centre adscrit a:

UNIVERSITAT DE BARCELONA

Computer Graphics

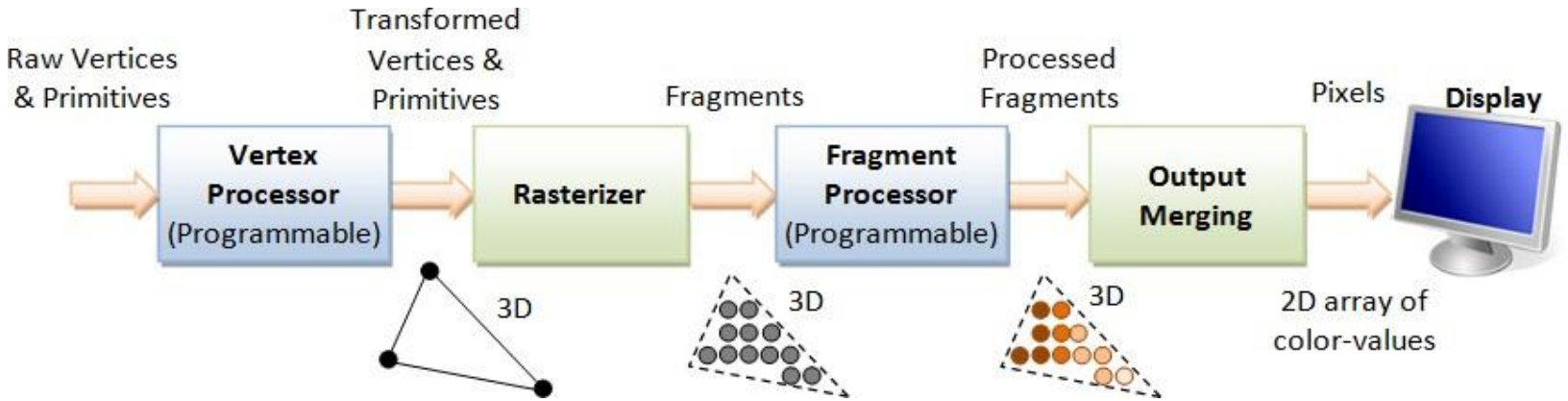
8.Non Realistic Rendering – Part 1

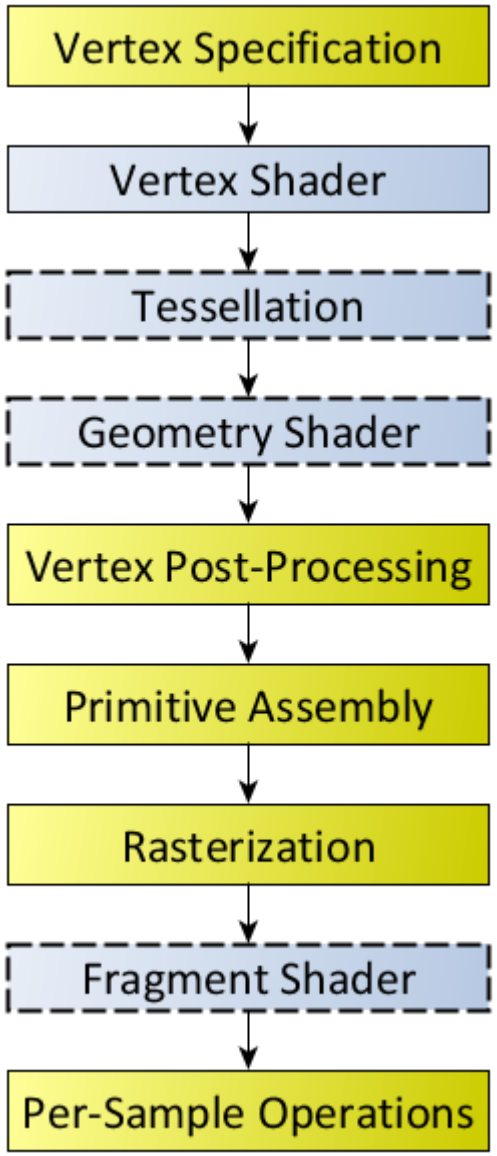
Dr Joan Llobera – joanllobera@enti.cat
Dr Jesus Ojeda – jesusojeda@enti.cat

