Curvas de Bézier

EP1 de MAC0210

Renato Lui Geh, NUSP: 8536030

1 Introdução

O EP foi feito na linguagem Python. Foi usada a biblioteca pyglet[1], que age como um *wrapper* de OpenGL para Python. Toda a parte de desenho e GUI foi feita com esta biblioteca.

Para rodar o EP, é preciso do Python 3 e que a biblioteca pyglet esteja instalada. Além disso, alguns cálculos foram feitos com NumPy.

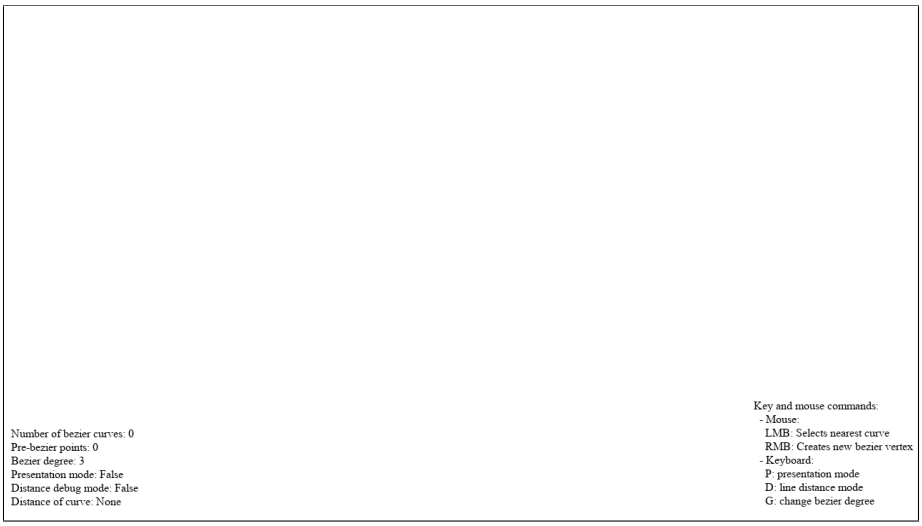
Para executar o EP, faça:

```
python3 main.py
```

2 Uso

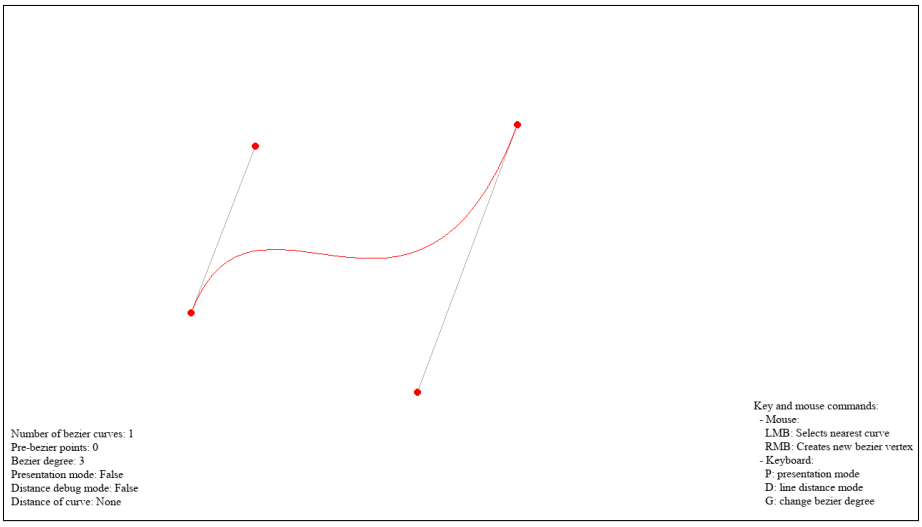
O arquivo `main.py` roda o programa, que constrói uma janela com um canvas vazio. A janela inicial apresenta as instruções de uso e algumas informações do ambiente atual.

