Universidade Federal de Uberlândia

Francielle Roberta Ferreira - 11811EEL042
Lucas Eduardo Oliveira Rosa - 11811EEL016

TETRIS

Métodos e Técnicas de Programação
Comentado

Uberlândia MG
2018

```
#include <stdio.h>
#include <stdlib.h>
#include <conio2.h>
#include <conio.h>
#include <windows.h>
#include <time.h>

int perde[6][5][5] =
{
    {   {1,1,1,1},
        {1,0,0,1},
        {1,1,1,1},
        {1,0,0,0},
        {1,0,0,0}
    },// um
    {   {1,1,1,1},
        {1,0,0,0},
        {1,1,1,0},
        {1,0,0,0},
        {1,1,1,1}
    },//dois
    {   {1,1,1,1},
        {1,0,0,1},
        {1,1,1,1},
        {1,0,1,0},
        {1,0,1,}
    },//tres
    {   {1,1,1,0},
        {1,0,0,1},
        {1,0,0,1},
        {1,0,0,1},
        {1,1,1,0}
    },
    {   {1,1,1,1},
        {1,0,0,0},
        {1,1,1,0},
        {1,0,0,0},
        {1,1,1,1}
    },
    {   {1,0,0,1},
        {1,0,0,1},
        {1,0,0,1},
        {1,0,0,1},
        {1,1,1,1}
    }
};
```

```
struct PC
{
    int PECA[4][4][4]; //armazenar peça atual
};

struct PC P;
int T[24][10];

// Definicao da peca em L e as suas possiveis 4 rotacoes
int L[4][4][4] =
{
    {   {0,0,0,0},
        {1,0,0,0},
        {1,0,0,0},
        {1,1,0,0}
    },// um
    {   {0,0,0,0},
        {0,0,0,0},
        {1,1,1,0},
        {1,0,0,0}
    },//dois
    {   {0,0,0,0},
        {1,1,0,0},
        {0,1,0,0},
        {0,1,0,0}
    },//tres
    {   {0,0,0,0},
        {0,0,0,0},
        {1,0,0,0},
        {1,1,1,0}
    }
};//quatro

// Definicao da peca em Li e as suas possiveis 4 rotacoes
int Li[4][4][4] =
{
    {   {0,0,0,0},
        {0,1,0,0},
        {0,1,0,0},
        {1,1,0,0}
    },// um
    {   {0,0,0,0},
        {0,0,0,0},
        {1,0,0,0},
        {1,1,1,0}
    },//dois
    {   {0,0,0,0},
        {1,1,0,0},
        {1,0,0,0},
        {1,0,0,0}
    },//tres
    {   {0,0,0,0},
        {0,0,0,0},
        {1,1,1,0},
        {0,0,1,0}
    }
};//quatro

// Definicao da peca em R e as suas possiveis 4 rotacoes
int R[4][4][4] =
{
    {   {1,0,0,0},
        {1,0,0,0},
        {1,0,0,0},
        {1,0,0,0}
    },// um
    {   {0,0,0,0},
        {0,0,0,0},
        {0,0,0,0},
        {1,1,1,1}
    },//dois
    {   {1,0,0,0},
        {1,0,0,0},
        {1,0,0,0},
        {1,0,0,0}
    },//tres
    {   {0,0,0,0},
        {0,0,0,0},
        {0,0,0,0},
        {1,1,1,1}
    }
};//quatro

// Definicao da peca em Q e as suas possiveis 4 rotacoes
int Q[4][4][4] =
{
    {   {0,0,0,0},
        {0,0,0,0},
        {1,1,0,0},
        {1,1,0,0}
    },// um
    {   {0,0,0,0},
        {0,0,0,0},
        {1,1,0,0},
        {1,1,0,0}
    },//dois
    {   {0,0,0,0},
        {0,0,0,0},
        {1,1,0,0},
        {1,1,0,0}
    },//tres
    {   {0,0,0,0},
        {0,0,0,0},
        {1,1,0,0},
        {1,1,0,0}
    }
};//quatro

// Definicao da peca em T e as suas possiveis 4 rotacoes
int Tr[4][4][4] =
{
    {   {0,0,0,0},
        {0,0,0,0},
        {1,1,1,0},
        {0,1,0,0}
    },// um
    {   {0,0,0,0},
        {0,1,0,0},
        {1,1,0,0},
        {0,1,0,0}
    },//dois
    {   {0,0,0,0},
        {0,0,0,0},
        {0,1,0,0},
        {1,1,1,0}
    },//tres
    {   {0,0,0,0},
        {1,0,0,0},
        {1,1,0,0},
        {1,0,0,0}
    }
};//quatro

// Definicao da peca em Z e as suas possiveis 4 rotacoes
int Z[4][4][4] =
{
    {   {0,0,0,0},
        {0,1,0,0},
        {1,1,0,0},
        {1,0,0,0}
    },//zero
    {   {0,0,0,0},
        {0,0,0,0},
        {1,1,0,0},
        {0,1,1,0}
    },//um
    {   {0,0,0,0},
        {0,1,0,0},
        {1,1,0,0},
        {1,0,0,0}
    },//dois
    {   {0,0,0,0},
        {0,0,0,0},
        {1,1,0,0},
        {0,1,1,0}
    }
};//tres

// Definicao da peca em Zi e as suas possiveis 4 rotacoes
