



# COMPUTER GRAPHICS



LEI / LCC  
DEPARTAMENTO DE INFORMÁTICA  
UNIVERSIDADE DO MINHO

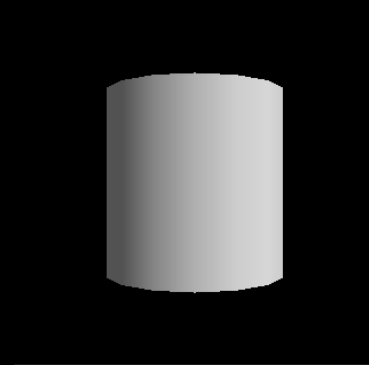
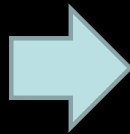
## Lighting

*Lights, Materials and Normals*



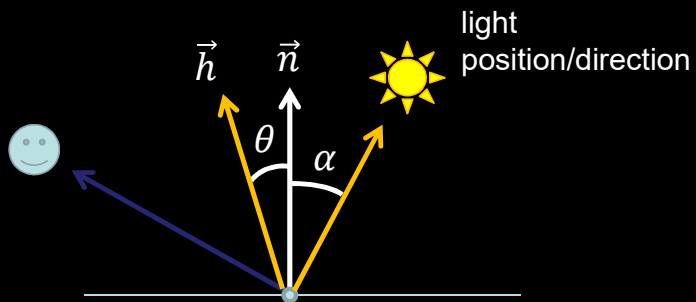
## Goal

- To get a lit cylinder





## Lighting: a quick refresh



Needed:

- Setup light
- Define material colors
- Add normals to vertices

diffuse

$$I_d = K_d \times L_d \times \cos(\alpha)$$

material colors

light colors

specular

$$I_s = K_s \times L_s \times \cos(\theta)^s$$

material shininess



# Approach

- To lit the cylinder:
  - Setup a light source:
    - Initialization:
      - Turn on lighting
      - Define light color
    - Render:
      - Define position of the light
  - Define a material for the cylinder <- Render
  - Add normals to the cylinder from script 4
    - Initialization:
      - Create an array with the normal vectors for each vertex
      - Create a VBO and copy data to GPU
    - Render:
      - bind, define semantics, draw



## Setup a light source

- Turn on lighting

- Done once in initialization

```
glEnable(GL_LIGHTING);  
glEnable(GL_LIGHT0);
```

Turn lighting on.  
Without this nothing  
will be lit.

Using a single light source. If  
more light sources are required it  
is necessary to enable them



## Setup a light source

- Define light color
  - Done once in initialization

```
float dark[4] = {0.2, 0.2, 0.2, 1.0};  
float white[4] = {1.0, 1.0, 1.0, 1.0};  
float black[4] = {0.0f, 0.0f, 0.0f, 0.0f};
```

```
// light colors  
glLightfv(GL_LIGHT0, GL_AMBIENT, dark);  
glLightfv(GL_LIGHT0, GL_DIFFUSE, white);  
glLightfv(GL_LIGHT0, GL_SPECULAR, white);
```

```
// controls global ambient light  
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, black);
```

for default values check:  
<https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glLight.xml>



## Setup a light source

- Define light Position/Direction

- In render function

```
float pos[4] = {1.0, 1.0, 1.0, 0.0};  
glLightfv(GL_LIGHT0, GL_POSITION, pos);
```

Last component  
defines if this is a point  
or a vector.

Using a vector produces a directional light, a  
point will provide a point light or a spotlight

- The light position/direction is affected by geometrical transformations, hence it needs to be set every frame.



## Define a material for the cylinder

- Materials are like colors, but more configurable
- `float dark[] = { 0.2, 0.2, 0.2, 1.0 };`
- `float white[] = { 0.8, 0.8, 0.8, 1.0 };`
- `float red[] = { 0.8, 0.2, 0.2, 1.0 };`
- - `glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, red);`
  - `glMaterialfv(GL_FRONT, GL_SPECULAR, white);`
  - `glMaterialf(GL_FRONT, GL_SHININESS, 128);`
- Materials should be set every frame, before drawing the object

