Computação Gráfica - Trabalho Prático 01

Guilherme Dantas C. Fagundes[1]

[1]Instituto de Ciências Exatas e Informática – Pontifícia Universidade Católica de Minas Gerais (PUCMG) Belo Horizonte – MG – Brazil

1. Introdução

Esse trabalho tem como principal objetivo implementar algoritmos de rasterização, transformação e recorte em ambientes 2D que foram apresentados em aula, gernando como objeto final, um executável para operar com as implementações feitas. Ao longo do texto, serão apresentadas as implementações feitas, explicações do código e um manual de uso.

2. Implementações

Os seguintes algoritmos foram implementados:

1. Transformações 2D - Rotação, translação, escala e espelhamento por eixos
2. Rasterização - Retas (Bresenham e DDA) e Circunferências (Bresenham)
3. Recorte - Equação paramétrica (Liang-Barsky) e Regiões codificadas (Cohen-Sutherland)

3. Organização e Funcionamento do Código

O código se inicia no *main.py* que está na raiz do projeto. Ele tem como responsabilidade ser o ponto de entrada da execução e invoca o objeto CanvasApp.

3.1. CanvasApp

O objeto CanvasApp tem funções relacionadas com a renderização do canvas, com os *handlers* vinculados as ações do usuário, tratamento de dados e gerenciamento das variáveis relacionadas as linhas no canvas, pixels para formação da reta e desenho da janela. A princípio ela instancia a janela em que toda a renderização será feita. Um canvas é instanciado que é onde o desenho será feito e onde o usuário irá selecionar os pixels que serão utilizados para gerar as retas.

No topo da janela, através de abas, as opções são dispostas por categorias. O usuário, após selecionar os pixels, escolhe uma opção e o CanvasApp fica responsável por passar os pixels selecionados, os valores preenchidos no campo (caso a operação necessite de dados a mais) e passa o canvas por referência para as funções do algoritmos que são responsáveis por plotar os objetos.

3.2. Pixel e Line

A aplicação utiliza de dois objetos para trafegar as informações entre os métodos. O objeto pixel é utilizado para armazenar valores de (x,y). Já o objeto Line é utilizado para armazenar os pares de pixels que limitam as retas. Ele armazena 2 objetos do tipo pixel, sendo um deles o inicial e o outro o final.

3.3. Algorithms

A classe *Algorithms* fica responsável por implementar todos os algoritmos que que fornecem as funcionalidades. Eles funcionam com base nas informações que cada um necessita. Algoritmos de recorte, recebem as linhas do canvas, os limites da janela e o canvas para que as novas linhas sejam desenhadas. Algoritmos de transformação recebem as linhas, o canvas e os valores de transformação (sejam eles vetores ou valores absolutos).

Todos algoritmos retornam as novas retas geradas e ficam responsáveis por desenhar no canvas. Os algoritmos deletam todas as linhas antes de desenhar as novas. Ou seja, não há um reaproveitamento dos pixels desenhados entre os pixeis originais, novos objetos Line são instanciados e desenhados do zero. Para desenhar no canvas, a função *create_rectangle* é utilizada passando somente 1 pixel por vez, por conta de sua parametrização, a ela recebe $(x, y, x + 1, y + 1)$., sendo x e y os dois pixels que os algoritmos determinam a serem desenhados.

As funções de recorte e transformações, utilizam do algoritmo de rasterização de Bresenham para desenhar as novas retas determinadas pelos algoritmos

4. Manual de Uso

Dentro do projeto disponibilizado, o executável pode ser encontrado na pasta *dist/main/main.exe*. Ao executar a seguinte tela será exibida:

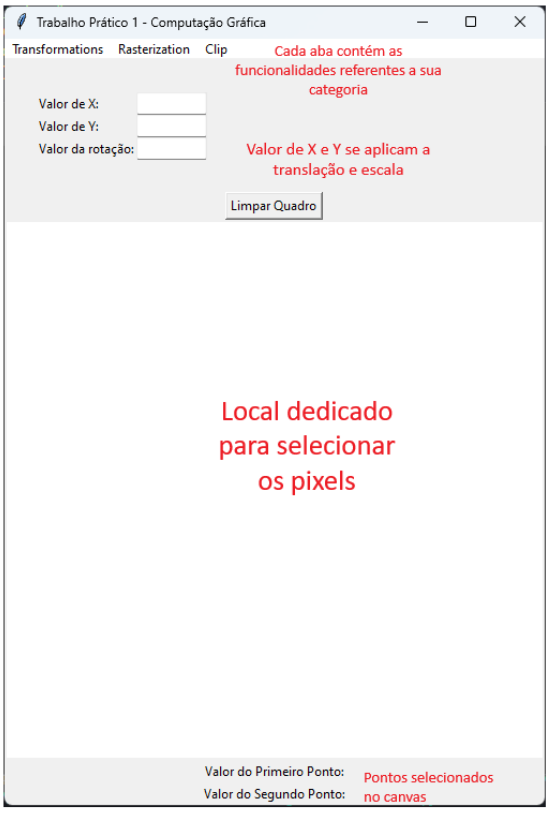

Figure 1. Tela da Aplicação

4.1. Escolha dos Pixels

O usuário pode selecionar os pixels no canvas clicando com o botão esquerdo do mouse. Ao selecionar, na parte inferior, será exibida a coordenada do pixel. Clicando duas vezes um par de pontos será formado e, a partir disso, o usuário pode começar a selecionar cada funcionalidade.

4.2. Transformations

1. Translation - Aplica translação nas retas do canvas, os campos X e Y devem estar preenchidos.
2. Scale - Aplica escala nas retas do canvas, os campos X e Y devem estar preenchidos.
3. Rotate - Aplica rotação nas retas do canvas, o campo valor da rotação deve estar preenchido.
4. Reflection (X/Y/YZ) - Nenhum campo precisa estar preenchido, ao clicar, a transformação é aplicada.

4.3. Rasterization

1. Bresenham - Aplica Bresenham usando como como pixels inicial e final os que foram exibidos na label ”Valor do Primeiro Ponto” e ”Valor do Segundo Ponto”
2. DDA - Aplica DDA usando como como pixels inicial e final os que foram exibidos na label ”Valor do Primeiro Ponto” e ”Valor do Segundo Ponto”
3. Círculo - Aplica Bresenham para desenhar um círculo usando como como pixel central o valor exibido na label ”Valor do Primeiro Ponto” e como raio a distância entre os valores exibidos em ”Valor do Primeiro Ponto” e ”Valor do Segundo Ponto”

4.4. Clip

Para utilizar as funções de recorte, o usuário deve utilizar o botão direito do mouse para selecionar a janela. A janela será usada com base em dois pixels selecionados, mostrando a área da janela em vermelho.

1. Cohen-Sutherland - Aplica Cohen-Sutherland em todas as retas com base na janela desenhada
2. Liang-Barsky - Aplica Liang-Barsky em todas as retas com base na janela desenhada

5. Informações adicionais

Existe a chance do Windows identificar o executável como nocivo ao sistema. Nesse caso, deve-se excluir o arquivo dos monitorados pelo Windows Defender. Caso isso não seja possível, a aplicação não utiliza de libs que estão fora do *Python Standard Library* (no caso, ela utiliza somente a tkinter), por conta disso, executar python main.py na raiz do projeto iniciará o projeto normalmente para uso.

O vídeo de demonstração pode ser encontrado no seguinte link: `https://youtu.be/NNerpzJoSj8?si=AsRaaKpAqU6qlsb0` O código no GitHub pode ser encontrado no seguinte link: