Resta Um (Versão C com Raylib) — Parte 2

Autor: Gabriel J Santos

Inicialização do Raylib

```
const Vector2 screenSize = {800, 800};
InitWindow(screenSize.x, screenSize.y, "Resta Um");
SetTargetFPS(60);
```

- `Vector2 screenSize` → tamanho da tela (800×800 pixels).
- `InitWindow()` → cria a janela do jogo.
- `SetTargetFPS(60)` → fixa o **frame rate** em 60 FPS para animações suaves.

Loop Principal do Jogo

```
while (!WindowShouldClose()) {
    // Update
    // Draw
}
```

- **Loop principal**: roda até o jogador fechar a janela ou apertar ESC.
- Divide o código em duas partes:
    1. **Update** → processamento de lógica, entrada do usuário, seleção de peças;
    2. **Draw** → renderização gráfica das peças e elementos do jogo.

Captura da Posição do Mouse

```
Vector2 MousePos;
MousePos.x = GetMouseX();
MousePos.y = GetMouseY();
selection Selection = get_selected(MousePos, R, spacing, screenSize, parts, 32);
```

- `GetMouseX()` e `GetMouseY()` → obtêm posição atual do mouse.
- `get_selected()` → retorna qual peça está sendo selecionada (ver Parte 1).

O valor de `R` define o **raio de seleção** (clicável).

Desenhando o Jogo

Começo do desenho

```
BeginDrawing();
ClearBackground(RAYWHITE);
```

- `BeginDrawing()` → inicia a renderização de um frame.
- `ClearBackground(RAYWHITE)` → limpa a tela para branco.

Título centralizado

```
char* title = " - - - Resta Um - - -";
int text_width = MeasureText(title, fontSize);
DrawText(title, translate_cartesian_to_screen(0, screenSize.x) - (text_width/2),
10, fontSize, LIGHTGRAY);
```

- `MeasureText()` → calcula largura do texto em pixels.
- Subtrair `text_width/2` → **centraliza horizontalmente**.

Destaque da peça selecionada

```
if (Selection.state == true) {
    DrawCircle(scaled_to_screen(Selection.Xcartesian, screenSize.x, spacing),
               scaled_to_screen(Selection.Ycartesian, screenSize.y, spacing),
               R+5, RED);
}
```

- Desenha um **círculo vermelho** maior em volta da peça selecionada.
- `scaled_to_screen()` → posiciona corretamente na tela.

Dá feedback visual do que o jogador está prestes a clicar.

Desenho das peças do tabuleiro

```
for (int i = 0; i < 32; i++) {
    int x = scaled_to_screen(parts[i].posX, screenSize.x, spacing);
    int y = scaled_to_screen(parts[i].posY, screenSize.y, spacing);
    if (parts[i].state == true) {
        DrawCircle(x, y, R, LIGHTGRAY);
    }
}
```

- Itera por todas as peças.
- Se `state == true`, desenha a peça como um **círculo cinza**.
- As peças removidas (`state == false`) não são desenhadas.

◇ Finalizando o frame

```
EndDrawing();
```

- Conclui a renderização do frame atual.
- Raylib automaticamente atualiza a tela com tudo que foi desenhado entre `BeginDrawing()` e `EndDrawing()`.

❌ Encerramento do Jogo

```
CloseWindow();
```

- Libera os recursos do Raylib e fecha a janela.

🧩 Fluxo Geral do Jogo

1. **Inicialização**
    - Janela, tabuleiro, FPS.
2. **Loop Principal**
    - Atualização: captura mouse, seleciona peça.
    - Desenho: limpa tela, desenha título, destaques, peças.
3. **Finalização**
    - Fecha janela ao sair do loop.

🔗 Resumo Visual do Loop

```
+---------------------------+
| while (WindowOpen)        |
|  +--------------------+   |
|  | Update             |   |
|  |  - Mouse pos       |   |
|  |  - Selection       |   |
|  +--------------------+   |
```

```
  |  | Draw                |  |
  |  |  - Background       |  |
  |  |  - Title            |  |
  |  |  - Highlight circle |  |
  |  |  - Pieces           |  |
  |  +---------------------+  |
  +---------------------------+
  CloseWindow()
```

Pontos Importantes

Conceito	Explicação
`BeginDrawing()/EndDrawing()`	Delimita o frame de renderização
`DrawCircle()`	Desenha peças e destaques
`MeasureText()`	Permite centralizar textos
`Vector2`	Estrutura para coordenadas 2D
`get_selected()`	Detecta interação do mouse
`R`	Define raio de clique e destaque
`spacing`	Define distância entre peças
`screenSize`	Determina escala e centro do tabuleiro