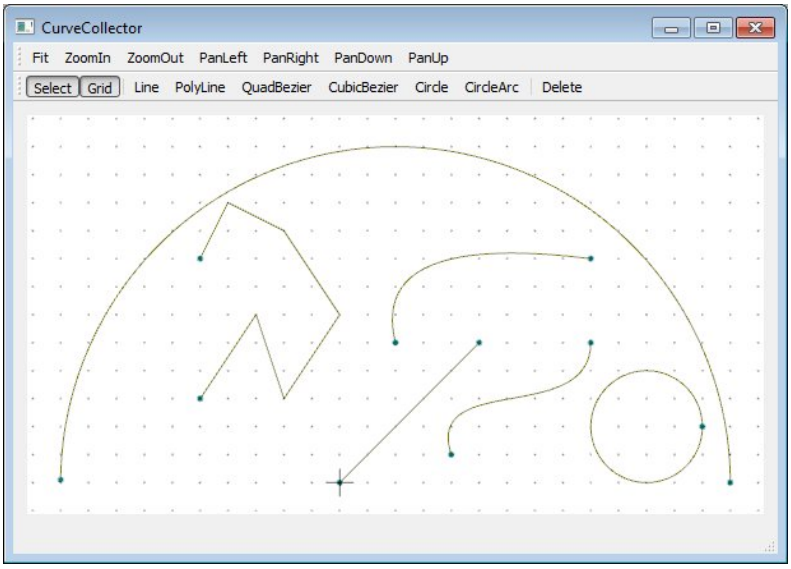
CIV 2802 – Sistemas Gráficos para Engenharia – 2025.1

3º Trabalho: Programa gráfico interativo com Qt e OpenGL

Criação via mouse e visualização de curvas digitais



Complemente o programa gráfico fornecido na *homepage* da disciplina
http://www.tecgraf.puc-rio.br/~lfm/compgraf-251 (procure terceiro trabalho).
Este programa utiliza sistema de interface Qt e o sistema gráfico OpenGL.
O entendimento do código do programa é parte do trabalho.

O programa trata dos seguintes tipos de curva:

Line: segmento de linha reta.
Polyline: linha poligonal
QuadBezier: curva Bezier quadrática.
CubicBezier: curva Bezier cúbica.
Circle: curva circular fechada.
CircleArc: curva de arco de círculo aberto.

Apenas a curva *Line* está implementada completamente. O objetivo do trabalho é adicionar ao programa o tratamento (modelagem e visualização) dos outros tipos de curvas:

Solicitado

1. Crie a interface gráfica do programa com o Qt Designer a partir da interface criada para o segundo trabalho, seguindo os passos que estão mostrados no Anexo A deste documento.

2. Completar as linhas de código no arquivo **`glcanvas.py`** que estão indicadas pelos comentários
   `# **** COMPLETE HERE: GLCANVAS_XX ****.`
   Essas linhas de código podem ser obtidas da solução do segundo trabalho.

3. Completar o código do programa no arquivo **`compgeom/compgeom.py`** no trecho que está indicado pelo comentário
   `# **** COMPLETE HERE: COMPGEOM_01 ****.`

4. Completar o código do programa no arquivo **`geometry/curves/polyline.py`** nos trechos que estão indicados pelos comentários
   `# **** COMPLETE HERE: POLYLINE_XX ****.`

5. Completar o código do programa no arquivo **`geometry/curves/quadbezier.py`** nos trechos que estão indicados pelos comentários
   `# **** COMPLETE HERE: QUADBEZIER_XX ****.`

