MISTERY CAVE

Rendering Avanzado

Alejandro López Vizuete
Claudia Ochagavías Recio

Universidad Rey Juan Carlos

ETSII
Escuela Técnica Superior de Ingeniería Informática

ÍNDICE

I. Introducción

El siguiente trabajo corresponde a la práctica 2 de la asignatura de Rendering Avanzado: Crear una simulación de un recorrido a través de una gruta subterránea. Con ello se pretende reforzar los conceptos de OpenGL que se han introducido en la asignatura de gráficos 3D y explorar las técnicas que se han estudiado en Rendering Avanzado para ilumunación global.

En esta pequeña memoria explicaremos cada una de las funcionalidades incluidas en el código, detallaremos cómo se han implementado y mostraremos imágenes de los resultados obtenidos.

El código también ha sido comentado para que resulte más fácil de comprender y seguir.

II. Bloque 1

A. Inicio a partir de la práctica 5 de la asignatura G3D

Para comenzar la práctica partimos inicialmente del código de la práctica 5 de la asignatura G3D, en la que habíamos realizado la implementación de un motor de render.

Inicialmente el código base contenía una escena conformada por una cámara, un cubo, cargado 4 veces, 2 shaders distintos (cada uno de ellos aplicados a dos de los cubos), y 4 luces, una para cada cubo. Las luces podían ser de tipo puntual, direccional o focal.

Tuvimos dificultades al empezar a utilizar el código base, ya que habia problemas en la escena, la cual se visualizaba de manera distinta en ambos ordenadores. Tras realizar varias pruebas con otros compañeros, comprobamos que el error no era del código, sino de uno de los dispositivos. Tambien tuvimos algunos pequeños problemas en el motor inicial, como la introducción de varios objetos diferentes, ya que solo era capaz de pintar un tipo de objeto.

B. Insertar varios objetos en la escena

Una vez comprobado que el código inicial funcionaba correctamente, y corregidos algunos errores menores, comenzamos la práctica cargando varios objetos en la escena. Hasta ahora solo se podía cargar un cubo, así que creamos un archivo .h nuevo para cargar un plano. Modificamos el código para que se pudiera cargar correctamente (“loadMallaSuelo”).

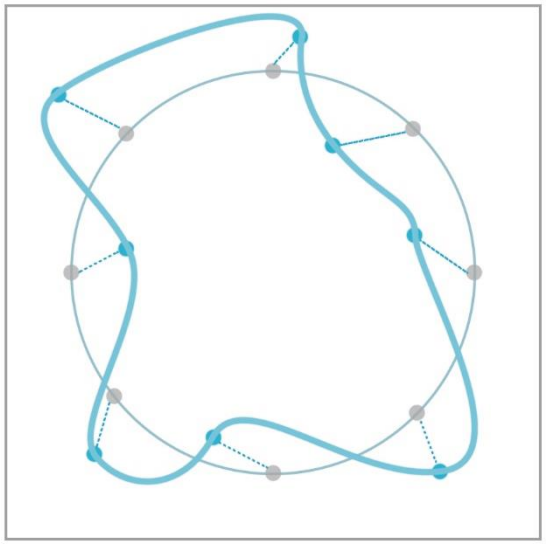
C. Recorrido de la cámara

Para crear la trayectoria que sigue el viajero a través de la gruta y las paredes a ambos lados inicialmente es necesario crear un recorrido para la cámara mediante una secuencica de splines. Realizamos el cálculo de las mismas con curvas de Catmull-Rom, de la manera que se explica a continuación:

Creamos un amplio círculo definido por 8 puntos principales, que conformarán el recorrido base de la cámara por la gruta. Todos los puntos, contenidos en el plano XZ, son trasladados una distancia random (dentro de unos límites), tanto en el eje X como en el Z. Así, cada vez que se ejecute el programa se creará una cueva con recorrido distinto.

Estos puntos son los que se utilizan para realizar las splines por Catmull-Rom. La cámara pasará por todos los puntos reiteradamente desde que se inicie el programa y hasta que el usuario deja de ejecutarlo.

