Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Computação Gráfica
AD1 - 2° semestre de 2009.

1) Qual a característica da subárea chamada de Visão Computacional? (0.5 ponto).

Esta área toma como ponto de partida uma imagem e a partir dela procura chegar a modelos matemáticos, através de uma série de técnicas, tais como a segmentação. Possíveis aplicações da área são o reconhecimento de células em imagens microscópicas, reconhecimento de placas de carros, reconhecimento facial, etc.

2) Discuta sobre a eficiência versus realismo na computação gráfica. (0.5 ponto).

A eficiência tem como principal requisito a rapidez do cálculo, já o realismo preza pela precisão dos mesmos. Para conseguir efetuar algumas tarefas de forma rápida em muitos casos é necessário simplificar os cálculos ou realizar pré-computações em estruturas de dados. Estas tarefas podem acarretar na perda de características que são encontradas na computação gráfica clássica.

3) O que são os Game Engines? (1.0 ponto).

Os Game engines ou motores de jogos são softwares que já possuem incorporados tarefas típicas de um jogo ou um sistema de computação gráfica tempo real. Os mesmos implementam o chamado Game loop, onde se realiza a leitura do input, aplica-se o cálculo de simulação física e inteligência artificial e finalmente faz-se a renderização da cena.

4) Descreva um método para gerar uma aproximação poligonal de uma elipse. Considere a elipse representada através da equação paramétrica abaixo (1.0 ponto).

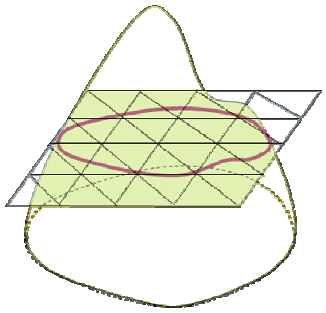
$$f(u) = \begin{cases} x(u) = 2\,sen(u) \\ y(u) = 3\,sen(u) \end{cases}$$
$$0 \le u \le 2\pi$$

```
void Display (void){
        const float PI = 3.1416f;
        float angle;
        int i;

        /* Limpa a tela(buffer de cores) e o buffer de
profundidades */
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        /* Desenha o pentagrama */

        glBegin(GL_LINE_LOOP);
        /* configura o angulo inicial em 0.0 graus */
        angle = 0.0f;
        for (i = 0 ; i<20 ; i++){
          glVertex2f(2*cos(angle), 3*sin(angle));
  angle += 2.0f*PI/20.0f;
        }
        glEnd();

        /* Troca os buffers */
        glutSwapBuffers ( );

}
```

5) Mostre como gerar uma curva poligonal através da descrição implícita de uma curva. Utilize figuras e gráficos para ilustrar o método (1.0 ponto).

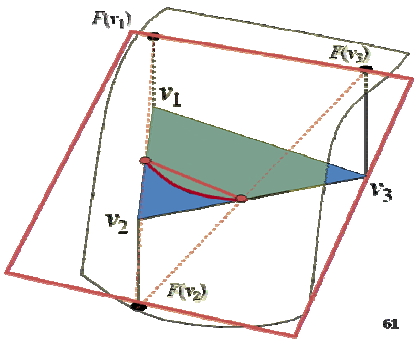
A dificuldade para poligonizar uma curva implícita se deve ao fato de que a solução de uma equação implícita determina um conjunto de pontos não-estruturados. Entretanto, é possível introduzir uma estruturação em tal conjunto através de uma triangulação do domínio da função. Com efeito, para gerar uma curva poligonal podemos aplicar o seguinte método:

1. *Determinar uma triangulação do domínio de F.*

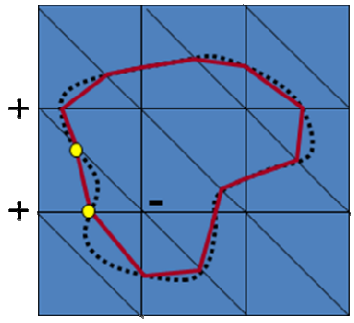


2. *Aproximar F em cada triângulo por uma função linear F'.*

3. *Solucionar F'(x,y)=0 em cada triângulo onde houver diferença de sinais entre os valores da função nos vértices. A solução é, em geral, um segmento de reta.*



4. *Cada segmento gerado é conectado aos segmentos nas células adjacentes, de acordo com a triangulação definida sobre o reticulado.*



6) O que são funções de mistura (blending functions)? Caracterize as funções de mistura para curvas de Bézier de grau três (1.0 ponto).

As funções de mistura formam uma base de funções, utilizadas na determinação dos pontos de uma curva interativa, com base em um conjunto de pontos de controle.

Intuitivamente, as funções de mistura determinam pesos que especificam como as coordenadas dos pontos de controle são combinadas para gerar a coordenadas de um ponto da curva, associada a um ou mais valores no espaço de parâmetros.

