

Universidad La Salle Noroeste

Ingeniería en Producción Multimedia

Tópicos de Física

Rubén Borbolla Castelo

189216

4to Semestre

Jesús Francisco Caro Cota

Cd. Obregón, Sonora. 19 de febrero de 2020

El objetivo principal de mi trabajo fue lograr el clásico efecto “glitch” visto numerosas veces en diversos trabajos multimedia, como lo son: videojuegos, videos artísticos, vídeos musicales, anime, caricaturas, etc. Haciendo uso de shaders personalizados hechos a partir de nodos en la plataforma de desarrollo gratuita Unity.

En aplicaciones interactivas de síntesis 2D, Las características para las regiones de relleno son generalmente elementales (coloración uniforme, relleno con un patrón regular ...). El diseño interactivo de los sombreados de color 2D plantea dos problemas principales: la dificultad de especificar un sombreado con precisión y la rapidez de los algoritmos de relleno relacionados.

Colorshading una región del plano es la acción de llenar su interior con colores que varían continuamente. Se da una condición de suavidad. Se presentan varias formas de shading.

Se proporcionan varios métodos y sistemas para sombrear una imagen de ultrasonido 2D, generada a partir de datos de ultrasonido, usando un gradiente determinado a partir de valores escalares de los datos de imagen de ultrasonido. Como un ejemplo, un método incluye la correlación de los valores de imagen de un conjunto de datos adquirido con un sistema de imágenes de ultrasonido con los valores de altura; determinar un gradiente de los valores de altura; aplicar sombreado a una imagen 2D generada a partir del conjunto de datos utilizando el gradiente determinado; y mostrando la imagen sombreada en 2D.

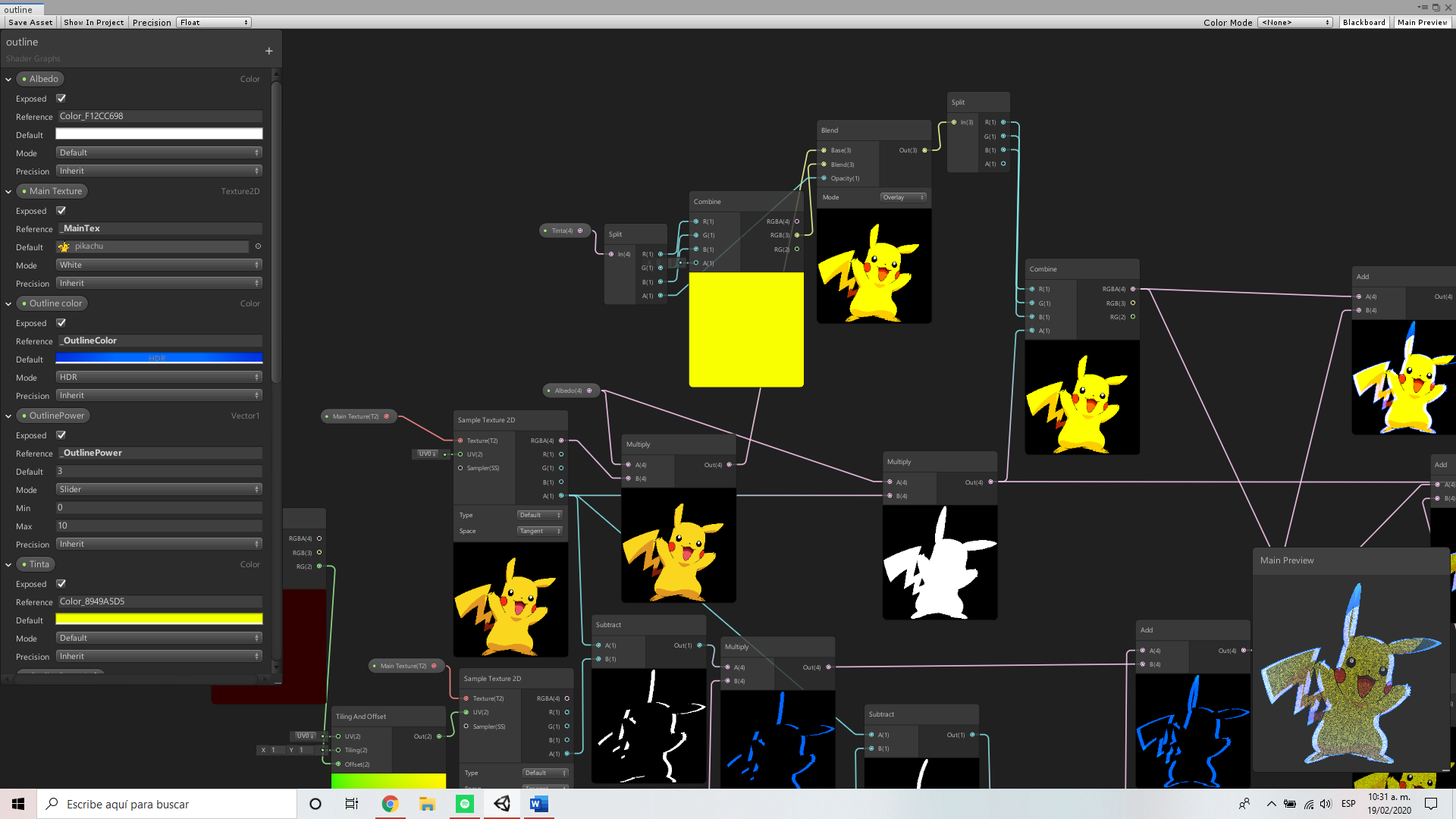
Lo primero que hice para poder lograr este efecto, fue elegir el modelo al cual le aplicaría el material con el shader que simula el efecto. Descargué directamente de Google imágenes aplicando los filtros de imágenes transparentes e imágenes con “fair use” un pikachu hecho en “pixel art”, ósea, dibujado usando como base pocos píxeles, pero debido a problemas en el desarrollo del shader y poca compatibilidad que este mismo tuvo, no me quedó otra opción por cambiar de Sprite, por lo tanto, me dirigí de nuevo a Google imágenes y descargué un Sprite 2D de pikachu diferente, ya no en formato pixel.