Para realizar las splines nos hemos basado principalmente en la referencia [1].



Ejemplo de un recorrido creado por la cámara sobre un plano.

VISTA DE LA CÁMARA

Utilizamos “glm::lookat” para modificar la matriz view de la cámara, dependiendo de la posición de la misma en la curva. Así, la cámara siempre estará mirando en la dirección de la tangente a la curva en cada una de las posiciones.

D. Creación de la cueva

A continuación, la creación de la cueva se ha hecho a partir de dos planos que son deformados en el eje Y positiva y negativamente.

Estos planos se han creado mediante funciones [2] en CPU, debido a que eran planos demasiados grandes para meterlos en un fichero .h.

Para ello, se ha creado un plano de tamaño 350 x 350, con 100 vértices de ancho y otros 100 vértices de largo. Se han definido las posiciones de los vértices, y las coordenadas de normales, texturas y color, y todo ello se ha implementado en la función de inicialización del objeto, para poder visualizarlo por pantalla.

Para realizar la deformación de los planos se ha utilizado el coseno. En el main, comparamos cada punto de los vértices con cada punto de la spline, y nos quedamos con las distancias mínimas. A esa distancia le aplicamos el coseno, para que el plano derive en una forma curva al deformarse. Para el techo aplicaremos +cos, y para el suelo -cos. El paso de tiempo establecido es de 0.01, por lo que, entre cada par de puntos tendremos 100 puntos intermedios. Para nuestros 8 puntos principales: 800 puntos.

E. Creación del agua

Para crear el agua, se ha introducido un plano simple, que se verá a lo largo de todo el recorrido de la cueva. A ese plano se le ha incorporado Perlin, que explicaremos en detalle en el siguiente punto. La incorporación de ruido de Perlin permite definir una gradación de color entre dos tonos diferentes (en nuestro caso utilizamos dos tonos azules) en función del rango dinámico que originalmente se ha obtenido con la suma de las octavas. Con este ruido también se ha generado un displacement mapping para crear una especie de olas en el agua.

Vista del agua con inclusión de Perlin,

tanto para el gradiente de color como para el displacement mapping.

Podremos observar este efecto de displacement mapping durante la ejecución de la práctica.

F. Incorporación de mapas de Perlin

Generamos el ruido de Perlin en el shader de vértices. Para ello, hemos creado una función, basada en las lecturas de referencia [3] y [4], que por cada vector "inPos" del shader te genera una variable float. Este valor corresponde al rudio de Perlin en esa posición. Por lo tanto, mezclando este ruido con el color azul correspondiente del agua, nos salen las diferentes tonalidades de azul. Para desplazar los vértices, lo único que hay que hacer es moverlos en la dirección de la normal un tamaño igual al ruido de Perlin correspondiente.

G. Incorporación de mapas celulares

Con los mapas celulares hemos tenido bastantes problemas, hemos probado diferentes técnicas para realizarlos y ninguna nos ha llegado a funcionar bien. También hemos probado a hacer el displacement mapping con Perlin, al igual que en el agua, pero los resultados no eran satisfactorios. Por lo que hemos optado por dejar esa parte del código comentado.

H. Renderizar modo puntos, alámbrico, polígonos rellenos

Para poder renderizar en los distintos modos, se han utilizado las diferentes teclas '1', '2' y '3' dentro de la función de eventos de teclado. Para cada una de ellas se ha ejecutado:

```
1: glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
2: glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
3: glPolygonMode(GL_FRONT_AND_BACK, GL_POINT);
```

I. Movimiento de la cámara

Hemos incluido varios movimientos de la cámara, que el usuario puede ejecutar a través del teclado:

PARAR O AVANZAR

Inicialmente la cámara se mueve continuamente a través de la cueva a una velocidad determinada, que puede ser más lenta o más rápida en función de la curva de Catmull-Rom creada.

Podemos parar o avanzar pulsando la tecla 'P'.

