Computación Gráfica - TP: Toon Shading

1. Resumen de tareas

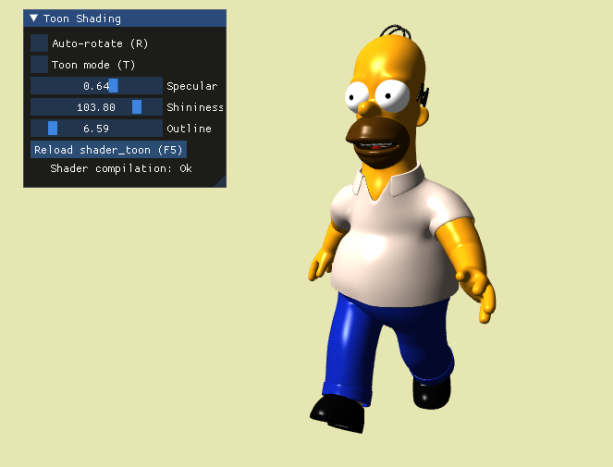
1. Analizar la implementación del modelo de Phong y relacionarla con lo desarrollado en la teoría.
2. Modificar la implementación de Phong para lograr un sombreado estilo *cartoon* (cómic o dibujo animado).
3. Analizar cómo se generan las lineas negras que resaltan los contornos.

2. Consigna detallada

El objetivo de este práctico es analizar una implementación básica del modelo de Phong y aplicar las modificaciones necesarias para generar uno de los efectos de renderizado no-fotorealista (NPR) más comunes, denominado *Toon*-shading o *Cell*-shading.

2.1. Sombreado de Phong

El código inicial incluye una implementación del modelo de sombreado de Phong (en el archivo `phong.frag`). Primero deberá analizar esta implementación e interpretar cómo el código se relaciona con las ecuaciones desarrolladas en la clase de teoría.



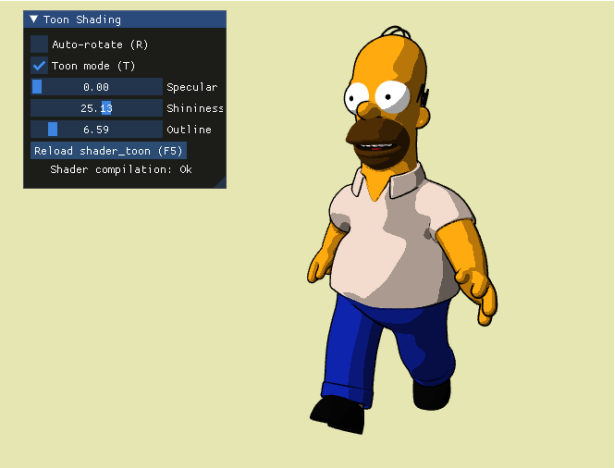
Esta implementación, por tratarse de un *shader* (en particular un *fragment shader*, se encuentra en el lenguaje *GLSL*). Puede recurrir al apunte de ”Introducción a OpenGL moderno”[1] para encontrar algunas ideas básicas sobre la programación con shaders.

2.2. Sombreado Plano

El archivo `toon.frag` es inicialmente idéntico a `phong.frag`. **El alumno debe modificar esta implementación para lograr un sombreado estilo *toon*.** Este sombreado se caracteriza por reemplazar la transición *suave* de color desde

[1] Disponible en el aula virtual, entre el material adicional de la unidad introductoria.

las zonas poco iluminadas a las más iluminadas, por una transición escalonada, utilizando unos pocos escalones ”planos” (de un mismo color, de aquí que también se le suele llamar sombreado *plano*).



Notar que no es necesario reiniciar el programa cada vez que se modifica el código fuente del shader. La GUI dispone de un botón (y un atajo de teclado: F5) para recargar el shader[2] sin necesidad de reiniciar el programa. Si el shader fuera incorrecto (por alguna razon falla su compilación), la GUI simplemente indicará que ocurrió un error, pero el detalla (los mensajes de error del compilador de shaders) aparecerán en la terminal de ejecución del programa.

Opcional: una vez que haya logrado una implementación funcional del sombreado (probablemente emulando en GLSL la forma de implementación que utilizaría en un programa C/C++), puede investigar el funcionamiento de las funciones `step` y `smoothstep` del lenguaje *GLSL* y analizar la posibilidad de utilizarlas para optimizar el funcionamiento de su programa.

2.3. Lineas de Contorno

La otra característica de este estilo de renderizado es que resalta los contornos con lineas negras. Este efecto ya está implementado en el práctico. **El alumno debe analizar cómo se logra este efecto en este caso.**

Para comprenderlo, deberá analizar el *vertex shader* utilizado (`lines.vert`). Dado que si no se ha estudiado aún la unidad de *Espacios y Transformaciones*, probablemente no entienda por completo el funcionamiento de un *vertex shader*, puede compararlo alguno de los otros dos *vertex shaders* (`phong.vert` o `toon.vert`, son idénticos) para encontrar cuál es la diferencia, y luego tratar de interpretarla.

Ayuda: la GUI dispone de un slider para variar el grosor de las lineas de contorno. Si intenta generar un contorno demasiado grueso, los defectos que se observan pueden ayudar a entender o confirmar el método implementado.

[2] solo se recarga el shader *toon* (de los archivos `toon.vert` y `toon.frag`), no así *phong* y *lines*.