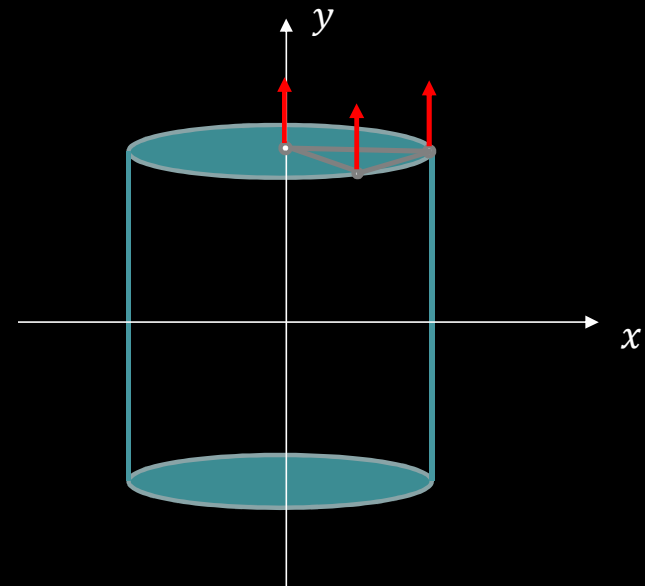
Note: setting the same color for ambient and diffuse because light's ambient color is already dark (see slide 6)





## Cylinder Normals

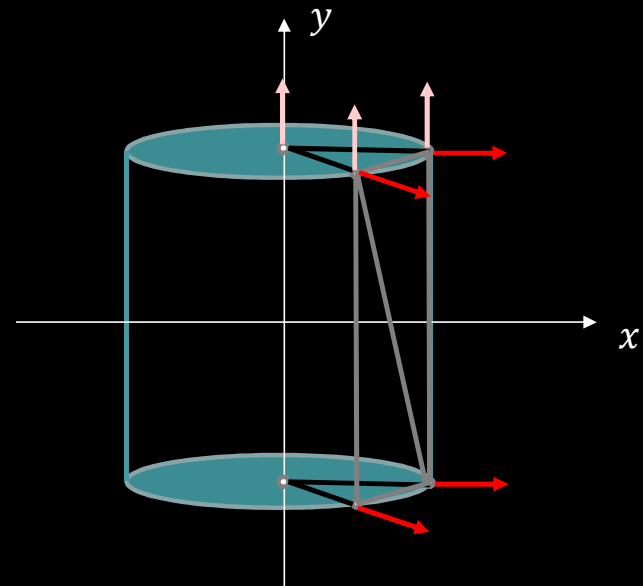
- A normal is a unit length vector perpendicular to the surface
- The top lid vertices have a normal  $\uparrow$  pointing upwards  $(0,1,0)$
- The bottom lid vertices have a symmetrical normal





## Cylinder Normals

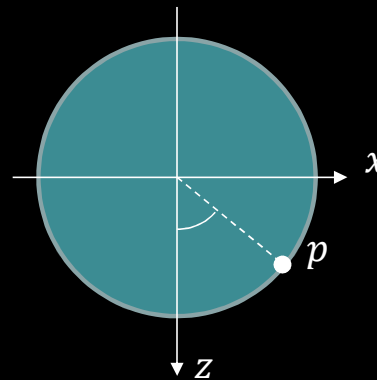
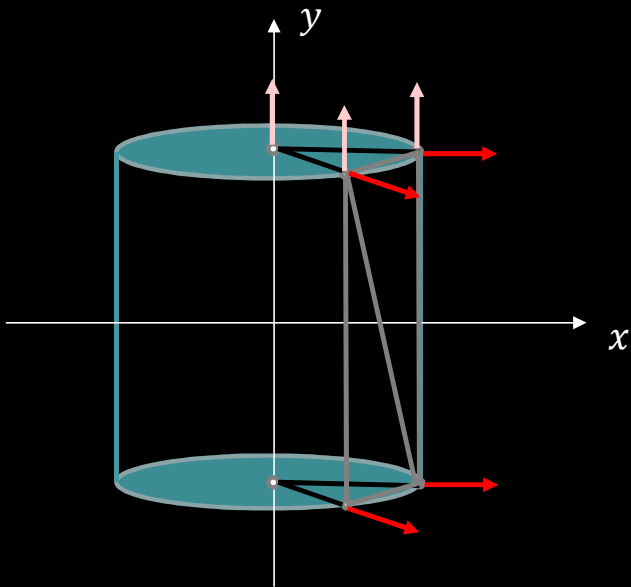
- Consider the triangles of the face of the cylinder
- These vertices have horizontal → normals pointing outwards
- Notice that we are not trying to get the normals of the triangular surface. Instead we want the normals of the underlying surface, the cylinder





## Cylinder Normals

- Since the normals are horizontal the  $y$  coordinate is zero



Consider a vertex on the lid.  
How to compute its coordinates?