6. Criar o arquivo **`geometry/curves/cubicbezier.py`** para a curva Bezier cúbica.

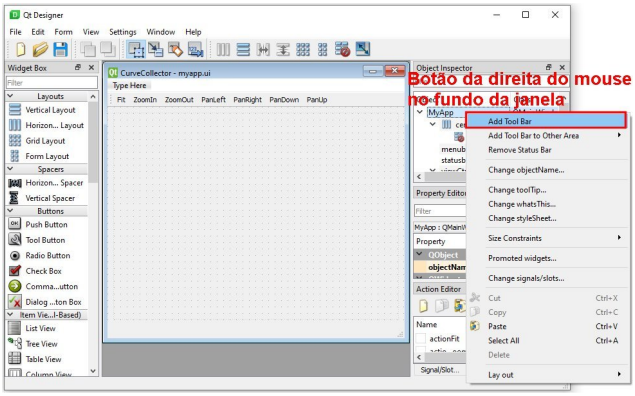
7. Completar o código do programa no arquivo **`geometry/curves/circle.py`** nos trechos que estão indicados pelos comentários
   `# **** COMPLETE HERE: CIRCLE_XX ****.`

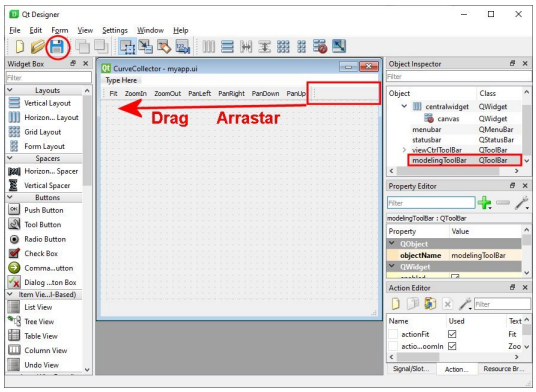
8. Criar o arquivo **`geometry/curves/circlearc.py`** para a curva de arco de círculo aberto.

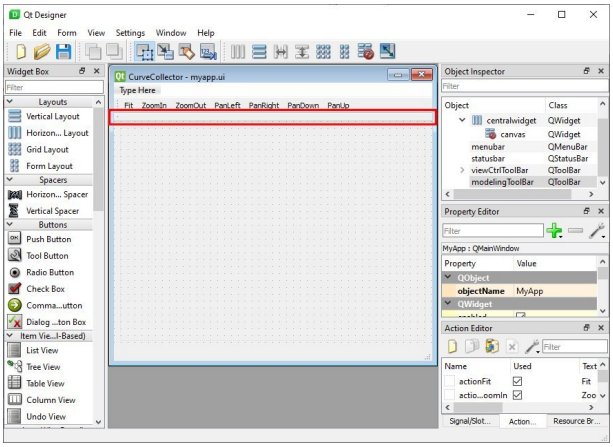
9. Enviar via e-mail em um arquivo **`.zip`** com os arquivos **`myapp.ui`**, **`myapp.py`**, todos os arquivos **`*.py`**, que correspondem à solução do trabalho, e uma pasta **`icons`** com os arquivos de imagens dos ícones criados para as ações de interface.

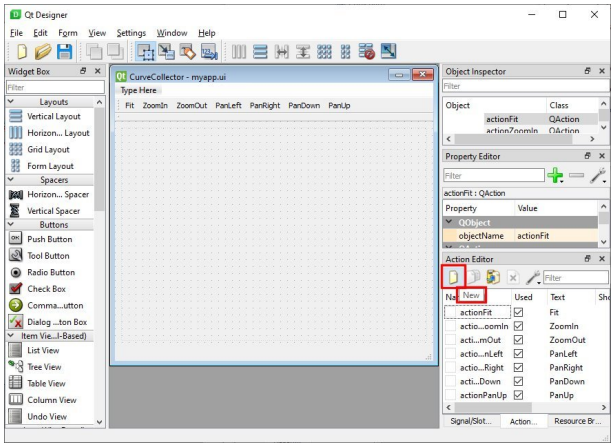
Anexo A – Criação da interface gráfica com o Qt Designer

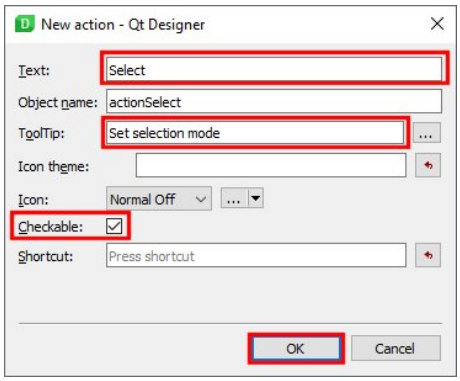
Na sequência são mostrados os passos para criação da interface gráfica do programa deste trabalho a partir da interface criada para o segundo trabalho.

