

OPENGL - Splines

Profa. Dra. Regina Célia Coelho

rccoelho@unifesp.br



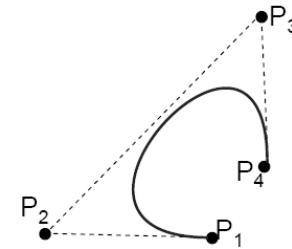
Definição de splines Bézier

- OpenGL cria *splines* baseada em bases de Bernstein utilizando *evaluators*.

- Uma curva Bézier é um vetor do tipo:

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

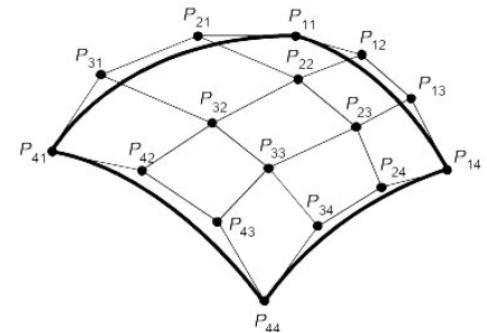
sendo que u varia em algum domínio (entre $[0, 1]$ para Bézier)



- Uma superfície utiliza duas variáveis:

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

- Para cada valor de u (ou u e v) a fórmula calcula pontos na curva (superfície).



Evaluators

- Permite a especificação de um curva ou superfície usando pontos de controle.
- Os pontos gerados pelo *evaluator* podem ser usados para desenhar:
 - ✓ pontos da superfície;
 - ✓ uma versão *wireframe* da superfície; ou
 - ✓ uma versão sombreada e iluminada da superfície.
- *Evaluators* podem ser usados para descrever *splines* racionais ou não e superfícies de qualquer grau.
- Isso inclui quase todos os tipos de *splines*, incluindo *B-Splines*, *NURBS* e *Bézier*.

Usando *evaluator*

- Para usar o *evaluator* :
 - ✓ definimos a função (curva ou superfície);
 - ✓ habilitamos esta função;
 - ✓ utilizamos o comando `glEvalCoord1()` ou `glEvalCoord2()` ao invés do `glVertex()`.
- Desta forma, os vértices da curva ou superfície podem ser utilizados como qualquer outro vértice (por exemplo, plotar pontos ou linhas).

Função glEvalCoord1()

```
glBegin(GL_LINE_STRIP);
    for (u = 0; u <= ufim; u++)
        glEvalCoord1f(GLfloat) u/ufim);
glEnd();
```

Neste exemplo, tomando valores $u/ufim$ no argumento, estamos pegando $ufim$ valores de u entre 0 e 1. Isso controla quantos pontos a curva final terá (partindo do primeiro ponto de controle até o último). Neste exemplo a curva terá $ufim+1$ pontos de controle (a quantidade de pontos vai de 0 até $ufim$).

O comando *glEvalCoord1f* avalia os pontos de um mapa unidimensional definido e habilitado anteriormente pela função *glMap* com os valores de u do parâmetro da chamada da função *glEvalCoord1f* na função que define uma curva Bézier. É como se chamássemos o *glVertex* para cada ponto e todos os pontos fossem combinados na função para cada valor de u .

Função *glMap1*

- Define um *evaluator* unidimensional que usa equações do polinômio de *Bernstein*, porém permitindo que u possa variar não somente entre 0 e 1, mas entre qualquer intervalo $[u_1, u_2]$.

glMap1{fd}(GLenum target, TYPEu1, TYPEu2, GLint stride, GLint order, const TYPE *points);

target: especifica como os pontos de controle serão apresentados (por exemplo, `GL_MAP1_VERTEX_3` indica representação (x, y, z) , `GL_MAP1_VERTEX_4` indica representação (x, y, z, w)) ;

u1, u2: limite de variação de u ;

stride: quantidade de valores no vetor entre o início de um ponto de controle e o início do próximo; no caso tridimensional, cada ponto possui 3 coordenadas, portanto, $stride = 3$ (3 valores entre o início de um ponto e o início do próximo)

