INFORMÁTICA GRÁFICA 2
Grado en desarrollo de videojuegos
Curso 2021-22
Práctica 2

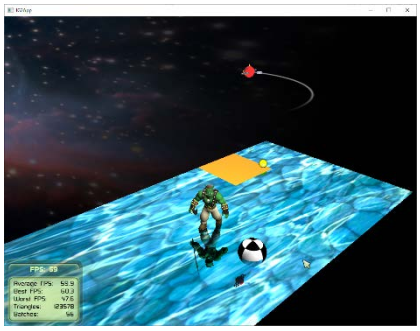
(Entrega 1 de la Práctica 2, apartados 1-)

1. Limpia el código y deja solo la escena con el avión, la bomba, Sinbad, el rio, la plataforma amarilla y la cara feliz. Es decir, elimina la plataforma roja, las nubes y la niebla. Si algún objeto dispone de foco, elimínalo y deja únicamente la luz direccional, encendida, con dirección (0, -1, -1) y con componente difusa (1.0, 1.0, 1.0). Configura Ogre seleccionando **`OpenGL 3+ Rendering Subsystem`**.

2. Vamos a definir un **`SkyPlane`** plano como combinación de texturas. Para ello define una entrada **`IG2/space`** en el archivo de material que especifique que este material no interactúa con la luz, que modula la textura de luces **`lightMap.jpg`** y la textura del cielo **`spaceSky.jpg`** de las unidades de textura 0 y 1 respectivamente y que estas unidades tienen las mismas coordenadas de textura. Configura una animación de rotación para la unidad de textura 1.

3. Añade a la escena un **`skyPlane`** vertical que tenga asociado el material **`IG2/space`** definido en el apartado anterior y que se renderice como se muestra en la captura adjunta. No es necesario que este **`skyplane`** venga de una clase, puedes añadirlo directamente en el **`setupScene()`** de **`IG2App`**. Experimenta con los parámetros del método **`setSkyPlane()`**; por ejemplo, haz pruebas con la normal y la distancia del plano, con la curvatura, con el zoom, etc.

4. Añade a la clase **`Plano`** un método **`setReflejo(Camera*)`** que añada una textura dinámica en la que renderizar el reflejo en el agua. Sigue las transparencias de clase y añade un atributo (**`MovablePlane*`**) para configurar el plano del reflejo de la nueva cámara.



5. **`(Opcional)`** Soluciona el reflejo de la cabeza de la cara feliz en el agua.

6. Añade a la clase **`Plano`** un método **`setEspejo(Camera*)`** que añada una textura dinámica a modo de espejo situado verticalmente sobre la pared derecha de la escena tal como se muestra en la captura adjunta.

7. **`(Opcional)`** Soluciona el reflejo de la cabeza de la cara feliz en el espejo.

En los apartados que siguen, para el material de los shaders utiliza un archivo que se llame `practica2GLSL.material`. Para la esfera debes utilizar la malla `uv_sphere.mesh` que proporciona, además de los vértices, normales y coordenadas de textura.

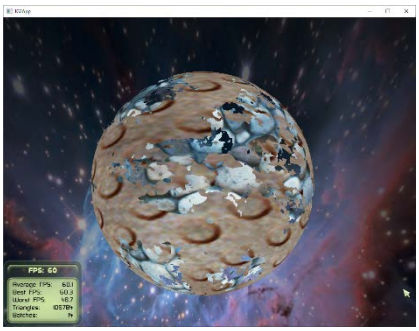
8. Define un shader de fragmentos que llamaremos **`BombaTeseladaFS.glsl`** que muestre la textura de fondo (**`spaceSky.jpg`**) en las zonas negras de la textura de la bomba, y la textura **`BumpyMetal.jpg`**, en las zonas blancas, tal como se ve en la captura adjunta. En este caso, el shader de vértices **`BombaTeseladaVS.glsl`** se limita a proporcionar las coordenadas de textura que necesita el shader de fragmentos y los vértices en coordenadas de recorte.

9. Define la modulación de texturas del **`SkyPlane`** que se hizo en el segundo apartado, pero usando shaders que llamaremos **`SpaceSkyVS.glsl`** y **`SpaceSkyFS.glsl`**. Recuerda que las texturas que se modulan son **`lightMap.jpg`** y **`spaceSky.jpg`**.

10. Modifica los shaders anteriores y define dos nuevos shaders **`SpaceVS.glsl`** y **`SpaceFS.glsl`** de forma que el primero pasa al segundo dos juegos de coordenadas de textura: **`vUv0`** para la textura **`lightMap.jpg`** y **`vUv1`** para **`spaceSky.jpg`**. A estas últimas se les aplica un zoom centrado de factor **`ZF=0.5`**. Recuerda que el intervalo de las coordenadas de textura es [0, 1] de forma que, para aplicarles un zoom centrado, primero hay que centrarlas, trasladándolas al intervalo [-0.5, 0.5], luego se escalan y por último hay que deshacer la traslación (**`c1.s=(c0.s-0.5)*ZF+0.5`**).

11. Pon animación al escalado de forma que la textura crezca y decrezca sin parar. Usa para ello un factor de escalado **`ZF=st*a+b`**, donde **`st`**$\in$**`[-1,1]`**. Para estos valores, la textura debe pasar desde la mitad de su tamaño a su tamaño original. Calcula los valores **`a`**, **`b`** y completa el vertex shader. Para dar valor a **`st`** usa una variable **`uniform float sintime`** que toma valores de forma automática con **`sintime_0_2pi`**, repitiéndose con el intervalo de segundos que prefieras. Las siguientes capturas intentan mostrar que el fondo se aleja y se acerca continuamente.

12. Define shaders para poner agujeros en la bomba. Para el de vértices reutiliza **`BombaTeseladaVS.glsl`**. El de fragmentos se llamará **`HolesFS.glsl`**. Utiliza una textura **`corrosión.jpg`** para descartar fragmentos que serán aquellos cuyo téxel tenga componente roja mayor que 0.6. Este shader utiliza los parámetros uniformes **`texFront`** y **`texBack`** para las texturas exterior e interior de la bomba, respectivamente. Así mismo, el shader utiliza la variable booleana predefinida **`gl_FrontFacing`** para saber si el fragmento es de una cara frontal o de una trasera, tal como se explica en transparencias. Las texturas que dan valor a los parámetros uniformes referidos a texturas son **`BumpyMetal.jpg`** y **`BeachStones.jpg`**, respectivamente. No olvides decir que no se haga **`culling`**. Abajo tienes unas capturas de la bomba con agujeros desde debajo del rio. Observa que los agujeros permiten ver tanto el interior de la esfera como lo que hay detrás de ella, ya sea el fondo o Sinbad.



13. Define los shaders **`HolesAndLightingVS.glsl`** y **`HolesAndLightingFS.glsl`** para iluminar la bomba con agujeros del apartado anterior, **pero en el shader de fragmentos**. Puedes hacerlo usando coordenadas mundiales o de cámara, lo que quieras. El vertex shader pasa al fragment shader, además de **`gl_Position`**, el vértice y la normal en las coordenadas, mundiales o de cámara, que hayas decidido. Recuerda usar entonces las correspondientes matrices para obtener los datos de salida. El fragment shader calculará el color correspondiente a la iluminación usando la componente difusa de la luz junto con su posición/dirección, en las coordenadas, mundiales o de cámara, que hayas elegido. Como coeficiente de reflexión difusa del material, el fragment shader utiliza el color correspondiente de la cara, según sea frontal o trasera. Las texturas irán moduladas entonces con estos coeficientes de reflexión difusa.

En las capturas de abajo, la luz es la suministrada con el proyecto que es direccional, pero con dirección (0, -1, -1) y luz blanca como componente difusa. Las texturas para las caras frontales y traseras son las mismas que en el apartado anterior. El color de las caras frontales es ocre siena (0.72 0.57 0.35) y el de las caras traseras es azul cerúleo (0.0, 0.6, 0.83).