Tainan Eduarda Silva Bueno
7° período - Ciência da Computação - Puc Minas - Computação Gráfica - Prof. Rosilane

---

Documentação

Descrição

Essa documentação é referente ao programa que foi desenvolvido para a disciplina de computação gráfica, onde foi implementado alguns algoritmos e mostrado os resultados em uma interface. O programa foi desenvolvido em Python, pelo notebook da ide do Jupyter, e utilizado da biblioteca “Tkinter” para fazer a interface.

Pré requisitos

- Algum editor de código que rode Python (.py ou .ipynb)
- Python 3 instalado
- Bibliotecas “tkinter” e “math” importadas

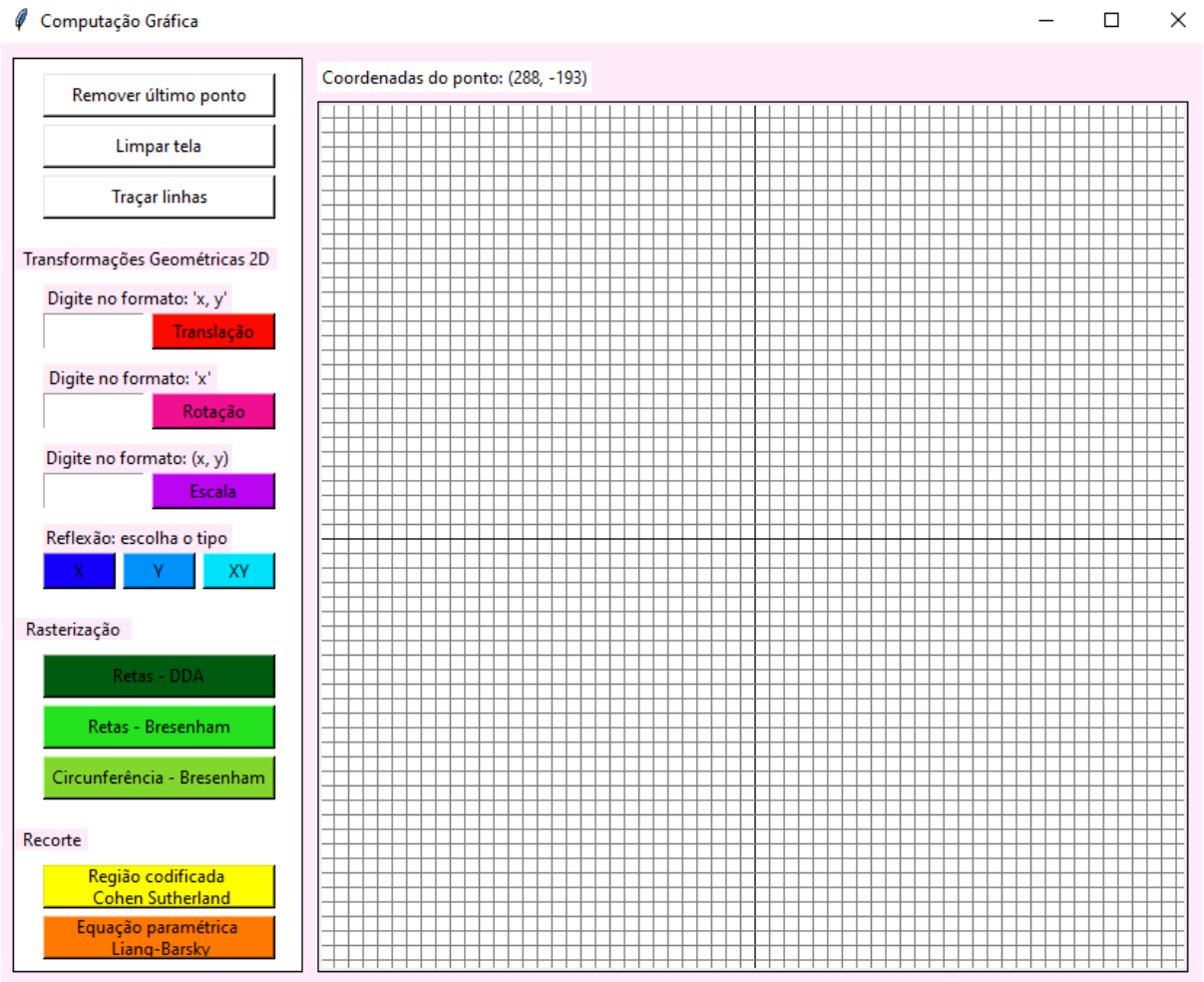
Rodando o programa

Se estiver com o código do notebook, tem que rodar cada cada célula e a última será a que vai mostrar a interface, na primeira célula já estão importadas as bibliotecas, só precisa de ter os outros pré requisitos.
Se estiver com o código .py, basta rodar tudo e já vai aparecer a interface, nas primeiras linhas do código já estão importadas as bibliotecas, precisa dos outros pré requisitos.

Estrutura do código

O notebook do código foi organizado em células, a primeira com as bibliotecas, a segunda com as funções dos algoritmos de computação gráfica, a terceira com as funções da interface e a quarta, com a criação da interface e inserção dos elementos, como botões e labels.

A interface possui dois frames (frame botões e frame quadriculado). No primeiro está organizado os labels, inputs e botões, chamando suas respectivas funções. No segundo tem um canvas que está desenhado com linhas para formar uma tela quadriculada, nesse canvas que serão mostrados os pontos, linhas e resultados.



O canvas possui alguns métodos como o “.bind()” que permite salvar eventos realizados pelo mouse, como o “<botao-1>” que salva as informações do lugar que foi clicado com o mouse esquerdo. Possui também métodos que desenham figuras geométricas, como “.create_rectangle”, “.create_oval”.
Quando uma parte do canvas é clicado, com a função “salvar_ponto”, é desenhado um oval em cima de onde foi salvo pelo evento “.bind()

```
canvas.bind("<Button-1>", salvar_ponto) #salva as informações do clicar com o botão esquerdo
```

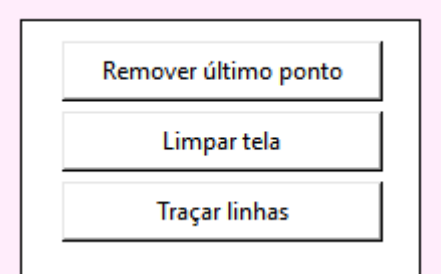
```
canvas.create_oval(x+300, y+300, x+300, y+300, outline="#23e320")
```

Os pontos selecionados são salvos em uma lista (pontos) e é com base nessa lista que são calculadas algumas funções. Por exemplo, se eu quiser fazer um recorte, ele vai pegar os 4 últimos pontos da lista.
Quando é traçado alguma reta, elas ficam salva em uma lista também (retas) e tem funções que já fazem o cálculo em cima da reta. Por exemplo, se eu quiser transladar uma reta, ele pega todas as retas que estão na lista.

A função “remover último ponto” apaga da lista de pontos o último que foi inserido e pega o “x, y” dele para remover da tela o ponto que foi desenhado.

A função “limpar tela” apaga todos os elementos da lista de pontos e retas, utilizando um método do canvas “.delete()”, apaga tudo que está no canvas.

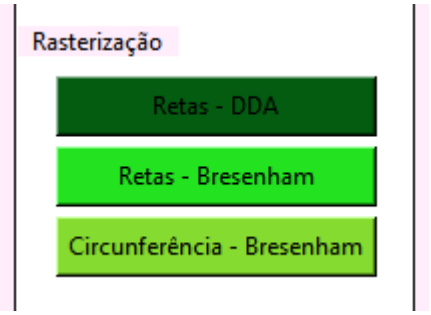
A função “traçar linha” liga todos os pontos com o “.create_line” que estão no array temporário, esse array pega todos os pontos que estão na tela até clicarem no botão de traçar linha. Após os pontos serem ligados, o array é esvaziado.



O botão “retas dda” realiza o algoritmo de Analisador Diferencial Digital para traçar retas, ele seleciona os dois últimos pontos e traça a reta colorindo os pixels, que são representados pelo “create_oval” que cria pontos pequenos.

O botão “retas bresenham” também traça linha de acordo com o algoritmo de Bresenham que seleciona os dois últimos pontos e vai desenhando pontos pequenos com o “create_oval” até formar um reta.

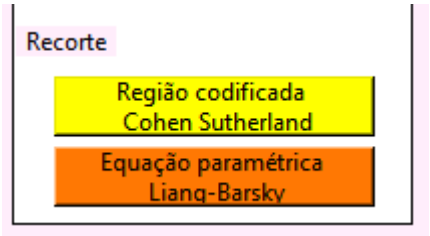
O botão “circunferência bresenham” desenha uma circunferência baseado no algoritmo de Bresenham que seleciona os dois últimos pontos, sendo o penúltimo pro centro e o último pro raio da circunferência, vai desenhando pontos pequenos até formar a linha da circunferência. Para poder aplicar as transformações na circunferência, foi criado uma tag que soma 300 ou -300 à posição [0][0] da circunferência (x do centro) e adiciona os pontos da coordenada na lista de retas, se o [0][0] for maior que 0 soma 300 e se for menor que 0 soma -300. Dessa forma, quando algumas das transformações for executar sua função, vai identificar as circunferências no meio das retas. Como a tela vai de 300 a -300, qualquer valor acima de 300 ou abaixo de -300 vai ser circunferência.