Outline

1. The graphics pipeline
2. Toon shaders examples and exercises
3. How to render the silhouette?

1. Reminder of the graphics pipeline

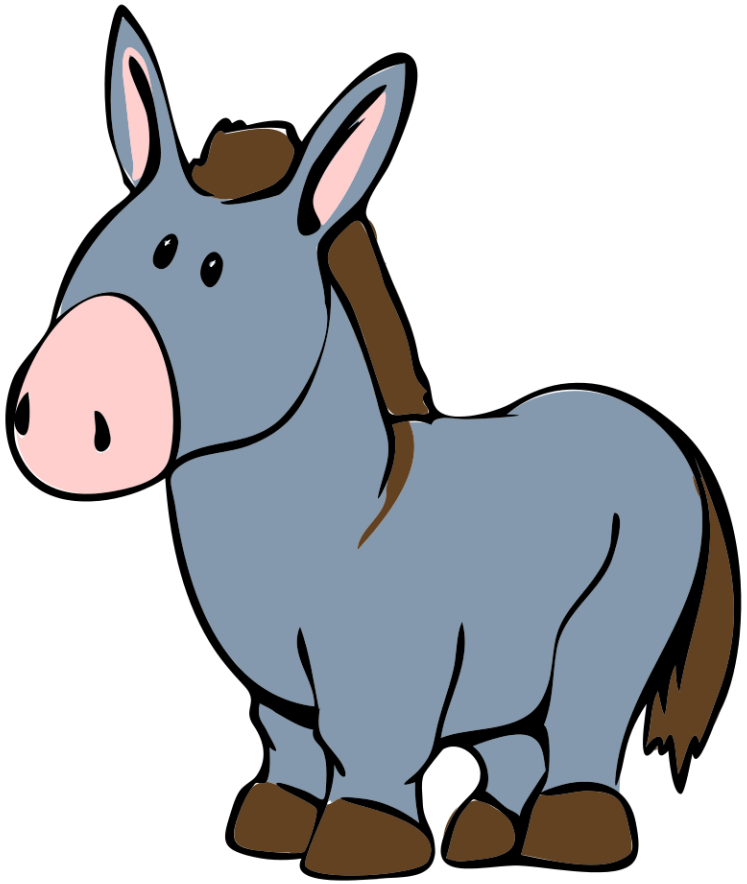


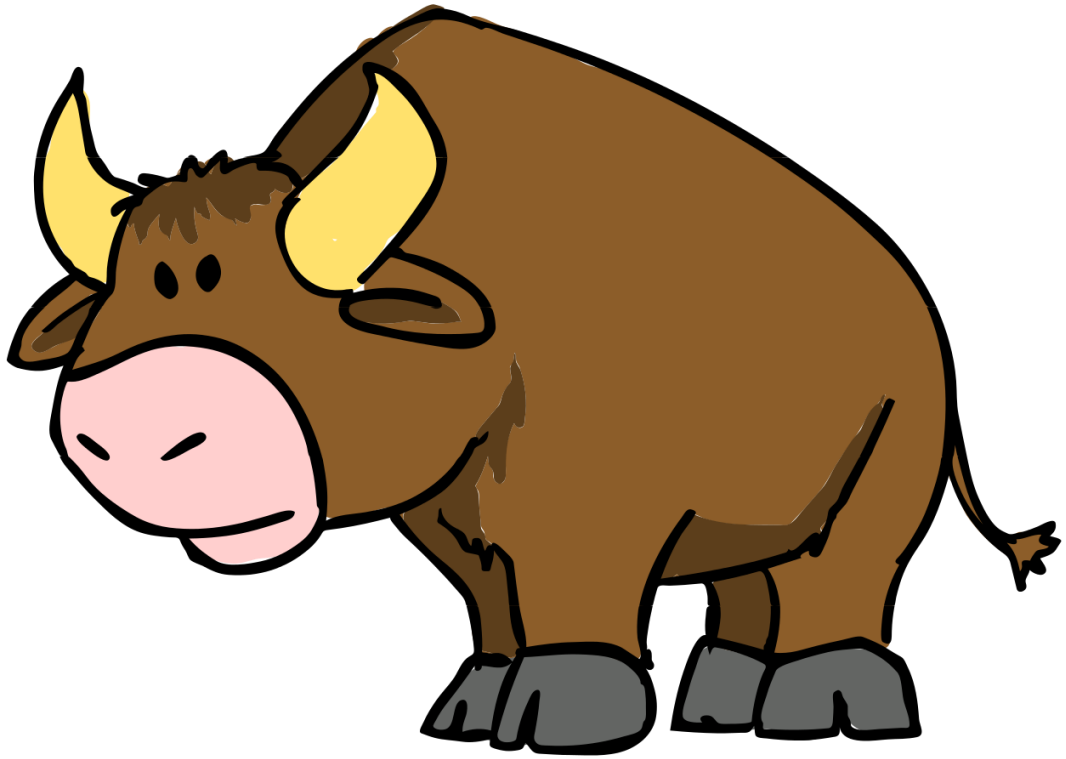


https://gamedevz.wordpress.com/2016/02/25/outline-toon-shader/

https://gist.github.com/xDavidLeon

Analysis of toon shading elements

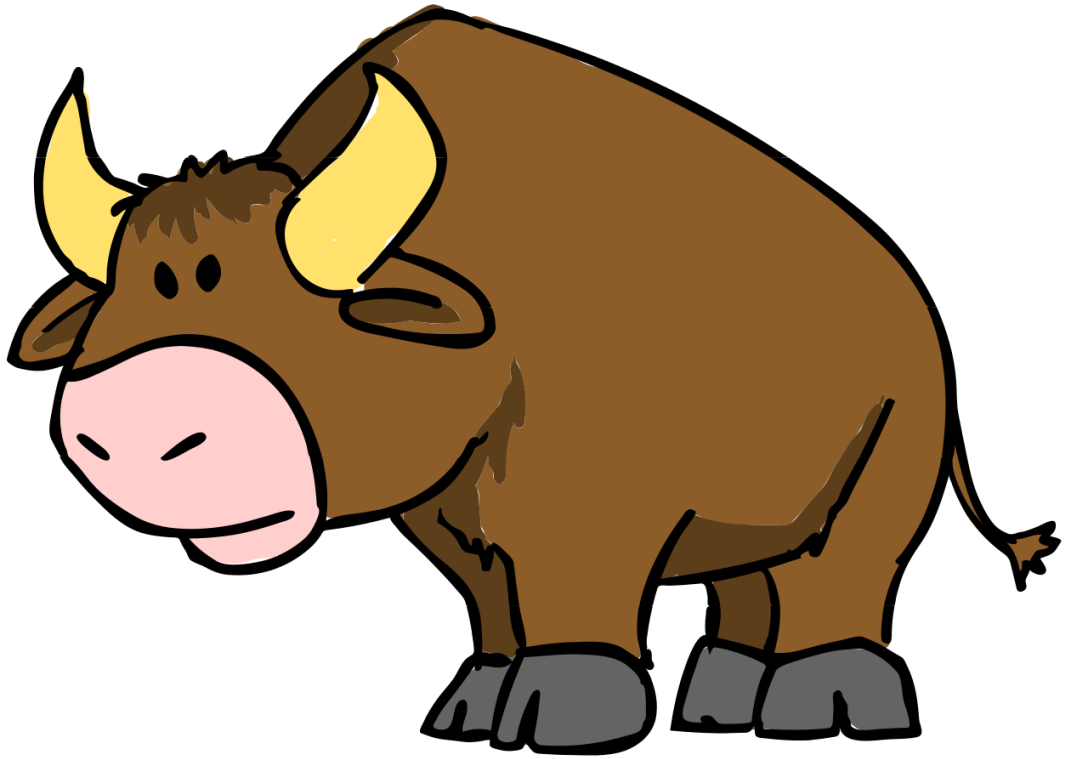




https://en.wikibooks.org/wiki/GLSL_Programming/Unity/Toon_Shading

Analysis of toon shading elements

- Plain colours, with abrupt transitions
- Shape outline



https://en.wikibooks.org/wiki/GLSL_Programming/Unity/Toon_Shading

Analysis of toon shading elements

- Plain colours, with abrupt transitions
- Shape outline (not always black)
- Reflections, if present, are flat

https://en.wikibooks.org/wiki/GLSL_Programming/Unity/Toon_Shading

