# Implementarea maparii texturilor intr-o aplicatie OpenGL

# Calculul coordonatelor textura

**Coordonatele textura se asociaza coordonatelor obiect ale suprafatei pe care se aplica.**

**Asocierea coordonatelor textura se poate efectua in 3 moduri:**

1. In aplicatia OpenGl:

- Coordonatele textura (u,v) se ataseaza varfurilor primitivelor (**Obiectele importate au coordonate (u,v) atasate varfurilor**):

- Coordonatele (u,v) sunt intrari pentru Vertex shader.

- Sunt intrari pentru Vertex shader

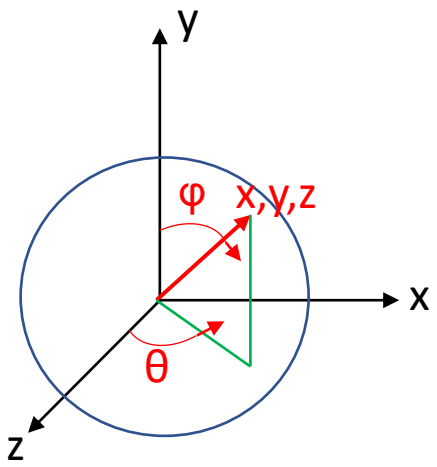
- Sunt declarate iesiri din Vertex shader

- Fragment shader primeste coordonate textura interpolate de GPU (intre coord textura ale varfurilor primitivei), care se folosesc pentru accesarea texturii.

2. In Vertex shader (VS):
   - Se pot calcula coordonate textura pentru varf folosind o functie de mapare standard
   - Sunt declarate iesiri din VS
   - Fragment shader primeste coordonate textura interpolate

3. In Fragment shader (FS):
   - se pot calcula coordonate textura pentru fragment folosind o functie de mapare standard

# Maparea sferica
# în sistemul de coordonate OpenGL




Ecuatiile parametrice ale sferei:

$x = r*\sin(\theta)*\sin(\varphi)$ $\quad 0 <= \theta < 2\pi$

$y = r*\cos(\varphi)$ $\quad 0 <= \varphi <= \pi$

$z = r*\cos(\theta)*\sin(\varphi)$

- Coordonatele sferice ale punctului (x,y,z):

$\theta = \text{arctg}(x/z)$

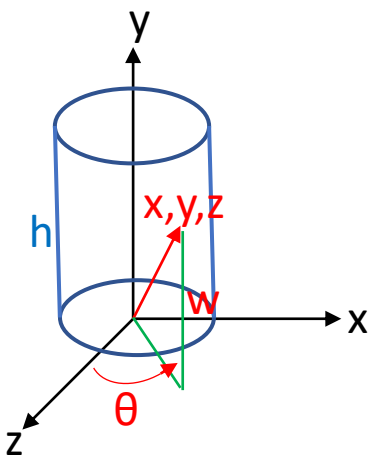
$\varphi = \arccos(y/r)$

**Functiile de mapare sferica**

$u = \theta/ 2\pi = (1/2\pi)*\text{arctg}(x/z)$

$v = \varphi/ \pi = (1/\pi)* \arccos(y/r)$

# Maparea cilindrica
# în sistemul de coordonate OpenGL (1)




Ecuatiile parametrice ale suprafetei cilindrice

x=r*sin(θ)
y= w*h  0<= θ< 2π,  0 < = w <= 1
z= r*cos(θ)

Coordonatele cilindrice (θ, *w*) ale punctului (x, y, z)

θ = arctg(x/z)
w = y/h

**Functiile de mapare cilindrica**

u = (1/2π)arctg(x/z)
v = y/h

Daca cilindrul este centrat in originea sistemului de coordonate:

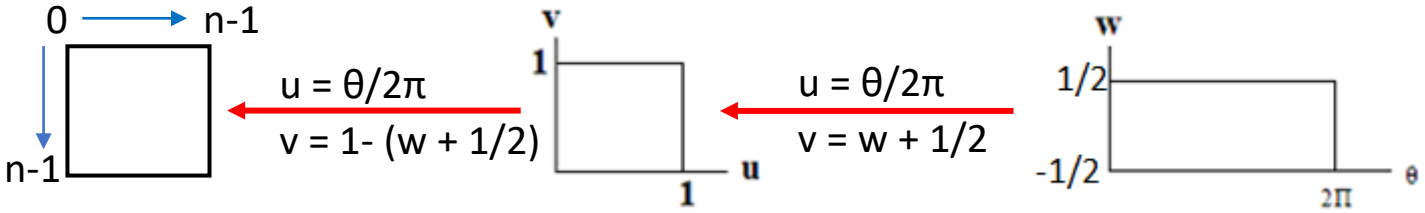
-1/2 < = w <= 1/2

v = w + 1/2
v = y/h + 1/2

# Maparea cilindrica în sistemul de coordonate OpenGL (2)

**Functiile de mapare cilindrica pentru o suprafata cilindrica centrata in originea sistemului de coordonate:**

$u = (1/2\pi)\text{arctg}(x/z)$
$v = y/h + 1/2$




**Functiile de mapare:**

$u = (1/2\pi)\text{arctg}(x/z)$
$v = 1- (y/h + 0.5)$

**Vertex shader**

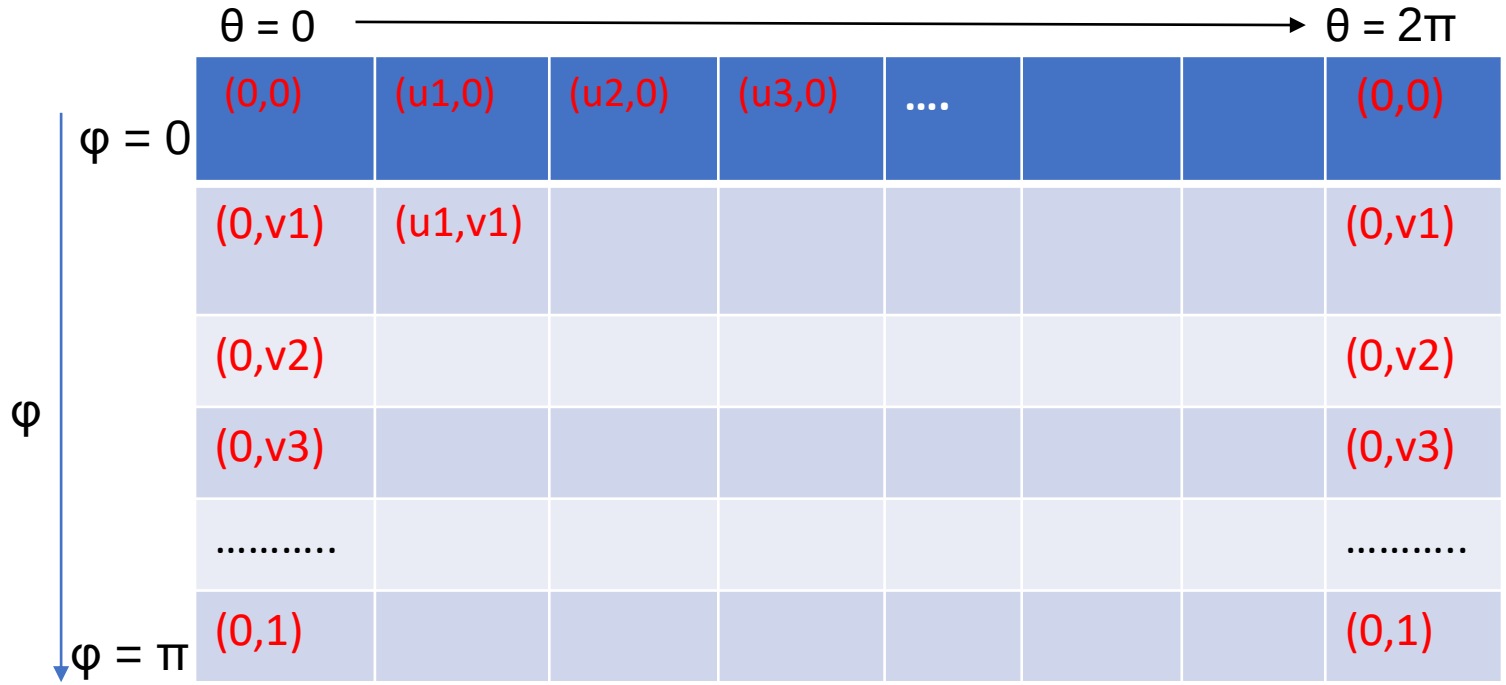
```
layout(location = 0) in vec3 v_position; //pozitia varfului in spatiul obiect
layout(location = 2) in vec2 v_texture_coord; // coordonate textura varf
uniform mat4 Model;
uniform mat4 View;
uniform mat4 Projection;
uniform int mapType;
out vec3 position; //pozitia varfului in spatiul obiect – pt calcul (u,v) in Fragment shader
out vec2 texcoord2;//pentru interpolare → in frag. shader se folosesc coordonate interpolate
void main()
{
  texcoord2 = v_texture_coord;
  position = v_position; //pozitie varf in coordonate obiect
  gl_Position = Projection * View * Model * vec4(v_position, 1.0);
}
```

**Fragment shader**

```
uniform sampler2D texture_1;
uniform int mapType;
in vec3 position; //pozitia fragmentului in spatiul obiect, pt calcul (uf,vf)
in vec2 texcoord2; // coordonate textura interpolate de rasterizator
layout(location = 0) out vec4 out_color;

void main()
{  if ( mapType > 2) { // coordonate textura interpolate
       texcoord = texcoord2; }
   vec2 texcoord;
   if (mapType == 1)  //mapare sferica
//position se considera punct de pe o sfera centrata in origine de raza length (position)
   {  texcoord.x = (1.0 / (2 * 3.14)) * atan (position.x, position.z);
      texcoord.y = (1.0 / 3.14) * acos (position.y / length (position));
    }
   if (mapType == 2) // mapare cilindrica
//position este un punct de pe un cilindru centrat in originea sist de coord, cu inaltime h =2
   { texcoord.x = (1.0 / (2 * 3.14)) * atan (position.x, position.z);
     texcoord.y = 1- ( position.y / 2 + 0.5); //cilindru cu inaltimea 2, centrat in origine
    }
    vec3 color = texture2D (texture_1, texcoord).xyz;
    out_color = vec4(color, 1);
}
```

# Calculul coordonatelor varfurilor de pe o sfera şi a coordonatelor textura asociate

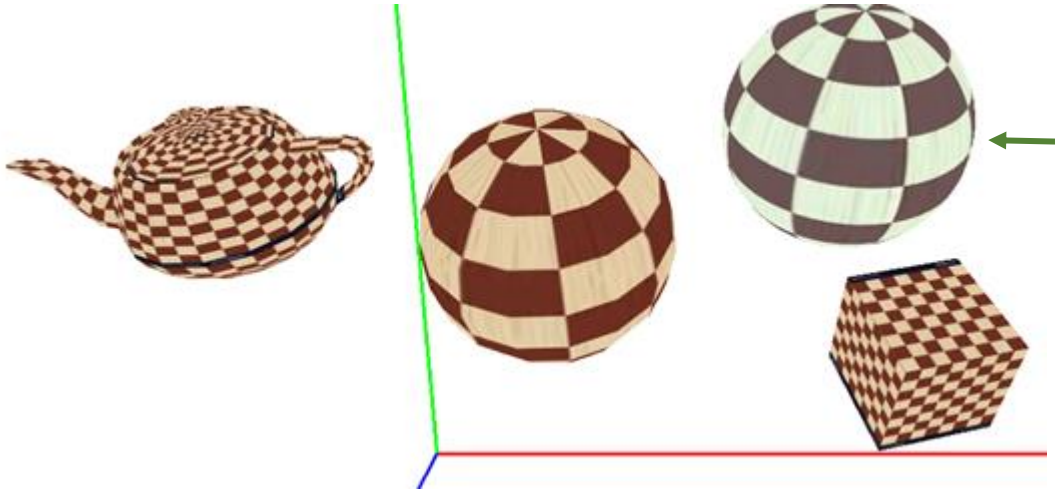



```
glm::vec3 pos = glm::vec3( radius * std::sin(theta) * std::sin(phi),//x
                           radius * std::cos(phi),                  //y
                           radius * std::cos(theta) * std::sin(phi)  //z
```