O botão de “região codificada cohen sutherland” recorta uma reta que pode ou não estar dentro de uma janela. Seleciona os quatro últimos pontos, sendo os 2 primeiro para o limite da janela (o primeiro é o x,y mínimo e o segundo, x,y máximo) e os 2 últimos para a reta, desenha uma linha para delimitar a janela e dado o algoritmo de Cohen Sutherland, recorta a reta.

O botão de “equação paramétrica liang barsky” recorta uma reta que não está na janela. Seleciona os quatro últimos pontos, sendo os 2 primeiro para o limite da janela (o

primeiro é o x,y mínimo e o segundo, x,y máximo) e os 2 últimos para a reta, desenha uma linha para delimitar a janela e dado o algoritmo de Liang Barsky, recorta a reta.



O botão de translação vai mudar o objeto de lugar dependendo do valor que for digitado. Ele precisa que tenha pelo menos 1 objeto e 2 pontos desenhados na tela e de dois valores, que serão inseridos pelo campo de input. Os dois valores têm que estar entre uma vírgula e podem ser números reaiz. Ele soma os valores aos pontos originais e desenha esse novo objeto com o “create_line”.

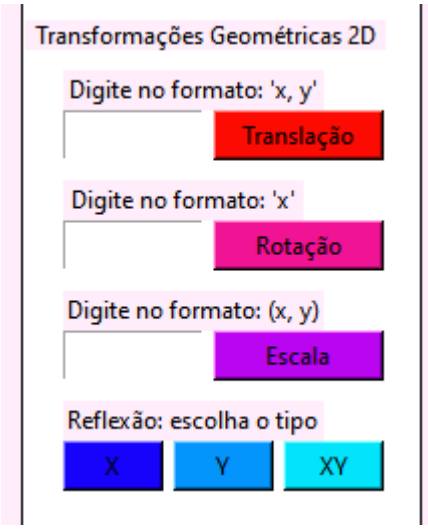
O botão de rotação vai rotacionar o objeto em volta da origem (0,0) dependendo do valor que for digitado. Ele precisa que tenha pelo menos 1 objeto e 2 pontos desenhados na tela e de um valor, que será inserido pelo campo de input. O valor pode ser um número real. Ele faz um cálculo com o valor e os pontos originais e desenha esse novo objeto com o “create_line”.

O botão de escala vai redimensionar o objeto dependendo do valor que for digitado. Ele precisa que tenha pelo menos 1 objeto e 2 pontos desenhados na tela e de dois valores, que serão inseridos pelo campo de input. Os dois valores têm que estar entre uma vírgula e podem ser números reais. Ele multiplica os valores aos pontos originais e desenha esse novo objeto com o “create_line”.

O botão de reflexão X vai refletir o objeto no eixo x. Ele precisa que tenha pelo menos 1 objeto e 2 pontos desenhados na tela. Ele multiplica -1 ao “y” dos pontos originais e desenha esse novo objeto com o “create_line”.