MÁS RÁPIDO O MÁS LENTO

Si queremos avanzar más rápido o más lento, utilizaremos las teclas 'O' y 'L' respectivamente. Estas solo funcionarán cuando la cámara se esté moviendo. Se podrá llegar hasta una velocidad mínima y una máxima.

ROTAR LA CÁMARA

Solo se podrá hacer cuando la cámara esté parada (se haya pulsado previamente ‘P’). Mediante las teclas ‘W’, ‘A’, ‘S’, ‘D’, de podrá rotar la cámara en diferentes trayectorias. Una vez que reanudemos el movimiento de la cámara (‘P’), esta volverá a mirar en la dirección de la tangente a la curva.

III. Bloque 2

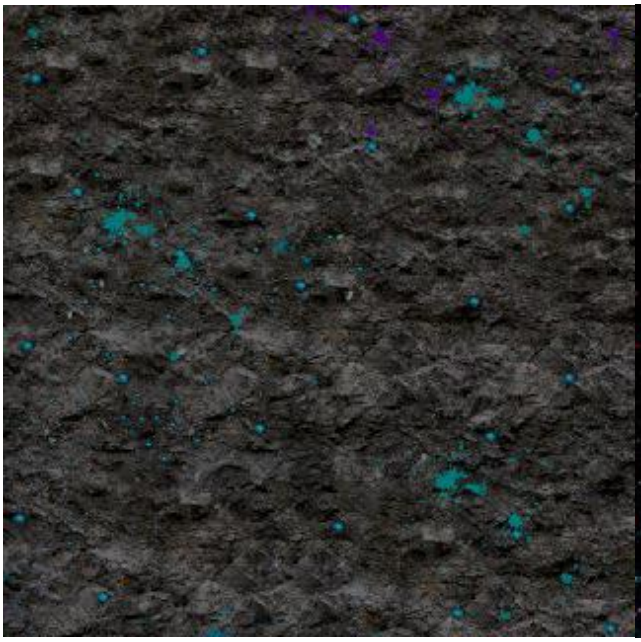
A. Paredes de la cueva

Para asemejar las paredes de la cueva a las de la gruta del enunciado, se ha creado una textura especial, compuesta por dos partes: la textura inicial y la textura emisiva.

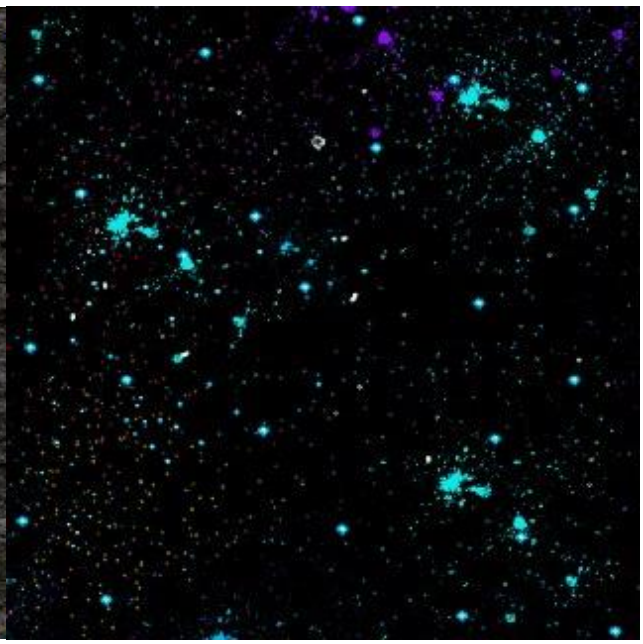
Para la textura inicial se ha utilizado una textura de roca. A la misma se le han añadido las pequeñas iluminaciones que tendría la cueva, basándonos en las imágenes de referencia de la cueva de Nueva Zelanda, la referida en las fotografías del enunciado.

Para la textura emisiva, se han incluido las mismas luces sobre un fondo negro.

Hay que destacar el gran tamaño de las texturas finales (4096 x 4096), ya que se aplican a toda la cueva una sola vez (no se repiten, cada parte del recorrido de la cueva es distinta), y la cueva en sí tiene un tamaño considerable.