Seja $B_i(u)$ uma função de mistura associada a um ponto de controle p_i. As coordenadas de um ponto da curva, associado a um valor u no espaço de parâmetros são dadas por $f(u) = \sum_{i=1}^{n} p_i B_i(u)$. As características da curva gerada dependem das propriedades de continuidade e diferenciabilidade das funções de mistura.

As curvas de Bézier de grau 3 são definidas em termos das funções de mistura de Bernstein, também de grau 3, as quais têm a seguinte forma:

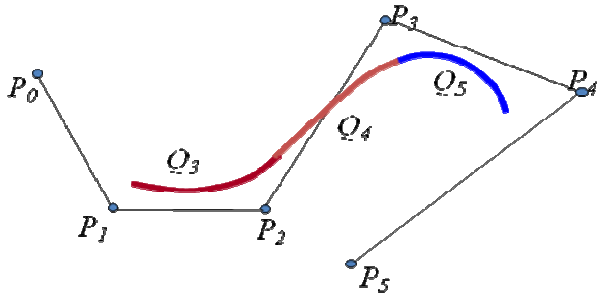
$$B_{i,3}(u) = \binom{3}{i} u^i (1-u)^{3-i}$$

$$B_{0,3} = (1-u)^3$$
$$B_{1,3} = 3u(1-u)^2$$
$$B_{2,3} = 3u^2(1-u)$$
$$B_{3,3} = u^3$$

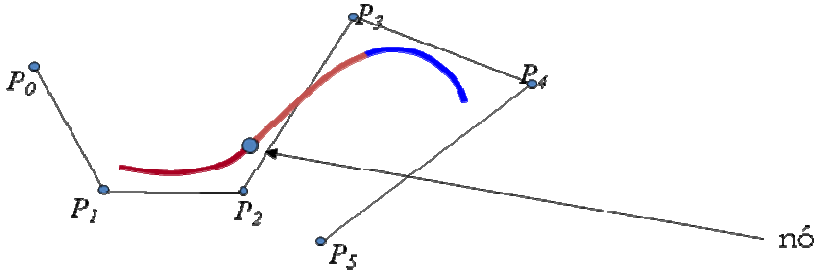
7) Descreva dois esquemas para fazer com que uma curva B-Spline interpole pontos de controle. Cite as diferenças entre os dois esquemas (1.0 ponto).

Uma B-Spline é uma curva completa polinomial por partes consistindo de um conjunto de m-2 segmentos de curva $Q_3,Q_4,\ldots,Q_m$ definidos por m+1 pontos de controle $P_0,P_1,\ldots,P_m$, $m \geq 3$.

Cada segmento de curva é definido por quatro pontos de controle e quatro funções de mistura. Cada ponto de controle influencia somente quatro segmentos de curva.

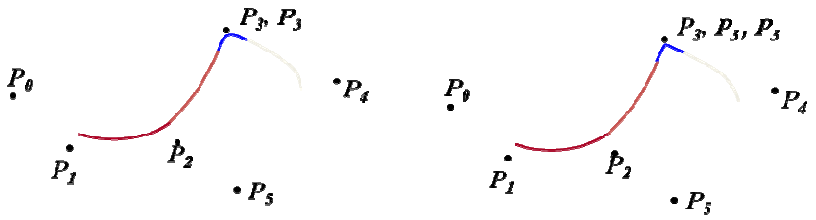


Os pontos associados a junções de dois segmentos adjacentes são denominados nós. O valor de u correspondente a um nó é denominado valor do nó. Uma B-spline é dita uniforme se valores dos nós são igualmente espaçados no espaço do parâmetro u.



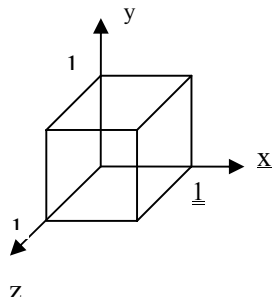
Em geral, B-Splines não interpolam seus pontos de controle. Entretanto, é possível alcançar tal efeito de duas formas:

- *Replicando pontos de controle, o que pode causar perdas de continuidade na curva*



- *Aumentando a multiplicidade dos nós, o que não causa perda de continuidade, mas que gera curvas B-Spline não uniformes, já que o intervalo no espaço de parâmetros entre os valores de nós não é mais o mesmo.*

8) Determine, com detalhes, as tabelas (listas) de faces, arestas e vértices de um cubo unitário (1.0 ponto).



