MÓDULO PRÁCTICO 1

-

MAPAS DE DESPLAZAMIENTO CON NIVELES DE DETALLE

Jaime Elcarte Fontcuberta
Manuel Akira Shinohara Soto

1. INTRODUCCIÓN

En este memoria se van a especificar, de forma breve, los procesos y tareas que se han llevado a cabo para la realización del módulo práctico 1. Usando como apoyo imágenes de los resultados obtenidos se explicarán los dos submódulos que componen esta práctica.

2. CONFIGURACIÓN DEL PROYECTO

Debido a que la práctica requiere de un motor render, se optó por utilizar un motor programado durante la asignatura de “Gráficos 3D”. A pesar de que el proyecto contiene todos los componentes básicos para comenzar a sacar renders, la escena de la que se partía era un simple cubo, por lo que decidimos meter una librería que nos facilitase meter otros modelos a la escena.
Sin embargo, aquí nos surgió un imprevisto, donde nuestro programa era incapaz de convertir un string de un número decimal separado por un punto a un float. En su lugar el programa debe leer la separación por una coma, y debido a que los archivos .obj vienen en su mayoría los números decimales separados por punto, tomamos la decisión de modificar los ficheros reemplazando los puntos por comas.

Gráfico 1: modelo de mono con fichero .obj modificado

3. SUBMÓDULO - 1

En este primer módulo de la práctica se ha realizado una toma de contacto con dos etapas dentro del cauce gráfico: geometría y teselación. Los shaders que implementan dichas etapas se van a ubicar en la segunda pasada del programa, pues la primera servirá para generar la malla del objeto en la escena.

3.1. Etapa de geometría

Durante esta etapa se pide generar primitivas básicas sobre la malla inicial:

- Puntos sobre los vértices
- Normales de los vértices y facetas
- Alambre sobre las aristas que componen las facetas

Para que el shader de geometría no arruine el pipeline del cauce gráfico es necesario que pase las variables de salida del shader anterior hasta las variables de entrada del shader que viene a continuación.
Una vez conseguido, para generar los **puntos** configuramos el shader para que a pesar de recibir como primitiva triángulos, sea capaz de sacar como primitivas puntos.

Fichero: pp_points.geom

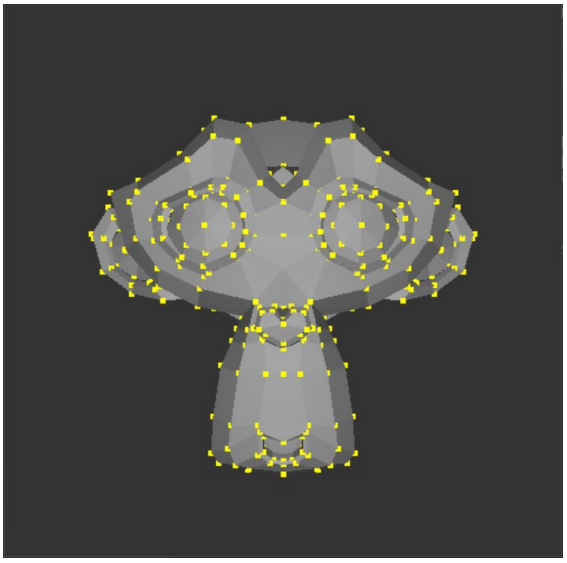


Gráfico 2: modelo de mono con puntos en los vértices

Generar las **normales** requiere de algunas operaciones sobre vectores como el producto vectorial. Vamos a recibir la misma primitiva en este caso, triángulos, y debido a que queremos generar líneas, las primitivas que se sacan seran line_strip.

Fichero: pp_normals.geom

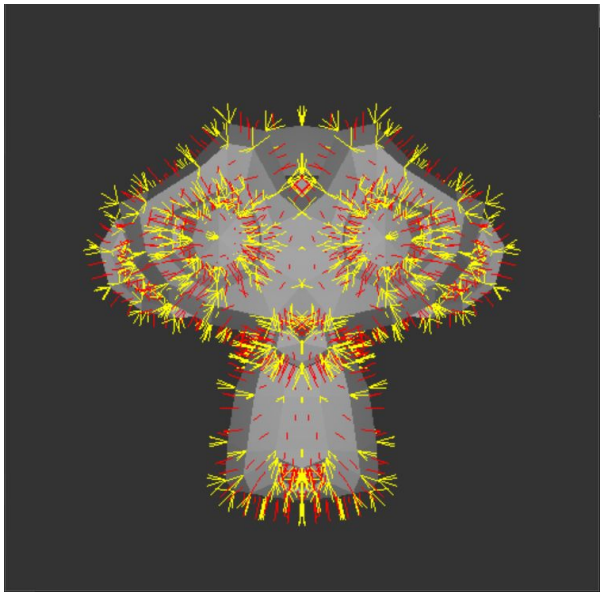


Gráfico 3: modelo de mono con normales de vértices y facetas

Por último generar un **alambrado** sobre la malla sólo requiere de conocer la posición de los vértices que forman las aristas y generar como primitivas line_strip.