Polar coordinates

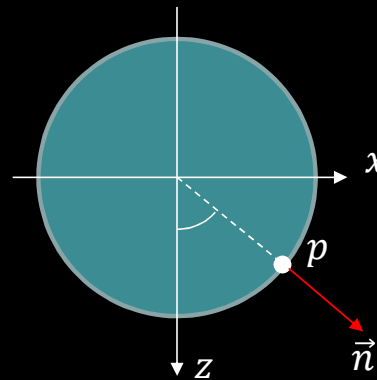
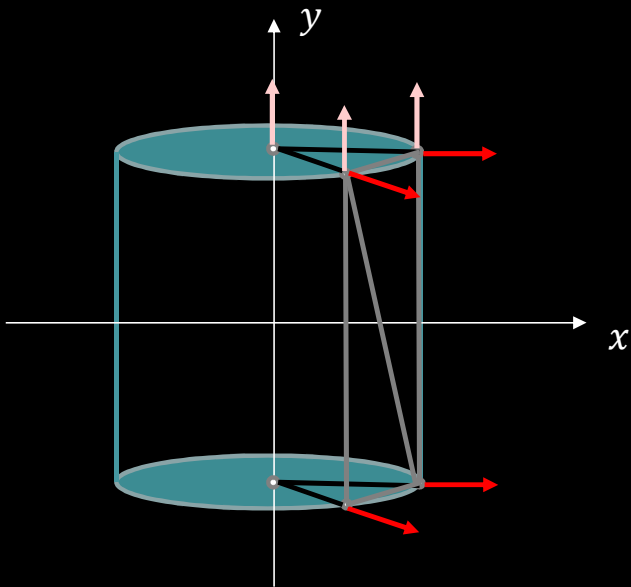
$$x = r \sin(\alpha)$$

$$z = r \cos(\alpha)$$



## Cylinder Normals

- Since the normals are horizontal the  $y$  coordinate is zero



Polar coordinates

$$x = r \sin(\alpha)$$

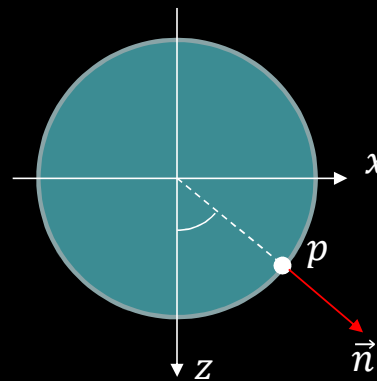
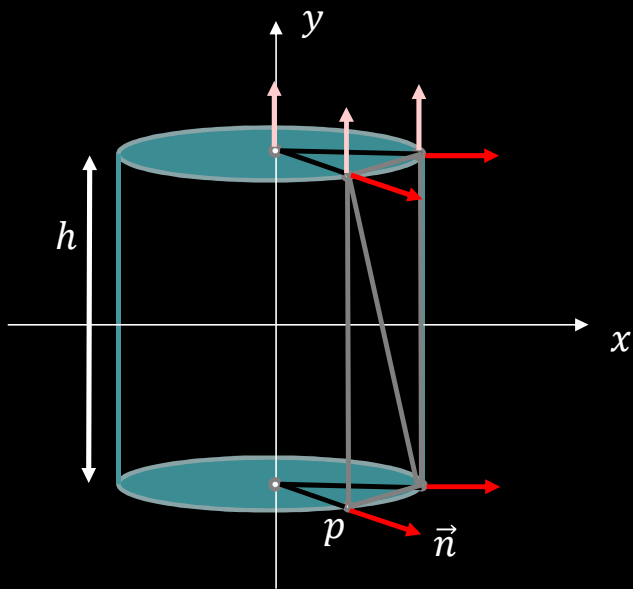
$$z = r \cos(\alpha)$$

Notice that the normal has the same direction than the vector from the center of the lid to the vertex.



## Cylinder Normals

- Since the normals are horizontal the  $y$  coordinate is zero



Polar coordinates

$$x = r \sin(\alpha)$$

$$z = r \cos(\alpha)$$

Therefore, if

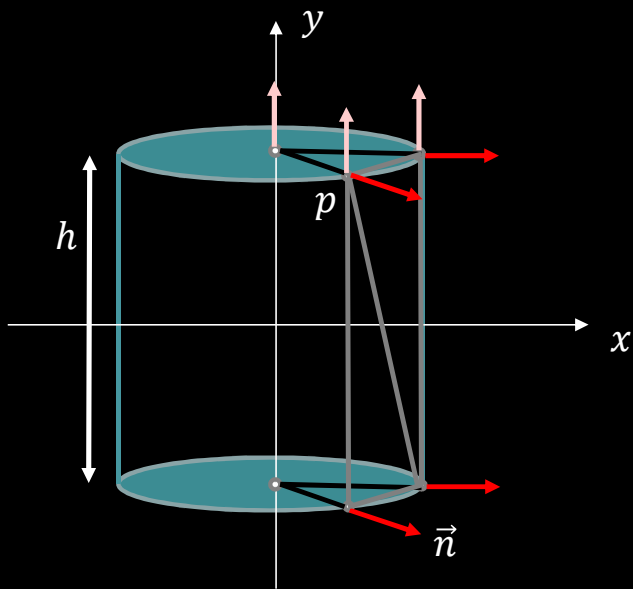
$$p = \left( r \sin(\alpha), \frac{h}{2}, r \cos(\alpha) \right)$$

then

$$\vec{n} = (\sin(\alpha), 0, \cos(\alpha))$$



## Cylinder Normals



- Note: position  $p$  gives rise to two distinct vertices. One from the bottom lid, and one from the face of the cylinder.
- Vertices are distinct if one of their components is different, and in this case the normals are different.
- This implies that in the position and normal arrays  $p$  must appear twice. Once belonging to the lid, and once belonging to the face of the cylinder



