Resta Um (Versão C com Raylib) — Parte 1

Autor: Gabriel J Santos

Objetivo do Código

O projeto implementa o clássico **"Resta Um"**, onde o jogador remove peças do tabuleiro até sobrar apenas uma. A versão digital usa **C** e **Raylib**.

O código pode ser dividido em três blocos principais:

1. Estruturas de dados e tabuleiro;
2. Funções auxiliares (matemática e lógica);
3. Interação com o mouse e seleção de peças (preparação para a interface).

Esta primeira parte foca nos **dois primeiros blocos**.

Estrutura de Dados — `struct Part`

```c
struct Part {
    int posX;
    int posY;
    bool state;
};
```

- `posX` **e** `posY`: posição no **plano cartesiano**, centralizado no meio do tabuleiro;
- `state`: indica se a peça está **ativa** (`true`) ou removida (`false`).

O vetor principal:

```c
struct Part parts[32];
struct Part empty_place[32];
```

- `parts[32]` → todas as peças do tabuleiro;
- `empty_place[32]` → posições vazias (inicialmente não usadas, mas útil para movimentos futuros).

Observação: o tabuleiro 7x7 possui 49 posições, mas apenas 32 são válidas para o "Resta Um".

Geração do Tabuleiro — `parts_generator()`

```c
void parts_generator(struct Part parts[], int size, bool state) {
    int index = 0;
```

```
    for (int i = -3; i <= 3; i++) {
        for (int j = -3; j <= 3; j++) {
            if (!(((i == 0) && (j == 0)) || ((abs(i) > 1) && (abs(j) > 1)))) {
                if (index == size) return;
                parts[index].posX = i;
                parts[index].posY = j;
                parts[index].state = true;
                index++;
            }
        }
    }
}
```

🔍 Explicação detalhada:

1. Laços `for` percorrem de **-3 a 3**, criando uma grade de 7x7 coordenadas cartesianas.
2. O `if` interno elimina posições inválidas:
   - `(i == 0 && j == 0)` → o centro começa vazio;
   - `(abs(i) > 1 && abs(j) > 1)` → elimina cantos diagonais externos.
3. `index` controla quantas peças já foram adicionadas.
4. `parts[index].state = true` → inicializa cada peça como ativa.

🧠 Exemplo visual do tabuleiro (coordenadas cartesianas):

```
(-3,3) (-2,3) (-1,3) (0,3) (1,3) (2,3) (3,3)
...
(-3,0) (-2,0) (-1,0) (0,0) (1,0) (2,0) (3,0)
...
```

Somente a **cruz central** e os quadrados próximos são válidos.

🖨️ Impressão de Peças — `parts_print_console()`

```
void parts_print_cosole(struct Part parts[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d , %d, %d, %d\n", i, parts[i].posX, parts[i].posY,
parts[i].state);
    }
}
```

- Mostra no console o **índice, coordenadas e estado** de cada peça.
- Útil para **depuração** e verificar se o `parts_generator()` funcionou corretamente.

Exemplo de saída:

```
0 , -1, 0, 1
1 , 0, -2, 1
...
```

Conversão de Coordenadas

◇ Plano cartesiano → tela

```c
int translate_cartesian_to_screen(int value, int screen_size) {
    return value + (screen_size / 2);
}
```

- Centraliza o valor cartesiano na tela.
- Ex.: se `value = 0` e `screen_size = 800`, retorna `400` → centro da tela.

◇ Tela → plano cartesiano

```c
int translate_screen_to_cartesian(int value, int screen_size) {
    return value - (screen_size / 2);
}
```

- Inverte a operação para obter coordenadas lógicas do mouse.

◇ Escalonamento com espaçamento

```c
int scaled_to_screen(int value, int screen_size, int spacing) {
    return translate_cartesian_to_screen(value * spacing, screen_size);
}
```

- Multiplica a coordenada lógica pelo **espaçamento entre peças** (ex.: 95 px).
- Converte para **coordenadas de tela**.

Inverso

```c
int scaled_to_screen_reverse(int value, int screen_size, int spacing) {
    int centered = translate_screen_to_cartesian(value, screen_size);
    return centered / spacing;
}
```

- Converte do pixel da tela para coordenada lógica do tabuleiro.

Resumo visual:

Valor Lógico	Escalonamento	Valor na Tela
-2	-2 * 95	400 - 190 = 210
0	0	400
2	2 * 95	400 + 190 = 590

📏 Cálculo de Distância — `get_distance()`

```
int get_distance(int x, int y, int x1, int y1) {
    int dx = x1 - x;
    int dy = y1 - y;
    return sqrt((dx * dx) + (dy * dy));
}
```

- Retorna a distância entre dois pontos `(x, y)` e `(x1, y1)`
- Usado para detectar se o **clique do mouse** está dentro do raio de uma peça.

🎯 Estrutura de Seleção — `struct selection`

```
struct selection {
    int Xcartesian;
    int Ycartesian;
    int X;
    int Y;
    bool state;
};
```

- `Xcartesian` e `Ycartesian` → coordenadas lógicas;
- `X` e `Y` → coordenadas da tela;
- `state` → `true` se alguma peça foi selecionada.

🖱️ Seleção de Peça — `get_selected()`

```
selection get_selected(Vector2 mouse, int R, int spacing, Vector2 screen, struct
Part parts[], int size) {
    selection selected;
    for (int i = 0; i < size; i++) {
        if (parts[i].state == true) {
```

```
            if (get_distance(
                    scaled_to_screen(parts[i].posX, screen.x, spacing),
                    scaled_to_screen(parts[i].posY, screen.y, spacing),
                    mouse.x, mouse.y) < R) {
                selected.Xcartesian = parts[i].posX;
                selected.Ycartesian = parts[i].posY;
                selected.X = scaled_to_screen(parts[i].posX, screen.x, spacing);
                selected.Y = scaled_to_screen(parts[i].posY, screen.y, spacing);
                selected.state = true;
            }
        } else {
            selected.state = false;
        }
    }
    return selected;
}
```

🔍 Explicação:

1. Percorre todas as peças ativas (`state == true`);
2. Calcula a distância do mouse até cada peça;
3. Se estiver dentro do **raio R**, retorna a peça selecionada;
4. Caso nenhuma peça seja selecionada, `state = false`.

🧾 Resumo Expandido

Função/Struct	Função	Importância
`struct Part`	Armazena peça	Base do tabuleiro
`parts_generator()`	Cria peças	Inicializa o tabuleiro
`parts_print_console()`	Imprime peças	Debug/Visualização
`translate_cartesian_to_screen()`	Converte coordenadas	Geometria da tela
`scaled_to_screen()`	Aplica espaçamento	Posição exata das peças
`get_distance()`	Distância entre pontos	Detecção de clique
`struct selection`	Dados de peça selecionada	Interação com o usuário
`get_selected()`	Seleciona peça clicada	Lógica de entrada