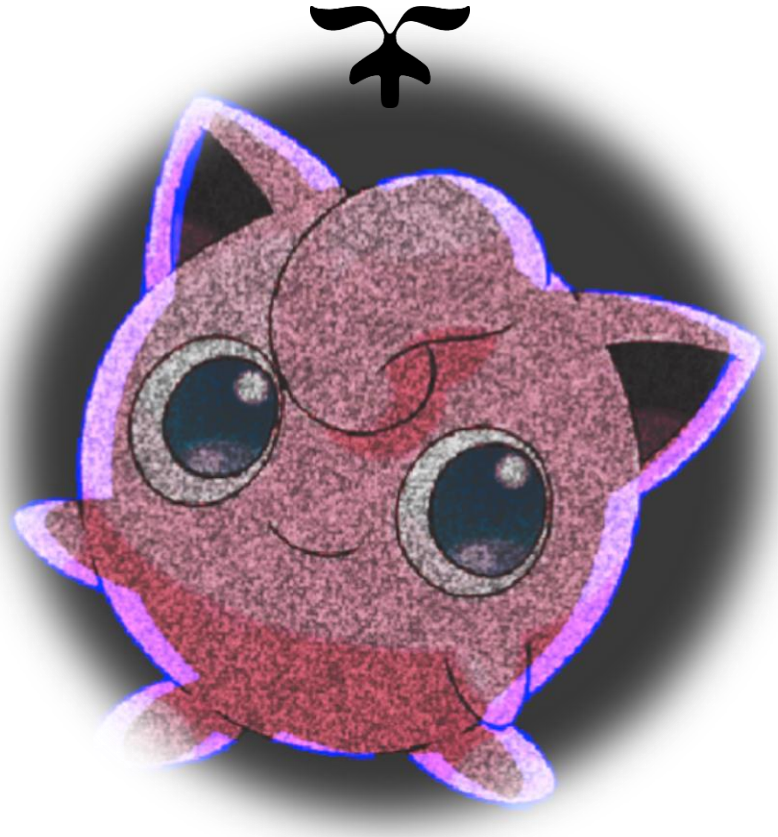
Proyecto Primer Parcial

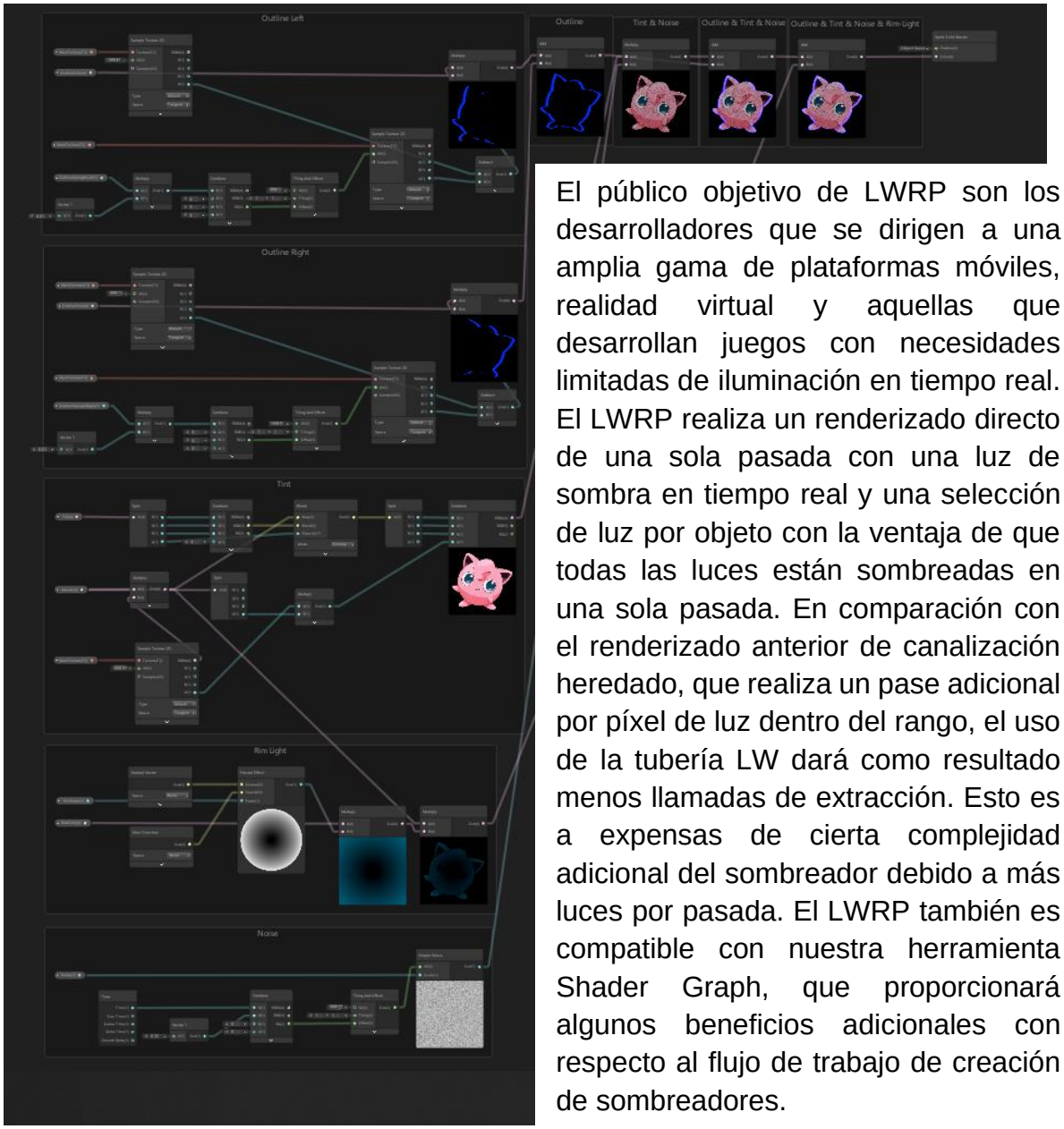
Tópicos de Física



Fátima Guadalupe Millanes Cázares
Ingeniería en Producción Multimedia
Cuarto Semestre, 189064
Maestro Jesús Caro Cota
Miércoles 19 de febrero de 2020

Asignación 5

El objetivo del LWRP es proporcionar un rendimiento optimizado en tiempo real en plataformas de rendimiento limitado al hacer algunas compensaciones con respecto a la iluminación y la sombra.



El público objetivo de LWRP son los desarrolladores que se dirigen a una amplia gama de plataformas móviles, realidad virtual y aquellas que desarrollan juegos con necesidades limitadas de iluminación en tiempo real. El LWRP realiza un renderizado directo de una sola pasada con una luz de sombra en tiempo real y una selección de luz por objeto con la ventaja de que todas las luces están sombreadas en una sola pasada. En comparación con el renderizado anterior de canalización heredado, que realiza un pase adicional por píxel de luz dentro del rango, el uso de la tubería LW dará como resultado menos llamadas de extracción. Esto es a expensas de cierta complejidad adicional del sombreador debido a más luces por pasada. El LWRP también es compatible con nuestra herramienta Shader Graph, que proporcionará algunos beneficios adicionales con respecto al flujo de trabajo de creación de sombreadores.

Lightweight Shaders

El LWRP tiene su propio proceso de renderizado y, por lo tanto, requiere sombreadores que están escritos pensando en él. Hemos desarrollado un nuevo conjunto de sombreadores estándar que se encuentran en el grupo de tuberías

ligeras en el menú desplegable de selección de sombreadores del material. Estos incluyen un sombreador PBR estándar, un sombreador estándar no PBR con un modelo de iluminación simplificado, un sombreador de terreno estándar y un sombreador estándar apagado. Vale la pena señalar que todos los sombreadores de stock no iluminados de Unity ya funcionan con LWRP. Esto incluye partículas heredadas, IU, skybox y sombreador de sprites.

Shaders

El rendering en Unity usa Materiales, Shaders y Texturas. Los tres tienen una relación cercana.

Materiales define cómo debe renderizarse una superficie, incluyendo referencias a las Texturas que utiliza, información de tiling, tintes de color y más. Las opciones disponibles para un Material dependen del Shader que el Material esté usando.

