Mallas poligonales y curvas cuádricas en OpenGL

Introducción

- Las aplicaciones openGL están construidas sobre un loop principal que se verá más adelante en las clases de openGL.
- El loop principal es donde se realizaran todas las transformaciones y tareas que se deban hacer en cada frame de nuestra aplicación.

Operaciones principales para dibujar mallas poligonales

- **`void glBegin(GLEnum mode);`**

  /* mode indica como estarán agrupados los vértices que se dibujarán*/

- **`void glVertex{234}{sifd}v( T coords );`**

  /* Será la función con la que especificaremos cada uno de los puntos de la malla poligonal*/

- **`void glEnd();`**

  /* Finaliza el dibujado de esa malla*/

glBegin

- glBegin(GLEnum mode);
  - Indica el comienzo de una malla poligonal.
  - Mode puede tomar los siguientes valores:
    - GL_POINTS: indica que cada vértice indicará un punto separado.
    - GL_LINES: cada par de puntos especificado será una línea.
    - GL_LINE_STRIP: cada punto nuevo se une al anterior mediante una linea. Por ejemplo: si ponemos v1, v2 y v3 vértices se dibujarán las lineas v1-v2 y v2-v3.
    - GL_LINE_LOOP: igual al anterior excepto que en el ejemplo se dibujará también v3-v1.
    - GL_TRIANGLES: dibuja un triángulo para cada terna de vértices especificada.
    - GL_TRIANGLE_STRIP: dibuja una malla de triángulos de la siguiente manera: si tenemos v1..v4 vértices se dibujarán los triángulos v1-v2-v3 y v2-v3-v4.

glBegin

- GL_TRIANGLE_FAN: igual al anterior pero se genera también el triángulo v4-v1-v2.
- GL_QUADS: funciona de la misma manera que GL_TRIANGLES pero toma cuaternas en lugar de ternas para dibujar cuadriláteros en lugar de dibujar triángulos.
- GL_QUAD_STRIP: una malla de cuadriláteros unidos al igual que pasaba con GL_TRIANGLE_STRIP.
- GL_POLYGON: dibuja un polígono convexo con los vértices especificados, el último vértice es unido al primero para hacer que el polígono sea realmente un polígono.

glVertex

- **`void glVertex{234}{sifd}v( T coords );`**
- Primero hay que aclarar que son variantes de la misma operación por ejemplo glVertex3f(1.0f,2.0f,3.0f); es una invocación válida.
- El 2, 3 o 4 indica la cantidad de parámetros que recibirá la función, pueden ser:
  - X e Y: en este caso Z=0.0 y W=1.0
  - X, Y y Z: en este caso W=1.0
  - X, Y, Z y W todos los valores son especificados

glVertex

- s, i, f o d indica el tipo que tendrán los parámetros:
  - **`s: short int.`**
  - **`i: int.`**
  - **`f: float.`**
  - **`d: double.`**
- Cualquier variante anterior puede llevar la v al final que indica que no se pasarán n parámetros sino un array con todos los valores necesarios. Si es glVertex3iv(...) se deberá pasar como parámetro un array de enteros con 3 valores.

glEnd

- Indica el final de la especificación de vértices.
- Entre el begin y el end se pueden especificar tantos vértices como se desee, todas las mallas a dibujar serán dibujadas, dependerá del mode especificado en el glBegin como serán tratados todos los glVertex que se utilicen entre el begin y el end.

NURBS

- Son un caso particular de las cuádricas.
- Está incluido en el glu no en la distribución original de openGL pero glu está dentro de la distribución estándar de openGL.
- Para dibujar una curva se debe poner al igual que para dibujar mallas poligonales un begin y un end, pero no son los mismos que antes.

NURBS

- Lo primero que tenemos que hacer es inicializar un renderer para las nurbs:
  - GLUnurbs* gluNewNurbsRenderer (void);

    /*Crea un renderer*/
  - void gluDeleteNurbsRenderer(GLUnurbs **nobj*);

    /*elimina la memoria del renderer*/
- void gluBeginSurface(GLUnurbs*)
  - Inicia el ingreso de la nurb.
- void gluEndSurface(GLUnurbs*)
  - Indica el fin de la especificación.

NURBS

- gluNurbsSurface(GLUnurbs* nurb,
  GLint sKnotCount,
  GLfloat* sKnots,
  GLint tKnotsCount,
  GLfloat* tKnots,
  GLint sStride,
  GLint tStride,
  GLfloat* control,
  GLint sOrder,
  Glint tOrder,
  GLEnum type);

NURBS

- GLUnurbs* nurb: el renderer de la curva.
- GLint sKnotCount: cantidad de nudos en la dirección paramétrica s.
- GLfloat* sKnots: nudos en la dirección paramétrica s
- GLint tKnotsCount: cantidad de nudos en la dirección paramétrica t.
- GLfloat* tKnots: nudos en la dirección paramétrica t.
- GLint sStride: especifica el offset entre dos puntos de control sucesivos en la dirección s.
- GLint tStride: especifica el offset entre dos puntos de control sucesivos en la dirección t.

NURBS

- GLfloat* control: puntos de control de la curva
- GLint sOrder: especifica el orden de la curva en la dirección s (cuádrica, cúbica, etc.)
- Glint tOrder: especifíca el orden de la curva en la dirección t.
- GLEnum type: GL_MAP2_VERTEX_3 o GL_MAP2_COLOR_4.