Fichero: pp_wired.geom

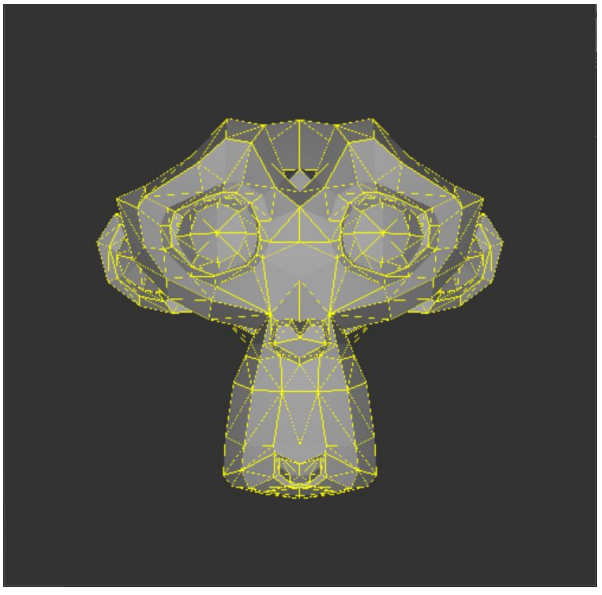


Gráfico 4: modelo de mono con aristas resaltadas

3.2. Etapa de teselación

La funcionalidad que se va a ver de esta etapa consiste en la subdivisión de la malla, y para ello se van programar 2 de los 3 shaders que la componen: Tessellation Control Shader (TCS) y Tessellation Evaluation Shader (TES). Para comprobar su funcionamiento se crean dos modelos nuevos, uno que representa un quad y otro un triángulo. Debido a que subdividir un triángulo requiere de una configuración diferente del quad, se usan dos pares de shaders diferentes para cada uno.
En la **TCS** vamos a indicar el número de subdivisiones tanto para el nivel exterior como el interior. Mientras que en la etapa de **TES** se va interpolando posición, normal, coordenadas de textura y color para los nuevos vértices que salen de dicha subdivisión.

Quads
Ficheros: pp_quad.tcs y pp_quad.tes

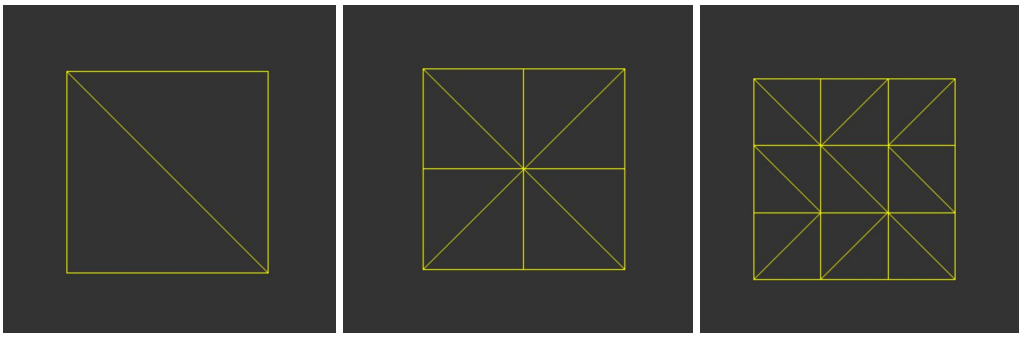


Gráfico 5: Subdivisión del quad. De izquierda a derecha 1, 2 y 3 subdivisiones tanto externas como internas

Triángulos
Ficheros: pp_triangle.tcs y pp_triangle.tes

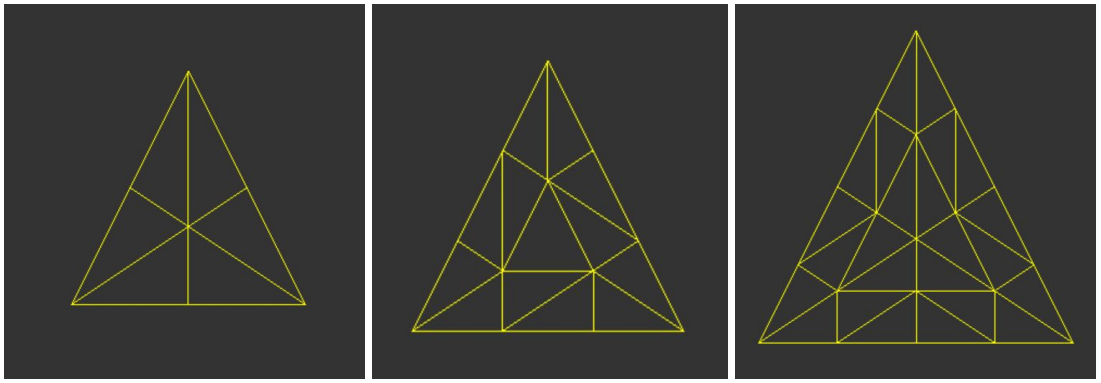


Gráfico 6: Subdivisión del triángulo. De izquierda a derecha 2, 3 y 4 subdivisiones tanto externas como internas