int Zi[4][4][4] =
{
    {   {0,0,0,0},
        {1,0,0,0},
        {1,1,0,0},
        {0,1,0,0}
    },// um
    {   {0,0,0,0},
        {0,0,0,0},
        {0,1,1,0},
        {1,1,0,0}
    },//dois
    {   {0,0,0,0},
        {1,0,0,0},
        {1,1,0,0},
        {0,1,0,0}
    },//tres
    {   {0,0,0,0},
        {0,0,0,0},
        {0,1,1,0},
        {1,1,0,0}
    }
};//quatro
```

```
void sorteia_peca(int ale, int aleB, int qnt[]) //sorteia peça que
vai cair e a proxima
{
    int i, j, k;

    // numero aleatorio que vem da funçao jogar
    if(ale == 0)
    {
        for(i = 0; i < 4; i++)
        {
            for(j = 0; j < 4; j++)
            {
                for(k = 0; k < 4; k++)
                {
                    P.PECA[i][j][k] = L[i][j][k];
                    qnt[0] = 2;
                    qnt[1] = 3;
                }
            }
        }
    }

    else if(ale == 1)
    {
        for(i = 0; i < 4; i++)
        {
            for(j = 0; j < 4; j++)
            {
                for(k = 0; k < 4; k++)
                {
                    P.PECA[i][j][k] = Li[i][j][k];
                    qnt[0] = 2;
                    qnt[1] = 3;
                }
            }
        }
    }

    else if(ale == 2)
    {
        for(i = 0; i < 4; i++)
        {
            for(j = 0; j < 4; j++)
            {
                for(k = 0; k < 4; k++)
                {
```

```
                P.PECA[i][j][k] = R[i][j][k];
                qnt[0] = 1;
                qnt[1] = 4;
            }
        }
    }
}

else if(ale == 3)
{
    for(i = 0; i < 4; i++)
    {
        for(j = 0; j < 4; j++)
        {
            for(k = 0; k < 4; k++)
            {
                P.PECA[i][j][k] = Q[i][j][k];
                qnt[0] = 2;
                qnt[1] = 2;
            }
        }
    }
}

else if(ale == 4)
{
    for(i = 0; i < 4; i++)
    {
        for(j = 0; j < 4; j++)
        {
            for(k = 0; k < 4; k++)
            {
                P.PECA[i][j][k] = Z[i][j][k];
                qnt[0] = 2;
                qnt[1] = 3;
            }
        }
    }
}

else if(ale == 5)
{
    for(i = 0; i < 4; i++)
    {
        for(j = 0; j < 4; j++)
        {
            for(k = 0; k < 4; k++)
            {
                P.PECA[i][j][k] = Zi[i][j][k];
                qnt[0] = 2;
                qnt[1] = 3;
            }
        }
    }
}

else if(ale == 6)
{
    for(i = 0; i < 4; i++)
    {
        for(j = 0; j < 4; j++)
        {
            for(k = 0; k < 4; k++)
            {
                P.PECA[i][j][k] = Tr[i][j][k];
                qnt[0] = 3;
                qnt[1] = 2;
            }
        }
    }
}

// printa proxima peça
if(aleB == 0)
{
    for(i = 0; i < 4; i++)
    {
        k = 0;
        for(j = 0; j < 4; j++)
        {
            gotoxy(61+j+k, 28+i);
            textcolor(LIGHTGREEN);
            if(L[1][i][j] == 1)
                printf("\xFE");
            else
            {
                textcolor(DARKGRAY);
                printf("\xB0");
                textcolor(WHITE);
            }
            k++;
        }
        textcolor(WHITE);
    }
}

else if(aleB == 1)
{
    for(i = 0; i < 4; i++)
    {
        k = 0;
        for(j = 0; j < 4; j++)
        {
            gotoxy(61+j+k, 28+i);
            textcolor(LIGHTGREEN);
            if(Li[1][i][j] == 1)
                printf("\xFE");
            else
            {
                textcolor(DARKGRAY);
                printf("\xB0");
                textcolor(WHITE);
            }
            k++;
        }
        textcolor(WHITE);
    }
}

else if(aleB == 2)
{
    for(i = 0; i < 4; i++)
    {
        k = 0;
        for(j = 0; j < 4; j++)
        {
