Práctica 10

Objetivo

El shader PBR estudiado en el tema usa materiales definidos a nivel global, que se aplican al objeto en su totalidad. Esto no permite, por ejemplo, definir una parte del objeto con un aspecto metálico y otra parte con un aspecto de madera, ni definir distintos niveles de rugosidad.
El objetivo de esta práctica es adaptar el shader anterior al uso de materiales definidos localmente mediante texturas.

Figura 1. Modelo disponible en https://artisaverb.info/Cerberus.html

Material entregado

Al ejecutar el código del proyecto correspondiente a esta práctica se mostrará el modelo mostrado en la figura, pero únicamente con la textura de albedo aplicada.
La interfaz gráfica permite controlar las 4 luces de la escena, aunque sin resultados en la imagen hasta que implementes el shader adecuadamente.

Tu trabajo

En esta práctica tendrás que portar el shader de PBR estudiado en clase y adaptarlo a su uso con materiales definidos por texturas. Hoy en día es bastante habitual definir los materiales de tipo PBR a partir de varias texturas (https://cc0textures.com/ o https://freepbr.com/). Por ejemplo, para el modelo de esta práctica tenemos las 5 texturas mostradas en la Figura 2.

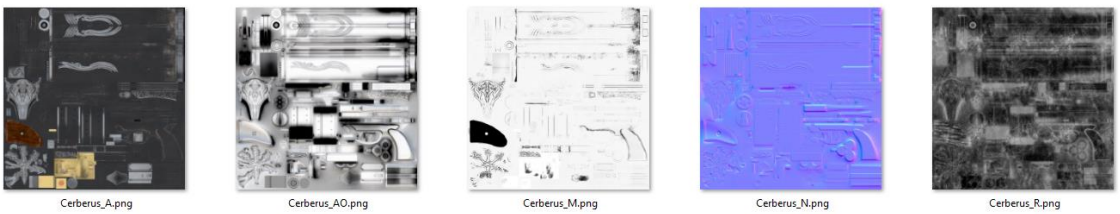

Figura 2. Texturas del modelo anterior: albedo, ambient occlusion, metalness, normal y roughness

Para facilitar la implementación de efectos PBR, se han definido dos UBO nuevos, que se pueden introducir en los shaders mediante las directivas:

```
$PBRLights
$PBRMaterial
```

El UBO `PBRLights` se define de la siguiente forma:

```
struct PBRLightSource {
  vec3 color;
  float intensity;
  vec4 positionWorld;
  vec4 positionEye;
  vec3 spotDirectionWorld;
  int directional;
  vec3 spotDirectionEye;
  int enabled;
  float spotExponent, spotCutoff, spotCosCutoff;
  vec3 attenuation;
  vec3 scaledColor;
};

layout (std140, binding=5) uniform PBRLights {
  PBRLightSource lights[4];
};
```

La definición es parecida al UBO de las fuentes de luz que hemos usado hasta ahora, con las siguientes diferencias:

- Las luces PBR sólo tienen un color
- Tienen una intensidad (de tipo float), que multiplica el valor de color para definir la cantidad de luz que emite la fuente
- El shader puede acceder a `scaledColor`, que contiene el producto del color por la intensidad

En esta práctica asumimos que todas las luces son puntuales.

Por otra parte, la directiva de los materiales PBR se expande a:

```
layout (std140, binding=4) uniform PBRMaterial {
  int textureCount;
};

uint numBaseColorTextures() { return bitfieldExtract(textureCount, 0, 3); }
uint numNormalMapTextures() { return bitfieldExtract(textureCount, 3, 3); }
uint numEmissionTextures() { return bitfieldExtract(textureCount, 6, 3); }
uint numMetalnessTextures() { return bitfieldExtract(textureCount, 9, 3); }
uint numRoughnessTextures() { return bitfieldExtract(textureCount, 12, 3); }
uint numAmbientOcTextures() { return bitfieldExtract(textureCount, 15, 3); }
layout (binding=0) uniform sampler2D baseColorMap;
layout (binding=1) uniform sampler2D baseColorMap1;
layout (binding=4) uniform sampler2D normalMap;
layout (binding=5) uniform sampler2D normalMap1;
layout (binding=8) uniform sampler2D emissionMap;
layout (binding=9) uniform sampler2D emissionMap1;
layout (binding=12) uniform sampler2D metalnessMap;
layout (binding=13) uniform sampler2D metalnessMap1;
layout (binding=16) uniform sampler2D roughnessMap;
```

```
layout (binding=17) uniform sampler2D roughnessMap1;
layout (binding=20) uniform sampler2D ambientOcMap;
layout (binding=21) uniform sampler2D ambientOcMap1;
```

Es decir, automáticamente se definen una serie de sampler2D, que tendrán asociada la textura correspondiente (si el modelo las define). Las funciones `num*Textures` permiten saber cuántas texturas de cada tipo define el modelo.
Las texturas son las siguientes:

- `baseColor`: es el color de albedo (color especular)
- `normalMap`: mapa de normales
- `emissionMap`: mapa de emisión, que permite que los objetos emitan luz por sí mismos
- `metalnessMap`: indica cómo de metálico es cada parte del modelo. Los valores están entre 0 (el material 100% dieléctrico) y un valor de 1 (100% metálico).
- `roughnessMap`: rugosidad del material, entre 0 y 1
- `ambientOcMap`: valor de oclusión ambiental (sirve para oscurecer las partes más escondidas del modelo, y sólo afecta a la componente ambiental).

Por ello, tu trabajo de esta práctica consiste en:

- adaptar el shader PBR del ejemplo visto en clase para su uso con texturas PBR (es decir, los parámetros de `albedo`, `metallic`, `roughness` y `ao` que antes se definían mediante *uniforms*, ahora vienen de texturas).
- Implementa el efecto de *bump mapping*, usando la textura correspondiente

Ten en cuenta que la textura de albedo está en el espacio gamma.

Ampliaciones

Incorpora más objetos con materiales PBR a tu escena.
Añade soporte de sombras arrojadas.