

## Práctico N° 3

### Introducción:

El objetivo de este práctico es plantear un conjunto de problemas gráficos a resolver para que el estudiante investigue y genere conocimiento en temas específicos.

### Ejercicio 1: ( ejecutable *displayList* )

Implemente una aplicación SDL + OpenGL en donde se dibuje un campo de figuras geométricas idénticas que se actualizan dependiente del tiempo transcurrido, distinguibles por su color y posición. La forma en la que se tiene que dibujar la escena es la siguiente:

- Dibujar un elemento en la posición (0,0,0) para poder identificar el origen de coordenadas.
- Pedir un identificador de *displaylist* (*glGenLists*)
- Se debe calcular la nueva posición de los elementos que conforman la figura geométrica y compilarlo en la *displaylist* recervada anteriormente. Durante el compilado no se puede modificar el estado del color que utiliza OpenGL.
- Dibujar 100 veces el *displaylist* compilado, asignando los colores y posiciones correspondientes a cada figura. Esta información es calculada y almacenada por única vez al inicio de la aplicación.
- Liberar la memoria asociada al *displaylist* utilizado ya que no se va a volver a utilizar (*glDeleteLists*).

La cámara se tiene que poder mover a través del plano XY utilizando las teclas direccionales.

**Notas:** dado que al compilar la figura no se modificó el color que utiliza OpenGL, se puede dibujar cada figura de un color distinto sin necesidad de compilar más de una *displayList*.

### Ejercicio 2: (ejecutable *sprites*)

Implemente una aplicación SDL + OpenGL que cumpla con las siguientes características:

- La inicialización de la matriz *GL\_PROJECTION* se debe hacer utilizando *glOrtho* o *gluOrtho2D* para especificar una proyección ortogonal.
- Dibujar el fondo con un gráfico enlosable (del inglés: *tileable*) horizontalmente (existe una continuidad entre los pixels más a la derecha con los pixels más a la izquierda).
- El movimiento del fondo tiene que presentar una característica de paralaje (las capas que se encuentran más lejos se mueven más lento).
- Se debe dibujar un *sprite* animado (secuencia de imágenes) tal que:
  - Una única textura contiene todos los fotogramas (*frames*) de la animación.
  - La secuencia de fotogramas a dibujar (coordenadas de textura) se deben cargar desde un archivo XML.
- Delante del *sprite* animado, se debe dibujar otra capa con paralaje.

**Notas:** se aconseja utilizar librerías específicas para lectura de archivos XML (ej: *tinyxml* o *pugixml*).

**Ejercicio 3:** (ejecutable *loadModel*)

Implemente una aplicación SDL + OpenGL que cargue un modelo 3D en algún formato de uso común (ej: \*.3ds). Además de dibujar el modelo cargado, se debe dibujar por lo menos un *quad* que cumpla la función de “piso”.

Se debe implementar una interacción sencilla entre el modelo y el entorno que cumpla con las siguientes características:

- La escena tiene “paredes” perpendiculares a los ejes de coordenadas sin un gráfico asociado (sólo se utiliza esta información para interactuar con el modelo).
- El modelo tiene que tener el movimiento de un proyectil:
  - Acelerado según el eje Y (gravedad) y uniforme en X y Z.
  - Las colisiones son inelásticas (hay pérdida de velocidad).
  - Cuando se detecta una colisión entre el modelo y una pared (o piso) se invierte el valor de la velocidad en el eje que corresponda (se asume que las paredes son perpendiculares a los ejes de coordenadas).
  - Las velocidades iniciales en X y Z son aleatorias.
  - Al pulsar la tecla *ESPACIO* se debe reiniciar la posición y velocidad del modelo.
- La rotación del modelo es proporcional a la velocidad según los ejes X y Z.
  - Se aconseja utilizar *quaterniones* para mantener la rotación del modelo, dado que la utilización de rotaciones euclidianas puede resultar más complicadas.

**Notas:** se aconseja utilizar librerías específicas para el cargado de modelos (ej: *assimp*)