Solução:

Lista de vértices:

Vértices	*Coordenadas*
v_1	*(0,0,0)*
v_2	*(1,0,0)*
v_3	*(1,0,1)*
v_4	*(0,0,1)*
v_5	*(0,1,0)*
v_6	*(1,1,0)*
v_7	*(1,1,1)*
v_8	*(0,1,1)*

Lista de arestas

Arestas	*Vértices*
e_1	$\{v_1,v_2\}$
e_2	$\{v_2,v_3\}$
e_3	$\{v_3,v_4\}$
e_4	$\{v_4,v_1\}$
e_5	$\{v_5,v_6\}$
e_6	$\{v_6,v_7\}$
e_7	$\{v_7,v_8\}$
e_8	$\{v_8,v_5\}$
e_9	$\{v_1,v_5\}$
e_{10}	$\{v_2,v_6\}$
e_{11}	$\{v_3,v_7\}$
e_{12}	$\{v_4,v_8\}$

Lista de faces

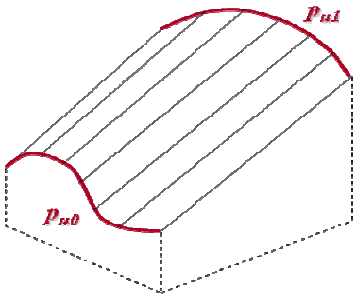
Faces	*Vértices*
f1	$\{e_3,e_{11},e_7,e_{12}\}$
f2	$\{e_4,e_{12},e_8,e_9\}$
f3	$\{e_1,e_9,e_5,e_{10}\}$
f4	$\{e_2,e_{10},e_6,e_{11}\}$
f5	$\{e_1,e_2,e_3,e_4\}$
f6	$\{e_7,e_6,e_5,e_8\}$

9) Explique o método de Coons para reconstrução de um retalho de superfície a partir de 4 curvas paramétricas(1.0 ponto).

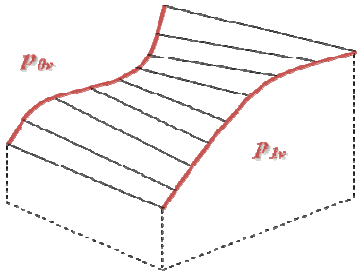
O método de parametrização de Coons consiste em combinar diversas interpolações lineares das curvas de bordo segundo o seguinte esquema:

- *Lofting vertical – interpolamos linearmente as curvas* p_{u0} *e* p_{u1}.
  $(1-v)p_{u0}(u)+vp_{u1}(u)$

- *Lofting horizontal – interpolamos linearmente as curvas p0v e p1v.*
  $(1-u)p_{0v}(v)+up_{1v}(v)$



Lofting vertical

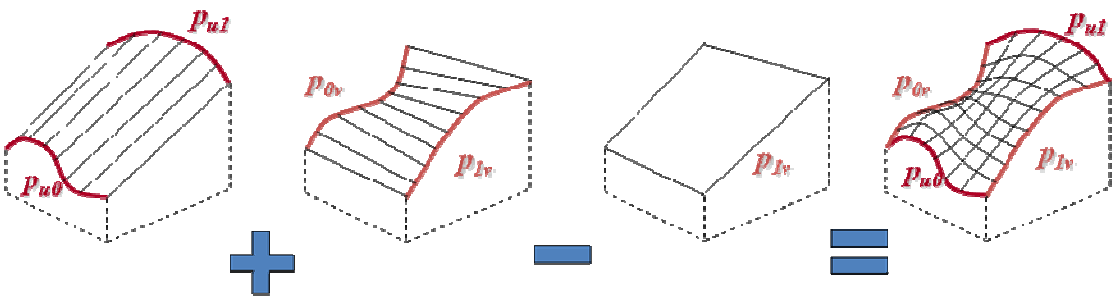


Lofting horizontal

- *Soma dos dois loftings – somamos as operações de lofting horizontal e vertical obtendo a parametrização*
  $C'(u,v)=(1-v)p_{u0}(u)+vp_{u1}(u)+(1-u)p_{0v}(v)+up_{1v}(v)$

Como resultado, temos que os bordos do retalho são dados pela soma de cada uma das curvas originais com a interpolação linear dos respectivos vértices

Para resolver tal problema, preservando as curvas originais, subtraímos o retalho obtido C'(u,v) da interpolação bilinear dos 4 vértices que são os pontos extremos das curvas de definição: $C(u,v) = C'(u,v)-B(u,v)$.



10) Escreva uma função, utilizando OpenGL, que desenhe um pentagrama na tela. GLUT (1.0 ponto).

```
void Display (void)
{
  const float PI = 3.1416f;
  float angle, size = 50.0;
  int i;

  /* Limpa a tela(buffer de cores) e o buffer de
profundidades */
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  /* Desenha o pentagrama */

  glBegin(GL_LINE_LOOP);
  /* configura o angulo inicial em 90 graus */
  angle = PI/2.0;
  for (i = 0 ; i<=4 ; i++){
    glVertex2f(size*cos(angle), size*sin(angle));
   /* incrementa o angulo em 144 graus */
   angle += 4.0*PI/5.0;
  }
     glEnd();

     /* Troca os buffers */
         glutSwapBuffers ( );
}
```