MASTER ELECTRONIC DESIGN

# Homework 5( )

**Miguel Tlapa Juárez**

**07/01/2014**


This document describes the system architecture and design about the body controller module, it's have block diagram and flowchart to describe software and hardware architecture.

# *Revision History*

| Date | Revision Number | Author/Editor | Modifications |
|---|---|---|---|
| June2014 | 0.1 | Miguel Tlapa | Created file |
| | | | |

# *Disclaimers*

**EXERCISES**

1. **Send a light structure from the app.**
2. **Set the light direction to be the camera direction**

ShaderLightsP.hlsl

Definir otra luz

Create Constant Buffer

```
cbuffer LightDef : register(b1)
{
    Light gLight;
}
```

Framework

Crear una nueva clase .h (LightHelper.h)

LightHelper.h

```
#include "DXUtil.h"
```

Crear la estructura de la luz en la APlicacion

Es del tipo XMFLOAT4 porque estamos del lado de la aplicacion

```
struct Light
{
    XMFLOAT4 Position;
    //-----------16 bytes
    XMFLOAT4 Direction;
    //-----------16 bytes
    XMFLOAT4 Color;
    //-----------16 bytes
    float  Kc;  //constant
    float  Kl;  //linear
    float  Kq;  //quadratic
    float  SpotAngle;
    //-----------16 bytes
};
```

LightsApp.cpp

```
4   #include "Framework\LightHelper.h"
```

Definir la Estructura

```
79      enum PSConstantBuffer
80      {
81          CB_EyeInfo,
82          CB_LightInfo,
83          CB_MaterialInfo,
84          PSNumConstantBuffers   //must be the last
85      };
```

```
88      ID3D11Buffer* mpConstP[PSNumConstantBuffers];
```

Definimos el tamaño de los arreglos

```
467 void LightsApp::InitConstantBuffers(){
```

```
481     constantBufferDesc.ByteWidth = sizeof(EYE);
482     HR(m_pDevice->CreateBuffer(&constantBufferDesc, nullptr, &mpConstP[CB_EyeInfo]));
483
484     constantBufferDesc.ByteWidth = sizeof(Light);
485     HR(m_pDevice->CreateBuffer(&constantBufferDesc, nullptr, &mpConstP[CB_LightInfo]));
486
487     constantBufferDesc.ByteWidth = sizeof(Material);
488     HR(m_pDevice->CreateBuffer(&constantBufferDesc, nullptr, &mpConstP[CB_MaterialInfo]));
```

**Enviar la posicion de la camara**

```
3  void LightsApp::Update(float dt)
```

```
282     m_pImmediateContext->UpdateSubresource(mpConstP[CB_EyeInfo], 0, nullptr, &eyeInfo, 0, 0);
```

**Definr la Estructura de la Luz**

```
284     // Defining the Light structure
285     Light gLight;
286     gLight.Position = EyePos;
287     gLight.Direction = EyeDir;
288     gLight.Color = XMFLOAT4(1, 1, 1, 1);
289     gLight.Kc = 1.0f;
290     gLight.Kl = 0.01f;
291     gLight.Kq = 0.02f;
292     gLight.SpotAngle = XMConvertToRadians(15);
```

**Enviamos la estructura de la Luz**

```
294     m_pImmediateContext->UpdateSubresource(mpConstP[CB_LightInfo], 0, nullptr, &gLight, 0, 0);
```

**Definir cuantos bufers estoy utilizando**

```
304 void LightsApp::Render(float dt)
```

```
322     m_pImmediateContext->VSSetConstantBuffers(0, NumConstantBuffers, mpConstantBuffers);
323     m_pImmediateContext->PSSetShader(mpPS, nullptr, 0);
324     m_pImmediateContext->PSSetConstantBuffers(0, PSNumConstantBuffers, mpConstP);
```

