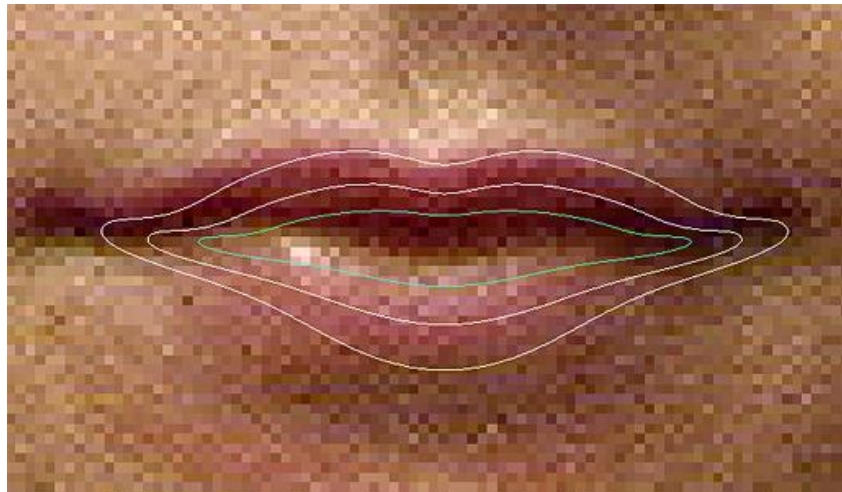


# Tópicos

- Curvas
- Superfícies
- Técnicas principais de Modelagem Geométrica

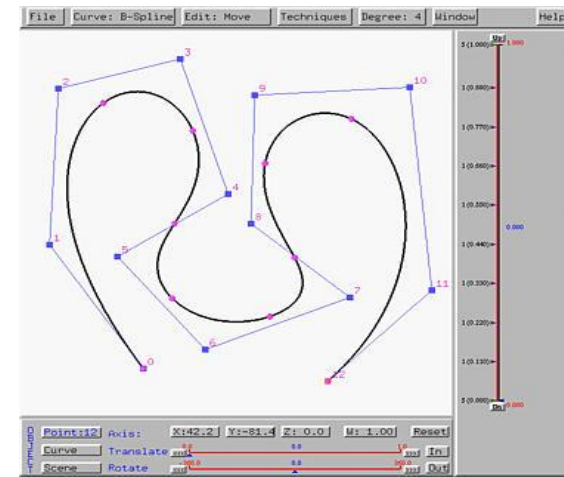
# Curvas e Superfícies - Introdução

- Curvas são a base, tanto da geração de formas simples, como círculos e elipses, quanto na criação de projetos complexos como automóveis, navios, aeronaves ou até mesmo faces e corpos humanos.



# Curvas e Superfícies - Introdução

- Desempenham um papel importante em várias áreas, como criação de objetos e visualização de fenômenos científicos.
- Uma curva pode ser representada:
  - Por uma sucessão de linhas retas que ligam pontos específicos.
  - Por um conjunto de pontos de controle que determina através de uma equação uma curva que passe por eles.



# Representação de Curvas

- A representação analítica de curvas pode usar ou não parâmetros, se subdividindo então em:
  - B.1. paramétricas
  - B.2. não-paramétricas
- As formas não-paramétricas podem ser, ainda,
  - B.2.1. explícitas
  - B.2.2. implícitas.

# Representação de Curvas

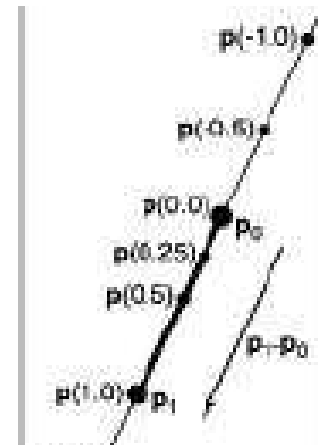
## B.1 Formas paramétricas

- As coordenadas são dadas em termos de um ou um conjunto de parâmetros ( $t$ ,  $\theta$ , etc.).

Exemplo de curvas no plano:  $P(t) = (x(t), y(t))$

### ➤ Segmentos

$$p(t) = p_0 + t(p_1 - p_0)$$



# Representação de Curvas

## B.1 Formas paramétricas

Exemplos:

$$P(t) = (x(t), y(t))$$

Circuferências

$$p(t) = (r \cos t, r \sin t)$$

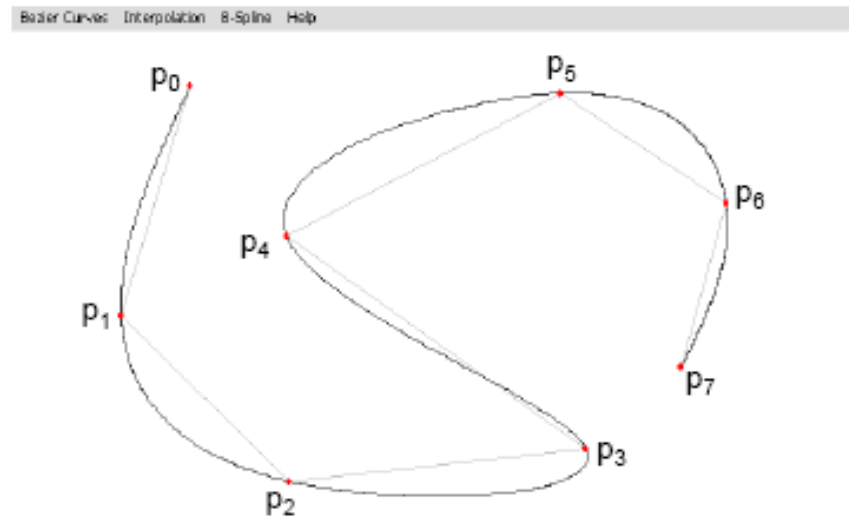


# Representação de Curvas

## B.1. Formas paramétricas

Exemplos:

$P(t) = (x(t), y(t))$   
**Curvas de Bézier**



$$P(t) = \sum B_{n,i}(t) p_i \text{ onde } B_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

Combinação convexa de  $p_i$

# Representação de Curvas

## B.2. Formas não-paramétricas

Não há parâmetros e *uma coordenada* da curva é dada em função da outra, ou seja

$$y = f(x) \text{ ou } x = f(y)$$

–Exemplos:

1) Equação de um semi-círculo de raio 2

$$y = \sqrt{2^2 - x^2} \quad \text{ou} \quad x = \sqrt{2^2 - y^2}$$

2) Equação de uma reta

$$y = 2x - 1 \text{ ou } x = \frac{1}{2}(y + 1)$$



# Representação de Curvas

**B.2.1 Forma *não-paramétrica explícita*** : É dada por uma equação do tipo  $y = f(x)$ , ou seja uma das coordenadas é explicitamente dada em função das outras. Exemplos:

1) Equação genérica explícita de uma parábola:

$$y = ax^2 + bx + c$$

2) Equação de uma reta

$$y = mx + b$$

3) Polinômios:

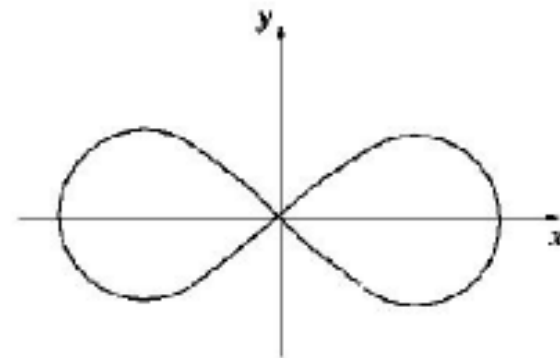
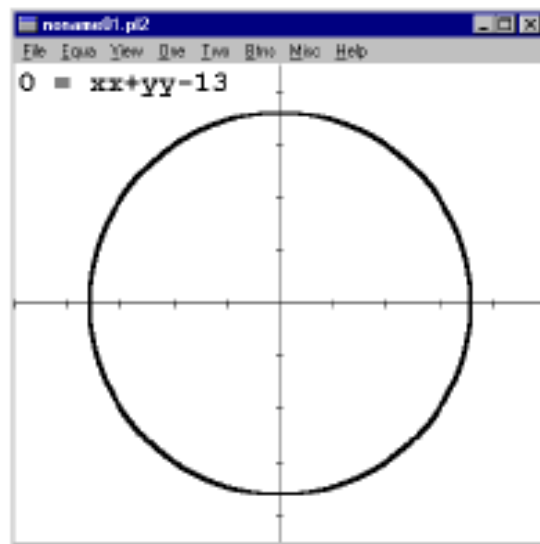
$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x^1 + a_0$$

# Representação de Curvas

- Obtém-se um valor de  $y$  para cada valor de  $x$  dado.

$$y = f(x) \text{ ou } z = f(x, y)$$

Não permite representar correspondências  
sobrejetivas!



Lemniscate:  $(x^2 + y^2)^2 - (x^2 - y^2) = 0$

## Representação de Curvas

- B.2.2. A Forma *não-paramétrica implícita* não tem essa limitação. Nela as coordenadas são relacionadas por uma função. A sua forma é  $f(x,y)=0$  ou  $f(x,y,z)=0$ .
- Exemplo:  $x^2 + y^2 = R^2$ ,  $f(x,y)= x^2 + y^2 - R^2= 0$

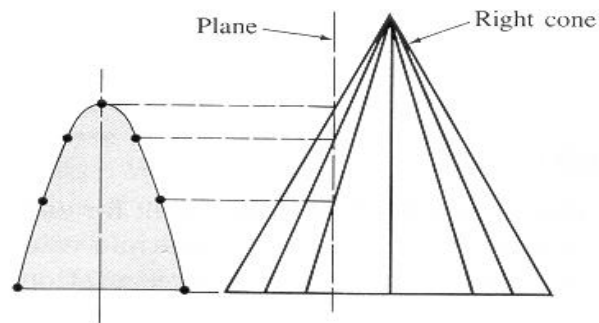
# Representação de Curvas

- Exemplo: seções cônicas.

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

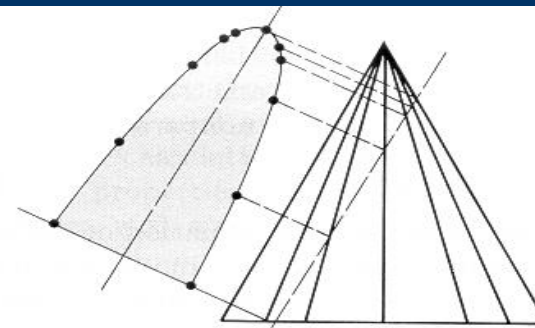
- Essa expressão representa a variedade de curvas planas denominadas seções cônicas. Essas curvas (cinco) são obtidas pelo corte de um cone por um plano, resultando em: círculo, elipse, parábola, hipérbole, reta.

# Representação de Curvas



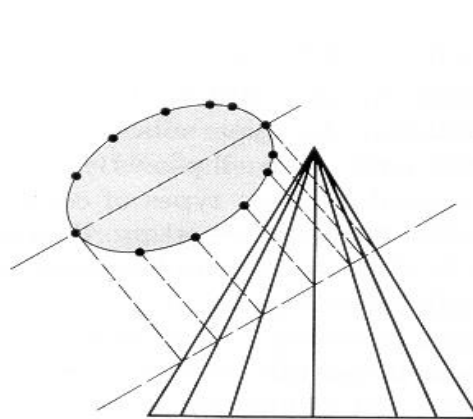
Hyperbola

(a)



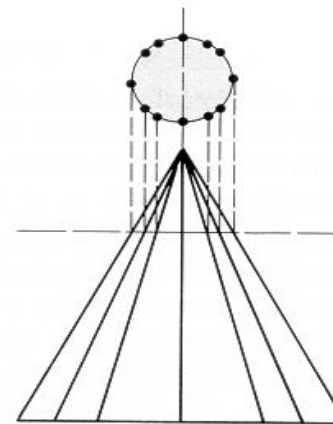
Parabola

(b)



Ellipse

(c)



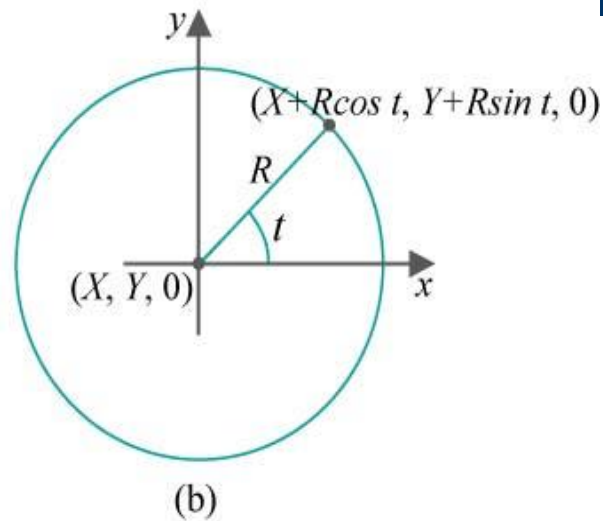
Circle

(d)

# Representação de Curvas

Cônica	Forma Paramétrica	Forma Implícita
Elipse	$x = a \cos \theta$ $y = b \sen \theta$	$\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0$
Parábola	$x = at^2, y = 2at$	$y^2 - 4ax = 0$
Hipérbole	$x = a \cosh \theta$ $y = b \sinh \theta$	$\frac{x^2}{a^2} - \frac{y^2}{b^2} - 1 = 0$

# Aproximando Objetos Uni-dimensionais



A equação paramétrica do círculo implementado é:

$$x = X + R \cos t, \quad y = Y + R \sin t, \quad z = 0, \quad 0 \leq t \leq 2\pi$$

Onde  $(X, Y, 0)$  é o centro e  $R$  é o raio do círculo.