Detalle de la textura de la cueva



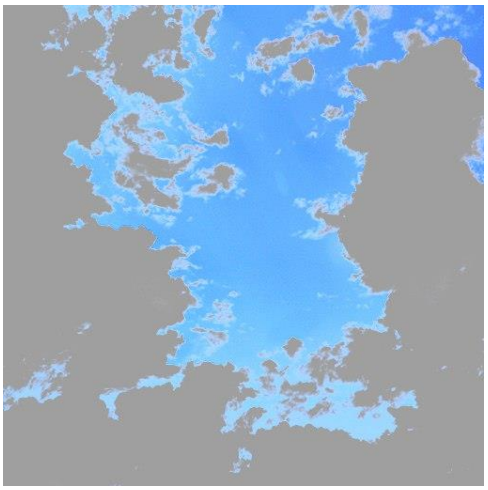
Detalle de la textura emisiva de la cueva

B. Entradas de luz

Para las entradas de luz se ha utilizado un plano, como para las paredes de la cueva, pero sin aplicar deformación y con los índices cambiados (por lo que el plano "mira" hacia abajo). Este plano aparece aleatoriamente en cualquiera de los puntos de control de la cueva (de los 8 puntos iniciales desplazados en el plano XZ), y a una altura determinada. El plano simplemente corta a las paredes de la cueva. Al aplicarle la textura, esta hace efecto de cielo, por lo que parece que la cueva tiene un hueco por el que se puede ver el exterior.

Para aplicar la textura de cielo, al igual que para las paredes, se ha utilizado una textura inicial y una emisiva.

Textura inicial cielo

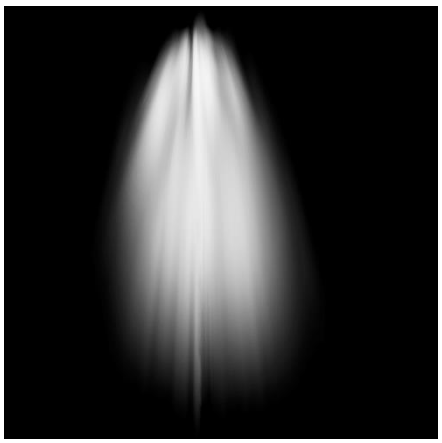


Textura emisiva cielo

C. God rays

Para realizar los God rays se ha colocado un plano de manera vertical en el centro de cada entrada de luz de la cueva. Se ha aplicado un blending a la textura, que hace que el plano se vuelva transparente.

La textura aplicada es la siguiente, en formato .png:



Textura God ray

D. Caminos en la cueva

Cada vez que el código se ejecuta, se crea un recorrido nuevo. Posteriormente la cámara realizará ese recorrido una y otra vez hasta que el programa se deje de ejecutar.