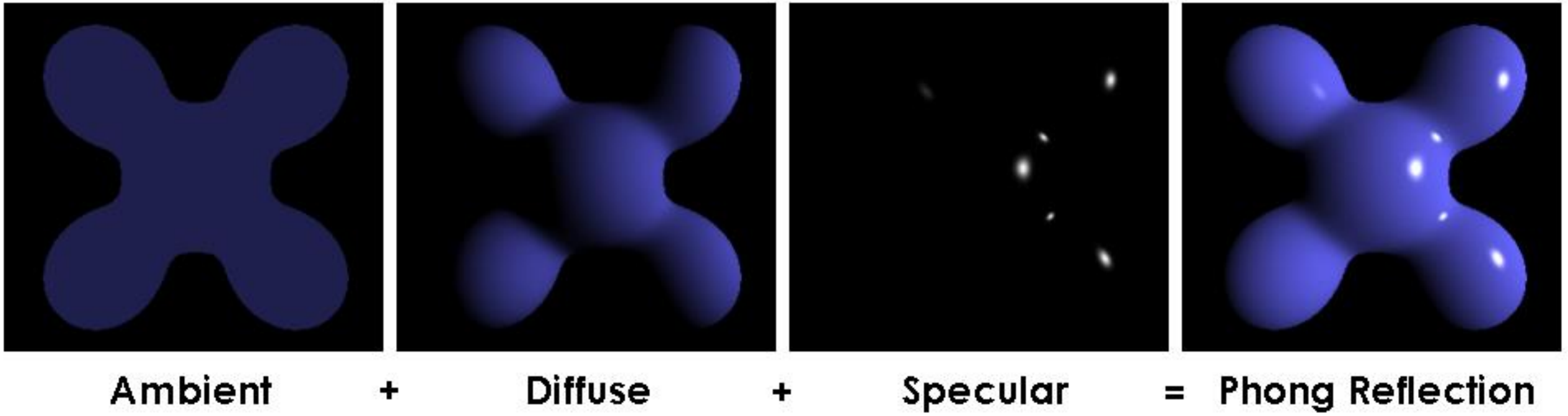
How to implement it?

- Modify the phong shading model for main textures and reflections
- Add “something else” for shape outline. We will need to discuss further line rendering

https://en.wikibooks.org/wiki/GLSL_Programming/Unity/Toon_Shading

Reminder: The Phong reflection model

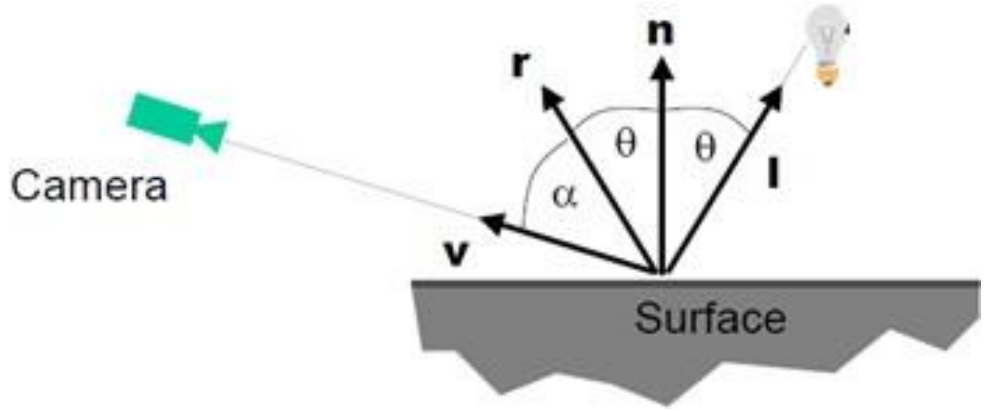
- The **Phong reflection model** (also called Phong illumination or Phong lighting) is an empirical model of the local illumination of points on a surface
- It describes the way a **surface reflects light** as a combination of the **diffuse reflection** of **rough surfaces** with the **specular reflection of shiny surfaces**
- The model also includes an **ambient** term to account for the small amount of light that is scattered about the entire scene.