Programa: circle.cpp (Capítulo 2)

# Aproximando Objetos Uni-dimensionais

- Explicando a implementação da equação do círculo
- Suponha  $N=4$   $t=2*PI*i/N$
- Para  $i=0 \Rightarrow t=2*PI*0/4 = 0 = 0$
- Para  $i=1 \Rightarrow t=2*PI*1/4 = PI/2 = 90$
- Para  $i=2 \Rightarrow t=2*PI*2/4 = PI = 180$
- Para  $i=3 \Rightarrow t=2*PI*3/4 = 3*PI/2 = 270$



# Curvas com OpenGL

Exercícios: Desenhe uma curva senoidal entre  $x=-2\pi$  e  $x=2\pi$ . Siga a estratégia do programa `circle.cpp`.

Equação paramétrica  $(x,y)=(t,\sin t)$

Equação explícita  $y=\sin x$ ;

Resolva a lista de exercícios no site da disciplina.

# Curvas de Bézier

- É uma técnica de aproximação de curvas.
- Uma curva de Bézier pode ser gerada por 3, 4, até  $n + 1$  pontos de controle (ajuste para um polinômio de grau  $n$ ).
- Geralmente utiliza-se quatro pontos de controle (*forma cúbica*).
- A curva passa pelo primeiro e pelo último ponto de controle.

# Curvas de Bézier

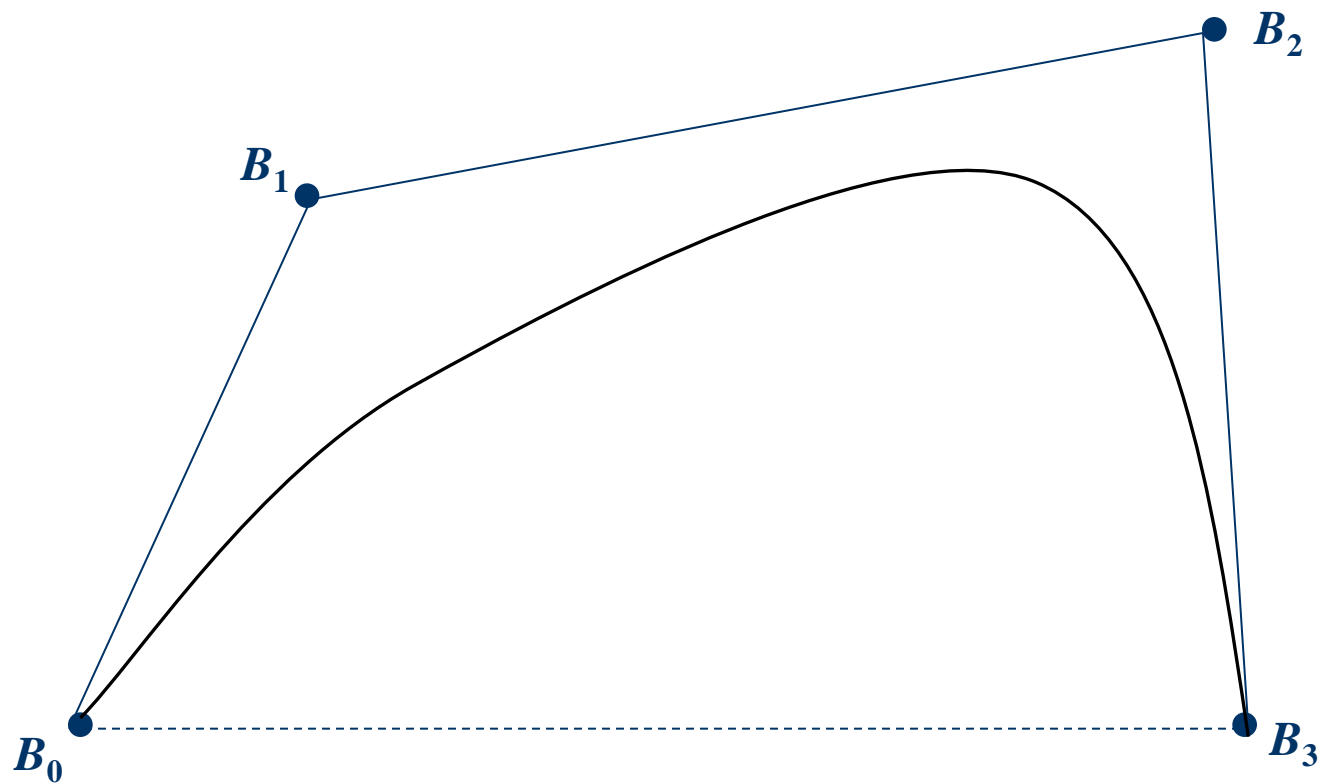


Figura 1

# Curvas de Bézier

- A curva paramétrica de Bézier é definida como:

$$P(t) = \sum_{i=0}^n B_i J_{n,i}(t), \quad 0 \leq t \leq 1$$

- Onde  $B_i$  representa cada um dos  $n+1$  pontos de controle considerados e  $J_{n,i}(t)$  são as funções que combinam a influência de todos os pontos (*blending functions*).
- Essas funções são descritas pelos polinômios de Bernstein como:

$$J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

# Curvas de Bézier

- onde  $n$  é o grau dos polinômios e:

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

( $i = 0, 1, \dots, n$ ) são os coeficientes binomiais.

- Essas funções  $J_{n,i}(t)$  satisfazem as condições:

$J_{n,i}(t) \geq 0$  para todo  $i$  entre 0 e  $n$ , e também:

$$\sum_{i=0}^n J_{n,i}(t) = 1, \quad 0 \leq t \leq 1$$

## Curvas de Bézier

$$J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

n=2, vejamos os J para i=0,1,2

$$J_{2,0}(t) = \binom{2}{0} t^0 (1-t)^2 = (1-t)^2$$

$$J_{2,1}(t) = \binom{2}{1} t^1 (1-t)^1 = 2t(1-t)$$

$$J_{2,2}(t) = \binom{2}{2} t^2 (1-t)^0 = t^2$$

## Curvas de Bézier

$$\sum_{i=0}^n J_{n,i}(t) = 1, \quad 0 \leq t \leq 1$$

$$1 = t + (1-t)$$

$$(a+b)^2 = a^2 + 2ab + b^2$$

$$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$

$$(t + (1-t))^2 = t^2 + 2t(1-t) + (1-t)^2$$

$$(t + (1-t))^3 = t^3 + 3t^2(1-t) + 3t(1-t)^2 + (1-t)^3$$

# Curvas de Bézier

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

$$(a+b)^2 = a^2 + 2ab + b^2$$

$$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$

$$(a+b)^5 = a^5 + 5a^4b + 10a^3b^2 + 10a^2b^3 + 5ab^4 + b^5$$



## Curvas de Bézier

- Expressões que definem as curvas de Bézier:
  - Para três pontos de controle  $\Rightarrow$  polinômios com grau 2.