            gotoxy(61+j+k, 28+i);
            textcolor(LIGHTGREEN);
            if(R[1][i][j] == 1)
                printf("\xFE");
            else
            {
                textcolor(DARKGRAY);
                printf("\xB0");
                textcolor(WHITE);
            }
            k++;
        }
```

```
            textcolor(WHITE);
        }
    }

    else if(aleB == 3)
    {
        for(i = 0; i < 4; i++)
        {
            k = 0;
            for(j = 0; j < 4; j++)
            {
                gotoxy(61+j+k, 28+i);
                textcolor(LIGHTGREEN);
                if(Q[1][i][j] == 1)
                    printf("\xFE");
                else
                {
                    textcolor(DARKGRAY);
                    printf("\xB0");
                    textcolor(WHITE);
                }
                k++;
            }
            textcolor(WHITE);
        }
    }

    else if(aleB == 4)
    {
        for(i = 0; i < 4; i++)
        {
            k = 0;
            for(j = 0; j < 4; j++)
            {
                gotoxy(61+j+k, 28+i);
                textcolor(LIGHTGREEN);
                if(Z[1][i][j] == 1)
                    printf("\xFE");
                else
                {
                    textcolor(DARKGRAY);
                    printf("\xB0");
                    textcolor(WHITE);
                }
                k++;
            }
            textcolor(WHITE);
        }
    }

    else if(aleB == 5)
    {
        for(i = 0; i < 4; i++)
        {
            k = 0;
            for(j = 0; j < 4; j++)
            {
                gotoxy(61+j+k, 28+i);
                textcolor(LIGHTGREEN);
                if(Zi[1][i][j] == 1)
                    printf("\xFE");
                else
                {
                    textcolor(DARKGRAY);
                    printf("\xB0");
                    textcolor(WHITE);
                }
                k++;
            }
            textcolor(WHITE);
        }
    }

    else if(aleB == 6)
    {
        for(i = 0; i < 4; i++)
        {
            k = 0;
            for(j = 0; j < 4; j++)
            {
                gotoxy(61+j+k, 28+i);
                textcolor(LIGHTGREEN);
                if(Tr[1][i][j] == 1)
                    printf("\xFE");
                else
                {
                    textcolor(DARKGRAY);
                    printf("\xB0");
                    textcolor(WHITE);
                }
                k++;
            }
            textcolor(WHITE);
        }
    }
}

void limpa_text(int xi, int yi, int xf, int yf)
{
    int i, j;
    for(i = yi; i < yf; i++)
    {
        for(j = xi; j < xf; j++)
        {
            gotoxy(j,i);
            printf(" ");
        }
    }
}

/* A funcao zeraMatriz zera todos os elementos da
variavel matriz que possui L linhas e C colunas*/

void zeraMatriz(int L,int C)
{
    int i, j;

    for(i = 0; i < L; i++)
        for(j = 0; j < C; j++)
            T[i][j] = 0;
}
/* A funcao imprimeMatriz imprime todos os elementos da
variavel matriz que possui L linhas e C colunas*/

void imprimeMatriz(int L,int C)
{
    int i, j,k;

    for(i = 0; i < L; i++)
    {
        k = 0;
        for(j = 0; j < C; j++)
        {
            gotoxy(j+72+k,i+8);
            if(T[i][j] == 1)
            {
                textcolor(LIGHTRED);
                printf("\xFE");  //imprime a peça já inserida na
matriz
                textcolor(WHITE);
            }
            else
            {
                textcolor(BLACK);
                printf("\xB0");
                textcolor(WHITE);
            }
            k ++;
        }
    }
}

void copiaPeca(int vet[][2],int rot, int x,int y)  //L = linhas;
C = colunas.