0 <= phi <= π  cu pasul d_phi
0 <= theta < 2π  cu pasul d_theta

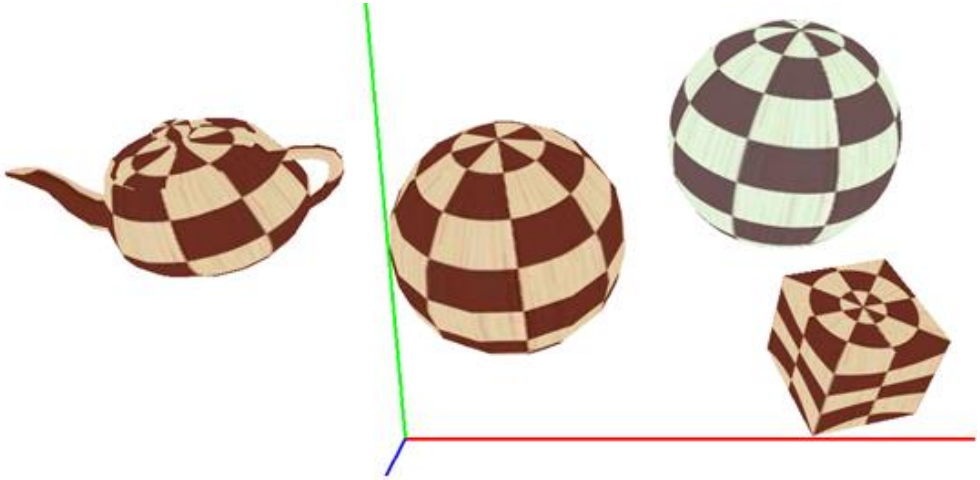
Coordonatele varfurilor de pe ultima coloana sunt identice cu cele de pe prima coloana.

Pentru un varf pos, coord textura sunt  u = theta / (2 * π), v= phi / π.

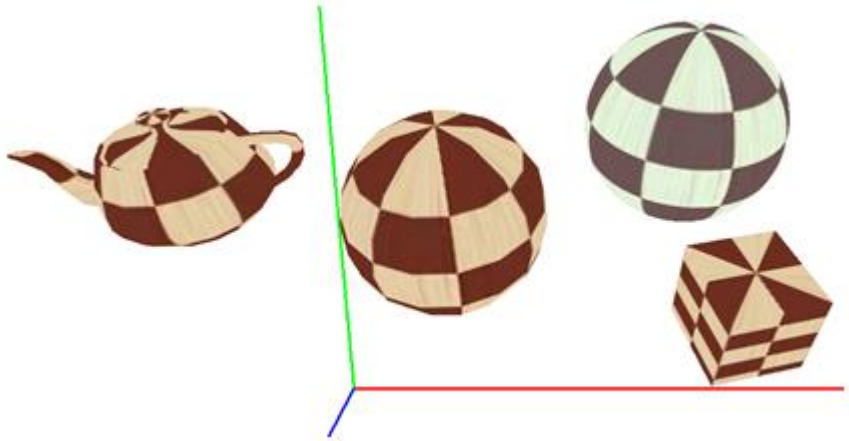


Sfera generată în aplicatie.
Coordonate textură atasate vârfurilor pentru mapare sferică.

Mapare folosind coordonatele textura ale obiectelor importate.



Mapare sferică în Fragment shader



Mapare cilindrică în Fragment shader