



# COMPUTAÇÃO GRÁFICA



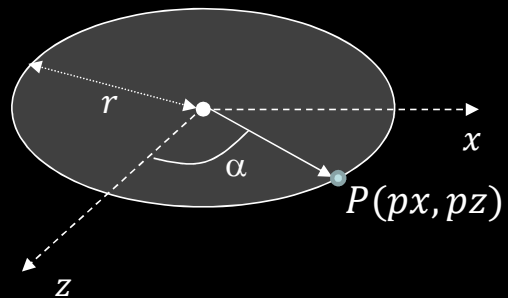
MIEI / LCC  
DEPARTAMENTO DE INFORMÁTICA  
UNIVERSIDADE DO MINHO

## Camera Motion Drawing a Cylinder



# Polar Coordinates

- Polar coordinates specify points based on an angle and a radius.



*Polar Coordinates*  
 $(\alpha, r)$

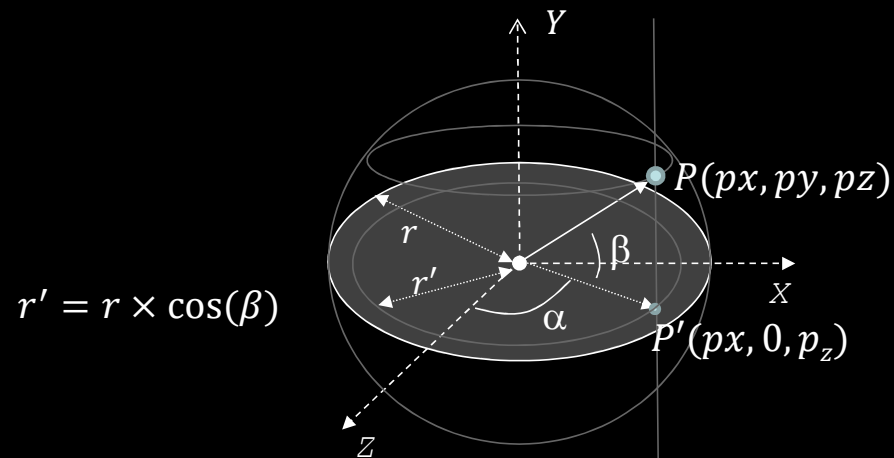


*Cartesian Coordinates*  
 $px = r * \sin(\alpha);$   
 $pz = r * \cos(\alpha);$



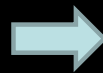
# Spherical Coordinates

- Specify a point on the surface of a sphere



*Spherical Coordinates*

$(\alpha, \beta, r)$   
 $-90 < \beta < 90$



*Cartesian Coordinates*

$z = r \times \cos(\beta) \times \cos(\alpha);$   
 $x = r \times \cos(\beta) \times \sin(\alpha);$   
 $y = r \times \sin(\beta);$



## Explorer Mode Camera

- The camera moves in the surface of a sphere, always looking at the centre of the sphere.
- Don't allow the camera to be upside down.

```
gluLookAt( px, py, pz,    // camera position
           lx, ly, lz,    // look at point
           ux, uy, uz)    // "up vector" (0.0f, 1.0f, 0.0f)
```

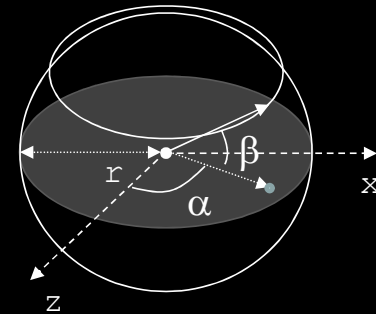
- The look at point is constant (0.0f, 0.0f, 0.0f)
- The camera position is defined based on spherical coordinates ( $\alpha$ ,  $\beta$ ,  $radius$ ) that must be converted to Cartesian coordinates ( $p_x$ ,  $p_y$ ,  $p_z$ )
  - ( $\alpha$ ,  $\beta$ ) determine the position of the camera in a sphere of radius  $r$ . Limit  $|\beta| < 1.5$  (radians)
  - $radius$  determines the distance of the camera to the look at point



## FPS Camera

- Camera Orientation
- The view direction is obtained using spherical coordinates. The look at point is defined based on the view direction and the actual camera position.
- The view direction is a vector  $D$  computed based on the two angles ( $\alpha$  and  $\beta$ ) that define the horizontal and vertical orientation respectively.
- Considering  $P = (p_x, p_y, p_z)$ , the camera position, and  $D = (d_x, d_y, d_z)$ , the view direction:

```
gluLookAt(px,py,pz,  
          px+dx, py+dy, pz+dz,  
          ux, uy, uz);
```

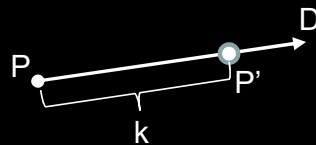




# FPS Camera

- Camera Motion
- Forward and backward movement is achieved using vector  $D$
- $D$  should be normalized
- To move the camera  $k$  units forward, and considering  $D$  to be a unit vector, implies recomputing the camera position as follows:

$$P' = P + k \times D$$

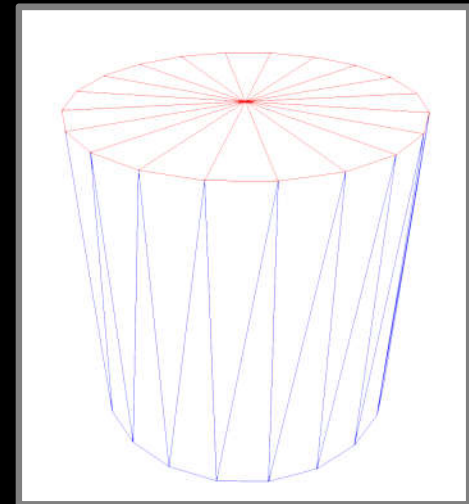


- Note: the look at point must also be displaced.



## Practical Assignment

- Build a cylinder using triangles defining the vertices based on polar coordinates
  - function `drawCylinder(float radius, float height, int slices)`
- Complete the code skeleton to build an interactive application using the keyboard to move the camera up/down and left/right (explorer mode) using spherical coordinates.
  - function `processKeys` and `processSpecialKeys`





## $\pi$ and math.h

- A value for PI is defined in constant `M_PI` in `math.h`
- To have this constant available we should write:

```
#define _USE_MATH_DEFINES // always before the include
#include <math.h>
```

- Usage example:

```
float x = M_PI / n;
```





## Useful functions

- Set a color:

```
glColor3f(r,g,b)
```

- note: call the function before sending the vertices to the GPU

- Change drawing mode:

```
glPolygonMode(faces, mode);
```

- **faces:** GL\_FRONT, GL\_BACK, GL\_FRONT\_AND\_BACK
- **mode:** GL\_FILL, GL\_LINE, GL\_POINT