Exercício IV (2012/I)

Disciplina: Computação Gráfica

Professora: Soraia R. Musse

Aula prática – Curvas e Superfícies

O **objetivo desta aula** é trabalhar com os diferentes métodos para gerar curvas usando a biblioteca OpenGL.
Você pode obter os códigos fontes que serão utilizados na aula de hoje na página da disciplina.
Este programa possui implementações do exercício anterior.

Inicialmente, verifique o conteúdo do programa fonte fornecido. Compile e execute o programa e verifique que aparecerá uma janela com fundo preto, com um grid representando o chão e 3 diferentes curvas.

As matrizes ***float BezierPoints[4][3]***, ***float HermitePoints[4][3]***, ***float BSplinePoints[4][3]***, armazenam os pontos de controle e as tangentes (quando for o caso) de cada curva. O método ***desenhaChao()*** é o responsável por desenhar o grid.

A função ***desenhaPontosDeControle()*** é a função que desenha os caracteres vermelhos na janela, indicando onde estão situados cada ponto de controle/tangente e os identificando. O método ***glRasterPos3d(GLdouble x, GLdouble y, GLdouble z)*** serve para indicar em qual posição da janela se deseja desenhar. O laço seguinte passa por todos os caracteres da string desejada e desenha cada um com o método ***glutBitmapCharacter(void*font, int character)***, onde o primeiro parâmetro é um apontador para uma constante que define a fonte. A função ***desenhaNomes()*** desenha os nomes das curvas com o mesmo princípio.

As funções ***desenhaBezier(), desenhaBSpline()*** e ***desenhaHermite()***, são as funções de desenhos das curvas. Estas funções aplicam as matrizes de representação das curvas, como comentado na aula passada. Por exemplo, na função de desenho da curva Hermite a matriz é:

$$P(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P(0) \\ P(1) \\ P'(0) \\ P'(1) \end{bmatrix}$$

Na função desenhaHermite() ela é aplicada no trecho de código:

```
float b0 =  2*t*t*t - 3*t*t + 1;
float b1 = -2*t*t*t + 3*t*t;
float b2 = t*t*t - 2*t*t + t;
float b3 = t*t*t - t*t;
```

Após, para calcular as posições x, y, z do novo ponto da curva, cada ponto da estrutura HermitePoints[][] é multiplicado por um desses valores b0, b1, b2 e b3.

Note que em cada uma das funções é executado um laço de 0 a 15. Este 15 é o nível de detalhe das curvas (Level Of Detail), ou seja, quantos pontos existirão nas curvas.

Altere o código, substituindo os números 15 por uma variável de nome LOD. Declare essa variável como ***unsigned int*** no início do código e atribua o valor inicial 15. Após, adicione o código abaixo, que permite modificar essa variável LOD no teclado com as teclas '+' e '-':

```
void GerenciaTeclado(unsigned char key,int,int) {
	if (key == 27)
		exit(0);

	switch(key)
	{

		case '+':
			++LOD;
			break;

		case '-':
			--LOD;

			if (LOD<2)
				LOD=2;
			break;

		default:
			break;
	}

	glutPostRedisplay();
}
```

Note que a condição if foi adicionada para evitar que fique poucos pontos, destruindo a curva. Logo, a curva necessita de pelo menos dois pontos.

Inclua na função main: ***glutKeyboardFunc(GerenciaTeclado);*** para ativar a função do teclado, e teste para ver o resultado.

Agora vamos fazer algumas alterações para permitir a manipulação dos pontos de controle/tangentes das curvas, o que irá permitir visualizar melhor os resultados e "brincar" com a curva. O primeiro passo é adicionar no início do código as seguintes variáveis:

```
bool bezier = false, hermite = false, bspline = false;
bool p1 = false, p2 = false, p3 = false, p4 = false;
```

Essas variáveis servirão para controlarmos qual curva e qual ponto de controle desejamos alterar.