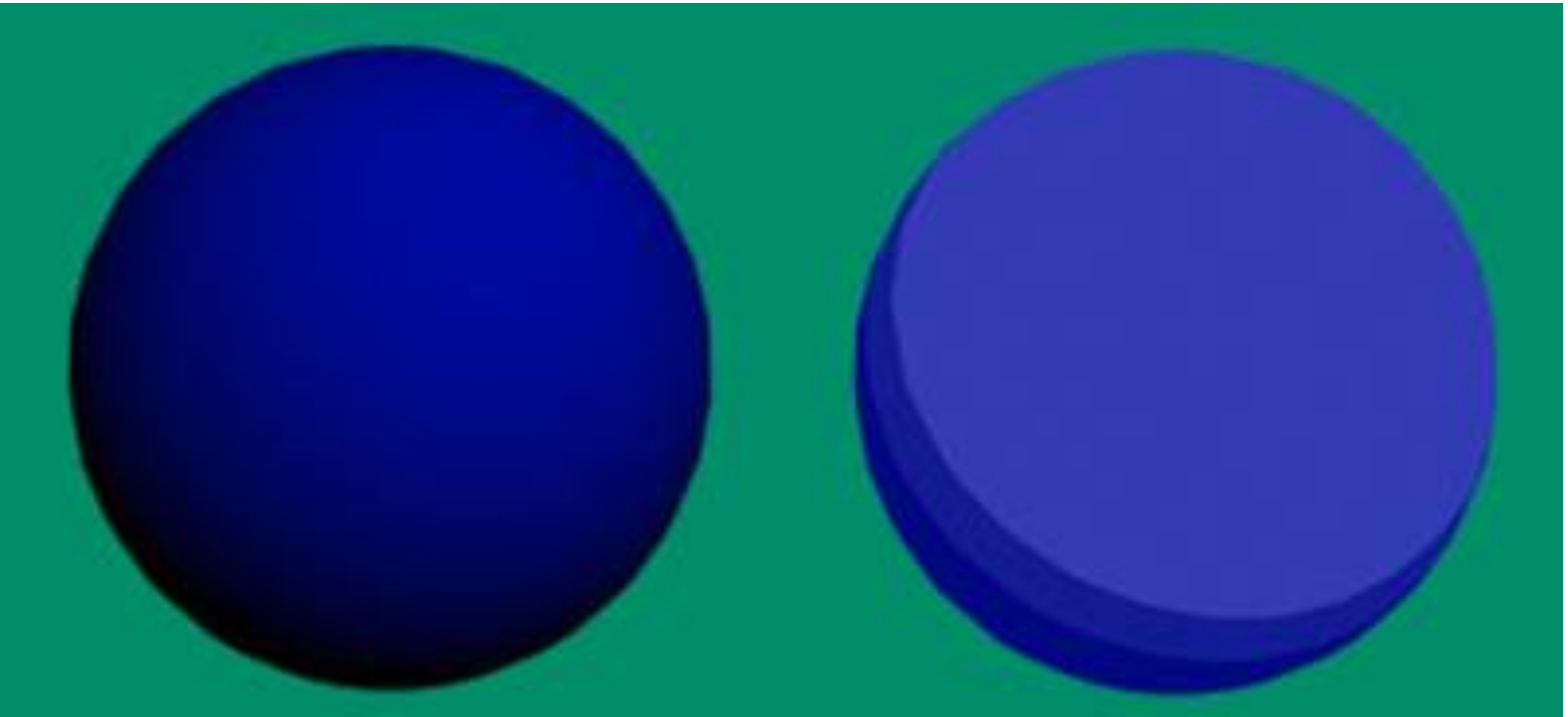


Phong shading & Toon Shading

$$L(\omega_r) = k_d(\mathbf{n}\cdot\mathbf{l})\frac{\Phi_s}{4\pi d^2}$$

- The **Phong reflection model** is the result of three partial radiance:
  - **The diffuse reflection**, where the reflection occurs to all directions
  - **The specular reflection**, where the reflection only occurs in the mirror angle
  - **The ambient light**, which represents the indirect light as a constant





To build Toon Shade from Phong Shade we want to:

1. **Have no specular component**
2. **Quantify he diffuse component**

U = L.N

if U < 0.2 then U = 0

if U >= 0.2 and U < 0.4 then U = 0.2

if U >= 0.4 and U < 0.5 then U = 0.4

if U >=0.5 then U = 1

Review of Fragment Shader

```
const char* cube_fragShader =
"#version 330\n\
in vec4 vert_Normal;\n\
out vec4 out_Color;\n\
uniform mat4 mv_Mat;\n\
uniform vec4 color;\n\
void main() {\n\
out_Color = vec4(

color.xyz * dot(vert_Normal, mv_Mat*vec4(0.0, 1.0, 0.0, 0.0))
+ color.xyz * 0.3, 1.0 );}";
```

Exercises:

