Informática Gráfica II

Ejercicio: Vertex & Fragment shaders

En estos ejercicios vamos a desarrollar varios shaders para ver algunos de los efectos que se pueden generar con programas GLSL. En particular, nos vamos a centrar en los shaders de vértices (VS) y fragmentos (FS).

Para hacer estos ejercicios, tienes disponible el código inicial en el CV, donde se muestra una escena básica con un suelo y Sinbad sobre él. En el directorio `media/IG2App` deberás tener el fichero `spaceSky.jpg`. Las texturas necesarias para este ejercicio están disponibles en el Tema 10.

Como recordatorio, las referencias a los shaders se hacen en el fichero `.material`. Cada shader irá en un fichero individual, con extensión `.glsl`. Para poder distinguir los shaders por tipo, el nombre de los ficheros de los shaders de vértices terminará en VS, y el nombre de los ficheros de los shaders de fragmentos, en FS.

Tenéis un ejemplo con la estructura completa de dos shaders en la *slide* 15 del Tema 10.

Shader con un color para la esfera

El primer shader a desarrollar es muy simple. En esencia, vamos a colorear la esfera de la escena de color naranja. Para ello, primero definiremos un nuevo material en nuestro fichero de materiales, que llamaremos `example/shaderOrange`.

En este material simplemente incluiremos – dentro del pase y la técnica – las referencias al shader de vértices – que llamaremos `shaderOrangeVS` – y al de fragmentos – al que llamaremos `shaderOrangeFS`.

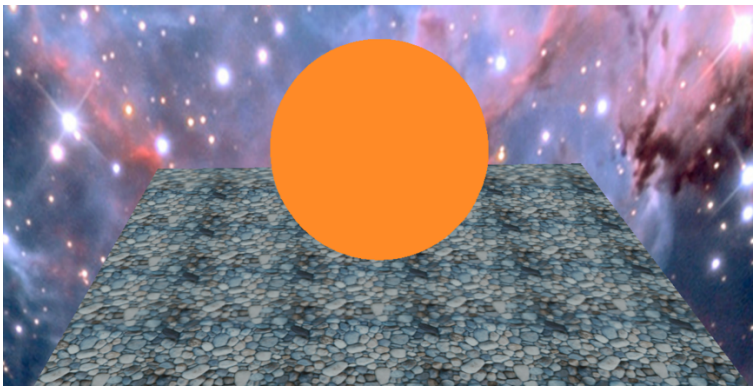
Recordad que las referencias a los shader de vértices y fragmentos se indican con las directivas `vertex_program_ref` y `fragment_program_ref`, respectivamente. En estas referencias no pasaremos ningún parámetro.

En el mismo fichero de materiales, definiremos el enlace con los shaders, donde pasaremos los parámetros correspondientes. Así, en el shader de vértices, indicaremos que el código de este shader se encuentra en el fichero `shaderOrangeVS.glsl` y le pasaremos, como parámetro – `uniform` – la matriz de proyección: `param_named_auto modelViewProjMat worldviewproj_matrix,` de forma que dentro del shader, podremos acceder a ésta mediante la variable `modelViewProjMat.`

En el enlace al shader de fragmentos no pasaremos ningún parámetro, e indicaremos que su código está en el fichero `shaderOrangeFS.glsl`.

Una vez definidos los shaders y los parámetros en el fichero de materiales, empezamos a codificar el shader de vértices. En este shader no vamos a modificar los vértices, pero sí debemos pasar la posición del vértice – a través de la variable `gl_Position` - al shader de fragmentos. El shader de fragmentos básicamente podrá el color naranja al fragmento, en la variable `fFragColor`. Podéis utilizar los valores (1, 0.5, 0.2, 1.0) para asignar el color naranja.

La salida de la escena deberá ser similar a la siguiente imagen:



Shader para modular texturas en el cielo

El segundo shader lo vamos a centrar en el cielo. El código inicial ya genera uno con la textura `spaceSky.jpg`. La idea es modular dos texturas, `lightMap.jpg` y `spaceSky.jpg`, para conseguir un brillo en el centro.

En el fichero de materiales definiremos el material `example/spaceSkyShader`, el cual tendrá las referencias a los shaders de vértices y fragmentos, además de las dos unidades de textura.

El enlace con el shader de vértices lo llamaremos `spaceSkyVs`, el cual definirá el fichero donde se encuentra el shader – `SpaceSkyVS.glsl` – y la matriz de proyección pasada como parámetro.

El enlace con el shader de fragmentos se indicará el fichero del shader – `SpaceSkyFS.glsl` – y pasará como parámetro las dos texturas, las cuales deberán definirse como `uniform sampler2D` en el shader.