[1]Disponível em `https://bitbucket.org/pyglet/pyglet/wiki/Home`



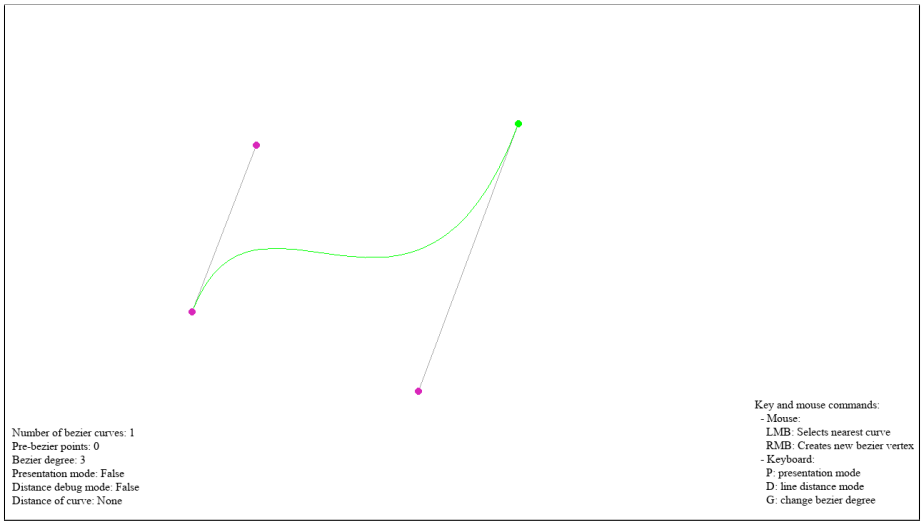
No canto inferior esquerdo ficam as informações atuais do ambiente. No canto inferior direito é possível encontrar os comandos disponíveis.

Com o botão direito do mouse, cria-se um dos pontos da curva de Bézier. Se o grau da curva é dois, são precisos três pontos; se três, quatro pontos são necessários.

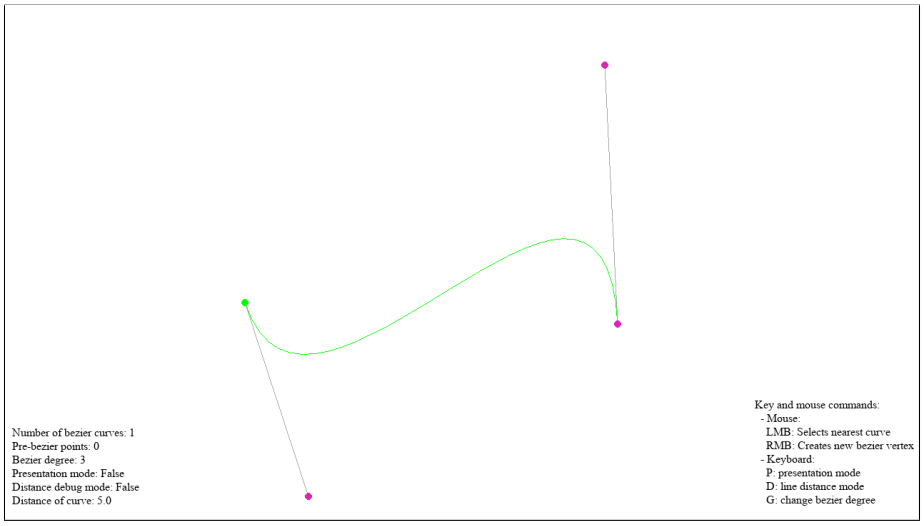


Após criadas, é possível selecionar uma curva com o botão esquerdo. Ao clicar com o botão esquerdo, seleciona-se a curva mais próxima do cursor. O modo de debug de distância (hotkey D) desenha o vetor diferença entre a posição do cursor e o ponto mais próximo da curva ao cursor. No painel de informação também é