ShaderLightsP.hlsl*

**Definir la estructura del material**

```
18 struct Material
19 {
20     XMFLOAT4 Ambient; //16
21     XMFLOAT4 Diffuse; //16
22     XMFLOAT4 Specular; //16
23     //----------------boundary 16 bytes
24     float  SpecularPower; //4
25     int    UseTexture; //4 bytes
26     float  Padding[2];
27     //----------------boundary 16 bytes
28 };
29
```

**Definimos el Buffer**

```
48 cbuffer MaterialInfo: register (b2)
49 {
50     Material gMaterial;
51 }
```

**Como ya le enviamos la luz desde la aplicación modificamos el argumento de DoSpecular**

```
67 float4 DoSpecular(Light light, float3 V, float3 L, float3 N)
```

```
79     return light.Color * pow(NdotH, gMaterial.SpecularPower);
```

```
145 float4 PShader(PIn IN) : SV_TARGET
```

```
175     float4 finalColor = gMaterial.Ambient +
176                         gMaterial.Diffuse*lr.Diffuse +
177                         gMaterial.Specular*lr.Specular;
```

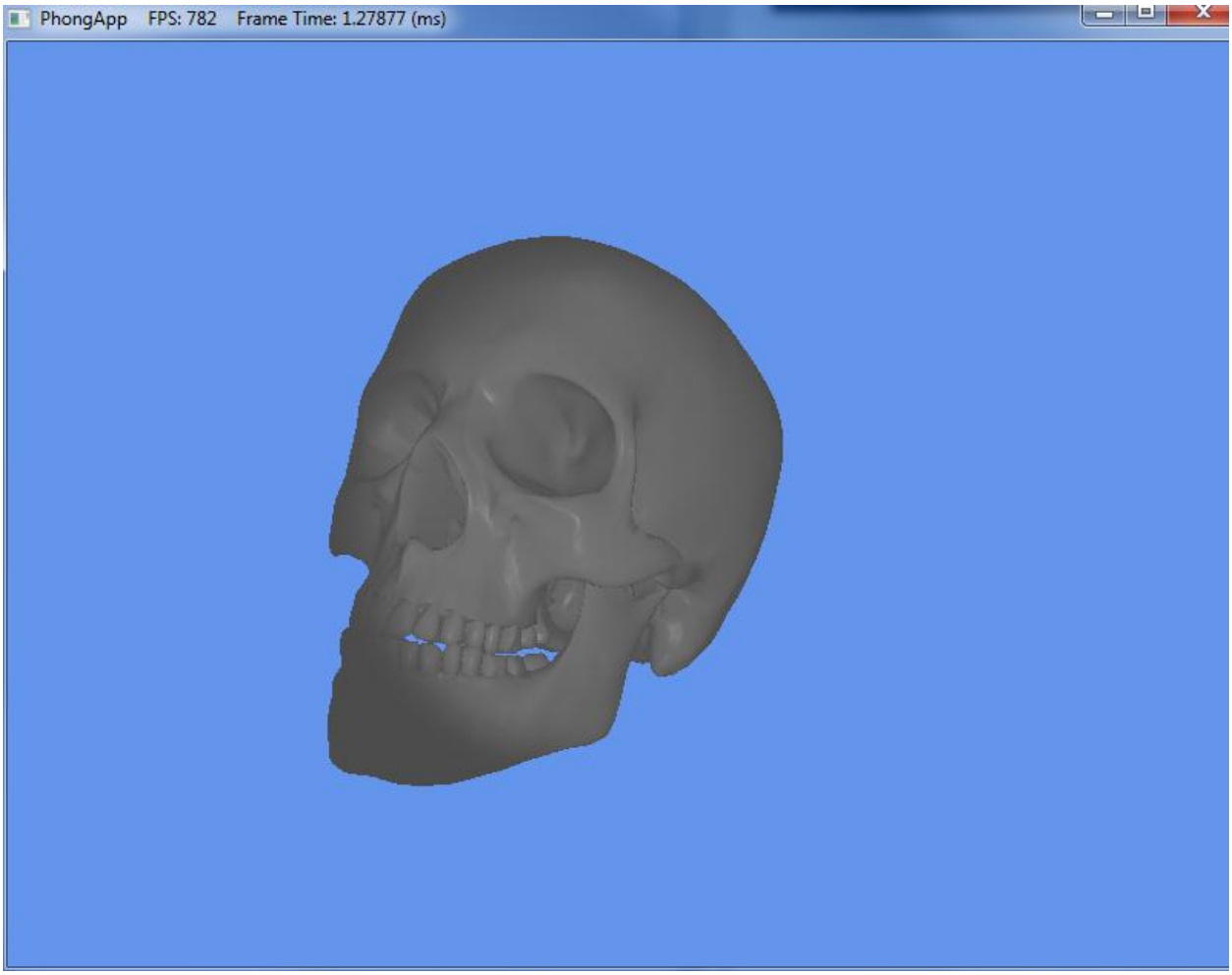
```
467 void LightsApp::InitConstantBuffers(){
```

