Documentação trabalho prático

“Dancing plates - Versão Pokemon”

Universidade Federal de Minas Gerais

Programação e Desenvolvimento de Software I

01/2023

Lucas Affonso Pires

1 - Apresentação:

O trabalho prático desenvolvido na disciplina de Programação e Desenvolvimento de Software tem como base o jogo dancing plates, cujo objetivo é controlar o jogador para que ele equilibre pratos dispostos uniformemente pela tela. Caso a variável que controla o equilíbrio do prato chegue ao valor mínimo, o prato cai e o jogo se encerra. Para o desenvolvimento do jogo, foi utilizado a biblioteca allegro, cujo comandos auxiliam na criação das funções do jogo.

Partindo desse pressuposto, para implementar o jogo e torná-lo mais visualmente agradável de acordo com a biblioteca allegro, foi utilizado como inspiração a franquia de jogos Pokemon, cujo animações, sprites e trilhas sonoras em bits são capazes de tornar o jogo muito mais agradável ao usuário.

2 – Manual do jogo:

Controles:

O jogo tem comandos muito simples, sendo eles, inclusive, explicados dentro do jogo. O jogador se move apenas horizontalmente com o usuário apertando as teclas “A” e “D”, fazendo com que o jogador se mova para a esquerda ou para a direita, respectivamente. Além disso, há também o comando executado ao apertar a tecla “ESPAÇO”, fazendo com que o jogador, caso esteja parado e localizado embaixo de uma das hastes que seguram o prato, equilibre-o. Outras teclas que são necessárias para alterar entre as fases e menus são devidamente indicadas dentro do jogo.

Lógica:

O objetivo principal do jogo é manter os pratos equilibrados a maior quantidade de tempo possível, para evitar que os Pokemons fujam. O indicador da energia do prato é a sua cor, variando da cor branca para a vermelha. Caso a energia se esgote e o prato se torne completamente vermelho, o jogador perde e sua pontuação é mostrada no final do jogo de acordo com a quantidade de tempo que ele sobreviveu. Caso a pontuação seja maior que uma anteriormente registrada, o jogador define um novo

recorde e isso é indicado visualmente na tela final. Para auxiliar o jogador, existem alguns sons que serão tocados conforme o decorrer do jogo, um indicando quando um prato aparece, e um indicando quando a energia de um prato está próxima de se esgotar.

3 - Descrição do código:

3.1 - Bibliotecas:

Todas as bibliotecas utilizadas no decorrer do programa são declaradas aqui.

3.2 - Constantes:

Para auxiliar no desenvolvimento do cenário e na disposição dos elementos nele presente, foram definidas no começo algumas constantes. Entre elas estão: A constante que define o FPS, as constantes que definem o tamanho da tela, o tamanho dos pratos, o tamanho do jogador, o tamanho das hastes e um chão base.

3.3 - Structs:

Existem apenas duas structs que são utilizadas no desenvolvimento do jogo:

Struct jogador - Responsável por definir valores essenciais ao jogador, como a posição em x na tela, a posição em y, a velocidade, valores referentes a direita e a esquerda controlando a movimentação, e alguns parâmetros referentes a animação das sprites utilizadas.

Struct prato - Responsável por definir valores essenciais ao prato, como a posição em x na tela, a posição em y, a energia que controla seu equilíbrio, uma condição de existência e uma condição de queda, e alguns valores que são referentes as hastes, já que essas são feitas juntamente aos pratos.

3.4 - Funções:

Existem uma gama de funções, referentes ao controle dos elementos durante a execução do jogo. Algumas funções executam as mesmas funções, mas precisam de parâmetros diferentes por conta das sprites, portanto, nesse caso, elas serão explicadas mutualmente.

Referentes ao ambiente:

Funções de fundo: *void faz_floresta, void faz_oceano, void faz_deserto, void faz_ceu* - Responsáveis por desenhar no fundo da tela uma imagem pré-definida, ambientando as diferentes fases do jogo.

Referentes ao jogador:

Funções de inicialização: *void comecajogador, void comecajogadordeserto, void comecajogadorceu* - Responsáveis por inicializar os valores referentes ao jogador e as sprites, tornando possível diferentes animações.

Funções de desenho: *void faz_jogador* - Responsável por desenhar o jogador na tela pela primeira vez

Funções de animação: *void animacao_jogador, void animacao_jogador_deserto, void animacao_jogador_ceu* - Responsáveis por fazer a alternância de diferentes sprites para gerar uma animação.