$$P(t) = \sum_{i=0}^n B_i J_{n,i}(t), \quad 0 \leq t \leq 1$$

$$P(t) = (1-t)^2 B_0 + 2t(1-t) B_1 + t^2 B_2,$$

$$P(t) = (x(t), y(t)) = (1-t)^2 (x_0, y_0) + 2t(1-t) (x_1, y_1) + t^2 (x_2, y_2)$$

$$\text{Para } B_0 = (-4, 0), B_1 = (0, 5), B_2 = (7, 0)$$

$$x(t) = (1-t)^2 x_0 + 2t(1-t) x_1 + t^2 x_2$$

$$x(t) = -4(1-t)^2 + 7t^2 = -4 + 8t - 4t^2 + 7t^2 = -4 + 8t + 3t^2$$

$$y(t) = (1-t)^2 y_0 + 2t(1-t) y_1 + t^2 y_2$$

$$y(t) = 10t(1-t) = 10t - 10t^2$$

$$P(t) = (-4 + 8t + 3t^2, 10t - 10t^2)$$

# Curvas de Bézier

$$P(t) = \begin{bmatrix} t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \end{bmatrix}$$

# Curvas de Bézier

- Expressões que definem as curvas de Bézier:
  - Para quatro pontos de controle  $\Rightarrow$  polinômios com grau 3.

$$P(t) = (1 - t)^3 B_0 + 3t(1 - t)^2 B_1 + 3t^2(1 - t)B_2 + t^3B_3,$$

onde  $t$  inicialmente é 0.

$$P(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 3 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

# Curvas de Bézier - Algoritmo

## Exercício:

- (1) O programa superficies.cpp permite o ingresso interativo (pelo cliques do mouse) de  $n+1$  pontos de controle e constrói a curva de Bézier de grau  $n$ , correspondente. Responda as seguintes questões:
  - a. Qual trecho de código constrói as funções de Bernstein (Jni)
  - b. Qual trecho de código constrói a curva de Bézier.
  - c. Em qual trecho de código a curva é desenhada
  - d. O programa funciona, mas pode ser melhorado. Identifique quais melhoras podem ser feitas.

# Curvas de Bézier - Algoritmo

Exercício:

(2) Rode e entenda os programas [bezierCurves.cpp](#) e [bezierCurveWithEvalMesh.cpp](#) que desenharam curvas de Bézier usando comandos de OpenGL. Responda as seguintes questões:

- a. Qual trecho de código constrói as funções de Bernstein (Jni)
- b. Qual trecho de código constrói a curva de Bézier.
- c. Em qual trecho de código a curva é desenhada
- d. O programa apresenta uma proposta de interação, você teria outra proposta.

## Curvas de Bézier - Problemas

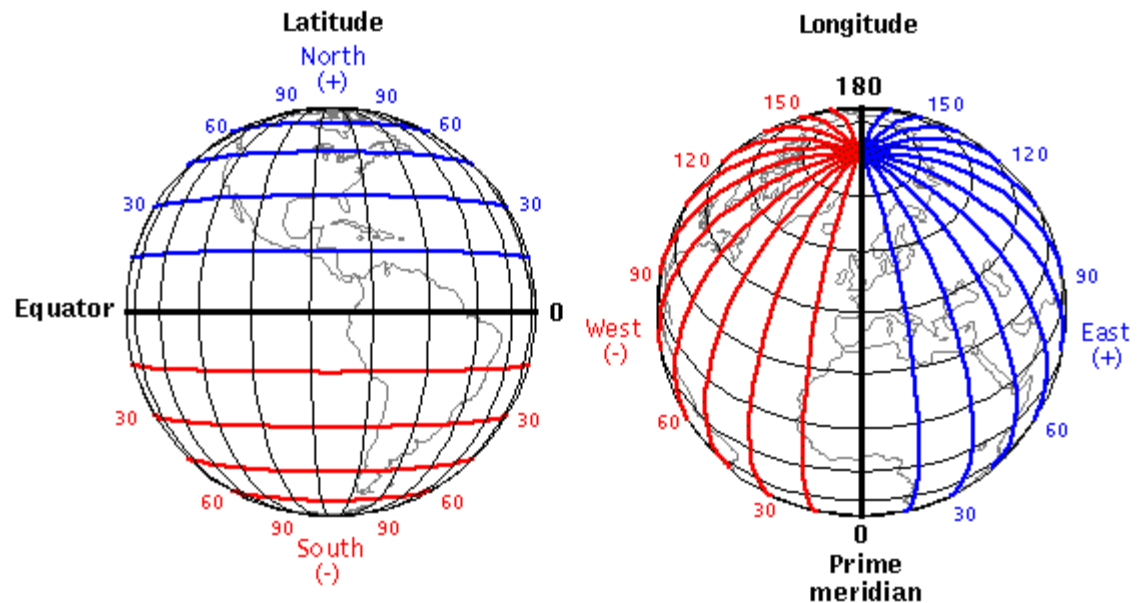
1. Falta de controle local : Uma alteração em um ponto no polígono de Bézier acarreta alterações em toda a curva de Bézier. Indesejável quando desejamos fazer ajustes finos.
2. O grau do polinômio cresce com o número de pontos de controle do polígono de controle.

# Aproximando Objetos Bi-dimensionais

Uma circunferência ou uma hélice são objetos unidimensionais pois são topologicamente equivalentes a uma linha. Agora veremos como aproximar objetos bi-dimensionais como uma esfera ou um cilindro.

A representação paramétrica de uma esfera segue o modelo de coordenadas geográficas do globo terrestre (latitude e longitude).

