UNIVERSIDADE FEDERAL DE MINAS GERAIS
PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE I

GUSTAVO MATTOS LOPES

TRABALHO PRÁTICO: DOCUMENTAÇÃO

"DANCING PLATES: THE GAME"

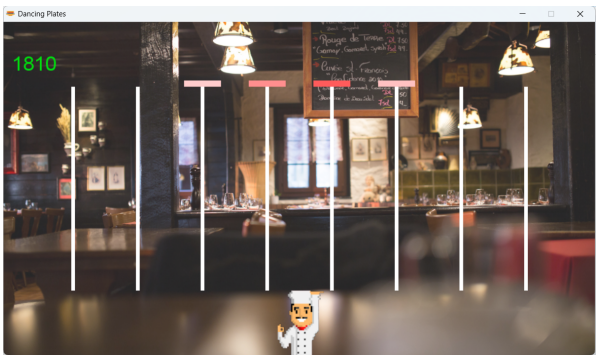
BELO HORIZONTE
1º SEMESTRE DE 2023

1. Introdução

O trabalho prático em questão relativo ao curso de Programação e Desenvolvimento de Software I objetiva pôr à prova as habilidades de desenvolvimento computacional do aluno e o domínio das técnicas de programação adquiridas através da referida disciplina. Nesse sentido, a proposta de elaboração do trabalho se trata da produção de um jogo eletrônico gráfico análogo ao "Dancing Plates", utilizando a linguagem de programação C em consonância à biblioteca Allegro5, adotada em prol da confecção de aplicações gráficas.

No jogo "Dancing Plates", o jogador controla o personagem a fim de manter um conjunto de pratos suspensos e equilibrados em postes, em que a partida é encerrada caso haja um longo intervalo de tempo transcorrido sem que um dos pratos seja estabilizado; ou seja, o jogador perde se um prato cai no chão. A seguir, serão apresentadas informações acerca da operação do jogo e detalhes de sua implementação.

2. Descrição e instruções do jogo

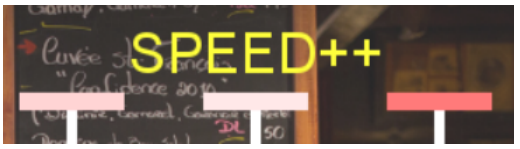


Ao inicializar o jogo, é aberta a página de menu principal, onde cabe ao usuário a escolha de uma das opções de jogo: com a participação de um ou dois jogadores. A dificuldade é a mesma para ambos os modos. Logo após clicar no botão referente à quantidade de jogadores, o jogo se inicia, a trilha sonora é ativada e os pratos vão surgindo à medida em que o tempo decorre, do centro para as extremidades da tela. Uma vez que os pratos aparecem, cada um deles está sujeito a um desequilíbrio próprio, denotado pela mudança gradual de cor (branco :: vermelho). Dessa forma, urge-se que os pratos sejam equilibrados pelo jogador, e isso somente é feito quando o jogador está parado e se a mão da *sprite player* estiver posicionada exatamente abaixo de cada poste que sustenta o prato. Ao promover o equilíbrio de um dos pratos, esse retorna gradualmente ao seu estado original (cor branca) e há um efeito visual sobre o poste tocado. Na circunstância em que um dos pratos atinge o desequilíbrio máximo, o prato cai e o jogo termina. A pontuação do jogador é constantemente exibida na tela e, ao fim, será comparada frente ao recorde de pontos vigente.

Para realizar a movimentação no modo de um jogador, o usuário deve usar as teclas 'A' e 'D' para produzir o movimento do personagem, respectivamente, para a esquerda e para a direita. Enquanto as teclas de movimento são pressionadas, a *sprite* se movimenta e para apenas quando a tecla é solta. Para equilibrar o prato, após posicionar corretamente o jogador em relação ao poste, deve ser pressionada a tecla ESPAÇO. Para a versão de dois jogadores, a

dinâmica do primeiro jogador é idêntica à descrita, enquanto o segundo jogador se movimenta por meio das setas do teclado e equilibra os pratos pela tecla ENTER.