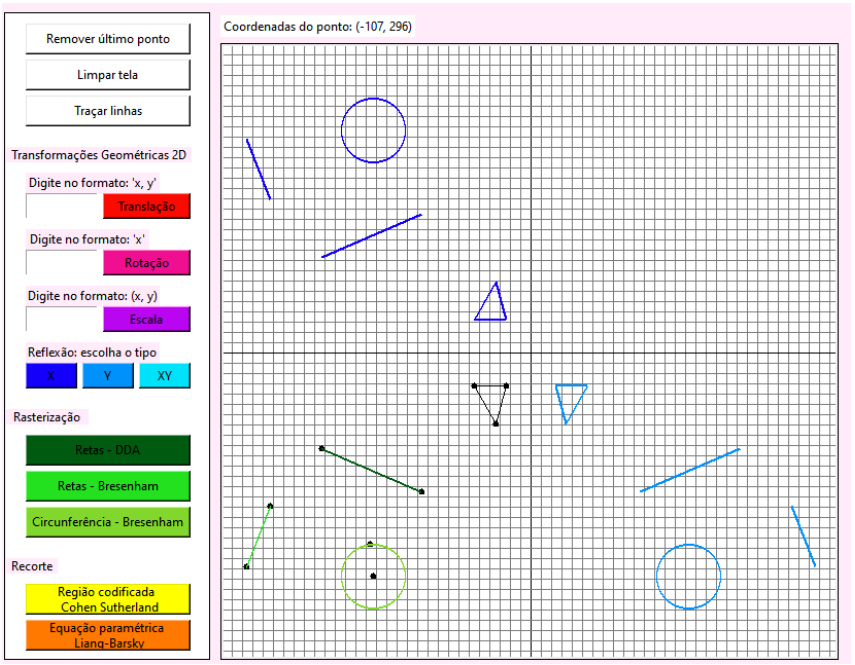
O botão de reflexão Y vai refletir o objeto no eixo y. Ele precisa que tenha pelo menos 1 objeto e 2 pontos desenhados na tela. Ele multiplica -1 ao “x” dos pontos originais e desenha esse novo objeto com o “create_line”.

O botão de reflexão XY vai refletir o objeto no eixo x e no eixo y. Ele precisa que tenha pelo menos 1 objeto e 2 pontos desenhados na tela. Ele multiplica -1 ao “x” e “y” dos pontos originais e desenha esse novo objeto com o “create_line”.