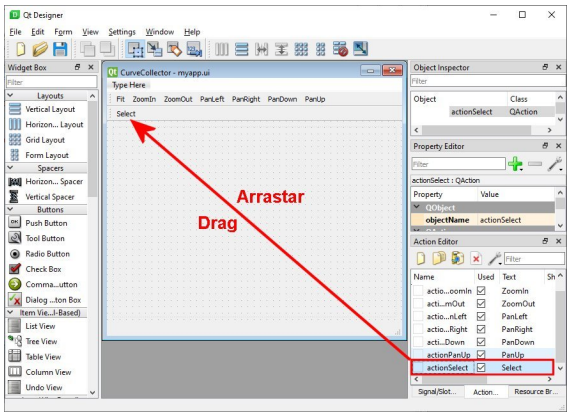


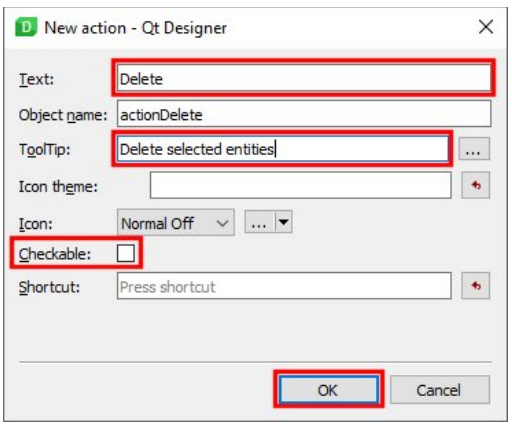


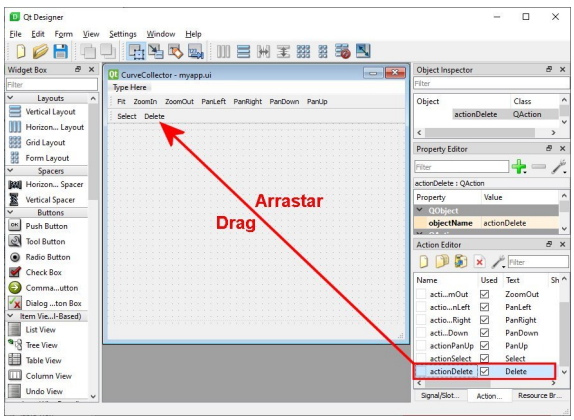


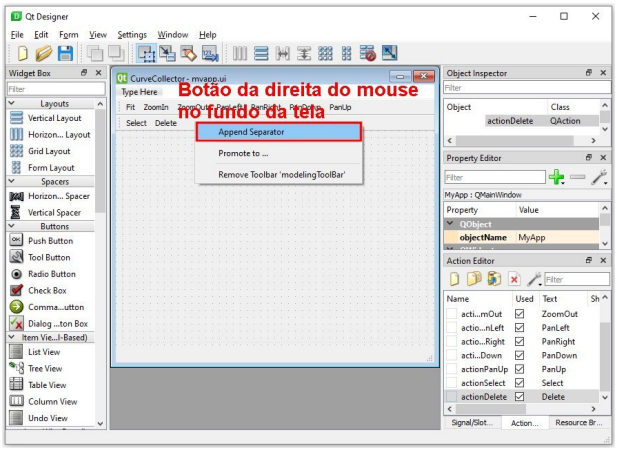


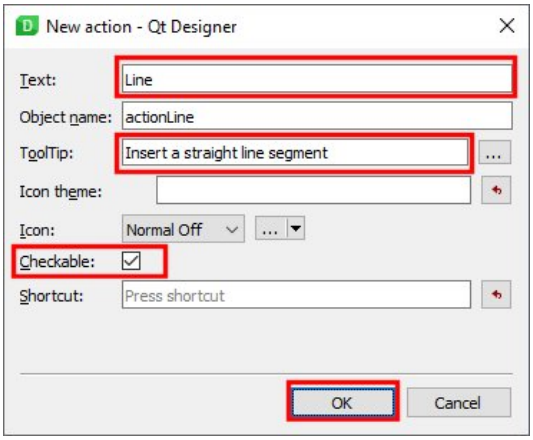


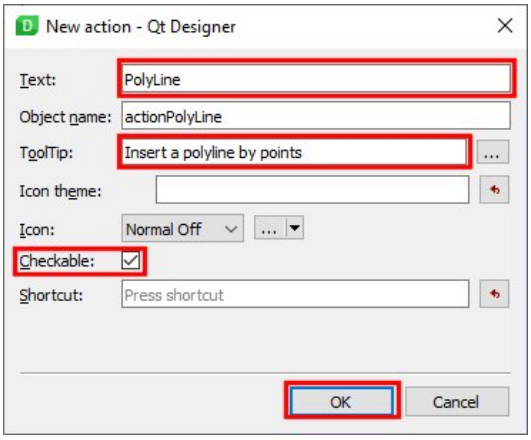


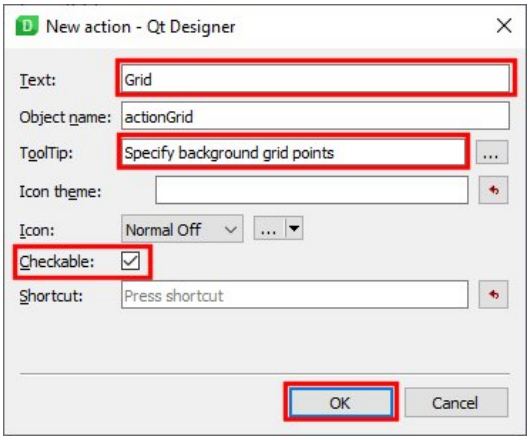


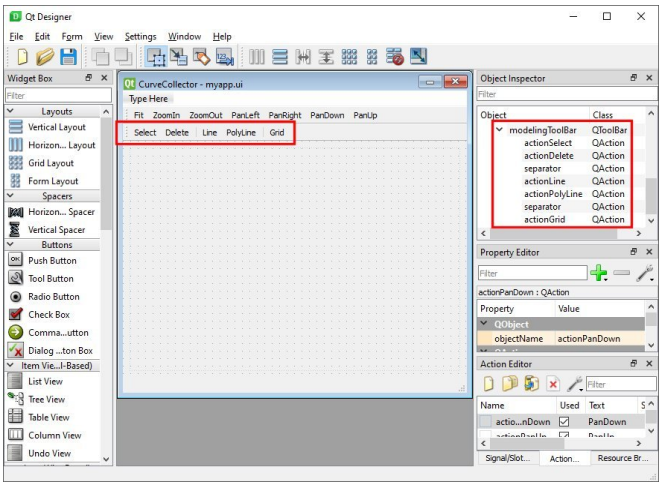



Botão da direita do mouse no fundo da tela
Append Separator
Promote to ...
Remove Toolbar 'modelingToolBar'


New action - Qt Designer
Text: Line
Object name: actionLine
ToolTip: Insert a straight line segment
Icon theme:
Icon: Normal Off
Checkable:
Shortcut: Press shortcut
OK
Cancel


New action - Qt Designer
Text: PolyLine
Object name: actionPolyLine
ToolTip: Insert a polyline by points
Icon theme:
Icon: Normal Off
Checkable:
Shortcut: Press shortcut
OK
Cancel

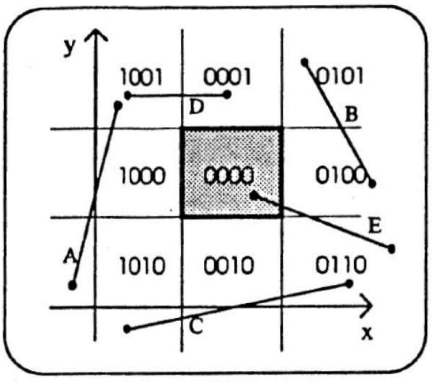

New action - Qt Designer
Text: Grid
Object name: actionGrid
ToolTip: Specify background grid points
Icon theme:
Icon: Normal Off
Checkable:
Shortcut: Press shortcut
OK
Cancel



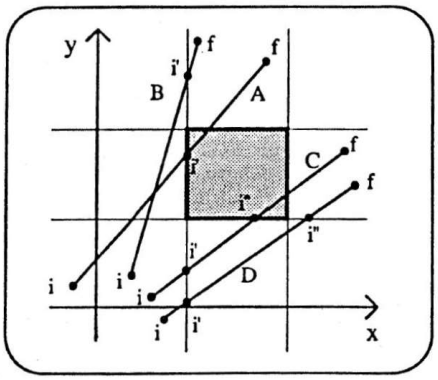
Anexo B – Alguns algoritmos e formulações utilizados

Seleção (Pick) de Segmento de Reta
(baseada no algoritmo de cerceamento de linhas)

Regiões com o mesmo código e posições triviais (A, B, C, D):



Algoritmo recursivo para recair nos casos triviais:



Esse procedimento está implementado nos dois métodos da classe **`CompGeom`** (Computational Geometry, veja arquivo **`compgeom/compgeom.py`**) mostrados abaixo:

```
# --------------------------------------------------------------------
# pickLine: check to see if given point (x,y) touches given line
# (x0,y0)-(x1,y1) based on given tolerance. Uses auxiliary method
# pickCode.
def pickCode(x, y, xmin, xmax, ymin, ymax):
    cod = [x < xmin, x > xmax, y < ymin, y > ymax]
    return cod

def pickLine(x0, y0, x1, y1, x, y, tol):
    # Define atraction window.
    xmin = x - tol
    xmax = x + tol
    ymin = y - tol
    ymax = y + tol

    cod1 = Curve.pickCode(x1, y1,  xmin, xmax, ymin, ymax)
    while True:
        cod0 = Curve.pickCode(x0, y0,  xmin, xmax, ymin, ymax)

        count = 0
        for i in range(0, 4):
            if cod0[i] and cod1[i]:  # Test no-trivial pick
                break
            else:
                count += 1
        if count != 4:
            break

        # Move point 0 to window limit.
        if cod0[0]:
            y0 += (xmin - x0) * (y1 - y0) / (x1 - x0)
            x0 = xmin
        elif cod0[1]:
            y0 += (xmax - x0) * (y1 - y0) / (x1 - x0)
            x0 = xmax
        elif cod0[2]:
            x0 += (ymin - y0) * (x1 - x0) / (y1 - y0)
            y0 = ymin
        elif cod0[3]:
            x0 += (ymax - y0) * (x1 - x0) / (y1 - y0)
            y0 = ymax
        else:
            return True

    return False
```

Ponto mais próximo em um segmento de reta em relação a um ponto fornecido

Esse procedimento deve ser utilizado para implementar o trecho de código no arquivo **`compgeom/compgeom.py`** identificados por:
`# **** COMPLETE HERE: COMPGEOM_01 ****.`

PONTO MAIS PRÓXIMO EM UM SEGMENTO DE RETA UTILIZANDO PRODUTO INTERNO

Projeção do ponto C na reta AB:

$$C' = A + t_{C'}(B - A)$$

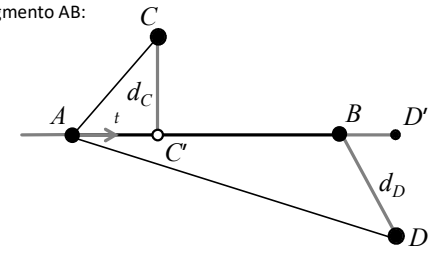
Ponto mais próximo de C no segmento AB:

$$P = C'$$

Valor paramétrico do ponto C' no segmento AB:

$$t_{C'} = \frac{\overrightarrow{AB} \circ \overrightarrow{AC}}{\left|\overrightarrow{AB}\right|^2}$$ (produto interno)

$$0 < t_{C'} < 1$$



Projeção do ponto D na reta AB:

$$D' = A + t_{D'}(D - C)$$

Valor paramétrico do ponto D' no segmento AB:

$$t_{D'} = \frac{\overrightarrow{AB} \circ \overrightarrow{AD}}{\left|\overrightarrow{AB}\right|^2}$$

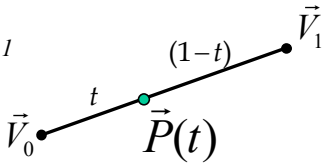
$$t_{D'} > 1$$

Ponto mais próximo de D no segmento AB:

$$P = B$$

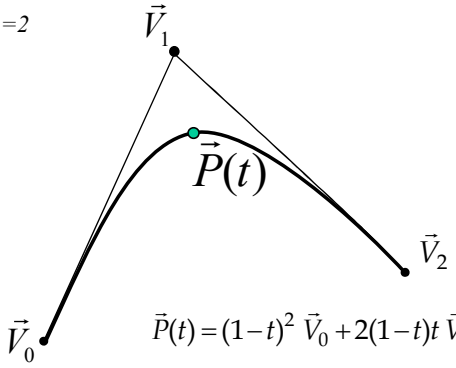
Anexo C – Formulação de curvas paramétricas do tipo Bezier linear, quadrática e cúbica

Bezier n=1



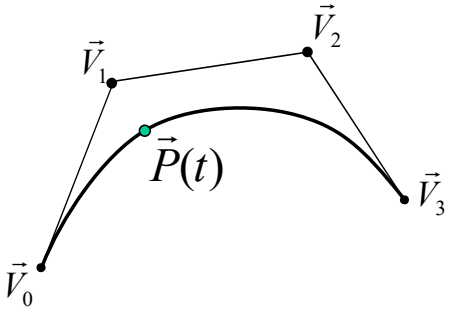
$$\vec{P}(t) = (1-t)\vec{V}_0 + t\vec{V}_1$$