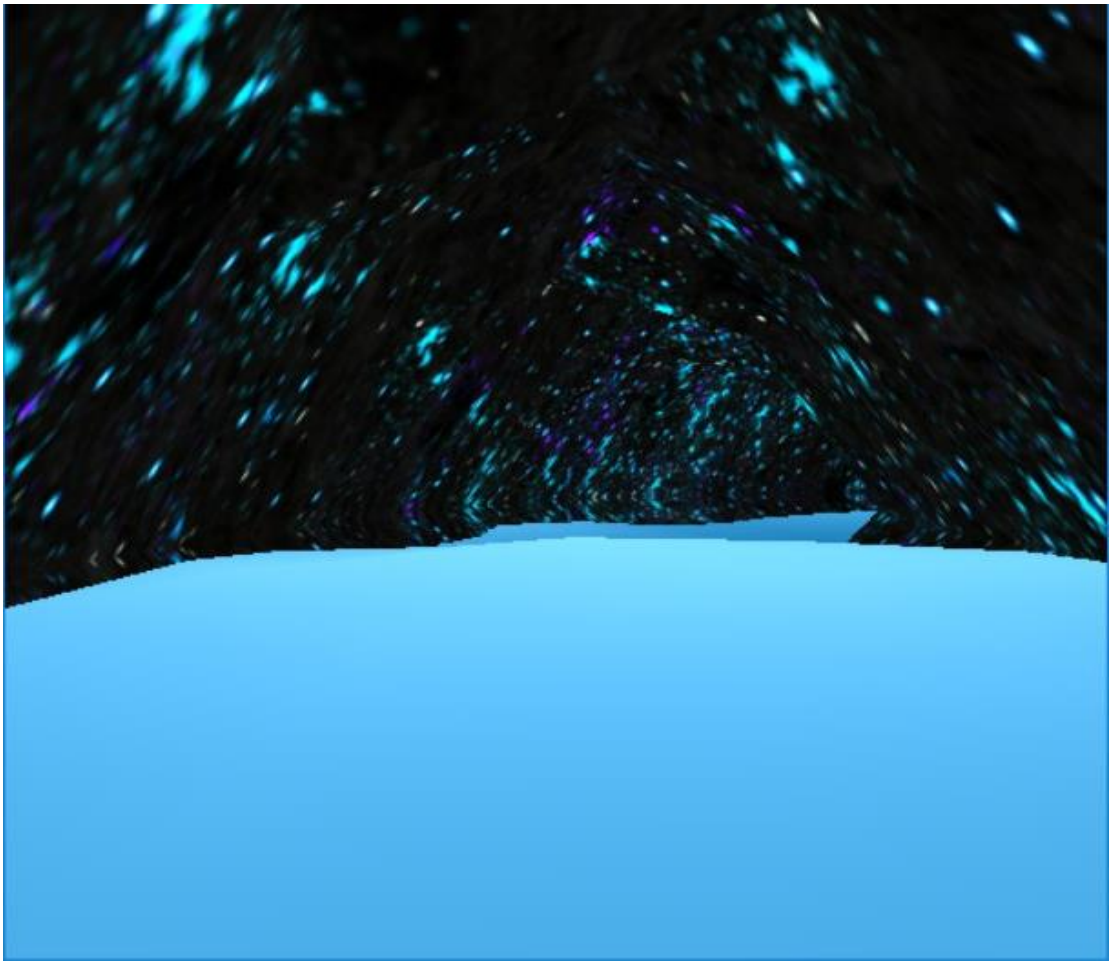
Puede ocurrir que el recorrido pase por el mismo punto del espacio dos veces, y se formen cruces. Aquí se creará un efecto que dará al usuario la sensación de tener múltiples caminos.

Estas intersecciones resultan muy interesantes, ya que pueden dar lugar a resultados inesperados (dos túneles casi a la par, formación rocosa en el centro del túnel, etc). Podremos ver varios de estos resultados en el siguiente apartado, resultados finales, y por supuesto, cuando ejecutemos el código de la práctica.

