Haskell para la computación gráfica

Ricardo Rubén González García

December 2020

1 Propósito

Demostrar que Haskell puede utilizarse para otras aplicaciones aparte de las implementaciones matematicamente cargadas a las que se asocia la programación funcional. Con este proyecto se intentará aumentar el interés, de implementar la programación declarativa para otras áreas más allá de las más obvias así como dar un pequeño trasfondo teórico al programa realizado.

2 Meta

La meta más grande del proyecto será crear un escenario 3D generado proceduralmente, sin embargo, por la falta de tiempo y las posibles complicaciones técnicas esta meta podría ser poco realista. Sin embargo, lo que se espera lograr como mínimo es: la importación de un objeto creado en algún software de modelado (leyendo un .obj o parecido), poner luces, texturas, rotaciones y movimiento de la cámara.

3 Bibliotecas

Investigando un poco acerca del tema, me encontré con las bibliotecas *Graphics.Rendering.OpenGL* y *Graphics.UI.GLUT* e incluso un pequeño tutorial para la graficación gráfica con Haskell.

Se usará la versión de OpenGl 2.1 debido a limitantes de hardware.

4 Posibles funciones:

Una vez que consiga resolver todos los problemas que vienen con todo esto de cabal, este sería mi plan a grandes rasgos:

1. Usar las funciones básicas de OpenGl para crear la ventana, el viewport y la cámara.

   Ya que tenga una ventana funcional, con display y responsiva, lo siguiente es crear el escenario.

2. Para crear el escenario, planeo usar coordenadas x,y,s. Las coordenas x,s serán suministradas a mano y serán como las coordenadas de un arreglo. La coordinada y, será creada por *Math.Noise.Modules.Perlin* para generar valores aleatorios parecidos entre sí (para que las alturas se vayan pareciendo entre sí y tengamos montañitas o valles y no puntos completamente disparejos). Se unirá todo con un triangle strip.

3. Las luces serán creadas con primitivas de OpenGl.

4. Con todo esto tendremos un terreno vacío, pero cambiante.

5. Faltaría los elementos del paisaje. La idea es buscar elementos ya creados en internet que tengan una extensión .obs. Los .obs tienen un formato ya predeterminado y es solo leer un archivo de texto https://en.wikipedia.org/wiki/Wavefront_.obj_file.

6. Ya con los elementos leídos, faltaría meterlos en la escena y sería con una función que los meta en puntos aleatorios (que ya estarían definidos por el perlin noise)

   Este sería el plan a grandes rasgos, sin embargo, puede que cambie un poco a la hora de la implementación.

5 Referencias:

Investigando un poco, estos son los sitios que creo que me podrían servir de ayuda, aunque no los vi a fondo.

https://wiki.haskell.org/OpenGLTutorial1

http://www.lcc.uma.es/~blas/apuntes/PDAv/T2008-2009/G7HOpenGLPresentacion.pdf

https://www.cin.ufpe.br/~haskell/hopengl/first.html