Tendo feito isso, na função GerenciaTeclado, adicione esse trecho de código:

```
case '1':
	p1 = true; p2 = false; p3 = false; p4 = false;
	printf("\nP1 selected.");
	break;
case '2':
	p1 = false; p2 = true; p3 = false; p4 = false;
	printf("\nP2 selected.");
	break;
case '3':
	p1 = false; p2 = false; p3 = true; p4 = false;
	printf("\nP3 selected.");
	break;
case '4':
	p1 = false; p2 = false; p3 = false; p4 = true;
	printf("\nP4 selected.");
	break;
```

Este código irá permitir a seleção do ponto de controle desejado através dos números 1, 2, 3 e 4 do teclado (No caso da Hermite, o 3 e o 4 selecionam as tangentes). Após crie uma função GerenciaTecladoEspecial, que irá gerenciar teclas especiais, de acordo com o código abaixo:

```
void GerenciaTecladoEspecial(int key, int x, int y)
{
	switch (key)
	{
	case GLUT_KEY_F1:
		bezier = false; hermite = false;	bspline = true;
		printf("\nBSpline active.");
		break;
	case GLUT_KEY_F2:
		bezier = false;	hermite = true;	bspline = false;
		printf("\nHermite active.");
		break;
	case GLUT_KEY_F3:
		bezier = true;	hermite = false; bspline = false;
		printf("\nBezier active.");
		break;
	}

	glutPostRedisplay();
}
```

Com este código, as teclas F1, F2 e F3 se tornam as teclas de seleção da curva que será alterada: F1 para BSpline, F2 para Hermite e F3 para Bezier.
Agora vamos adicionar um trecho de código (dentro desta mesma função) para alterar as posições dos pontos de controle/tangentes que serão selecionados. A tecla **GLUT_KEY_LEFT**,

irá mover o ponto selecionado para a esquerda, e a tecla **GLUT_KEY_RIGHT** irá mover o mesmo para a direita:

```
case GLUT_KEY_LEFT:
        if (bezier)
        {
                if (p1) BezierPoints[0][0] -= 0.2;
                if (p2) BezierPoints[1][0] -= 0.2;
                if (p3) BezierPoints[2][0] -= 0.2;
                if (p4) BezierPoints[3][0] -= 0.2;
        }
        if (hermite)
        {
                if (p1) HermitePoints[0][0] -= 0.2;
                if (p2) HermitePoints[1][0] -= 0.2;
                if (p3) HermitePoints[2][0] -= 0.2;
                if (p4) HermitePoints[3][0] -= 0.2;
        }
        if (bspline)
        {
                if (p1) BSplinePoints[0][0] -= 0.2;
                if (p2) BSplinePoints[1][0] -= 0.2;
                if (p3) BSplinePoints[2][0] -= 0.2;
                if (p4) BSplinePoints[3][0] -= 0.2;
        }
        break;

case GLUT_KEY_RIGHT:
        if (bezier)
        {
                if (p1) BezierPoints[0][0] += 0.2;
                if (p2) BezierPoints[1][0] += 0.2;
                if (p3) BezierPoints[2][0] += 0.2;
                if (p4) BezierPoints[3][0] += 0.2;
                }
        if (hermite)
        {
                if (p1) HermitePoints[0][0] += 0.2;
                if (p2) HermitePoints[1][0] += 0.2;
                if (p3) HermitePoints[2][0] += 0.2;
                if (p4) HermitePoints[3][0] += 0.2;
        }
        if (bspline)
        {
                if (p1) BSplinePoints[0][0] += 0.2;
                if (p2) BSplinePoints[1][0] += 0.2;
                if (p3) BSplinePoints[2][0] += 0.2;
                if (p4) BSplinePoints[3][0] += 0.2;
        }
        break;
```

Não se esqueça de adicionar também na função main o método **glutSpecialFunc(GerenciaTecladoEspecial)** para ativar a função do teclado.

Para manipular o ponto P1 da curva de Bezier, por exemplo, pressione F3 para selecionar a curva de Bezier, e em seguida pressione 1 para selecionar o ponto P1.
Execute o programa e teste para ver os resultados.

Agora faça a mesma coisa para as teclas **GLUT_KEY_UP** e **GLUT_KEY_DOWN**, para que estas movam o ponto para cima e para baixo (incrementando e decrementando a coordenada y). Também adicione as teclas **GLUT_KEY_PAGE_UP** e **GLUT_KEY_PAGE_DOWN,** de forma com que elas incrementem e decrementem a coordenada z do ponto.

Exercício: Tente alterar o código para criar outro segmento de curva (escolha a curva de sua preferência: Bezier, Hermite ou BSpline) e conectá-la ao outro segmento correspondente no grau de conectividade C0 (somente unir o ponto final da primeira com o ponto inicial da segunda).