

# CG-TP1-Paint

---

Trabalho prático 1 de Computação Gráfica

Aluno: Luiz Fernando Oliveira Maciel

## Repositório

---

[CG-TP1-Paint](#)

## Vídeo de apresentação

---

[Computação Gráfica - Apresentação Trabalho Prático 1](#)

## Instalação

---

Caso esteja usando Windows, é possível rodar o programa através do arquivo main.exe presente no repositório. Para rodar o código, siga as informações abaixo.

## Dependências

---

- Python 3.11.6 ou superior

- Pip:

O script irá tentar baixar automaticamente o pip. Caso isso não seja possível, tente o seguinte:

- Linux ou MacOS

```
python -m ensurepip --upgrade
```

- Windows

```
py -m ensurepip --upgrade
```

Caso não funcione, baixe e execute no terminal o script no link a seguir: <https://bootstrap.pypa.io/get-pip.py>.

- Requests:

Após instalar o pip, execute no cmd ou terminal:

```
python -m pip install requests
```

- Tkinter:

No Windows, o Tkinter vem baixado por padrão para as versões do Python 3.11.6 ou superior. Caso o código reclame da falta do tkinter ao executar, faça o seguinte:

- MacOS (usando brew)

```
brew install tkinter
```

- Linux (Debian)

```
sudo apt-get install python-tk
```

- Linux (Arch)

```
sudo pacman -S tk
```

- Linux (Fedora)

```
sudo dnf install python3-tkinter
```

- Linux (RHEL, CentOS, Oracle)

```
sudo yum install -y tkinter tk-devel
```

## Executando

---

No diretório contendo os arquivos, execute: `python main.py`

Ou (dependendo do seu sistema operacional/versão do python): `python3 main.py`

# Estrutura de Arquivos

O projeto está estruturado da seguinte forma:

```
CG-TP1-Paint/  
|__ .gitignore  
|__ classes.py  
|__ constants.py  
|__ lib.py  
|__ main.exe  
|__ main.py  
|__ README.md
```

- `classes.py`: Armazena as classes do projeto
- `constants.py`: Armazena o tamanho da janela da aplicação, tamanho dos pixels e tamanho do grid.
- `main.exe`: Executável para Windows (criado utilizando pyinstaller)
- `lib.py`: Imports e afins do projeto.
- `main.py`: Arquivo principal do projeto, contendo a criação da janela do Tkinter e chamada inicial para a classe inicializadora do projeto (classe Paint)
- `README.md`: O arquivo que você está lendo agora :)

## Funcionalidades Principais

- **Grade**: O tamanho dos pixels da grade e a resolução real, como já falado previamente, são definidos no arquivo `constants.py`. As coordenadas reais (ex. nos eventos de click na tela) precisam ser convertidos corretamente para a grade do canvas. Essa conversão é uma simples divisão da coordenada pelo tamanho do pixel. Isso é feito pela função `convert_to_grid` da classe Pixel (será citada na seção abaixo).
- **GUI**: Construída com Tkinter.
- **Desenho de Linhas**: Linhas podem ser criadas com dois algoritmos diferentes, DDA e Bresenham. Após clicar nos botões "Draw Line (DDA)" ou "Draw Line (Bresenham)", selecione dois pontos na tela para desenhar uma linha.
- **Desenho de Círculos**: Os círculos são desenhados utilizando o algoritmo de Bresenham. Após clicar no botão "Draw Circle (Bresenham)", selecione dois pontos na tela (o primeiro será o centro da circunferência) para desenhar um círculo.
- **Desenho de Pixels**: Botão "Draw Pixel" (essa opção vem selecionada por padrão ao abrir a aplicação).
- **Transformações**: Existem as opções de translação, rotação, escala e reflexão. Essas funções operam sobre todas as estruturas desenhadas na tela.
  - **Translação**: Os valores x e y digitados serão somados à cada estrutura;
  - **Rotação**: Digite um ângulo, e o x e y do valor central. As estruturas serão rotacionadas ao redor desse ponto (x, y).
  - **Escala**: As estruturas serão escaladas pelos valores x e y digitados. Para círculos, apenas o valor x será utilizado (multiplica o tamanho do raio)
  - **Reflexão**: Opções de refletir no eixo x, eixo y (ou ambos), e valores x e y do ponto que será considerado como a origem desses eixos.
- **Seleção de Cores**: Utilize o botão "Color Picker" para selecionar a cor da estrutura a ser desenhada.
- **Limpeza da Tela**: Limpa a tela.

