Práctico Nº 2

Introducción:
El objetivo de este práctico es guiar al estudiante en la investigación de técnicas para realizar *rendering* de objetos en movimiento, cargado de datos, rendering de luces y simulación de transparencias.

Ejercicio 1: (ejecutable *movement*)

Basado en el Ejercicio 3 del práctico 1, genere una aplicación SDL + OpenGL en donde los dos objetos que se dibujan (triángulo y cuadrado) roten sobre sí mismos según el eje Y, y además roten según el eje Y relativo al punto (0, 0, -6).
Ambos movimientos tienen que tener una frecuencia de 0.2 Hz (una vuelta completa cada 5 segundos).

Notas: para lograr un movimiento independiente del *Frame Rate*, hay que calcular la nueva posición de los objetos dependiendo del tiempo transcurrido entre dos render consecutivos (anterior y actual).

Extras: puede probar diferentes/múltiples ejes de rotación tanto relativa como absoluta, y diferentes velocidades/aceleraciones angulares.

Ejercicio 2: (ejecutable *loadPolygon*, datos *polygons.txt*)

Genere una aplicación SDL + OpenGL que cargue los datos de posición/color de vértices y los dibuje con las siguientes características:

- El formato del archivo es el siguiente:
  - La línea que empieza con la letra ***C*** define un color (usar *glColor3f*)
  - La línea que empieza con la letra ***V*** define una posición (usar *glVertex3f*)
  - El resto de la línea corresponde al dato de color/posición.
- Los datos cargados se deben dibujar utilizando la primitiva *GL_TRIANGLES*.
- El conjunto de triángulos debe dibujarse relativos al punto (0, 0, -6), tiene que rotar según el eje Y relativo a dicho punto a 0.2 Hz.
- Al presionar la tecla *ESPACIO* se tiene que recargar los datos desde el archivo, y desde ese momento se tiene que dibujar la nueva información. De esta manera se puede actualizar la geometría sin la necesidad de re-ejecutar o re-compilar.

Notas: se aconseja definir un tipo de dato apropiado para la información extraída desde el archivo, almacenarlos utilizando un *std::list* o *std::vector* y por último realizar una recorrida para hacer el dibujado en pantalla.

Ejercicio 3: (ejecutable *lights*, datos *lightPolygons.txt*)

Genere una aplicación SDL + OpenGL que cargue los datos de posición/normal de vértices desde un archivo y dibuje con las siguientes características:

- El formato del archivo es el siguiente:
  - La línea que empieza con la letra ***V*** define una posición (usar *glVertex3f*)
  - La línea que empieza con la letra ***N*** define una normal (usar *glNormal3f*)
  - El resto de la línea corresponde al dato de posición/normal.
- Los datos cargados se deben dibujar utilizando la primitiva *GL_QUADS*.

- Al presionar la tecla *ESPACIO* se tiene que recargar los datos desde el archivo, y desde ese momento se tiene que dibujar la nueva información. De esta manera se puede actualizar la geometría sin la necesidad de re-ejecutar o re-compilar.
- Se dibuja una luz que gira alrededor del modelo a razón de 0.2 Hz.
- La luz es de tipo difusa de color BLANCO

La rutina de rendering tiene que realizar los siguientes pasos en orden:

1. Limpiar los buffers (color y profundidad) y cargar la identidad en la matriz *GL_MODELVIEW*.
2. Realizar una traslación (0,0,-6).
3. Invocar la función *glPushMatrix*.
4. Realizar una rotación según el eje Y el ángulo acumulado para generar un movimiento circular uniforme a razón de 0.2 Hz.
5. Realizar una traslación de (2, 1, 1.5). Ésta va a ser la posición relativa de la luz según el punto (0, 0, -6) rotado previamente.
6. Deshabilitar las luces y dibujar un objeto centrado en el (0, 0, 0) (según el sistema de coordenadas local) para poder visualizar la posición de la luz.
7. Habilitar las luces y posicionar la *GL_LIGHT0* en el (0, 0, 0). El cuarto parámetro de la posición de la luz tiene que ser 1.
8. Invocar la función *glPopMatrix*.
9. Realizar una rotación de 30º según el eje Y.
10. Dibujar el modelo cargado desde el archivo.

Ejercicio 4: (ejecutable *blendMode*)

Genere una aplicación SDL + OpenGL con las siguientes características:

- Se deben dibujar 8 cuadriláteros de color y sin transparencia tal que estén uno al lado del otro y la dimensión según el eje X sea mayor que la dimensión según el eje Y. Además se tiene que cubrir la mayor parte del *display*.
- Se deben dibujar 8 cuadriláteros de color y con transparencia variable tal que estén uno al lado del otro y la dimensión según el eje Y sea mayor que la dimensión según el eje X. Además se tiene que cubrir la mayor parte del *display*.
- Cada cuadrilátero transparente tiene que ser dibujado sobre todos los cuadriláteros opacos. No puede estar sobre otro cuadrilátero transparente.
- Los colores a utilizar para pintar los cuadriláteros deben ser:
  - *glColor3f(0, 0, 0)* y *glColor4f(0, 0, 0, alpha)*
  - *glColor3f(1, 0, 0)* y *glColor4f(1, 0, 0, alpha)*
  - *glColor3f(0, 1, 0)* y *glColor4f(0, 1, 0, alpha)*
  - *glColor3f(1, 1, 0)* y *glColor4f(1, 1, 0, alpha)*
  - *glColor3f(0, 0, 1)* y *glColor4f(0, 0, 1, alpha)*
  - *glColor3f(1, 0, 1)* y *glColor4f(1, 0, 1, alpha)*
  - *glColor3f(0, 1, 1)* y *glColor4f(0, 1, 1, alpha)*
  - *glColor3f(1, 1, 1)* y *glColor4f(1, 1, 1, alpha)*
- El valor de *alpha* utilizado para definir el color de los cuadriláteros transparentes tiene que variar entre 0 y 1. El cambio de transparencia tiene que ser independiente del *Frame Rate*.