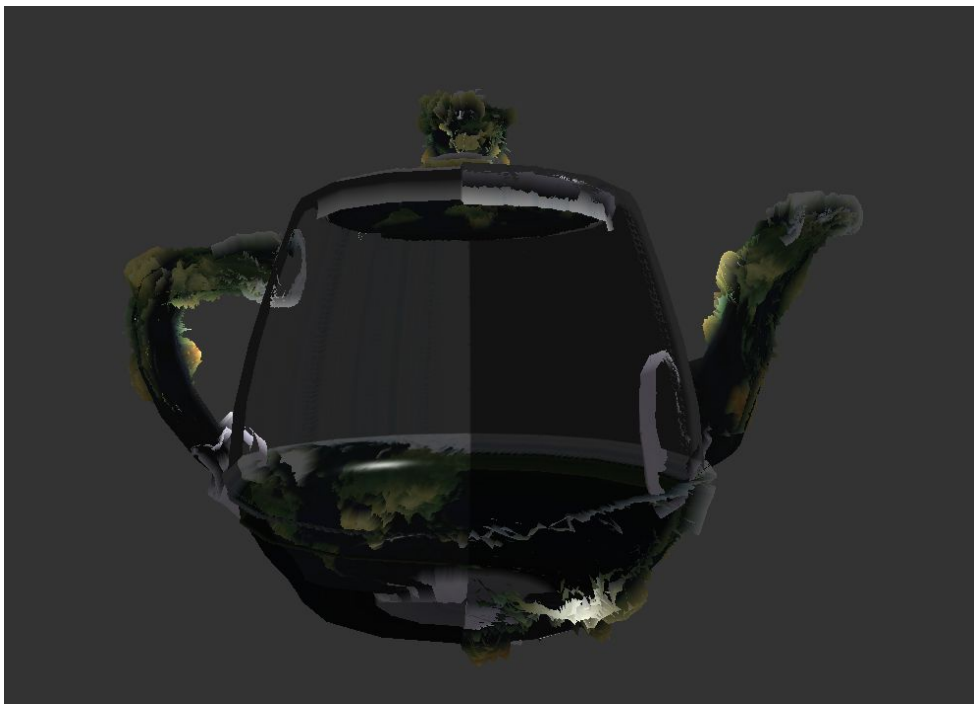
4. SUBMÓDULO - 2

En este segundo módulo de la práctica se ha implementado un modelo de mapas de desplazamiento y niveles de detalle dinámicos. Estas técnicas se han implementado a lo largo de las distintas etapas. Hemos utilizado el modelo de una tetera, y una imagen de un mapamundi como textura a utilizar.

Ficheros: pp_map.vert, pp_map.tcs, pp_map.tes, pp_map.geom y pp_map.frag

4.1. Mapas de desplazamiento

Para realizar esta técnica hemos modificado varios shaders, concretamente el shader de vértices y el shader de evaluación de teselación, para que el desplazamiento se realice en la dirección del vector normal de cada vértice. Para que cada vértice tenga un desplazamiento distinto, hemos tomado una componente del color y la hemos utilizado como factor de desplazamiento.



Gráficos 7, 8 y 9: Variación en el desplazamiento de los vértices en la dirección de sus vectores normales

4.2. Niveles de detalle dinámicos

La implementación de esta técnica requiere implementar un algoritmo de variación de la teselación en el shader de control teselación. En este caso hemos utilizado como medida la distancia media entre los vértices de cada arista y la posición de la cámara. Además, es necesario tomar el color de la textura asociado a cada vértice en el shader de evaluación de teselación. De esta forma, la definición de la textura varía en función de la distancia a la que se encuentra el objeto de la cámara.

Gráficos 10, 11 y 12: Variación en el nivel de detalle en función de la distancia a la cámara

5. Controles

Ubicación del proyecto: OpenGL-Engine/P4OGL/P4OGL.sln

- Desplazamiento de la cámara:
    - W (Adelante)
    - A (Izquierda)
    - S (Atras)
    - D (Derecha)
    - N (Abajo)
    - M (Arriba)
- Rotación de la cámara:
    - J (Izquierda)
    - L (Derecha)
- Variación de la teselación (solo en submódulo 1):
    - + (Aumentar)
    - - (Disminuir)
- Variación del desplazamiento (solo en submódulo 2):
    - + (Aumentar)
    - - (Disminuir)

6. Referencias

- E. Meiri. (2010, Oct 19). OpenGl Step by Step - OpenGL Development [Online]. Available: http://ogldev.atspace.co.uk/index.html
- J. de Vries. (2014, June). Learn OpenGL [Online]. Available: https://learnopengl.com/