Ya que tenía listo el Sprite que usaría, lo incorporé dentro del proyecto, le cambié el tipo de textura de Default a Sprite (2D y UI) y procedí a aplicarle el cambio. Ahora que la imagen ya era detectada por Unity como Sprite, procedí a añadirlo en escena, para esto cree un objeto 2D - Sprite y lo nombre como “Pikachu”, dentro del inspector del mismo objeto, en la sección de renderizado de Sprite, cargué el Sprite de pikachu que había incorporado con anterioridad al proyecto y listo, tenía completamente listo al modelo que le aplicaría el shader en escena.

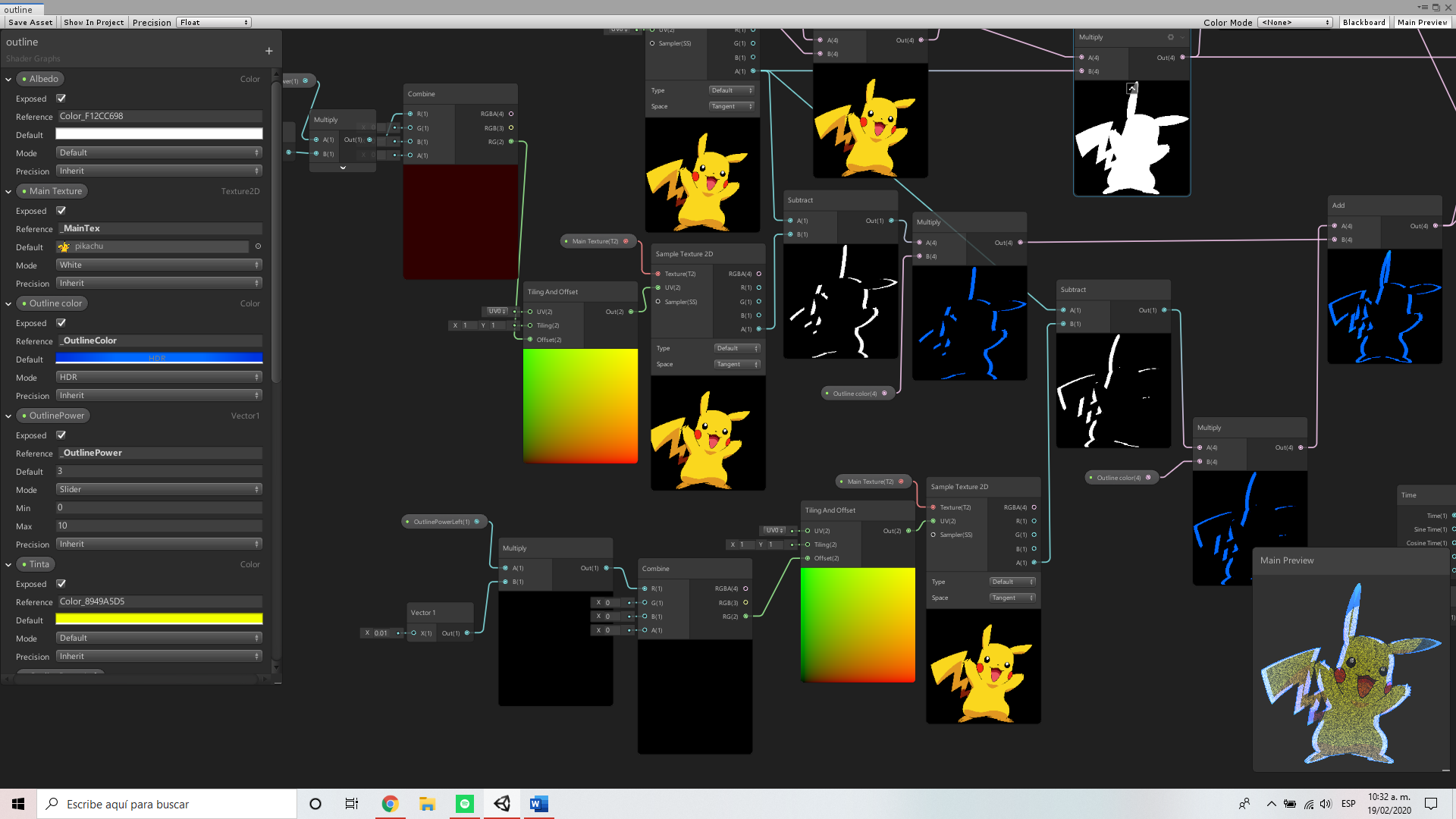
Procedí a crear el shader que le daría el efecto glitch al modelo, para eso, primero cree una carpeta llamada Shaders para tener una mejor organización, en esa misma carpeta, cree un shader con renderización en 2D – Sprite unlit graph, lo nombre como “Basic2D”, abrí el archivo dando doble clic para que se abriera el editor.

Lo primero que hice fue crear un entintado para el modelo, para hacerlo de la forma correcta y que este no esté completamente pintado y solo sea como una capa transparente que solo pinte al modelo y no al fondo, tuve que realizar los siguientes pasos:

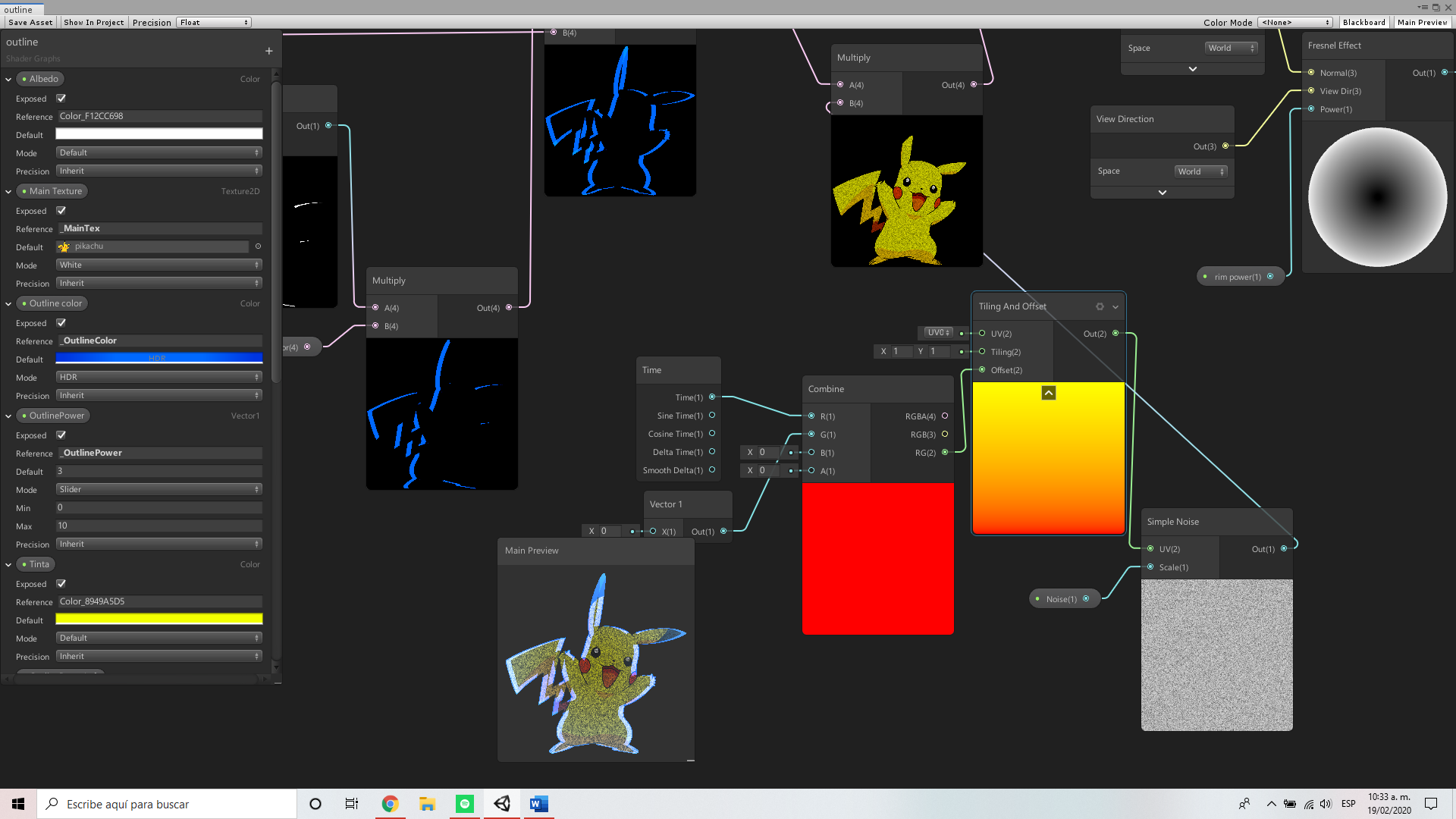
* En Shader Graphs, cree un color y un texture2D, al color lo llamé albedo y al texture2D lo llamé Main texture, además de cargar el Sprite de pikachu.
* Mandé el Main Texture al editor de nodos y lo conecté con un Sample Texture 2D a Texture(T2), y la salida RGBA de este mismo lo multipliqué en un Multiply con el Albedo creado con anterioridad.
* Procedí a crear otro color llamado Tinta para poder cambiarle el color a Pikachu, a este le subí todo el alfa para que se pudiera apreciar, lo arrastré al editor y mandé el nodo a un Split.
* Del Split mandé los nodos a un Combine en sus respectivas conexiones con los nodos, exceptuando a el alfa que se manda directamente a un Blend en la conexión de Opacity.
* La salida RGB de Combine la conecté a La entrada Blend(3) de Blend, y la salida del multiply del paso 2 la mandé a Base(3) del Blend, después el Out de Blend a un nuevo Split.
* Finalmente multipliqué el alfa de la textura y el albedo, procedí a Combinar el RGB y la salida del multiplicador (A) y ya tenía un pikachu perfectamente entintado a mi gusto.