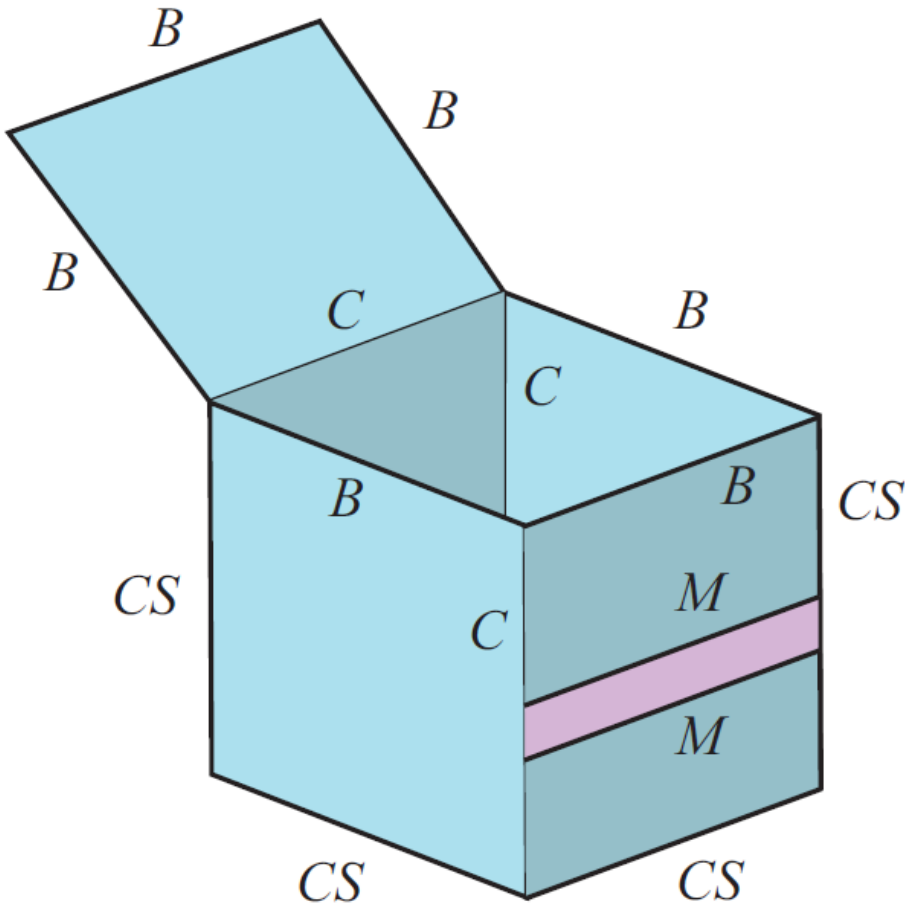
1. Load the 3D character of your delivery one (or a 3D character, if you did not do delivery one).
2. Modify the existing shader to implement only the diffuse component of the Phong Model (draw an object to be the light source, make it move)

    Steps

    1. Set a position of the light source, and paint an object there (for example, the cube primitive)
    2. Adapt the default cube shader to take as input the light position and calculate a diffuse lightning from it
    3. Move the light source and see how the diffuse shader changes (and it does not change with the camera movements)

3. Replace the diffuse component with a Toon shader
4. Add the specular component to the shader
5. Modify the shader in order a uniform can control the presence of the specular component
6. Modify the specular component in order to render the reflection as a flat model
7. Make the reflection component apply only to eyes

Line Rendering



Line Rendering

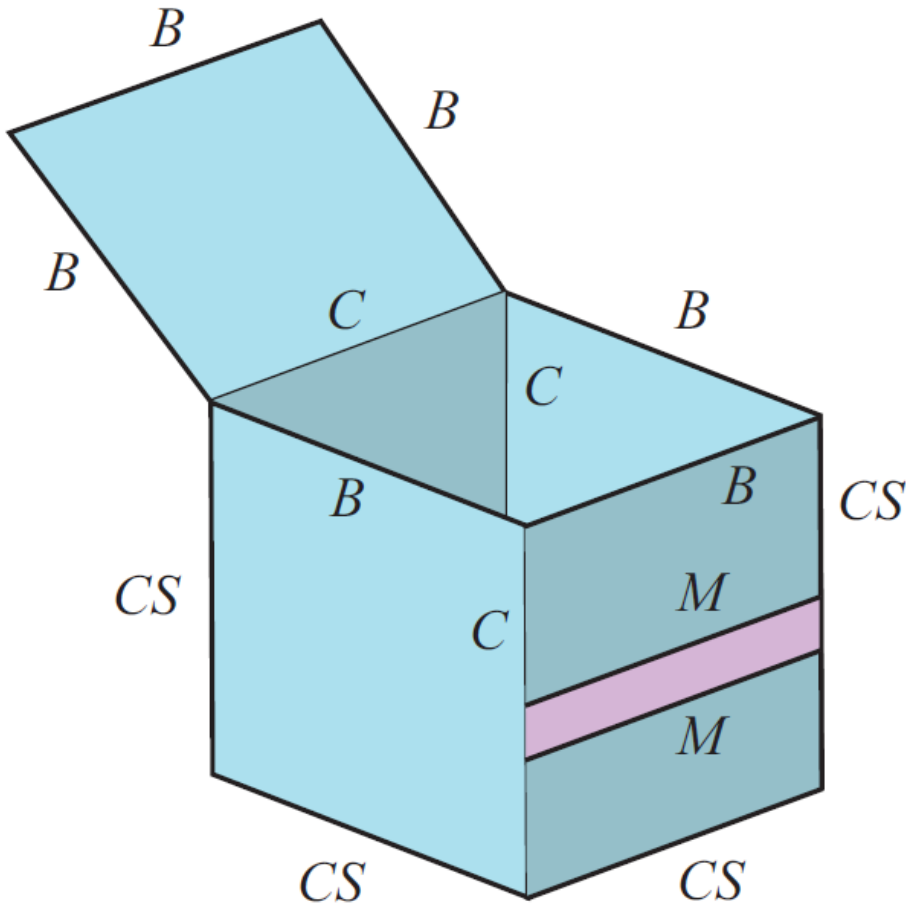
We have:

B: boundary

C: crease

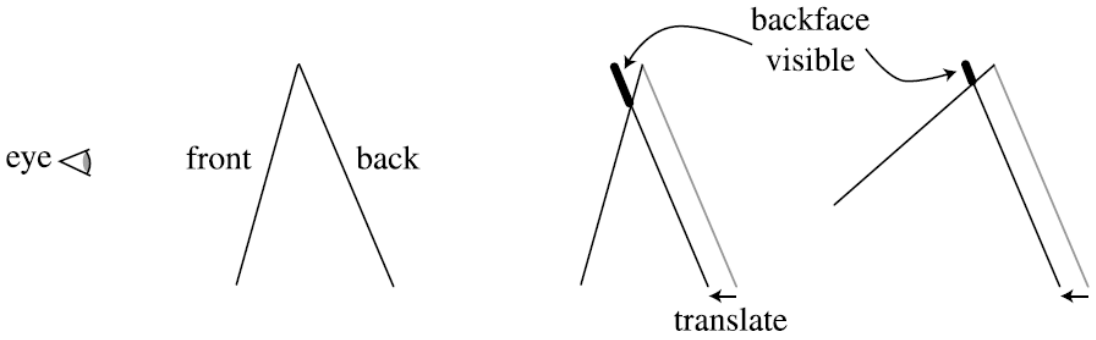
M: material

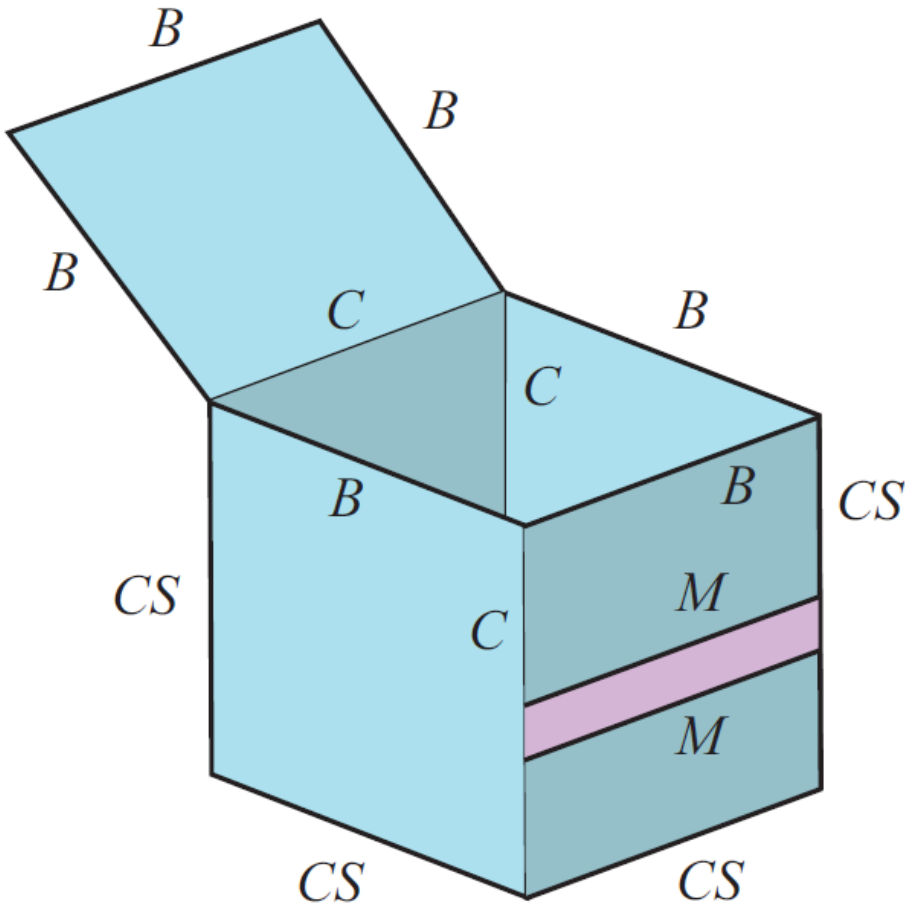
S: silhouette



Line Rendering – geometry approach

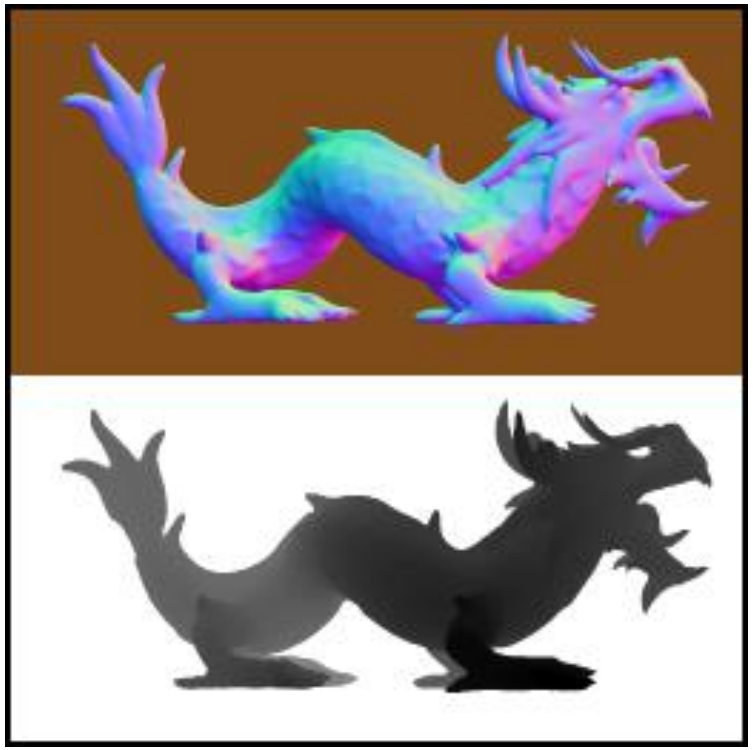
- Move the geometry to get a shape, then paint it in one flat colour.
  - How do you decide which geometry you need to move?
  - In the drawing on the right, which lines would be drawn?





Line Rendering – use normals or depth?