{
    int L = 4, C = 4, i, j, k = 0;

    for(i = L - 1; i >= 0 && (y + 1 - L + i) >= 0; i--)
    {
        for(j = 0; j < C ; j++)
        {
            if(P.PECA[rot][i][j] != 0)
            {
                T[y +1- L + i][j + x] = 1;
                vet[k][0] = y + 1 - L + i;
                vet[k][1] = j + x;
                k++;
            }
            else
            {
                T[y +1- L + i][j + x] = T[y + 1 - L + i][j + x];
            }
        }
    }
}

//apaga a posiçao anterior da peça
void apagaPeca(int vet[][2])
{
    int j;
    int ind_l, ind_c;
    for(j = 0; j < 4; j++)
    {
        ind_l = vet[j][0];
        ind_c = vet[j][1];
        T[ind_l][ind_c] = 0;
    }
}

void monta_moldura(int xi,int yi,int xf,int yf)
{
    int i = 0;
    textcolor(MAGENTA);
    for(i = xi; i <= xf; i++)
    {
        gotoxy(i, yi);
        printf("\xB2");
        gotoxy(i, yf);
        printf("\xB2");
        delay(30);
    }
    for(i = yi; i < yf; i++)
    {
        gotoxy(xi, i);
        printf("\xB2");
        gotoxy(xf, i);
        printf("\xB2");
        delay(30);
    }
    textcolor(WHITE);
}

void titulo()
{
    char nome[50] = {'T','E','T','R','I','S'};
    int i;

    textcolor(LIGHTRED);
    gotoxy(75,3);
    printf("Fran e Lucas PRESENTS ");
    delay(800);
    system("cls");
    textcolor(LIGHTGREEN);
    for(i = 0; i < 6; i++)
    {
        gotoxy(i + 79, 3);
        printf("%c   ", nome[i]);
        delay(20);
    }
    delay(1000);
    limpa_text(0,0,30,2);
    textcolor(WHITE);
}

// num = seta, col e linha  = primeiras posição, colunas = num de
colunas
int cond_colun(int num, int rot, int qnt_c[], int col, int
lin_atual)
{
    int i, linhas, max, colunas;
    lin_atual--;
    col--;
    if(rot == 0 || rot == 2)
    {
        linhas = qnt_c[1];
        colunas = qnt_c[0];
    }

    else if(rot == 1 || rot == 3)
    {
        linhas = qnt_c[0];
        colunas = qnt_c[1];
    }

    max = col + colunas;

    gotoxy(10,10);
    printf("col %d", col);
    //delay(1000);

    if(num == 75)
    {
        for(i = lin_atual - linhas; i < lin_atual; i++)
        {
            if((T[i][col] + T[i][col-1]) == 2)
            {
                return 0;
            }
            else
            {
                return 1;
            }
        }
    }
    else if(num == 77)
    {
        for(i = lin_atual - linhas; i < lin_atual; i++)
        {
            if((T[i][max] + T[i][max+1]) == 2 )
            {
                return 0;
            }
            else
            {
                return 1;
            }
        }
    }
    return 1;
}

void mudacol(int col[], int rot[1], int qnt[],int x[], int linha,
int time[])
{
    char seta;
    int colunas;

    if(rot[1] == 0 || rot[1] == 2)
        colunas = qnt[0];

    else if(rot[1] == 1 || rot[1] == 3)
        colunas = qnt[1];

    int max = col[1] + colunas;
    seta = getch();

    if(seta == 75 && col[1] > 0 &&
cond_colun(seta,rot[1],qnt,col[1],linha)) //empiricamente
        col[1] = col[1] - 1;

    else if(seta == 77 && max < 10 &&
cond_colun(seta,rot[1],qnt,col[1],linha))
        col[1] = col[1] + 1;

    else if(seta == 72)
        rot[1] = (rot[1] + 1) % 4;

    else if(seta == 112)
    {
        gotoxy(78,19);
```

```
            textcolor(4546546);
            printf("PAUSADO");
            textcolor(BLACK);
            gotoxy(10,10);