<https://www.geogebra.org/m/HJs4kEtb>



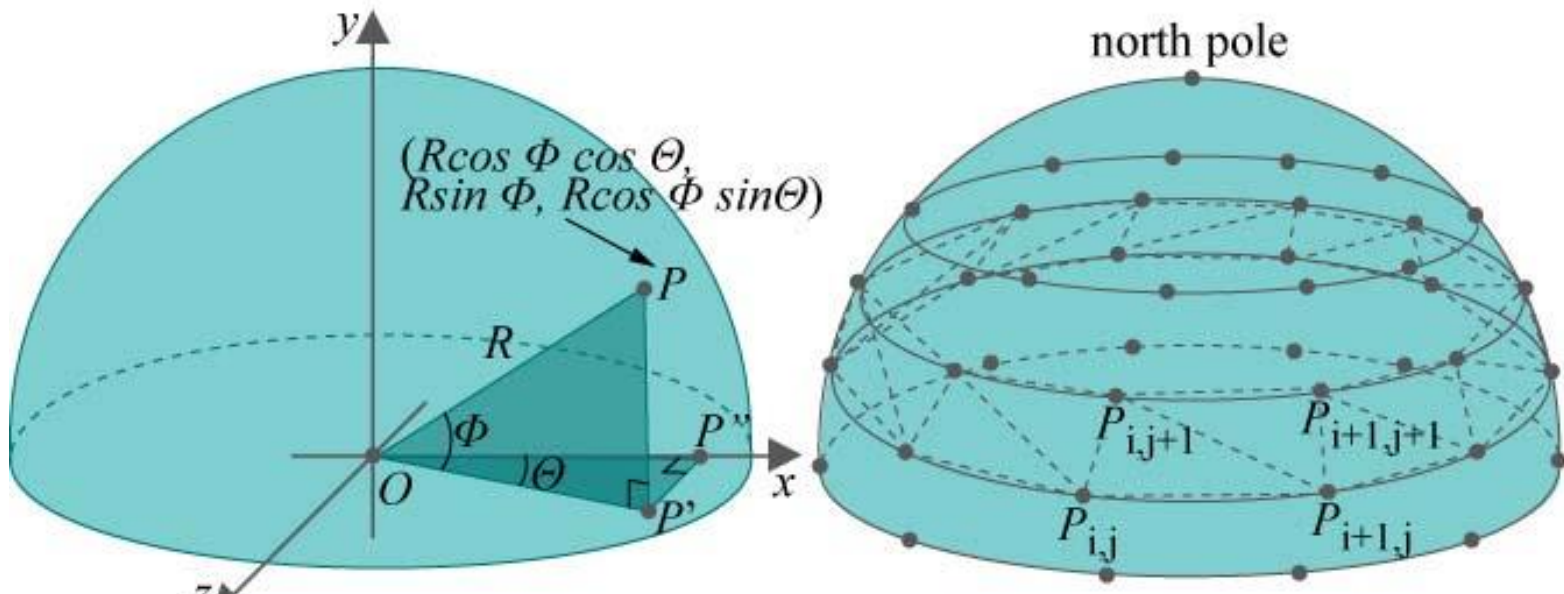
# Aproximando Objetos Bi-dimensionais

Considere um hemisfério de raio  $R$ , centralizado na origem  $O$  e com sua base circular sobre o plano  $xz$ . Suponha que  $\theta$  e  $\phi$  sejam a longitude theta e a latitude phi.

As coordenadas cartesianas são:

$$(R \cos \phi \cos \theta, R \sin \phi, R \cos \phi \sin \theta),$$

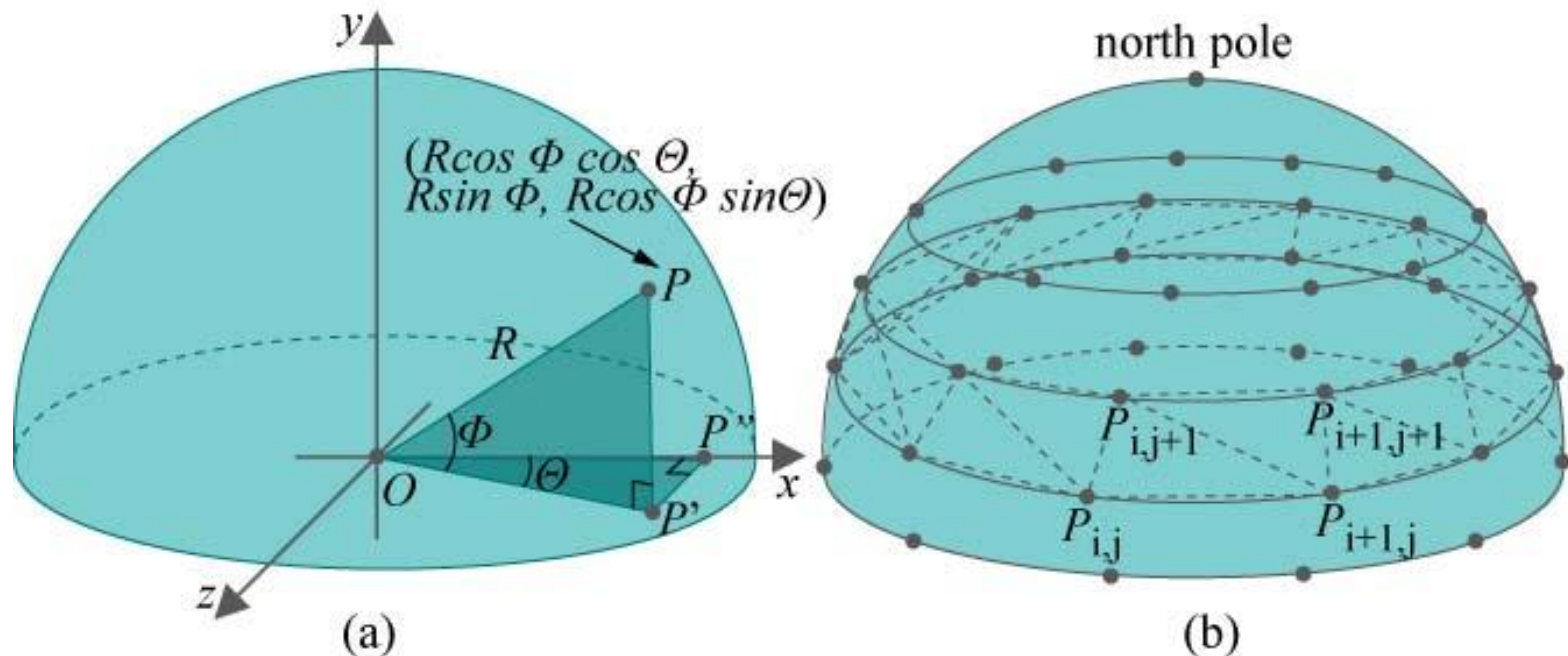
$$0 \leq \theta \leq 2\pi \text{ e } 0 \leq \phi \leq \pi/2$$





# Aproximando Objetos Bi-dimensionais

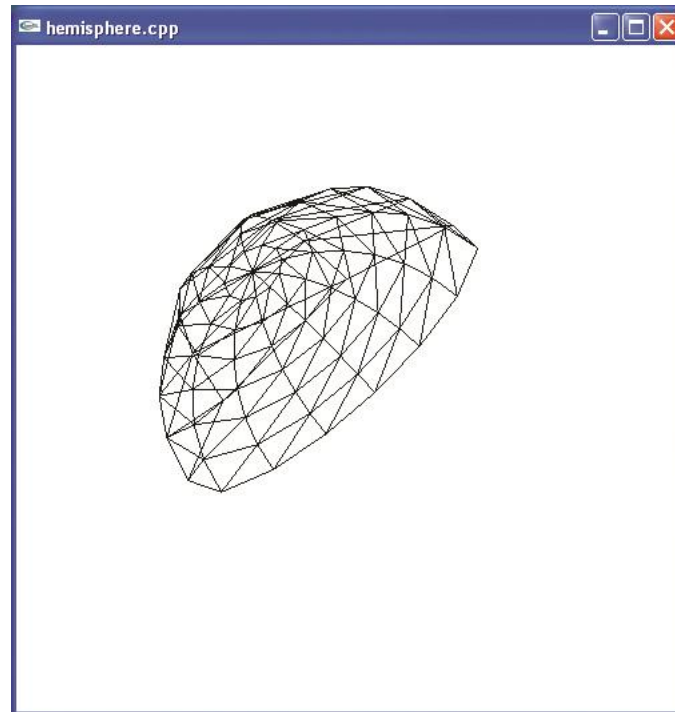
Amostramos o hemisfério em uma malha de  $(p+1)(q+1)$  pontos  $P_{ij}$ ,  $0 \leq i \leq p$ ,  $0 \leq j \leq q$ , onde a longitude de  $P_{ij}$  é  $\frac{2\pi i}{p+1}$  e a latitude de  $P_{ij}$  é  $\frac{\pi j}{q+1}$ . Os  $(p+1)$  pontos longitudinalmente igualmente espaçados são escolhidos entre cada uma das  $q+1$  latitudes igualmente espaçadas. Na figura  $p=10$  e  $q=4$ .