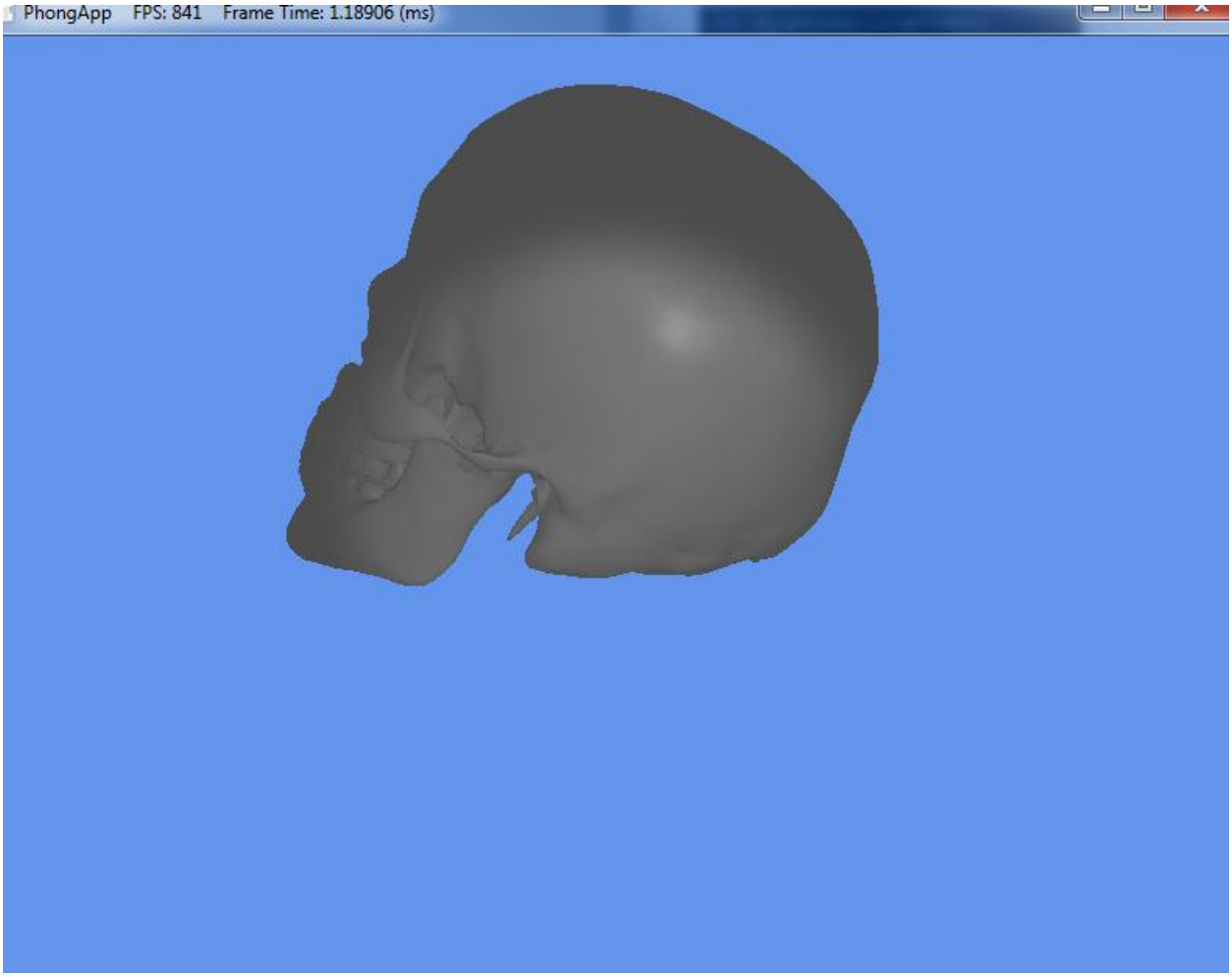
```
487     constantBufferDesc.ByteWidth = sizeof(Material);
488     HR(m_pDevice->CreateBuffer(&constantBufferDesc, nullptr, &mpConstP[CB_MaterialInfo]));
```

