CheatSheet IDI ExamenLab (OpenGL i GLSM)

Transformacions Geomètriques (*TG*)

```
glm::mat4() = glm::scale(glm::mat4(), glm::vec3());
glm::mat4() = glm::translate(glm::mat4(), glm::vec3());
glm::mat4() = glm::rotate(glm::mat4(), (float), glm::vec3());
```

Aplicació

A la funció Transform:

```
TG = I;
TG = glm::translate(TG, posObjecte);
TG = glm::scale(TG, glm::vec3(escala_x, escala_y, escala_z);
TG = glm::rotate(TG, angle_x, glm::vec3(1,0,0));
TG = glm::rotate(TG, angle_x, glm::vec3(0,1,0));
TG = glm::rotate(TG, angle_x, glm::vec3(0,0,1));
TG = glm::translate(TG, centreObjecte);
```

Project i View Transform

Project Transform

```
glm::mat4() = glm::perspective((float) , (float), (float), (float));
glm::mat4() = glm::ortho((float), (float), (float), (float), (float), (float));
```

Aplicació

```
float radiEsc = distance(CapsaEscenaMin, CapsaEscenaMax)/2.f;
float angle_inicial = glm::asin(radiEsc/(2*radiEsc));
```

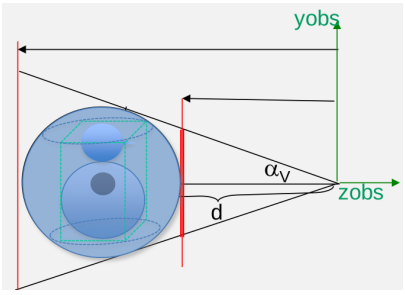

Figure 1: Esfera Escena

```
float FOV = angle_inicial*2;
float ra = amplada/altura;

Proj = glm::perspective(FOV , ra, radiEsc, 3.0f*radiEsc);
Proj = glm::ortho(-radiEsc, radiEsc, -radiEsc, radiEsc, radiEsc, 3.0f*radiEsc);
```

View Transform (Euler)

```
float d = glm::distance(OBS, VRP);
View = glm::translate(glm::vec3(1.f), glm::vec3(0.f, 0.f, -1*d));
c View = glm::rotate(View, Phi, glm::vec3(0,0,1));        //Phi = φ
c View = glm::rotate(View, Theta, glm::vec3(1,0,0));    //Theta = θ
c View = glm::rotate(View, Psi, glm::vec3(0,1,0));        //Psi = ψ
View = glm::translate(View, glm::vec3(-1*VRP.x, -1*VRP.y, -1*VRP.z));
```

Resize

```
ra = ample / (float)alt;

if(ra < 1)
{
    FOV = 2.f*glm::atan(glm::tan(angle_inicial )/ra);
}else
{
    FOV = 2*angle_inicial;
}
```

Shaders

Càlcul de color al Vertex Shader

```
mat3 matNormalInvers = inverse (transpose (mat3 (view * TG)));

vec3 coordSCO = (matNormal * vec4 ( vertex, 1.0)).xyz ;

vec3 posF = (/*view * */ vec4(posFocus , 1.0)).xyz;

vec3 L = posF- coordSCO;

vec3 normalNormal = matNormalInvers * normal;

L = normalize(L);

normalNormal = normalize(normalNormal );

vec3 a;

fcolor = Phong(normalNormal, L, vec4(coordSCO, 1.0));

gl_Position = proj * view * TG * vec4 (vertex, 1.0);
```

Càlcul de color al Fragment Shader

```
mat4 matNormal = view * TG;

vec3 coordSCO = (matNormal * vec4 (vert, 1.0)).xyz ;

vec3 L = posF - coordSCO;

vec3 normalNormal = matNormalInvers * norm;

L = normalize(L);

normalNormal = normalize(normalNormal );

vec3 a;
```

```
a = Phong(normalNormal, L, vec4(coordSCO, 1.0));
FragColor = vec4(a,1);
```