Shaders son pequeños scripts que contienen los cálculos matemáticos y algoritmos para calcular el Color de cada píxel procesado, en función de la entrada de iluminación y la configuración del Material.

Texturas son imágenes bitmap. Un Material puede contener referencias a las texturas, de modo que el Shader del Material pueda usar las texturas al calcular el color de la superficie de un GameObject. Además del color básico (Albedo) de la superficie de GameObject, las texturas pueden representar muchos otros aspectos de la superficie de un material, como su reflectividad o rugosidad.

Un material especifica un shader específico para usar, y el shader utilizado determina qué opciones están disponibles en el material. Un Shader especifica una o más variables de Textura que espera usar, y el Inspector de Materiales en Unity le permite asignar sus propios Elementos de Textura a estas variables de Textura.

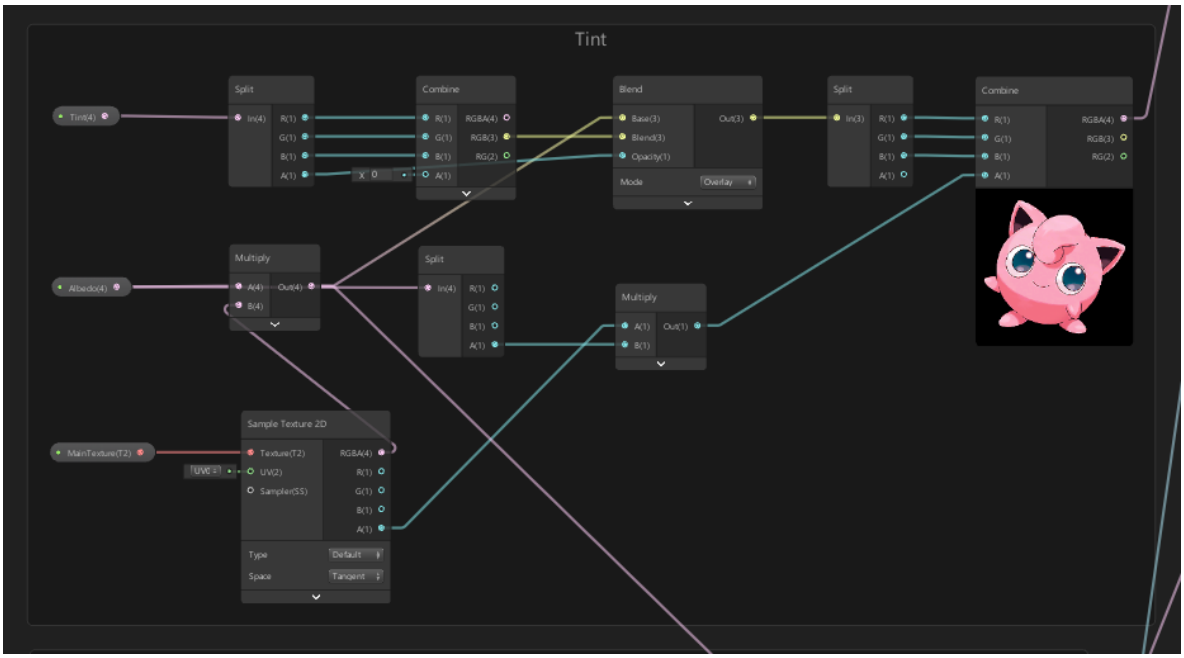
Para la mayoría de los rendering normales (como rendering de personajes, escenarios, entornos, GameObjects sólidos y transparentes, superficies duras y blandas), el Standard Shadersuele ser la mejor opción. Este es un shader altamente personalizable que es capaz de renderizar muchos tipos de superficies de una manera altamente realista.

Albedo Color

Algunas veces es útil especificar un solo color para el valor Albedo, pero es mucho más común asignar un mapa de textura para el parámetro Albedo. Esto debería representar los colores de la superficie del objeto. Es importante tener en cuenta

que la textura Albedo no debe contener ninguna iluminación, ya que la iluminación se agregará en función del contexto en el que se ve el objeto.