**Liberar Espacio en Memoria**

```
175 LightsApp::~LightsApp()
176 {
```

```
189     Memory::SafeRelease(mpConstP[CB_EyeInfo]);
190     Memory::SafeRelease(mpConstP[CB_LightInfo]);
191     Memory::SafeRelease(mpConstP[CB_MaterialInfo]);
192
193     Memory::SafeRelease(mpWireframeRS);
```



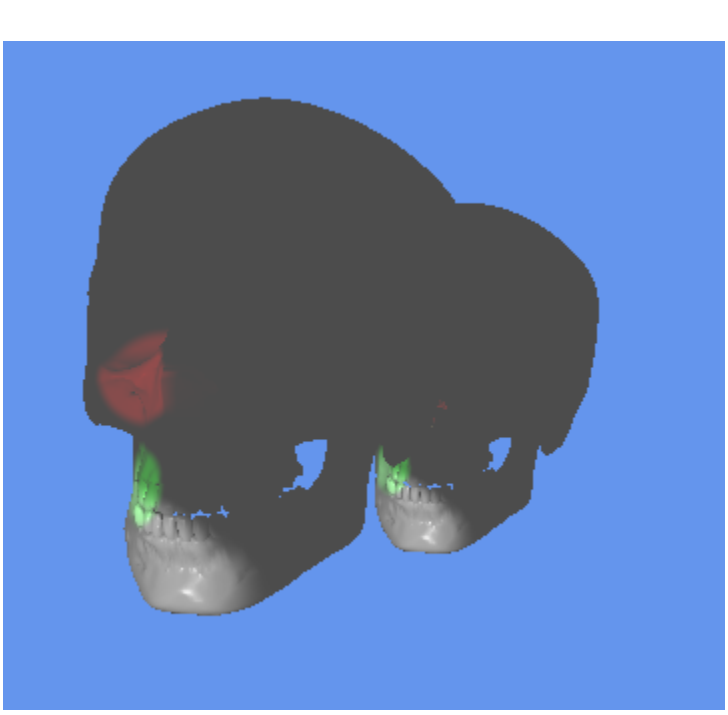


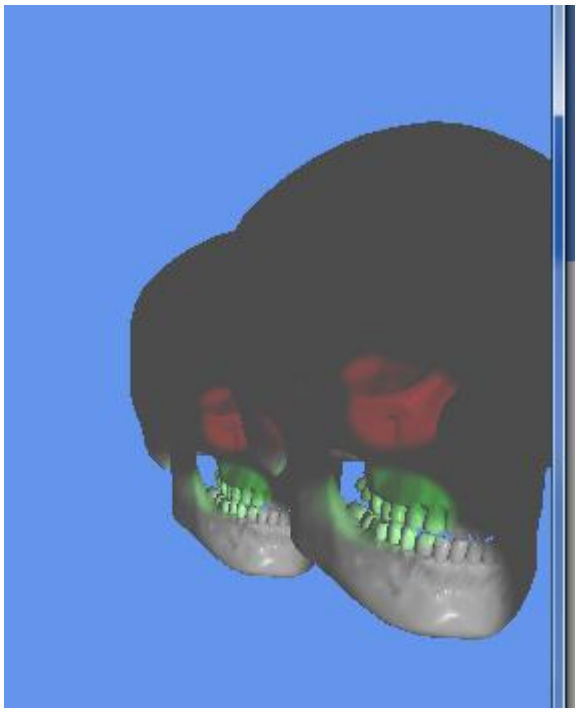

**3. Investigate how to implement a light that affects properly two different objects.**

```
330 void LightsApp::Render(float dt)
331 {
```

```
XMMATRIX t = XMMatrixTranslation(0, 6, -10);
XMStoreFloat4x4(&mSkullWorld, t);

m_pImmediateContext->UpdateSubresource(mpConstantBuffers[CB_Object], 0, nullptr, &mSkullWorld, 0, 0);
m_pImmediateContext->DrawIndexed(mModelIndexCount, 0, 0);
```





4. **Implement 3 different lights in different places**

```
233 void LightsApp::Update(float dt)
234 {
```

**Definir el tamano del arreglo de las 3 luces**

```
291     Light gLight[3];
```

**Definir valors para cada uno de los elementos del arreglo y solo cambiar el valor de las posiciones y los colores para poder ver el cambio**

```
291     Light gLight[3];
292     //gLight.Position = EyePos;
293     gLight[0].Position = XMFLOAT4(0.0f, 4.0f, -10.0f, 1.0f);
294     gLight[0].Direction = EyeDir;
295     gLight[0].Color = XMFLOAT4(1, 1, 1, 1);