Ao longo do jogo, serão disponibilizados *power-ups* ao jogador, sortidos entre aumento de velocidade, aumento de espessura dos postes e congelamento do desequilíbrio dos pratos. Todos eles possuem duração de dez segundos e, para liberá-lo, é necessário equilibrar (uma vez) o prato em que a novidade aparece, que também é escolhido aleatoriamente. O *power-up* gerado aparece por escrito na tela e desaparece assim que sua validade é esgotada.



3. Implementação do programa

3.1. Execução principal

Será descrita, a priori, a funcionalidade do programa principal (função main) e, subsequentemente, cada função criada será explicitada em detalhes.
OBS.: as linhas 12 a 45 do código se encontram devidamente comentadas com relação à sua aplicação e se referem à declaração das constantes e variáveis globais usadas no jogo, ademais das estruturas de dados para jogador e pratos.

Na função main:

É definida uma *seed* em ‘srand()’ com base na função ‘time()’, para que, a cada execução do programa, sejam gerados valores distintos pelas funções ‘rand()’ implementadas, visando melhorar a experiência do usuário ao jogar.

São definidos ponteiros dos tipos ALLEGRO_DISPLAY, ALLEGRO_EVENT_QUEUE, ALLEGRO_TIMER para representarem, respectivamente, a tela a ser gerada, a lista de eventos do programa e o contador de tempo durante a execução do jogo.

Funções do Allegro são executadas para inicializar o próprio Allegro e seus módulos que permitem desenhar primitivas geométricas na tela, reconhecer eventos de teclado e mouse e carregar imagens, fontes e sons. Além disso, a partir dos ponteiros criados anteriormente, é montada a tela com as dimensões definidas em pixels, cujo título da janela é definido como “Dancing Plates”, é criada uma fila de eventos, que reconhecerá eventos do tipo tela, tempo, teclado e mouse e é gerado o contador que incrementa uma unidade a cada 1.0/FPS segundos (que, no caso, FPS é uma constante global de valor 100, em que o contador aumenta um a cada dez milissegundos). A fonte, o som e as imagens são carregadas do diretório ‘./assets’ mediante ponteiros. É usado a estrutura de dados ALLEGRO_BITMAP para manipular as

imagens no jogo, a qual foi utilizada para carregar os arquivos .png relativos à *sprite* dos jogadores, o ícone da janela e os *backgrounds* adotados.

Para a exibição do menu principal, foi criado um laço while condicionado à variável *start*. Nesse loop, desenha-se a tela e os botões mediante a função '**draw_beginning()**' e atualiza a exibição pela função 'al_flip_display()'. Ainda nesse bloco de código, é esperado por um evento na fila de eventos, que é armazenado na variável *event*, cujo campo *type* é comparado aos eventos do tipo clique de mouse e fechamento da tela. Se o usuário fecha a tela, o programa se encerra, retornando 0. Caso haja clique no mouse e esse ocorra dentro do escopo dos botões (identificado por sua posição em coordenadas), a variável *start* é atualizada para 0, quebrando o laço atual e, caso o botão de dois jogadores tenha sido ativado, a variável *playing2* é atualizada para 1 de modo a indicar que haverá mais um jogador.

Antes de adentrar no laço relativo ao jogo propriamente dito, são inicializados um ou dois jogadores pela função '**initialize_player()**' e o vetor referente aos pratos pela função '**initialize_plates()**'. São criadas variáveis para definir o tempo de aparição do *power-up* e se o mesmo se encontra ativo e, ademais, o contador é iniciado - 'al_start_timer()'.

No laço while que representa a jogabilidade, a música carregada é iniciada e espera-se por um evento da lista de eventos. Note que a possibilidade do evento ser do tipo timer permite atualizar a exibição na tela em cerca de 100 frames por segundo (FPS), como definido anteriormente. Assim, a cada atualização do contador, o cenário é desenhado ('**draw_scenario()**'), os postes são atualizados ('**modify_poles()**'), bem como os jogadores e os pratos ('**update_player()**', '**update_plates()**'), que também são desenhados ('**draw_player()**', '**draw_plates()**'). A tela é atualizada com tais modificações e em tal velocidade supracitada, dando ao usuário uma percepção de dinamicidade. Além disso, dentro de cada repetição relativa à atualização do contador, é controlado a aparição, a ativação e o término do *power-up*, pelas variáveis criadas e as funções '**powerup_apparition()**', '**powerup_activated()**' e '**powerup_disabled()**'. Caso o evento seja de outros tipos, ocorre:

- ➔ Evento de fechamento de tela: loop do jogo é quebrado; passa à exibição do recorde;
- ➔ Evento de tecla pressionada: compara se as teclas se referem às do movimento ou às do equilíbrio e altera os booleanos de movimento e equilíbrio da *struct* Jogador;
- ➔ Evento de tecla liberada: altera os booleanos de movimento e equilíbrio da *struct* Jogador para o valor 0 (*false*) caso as teclas sejam correspondentes.

O loop do jogo propriamente dito também é encerrado em caso de derrota, cuja atividade será discorrida ao tratar da função '**draw_plates()**'. Quando terminado, a melodia de fundo é destruída e a variável pontos armazena o tempo de duração do jogo (que equivale a pontuação do jogador). Para acessar o recorde vigente na máquina, utiliza-se a função '**record()**'. O próximo laço while está atrelado à tela de exibição da pontuação/recorde, que é montada conforme relação das duas variáveis a partir da função '**draw_record_screen()**'.

Ao final, quando é fechada a tela de recordes, são executados procedimentos de fim de jogo (destruir o temporizador, fechar a tela, limpar memória...) e o programa é por fim encerrado.

3.1. Funções auxiliares

void draw_beginning(ALLEGRO_BITMAP *, ALLEGRO_FONT *);

➔ A função '**draw_beginning**' recebe por parâmetro os ponteiros para a imagem de fundo da tela e para a fonte do texto. A função implementa a tela inicial e desenha seus botões.

void draw_scenario(ALLEGRO_BITMAP *, ALLEGRO_TIMER *, ALLEGRO_FONT *);

➔ A função '**draw_scenario**' recebe por parâmetro os ponteiros para a imagem de fundo, para a fonte do texto e para o temporizador. A função implementa a tela de jogo, desenha os postes (primitiva de retângulo) e imprime o valor do temporizador (pontuação) na tela.

void modify_poles(Prato *, Jogador);

➔ A função '**modify_poles**' recebe por parâmetro a variável e o vetor referentes, respectivamente, ao jogador e aos pratos. Caso o jogador esteja parado e equilibrando um dos pratos, na posição correta de um poste, esse é desenhado em uma cor aleatória, denotando a ação de equilíbrio. O novo poste sobrescreve aquele outrora desenhado. Note que a função é chamada duas vezes, para os dois jogadores, caso exista.

void initialize_player(Jogador *);

➔ A função '**initialize_player**' recebe por parâmetro o ponteiro para a variável do jogador. São atribuídos, por referência, os valores base da *struct* para inicializar o jogador (posição, velocidade, booleanos de movimento e equilíbrio e index).

void draw_player(Jogador, ALLEGRO_BITMAP *);

➔ A função '**draw_player**' recebe por parâmetro a variável do jogador e o ponteiro de sua *sprite*. A *sprite* do jogador é desenhada na posição horizontal indicada pela *struct*, ajustado para que o valor x do jogador esteja centralizado na mão do personagem (*sprite*).

void update_player(Jogador *);

➔ A função '**update_player**' recebe por parâmetro o ponteiro da variável do jogador. Caso os booleanos de movimento indiquem verdadeiro, a função altera, por referência, a posição x do jogador, subtraindo (esq.) ou somando (dir.) o valor da velocidade. Note que isso só ocorre se o personagem não atravessar as extremidades da tela.

float gener_time_for_plate(int);

➔ A função '**draw_player**' recebe por parâmetro a variável que indica o índice de cada prato. Gera-se o tempo de aparição de cada um, que é retornado pela função. O mecanismo adotado para que os pratos centrais apareçam antes foi a criação de um vetor com intervalos pré-estabelecidos.

void initialize_plates(Prato *);

➔ A função '**initialize_plates**' recebe por parâmetro o ponteiro para o vetor de pratos. São atribuídos, por referência, os valores base da *struct* para inicializar cada prato (posição, tempo para aparição, booleanos de aparição e energia - vinculada à estabilidade).

