**Documentação - Trabalho Prático**

**Universidade Federal de Minas Gerais**
**Programação e Desenvolvimento de Software 1**
**2023-1**

**Rafic Alves Magalhães Farah Leão**
**Matrícula: 2020108075**

## 1.Introdução

Para a realização do trabalho prático, foi proposto pelo professor o desenvolvimento de um jogo baseado no Dancing Plate, de Atari 2600, numa versão simplificada, baseada em C e com a utilização da biblioteca para criação de jogos "Allegro".

No jogo desenvolvido, o personagem controlado pelo jogador, representado por um triângulo, deve equilibrar pratos em cima de postes conforme eles aparecem ao longo do tempo, conforme acumula pontos. Se um dos pratos ficar muito tempo sem ser equilibrado, ele cai, uma tela mostrando a pontuação do jogador aparece, e a tela do jogo se fecha. Caso a pontuação do jogador tenha sido maior do que a maior pontuação registrada na memória, aparece um texto mostrando a antiga maior pontuação e o recorde atual. Caso o jogador feche a tela do jogo, a tela de pontuação também aparece conforme explicado anteriormente.

## 2. Implementação

### 2.1 Estruturas

O jogo foi desenvolvido baseado em 3 estruturas principais:

1. O Jogador;
2. O Prato;
3. O poste;

O 'Jogador' possui 4 variáveis que descrevem o seu posicionamento e deslocamento na tela: o 'x', que indica a posição do jogador e é inicializada no meio da tela; o 'mov_esq' e o 'mov_dir', inicializados como 0, indicam se o jogador se movimenta ou não pra direita ou pra esquerda, em que 0 indica que o jogador está parado, e 1 indica que ele se move para algum dos dois lados; 'cor' determina a cor

do triângulo controlado pelo jogador; e 'vel' indica a velocidade em que o triângulo se movimenta.

O 'Prato' é inicializado como um array de tamanho 8, e possui 5 variáveis: o 'x', que indica a posição do prato e é inicializada baseada em qual índice do array o prato está localizado; a 'energia', que é inicializada com 1 e aumenta conforme o tempo até o valor de 255, quando o prato cai e o jogo acaba; 'status', que é inicializado como 0 e indica se o prato já apareceu (aparece quando muda pra 1); e 'tempo', que indica o tempo para o prato aparecer, e é inicializado aleatoriamente baseado em números pré-determinados de modo que os pratos mais ao centro apareçam antes.

O 'Poste', assim como o Prato, é inicializado como um array de tamanho 8 e possui 4 variáveis: o 'x' e o 'y', que indicam a altura e a largura dos postes; o 'status', que é inicializado como 0 e vira 1 caso o jogador se posicione abaixo dele e aperte a tecla "espaço", e indica se o poste está sendo equilibrado ou não; e 'cor', que indica a cor do poste e é inicializada como a cor branca e muda para vermelho caso o status seja 1.

### 2.2 Funções e Procedimentos Auxiliares

- void desenha_cenario(): Pinta o fundo da tela do jogo de rosa e desenha os postes em branco;
- void inicializa_jogador(Jogador *j): Determina os valores iniciais do jogador;
- void desenha_jogador(Jogador j): Executa a função "al_draw_filled_triangle()", que desenha um triângulo verde com as dimensões determinadas na chamada da função.
- void atualiza_jogador(Jogador *j): Responsável pela movimentação do jogador, muda o valor da variável 'x' do jogador caso 'mov_esq' ou 'mov_dir' sejam iguai a 1;
- void inicializaPratos(Prato pratos[ ]): Determina os valores iniciais para cada um dos pratos;
- void desenhaPrato(Prato pratos[ ]): Desenha os pratos conforme seus status mudam. Caso a energia do prato passe de 255, desenha o prato caído no chão e executa a função "al_rest(1)";
- void atualizaPrato(Prato pratos[ ], Poste poste [ ], ALLEGRO_TIMER* timer): atualiza o status dos pratos para 1

caso pratos[i].tempo tenha sido alcançado e aumenta ou diminui pratos[i].energia conforme o tempo passa ou o prato é equilibrado;

- void inicializaPoste(Poste poste[ ]): Determina os valores iniciais para cada um dos postes;
- void atualizaPoste(Poste poste[ ], Jogador *j): Caso o status do poste seja 1, muda a cor do poste para vermelho;
- void drawScore(char *txt, float score, ALLEGRO_FONT *fonte): Desenha a pontuação atual do jogador no canto inferior direito da tela;
- int newRecord(float score, float *record): Abre ou cria um arquivo chamado "recorde.txt" e o lê. Caso a pontuação atual do jogador seja maior do que a pontuação registrada no arquivo de texto, coloca a pontuação atual no arquivo de texto. Retorna 0 caso o recorde de pontuação não seja ultrapassado e 1 caso o contrário.