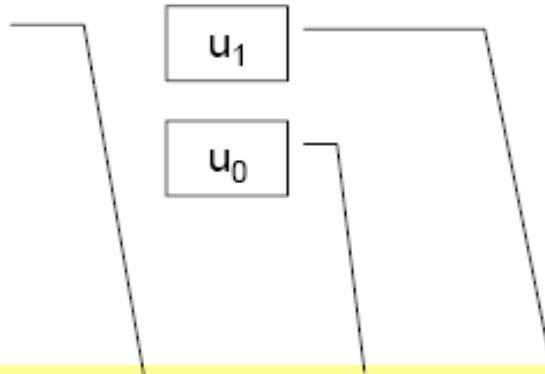
order: ordem da *spline* ($grau = order - 1$);

points: ponteiro para o primeiro ponto de controle do vetor de pontos.

Função *glMap1**

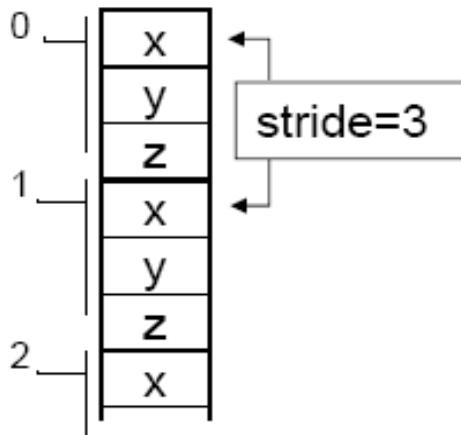
Parâmetros de *glMap1**

GL_MAP1_VERTEX_3
 GL_MAP1_VERTEX_4
 GL_MAP1_INDEX
 GL_MAP1_COLOR_4
 GL_MAP1_NORMAL
 GL_MAP1_TEXTURE_COORD_1
 GL_MAP1_TEXTURE_COORD_2
 GL_MAP1_TEXTURE_COORD_3
 GL_MAP1_TEXTURE_COORD_4



Normalmente:
 $[u_0, u_1] = [0, 1]$

```
void glMap1{fd} (GLenum target, TYPEu1, TYPEu2,
                  GLint stride, GLint order, const TYPE*points)
```



1 + Grau da curva

Array com pontos
de controle

x	y	z
x	y	z
x	y	z
x	y	z

```
glMap1{fd} (GL_MAP1_VERTEX_3, 0, 1,
            3, 4, &ctrlpoints[0][0])
```

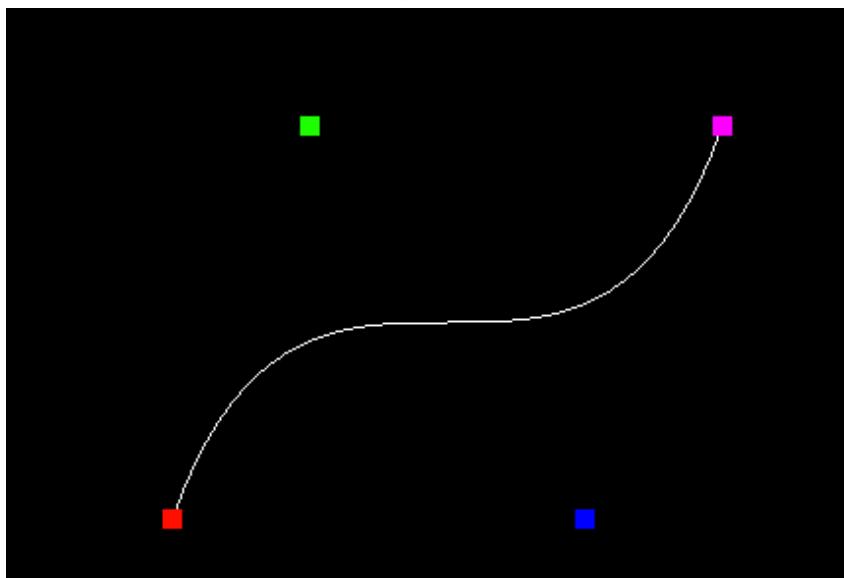
Exercício

Rode o programa exemplo que gera uma curva *spline* de Bézier.

