

Final Delivery

Materia: Mecánicas de Juego Avanzadas

Gerardo Daniel Santos Cuevas

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
Boids Implementation	Create a Boids AI that manages the movements of a group of animals in the scene.	2 points
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
Rogue Exercise		Max points: 2
Any of the previous exercises		
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

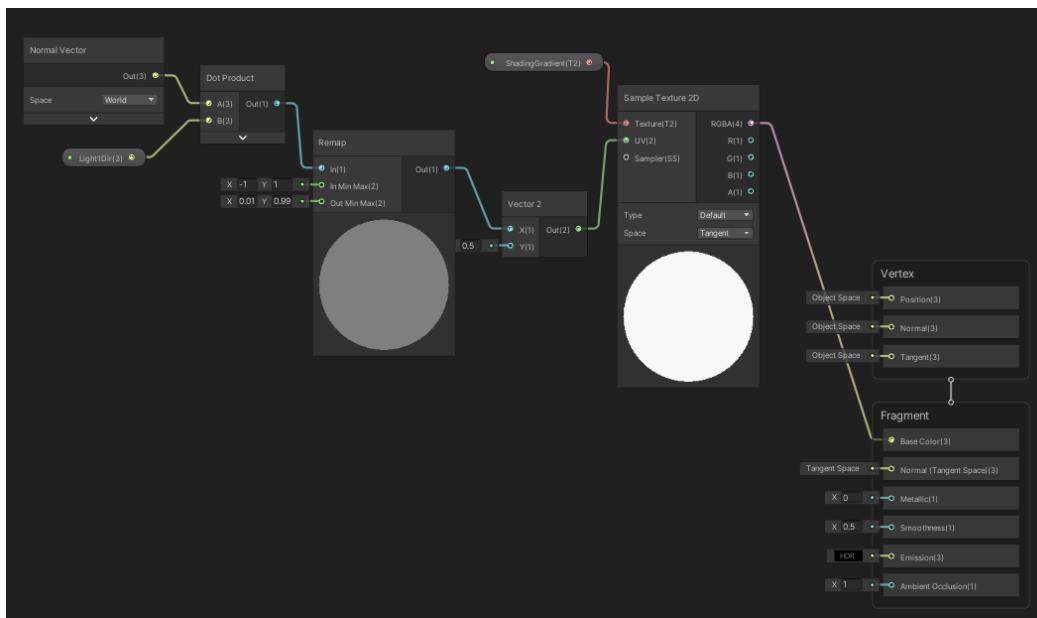
Implement multiple light handling & spotlight

MAke your own PBR Shader react to more than 1 light and implement a new type of light for it: spotlight

2 points

Ubicación:

ShaderScenary\Assets\Scenes\Delivery2\Delivery2_Profiles/PBRLightHandler



* Nodos explicados de izquierda a derecha

Nombre del Objeto en la escena: Light Handling; Contiene un barril y una spotlight que le complementa el efecto pedido.

Normal Vector: Para tener el vector de la superficie del objeto

Light Dir: Para tener el vector de dónde se encuentra la luz creada

Dot Product: Usamos esta función para darle un valor único a el vector de la normal y de la lightDireccion.

$$\text{DotProduct} = \text{LightDir} + \text{NormalVector};$$

Remap: Para darle un valor a el producto punto en X y Y, así tiene una limitante entre -1 y 1

Vector2: Para darle unas nuevas coordenadas a la texturas del mapa UV del objeto

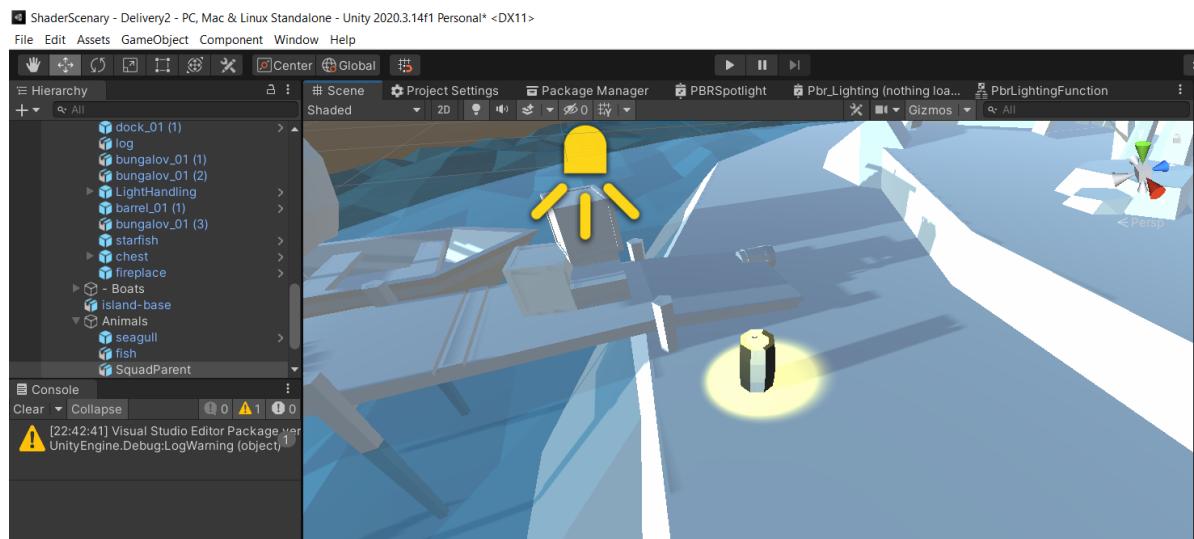
$$\text{Vector2} = \text{Vector2}(\text{Remap}, 0.5);$$

