Evaluadores y NURBS

Integrantes:
Balboa Palma, Merly
Mirano Mitchell
Vasquez Sebastian

UNMSM–Computación Cientfica

6 de agosto de 2022

Contenido

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

Introducción

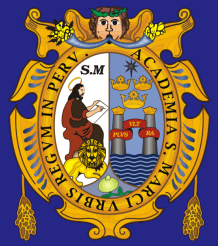
Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

- Las curvas y superficies suaves se dibujan aproximándolas con grandes o pequeños segmentos de línea o polígonos.
- Muchas curvas y superficies pueden describirse matemáticamente mediante un pequeño número de parámetros.
- Guardar 16 puntos de control para una superficie requiere menos almacenamiento que guardar 1000 triángulos junto con la información del vector normal en cada vértice. Además, sólo se aproximan a la superficie real, pero los puntos de control describen con precisión.

Introducción

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

- Si se quiere usar evaluadores para dibujar curvas y superficies utilizando otras bases, debemos convertir su base en una base de Bézier.
- Cuando se renderiza una superficie Bézier o parte de ella utilizando evaluadores, es necesario especificar el nivel de detalle de su subdivisión.



Evaluadores

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

- Los evaluadores proporcionan una forma de especificar puntos en una curva o superficie utilizando puntos de control.
- La curva o superficie puede ser renderizada con cualquier precisión. Además, los vectores normales pueden ser calculados automáticamente para las superficies.
- Los puntos generados por un evaluador pueden utilizarse de muchas maneras: Para dibujar puntos donde estaría la superficie, una versión alámbrica de la superficie y una superficie completamente iluminada, sombreada e incluso texturizada.

Evaluadores

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

- Se utilizan evaluadores para describir cualquier polinomio o splines polinómicos racionales o superficies de cualquier grado, incluye las B-splines, las NURBS (Non-Uniform Rational B-Spline), curvas y superficies Bézier y splines Hermite.
- La función NURBS de la GLU es una interfaz de alto nivel: Los procesos NURBS encapsulan gran cantidad de código complejo. Gran parte del renderizado final se realiza con el evaluador, pero para ciertas condiciones (por ejemplo, recorte de curvas), los procesos NURBS utilizan polígonos planos para el renderizado.

Evaluadores

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

Una curva de Bézier es una función vectorial de una variable

$$C(u) = [X(u)\, Y(u)\, Z(u)]$$

Donde u varía en un dominio $[0, 1]$. Una superficie de Bézier es una función vectorial de dos variables.

$$S(u, v) = [X(u, v)\, Y(u, v)\, Z(u, v)]$$

Donde u y v pueden variar en algún dominio.

El rango puede tener una salida tridimensional, bidimensional para curvas en un plano o coordenadas de textura,o una salida cuatridimensional para especificar información RGBA.

Evaluadores

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

Para utilizar un evaluador, primero hay que definir la función **C()** o **S()**, habilitarla y luego utilizar el comando **glEvalCoord1()** o **glEvalCoord2()** en lugar de **glVertex*()**.

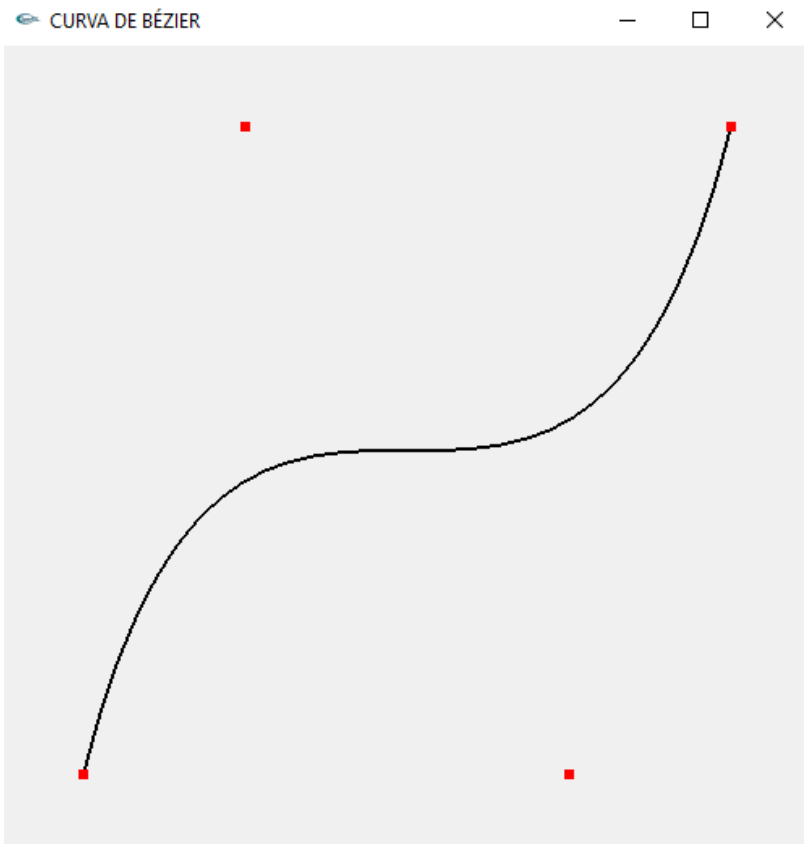
- Los vértices de la curva o de la superficie pueden utilizarse como cualquier otro vértice, para formar puntos o líneas. Además, otros comandos generan automáticamente series de vértices que producen una malla regular uniformemente espaciada en u (o en u y v).

Evaluadores unidimensionales

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

Ejemplo 1.

Dibuja una curva cúbica de Bézier usando cuatro puntos de control, como se muestra en la Figura 1.



Código

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

La curva cúbica de Bézier se describe mediante cuatro puntos de control.

```
GLfloat ctrlpoints[4][3] =
           {{-4.0, -4.0, 0.0}, { -2.0, 4.0, 0.0},
           {2.0, -4.0, 0.0}, {4.0, 4.0, 0.0}};
void init(void){
       glClearColor(0.94, 0.94, 0.94, 0.0);
       glShadeModel(GL_FLAT);
       glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 4,
                              &ctrlpoints[0][0]);
       glEnable(GL_MAP1_VERTEX_3);
 }
```

La matriz **ctrlpoints[][]** es uno de los argumentos de **glMap1f()** y **glEnable()** habilita el evaluador unidimensional para vértices tridimensionales.

Código

La curva se dibuja en la rutina **display()** entre las llamadas **glBegin()** y **glEnd()**.

```
glColor3f(0.0, 0.0, 0.0);
glBegin(GL_LINE_STRIP);
   for (int i = 0; i <= 30; i++)
       glEvalCoord1f((GLfloat) i/30.0);
glEnd();
glPointSize(6.0);
glColor3f(1.0, 0.0, 0.0);
glBegin(GL_POINTS);
    for (int i = 0; i < 4; i++)
       glVertex3fv(&ctrlpoints[i][0]);
glEnd();
```

El comando **glEvalCoord1f()** es como emitir un **glVertex()** con las coordenadas de un vértice de la curva correspondiente al parámetro de entrada u.

Definición y evaluación de un evaluador unidimensional

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

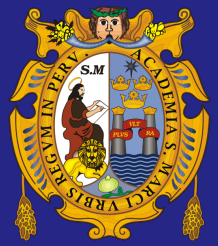
El polinomio de Bernstein de grado n (u orden $n+1$) es dado por:

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}$$

Si *Pi* representa un conjunto de puntos de control (unidimensionales, bidimensionales, tridimensionales o cuatridimensionales), entonces la ecuación:

$$C(u) = \sum_{i=0}^{n} B_i^n(u) P_i$$

Donde $B_i^n(u)$ son elementos de la distribución binomial respecto a u y los P_i son valores de la función que queremos aproximar.



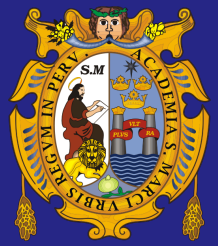
Definición y evaluación de un evaluador unidimensional

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

- Representa una curva de Bézier cuando u varía de 0 a 1.
- Para representar la misma curva, pero permitiendo que, u varíe entre $u1$ y $u2$ en lugar de 0 y 1, evaluamos:

$$C\left(\frac{u-u_1}{u_2-u_1}\right)$$

El comando **glMap1()** define un evaluador unidimensional que utiliza estas ecuaciones.



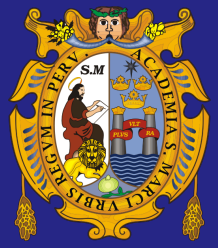
Definición y evaluación de un evaluador unidimensional

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

Comando

glMap1{fd} (GLenum target, TYPE u1, TYPE u2, GLint stride, GLint order, const TYPE *points);

- **GLenum target:** Especifica lo que representan los puntos de control, por lo tanto cuántos valores deben ser suministrados en puntos. Los puntos pueden representar vértices, datos de color RGBA, vectores normales o coordenadas de textura.
- **TYPE u1 y TYPE u2:** u1 y u2, indican el rango de la variable u.
- **GLint stride:** Es el número de valores de precisión simple o doble (según el caso) en cada bloque de almacenamiento. Por lo tanto, es un valor de desplazamiento entre el comienzo de un punto de control y el comienzo del siguiente.



Definición y evaluación de un evaluador unidimensional

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

Comando

glMap1{fd} (GLenum target, TYPE u1, TYPE u2, GLint stride, GLint order, const TYPE *points);

- **GLint order:** El orden es el grado más uno, y debe coincidir con el número de puntos de control.
- **const TYPE *points:** Los puntos apuntan a la primera coordenada del primer punto de control.

Utilizando la estructura de datos de ejemplo para glMap1*(), utilice lo siguiente para los puntos:

$$(GLfloat*)(\&ctrlpoints[0].x)$$

Definición y evaluación de un evaluador unidimensional

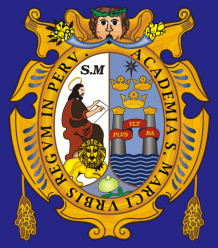
Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

Se utilizan los valores de los parámetros listados en la Tabla 1 para activar cada evaluador definido antes de invocarlo.

Cuadro: Tipos de puntos de control para **glMap1*()**.

Parámetro	Significado
GL_MAP1_VERTEX_3	coordenadas de los vértices x, y, z.
GL_MAP1_VERTEX_4	coordenadas de los vértices x, y, z, w.
GL_MAP1_INDEX	índice de color.
GL_MAP1_COLOR_4	R, G, B, A.
GL_MAP1_NORMAL	coordenadas normales.
GL_MAP1_TEXTURE_COORD_1	s coordenadas de textura.
GL_MAP1_TEXTURE_COORD_2	s, t coordenadas de textura.
GL_MAP1_TEXTURE_COORD_3	s, t, r coordenadas de textura
GL_MAP1_TEXTURE_COORD_4	s, t, r, q coordenadas de textura.

Pasamos el valor apropiado a **glEnable()** o **glDisable()** para activar o desactivar el evaluador.



Definición y evaluación de un evaluador unidimensional

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

- Se puede evaluar más de un evaluador a la vez. Si el evaluador GL_MAP1_VERTEX_3 y GL_MAP1_COLOR_4 son definidos y habilitados, entonces **glEvalCoord1()** generan una posición como un color.
- Si se define y activa más de un evaluador del mismo tipo, se utiliza el de mayor dimensión.
- Utilizamos **glEvalCoord1*()** para evaluar un mapa unidimensional definido y habilitado.

```
void glEvalCoord1{fd} (TYPE u);
void glEvalCoord1{fd} v (TYPE *u);
```

Provoca la evaluación de mapas unidimensionales habilitados. El argumento *u* es el valor (o un puntero al valor, en la versión vectorial del comando) de la coordenada del dominio.

Definición de valores de coordenadas uniformes en una dimensión

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

- Se puede utilizar **glEvalCoord1()** con cualquier valor para u, pero el uso más común es con valores espaciados uniformemente, como se muestra en el Ejemplo 1.
- Para obtener valores uniformemente espaciados, definimos una unidimensional usando **glMapGrid1*()** y aplícamos usando **glEvalMesh1()**.

```
void glMapGrid1{fd} (GLint n, TYPE u1, TYPE u2);
```

Define una cuadrícula que va de $u1$ a $u2$ en n pasos, que están espaciados uniformemente.

```
void glEvalMesh1(GLenum mode, GLint p1, GLint p2);
```

Definición de valores de coordenadas uniformes en una dimensión

Integrantes:
Balboa
Palma, Merly
Mirano
Mitchell
Vasquez
Sebastian

- Aplica la cuadrícula de mapa definida actualmente a todos los evaluadores habilitados. El modo puede ser **GL_POINT o GL_LINE**, dependiendo de si desea dibujar puntos o una línea conectada a lo largo de la curva.
- Tiene exactamente el mismo efecto que emitir una **glEvalCoord1()** para cada uno de los pasos, incluyendo $p1$ y $p2$, donde $0 \leq p1, p2 \leq n$. Es equivalente a lo siguiente:

```
glBegin(GL_POINTS); /* o glBegin(GL_LINE_STRIP);*/
    for (i = p1; i <= p2; i++)
        glEvalCoord1(u1 + i*(u2-u1)/n);
glEnd();
```

Excepto si **i = 0** o **i = n**, entonces se llama a **glEvalCoord1()** con exactamente $u1$ o $u2$ como parámetro.