

## Práctico N° 1

### Introducción:

El objetivo de este práctico es introducir al estudiante en el manejo de la librería gráfica OpenGL y el empleo de SDL para manejar ventanas y eventos (input - output)

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

Genere una aplicación SDL + OpenGL con las siguientes características:

- SDL debe utilizar *Double Buffer*.
- Modo de video con resolución de 640x480, 32 bits de color y las banderas *SDL\_HWSURFACE* y *SDL\_OPENGL*.
- El *Clear Color* tiene que ser (0.3, 0.3, 0.3)
- En cada *frame* nuevo a dibujar se tiene que limpiar el *Color Buffer*.
- Se tiene que manejar el evento de presionar la tecla *ESCAPE* y cerrar la aplicación cuando esto ocurra.

**Notas:** se tienen que consumir todos los eventos generados en cada *frame* y se tiene que invocar a la función *SDL\_gl\_swapbuffers*, de lo contrario la ventana quedará colgada.

**Extras:** puede manejar el evento de presionar otras teclas y asignarle un comportamiento a elección (ej: cambiar alguno de los componentes del *Clear Color*).

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

Basado en el Ejercicio 1, genere una aplicación SDL + OpenGL con las siguientes características:

- El *Clear Color* tiene que ser *Negro*.
- Al inicializar la aplicación cambiar a modo de matriz *GL\_PROJECTION*.
- Utilizar la función *gluPerspective* para cargar en la matriz de proyección los siguientes datos:
  - fovy a 45 grados
  - aspect ratio a 640/480 (o el que corresponda por el tamaño de la ventana)
  - *Near Plane* a 0.1
  - *Far Plane* a 100
- Cambiar a modo de matriz *GL\_MODELVIEW*.
- En cada *frame* nuevo a dibujar se tienen que limpiar el *Color Buffer*, el *Depth Buffer* y cargar la identidad en la matriz *GL\_MODELVIEW*.

#### Parte a)

- Dibujar un triángulo con sus vértices en las coordenadas (-1.5,1,-6), (-2.5,-1,-6) y (-0.5,-1,-6).
- Dibujar un cuadrilátero con sus vértices en las coordenadas (0.5,1,-6), (2.5,1,-6), (2.5,-1,-6) y (0.5,-1,-6).

#### Parte b)

- Hacer una traslación de (-1.5,0,-6)
- Dibujar un triángulo con sus vértices en las coordenadas (0,1,0), (-1,-1,0) y (1,-1,0)
- Hacer una traslación de (3,0,0)
- Dibujar un cuadrilátero con sus vértices en las coordenadas (-1,1,0), (1,1,0), (1,-1,0) y (-1,-1,0).

### Ejercicio 3: (ejecutable *colorPolygon*)

Basado en el Ejercicio 2, genere una aplicación SDL + OpenGL donde la escena sea un triángulo cuyos vértices son de color Rojo, Verde y Azul, y un cuadrilátero de color Cian. Presionando la tecla F11 tiene que cambiar entre modo Ventana y Pantalla Completa.

**Notas:** cuando se cambia de modo hay que volver a asignar los valores correctos de la matriz de proyección.

### Ejercicio 4: (ejecutable *texturePolygon*)

Basado en el Ejercicio 3, genere una aplicación SDL + OpenGL donde la escena sea un triángulo colorido (sin textura) y al lado un cuadrilátero al que se le mapea una textura.

Dicha textura (*opengl.png*) es de 512x512 pixels y tiene las siguientes características:

- Contiene el logo de OpenGL sobre fondo blanco.
- Contiene un recuadro rojo por afuera del logo de un pixel de grosor.
- El resto de la imagen es azul.
- Sobre el área azul se especificaron algunas distancias de interés.

**Parte a)** Dibuje el cuadrilátero realizando un mapeado completo de la textura sobre el mismo, de esta manera los límites de la imagen coinciden con los límites de la textura.

**Parte b)** Dibuje el cuadrilátero realizando un mapeado parcial de la textura, de tal manera que los límites del cuadrilátero coincidan con el recuadro rojo. Con las teclas *UP* y *DOWN* se podrá cambiar la escala al cuadrilátero. Dicho escalado tiene que ser referente al centro del cuadrilátero. Se debe mantener el *Aspecto* (relación entre ancho y alto) de la textura en todo momento.

**Notas:** antes de cargar la imagen como una textura en OpenGL, defina los parámetros de textura 2D *GL\_TEXTURE\_MIN\_FILTER* y *GL\_TEXTURE\_MAG\_FILTER* con el valor *GL\_LINEAR*.

El mapeo de texturas por defecto es con repetición, y puede generar artefactos en los bordes del cuadrilátero para ciertas relaciones entre *texels* y *pixels*. Modificando el parámetro de textura para que no haga *WRAP* se soluciona dicho artefacto.

**Extras:** puede probar realizar el cargado de texturas con diferentes tipos de filtros (también con *MIPMAPS*).

También puede probar con diferentes valores para el parámetro de textura *GL\_TEXTURE\_WRAP* y con coordenadas de texturas mayores que 1 y menores que 0.