Después de esto le agregué el efecto “Outline”, para esto, Multipliqué dos vectores para poder controlar su grosor, después los combiné y los mandé a un Tiling and Offset, la salida de este mismo la conecte con un Sample texture 2D, la cuál tenía cargada a pikachu, mandé el alfa de esta misma a un Substract, la cuál básicamente restaba el alfa del pikachu original y el alfa el cuál yo estaba alterando, a este mismo lo multipliqué con un color para darle el color deseado al Outline, el único problema es que solo se ve de un lado, esto se debe a que solo estamos moviendo el alfa, para poder lograr el efecto “espejo” al outline, lo que hice fue duplicar el proceso del outline pero solo le cambie los valores del vector a un valor negativo y finalmente los sumé para conectarlo con el pikachu entintado.



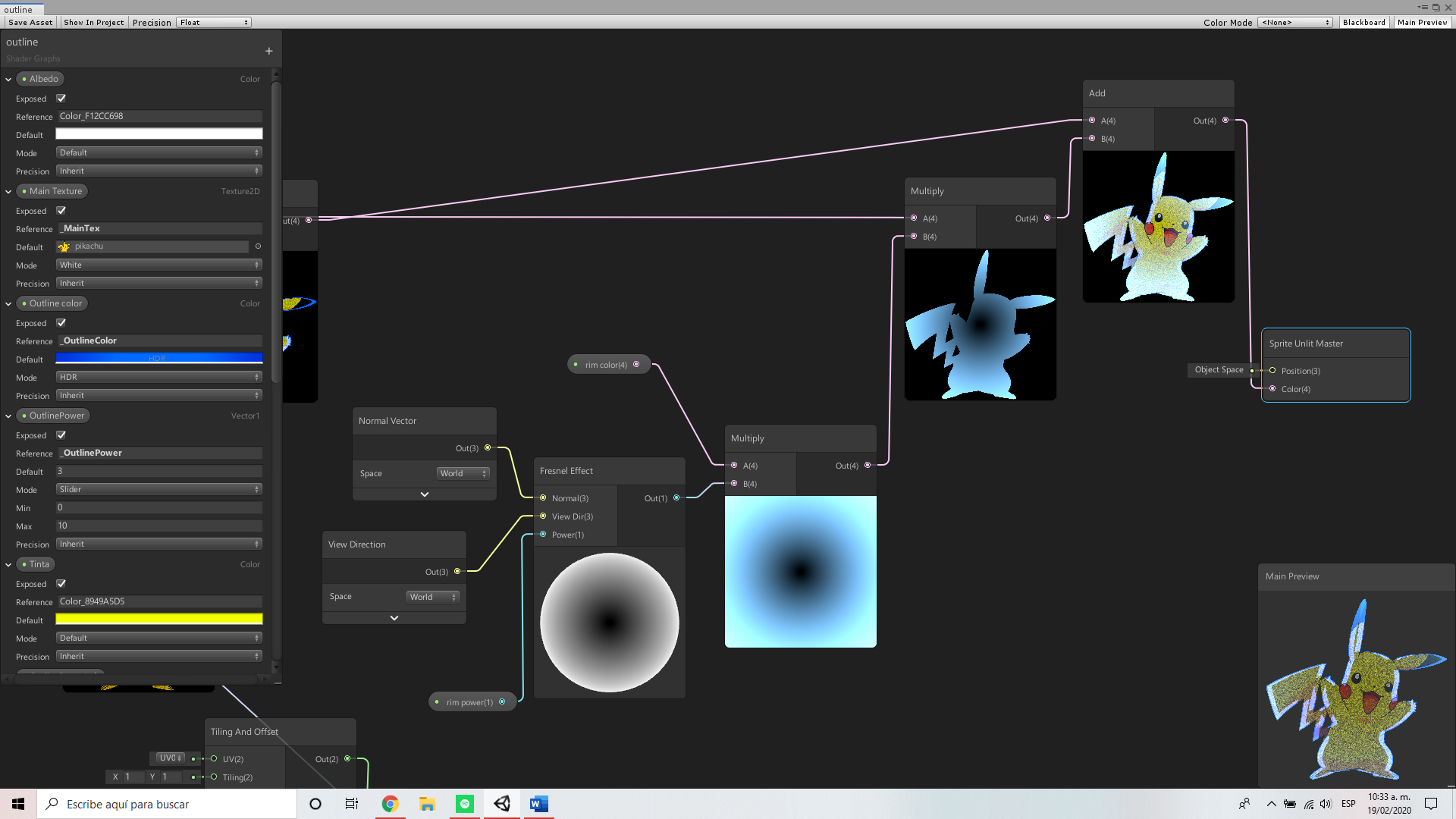
Ahora que tenía un pikachu completamente entintado y además con un Outline, procedí a añadirle un ruído, el cuál le daría este efecto de glitch. Para esto combiné un Time y un vector en R y G respectivamente para después mandar el RG a un Offset y la salida de este a un Simple Noise en UV. Cree un Vector que controlaría la escala del noise y lo conecté en Scale, la salida del simple noise la multipliqué con la textura de pikachu y finalmente la sumé con el pikachu ya entintado y con Outline.



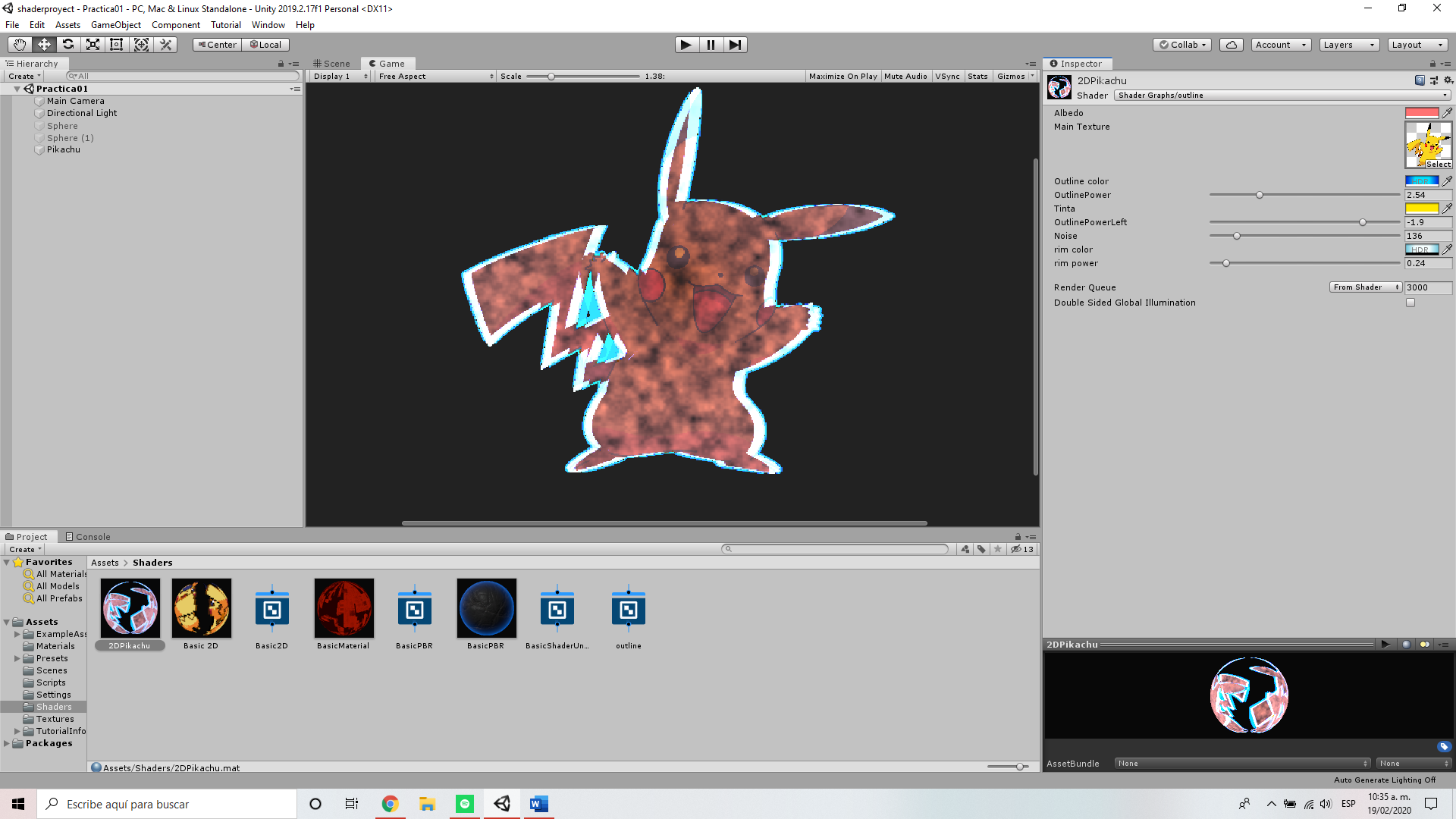
Para el toqué final, decidí añadirle un Fresnel effect, para lograr esto conecté un Normal vector, un view direction y un rim power a un Fresnel Effect, y la salida de este mismo lo multipliqué con un color, para darle a este mismo una variación de color para que no se viera completamente blanco. Finalmente el resultado de esta multiplicación la volví a multiplicar, pero ahora con el alfa de la textura inicial de pikachu para que se ajustara solo a la figura del modelo.