El valor alfa del color Albedo controla el nivel de transparencia del material. Esto solo tiene efecto si el Modo de renderizado para el material se establece en uno de los modos transparentes, y no opaco. Como se mencionó anteriormente, elegir el modo de transparencia correcto es importante porque determina si aún verá reflejos y reflejos especulares a valor total, o si también se desvanecerán de acuerdo con los valores de transparencia.



Textura de albedo: en el flujo de trabajo especular representa el color de la luz difusa que rebota en la superficie. No necesariamente tiene que ser muy detallado, mientras que la textura incluye detalles únicos significativos.

Especular: los no metales (aislantes) son relativamente oscuros y en escala de grises, mientras que los valores de los metales son brillantes y podrían estar coloreados.

La suavidad es un elemento clave en los materiales. Aporta variación, imperfecciones y detalles a las superficies y ayuda a representar su estado y edad.

La oclusión indica cuán expuestos están los diferentes puntos de la superficie a la luz del entorno circundante. La oclusión ambiental resalta los detalles y la profundidad de la superficie al silenciar el ambiente y la reflexión en áreas con poca luz indirecta.

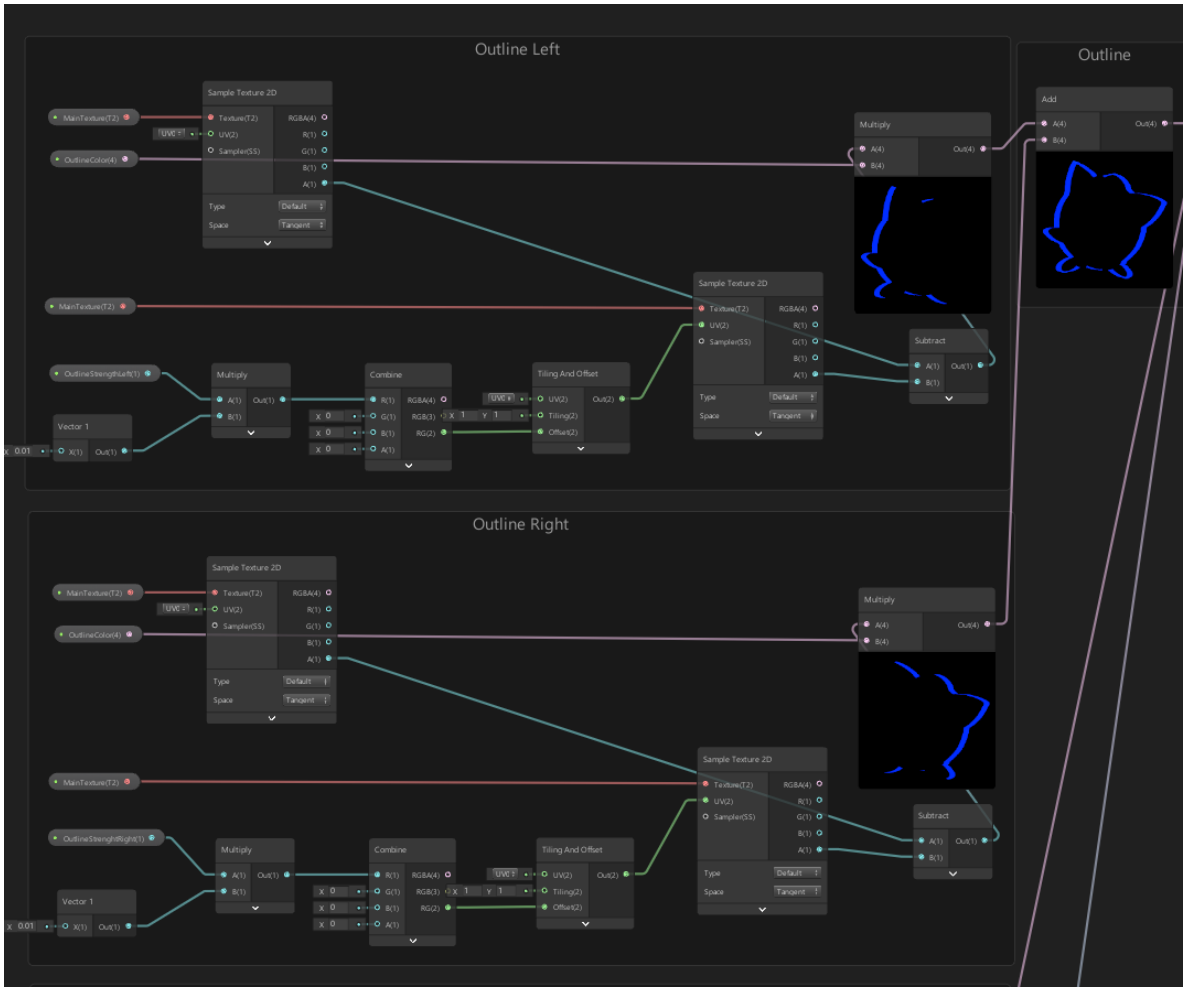
Node

Un nodo define una entrada, salida u operación en el gráfico de sombreador, dependiendo de sus puertos disponibles. Un nodo puede tener cualquier número de puertos de entrada y / o salida. Puede crear un gráfico de sombreador conectando estos puertos con bordes. Un nodo también puede tener cualquier número de controles, estos son controles en el nodo que no tienen puertos.