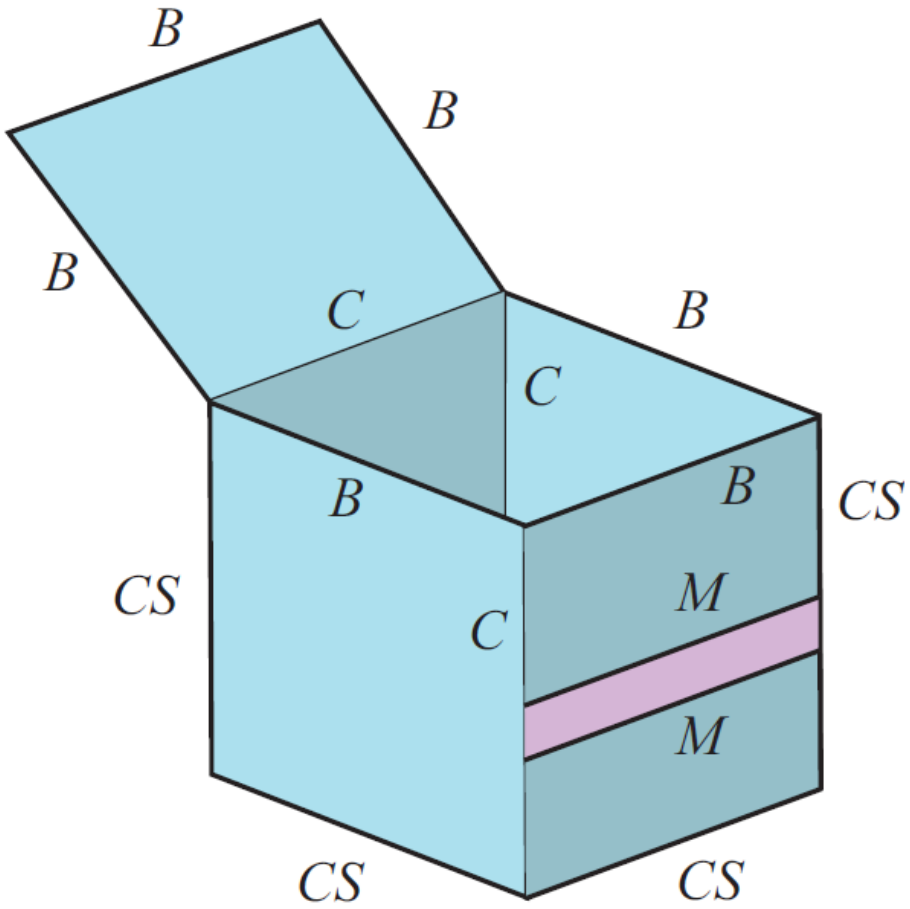
- Image 1 shows an example of 3D model rendered with contour rendering
- We can look into the change in the normal direction (see image 2)
  - Where do we find the normal of an image?
- We can look into big variations of depth (see image 3)
  - Where do we find the depth of an image?



Source: https://prideout.net/blog/old/blog/index.html@p=54.html

Line Rendering – use normals or depth?