# Aproximando Objetos Bi-dimensionais

Experimento 2.26. Rode hemisphere.cpp que implementa

Experimento 2.27.



# Aproximando Objetos Bi-dimensionais

even now as the syntax is fairly intuitive. The set of three `glRotatef()`s, particularly, comes in handy to re-align a scene.

**Exercise 2.31. (Programming)** Modify `hemisphere.cpp` to draw:

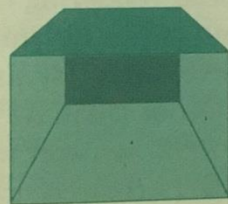
- (a) the bottom half of a hemisphere (Figure 2.35(a)).
- (b) a  $30^\circ$  slice of a hemisphere (Figure 2.35(b)).

Make sure the 'P/p/Q/q' keys still work.

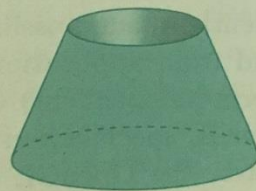
**Exercise 2.32. (Programming)** Just to get you thinking about animation, which we'll be studying in depth soon enough, guess the effect of replacing `glTranslatef(0.0, 0.0, -10.0)` with `glTranslatef(0.0, 0.0, -20.0)` in `hemisphere.cpp`. Verify.

And, here are some more things to draw.

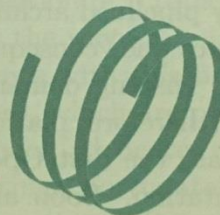
**Exercise 2.33. (Programming)** Draw the objects shown in Figure 2.36. Give the user an option to toggle between filled and wireframe renderings.



Lampshade



Another lampshade



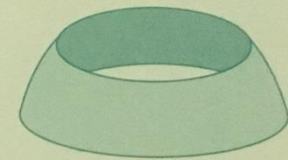
Spiral band



Rugby football

Figure 2.36: More things to draw.

A suggestion for the football, or ellipsoid, is to modify `hemisphere.cpp` to make half of an ellipsoid (a hemi-ellipsoid?). Two hemi-ellipsoids back to back would then give a whole ellipsoid.



(a)



(b)

Figure 2.35: (a) Half a hemisphere (b) Slice of a hemisphere.

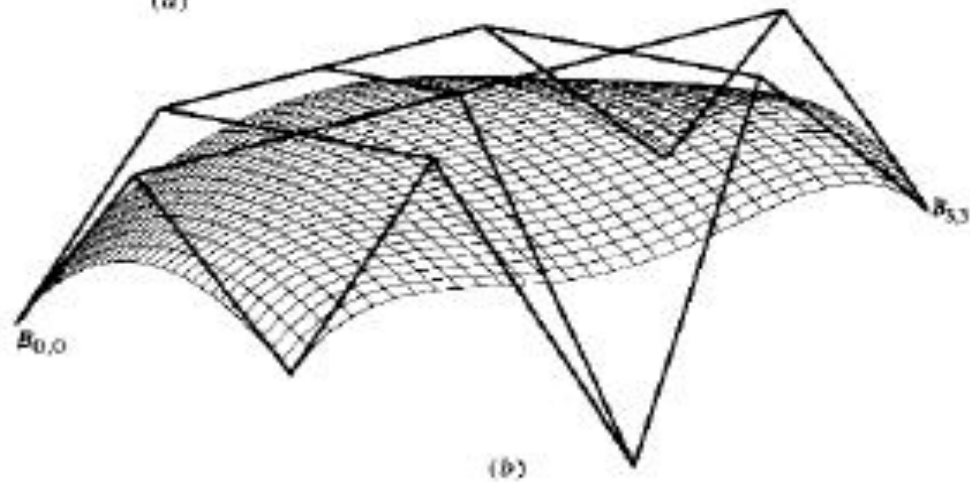
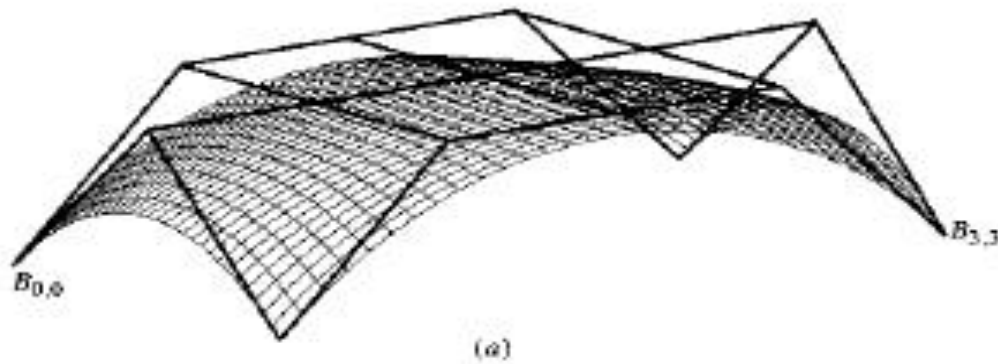
# Superfícies Bézier

- Generalização da idéia de curva de Bézier.
- Sejam  $B_{ij}$ ,  $i=0,\dots,m$ ,  $j=0,\dots,n$ , um conjunto de pontos no  $\mathbb{R}^3$  de tal forma que sua projeção no plano  $xOy$  seja formada pelos vértices de  $mn$  retângulos de mesmas dimensões. A superfície de Bézier definida no domínio  $[0,1] \times [0,1]$  é

$$Q(u, v) = \sum_{i=0}^n \sum_{j=0}^m B_{ij} J_{ni}(u) K_{mj}(v)$$

Onde  $J_{ni}$  e  $K_{mj}$  são os polinômios de Bernstein.

# Superfícies Bézier



# Curvas e Superfícies Bézier em OpenGL

1. Leia e entenda os programas [bezierSurface.cpp](#) e [bezierCanoe.cpp](#) que desenharam superfícies de Bézier usando comandos de OpenGL. Explique os comandos que geram as superfícies em ambos programas.

# Quádricas da GLU

A biblioteca GLU provê a renderização automática de objetos tridimensionais como esferas, cilindros e discos.

- Esferas:

`gluSphere(GLUquadricObj *obj, GLdouble radius, GLint slices, GLint stacks)`

- Cilindros:

`gluCylinder(GLUquadricObj *obj, GLdouble baseRadius, GLdouble topRadius, GLdouble height, GLint slices, GLint stacks)`

`topRadius == zero`, permite criar cone.

- Discos:

`gluDisk(GLUquadricObj *obj, GLdouble innerRadius, GLdouble outerRadius, GLint slices, GLint loops)`

`innerRadius != zero`, permite criar discos com furos.