## Classes

### Paint

- **Descrição**: Classe principal com a interface do projeto.
- **Atributos**:
  - `master`: O widget raiz da aplicação (classe Tk do Tkinter)
  - `position_label`: Rótulo com a posição do cursor, apresentada no canto inferior direito e atualizada dinamicamente.
  - `structures`: Uma lista com todas as estruturas desenhadas no canvas.
  - `first_click`: A posição do primeiro click do usuário (usado para funcionalidades que usam dois clicks, como a criação de linhas, círculos, e recortes).
  - `clip_mode`: O modo de recorte atual (Cohen-Sutherland ou Liang-Barsky).
  - `draw_mode`: O modo de desenho atual (pixel, linha DDA, linha Bresenham, círculo Bresenham).
  - `color`: A cor atual selecionada para desenho (pode ser alterada pelo usuário com o color picker).
  - `canvas`: Basicamente onde tudo acontece.
- **Métodos**:
  - `__init__(self, master)`: Inicializa a interface gráfica do Paint.
  - `clear_canvas(self)`: Limpa o canvas e reseta os atributos.
  - `create_canvas(self)`: Cria o canvas, junto com os botões da aplicação e os valores padrões para cor e afins.
  - `display_current_position(self, event)`: Atualiza o rótulo `position_label` com as coordenadas do cursor.
  - `change_color(self)`: Abre um seletor de cores para alterar a cor de desenho.
  - `set_draw_mode(self, draw_mode)`: Define o modo de desenho atual.
  - `set_clip_mode(self, clip_mode)`: Define o modo de recorte atual.
  - `draw_temporary_pixel(self, click)`: Desenha um pixel temporário (pontinho azul que aparece para as funcionalidades com dois clicks).
  - `on_click(self, event)`: Lida com os clicks do usuário no canvas, executando a funcionalidade selecionada atualmente (`draw_mode` ou `clip_mode`).
  - `clip(self, start, end, algorithm)`: Realiza o recorte das estruturas em uma área selecionada.
  - Transformações:
    - `create_translate_dialog(self)`: Cria a janela da translação.
    - `translate_structures(self, x, y)`: Translada todas as estruturas desenhadas. Executado no confirmar da janela de translação (assim como as outras transformações descritas a seguir)
    - `create_rotate_dialog(self)`: Cria a janela da rotação.
    - `rotate_structures(self, angle, center_x, center_y)`: Rotaciona todas as estruturas desenhadas.
    - `create_scale_dialog(self)`: Cria a janela da escala.

- `scale_structures(self, scale_x, scale_y)` : Escala todas as estruturas desenhadas.
- `create_reflect_dialog(self)` : Cria a janela de reflexão.
- `reflect_structures(self, reflect_on_x, reflect_on_y, center_x, center_y)` : Reflete as estruturas desenhadas.

## Pixel

- **Descrição:** Representa um único pixel na tela.
- **Atributos:**
  - `x` : Coordenada x do pixel.
  - `y` : Coordenada y do pixel.
  - `color` : Cor do pixel.
  - Argumento `convert` : Booleano indicando se as coordenadas do pixel passadas na inicialização do objeto devem ser convertidas para coordenadas da grade (basicamente se o método `convert_to_grid` será chamado ou não -> quando estamos desenhando pixels, isso é importante já que a coordenada do mouse não está convertida, mas na hora de, por exemplo, desenhar uma linha, as coordenadas dos pixels gerados pelo algoritmo escolhido já estarão convertidos para a grade).
- **Métodos:**
  - `__init__(self, point, color, convert=True)` : Cria um novo pixel.
  - `draw(self, canvas)` : Desenha o pixel no canvas. Esse método é bastante usado pelas outras classes (ex. o `draw` da classe linha chama o `draw` da classe pixel para cada pixel da linha).
  - `convert_to_grid(pos)` : Método estático para converter as coordenadas reais da tela para coordenadas da grade do canvas (valores especificados no arquivo `constants.py`).
  - `translate(self, x, y)` : Translada o pixel somando os valores de x e y passados às coordenadas dele.
  - `rotate(self, angle, center)` : Rotaciona o pixel em torno de um ponto central.
  - `scale(self, x, y)` : Escala a posição do pixel (simples multiplicação dos valores passados às coordenadas).
  - `reflect(self, reflect_on_x, reflect_on_y, center_x, center_y)` : Reflete o pixel em torno do(s) eixo(s) especificado(s). Os valores de `center_x` e `center_y` representam o ponto de origem dos eixos
  - `clip(self, start, end, algorithm)` : Realiza o recorte verificando se o pixel está dentro da área delimitada (Point Clipping).

