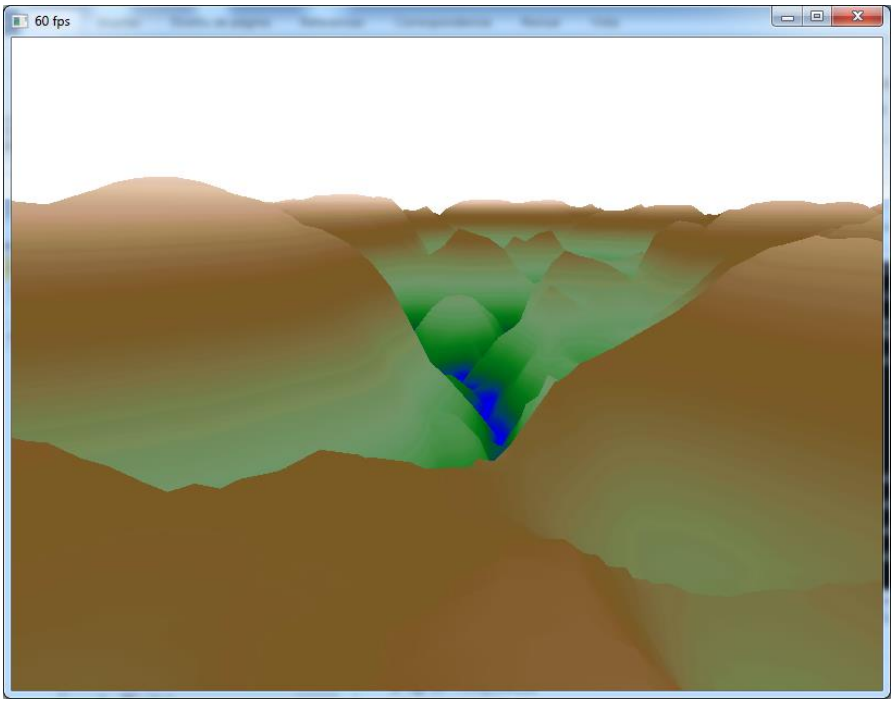
Práctica 7

Objetivo

En esta práctica se estudiará el uso de los shaders de teselación para implementar un visualizador de paisajes virtuales.

Tu trabajo



El objetivo de esta práctica es implementar un paisaje tridimensional utilizando el dibujo instanciado y la teselación de OpenGL. La aplicación final deberá dibujar un parche plano con 64 x 64 instancias, que el teselador irá subdividiendo teniendo en cuenta la cercanía a la cámara de cada instancia. Usando la técnica del *displacement mapping*, se modificará la altura de cada vértice generado durante la teselación, según una textura en blanco y negro que codifica la altura en cada punto (entre 0 y 1, pero luego puedes aplicarle una escala en el shader para obtener el resultado deseado). Además, el color final de cada fragmento se obtendrá a partir de la altura del vértice, utilizándola como índice para acceder a un color almacenado en una textura 1D.
Esta práctica está inspirada en el ejemplo de la página 323 de la Superbiblia, 7ª edición. Puedes usar dicho ejemplo como referencia.

En el código suministrado sólo se dibuja un quad que representa todo el terreno, al que se le aplica una teselación fija. En primer lugar, deberás dibujar, en vez de un quad, 64*64 quads, utilizando el dibujo instanciado de OpenGL. En el shader de vértice deberás calcular la posición de cada vértice y su coordenada de textura, para replicar el estado inicial (es decir, cubriendo la misma área, y usando toda la textura, sin repeticiones).

Después, en el shader de control de teselación, calcula en tiempo de ejecución los niveles de teselación, para usar polígonos más grandes (factores de teselación menores) en aquellos parches que estén más lejos de la cámara, y viceversa. Asegúrate de no introducir agujeros en la malla.
En el shader de evaluación de teselación tendrás que implementar la técnica del *displacement mapping*, elevando cada vértice a la altura correspondiente del mapa de alturas almacenado en la textura.
Por último, en el shader de fragmento, tendrás que obtener el color final a partir de la altura del fragmento, utilizando la escala de colores dada en el fichero `colorScaleHM.png` (tendrás que crear un objeto `Texture1D` y conectarlo con el shader). Dicha textura tiene los colores preparados para que los valores más pequeños (cercanos al cero) aparezcan azules, representando agua, y los mayores (cercanos al uno) aparezcan blancos, representando la cima de las montañas:

Minimiza los efectos *popping* configurando adecuadamente el espaciado de teselación.
Nota: la aplicación debería funcionar a la velocidad de refresco del monitor. Si tu aplicación va demasiado lenta, revisa tu función de dibujado y la teselación que se está produciendo realmente (visualiza la escena en alámbrico pulsado Ctrl+M y controla el número de primitivas rasterizadas con Ctrl+X).

Extensiones

Podrás obtener más nota introduciendo un “sol” a la escena, de tal forma que en cada punto se calcule la iluminación dependiendo del ángulo que forma su normal con la dirección al sol (define la posición del sol como un uniform de tu programa, y conéctalo a un `Vec3SliderWidget` para que se pueda mover desde la GUI).
Para calcular la normal en cada vértice tienes que usar el mapa de altura. Este proceso se puede hacer online, en el shader, muestreando los texels vecinos de cada vértice para calcular la normal, o precalcularlo a partir del mapa de alturas y almacenarlo en una textura (en este caso, un mapa de normales). Gimp y Photoshop tienen plugins para hacer este proceso.