Puede contraer un nodo haciendo clic en el botón Contraer en la esquina superior derecha del nodo. Esto ocultará todos los puertos no conectados.

Algunos nodos tienen una vista previa en el nodo. Esta vista previa muestra el valor de salida principal en esa etapa en el gráfico. La vista previa se puede ocultar con el botón Contraer en la parte superior de la vista previa. Esto se muestra cuando el cursor del mouse se encuentra sobre el nodo. También puede contraer y expandir las vistas previas en todos los nodos a través del menú contextual en la ventana de gráfico de sombreador.

Outline



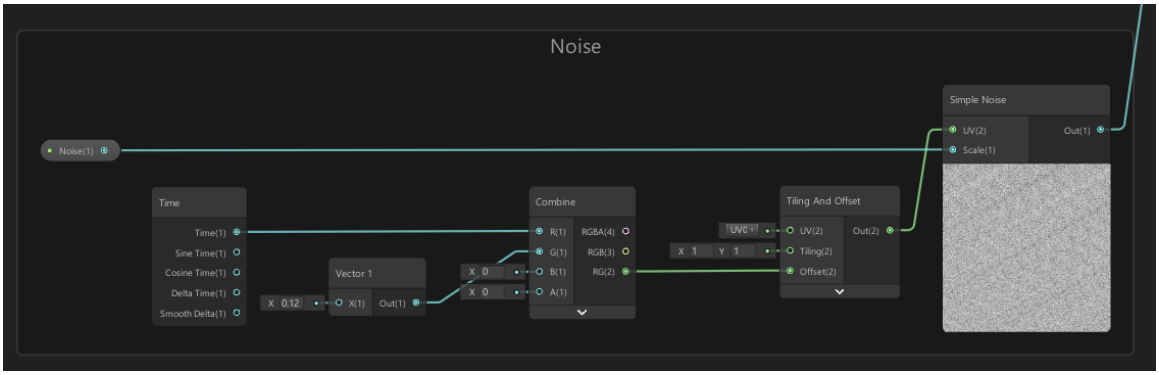
El componente Outline agrega un efecto simple de contorno a componentes gráficos tal como Text o Image. Debe estar en el mismo GameObject que el componente gráfico.

Noise

Genera un ruido simple, o valor, basado en la entrada UV. La escala del ruido generado se controla mediante la escala de entrada.

El efecto de imagen Ruido y grano simula el ruido y el grano de la película, que es un efecto típico que ocurre en una película o fotografía. Esta implementación especial de ruido incluso se puede utilizar para mejorar el contraste de la imagen, ya que utiliza un modo de mezcla especial. También permite escenarios de ruido típicos, como el ruido de luz de bajo nivel o el suavizado de halo brillante o bordes florecidos.

Agregar ruido a sus partículas es una forma simple y efectiva de crear patrones y efectos interesantes. Por ejemplo, imagine cómo se mueven las brasas de un incendio, o cómo se arremolina el humo a medida que se mueve. Se podría usar un ruido fuerte de alta frecuencia para simular las brasas de fuego, mientras que el ruido suave de baja frecuencia sería más adecuado para modelar un efecto de humo.