## Structure

- **Descrição:** Interface para as outras estruturas (linhas e círculos).
- **Atributos:**
  - `pixels` : Lista de pixels que compõem a estrutura.
  - `color` : Cor da estrutura.
  - `clipped` : Indica se a estrutura foi recortada ou não.
- **Métodos:**
  - `__init__(self, pixels, color)` : Cria uma nova estrutura recebendo uma lista de pixels e uma cor.
  - `add_point(self, point)` : Adiciona um novo ponto à estrutura.
  - `clear_pixels(self)` : Remove o array de pixels da estrutura.
  - `draw(self, canvas)` : Desenha a estrutura no canvas.
  - `translate(self, x, y)` : Translada a estrutura em x e y.
  - `rotate(self, angle, center)` : Rotaciona a estrutura em torno de um ponto central.
  - `scale(self, x, y)` : Escala a estrutura horizontalmente e verticalmente.
  - `reflect(self, reflect_on_x, reflect_on_y, center_x, center_y)` : Reflete a estrutura em torno do eixo especificado.
  - `clip(self, start, end, algorithm)` : Realiza o recorte da estrutura de acordo com a área delimitada (também Point Clipping).

## Line

- **Descrição:** Representa uma linha desenhada entre dois pontos (ou feita com DDA, ou com Bresenham).
- **Atributos:**
  - `start` : Posição do início da linha (objeto Pixel).
  - `end` : Posição do final da linha.
  - `color` : Cor da linha.
  - `pixels` : Pixels da linha (após ser gerada com um dos algoritmos)
  - `line_type` : Algoritmo utilizado para desenhar a linha (DDA ou Bresenham).
- **Métodos:**
  - `__init__(self, start, end, color, line_type="line_dda")` : Inicializa uma nova linha com um ponto inicial e final, cor e tipo da linha.
  - `get_line(self)` : Encontra os pixels que formam a linha (de acordo com o atributo `line_type`).
  - `get_line_dda(self)` : Utiliza o algoritmo DDA para traçar a linha.
  - `get_line_bresenham(self)` : Utiliza o algoritmo de Bresenham para traçar a linha.
  - `translate` : Translada os pontos de início e fim da linha, e depois gera a linha novamente.
  - `rotate` : Rotaciona a linha em torno de um ponto central (rotaciona ambos os pontos e gera de novo a linha).
  - `scale` : Escala a linha horizontalmente e verticalmente (simula a translação da linha para a origem diminuindo as coordenadas do ponto inicial no ponto final, escala a posição do ponto final, inverte a translação e depois gera a linha de novo com base no novo ponto final).
  - `reflect` : Reflete a linha em torno de um eixo especificado (reflete os pontos de início e fim e gera de novo a linha).
  - `clip` : Recorta a linha a partir da área entre dois pontos utilizando ou Cohen-Sutherland (método `clip_cohen`) ou Liang-Barsky (método `clip_liang`). Usei as implementações do livro (Computer Graphics C version)
  - `clip_cohen` : Implementação do método de Cohen-Sutherland
  - `cohen_get_code` : Calcula o código do vértice
  - `clip_liang` : Implementação do método de Liang-Barsky
  - `liang_clip_test` : Calcula se a linha deve ser rejeitada ou se os parâmetros de interseção devem ser ajustados

## Circle

- **Descrição:** Representa um círculo através de um centro e um raio.
- **Atributos:**

- `center` : O centro do círculo.
- `radius` : O raio do círculo.
- `color` : A cor do círculo.
- `pixels` : A lista com os pixels do círculo (após serem gerados com o `get_circle`).
- `clipped` : Indica se o círculo foi recortado (algumas mudanças são feitas nas transformações caso isso seja verdadeiro - não era um requisito do projeto mas ficou mais bem apresentável assim enquanto não temos um algoritmo de recorte para círculos/polígonos).

- **Métodos:**

- `__init__(self, center, radius, color)` : Inicializa um novo círculo com centro, raio e cor especificados.
- `get_circle(self)` : Calcula os pixels que formam o círculo.
- `plot_points(self, xc, yc, x, y)` : Adiciona os pontos do círculo à estrutura.
- Métodos herdados de `Structure` : Métodos para manipular o círculo, como translação, rotação, escala e reflexão.
- Adendo sobre as transformações :
  - Caso o círculo tenha sido recortado, os métodos apenas operam sobre a lista de pixels do círculo. A apresentação dessa forma fica melhor, visto que anteriormente o recorte apenas era desfeito quando um desses métodos era chamado (já que eles recalculam o círculo ao final).
- `translate` : Translada o ponto central do círculo.
- `rotate` : Rotaciona o ponto central do círculo.
- `scale` : Multiplica o tamanho do raio do círculo (apenas o `x` é usado, porém o método recebe tanto `x` quanto `y` para manter os parâmetros iguais para todas as classes que estendem `Structure`).
- `reflect` : Reflete o centro do círculo.