A sequência para a montagem da *spline* é:

- ✓ defina um *glMap1*;
- ✓ habilite o uso deste mapa. Para isso faça *glEnable(target)*, sendo que a constante **target** é a mesma definida no *glMap1*;
- ✓ utilize a função *glEvalCoord1* para definir os pontos da *spline*

Plote também os pontos de controle que estão definidos.

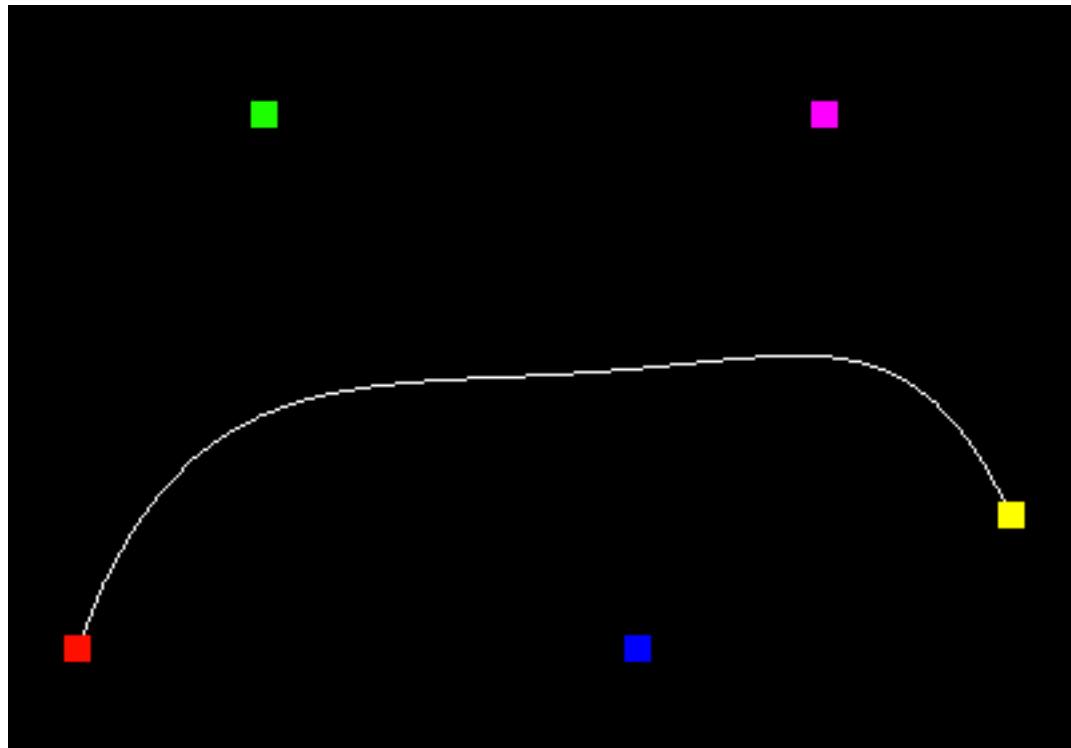


Exercício

- Para um exemplo de 6 pontos, quanto deverá ser a ordem definida na função *glMap*?
- O que acontece se alterarmos essa ordem? Teste no exemplo anterior colocando a ordem maior e menor que 4.
- Altere a quantidade de pontos que *glEvalCoord1f* irá calcular (por exemplo, inclua mais pontos) e verifique o resultado. Explique o que alterou e por que.

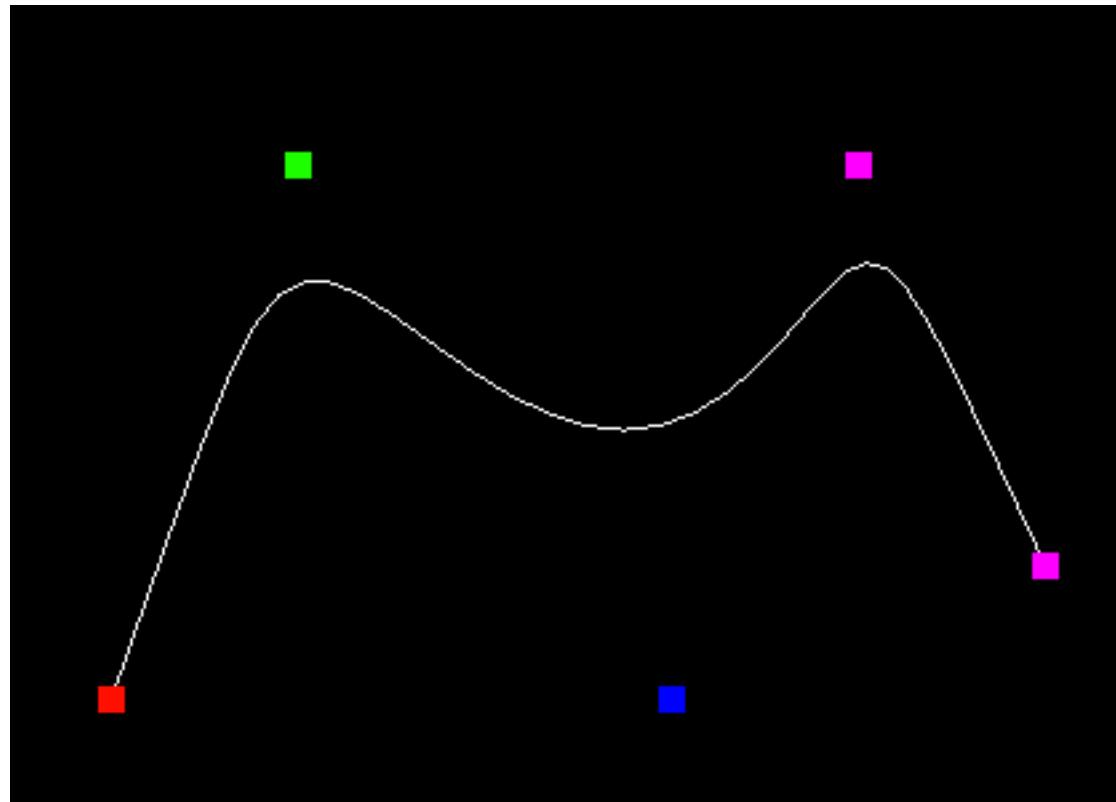
Exercício

- Acrescente o ponto $(6.0, -2.0, 0.0)$.



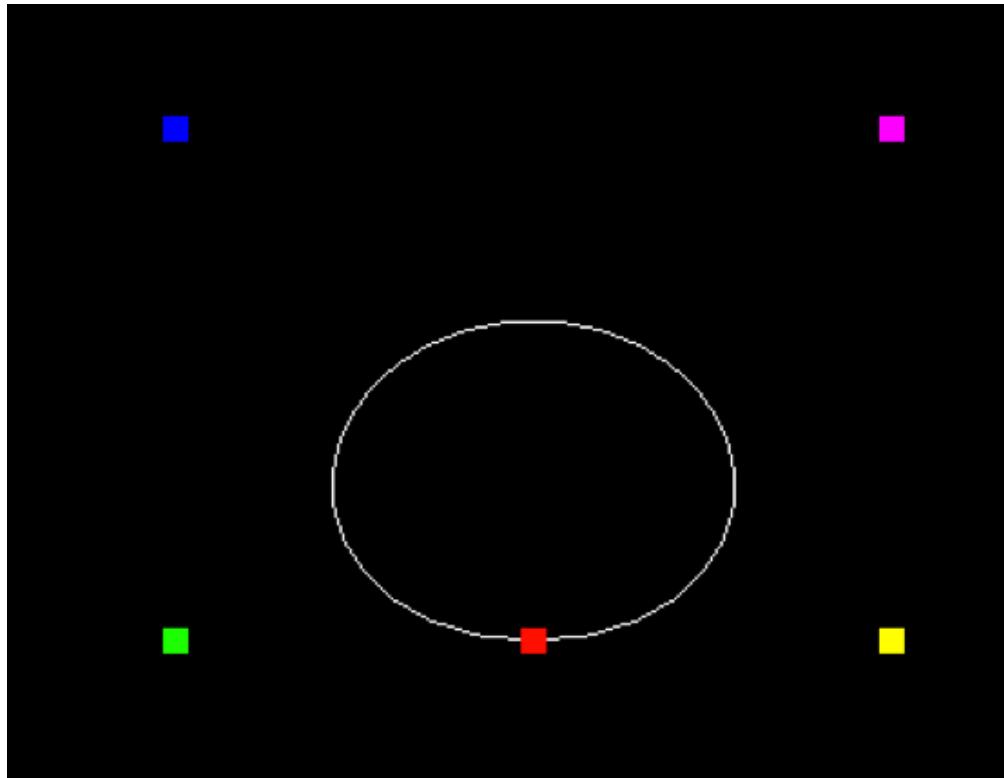
Exercício

- Atribua pesos para os pontos intermediários de forma a fazer a curva resultante adquirir o seguinte formato:



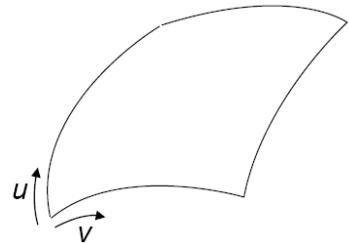
Exercício

➤ Altere seu programa para que, utilizando projeção ortogonal, você tenha o seguinte resultado (você terá que definir novos pontos de controle):



Evaluators bidimensionais

- Duas variáveis de controle: (u, v)



```
void glMap2{fd} (GLenum target,  
                  TYPEu1, TYPEu2, GLint ustride, GLint uorder,  
                  TYPEv1, TYPEv2, GLint vstride, GLint vorder,  
                  TYPE points);
```

```
void glEvalCoord2{fd} (TYPE u, TYPE v);
```

Exercício

- Defina uma superfície *spline* com os seguintes pontos de controle (use projeção ortogonal):

```
GLfloat ctrlpoints[4][4][3] = {  
    {{-1.5, -1.5, 4.0}, {-0.5, -1.5, 2.0}, {0.5, -1.5, -1.0}, {1.5, -1.5, 2.0}},  
    {{-1.5, -0.5, 1.0}, {-0.5, -0.5, 3.0}, {0.5, -0.5, 0.0}, {1.5, -0.5, -1.0}},  
    {{-1.5, 0.5, 4.0}, {-0.5, 0.5, 0.0}, {0.5, 0.5, 3.0}, {1.5, 0.5, 4.0}},  
    {{-1.5, 1.5, -2.0}, {-0.5, 1.5, -2.0}, {0.5, 1.5, 0.0}, {1.5, 1.5, -1.0}}};
```

- Plote os pontos de controle.

Exercício

- Inclua teclas para controlar rotações nos 3 eixos de coordenadas (uma tecla para cada eixo).
- Altere a quantidade de pontos que *glEvalCoord2f* irá calcular e verifique o resultado.

Outra forma de Definir Splines

- Os laços no exemplo anterior podem ser substituídos por duas outras funções:
 - ✓ `glMapGrid2f(un, u1, u2, vn, v1, v2)`: define uma grade bidimensional.
 - un = quantidade de valores que teremos entre $u1$ e $u2$;
 - $(u1, u2)$ = intervalo de variação de u ;
 - vn = quantidade de valores que teremos entre $v1$ e $v2$;
 - $(v1, v2)$ = intervalo de variação de v .

Outra forma de Definir Splines

- ✓ `glEvalMesh2(GL_LINE, p1, p2, q1, q2)`: calcula uma grade de pontos bidimensionais, ou seja, aplica a grade definida anteriormente (pelo comando `glMapGrid2f`) ao *evaluator* definido (por `glMap`) e habilitado.
 - `GL_LINE` pode ser substituído por `GL_FILL` ou `GL_POINT`;
 - $(p1, p2)$ = especifica o primeiro e o último valor inteiro para a variável i do domínio da grade (normalmente $p1=0$ e $p2$ = quantidade de valores de u);
 - $(q1, q2)$ = mesma observação de $(p1, p2)$, porém referente a j (ou v).