# Tarea7

## *Terrain & Fog*

### ***Implementar shaders para Terreno y neblina utilizando multitexturizado y niebla.***

**Instrucciones**:

- Leer:

  https://docs.gameloft.org/3d-training/#Multitexturing

***Para Terreno:***

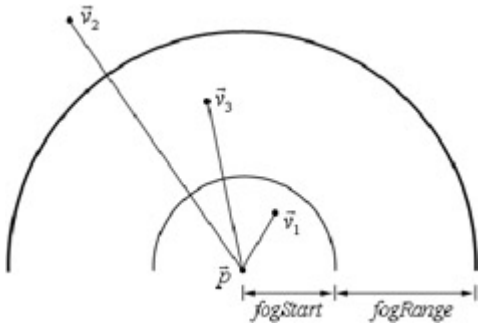
- Se usaran las siguientes Texturas:
  - Grass.tga
  - Dirt.tga
  - Rock.tga
- La combinación de las texturas dará el color del terreno. Para combinar las texturas se utilizara la textura Terrain_blend_map.tga.
- Para hacer el sampling de la texturas Grass, Rock, Dirt, se usara una multiplicación de las **uv * tillingFactor** (v_uvTiling) calculado en le vertex shader:
  - v_uvTiling = Las coordenadas de la textura obtenidas (uv) del archivo de geometría (.nfg) * el tiling factor. (v_uvTiling se mandara al fragment shader como varying).
  - Tilingfactor = Determina como se verá el terreno visualmente, eliminando el pixileado de la imagen, se debe de mandar como uniform a vertex shader.
- Para trabajar el Terrain_Blend_Map; las coordenadas de la textura (uv) se tomaran del archivo de geometría (aun si el "blend map" es más chico que la dimensión de tu terreno, las coordenadas para los fragmentos obtenidos durante la fase de interpolación en el pipeline permite diferentes pesos (weights) para hacer la combinación de terrain/grass/dirt).
- En el archivo RM.txt, el "wrapping mode" debe de ser GL_REPEAT para las 4 texturas.
- Activa los "mipmaps" para todas las texturas (activa y desactiva los mipmaps para observar errores visuales).

- Poner el filtro de minificación (minification filter) a GL_LINEAR_MIPMAP_LINEAR y el de magnificación a GL_LINEAR.

***Para la Niebla:***

- Leer:

  https://docs.gameloft.org/3d-training/#Linear_Fog




- Desde la posición de la cámara **P** hasta el **fogStart** (donde fogStart es la distancia desde la cámara), no habrá neblina, por lo tanto el color final será solo el color del terreno.
- Desde **fogStart** hasta inicio de **fogRange**, el color será calculado por la combinación del color del terreno y el color de la neblina (la combinación será calculada en base a un **factor** que toma en cuenta la distancia de la cámara).
- Desde **fogRange**, el color final será el color de a neblina.
- El cálculo para el factor de combinación es:

  float factor = **clamp** ((distanceCameraToFragment – fogStart) / fogRange, 0.0, 1.0)

  Donde:

  float distanceCameraToFragment = **distance**(cameraPos, positionOfFragmentInWorldCoordinates);

  *El **clamp** devuelve un valor entre valores establecidos como un mínimo y un máximo, según sea el más cercano si es menor al mínimo o excede al máximo.*

  *clamp = (value, minValue, maxValue);*

*La función **distance** calcula la distancia entre dos puntos.*

**Las distancias deben ser calculadas en el fragment shader obligatoriamente**, porque la formula utiliza un cálculo no-lineal, por lo tanto en si se trabaja en el vertex shader no se interpolara bien durante la interpolación lineal en el pipeline.

- La combinación del terreno se logra con la función **mix()**

  **vec4 colorFinal = mix(TerrainColor, fogColor, factor);**

  **La función **mix()** interpola entre dos valores. La interpolación lineal se hace entre **x** y **y** usando **a** como peso/factor entre ellos, el cálculo se hace de la siguiente manera: $x \cdot ( 1 - a ) + y \cdot a$*

- Cuando calcules el color final (glFragColor), poner 1.0 en el canal alpha (el terreno debe ser siempre opaco).

- Poner en el archivo SM.txt los valores para la neblina (fog), para evitar "harcodear" los shaders:

  *color*

  *start*

  *range*

  (Separar la entrada en SM.txt nombrando una sección **#FOG**)

  Ejemplo:

  .

  .

  ROTATION 0.0, 0.0, 0.0

  **#FOG**

  **FOG_COLOR 0.5, 0.5, 0.5**

  **FOG_START 2.0**

  **FOG_RANGE 10.0**

**Guía**:

1. Para Tomar los valores de las texturas usa un vec4.

   Ejemplo:

```
vec4 colorGrass = texture2D(u_s_texture[0], v_uvTiling);
vec4 colorDirt = texture2D(u_s_texture[1], v_uvTiling);
vec4 colorRock = texture2D(u_s_texture[2], v_uvTiling);
vec4 colorBlend_map = texture2D(u_s_texture[3], v_uv);
```

2. Para obtener el color final del terreno manejamos una textura como blendMap la cual tiene solo colores Rojo, Verde, Azul (**R**ed, **G**reen, **B**lue), multiplicaremos las texturas por cada canal de colores según corresponda.

   Ejemplo:

   terrainColor = (colorRock* `colorBlend_map`.x + colorDirt* `colorBlend_map`.y +…) / (`colorBlend_map`.x+ `colorBlend_map`.y+ `colorBlend_map`.z);

3. Para el valor de **Tillingfactor** prueba con los siguientes valores 16, 64, 128 y selecciona el que mejor te guste.

**Nota:**

Una manera de utilizar texturas en forma de array en el Fragment shader es:

1. Declarar el sampler en forma de array en el shader:

```
uniform sampler2D u_s_texture[4];

void main()
{
```

2. Tomar la localización de las uniform en el Shader.cpp:

```
textureUniform[0] = glGetUniformLocation(program, "u_s_texture[0]");
textureUniform[1] = glGetUniformLocation(program, "u_s_texture[1]");
textureUniform[2] = glGetUniformLocation(program, "u_s_texture[2]");
textureUniform[3] = glGetUniformLocation(program, "u_s_texture[3]");
```

3. Mandar las texturas como uniforms:

```
int n = 0;
	for(i = 0; i < nTextures; i++)
	{
		if(pShaders->textureUniform[i] != -1)
		{
			glActiveTexture(GL_TEXTURE0 + n);
			glBindTexture(GL_TEXTURE_2D, pTextureHandler[i]->GetTextureHandle());
			glUniform1i(pShaders->textureUniform[i], n);
			n++;
		}
	}
```

4. En el fragment shader quedaría algo como:

```
vec4 blend = texture2D(u_s_texture[0], v_uv);
vec4 dirt = texture2D(u_s_texture[1], v_uvTiling);
vec4 grass = texture2D(u_s_texture[2], v_uvTiling);
vec4 rock = texture2D(u_s_texture[3], v_uvTiling);
```

**Resultado**: