

**Propósito**

El propósito que se tiene es el poder realizar lo que son cuatro efectos en sprites, donde estos deberán tener el outline, Fresnel, glitch y entintado y que puedan ser modificados, ya sea por un cambio de color, por su grosor o por su velocidad y todos tienen que funcionar de manera equitativa

**Introducción**

En el proyecto entregado se puede observar cómo fue la realización del trabajo en la materia de tópicos de física, por todo lo que se recorrio y por las dificultades que hubo y como se solucionaron

**Desarrollo**

La manera en que se planteo hacer el trabajo fue primero que nada creando el proyecto en LWPR, mas que nada porque es lo que hemos estado usando y lo que mas aprendimos en clase y porque es de alto rendimiento, ya creado el proyecto, seleccionando ls sprites que serán usados y guardados en otra carpeta aparte y creando también la carpeta donde ira el trabajo principal, en un principio se tenia planeado que los efectos serian en un solo Sprite, pero debido a las dificultades y la gran cantidad de errores que daba hacer eso, me decidí por hacer dos sprites con dos efectos en cada uno, donde se estuvo buscando previamente el efecto de cada uno y la funcionalidad de los mismos.

Primero se optó por la creación del Sprite con la función del outline y el entintado, y el otro Sprite se fue por el efecto glitch y el Fresnel, debido a los problemas que había al tratar de juntar todos, se opto mejor por dejarlos por separado.

**Conclusión**

Para concluir podemos decir que el trabajo que se tiene por detrás del shading, es mucho tiempo y dedicación, pues uno tiene que saber para qué sirve cada herramienta que unity te presenta, para mayor facilidad y efectividad.

**Breve historia del shader**

En el 2000 la serie 2 de tarjetas GeForce permitía a la GPU hacerse cargo de funciones de transformación e iluminación que hasta ahora debía hacerlas la CPU, sin embargo, no fue hasta la GeForce 3 (2001) que se incluyó la posibilidad de programarlas con la primera versión del modelo de sombreado. Existen numerosas versiones, se debe tener en cuenta que cuanto más reciente es la versión más limita el número de tarjetas gráficas sobre las que el programa puede operar correctamente.

**Diferentes lenguajes en el sombreado**

Hoy en día se puede lograr sin el uso de algún lenguaje de programación, ya que ahora tenemos herramientas que nos facilitan mucho esos temas, pero si tenemos que hablar que lenguajes se utilizaban y aun se utilizan para el mismo serian:

HLSL es la implementación propiedad de Microsoft, la cual colaboró a Nvidia para crear un lenguaje de sombreado. Este lenguaje se debe utilizar junto a DirectX (la primera versión para la que se puede utilizar es DirectX 8.0). Anteriormente al DirectX 8 (DirectX 7, 6, 5...) se utilizaba otro método el cual era más complicado y complejo para ser utilizado. (Entre lo que era el lenguaje, creación de objetos, sonidos, partículas, entre otras).

También esta GLSL es el lenguaje desarrollado por el grupo Khronos. Está diseñado específicamente para su uso dentro del entorno de OpenGL. Sus diseñadores afirman que se ha hecho un gran esfuerzo para lograr altos niveles de paralelismo. Su diseño se basa en C y RenderMan como modelo de lenguaje de sombreado.

CG lenguaje propiedad de la empresa Nvidia resultante de su colaboración con Microsoft para el desarrollo de un lenguaje de sombreado. Su principal ventaja es que puede ser usado por las APIs OpenGL y DirectX. Otra ventaja de este lenguaje es el uso de perfiles. Estos lenguajes no son totalmente independientes del hardware por lo tanto es recomendable crear programas específicos para diferentes tarjetas gráficas. Los perfiles de CG se encargan de elegir para su ejecución el más adecuado de los programas disponibles para el hardware.

Y hay mas por detrás pero tienen un uso menos común al de los principales mencionados anteriormente, dado a que estos tienen una mayor facilidad y utilidad al lograrse.

Ronja (2019). “Ronja´s shader tutorials”. Recuperado el 18 de febrero del 2019. Página web: <https://www.ronja-tutorials.com/>

David León (2015). “Next-Gen Cel shading in Unity 5”. Recuperado el 18 de febrero del 2020. Pagina web:<https://www.gamasutra.com/blogs/DavidLeon/20150702/247602/NextGen_Cel_Shading_in_Unity_5.php>

Wilmer Lin (2019). “Shader Graph in Unity for beginners”. Recuperado el 18 de febrero del 2020. Página web: <https://www.raywenderlich.com/3744978-shader-graph-in-unity-for-beginners>

Preet Minhas (2017). “An introduction to materials & standard shader in Unity”. Recuperado el 18 de febrero del 2020. Página web: <https://www.codementor.io/unity3d/tutorial/an-introduction-to-materials--standard-shader-in-unity>

Claudia Dopáoslas (2017). “Physically based shader development for unity 2017”. Recuperado el 18 de febrero del 2020. Página web: <https://books.google.com.mx/books?id=FgBCDwAAQBAJ&pg=PR19&lpg=PR19&dq=shading+unity&source=bl&ots=AbZYsdkOtD&sig=ACfU3U28JkJXoWpUdjbZ4GppKeqoJvrM6w&hl=es-419&sa=X&ved=2ahUKEwixtdu_wdvnAhUHITQIHcRDDP44FBDoATACegQICBAB#v=onepage&q=shading%20unity&f=false>

# OpenGl (NT). “OpenGL API Documentation Overview”. Recuperado el 19 de Febrero del 2020. Pagina web:<https://www.opengl.org/documentation/specs/>

# Davis (2020). “vertex Shaders”. Recuperado el 19 de febrero del 2020. Pagina web: <https://www.pcmag.com/encyclopedia/term/vertex-shader>

# Unity (2019) “Meshes, Materials, shaders and textures”. Recuperado el 19 de febrero del 2020. Pagina web: [https://docs.unity3d.com/Manual/Shaders.htmlhttps://docs.unity3d.com/Manual/Shaders.html](https://docs.unity3d.com/Manual/Shaders.html)

# Unity (2019). “Shading language used in Unity”. Recuperado el 19 de febrero del 2020. Pagina web: <https://docs.unity3d.com/Manual/SL-ShadingLanguage.html>

# Unity (2019). “Vertex and fragment shaders examples”. Recuperado el 19 de febrero del 2020. Pagina web: <https://docs.unity3d.com/Manual/SL-VertexFragmentShaderExamples.html>

# 

# Unity (2019). “Creating and using materials”. Recuperado el 19 de febrero del 2020. Página web: <https://docs.unity3d.com/Manual/Materials.html>