void draw_plates(Prato *, int *, ALLEGRO_TIMER *, ALLEGRO_FONT *, ALLEGRO_BITMAP *);

- ➔ A função '**draw_plates**' recebe por parâmetro os ponteiros para o vetor de pratos, para a variável *playing*, para o temporizador, para a fonte e para a imagem de fundo. Os pratos são desenhados (primitiva de retângulo) em cores de acordo com cada energia.
- ➔ Caso a energia de um prato seja máxima (255, tornando sua cor = RGB(255, 0, 0)), a própria função encerra o jogo, parando o temporizador, alterando *playing* por referência para 0 (finalizando o loop na main) e promovendo a queda do prato (atualização da tela, com apenas o resultado da execução da função '**draw_scenario**' e o respectivo prato desenhado, diminuído o valor de sua posição y gradualmente, em um intervalo de 0.1s).

void update_plates(Prato *, ALLEGRO_TIMER *, Jogador, int);

- ➔ A função '**update_plates**' recebe por parâmetro os ponteiros para o vetor de pratos e para o temporizador, além das variáveis do jogador e do booleano para o efeito de congelamento, que, caso o *power-up* esteja ativo, a energia dos pratos não aumentará. Caso contrário, a energia de cada prato exibido aumenta em 10.2 a cada 1s (10.2 * 25 = 255, ou seja, serão 25s até o desequilíbrio máximo). Ademais, altera o booleano de aparição dos pratos caso o tempo para aparecer tenha sido superado e, se o jogador estiver equilibrando um prato em específico, sua energia é atualizada de forma decrescente.

long long int record(long long int, int);

- ➔ A função '**record**' recebe por parâmetro a variável que armazena a pontuação no jogo (obtido do temporizador). Ocorre a leitura do recorde vigente do arquivo .bin no mesmo diretório do programa. Se a pontuação atual é maior que o recorde, altera o arquivo e escreve a pontuação como novo recorde. Note que há dois valores no arquivo, referentes ao recorde para cada modo de jogo (1P ou 2P). Logo, é preciso manter um dos valores ao atualizar o arquivo. A função retorna o recorde, antigo ou novo, e imprime no terminal.

void draw_record_screen(long long int, long long int, ALLEGRO_FONT *);

- ➔ A função '**draw_record_screen**' recebe por parâmetro o ponteiro para a fonte e as variáveis que armazenam o recorde e a pontuação obtida no jogo em vigor. Na própria função, compara-se os valores dos pontos e do recorde. Caso sejam idênticos, escreve-se na tela que o jogo acabou e um novo recorde foi obtido. Se não, a tela aparece com o valor do recorde vigente e da pontuação atual escritos.

void powerup_apparition(Jogador *, Prato *, int *, int);

- ➔ A função '**powerup_apparition**' recebe por parâmetro os ponteiros para a variável do jogador, para o vetor de pratos e para o indicador de atividade do *power-up*. Além disso, recebe a variável relativa ao index do prato sortido para que o *power-up* apareça. Assim, ao chamar a função, desenha-se uma primitiva de círculo amarelo que representa o poder especial e, quando o usuário equilibra o prato correspondente, é gerado um valor aleatório entre 1 e 3 para o indicador do *power-up*, alterado por referência.

void powerup_activated(int *, Jogador *, int, Jogador *, Prato *, int *);
void powerup_disabled(int *, Jogador *, int, Jogador *, Prato *, int *);

➔ As funções '**powerup_activated**' e '**powerup_disabled**', uma vez que operam de maneira similar, recebem os mesmos parâmetros: o ponteiro para o indicador de atividade do *power-up*, para ambos os jogadores, para o vetor de pratos e para o booleano que indica se o poder de congelamento está ativo. Além disso, recebe a variável *playing2*, que indica se há a presença de segundo jogador (caso haja, o poder será ativado para ambos, independente de qual deles o obteve). Assim, as funções utilizam o valor do indicador de atividade do *power-up* (entre 1 e 3), para definir qual dos poderes existentes será ativado e desativado em sequência - após dez segundos. A função '**powerup_disabled**' promove o retorno aos padrões originais e altera, por referência, o valor do indicador para 0 (o power-up não estará mais ativo).