Funções de reset de animação: *void reset_animacao_jogador, void reset_animacao_jogador_deserto, void reset_animacao_jogador_ceu* - Responsáveis por voltar a animação ao quadro original.

Funções de posição: *void att_jogador, void att_jogador_oceano, void att_jogador_deserto, void att_jogador_ceu* - Responsáveis por controlar a posição do jogador na tela e gerar a movimentação.

Referentes aos pratos:

Função de inicialização: *void comecaprato* - Responsável por inicializar os valores referentes aos pratos.

Funções de desenho e som: *void faz_prato, void faz_prato2* - Responsáveis por desenhar o prato na tela e mudar sua cor gradualmente, além de indicar por sons quando a energia está se esgotando. Além disso, desenha as pokeballs.

Função de geração: *void gera_prato* - Responsável por gerar aleatoriamente os pratos e tocar um som quando isso acontece. Além disso, levemente favorece o aparecimento dos pratos no centro da tela.

Função de energia: *void att_energia* - Responsável por controlar a diminuição da energia com o passar do tempo.

Funções para alterar a energia: *void modifica_energia, void modifica_energia_deserto, void modifica_energia_ceu* - Responsáveis por adicionar energia ao prato quando o jogador aperta ESPAÇO e indiciar por meio de um som.

Função de queda: *void prato_caiu* - Responsável por desenhar o prato no chão caso sua energia se esgote.

Referentes aos Pokemons:

Funções de desenho: *void faz_pokemon, void faz_pokemon_oceano, void faz_pokemon_deserto, void_faz_pokemon_ceu* - Responsáveis por desenhar os pokemons do lado das pokeballs.

Referentes às hastes:

Funções de desenho: *void poste_muda, void poste_muda_deserto, void poste_muda_ceu* - Responsáveis por alterar a cor do poste quando o jogador aperta ESPAÇO embaixo dele

Referente ao recorde:

Função de arquivamento e retorno: *recorde* - Responsável por armazenar o recorde caso a pontuação seja a mais alta. Esta função sempre retorna o recorde à main.

3.5 - Main:

A int main pode ser dividida em 3 partes principais: As inicializações do programa, sendo elas referentes as funções allegro, ao áudio, à parte visual e as variáveis do programa. O loop principal, que executa o jogo. E os procedimentos de finalização pós jogo, que finalizam o programa. Por ser mais extenso, o loop principal será deixado para o final

Inicializações:

Nessa parte, são declaradas todas as variáveis e ponteiros que serão utilizados no decorrer do jogo, incluindo áudios, sprites, valores e etc. Ademais disso, são também inicializadas as funções allegro necessárias para que o jogo ocorra.

Procedimentos de finalização:

Nessa parte, todos os ponteiros declarados, sejam eles de display, de áudio ou de sprites são destruídos para liberar a memória.

Loop principal:

O loop principal em si, pode ser divido em estados diferentes. Entre eles estão o menu, o menu2, o tutorial, a fase1, a fase2, a fase3, a fase4, o fimdejogo, as reinicializações e as funções de tecla.

Estados visuais:

Menu - Apenas um visual introdutivo ao jogo, pode levar ao Menu2 ou então finalizar o jogo.

Menu2 - Aqui estão as opções de escolha de fases e do tutorial.

Tutorial - Uma tela que indica brevemente a lógica do jogo e os comandos que são necessários para que se possa jogar o jogo.

Fim de jogo - A tela responsável por mostrar ao jogador qual a pontuação feita durante a fase, qual o recorde, e dar a ele as opções de tentar qualquer fase novamente ou então retornar ao Menu2.

Telas de jogo:

Todas as fases seguem a mesma lógica de gameplay, mudando apenas o visual.

Fase1 - Cenário de uma floresta.

Fase 2 - Cenário de uma praia próxima ao oceano.

Fase 3 - Cenário de um deserto.

Fase 4 - Cenário localizado no céu.

Comandos auxiliares:

Reinicializações - Servem para que o jogador possa voltar a qualquer fase sem ter que finalizar o jogo e abri-lo novamente.

Funções de tecla - Responsáveis por realizar ações se apertadas ou soltas durante a execução do jogo. Algumas funções de tecla estão presentes nos menus e telas

finais, mas são apenas para a navegação entre as telas. Estas são relativas à jogabilidade