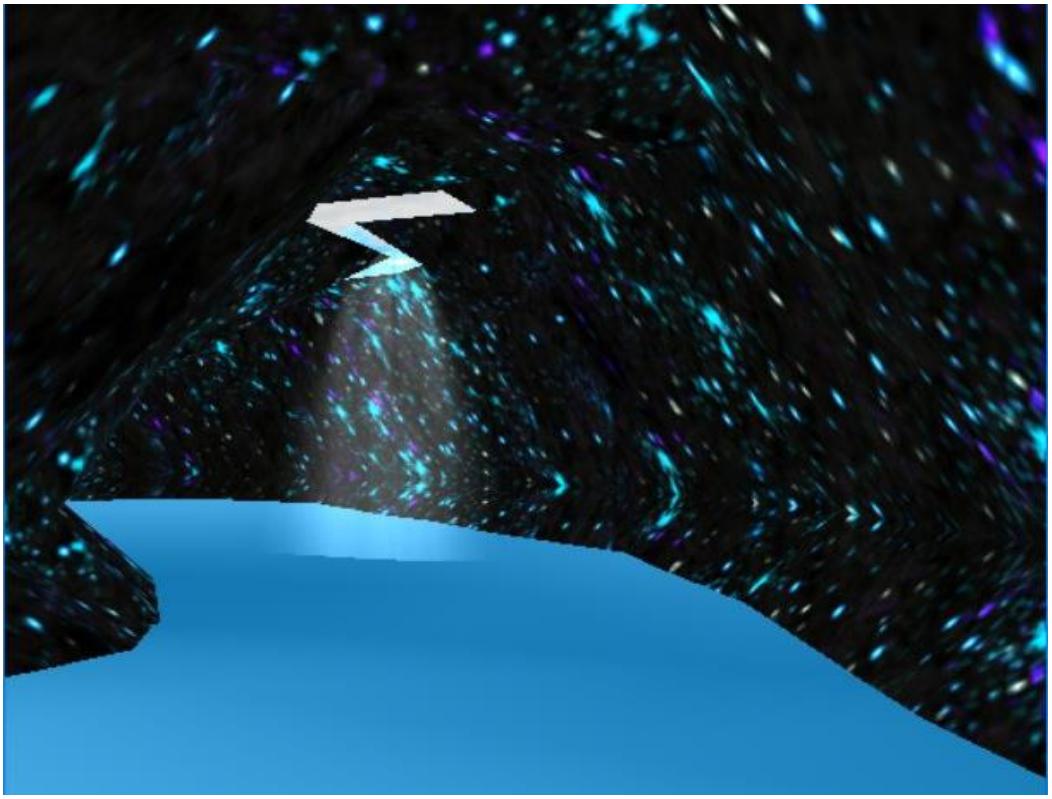
IV. Resultados finales

A continuación se presentan varias imágenes con los resultados finales de la cueva. También hemos creado un vídeo del resultado final, que se puede encontrar en el siguiente enlace:

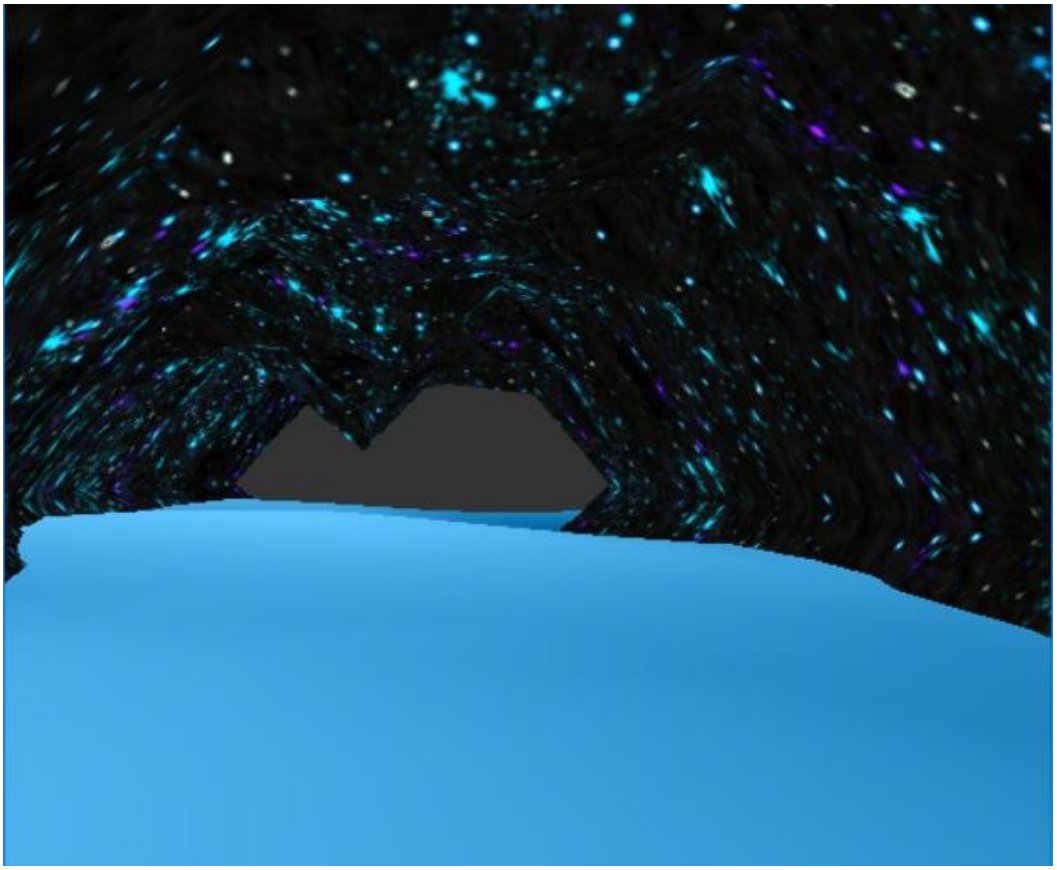
https://www.youtube.com/watch?v=rQINSGy5uKs



