

**Proyecto Shading Primer Parcial**

**Alumno:** Jesús Alberto Hoyos Félix

**Carrera:** Ing. Producción Multimedia

**Matricula:** 189180

**Materia:** Tópicos de Física

**Fecha:** 19/02/20

**Lugar:** Ciudad Obregón, Sonora, México

**Repositorio:** https://github.com/Alberth-h/ProyectoShadingPrimerParcialTopicosDeFisica

**Objetivos**

Este proyecto tiene como objetivo el desarrollo de 2 shadergraphs, en donde, uno consiste en mezclar un outline con un entintado para un Sprite, y el otro un efecto de glitch u holograma mas un efecto de fresnel, y colocarlos en un sprite animado. Estos shadergraphs se realizaron en Unity, en su sistema nodal de shadergraph.

**Desarrollo**

Para el desarrollo de los shadergraphs, cree un proyecto nuevo en Unity, en cual importe una hoja de sprites, mediante el editor de sprites, corte cada una de ellas para poder realizar una animación de movimiento del personaje.

Se crearon 2 shadergraphs de tipo 2D Render Sprite Lite Graph, uno con el nombre de “OutlineInkedCombineShader”, el cual corresponde a la mezcla entre el outline y el entintado del Sprite, y el otro con el nombre de “GlitchFresnelShader”, el cual corresponde a la mezcla entre el efecto glitch y fresnel” y 2 materiales a los que se les asigno los 2 shadergraphs respectivamente con los nombres de “OutlineInkedCombineMaterial” y “GlitchFresnelMaterial”.

Para el desarrollo del ShaderGraph “OutlineInkedCombineShader” se crearon las propiedades de una textura principal, el color del outline, el tamaño del outline y el color del entintado. Mediante el sistema nodal de Unity, se utilizaron los nodos de tipo Sample Texture 2D, Tilling and Offset, Multiply, Combine, Negate, Add, Clamp y Subtract, para realizar un corte en el Sprite, adquiriendo su área más un poco de espacio extra para que este obtuviera un el efecto de outline, y con los nodos de tipo Sample Texture 2D y Multiply, se realizo un entintado para el Sprite y mediante un add se juntaron ambos efectos en el ShaderGraph. Finalmente, en el Material “OutlineInkedCombineMaterial” se le asigno el shadergraph de “OutlineInkedCombineShader” y el material se le asigno al Sprite para que obtuviera el efecto.

Para el desarrollo de ShaderGraph “GlitchFresnelShader” se crearon las propiedades de una textura principal, la velocidad del scroll, la velocidad de flick, y un color para el efecto de glitch, y para el efecto de fresnel, un color y otra propiedad para el tamaño de este. Mediante el sistema nodal de Unity, se utilizaron los nodos de tipo Sample Texture 2D, Tilling and Offset, Multiply, Combine, Voronoi, Time, Noise Sine Wave, para realizar el efecto de glitch, en donde con los Times crean un efecto de animación en la textura y con el voronoi y el Noise se logro crear un efecto de líneas y de desvanecido, todo esto conformo el efecto de glitch, y con un Nodo de Fresnel y un Multiply se creo el efecto de Fresnel y para recortar el área sobrante de ambos efectos se realizó la misma técnica para realizar el Outline. Finalmente, en el Material “GlitchFresnelMaterial” se le asigno el shadergraph de “GlitchFresnelShader” y el material se le asigno al Sprite para que obtuviera el efecto.

**Conclusión/Resultados**

El shadergraph “OutlineInkedCombineShader” funciono correctamente, ya que realiza un efecto de outline y también un entintado en el sprite asignado para este.

El shadergraph “GlitchFresnelShader” funciono correctamente pero no de la forma esperada, el efecto glitch se logro correctamente, sin embargo, el efecto de fresnel se esperaba que fuera más notorio y más suave.

**Referencias**

Morant Capellino, J. L. Desarrollo de un videojuego en Unity con generación de escenario procedural.

He, Y., Foley, T., & Fatahalian, K. (2016). A system for rapid exploration of shader optimization choices. ACM Transactions on Graphics (TOG), 35(4), 1-12.

Lee, M., Green, B., Xie, F., & Tabellion, E. (2017). Vectorized production path tracing. In *Proceedings of High Performance Graphics* (pp. 1-11).

刘智珺. (2015). Design and application of transaction scheduling algorithm. International Conference on Mechanics & Mechatronics.

Liu, Z. J., & Pan, X. F. (2016). Design and application of transaction scheduling algorithm. In Mechanics and Mechatronics (ICMM2015) Proceedings of the 2015 International Conference on Mechanics and Mechatronics (ICMM2015) (pp. 1102-1108).

Panagiotidis, A., Reina, G., Burch, M., Pfannkuch, T., & Ertl, T. (2015, August). Consistently GPU-accelerated graph visualization. In *Proceedings of the 8th International Symposium on Visual Information Communication and Interaction* (pp. 35-41).

Keramidas, G., Kokkala, C., & Stamoulis, I. (2015). Clumsy value cache: An approximate memoization technique for mobile GPU fragment shaders. In Workshop on Approximate Computing (WAPCO’15).

Morant Capellino, J. L. Desarrollo de un videojuego en Unity con generación de escenario procedural.

Delva, M., Hamaide, J., & Ladlani, R. (2018). Semantic-based shader generation using shader shaker. In *GPU Pro 360 Guide to 3D Engine Design* (pp. 289-304). AK Peters/CRC Press.

He, Y., Foley, T., Tatarchuk, N., & Fatahalian, K. (2015). A system for rapid, automatic shader level-of-detail. ACM Transactions on Graphics (TOG), 34(6), 1-12.

Bailey, M., & Cunningham, S. (2016). Graphics shaders: theory and practice. AK Peters/CRC Press.

Helsing, J. K., & Elster, A. C. (2015, September). Noise Modeler: An Interactive Editor and Library for Procedural Terrains via Continuous Generation and Compilation of GPU Shaders. In International Conference on Entertainment Computing (pp. 469-474). Springer, Cham.

Davila, R. R. (2015). Bounding the forcing number of a graph (Doctoral dissertation).

He, Y., Foley, T., Hofstee, T., Long, H., & Fatahalian, K. (2017). Shader components: modular and high performance shader development. ACM Transactions on Graphics (TOG), 36(4), 1-11.

Polson, B. (2015). Pipeline design patterns. In ACM SIGGRAPH 2015 Courses (pp. 1-59).

Arens, S. (2015). A dataflow-based shader framework for visualizing dissections of the heart using individual patient data (Doctoral dissertation, University of Paderborn).

Lee, M., Green, B., Xie, F., & Tabellion, E. (2017). Vectorized production path tracing. In Proceedings of High Performance Graphics (pp. 1-11).

Yang, Y., & Barnes, C. (2018, May). Approximate Program Smoothing Using Mean‐Variance Statistics, with Application to Procedural Shader Bandlimiting. In Computer Graphics Forum (Vol. 37, No. 2, pp. 443-454).

Crawford, L., & O'Boyle, M. (2018, April). A cross-platform evaluation of graphics shader compiler optimization. In 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) (pp. 219-228). IEEE.

Huang, J., Li, S., & Wang, H. (2018). Relation between the skew-rank of an oriented graph and the independence number of its underlying graph. Journal of Combinatorial Optimization, 36(1), 65-80.