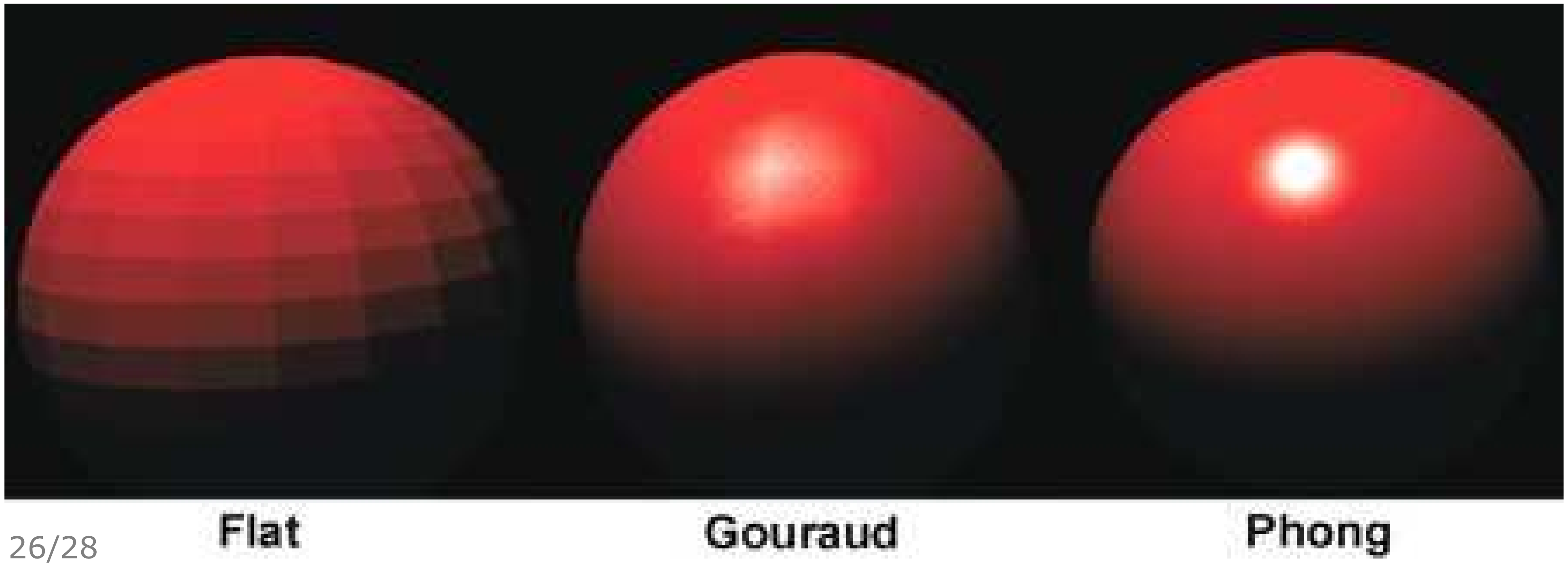


Comparison

Every point on triangle now has a unique normal

Produces much smoother AoI & Specular lighting

(At the expense of rendering speed - naturally)



Questions ?

Extra Feature: Barycentric Animation

Barycentric