

NORMAL MAPS

Sergio Peñaloza

11100
11010
01011
11011
00100
01100
01010
11011
00000

001001011111101111110101110
0011101101000100001000000
111001011101100011110101111

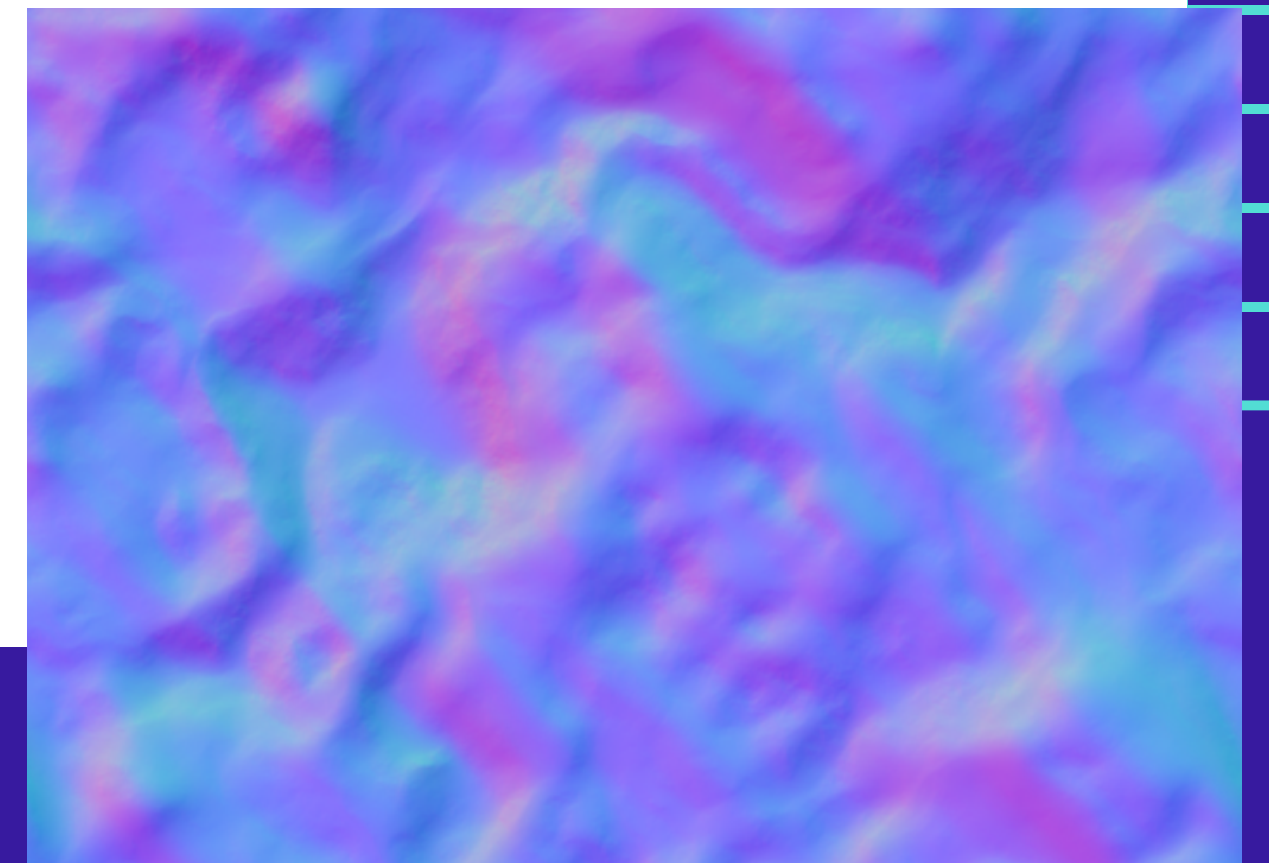
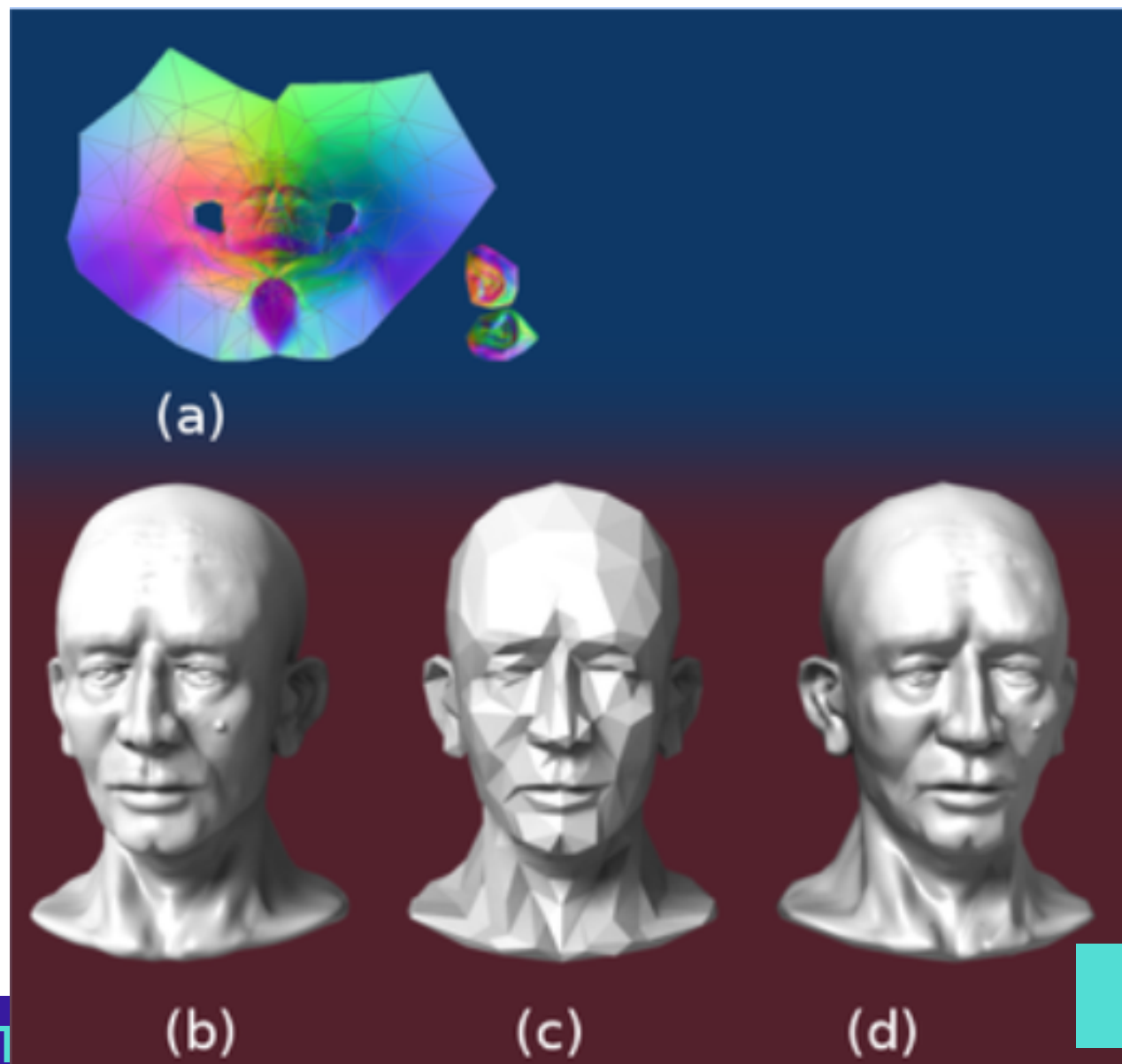
Concepto General

A : Normal map en espacio de objeto

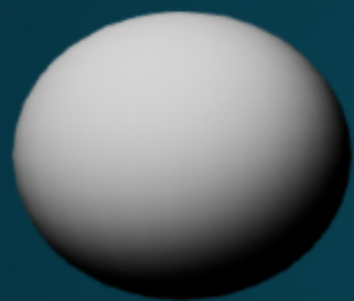
B : Modelo high poly

C : Modelo low poly usando face normals

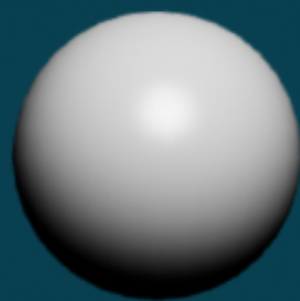
D : Modelo low poly con normal map aplicado



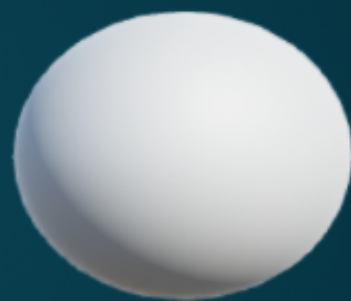
111011111101011100100111011010001000
100000001110010111011000111101011111
1111000010101001011110101011101100011
10101111110111100001010100101111010101
1011010001011011001100100111010101110
0100000110010010111111011111101011100
00111011010001000010000000111001011
011
111
010
010
111



Lambert



Blinn-Phong



BRDF

**Algunas
Ecuaciones de
Iluminación
que utilizan
normales**

000001110010111011000111101011111101111000010
001011110101011011010001011011001100100111010
1100100000110010010111111011111110101110010011
0100010
01000000011100101110110001111010111110111100

IMPLEMENTACION (VERTEX/FRAGMENT SHADERS)

```
//This function uses tex2D, use it only in fragment shader
//Assuming the normal map texture channels RGB match Normal Map XYZ
float3 CalculateNormals(sampler2D normalMapTexture, float2 textureSamplingUvs, float3 meshNormal, float3 meshTangent)
{
    //Tangent, Binormal, Normal"
    //Tangent to mesh normal matching space
    //transformation matrix
    const float3x3 TBN = transpose(float3x3(meshTangent, cross(meshNormal, meshTangent), meshNormal));
    const float3 textureNormal = tex2D(normalMapTexture, textureSamplingUvs).xyz * 2.0 - 1.0;
    return mul(TBN, textureNormal);
}
```

RETOS

■ 1

Implementar un normal map invertido (las protuberancias se convierten en hendiduras) con surface shaders o shader graph.



■ 2

Agregar un parámetro que rote las normales después de ser calculadas con surface shaders o shader graph.