296     gLight[0].Kc = 1.0f;
297     gLight[0].Kl = 0.01f;
298     gLight[0].Kq = 0.02f;
299     gLight[0].SpotAngle = XMConvertToRadians(15);
300
301     //gLight.Position = EyePos;
302     gLight[1].Position = XMFLOAT4(0.0f, 8.0f, -10.0f, 1.0f);
303     gLight[1].Direction = EyeDir;
304     gLight[1].Color = XMFLOAT4(1, 0, 0, 1);
305     gLight[1].Kc = 1.0f;
306     gLight[1].Kl = 0.01f;
307     gLight[1].Kq = 0.02f;
308     gLight[1].SpotAngle = XMConvertToRadians(5);
309
310     //gLight.Position = EyePos;
311     gLight[2].Position = XMFLOAT4(0.0f, 6.0f, -10.0f, 1.0f);
312     gLight[2].Direction = EyeDir;
313     gLight[2].Color = XMFLOAT4(0, 1, 0, 1);
314     gLight[2].Kc = 1.0f;
315     gLight[2].Kl = 0.01f;
316     gLight[2].Kq = 0.02f;
317     gLight[2].SpotAngle = XMConvertToRadians(5);
```

**Enviar la primera posicion del arreglo gLigjht**

```
320     m_pImmediateContext->UpdateSubresource(mpConstP[CB_LightInfo], 0, nullptr, &gLight[0], 0, 0);
```

**Modificar el valor del Tamano del Arreglo en el InitConstantBuffers()**

```
493 void LightsApp::InitConstantBuffers(){
510     constantBufferDesc.ByteWidth = sizeof(Light)*3;
```

**Modificar el valor del arreglo en la estructura**

ShaderLightsP.hlsl

```
cbuffer LightDef : register(b1)
{
    Light gLight[3];
}
```

```
45  float4 PShader(PIn IN) : SV_TARGET
46  {
```

