Abril 2019

Lightweight Render Pipeline

En primer lugar, me interesé por utilizar las más actuales tecnologías para la creación de *shaders*, por lo que pensé en usar una pipeline personalizada y el nuevo sistema de codificación de *shaders* llamado “*Shader* *Graph*”, entre las dos plantillas que ofrecía Unity, “HDRP o LWRP”, Me decanté por usar “LWRP” por simpleza para la práctica, ya que el uso de un *Custom* *Render* *Pipeline* era algo no obligatorio.

Uno de los problemas de usar un “CRP” era que los standard *shaders* de Unity no son compatibles con dichas pipelines, lo cual se puede solucionar con una serie de instrucciones en el *shader* para el renderizador. Además, Unity ofrece una amplia documentación sobre estos pipelines, e incluso múltiples *templates* oficiales para adaptar *shaders* a este sistema, indicando todos los *includes* necesarios, así como especificaciones e instrucciones a GPU, facilitando así mucho la implementación.

(Nota: ya que gran parte del shader( debido a requisitos de compatibilidad para la “Scriptable Render Pipeline”) implican añadir múltiples líneas de includes y parámetros ofrecidos por unity3D, y ese código es propio de los autores del motor, a sabiendas del extremismo tras la corrección de la nomenclatura, indicar que he notado una poca incoherencia de nomenclatura en su código, a pesar de esto, no me parece correcto corregirlo.)

¿Shader graph o Coded Shaders?

Ya que el enunciado del proyecto indicaba que uno de los *shaders* podría realizarse sobre “*Shader* *Graph*”, la nueva herramienta de codificación visual de *shaders* de Unity, consideré que la mejor manera de demostrar mis capacidades, sería la de realizar cada *shader* en un entorno diferente, teniendo así que adaptarme al sistema PBR de *shaders*, así como la codificación de archivos “.*shader*”, consiguiendo que ambos *shaders* convivan bajo el mismo proyecto, teniendo que configurar el “.*shader*” para su funcionamiento en un LWRP.

Texturas Adicionales

La filosofía que he seguido durante la programación era la de usar el minimo de texturas posibles, usando siempre algoritmos matemáticos para poder representar los efectos gráficos deseados. A pesar de esto, algunas texturas si eran obligatorias, tales como el Albedo de los modelos, así como capas de mascara para poder aplicar efectos sobre partes individuales, limitando a 4 texturas el proyecto.

Algunas de las operaciones matemáticas usadas para efectos gráficos han sido:

* El efecto de carga se consigue mediante un mapa de UV, la carga lineal se consigue usando solo un canal del mapa de UV y aplicando un step a dicho valor, mientras que la carga radial se consigue con un remap del mapa de UV y el cálculo del arcotengente al cuadrado, si después hacemos un remap de –π a π conseguimos un perfecto degradado al cual aplicando un step conseguimos una carga circular. (Nota: Ambos efectos de carga están al 33% de su carga total, se puede ver que son completamente relativos, ya que ambos usan la misma variable para ir del 0% al 100%).



* Las texturas para dar el efecto de “Magia” se consiguen mediante la fusión de los elementos “Voronoi Map” y un “Gradient Noise” mediante un blend, mediante la creación de un vector2 a partir del tiempo, multiplicando la Y y la X por diferentes valores, aplicando el resultado final al tiling de las texturas, podemos conseguir un desplazamiento en cualquier dirección a cualquier velocidad. Posteriormente invierto los colores y el resultado “mágico” es muy bueno visualmente, ya que además se aplica el seno a partir del tiempo para generar un parpadeo en columnas, haciendo así que también sea fiel al ejemplo del proyecto.



* El shader codificado trata de ser lo más completo posible, ya que emite sombras, es afectado por iluminación global y multiples focos así como propiedades "albedo", "metallic", "specular", "smoothness", "occlusion", "emission" y "alpha". El cambio de colores se consigue a partir de la textura y una máscara de recorte, así como de un color llamado “\_DetailColor”, mediante una multiplicación del color por los valores de la textura y la aplicación de la máscara de recorte, así como un blend de texturas, podemos conseguir que las partes grises mantengan el color, mientras que las de color puedan variar.



Documentación

Documentación de Unity sobre LWRP Shaders:

<https://docs.unity3d.com/Packages/com.unity.render-pipelines.lightweight@4.6/manual/index.html>

LWRP vs Standard Unity:

<https://docs.google.com/document/d/1MgoycUhS9xQKXxbTy1yHt7OCByI10rds3TyRBCSlFmg/edit>

LWRP y HDRP Unity Templates:

<https://github.com/Unity-Technologies/ScriptableRenderPipeline/pulls>

Template Explanation of PBR / CRP:

<https://blogs.unity3d.com/es/2018/04/26/how-to-get-the-most-out-of-the-new-unity-project-templates-in-2018-1/>