HelloTriangle

Dilligent-based Triangle render

Output original

Figure 1: image

Output modificado

Este Output se consiguió modificando los vertices hard-coded que renderizan el triangulo cambiando el signo de las 2 esquinas, de forma tal que el triangulo salga al revés, esto funciona porque el renderizado del triangulo se basa en vertices calculados por medio de valores predeterminados.

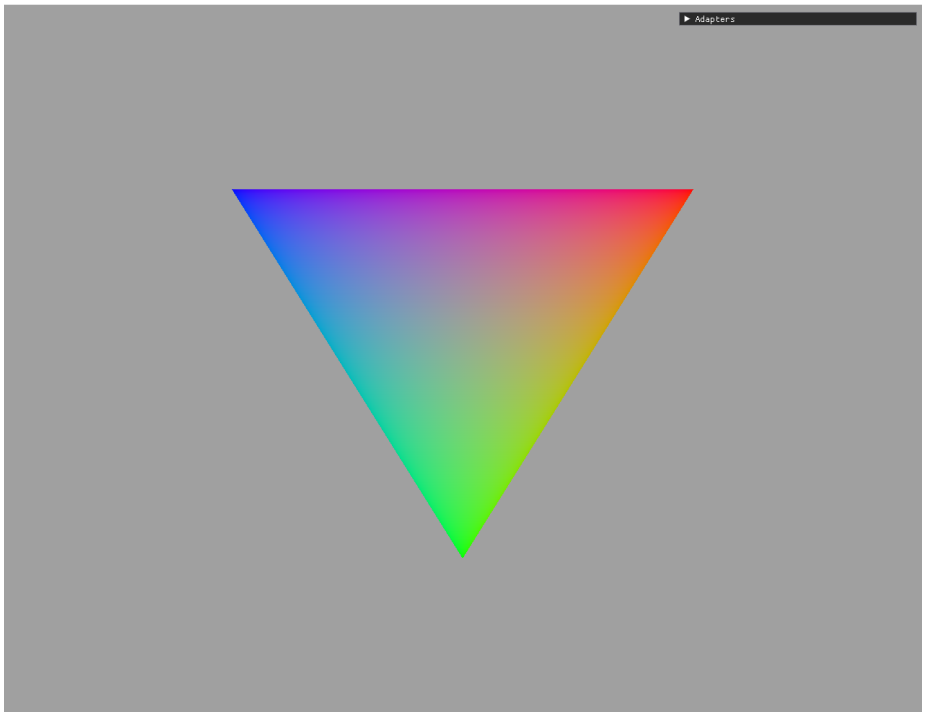

Figure 2: image

```
void main(in  uint    VertId : SV_VertexID,
          out PSInput PSIn)
{
    float4 Pos[3];
    Pos[0] = float4(+0.5, +0.5, 0.0, 1.0);
    Pos[1] = float4( 0.0, -0.5, 0.0, 1.0);
    Pos[2] = float4(-0.5, +0.5, 0.0, 1.0);

    //Para el output original
    //Pos[0] = float4(-0.5, -0.5, 0.0, 1.0);
    //Pos[1] = float4( 0.0, +0.5, 0.0, 1.0);
    //Pos[2] = float4(+0.5, -0.5, 0.0, 1.0);


    float3 Col[3];
    Col[0] = float3(1.0, 0.0, 0.0); // red
    Col[1] = float3(0.0, 1.0, 0.0); // green
    Col[2] = float3(0.0, 0.0, 1.0); // blue

    PSIn.Pos   = Pos[VertId];
    PSIn.Color = Col[VertId];
}
)";
```

Figure 3: image