Para un control máximo sobre el ruido, puede habilitar la opción Ejes separados. Esto le permite controlar la fuerza y la reasignación en cada eje de forma independiente.

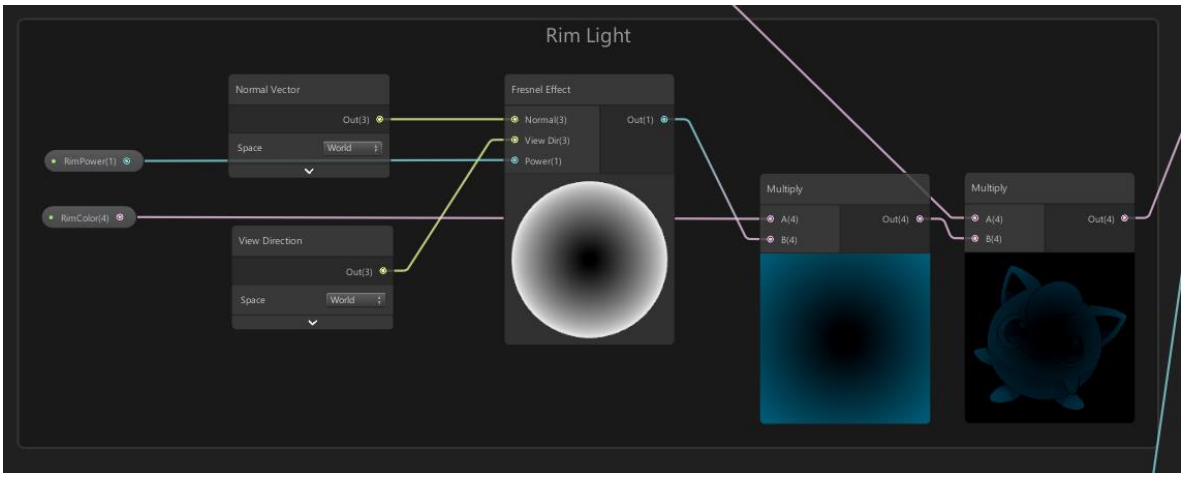
La implementación de DirectX 11 / OpenGL 3 es totalmente independiente de cualquier lectura de textura y, por lo tanto, es una buena opción para el hardware de gráficos moderno.

La versión estándar utiliza una textura de ruido que debería tener una luminosidad promedio de 0.5 para evitar cambios de brillo no deseados de la imagen resultante. La textura predeterminada utilizada es un ejemplo de esto.

Rim Light (Fresnel Effect)

Una señal visual importante de objetos en el mundo real tiene que ver con cómo se vuelven más reflexivos en ángulos de pastoreo (ilustrados abajo). Esto se conoce como efecto Fresnel.

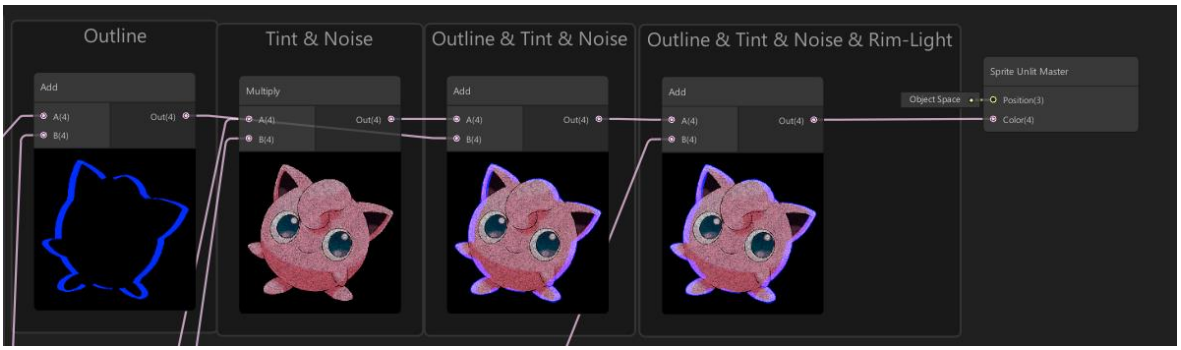
El efecto fresnel visible en los ángulos de pastoreo en relación con el espectador es cada vez más evidente a medida que la superficie de un material se vuelve más suave.