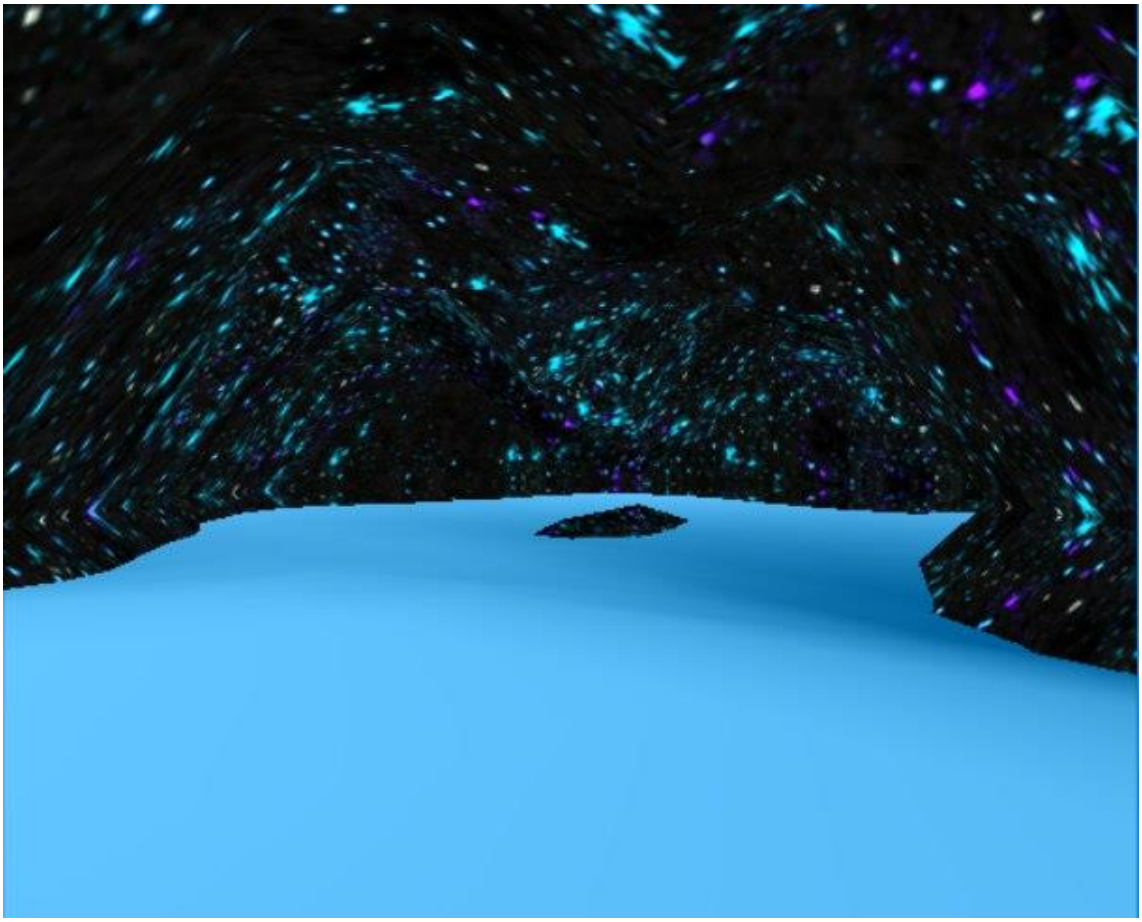
Recorrido por la cueva



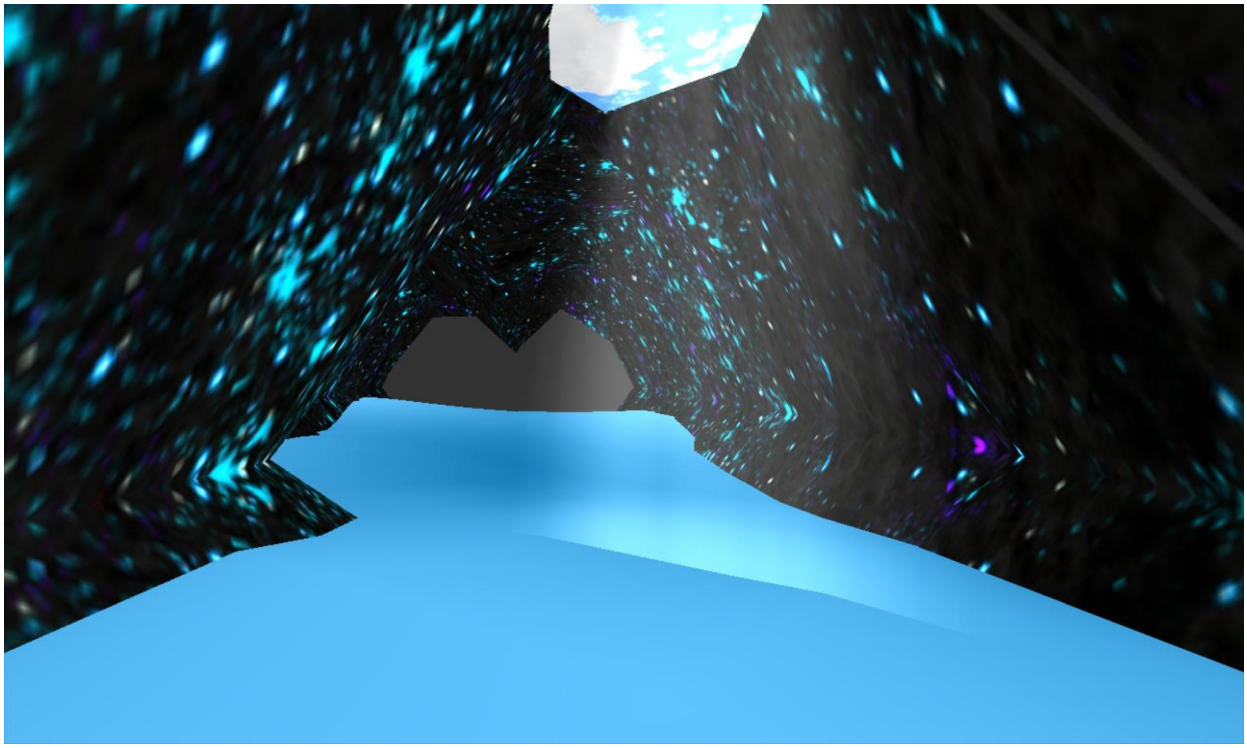
Aquí se puede apreciar la entrada de luz y el God ray



Dos túneles interseccionando



Intersección de dos túneles, formación rocosa



Conjunto de varios efectos

V. Referencias

[1] https://www.mvps.org/directx/articles/catmull/

[2] https://stackoverflow.com/questions/5915753/generate-a-plane-with-triangle-strips

[3] https://stackoverflow.com/questions/21272465/glsl-shadows-with-perlin-noise

[4] https://gist.github.com/patriciogonzalezvivo/670c22f3966e662d2f83