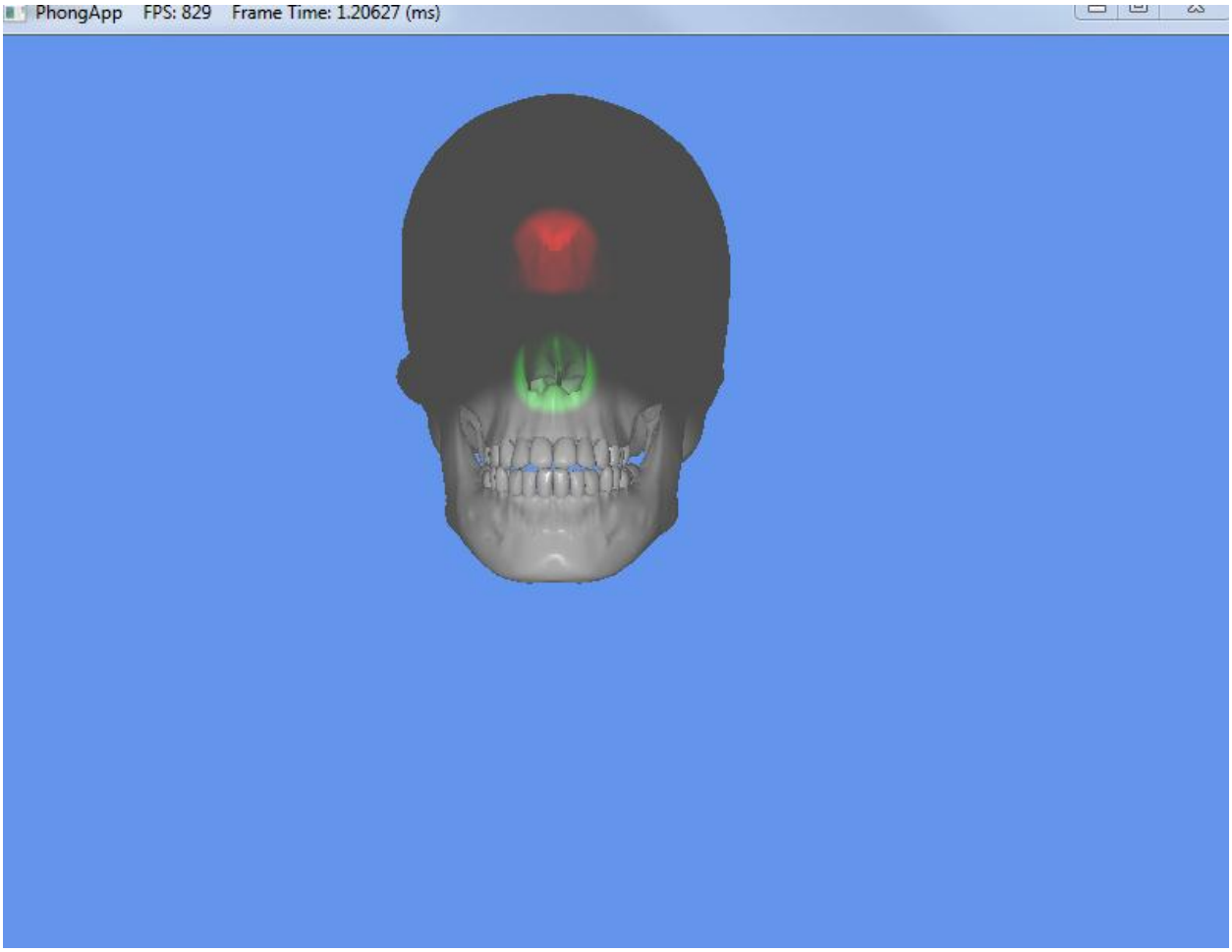
**Definir**

```
6       float4 diffuse ;
7       float4 specular;
```

```
for (int i = 0; i < 3; i++)
{
    LightingResult lr = DoSpotLight(gLight[i], V, N, P);

    diffuse += lr.Diffuse;
    specular += lr.Specular;
}

float4 finalColor = gMaterial.Ambient +
                    gMaterial.Diffuse*diffuse +
                    gMaterial.Specular*specular;
```