            system("pause");
            textcolor(WHITE);
            limpa_text(77,18,86,20);
        }
        else if(seta == 80)
        {
            time[0] = 0;
        }
        else if(seta == 27)
        {
            x[1] = 386;
        }
        else
        {
            x[1] = 1;
        }
    }

    // verifica se ha uma peça abaixo da que esta descendo
    int chegada(int col, int linha, int rot, int qntC[])
    {
        int i = 0, aux = 0, maximo = 0, colunas = 0;

        if(rot == 0 || rot == 2)
            colunas = qntC[0];

        else if(rot == 1 || rot == 3)
            colunas = qntC[1];

        maximo = col + colunas;

        for(i = col; i < maximo; i++)
        {
            aux = T[linha][i] + T[linha + 1][i];

            if(aux == 2)
            {
                return 1;
            }
        }
        return 0;
    }
```

```
// elimina linha quando esta completa (colocando 0 na linha
completa)
void eliminaL(int vet[], int v)
{
    int i, j;

    for (i = 0; i < v; i++)
    {
        for(j = 0; j < 10; j++)
        {
            T[vet[i]][j] = 0;
        }
    }
}

// verifica se a linha esta cheia
int verificaL(int lin, int col, int vet[])
{
    int i, j, k = 0, soma;

    for(i = 0; i < 24; i++)
    {
        soma = 0;
        for(j = 0; j < 10; j++)
        {
            soma = soma + T[i][j];
        }

        if(soma == 10)
        {
            vet[k] = i;
            k++;
        }
    }
    return k;
}

//peças acima da linha completa desce
void peca_down(int linha)
{
    int i,j;

    for(i = linha; i > 0; i--)
    {
        for(j = 0; j < 10; j++)
```

```
            {
                T[i][j] = T[i-1][j];
            }
        }
}

// seleciona dificuldade
int dificuldade()
{
    int nivel;

    textcolor(LIGHTCYAN);
    gotoxy(24,9);
    printf("DIFICULDADE");

    textcolor(LIGHTGREEN);
    gotoxy(24,11);
    printf("Nivel: ");
    textcolor(LIGHTGRAY);
    gotoxy(20,12);
    printf("Niveis vao de 0 a 30");
    gotoxy(32,11);
    scanf("%d", &nivel);
    textcolor(WHITE);

    limpa_text(19,8,40,13);
    return nivel;
}

void HideCursor()
{
    CONSOLE_CURSOR_INFO cursor = {1, FALSE};
    SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE),
&cursor);
}

void jogar(int nivel)
{
    int i = 0, j = 0, vet_v[10000], lin_el = 0;
    int col[2], rot[2], m, cor = 0, x[1], time[1];
    int vet_apagar[4][2] = {{10,0},{10,0},{10,0},{10,0}};
    int qnt_c[2], vet[4], valor = 0, pontos = 0;
    int muda_nivel = 10;

    x[1] = 1;
```

```
for(i = 0; i < 10000; i++)
{
    vet_v[i] = rand()%7;
}
i = 0;
monta_moldura(59,27,68,32);

while(i < 3)
{
    textcolor(LIGHTGREEN);
    gotoxy(78,19);
    printf("%d...", i+1);
    delay(700);
    i++;
}
gotoxy(78,19);
printf("PRONTO");
delay(500);
textcolor(WHITE);
limpa_text(75,18,90,20);

zeraMatriz(24,10);
imprimeMatriz(24,10);
delay(1000);

int linha = 0;

/*PRIMEIRO LAÇO*/

while(j < 100000)
{
    col[1] = 4;
    linha = 0;
    rot[1] = (rand() % 4);
    sorteia_peca(vet_v[j],vet_v[j+1],qnt_c);

    int vet_apagar[4][2] = {{10,0},{10,0},{10,0},{10,0}};

    /*SEGUNDO LAÇO*/
    //monta_moldura();

    textcolor(LIGHTCYAN);
    gotoxy(58,9);
    printf("NIVEL:   %d", nivel);

    gotoxy(58,10);
```

```c
printf("PONTOS:  %d", pontos);

gotoxy(58,11);
printf("LINHAS:  %d", lin_el);
textcolor(WHITE);

valor = 0;