## VBOs: Normals and Vertices

The position and normal arrays must have the same vertex order

positions

|        |        |        |        |        |        |        |        |        |          |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|----------|--------|--------|--------|
| $p_0x$ | $p_0y$ | $p_0z$ | $p_1x$ | $p_1y$ | $p_1z$ | $p_2x$ | $p_2y$ | $p_2z$ | $\cdots$ | $p_nx$ | $p_ny$ | $p_nz$ |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|----------|--------|--------|--------|

normals

|        |        |        |        |        |        |        |        |        |          |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|----------|--------|--------|--------|
| $n_0x$ | $n_0y$ | $n_0z$ | $n_1x$ | $n_1y$ | $n_1z$ | $n_2x$ | $n_2y$ | $n_2z$ | $\cdots$ | $n_nx$ | $n_ny$ | $n_nz$ |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|----------|--------|--------|--------|



## OpenGL – Normals and VBOs

- The process to use VBOs with normals is similar to the one we used before with vertex positions.
- VBO Init
  - Step 1 a) Enable Buffers

```
glEnableClientState(GL_VERTEX_ARRAY);  
glEnableClientState(GL_NORMAL_ARRAY);
```





# OpenGL – Normals and VBOs

- VBO Init

- Step 1 b – Allocate and fill the vertex and normal arrays

```
// vertex array
float *vertexB;
// fill the array
...
// normal array
float *normalB;
// fill the array
...
```

- Step 1 c (optional) – Allocate and fill the index array

```
unsigned int *indices;
...
```



## OpenGL – Normals and VBOs

- VBO Init
- Step 1 d : Create the VBOs

```
GLuint buffers[2];  
// two buffers: vertex coordinates and normals  
float *vertexB, *normalB;  
...  
// create two buffers  
glGenBuffers(2, buffers);  
  
// bind and copy data  
glBindBuffer(GL_ARRAY_BUFFER, buffers[0]);  
glBufferData(GL_ARRAY_BUFFER, arraySize, vertexB, GL_STATIC_DRAW);  
  
glBindBuffer(GL_ARRAY_BUFFER, buffers[1]);  
glBufferData(GL_ARRAY_BUFFER, arraySize, normalB, GL_STATIC_DRAW);
```



## OpenGL – Normals and VBOs

- Draw with VBOs

- Step 2 a – Semantics

- For each buffer: what will it be used for

```
glBindBuffer(GL_ARRAY_BUFFER, buffers[0]);  
glVertexPointer(3, GL_FLOAT, 0, 0);
```

```
glBindBuffer(GL_ARRAY_BUFFER, buffers[1]);  
// normals have always 3 components  
glNormalPointer(GL_FLOAT, 0, 0);
```



## OpenGL – Normals and VBOs

- Draw with VBOs

- Step 2 b: Drawing

- With an index list

```
glDrawElements(GL_TRIANGLES, count, GL_UNSIGNED_INT, indices);
```

- Without an index list

```
glDrawArrays(GL_TRIANGLES, first, count);
```

Note: count is the number of vertices/indices to draw



## Assignment

- Define the normal vectors for the cylinder
- Add all the required instructions to draw a cylinder lit by a directional light
- Try using the specular component



## Questions?

- What happens if we perform some geometrical transformation before placing the light? For instance:

```
glRotatef(45, 0,1,0);  
glLightfv(GL_LIGHT0, GL_POSITION, dir);
```

- What happens if the light is placed before the gluLookAt?

```
gluLookAt(camX, camY, camZ,  
          0.0, 0.0, 0.0,  
          0.0f, 1.0f, 0.0f);  
glLightfv(GL_LIGHT0, GL_POSITION, dir);
```

VS

```
glLightfv(GL_LIGHT0, GL_POSITION, dir);  
gluLookAt(camX, camY, camZ,  
          0.0, 0.0, 0.0,  
          0.0f, 1.0f, 0.0f);
```



## Questions?

- What happens when we provide normals with length  $\neq 1$  ?
- What happens if we use  $(1,0,0)$  as the light color, and  $(0,1,0)$  as the objects color?
  - Why?
  - How to fix this assuming that we really want a red light lighting a green object?