*Veja e rode o programa `gluQuadrics.cpp` da pasta `SumantaGuha`*



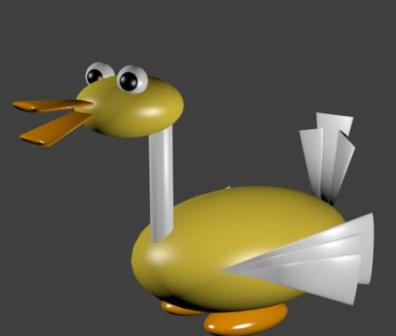
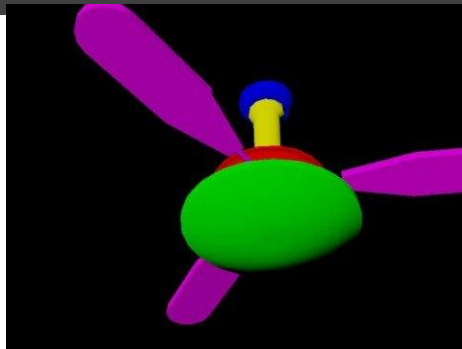
# Modelagem usando Superfícies Bézier e primitivas

1. Leia e entenda o programa torpedo.cpp que desenha um torpedo composto de diferentes pedaços:
  - (i) Corpo: cilindro da GLU
  - (ii) Nariz: hemisferio
  - (iii) Três barbatanas: discos parciais da GLU
  - (iv) Disco traseiro : disco da GLU
  - (v) Haste da hélice: cilindro da GLU
  - (vi) Três pás da hélice: pedaços bicúbicos Bezier
2. Desenhe um avião composto de vários pedaços. Use superfícies de Bézier e quádricas.



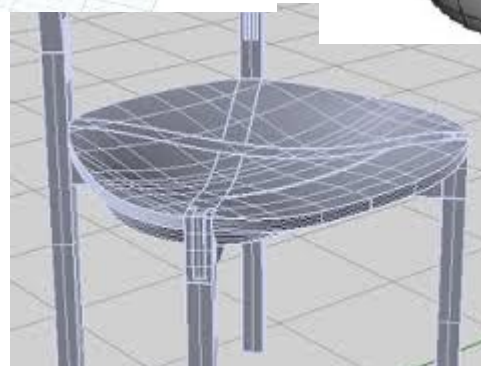
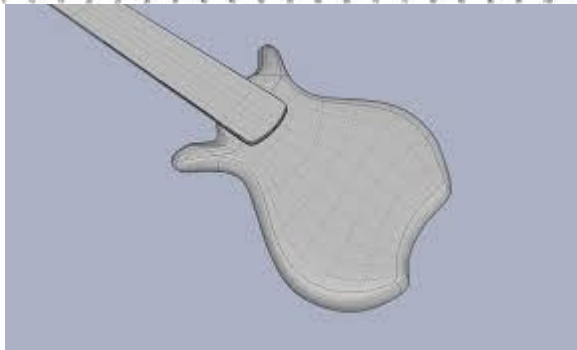
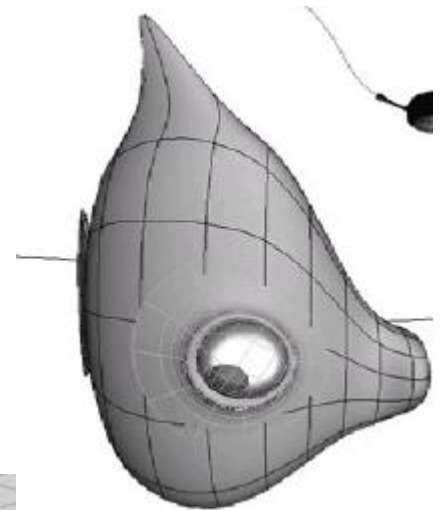
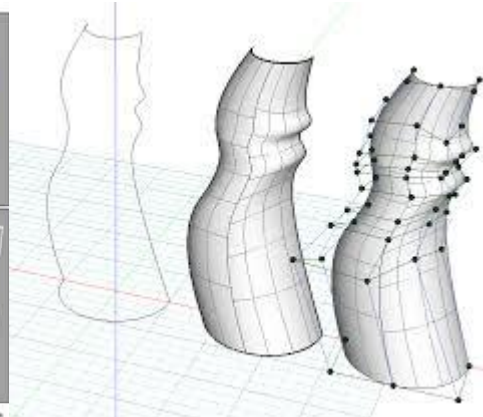
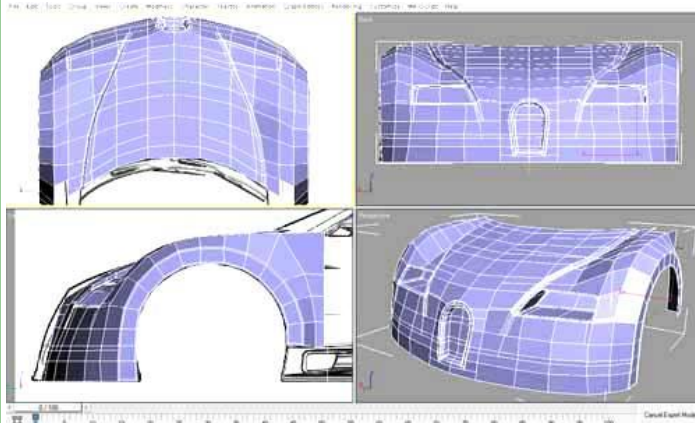
# Superfícies Bézier e quádricas em OpenGL

3. Modele os seguintes objetos.



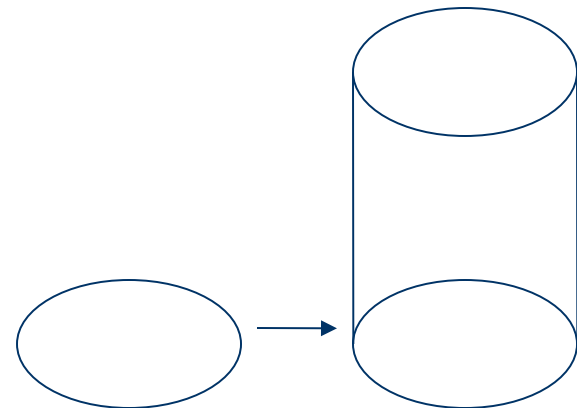
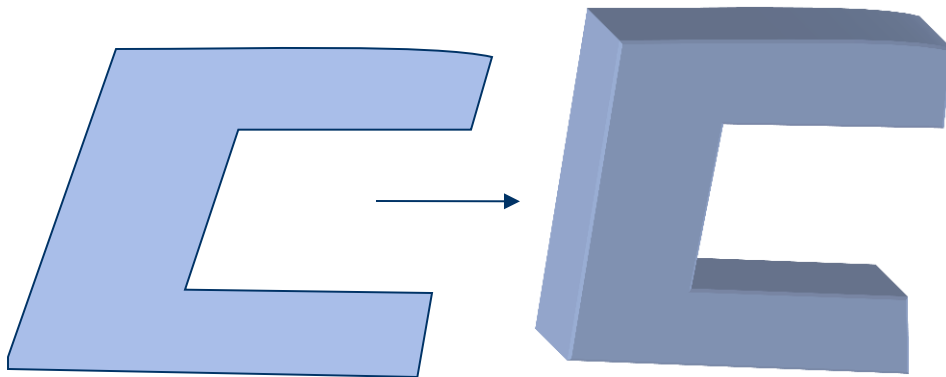
# Superfícies Bézier em OpenGL