El shader de vértices realizará la transformación del vértice con la matriz de proyección `modelViewProjMat`, además de copiar las coordenadas de textura `uv0` que obtiene por defecto como parámetro de entrada `in vec2`. El shader de fragmentos recibe como entrada las coordenadas de textura, que se corresponden con la salida del shader de vértices. Esencialmente, el shader de fragmentos generará el color resultado de modular el color de las dos texturas implicadas.

Cada color se obtiene a través de la función `texture(tex, coord)`, donde `tex` es un sampler2D y `coord` son las coordenadas de textura (tipo `vec2`)

El resultado final se debe almacenar en `fFragColor`, por ejemplo con fFragColor = vec4(color, 1.0), siendo `color` el resultado de modular los colores de ambas texturas.

Recuerda que el nombre de la variable de salida (`out`) de las coordenadas de textura en el shader de vértices, debe coincidir con el nombre en el shader de fragmentos.

Shader para aplicar intensidad de la luz al cielo

Este apartado es una evolución del anterior. Puedes crear un material y shaders nuevos, o modificar el anterior.

La idea es que desde el material pasemos un valor para la luz, de forma que ésta se aplique al cielo y se pueda oscurecer y/o aclarar dependiendo del valor pasado como parámetro.

Así, desde el nuevo material pasaremos la variable `intLuzAmb` *float valorDeLaLuz* al shader de fragmentos.

En el shader de fragmentos utilizaremos este valor, junto con la modulación del color generado de las dos texturas, para crear el color resultante.

Utiliza distintos parámetros en el material para `intLuzAmb` y comprueba que efectivamente el cielo se aclara/oscurece.

Shader para aplicar varias texturas a una esfera

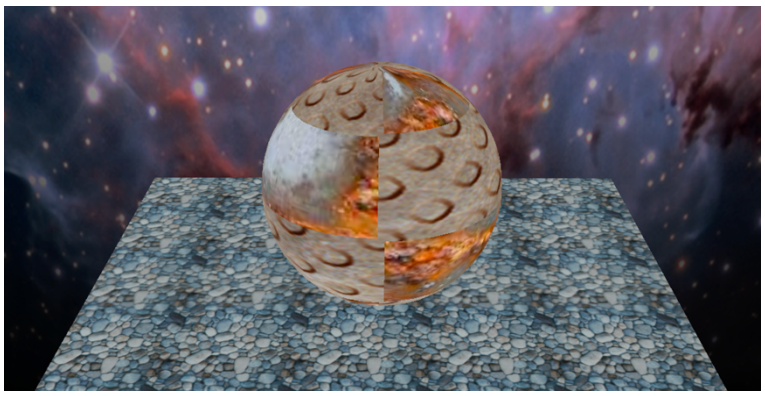
En este shader vamos a dar un efecto a la esfera para que muestre dos texturas distintas. Para ello usaremos un total de tres texturas. La primera será `checker.png`, la cual simula cuadrados como los de un tablero de ajedrez. La idea es que en los cuadrados negros se aplique una textura, y en los blancos, otra. En la imagen de este ejemplo se han utilizado las texturas `corrosion.jpg` y `BumpyMetal.jpg`.

El shader de vértices es igual que el del ejercicio anterior.

Para el shader de fragmentos:

- Debemos definir las tres unidades de textura en el material.
- Debemos pasar las tres texturas desde el material al shader de fragmentos.
- Recuerda que en el shader las texturas son de tipo `sampler2D`.
- Desde el material se pasa el índice de la unidad de textura.
- Utilizaremos `tex_address_mode clamp` en las unidades de textura.

Ahora sólo queda aplicar el color. En resumen: si para la coordenada de textura actual se aplica el color 1 (negro), usamos el color de una textura (segunda unidad de textura), y si no (si es blanco), usamos el color de la otra (tercera unidad de textura).



Shader para crear una esfera hueca

Este shader es una modificación del shader anterior. Vamos a generar una esfera hueca para que sea vea dentro a Sinbad. Para ello usaremos el mismo shader de vértices que en el ejercicio anterior.

El shader de fragmentos utilizará únicamente dos unidades de textura, `checker.png` y `BumpyMetal.jpg`.

En este caso, si para la coordenada de textura actual se aplica el color 1, descartaremos el fragmento, y si no, usamos el color de la segunda unidad de textura.

La siguiente imagen muestra el efecto conseguido, aunque no está bien del todo. ¿Qué falta para conseguir el efecto completo? ¿Cómo podría arreglarse?