DELIVERY 2

Víctor Blas & Franco Rivadeneyra

Post processing		Max points: 3
Color Space Handling	Make textures in Unity assume linear input. Handle yourself the gamma correction for your PBR shader (not in post processing, in the real shader).	0.5 points
Tone mapping	Make an Eye adaptation effect based on screen luminance without using the Unity post process (implement all the steps yourself)	2 points
Vignetting/Blur/Pixelate	Using your own implementation create a context aware implementation.	1 point
Bloom	Using blur implement a bloom effect and make it work even when using your own pbr shader	2 points
Shader in Unity		Max points: 3
Adding features to our PBR shader	Make your own PBR Shader cast and receive shadows (using unity shadow pass) & be able to use textures.	1 point
Adding reflections	Make the shader be able to receive data from the reflection probes on the scene. If reflections are correctly sampled taking into account PBR values this exercice is worth 2 points.	1 or 2 points
Implement multiple light handling & spotlight	MAke your own PBR Shader react to more than 1 light and implement a new type of light for it: spotlight	2 points
Create materials for the whole scene using your material	Using your shader, create materials that showcase diverse physical properties & any other extra implementations made. Only opaque & objects not used for other exercises are required to use your shader	2 points
Compute Shaders		Max points: 2
Make a GPU particle system	Create a simple GPU particle system for some effect on the scene and calculate it's motion using a compute shader (not using shader GPU particles)	1 point
Boids Implementation	Create a Boids AI that manages the	2 points

	movements of a group of animals in the scene. They should follow all the classic boids rules (avoidance, cohesion, alignment) & react to the scene geometry (don't allow your Boids to go through stuff).	
Additional Implementations		Max points: 2
Vertex shader animation	Create a vertex shader that animates a certain object (or objects) of the scene. Complexity will be taken into account. Explain on the read me what effect you want to create and how you do it.	1 point
Texture animation	Using uv displacement create an animation on some texture of the scene. If only a simple animation is done punctuation will be 0.5. If extra animations, blendings or other effects are included the maximum punctuation can go up to 1 point	0.5 points or 1 point
Triplanar textures	Implement this kind of texturing in some scene object. If only colors or textures are swapped 0.5 points max, if other implementations are created punctuation can go up to 1 point	0.5 points or 1 point
Texture Blending	Create a texture blending effect either by vertex color, texture or other factor. Complexity will be taken into account.	1 point
Emissive Mat	Create an emissive material that affects the scene meaningfully. Explain in the READ ME de desired effect	0.5 points
Transparent Mat	Create a transparent material that affects the scene meaningfully. Explain in the READ ME de desired effect	0.5 points
Rogue Exercise		Max points: 2
Any of the previous exercises		

Post Processing

- **Pixelate**
    - Ubicación:
        - Archivo: Assets/Universal Pipeline
        - Escena: Main Camera
    - Archivos:
        - Blit -> ScriptableRenderFeature
        - Pixelated -> Shader Graph
        - PixelatedCamera -> Script (lógica de cuando se aplica)
        - PixelatedMat -> Material
    - Cuando la cámara se aleja mínimamente de su posición original, distorsionamos todo mediante pixelado, como si nos advirtiera que nos estamos saliendo del perímetro permitido.
    - El shader graph se basa en, a partir de los UVs, multiplicarlos por un float, que determinará la cantidad de pixeles, para luego eliminar su parte decimal con el floor y así, dividiendo otra vez por el mismo float, obtener dicho número de pixeles con los colores aproximados.

Shader in Unity

- **Adding features to our PBR shader**
    - Ubicación:
        - Archivo: Assets/Shaders in Unity
        - Escena: Meshes/Nature/STONES/stone_02 (1)
    - Archivos:
        - PBR_Ult -> Shader
        - Shadow_Mat -> Material
    - Esto está aplicado en una piedra que hay al lado de un barril, en la que se contempla una sombra de dicho objeto. Usamos un PBR modificado al nuestro, con la diferencia principal que supone el cambio del CSPROGRAM por el HLSLPROGRAM

Compute Shaders

- **Make a GPU particle System**
  - Ubicación:
    - Archivo: Assets/ComputeShaders/Humo
    - Escena: HumoManager
  - Archivos:
    - Humo -> Compute Shader
    - HumoManager -> Script
  - Hemos hecho un sistema de partículas muy simple, el Compute Shader se encarga de ascender las partículas y devolverlas a su posición original cuando superan la altura máxima. Se pueden ajustar la cantidad de partículas y la altura máxima a la que pueden llegar.