Para terminar, agregué el resultado de la última multiplicación a el pikachu con el ruido, el Outline y el entintado.

La suma de todo esto la conecté con el Color del Sprite Unlit Master, y listo, ya tenía el shader completamente terminado y listo para aplicarse.



Finalmente, añadí el shader a un nuevo Material y ese material se lo puse al Modelo de Pikachu.



Referencias:

Šmíd, A. (2017). Comparison of unity and unreal engine. *Czech Technical University in Prague*.

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

Doppioslash, C. (2018). Physically Based Shader Development for Unity 2017.

Doppioslash, C. (2017). *Physically Based Shader Development for Unity 2017: Develop Custom Lighting Systems*. Apress.

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

Wright, B., Anderson, M., McBride, A., Falt, H., Peixe, D., & DeRosa, T. (2019). 2D animation in the VR clouds: the making of disney's" a kite's tale". In *ACM SIGGRAPH 2019 Talks* (pp. 1-2).

Raappana, J. (2018). Great Particles and How to Make Them: Development of guidelines for the Unity Particle System.

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

Gil, A., Khurshid, A., Postal, J., & Figueira, T. (2019, October). Visual assessment of equirectangular images for virtual reality applications In Unity. In *Anais Estendidos da XXXII Conference on Graphics, Patterns and Images* (pp. 237-242). SBC.

Zucconi, A., & Lammers, K. (2016). *Unity 5. x Shaders and Effects Cookbook*. Packt Publishing Ltd.

Dean, J. (2016). *Mastering Unity Shaders and Effects*. Packt Publishing Ltd.

Halladay, K. (2019). *Practical Shader Development: Vertex and Fragment Shaders for Game Developers*. Apress.

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.

Busquets Duran, R. (2019). *Procedural textures generation: adaptation into a Unity tool* (Bachelor's thesis, Universitat Politècnica de Catalunya).

Gallego López, V. (2017). Creation of optimized hyperrealism for unity.

何海菠, & 吕超. (2017). 计算机组装与维护三维仿真系统——Unity 中 Shader 编程. *电脑迷*, (1), 79.

Blackman, S., & Tuliper, A. (2016). *Learn Unity for Windows 10 Game Development*. Apress.

Doppioslash, C. (2018). Your First Unity Shader. In *Physically Based Shader Development for Unity 2017* (pp. 17-32). Apress, Berkeley, CA.

Gil, A. M., & Figueira, T. S. (2019, July). Equirectangular image quality assessment tool integrated into the unity editor. In *International Conference on Human-Computer Interaction* (pp. 374-381). Springer, Cham.

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

Glassett, J. L., & McDonald, J. J. (2019). Spectrally arbitrary patterns over rings with unity. *Linear Algebra and its Applications*, *576*, 228-245.