Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Programación Optimizada para Videojuegos

Master Universitario en Desarrollo de Videojuegos

Tercera actividad WebGL Shadows

Juan Siverio Rojas

La Laguna, 28 de noviembre de 2022

Contenido

1.1 Objetivo

Tras realizar el tutorial sobre iluminación y sombras planares, trata de mostrar un color adecuado para la sombra utilizando un segundo shader de fragmentos especial para las sombras. Para ello, compila un segundo programa de shaders y renderiza la sombra planar con este segundo programa.

1.2 Abstract:

I modified the level.json file to customize the color shadow for each object. I modified the Retrievel() method and Actor() class to show this new attribute. Also, I created a new shader of fragments, which I will only use to render shadows.

This new shader of fragments is executed from a new function called drawProgramShadow(), which iterates finding shadows what to render with the new shaders.

1.3 Proceso

Decido realizar una personalización del color de la sombra de cada objeto, y no la personalización de un único color de sombra general.

1. Modifico el *level.json* agregando el atributo *"shadowColor"* a cada objeto.

2. Modifico el método RetrieveLevel() y la clase *Actor,* para que reflejen este nuevo atributo.

3. Creo un segundo shader de fragmentos, que tan solo tiene un atributo que será el valor de *shadowColor* para cada malla.

```
private const string fsSourceShadow=@"
precision mediump float;
uniform vec4 uShadowColor;
void main(){

gl_FragColor=uShadowColor;

}";
```

Ilustración 1 Shader de fragmentos para sombras

4. Creo una nueva rutina Draw() que primero realiza el renderizado de todas las mallas, para después cambiar de shareds y volver a analizar todos los objetos pero esta vez solo utilizando la información de las sombras.

```
public async Task Draw(){

    // Object independent operations
    await this._context.ClearColorAsync(0, 0, 1, 1);
    await this._context.ClearDepthAsync(1.0f);
    await this._context.DepthFuncAsync(CompareFunction.LEQUAL);
    await this._context.EnableAsync(EnableCap.DEPTH_TEST);
    await this._context.ViewportAsync(0,0,this._context.DrawingBufferWidth,this._context.DrawingBufferHeight);

    await this.getAttributeLocations(program);
    await this._context.UseProgramAsync(program);
    await this._context.UniformMatrixAsync(this.projectionUniformLocation,false,this.ProyMat.GetArray());
    await this._context.UniformAsync(this.ambientLightLocation,ActiveLevel.AmbientLight.GetArray());

    await this._context.BeginBatchAsync();
    await this._context.ClearAsync(BufferBits.COLOR_BUFFER_BIT | BufferBits.DEPTH_BUFFER_BIT);
    drawProgram();
    await this._context.EndBatchAsync();


    await this.getAttributeLocations(programShadow);
    await this._context.UseProgramAsync(programShadow);
    await this._context.UniformMatrixAsync(this.projectionUniformLocation,false,this.ProyMat.GetArray());
    await this._context.BeginBatchAsync();
    //await this._context.ClearAsync(BufferBits.COLOR_BUFFER_BIT | BufferBits.DEPTH_BUFFER_BIT);
    drawProgramShadow();
    await this._context.EndBatchAsync();

}
```

Ilustración 2. Nueva rutina Draw()

5. Esto se realiza apoyándose en las dos nuevas funciones, una para mallas, que apenas ha sufrido cambios y otra para sombras.

```
public async Task drawProgramShadow()
{
// Loop on objects
    foreach( var keyval in ActiveLevel.ActorCollection)
    {
        GameFramework.Actor actor = keyval.Value;
        if(!actor.Enabled)
            continue;

        if(actor.Type==SimpleGame.GameFramework.ActorType.StaticMesh)
        {
            MeshBuffers mBuffers = BufferCollection[actor.StaticMeshId];

            await this._context.UniformAsync(this.shadowColor,actor.shadowColor.GetArray());

    // Buffers to attributes
            await this._context.BindBufferAsync(BufferType.ARRAY_BUFFER, mBuffers.VertexBuffer);
            await this._context.EnableVertexAttribArrayAsync((uint)this.positionAttribLocation);
            await this._context.VertexAttribPointerAsync((uint)this.positionAttribLocation,3, DataType.FLOAT, false, 0, 0L);

            await this._context.BindBufferAsync(BufferType.ELEMENT_ARRAY_BUFFER, mBuffers.IndexBuffer);

            foreach(var smv in actor.ModelViewShadow)
            {
para establecer las sombras
                await this._context.UniformMatrixAsync(this.modelViewUniformLocation,false,smv.GetArray());
                await this._context.DrawElementsAsync(Primitive.TRIANGLES,mBuffers.NumberOfIndices,DataType.UNSIGNED_SHORT, 0);
            }

        }
    }
}
```

Ilustración 3. Función para renderizado de sombras

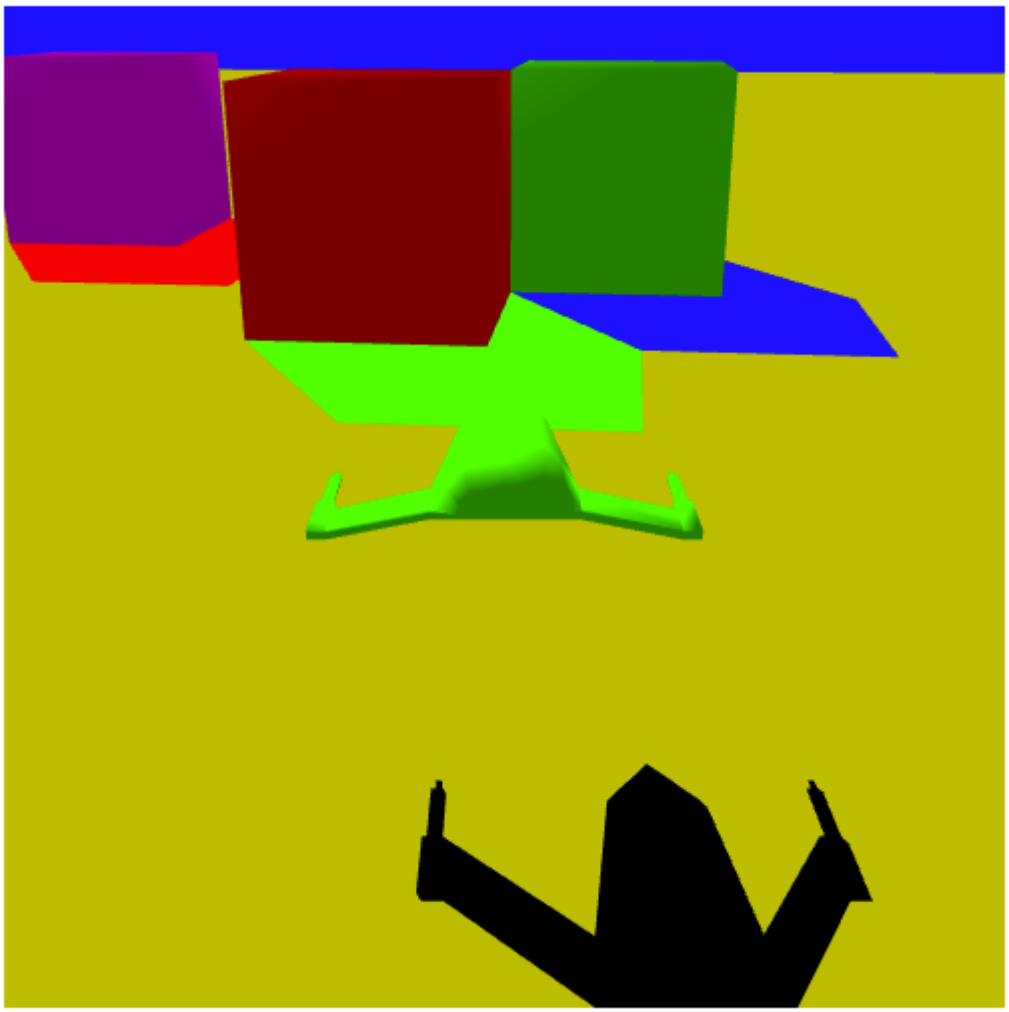

Ilustración 4.Cuatro objetos con diferente color de sombra cada uno.