- **Boids Implementation**
  - Ubicación:
    - Archivo: Assets/ComputeShaders/Gaviota
    - Escena: BoidsManager

- Archivos:
  - Boids -> Compute Shader
  - BoidsManager -> Script
- Hemos implementado una bandada de gaviotas que sigue las fuerzas de flocking, separación/cohesión/alineación, junto con las fuerzas de esquive, que solo se aplican a la montaña. Se pueden ajustar casi todos los parámetros, como la colisión, la velocidad, incluso podemos desplazar el target, que es un objeto a parte llamado BoidsTarget.

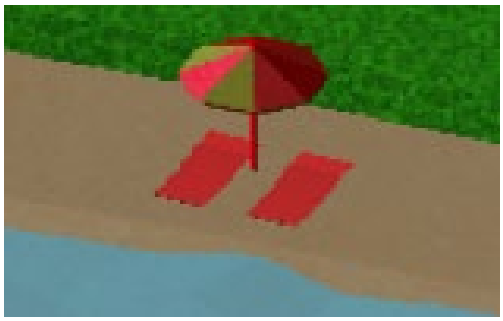
Additional Implementations

- **Vertex Shader Animation**
  - Ubicación:
    - Archivo: Assets/Materials/Peces
    - Escena: Meshes/Animals (en fish y shark)
  - Archivos:
    - MovementAnimation_Graph ->Shader Graph
    - Peces -> Material
  - Hemos hecho la animación de nadar del tiburón y del pez. Para crear esto hacemos un lerp entre la posición del modelo, la posición animada y un vector 3 que será la dirección nadará el pez. Para conseguir la posición animada se creará con un nodo position que luego se dividirá en los canales RGBA y nosotros utilizaremos el canal B en este caso, después crearemos el nodo add dónde se multiplicará el tiempo para sacar la velocidad. Por otra parte, para la posición del objeto habrá que dividirse de nuevo los canales RGBA, esta vez se utilizará todos, se combinarán y se unirá todo en un nodo add donde sera multiplicado por el seno. Por último, se juntaría todo con un nodo add para enlazar todo hacia el nodo lerp y se pondría en la vertex position.

- **Triplanar Textures**
  - Ubicación:
    - Archivo: Assets/Materials/Tripplanar Mapping
    - Escena: Meshes/Nature/mountains
  - Archivos:
    - Triplannar_Shader ->Shader Graph
    - Shader Graphs_Triplannar_Shader -> Material
  - Hemos usado este método para la estructura grande, la montaña, las texturas que hemos usado es de césped y tierra. Primero crearemos un nodo position en el espacio de mundo, donde se dividirá en tres Vectores 2 donde cada uno de ellos se pondrán las texturas que se ha mencionado antes. Por otro lado, se creará un nodo de vector normal donde pasará por un nodo absolute que luego se juntará al power donde se normalizará y se multiplicará con un vector 3. Luego se dividirá en tres nodos de multiplicación, y se juntará lo que teníamos anterior de las texturas con el nodo add y por último se añadiría el resultado al base color del fragment.

- **Emissive Mat**
  - Ubicación:
    - Archivo: Assets/Materials/Emissive
    - Escena: Meshes/Nature/towel-beach
  - Archivos:
    - Emissive ->Shader Graph
    - Shader Graphs_Emissive -> Material

- Como ejercicio complementario, quisimos implementar el emissive material ya que lo encontramos sencillo. Con un fresnel, que se multiplica por el color que queramos, en nuestro caso el rojo, obtenemos el output que llevaremos a la propiedad emission del fragment, mientras que mantenemos el base color. El resultado aparece en las toallas que hay en la escena.



- **Transparent Mat**
  - Ubicación:
    - Archivo: Assets/Materials/Nubes
    - Escena: Meshes/Nature/CLOUDS
  - Archivos:
    - Nubes_ShaderGraph ->Shader Graph
    - Nubes -> Material
  - Este material lo hemos implementado para las nubes. Para crear esto lo que hemos hecho es usar el fresnel con una variable que se puede modificar, luego se multiplica por el color que quieras y luego lo uniríamos al base color del fragment.