4. Modele os seguintes objetos: carro, vestido, peixe, violão, cadeira com uso de superfícies de Bézier.



## Varredura (*Sweeping*)

- Uma superfície é descrita quando uma curva  $C1$  (curva geratriz) é deslocada no espaço, ao longo de uma trajetória dada por uma outra curva  $C2$  (caminho ou diretriz).
- Varredura translacional (Extrusão ou superfícies geradas por deslocamento)

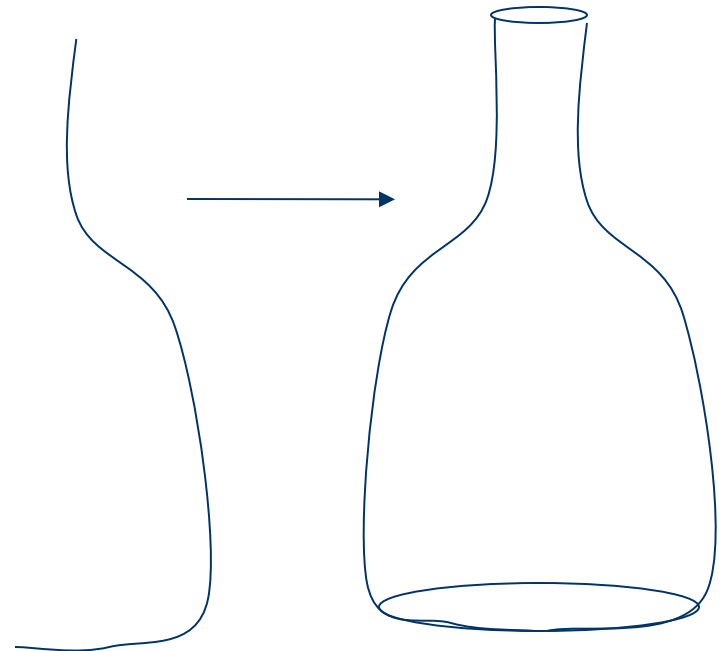
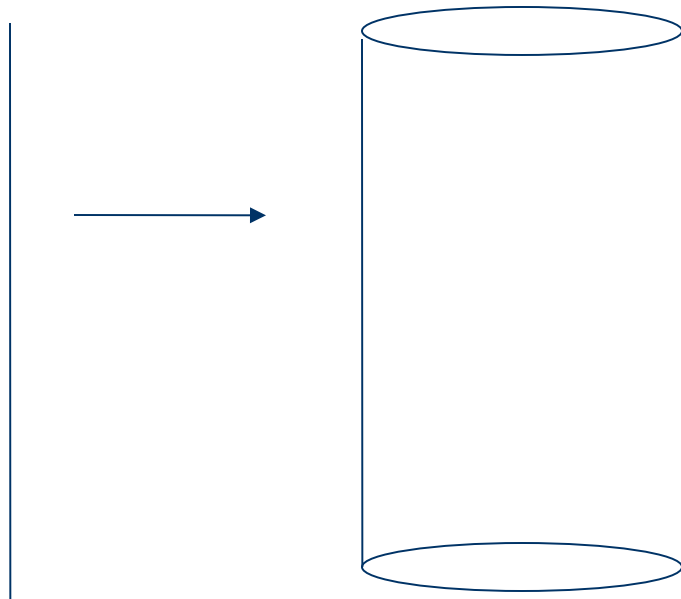


# Varredura (*Sweeping*)



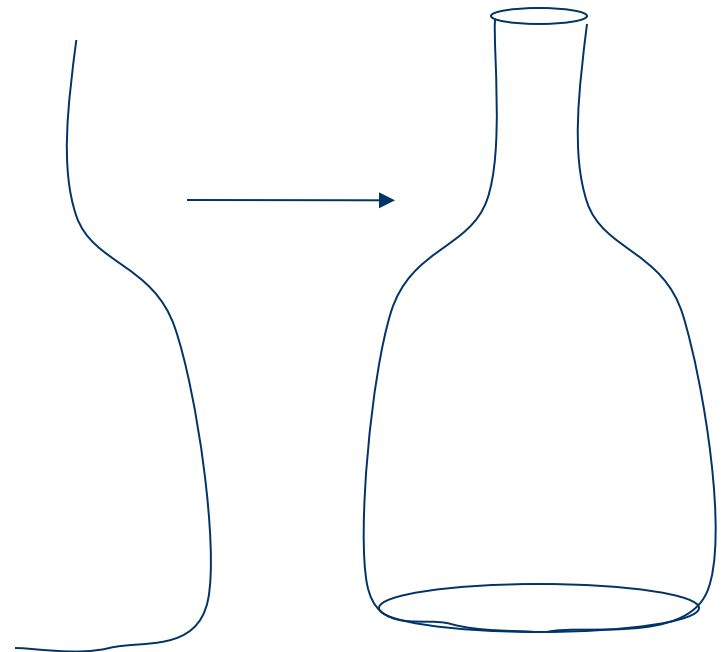
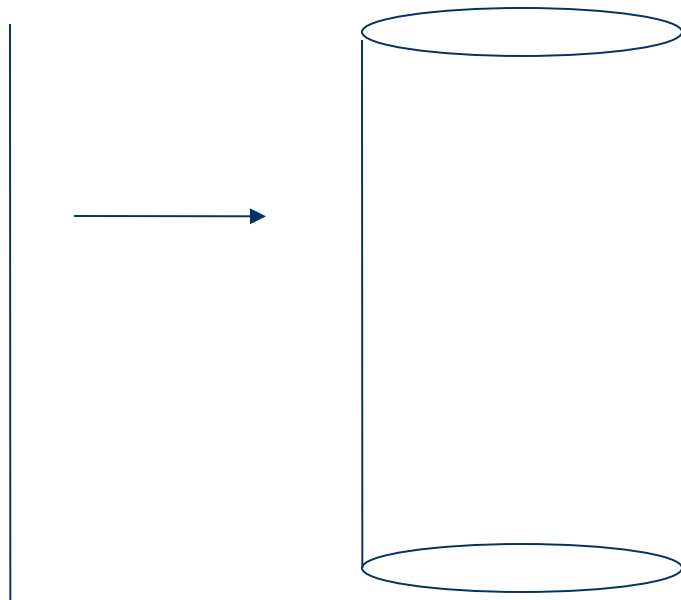
# Varredura (*Sweeping*)

- Varredura rotacional (ou superfícies de revolução)



# Varredura (*Sweeping*)

- Varredura rotacional (ou superfícies de revolução)



# Varredura (*Sweeping*)