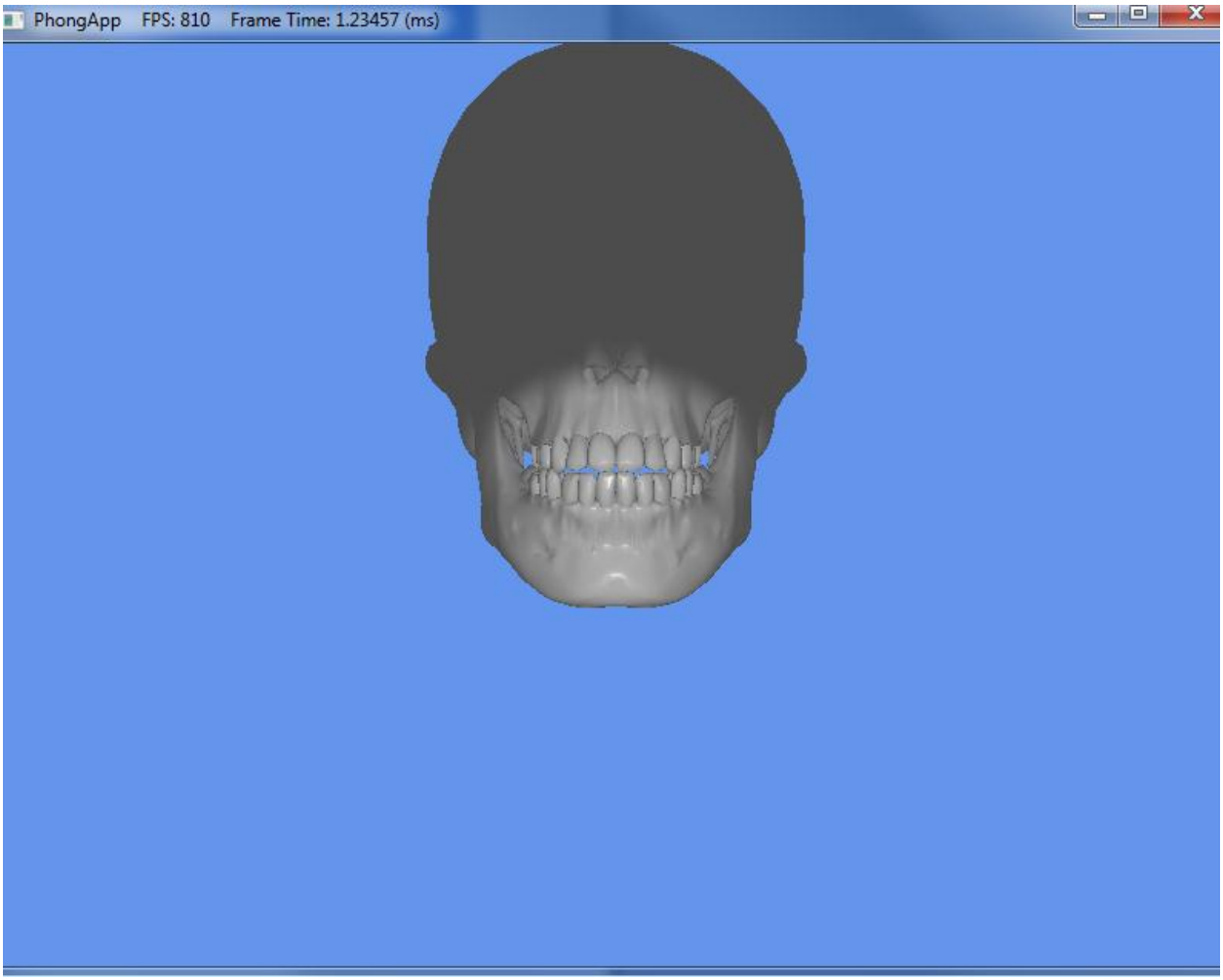

5. **Send the position of the lights from the app**

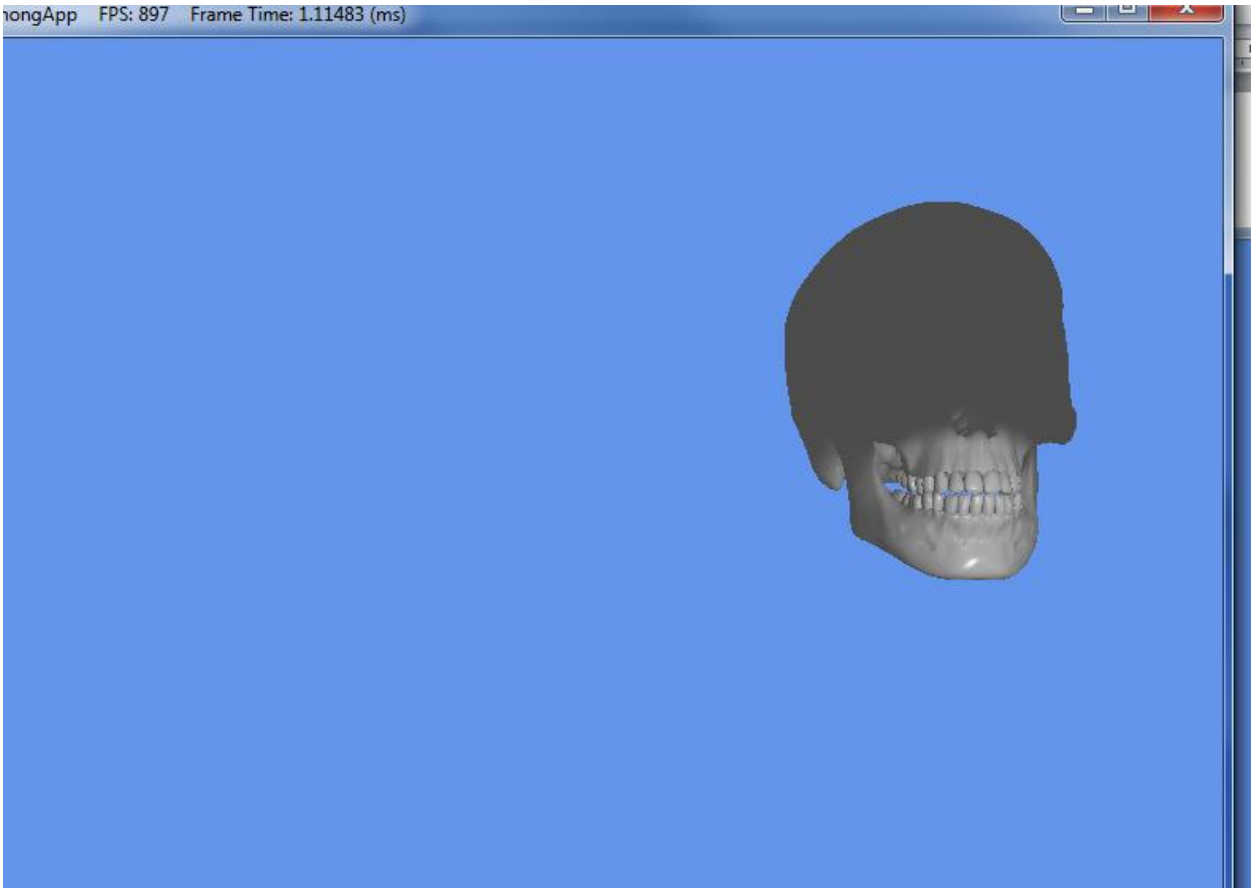
```
233 void LightsApp::Update(float dt)
234 {
```

**Defini de manera fija la posicion de la camara y comente la linea que actualiza automaticamente la posicion de la camara.**

```
291     Light gLight;
292     //gLight.Position = EyePos;
293     gLight.Position = XMFLOAT4(0.0f, 4.0f, -10.0f, 1.0f);
294     gLight.Direction = EyeDir;
```

**Muevo las teclas y la posicion de la luz queda fija en el ejercicio 1 si muevo las teclas tambien se mueve la camara.**


PhongApp FPS: 810 Frame Time: 1.23457 (ms)


ongApp FPS: 897 Frame Time: 1.11483 (ms)