Bezier n=2



$$\vec{P}(t) = (1-t)^2\,\vec{V}_0 + 2(1-t)t\,\vec{V}_1 + t^2\,\vec{V}_2$$

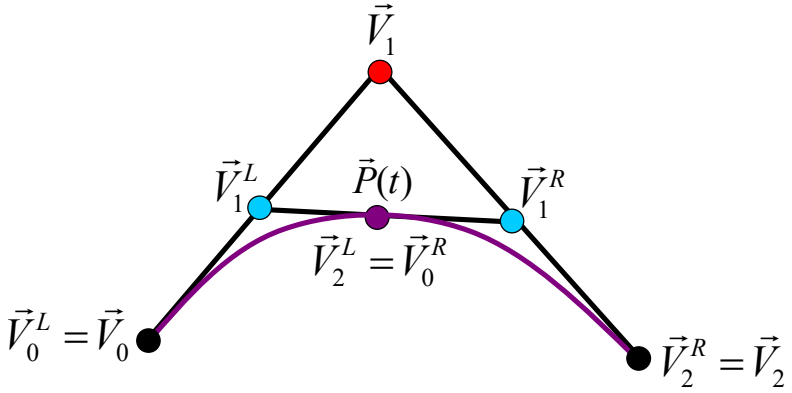
Bezier n=3



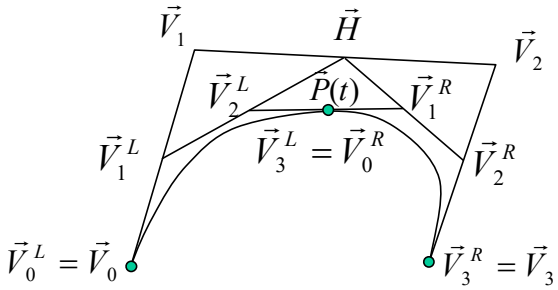
$$\vec{P}(t) = (1-t)^3\,\vec{V}_0 + 3(1-t)^2 t\,\vec{V}_1 + 3(1-t)t^2\,\vec{V}_2 + t^3\,\vec{V}_3$$

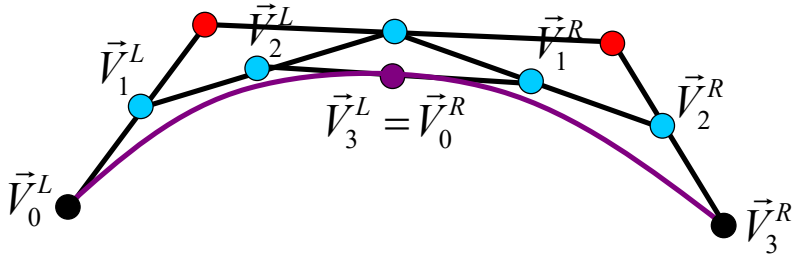
Subdivisão de curvas paramétricas do tipo Bezier quadrática e cúbica

Subdivisão de Bézier Quadrática



Subdivisão de Bézier Cúbica





Derivada (tangente) de curvas Bezier quadrática e cúbica

Derivada (tangente) de uma Bezier quadrática (n=2) é uma Bezier linear (n=1):

$$\vec{P}(t) = (1-t)^2\ \vec{V}_0 + 2(1-t)t\ \vec{V}_1 + t^2\ \vec{V}_2$$

$$\frac{d}{dt}\vec{P}(t) = (1-t)\cdot 2(\vec{V}_1 - \vec{V}_0) + t\cdot 2(\vec{V}_2 - \vec{V}_1)$$

$$\frac{d}{dt}\vec{P}(t) = (1-t)\vec{V}_0^* + t\vec{V}_1^*$$

$$\vec{V}_0^* = 2(\vec{V}_1 - \vec{V}_0) \qquad \vec{V}_1^* = 2(\vec{V}_2 - \vec{V}_1)$$

Derivada (tangente) de uma Bezier cúbica (n=3) é uma Bezier quadrática (n=2):

$$\vec{P}(t) = (1-t)^3\ \vec{V}_0 + 3(1-t)^2 t\ \vec{V}_1 + 3(1-t)t^2\ \vec{V}_2 + t^3\ \vec{V}_3$$

$$\frac{d}{dt}\vec{P}(t) = -3(1-t)^2\ \vec{V}_0 + \left[-6(1-t)t + 3(1-t)^2\right]\vec{V}_1 + \left[-3t^2 + 6(1-t)t\right]\vec{V}_2 + 3t^2\ \vec{V}_3$$

$$\frac{d}{dt}\vec{P}(t) = (1-t)^2\cdot 3(\vec{V}_1 - \vec{V}_0) + 2(1-t)t\cdot 3(\vec{V}_2 - \vec{V}_1) + t^2\cdot 3(\vec{V}_3 - \vec{V}_2)$$

$$\frac{d}{dt}\vec{P}(t) = (1-t)^2\vec{V}_0^* + 2(1-t)t\vec{V}_1^* + t^2\vec{V}_2^*$$

$$\vec{V}_0^* = 3(\vec{V}_1 - \vec{V}_0) \qquad \vec{V}_1^* = 3(\vec{V}_2 - \vec{V}_1) \qquad \vec{V}_2^* = 3(\vec{V}_3 - \vec{V}_2)$$