Documentação do projeto

Equipe:

- Cristiane Kelly Amaro da Silva
- Pedro Melquisedeque Cardoso De Lima

Especificações de funções e Primitivas utilizadas

a) Set Pixel

Funções responsáveis por desenhar **pixels individuais**, base de toda a rasterização:

- `set_pixel(x, y, color)`

b) Primitivas de Rasterização (Linha, Círculo e Elipse)

Funções que desenham formas básicas usando algoritmos clássicos:

- **Linha**
  - `draw_line(x0, y0, x1, y1, color)` → utilizada para fazer o tiro
    ➜ Algoritmo de **Bresenham**
- **Círculo**
  - `draw_circle(cx, cy, r, color)` → utilizada para fazer o meteoro
    ➜ Algoritmo do **Ponto Médio do Círculo**

- **Elipse**
  - `draw_ellipse(cx, cy, rx, ry, color)` → utilizada na introdução do jogo
    ➜ Algoritmo do **Ponto Médio da Elipse**

c) Preenchimento de Regiões (Flood Fill/Boundary Fill e Scanline)

Algoritmos de preenchimento por conectividade e por varredura:

- **Flood Fill**
  - `flood_fill(x, y, target, replacement)` → utilizada principalmente para o preenchimento do meteoro

- **Boundary Fill**
  - `boundary_fill(x, y, fill_color, boundary_color)` → utilizado na viewport
- **Scanline**

- `scanline_fill(points, fill_color)` → utilizado para preenchimento da nave, com aplicação de textura procedural

d) Transformações Geométricas (Rotação, Translação e Escala)

Funções que alteram posição, orientação ou escala:

- **Rotação**
  - `rotate(angle)` → utilizado na introdução do jogo para girar a nave
- **Translação (implícita no código)**
  - Aplicada em:
    - `translation()` → translação matricial - função principal
    - `draw_ship()` → utiliza a função translation() para realizar o movimento da nave
- *(Escala não está explícita como função, mas o zoom simula escala visual)*

e) Animação 2D

Funções que usam **tempo**, **frames** ou **movimento contínuo**:

- `intro_logo()`
- `draw_logo()`
- `scrolling_story()`
- Animação do fogo da nave (uso de `pygame.time.get_ticks()`)
- Loop principal (`while True` com `clock.tick(FPS)`)

f) Janela e Viewport

Funções e lógicas que tratam **recorte visual** e **mapeamento de coordenadas**:

- Uso de:

    - `VIEWPORT`
    - `ZOOM_VIEWPORT`
- `draw_zoom_system()`
- Função interna:
    - `map_coords(world_x, world_y)`

g) Recorte de Cohen-Sutherland (Clipping)

Implementação completa do algoritmo de recorte de linhas:

- `get_outcode(x, y, rect)`
- `draw_line_clipped(x0, y0, x1, y1, rect, color)`

h) Mapeamento de Textura

Textura **procedural** (não por imagem):

- Parte interna de:
    - `scanline_fill(points, fill_color)`

Trecho específico:

```
if (x // 1 + y // 1) % 2 == 0:
    set_pixel(x, y, (0, 0, 255))
else:
    set_pixel(x, y, (0, 0, 180))
```

➜ **Textura procedural discreta**

i) Input (Teclado e/ou Mouse)

Funções e trechos que tratam entrada do usuário:

- `shoot()`
- `menu()`
- `instructions()`
- `scrolling_story()` (pular com tecla)
- Loop principal:
    - `pygame.KEYDOWN`
    - `pygame.key.get_pressed()`

j) *Menus e Interações Gráficas Avançadas são adicionais

- `menu()`
- `instructions()`
- `game_over()`