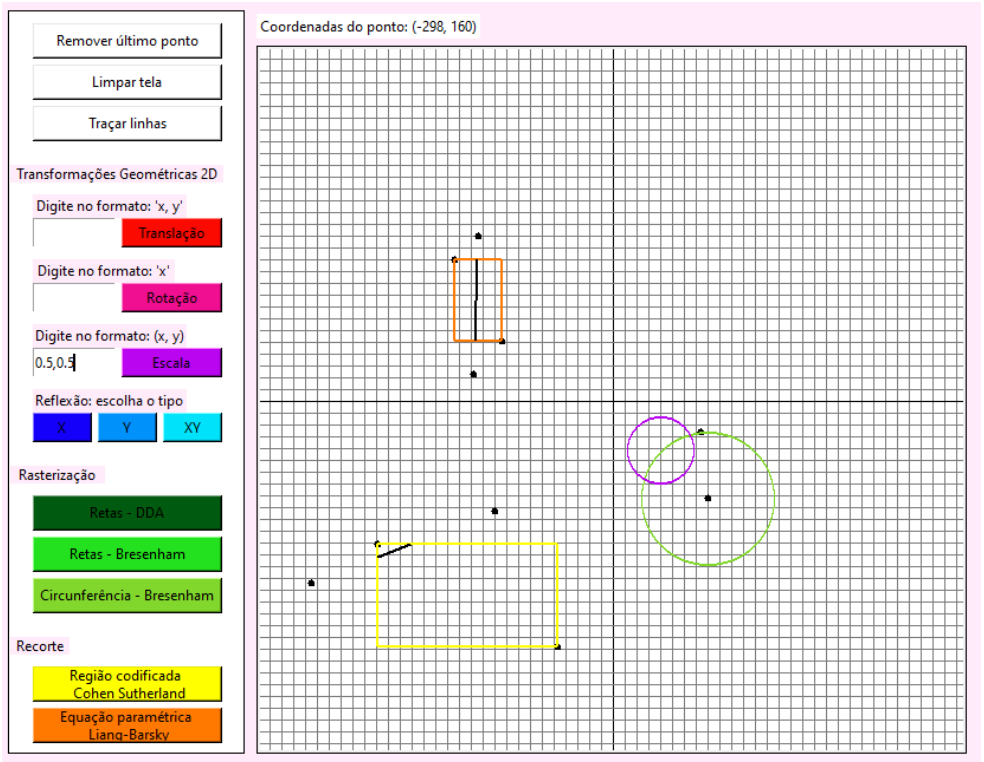


Exemplos

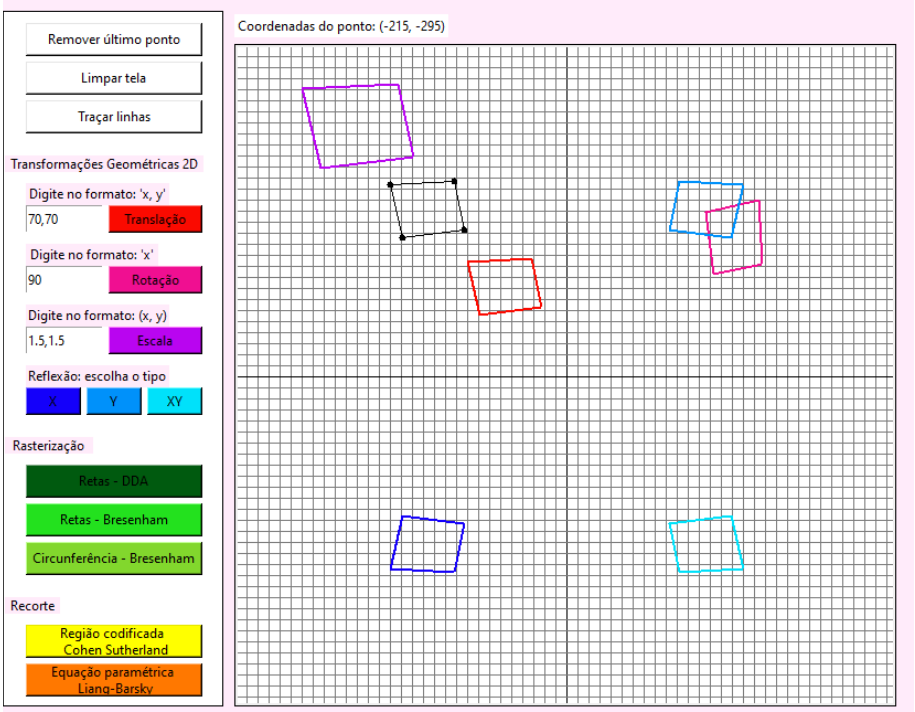
1) 2 pontos para reta dda, 2 pontos para reta bresenham, 2 pontos para circunferência bresenham e 3 pontos para o triângulo. Após isso, foi realizado as funções de reflexão x e reflexão y.



2) 4 pontos para o recorte cohen sutherland, 4 pontos para o recorte liang barsky, 2 pontos para circunferência de bresenham. Após isso escala (0.5,0.5), que transformou apenas a circunferência.

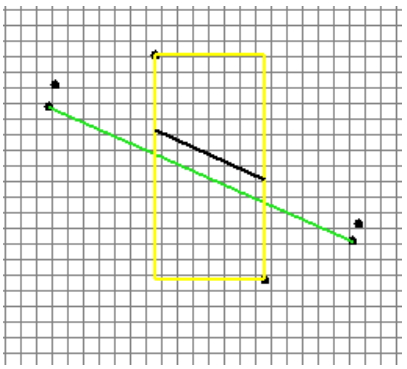


3) 4 pontos para traçar o quadrado. Após isso, translação (70,70), rotação (90), escala (1.5,1.5), reflexão x, reflexão y e reflexão xy.



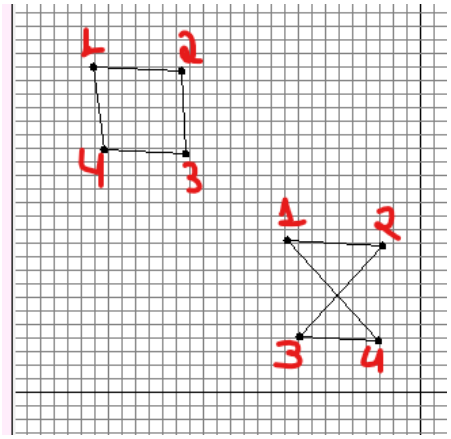
<u>Limitações</u>

- Para as funções de recorte precisam ser selecionados 4 pontos e na ordem certa, sendo os 2 primeiros para janela e os 2 últimos para a reta, então a reta que será recortada não pode ser qualquer uma já desenhada antes ou ter mais de uma.



A reta verde é a reta (bresenham) desenhada anteriormente e a reta preta é a que foi traçada junto com os 2 pontos da janela para fazer o recorte.

- Para traçar as retas e formar um “objeto”, que funciona selecionando alguns pontos e depois clicando no botão de “traçar linha”, a ordem também é importante. Ele vai ligar o último ponto selecionado no primeiro, então acaba que se os dois não estiverem perto ou não ser a reta que vai formar um polígono regular, pode ser que o objeto não saia como esperado.



Esses números são as ordens em que os pontos foram selecionados.

- Falta de mensagens de erro para o usuário, que se caso ele estiver fazendo algo de errado, mostrar pra ele a forma de resolver.

Link para vídeo de explicação:
https://drive.google.com/file/d/1vI3pfWDEzE_kcku6SCNO_8ZVZwrLL_e8/view?usp=sharing