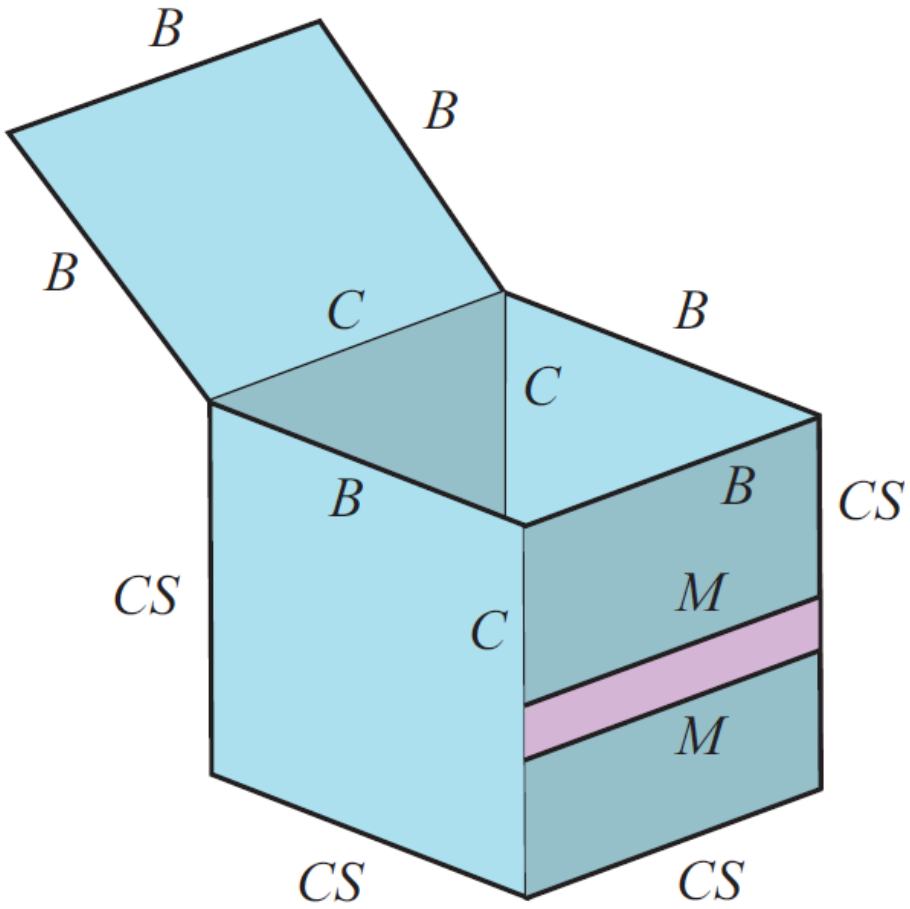
- What will change in terms of rendering pipeline, using either depth or normals?
- What lines of the right figure will be drawn with either method?
- Can you come up with a different method that would render some of the lines in the left that won't be drawn with normals nor with depth?



Source: https://prideout.net/blog/old/blog/index.html@p=54.html

Line Rendering – a first implementation

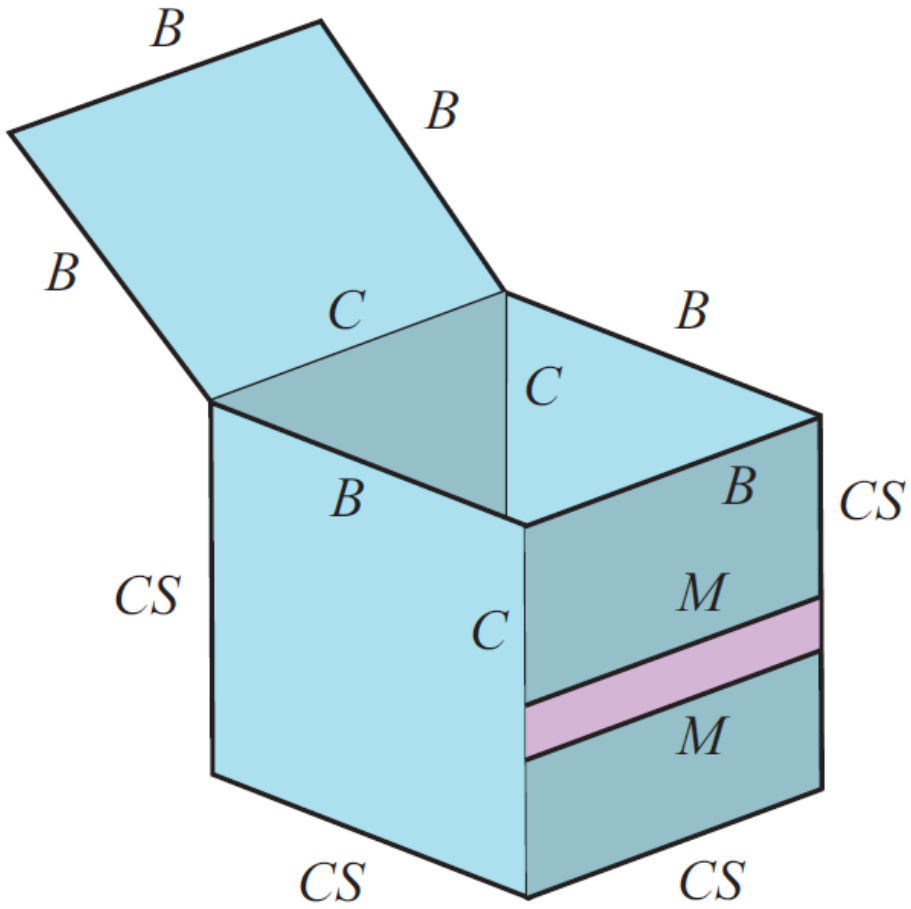
- What lines of the right figure will be drawn with either method?
- Can you come up with a different method that would render some of the lines in the left that won't be drawn with normals nor with depth?



Source: https://prideout.net/blog/old/blog/index.html@p=54.html

Line Rendering – use normals or depth?

- What lines of the right figure will be drawn with either method?
- Can you come up with a different method that would render some of the lines in the left that won't be drawn with normals nor with depth?



Source: https://prideout.net/blog/old/blog/index.html@p=54.html

Resources

See online references indicated in the previous slides

Also:

- [Kessenich] Kessenich et al. OpenGL Programming Guide. Chapter 7. Light and Shadow
- [Akenine-Möller] Akenine-Möller et al. Real-Time Rendering. Third Edition, CRC Press (chapter 11)
- https://learnopengl.com/Lighting/Basic-Lighting
- https://en.wikipedia.org/wiki/Phong_reflection_model