Shading Gradient: Es una textura en la cual tendrá como base el objeto

Sample Texture: Combina las UV dadas por los nodos anteriores más la textura añadida, se agrega a la base color del shader.

SampleTexture = ShaderGradient + Vector2;

Explicando el shader se basa en tener luz de la luz creada, el objeto es afectada por esta en sus coordenadas UV, mas aparte en la escena tiene una spotlight que le afecta, ya sea con color e iluminación, haciendo que le afecte más de una luz a este shader, contanto la spotlight.



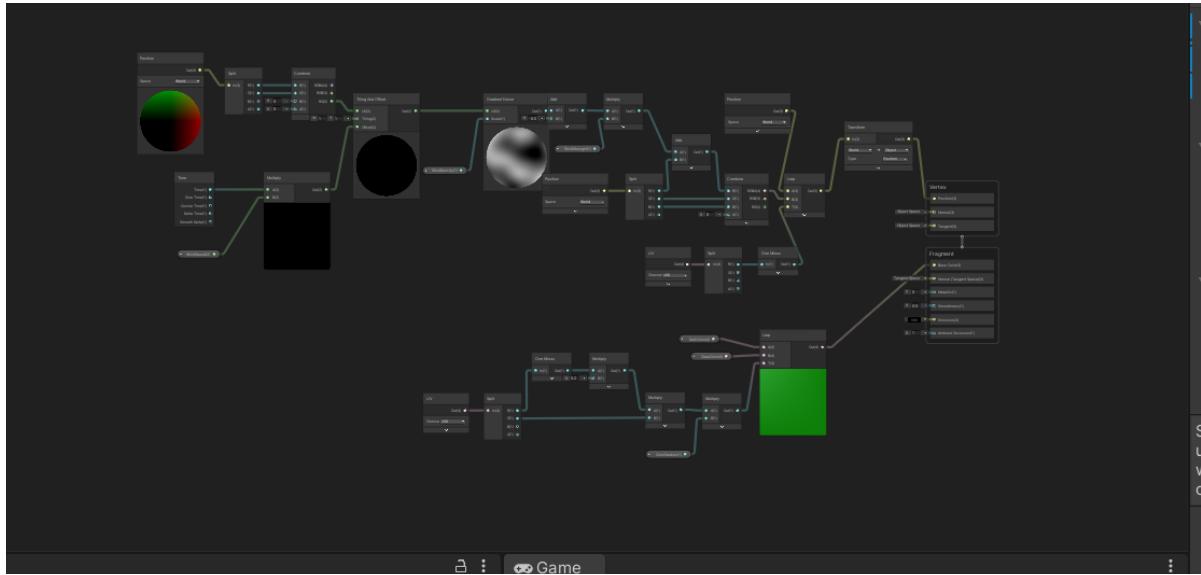
Create materials for the whole scene using your material	Using your shader, create materials that showcase diverse physical properties & any other extra implementations made. Only opaque & objects not used for other exercises are required to use your shader	2 points
--	--	----------

Ubicacion de Scripts y Prefab:

ShaderScenary\Assets\Scenes\Delivery2\Delivery2_Profiles\Scene Materials

Nombre de los Objetos en la escena a destacar: Mountains, Palms, Grass

Hojas de las palmas



* Nodos explicados de izquierda a derecha

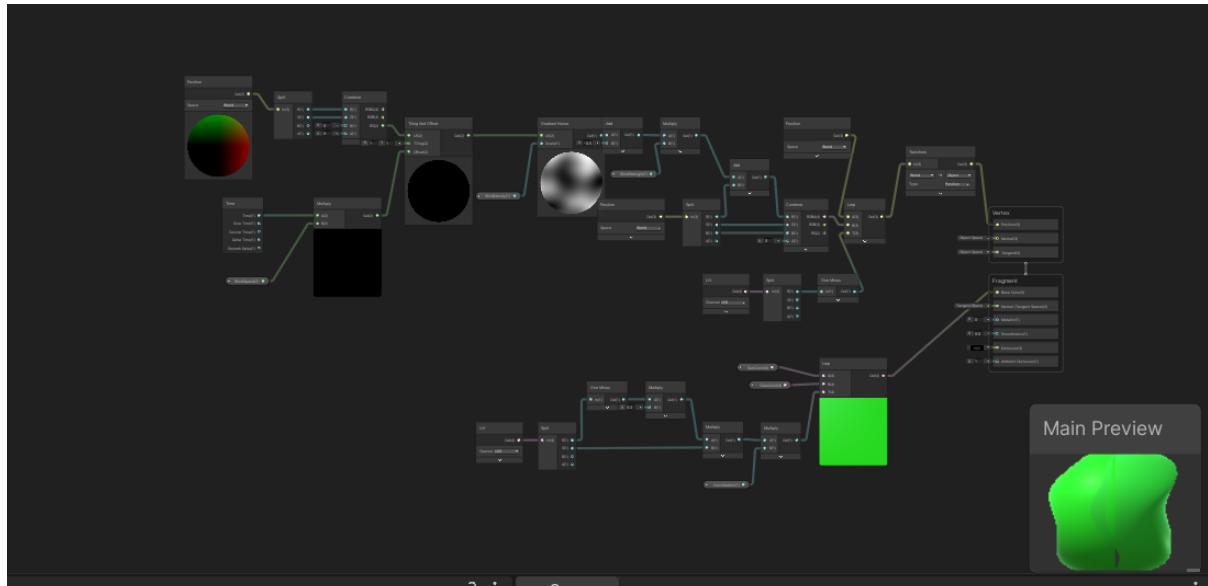
Para comenzar con el shader se necesita tener su **Position** y de este sacar con **split** sus posiciones en x,y que son los que vamos a mover, a estos le multiplicamos el **Time** por un valor para hacer que se muevan los uv con el offset.

*Tiling and offset = (uv(Combine(Position.x, Position.y)), Tiling(1,1), Offset(Time * Speed))*

Después le agregamos el **Gradient noise** para generar ruido, para posteriormente combinar estos valores en la posición del objeto. para finalizar y juntar los valores en Y que es donde se va a mover, x será la posición normal del objeto al igual que en z, pero tomando los uv como referencia. Todo lo pasamos al **Transform** que lo pasara del mundo al objeto y así darle el valor final de la posición al objeto.

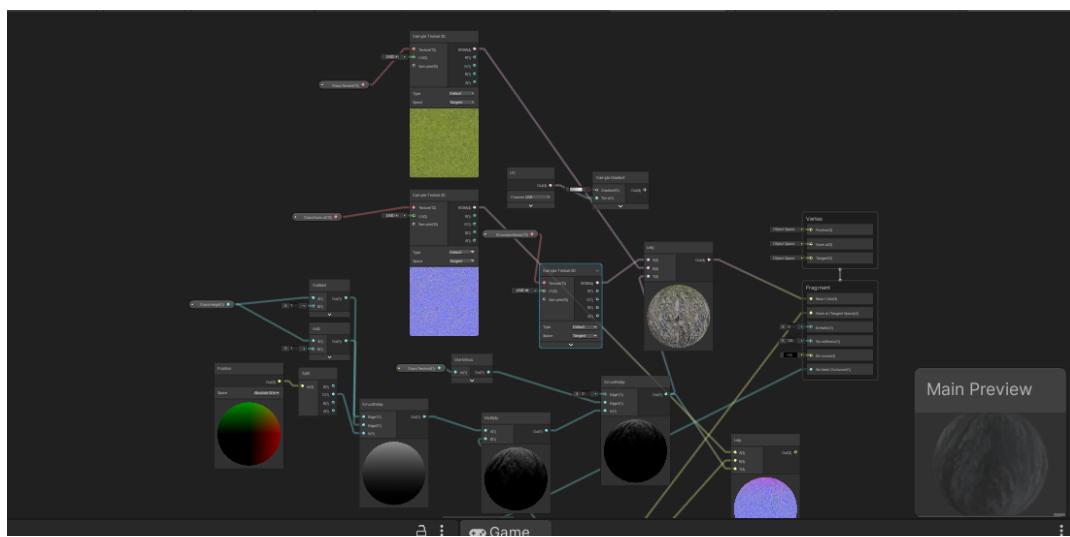
Transform.Gameobject.position = Vector3(Position.x, RuidoCombinadoconeltiempo.y, Uv.z);

Pasto

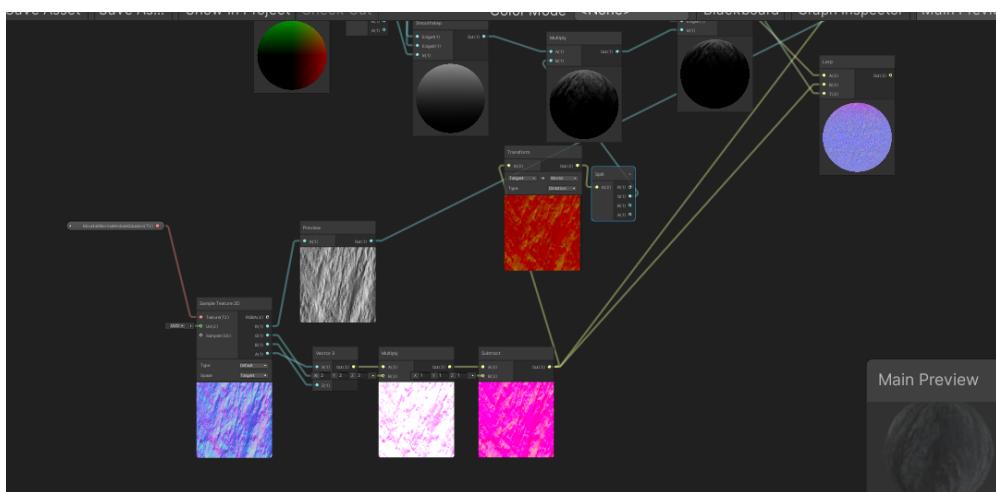


Para el pasto se realizó casi el mismo procedimiento que el de la palma, solo se redujeron algunos valores para que este no se moviera con tanta rapidez y sea más lento y natural.

Montaña



Para la montaña la teoría principal es poder juntar dos texturas en las cuales con las referencias de los uv se pueda aplicar una sobre la otra en cierta posición del objeto, ya sea si la parte del objeto es muy inclinada, domine una textura y en las partes más planas domine la otra.



La parte de abajo se enfoca en distribuir las uv de las normales para así poder decirle cómo lo va a dibujar. Para esto necesitamos multiplicar los valores de la normal y usar el **Subtract** para poder darle las normales al material.

Complementando y agregando otros materiales a la escena, así se logra ver completamente esta maravillosa isla low poly :D

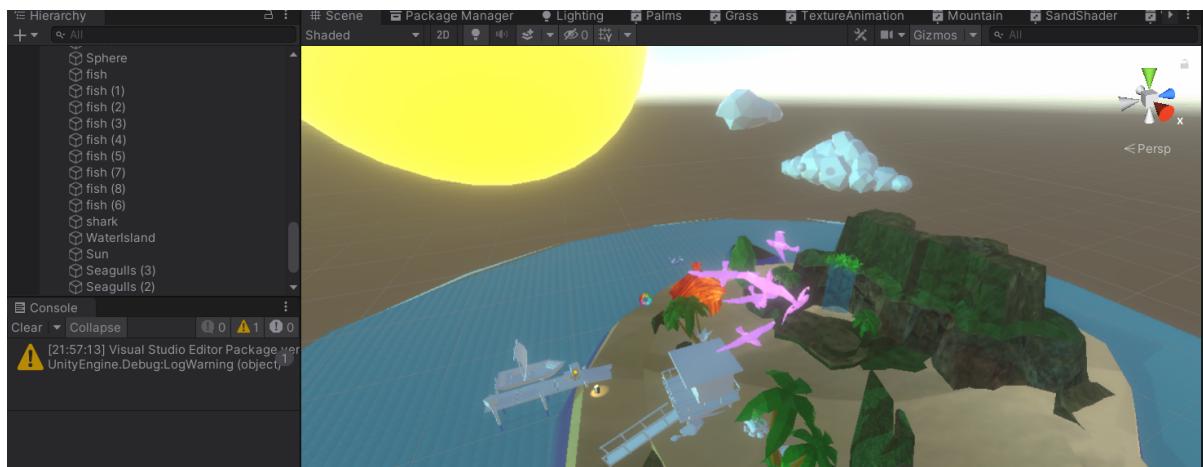


Boids Implementation	Create a Boids AI that manages the movements of a group of animals in the scene.	2 points
----------------------	--	----------

Ubicacion de Scripts y Prefab:

ShaderScenary\Assets\Scenes\Delivery2\Delivery2_Profiles\BoisAI

Nombre de los Objetos en la escena: Fish



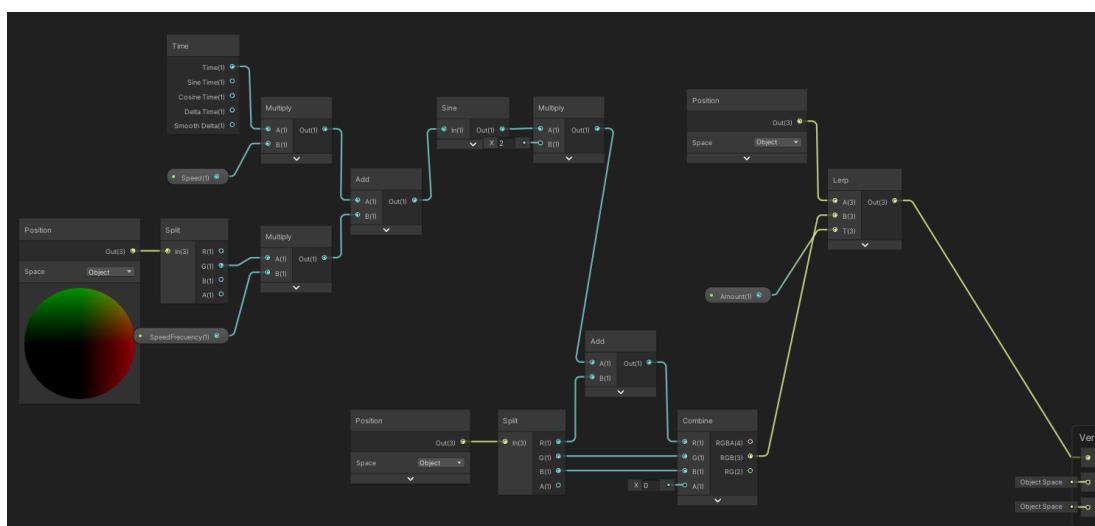
Hice los Boids con el grupo de Gaviotas, que se pudieran volar juntas a una dirección mientras hacían su comportamiento de seguirse entre estas, tener una cohesión, más una alineación para que no se pegaran, estos se repelen cuando están muy cerca del otro.

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
-------------------------	--	---------

Ubicación:

ShaderScenary\Assets\Scenes\Delivery2\Delivery2_Profiles\VertexAnimation

Objeto en la escena: Shark



* Nodos explicados de izquierda a derecha

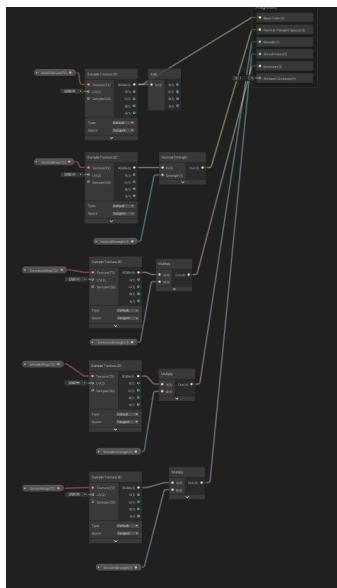
Primero necesitamos la **posición** del objeto a mover y en qué eje, para esto usaremos **split** y lo multiplicaremos en el eje G que en este caso seria para la rotación en Y * **speedFrecuency** para agregarlo al seno del objeto.

Sumado a esto para crear el movimiento usaremos el nodo de **Time * Speed**.

Por otro lado también necesitaremos dividir su posición para poder agregarle el resultado de la parte de arriba y así poder combinarlos.

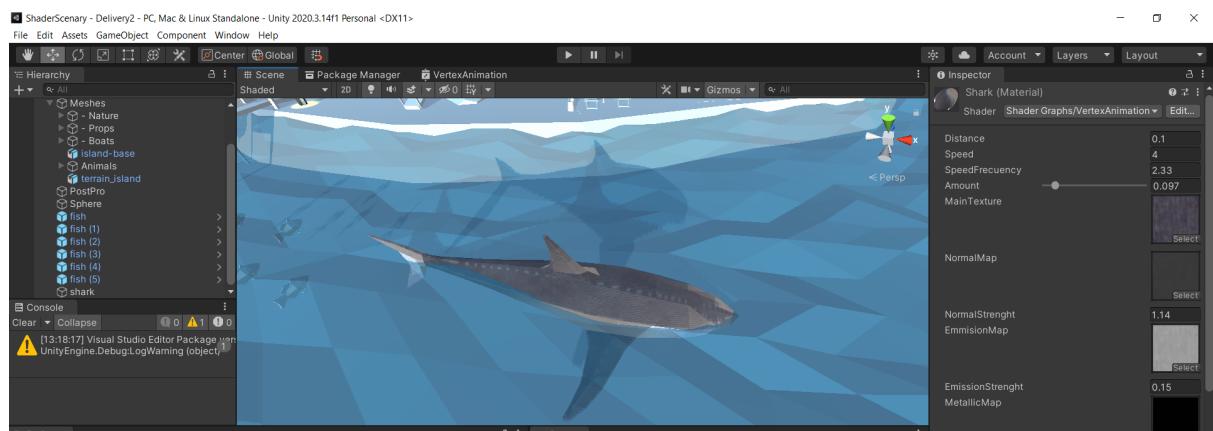
Una vez conseguidos todos estos datos los pondremos en un lerp para saber en qué posición afectará al objeto y que tanto.

GameObject.Position = Lerp(position, Amount)



En esta parte se agregaron las propiedades de la **Textura**, **Normal Map**, **Emmision Map**, **Metallic Map**, **Smooth esMap**. Para darle un mejor efecto y poder moldearlo al gusto y quede un mejor efecto.

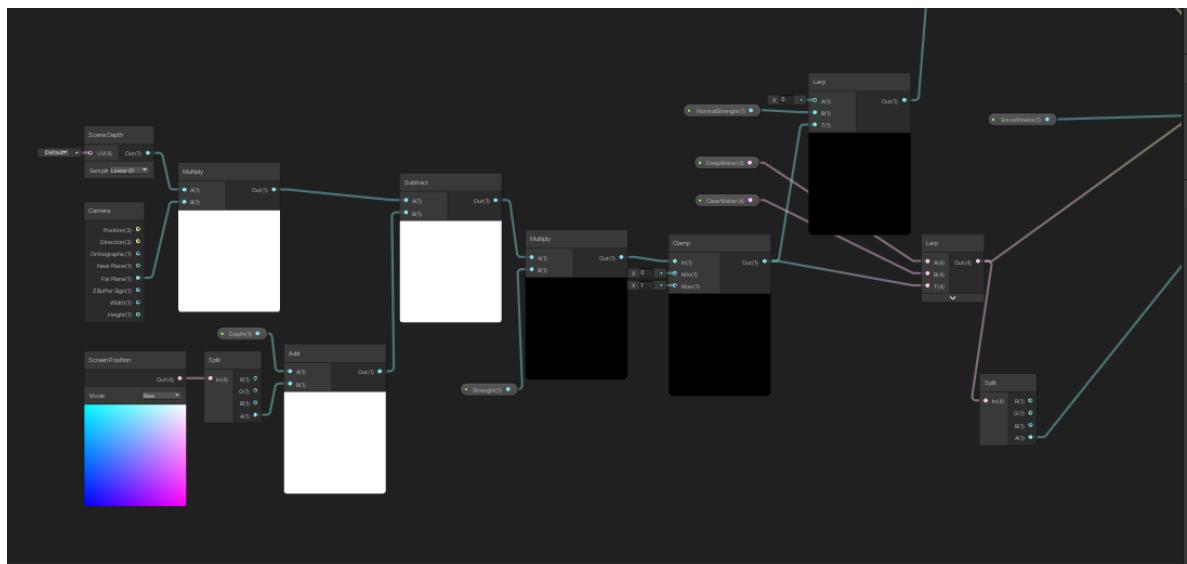
Dando el siguiente resultado del movimiento de un tiburón dentro del mar:



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
-------------------	---	-----------------------

Ubicación: ShaderScenary\Assets\Scenes\Delivery2\Delivery2_Profiles\Texture Animation

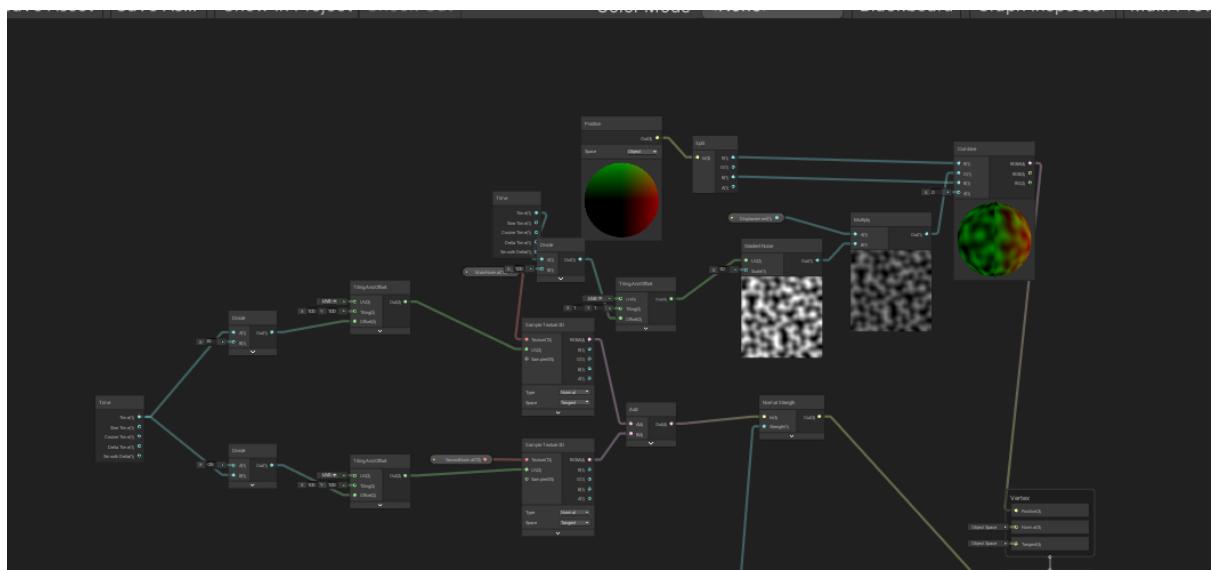
Objeto en la escena: Water Island



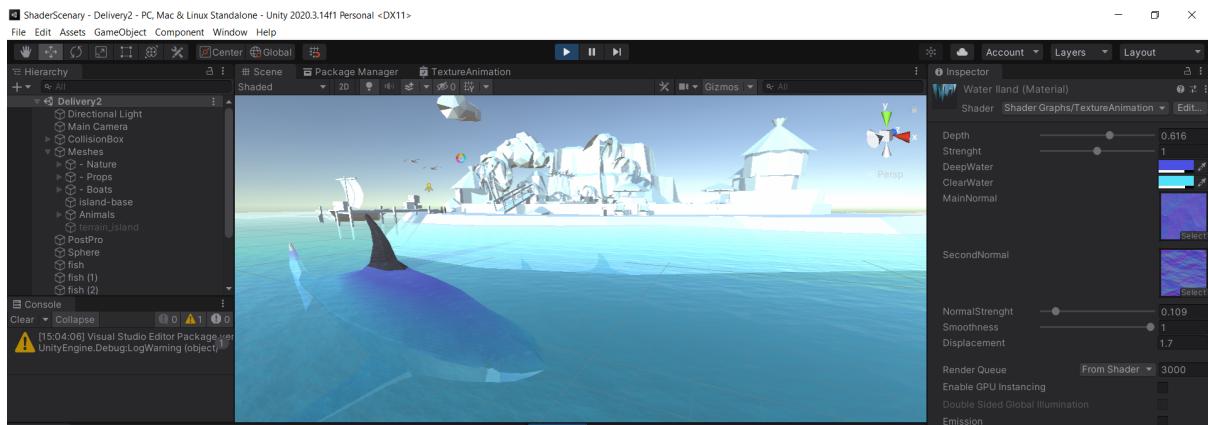
* Nodos explicados de izquierda a derecha

Este shader se divide principalmente en dos partes, esta primera parte que consiste en tener la posición de la **Cámara** para poder multiplicarlo por la **Scene Depth**. Del otro lado sacaremos la **Screen Position** y con el **split** Sacaremos su alpha para posteriormente multiplicarlo, con el **Subtract** atraemos estos valores. Después en el **Lerp** le agregamos los colores que tendrá el agua cuando haya un objeto cerca y otra más de profundidad dando así el **BaseColor**.

*BaseColor = (ClearWater + deepwater) + Screen Position * Depth.*



Ahora nos vamos a encargar de mover el agua con sus **UV** y tener ese efecto de olas. Empezaremos con **Time** para agregar movimiento, después lo dividiremos para darle un efecto más natural. Después en las **Textures** agregaremos las normales en las que se va a basar las uv para darle movimiento. A estas les agregamos **Noise** para darle un efecto más desordenado al objeto, la suma de todo esto la agregamos a la **position** del shader para que se pueda mover, dando el siguiente efecto que es el de las olas de mar, así como se ve y como se mueve:

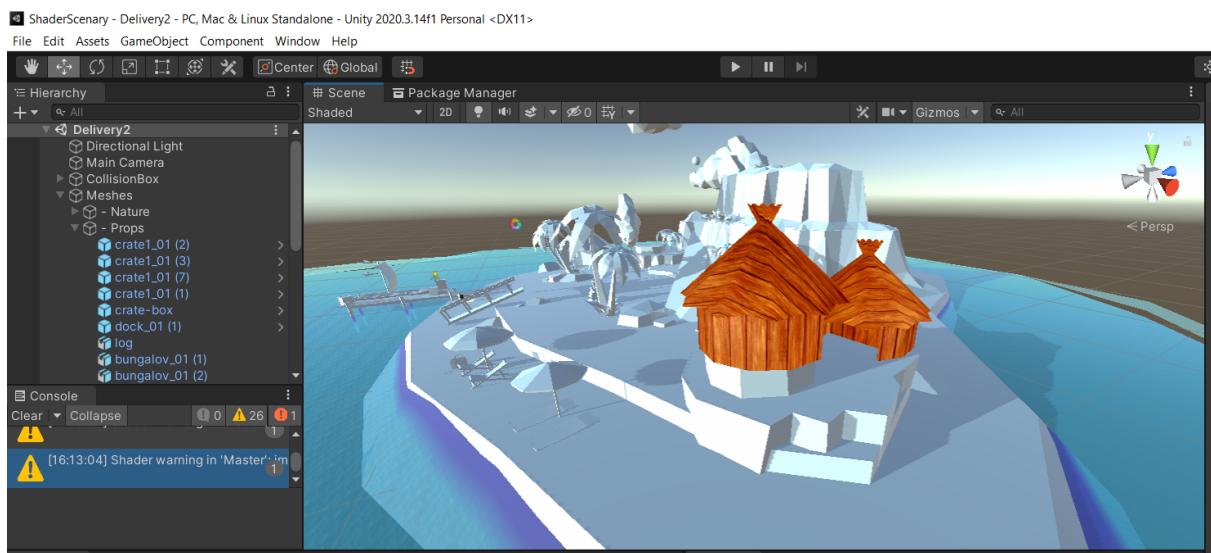


Rogue Exercise		Max points: 2
Any of the previous exercises		
Texture Blending	Create a texture blending effect either by vertex color, texture or other factor. Complexity will be taken into account.	1 point

Ubicación del material junto con el script: Shader

Scenery\Assets\Scenes\Delivery2\Delivery2_Profiles\Texture Blending

Objetos en la escena: Bungalow

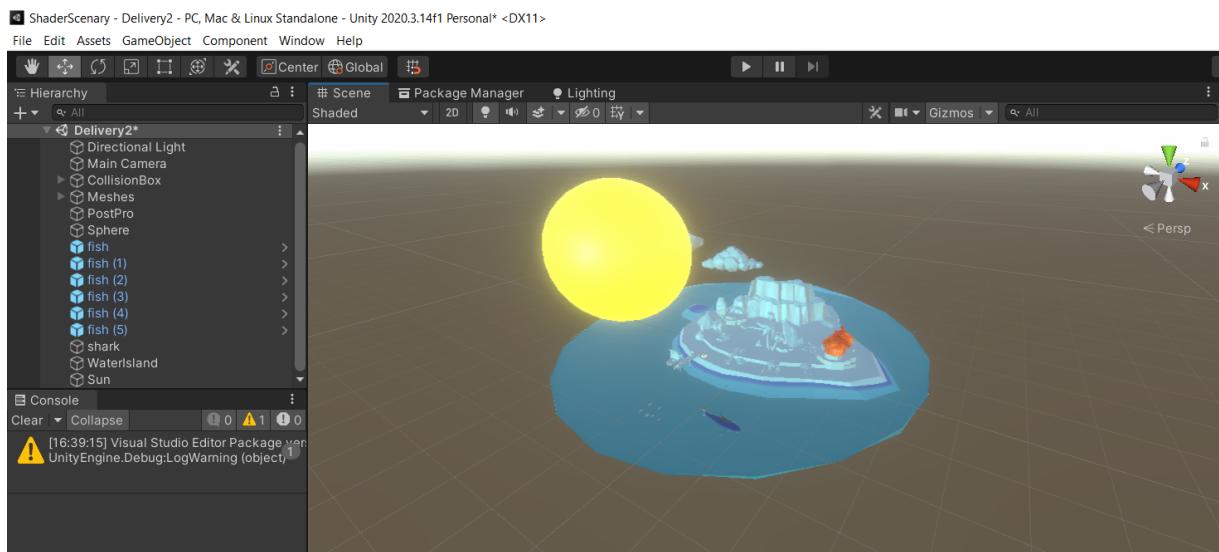


El efecto del blending en este shader hace que junte el valor de las dos texturas y las vaya intercalando con el _Sin Time y hace que le de ese efecto de transformación de cabaña de paja a de madera.

Emissive Mat	Create an emissive material that affects the scene meaningfully. Explain in the READ ME de desired effect	0.5 points
--------------	---	------------

Ubicación: Scenery\Assets\Scenes\Delivery2\Delivery2_Profiles\Emissive Material

Objeto en la escena: Sun



El efecto deseado es que sea un sol que afecte a toda la isla con su emisión y brillo.

Cualquier duda o aclaración estaré al
pendiente :D