Hay dos cosas a tener en cuenta en este ejemplo: En primer lugar, estas reflexiones sólo aparecen alrededor de los bordes de la esfera (es decir, cuando su superficie está en un ángulo de pastoreo), y también que se hacen más visibles y más nítidas a medida que sube la suavidad del material.

En el Standard Shader no hay control directo sobre el efecto Fresnel. En su lugar, se controla indirectamente a través de la suavidad del material. Las superficies lisas presentarán un Fresnel más fuerte, las superficies totalmente ásperas no tendrán Fresnel.

Conclusión



Bibliografía

Cooper, T. (2018). The Lightweight Render Pipeline: Optimizing Real Time Performance. Recuperado de: https://blogs.unity3d.com/es/2018/02/21/the-lightweight-render-pipeline-optimizing-real-time-performance/

Doppioslash, C. (2018). Your First Unity Lighting Shader. In Physically Based Shader Development for Unity 2017 (pp. 51-64). Apress, Berkeley, CA.

Doppioslash, C. (2018). How Shader Development Works. In Physically Based Shader Development for Unity 2017 (pp. 3-16). Apress, Berkeley, CA.

Hudon, M., Grogan, M., Pagés, R., & Smolić, A. (2018) 2D Shading for Cel Animation. DOI: 10.1145/3229147.3229148. Recuperado de: https://www.researchgate.net/publication/325788674_2D_Shading_for_Cel_Animation

Hudon, M., Grogan, M., Pagés, R., & Smolić, A. (2018) 2DToonShade: A stroke based toon shading system. Recuperado de: https://www.sciencedirect.com/science/article/pii/S2590148619300032

Körner, E. (2015). Working with Physically-Based Shading: a Practical Approach. Recuperado de: https://blogs.unity3d.com/es/2015/02/18/working-with-physically-based-shading-a-practical-approach/

Linowes, J. (2015). Unity virtual reality projects. Packt Publishing Ltd.

Oughstun, K. E., & Palombini, C. L. (2018). Fresnel reflection and transmission coefficients for temporally dispersive attenuative media. Radio Science, 53(11), 1382-1397.

Thorn, A., Doran, J. P., Zucconi, A., & Palacios, J. (2019). Complete Unity 2018 Game Development: Explore techniques to build 2D/3D applications using real-world examples. Packt Publishing Ltd.

Unity Technologies. (2019). Albedo color and Transparency. Recuperado de: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterAlbedoColor.html

Unity Technologies. (2019). The fresnel effect. Recuperado de: https://docs.unity3d.com/Manual/StandardShaderFresnel.html

Unity Technologies. (2018). Standard Shader. Recuperado de: https://docs.unity3d.com/es/2018.4/Manual/shader-StandardShader.html

Unity Technologies. (2018). Materiales, Shaders y Texturas. Recuperado de: https://docs.unity3d.com/es/2018.4/Manual/Shaders.html

Unity Technologies. (2015). Nodes. Recuperado de: https://docs.unity3d.com/Packages/com.unity.shadergraph@5.10/manual/Node.html

Unity Technologies. (2020). Simple Noise Node. Recuperado de: https://docs.unity3d.com/Packages/com.unity.shadergraph@7.1/manual/Simple-Noise-Node.html

Unity Technologies. (2020). El efecto Fresnel. Recuperado de: https://docs.unity3d.com/es/current/Manual/StandardShaderFresnel.html

Unity Technologies. (2017). Noise And Grain. Recuperado de: https://docs.unity3d.com/550/Documentation/Manual/script-NoiseAndGrain.html

Unity Technologies. (2019). Noise module. Recuperado de: https://docs.unity3d.com/Manual/PartSysNoiseModule.html

Unity Technologies. (2019). Mathf.Clamp. Recuperado de: https://docs.unity3d.com/ScriptReference/Mathf.Clamp.html

Unity Technologies. (2018). Outline. Recuperado de: https://docs.unity3d.com/es/2018.4/Manual/script-Outline.html