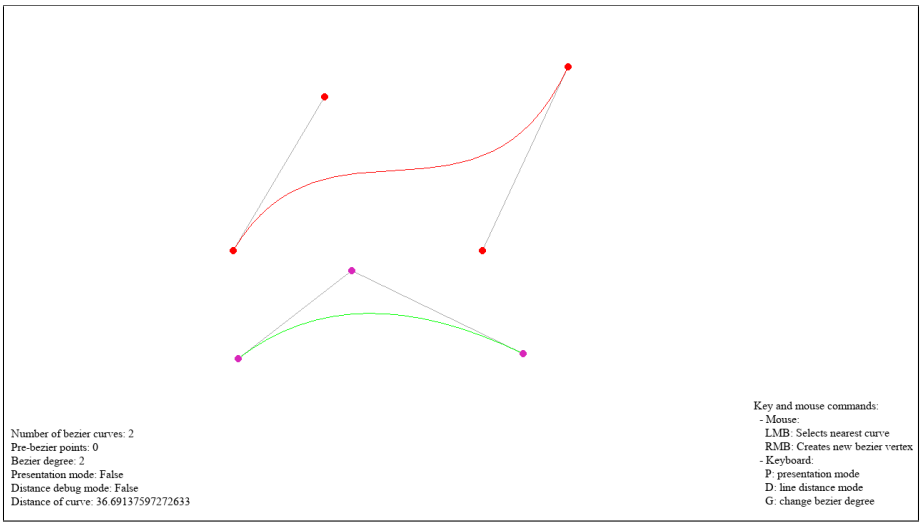
possível ver a distância calculada “Distance to curve”.



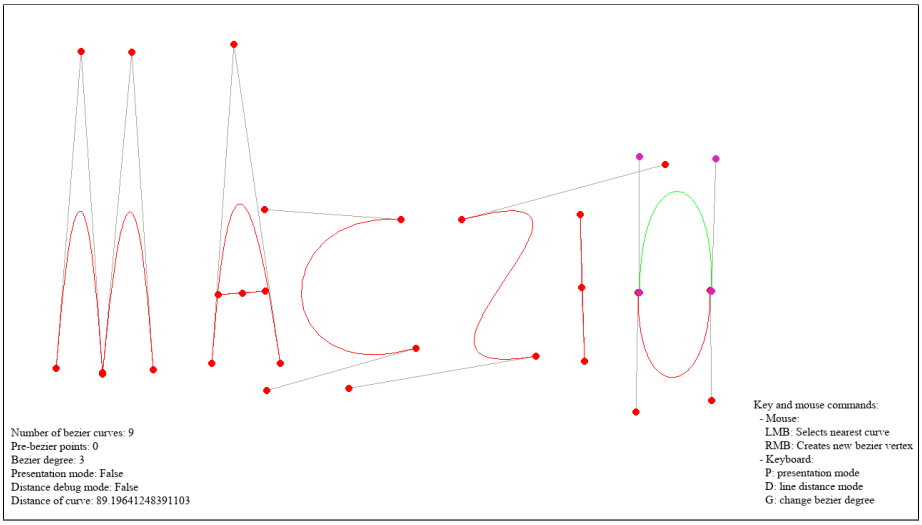
Após selecionada, a curva pode ser alterada modificando as posições dos seus pontos. Selecionam-se os pontos clicando nas bolas rosas da curva. O vértice selecionado é então redesenhado com a cor verde. Alternativamente, é possível ciclar pelos pontos de forma mais rápida usando TAB. Após selecionado, o ponto pode ser movido apertando com o botão esquerdo e arrastando.



Com a tecla G, é possível mudar o grau das curvas a serem criadas. Foram implementadas curvas de grau dois e três.

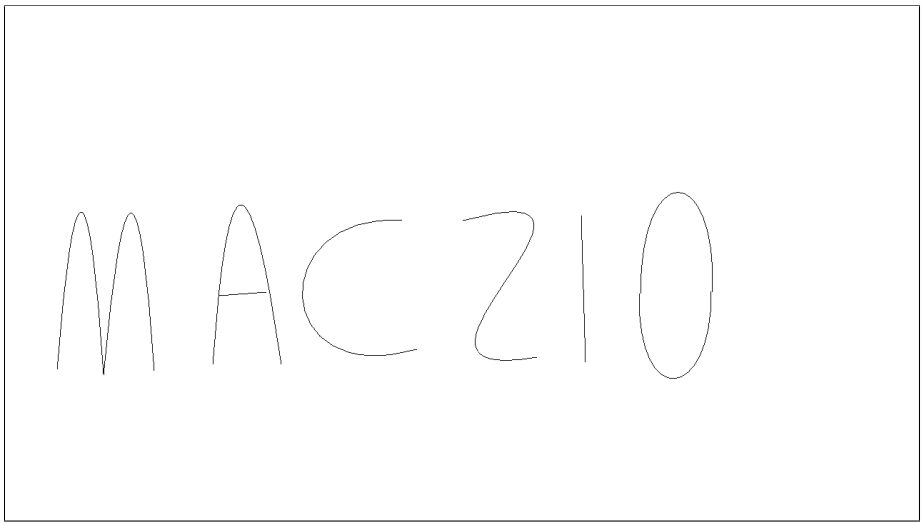


O programa lida bem com várias curvas, e mesmo quando há interseção das curvas e de seus vértices, é possível seleciona-las e movimenta-las livremente.



É possível entrar em modo apresentação (hotkey P). Quando em modo de apresentação, o programa deixa de desenhar os pontos da curva, os paineis de instrução e comandos, e deixa de colorir as retas selecionadas.

É possível remover uma curva selecionando-a e apertando a tecla DELETE. Para limpar o canvas e remover todas as curvas, aperte SHIFT+X.



3 Estrutura do código

O arquivo `main.py` que é usado para começar o EP cria um objeto da classe `Frame`, localizado em `frame.py`. Esta classe cuida da criação da janela, do desenho de cada elemento do canvas, e do input de mouse e teclado.

Toda vez que o botão direito é usado dentro do canvas, o `Frame` registra os pontos e verifica o número de pontos criados. Se este número for igual ao grau da curva de Bézier somado a um, então cria-se um novo objeto `Bezier` (em `bezier.py`), que representa a curva.

Esta classe trata de tudo sobre a curva. Seu construtor toma os pontos da curva e seu grau. Deste jeito podemos computar os coeficientes do polinômio (representado pela variável `Bezier.k`) e os coeficientes de sua derivada (variável `Bezier.dk`). Esse calculo é feito pelo método `recompute`, que é chamado toda vez que mudamos os pontos da curva.

A curva é desenhada pelo método de DeCasteljau. A função `_de_casteljau` e `_de_casteljau2` desenham as curvas de grau três e dois respectivamente.

Para computar a distância de uma curva de Bézier B até um ponto q, usa-se o método `Bezier.distance`. A distância é computada tomando o mínimo da distância euclideana:

$$\arg\min_{0\leq t\leq 1} d(q, B) = \sqrt{(B_x(t) - q_x)^2 + (B_y(t) - q_y)^2}$$

Derivando e igualando a zero, temos um polinômio cujas raízes representam os máximos e mínimos da distância. Usamos a função `np.roots` da biblioteca NumPy para achar as raízes do polinômio. Em seguida, descartamos todas as raízes que não estejam em $\mathbb{R}$ e tomamos o mínimo dos restantes.

Os métodos `Bezier.f` e `Bezier.df` tomam um t e avaliam $(B_x(t), B_y(t))$ e $\frac{\partial}{\partial t}(B_x(t), B_y(t))$.

A classe `Label` desenha os painéis de informação e instrução, e pode ser encontrada no arquivo `label.py`.

O arquivo `utils.py` contém funções de conveniência, como por exemplo a distância de um ponto até um segmento de reta, o mínimo entre dois pontos e um outro ponto, e uma rotina para desenhar bolas através de triângulos (`GL_TRIANGLE_FAN` do OpenGL).