time[0] = 500;

while(linha < 24)
{
    if(kbhit())
    {
        mudacol(col, rot, qnt_c, x, linha, time);

        if(x[1] == 386)
        {
            break;//quebra o laço
        }
    }

    if(linha > 0)
        apagaPeca(vet_apagar);

    copiaPeca(vet_apagar,rot[1], col[1], linha);

    imprimeMatriz(24, 10);

    delay(time[0]/nivel);

    if(chegada(col[1],linha, rot[1],qnt_c))
    {
        if(linha < 2 )
        {
            x[1] = 386;
        }
        break;//quebra o laço
    }
    linha++;
}
if(verificaL(linha, col[1], vet))
{
    valor = verificaL(linha, col[1], vet);
    eliminaL(vet, valor);
    lin_el = lin_el + valor;
    for(i = 0; i < valor; i++)
```

```
                peca_down(vet[i]);
        }

        if(valor == 1)
        {
            pontos += 40 * (nivel);
        }

        else if(valor == 2)
        {
            pontos += 100 * (nivel);
        }

        else if(valor == 3)
        {
            pontos += 300 * (nivel);
        }

        else if(valor == 4)
        {
            pontos += 1200 * (nivel);
        }

        if(lin_el == muda_nivel)
        {
            nivel++;
            muda_nivel += 10;
        }

        if(x[1] == 386)
        {
            break;
        }
        j++;
    }

    for(i = 0; i < 6; i++)
    {
        limpa_text(79,18,87,23);
        for(j = 0; j < 5; j++)
        {
            m = 0;
            for(int k = 0; k < 4; k++)
            {
                gotoxy(k+79+m,j+18);
                if(perde[i][j][k] == 1)
                {
                    textcolor(LIGHTGREEN);
                    printf("\xDB\xDB");
                    delay(50);
                }

                else
                {
                    printf(" ");
                }
                m++;
            }
        }
        delay(200);
    }
    limpa_text(78,17,90,23);
}

int main()
{
    HideCursor();
    int opcao, i, nivel = 1;
    srand(time(NULL));

    titulo();

    do
    {
        gotoxy(27,15);
        monta_moldura(71,7,91,32);
        textcolor(546546);
        gotoxy(28,9);
        printf("TETRIS");
        textcolor(LIGHTGREEN);
        gotoxy(24,11);
        printf("[1]-JOGAR");
        gotoxy(24,12);
        printf("[2]-DIFICULDADE");
        gotoxy(24,13);
        printf("[3]-SAIR");
        textcolor(LIGHTGRAY);
        gotoxy(27,15);
        printf("Opcao: ");
        scanf("%d", &opcao);
        textcolor(WHITE);
        switch(opcao)
        {
        case 1:
            limpa_text(20,5,40,20);
            jogar(nivel);
            break;
        case 2:
            limpa_text(20,5,40,20);
            nivel = dificuldade();
            break;
        case 3:
            system("cls");
            textcolor(BLACK);
            for(i = 0; i < 30; i++)
            {
                system("cls");
                textbackground((i%4)+10);
                gotoxy(50,15);
                printf("TCHAU");
                delay(0.5);
            }
            system("cls");
            textbackground(BLACK);
            textcolor(WHITE);
            return 0;
            break;
        default:
        {
            textcolor(RED);
            gotoxy(24,17);
            printf("OPCAO INVALIDA!");
            delay(1000);
            system("cls");
            textcolor(WHITE);
        }
        }
    }
    while(opcao != 3);

    return 0;
}
```