DOCUMENTAÇÃO - Keystone Kappers

Integrante: Cauã Rodrigues de Oliveira
Professor: Pedro Olmo Stancioli Vaz de Melo
Disciplina: Programação e Desenvolvimento de Software I
Semestre: 02/2025

Belo Horizonte
2025

1 Introdução

Para a concluir a disciplina Programação e Desenvolvimento de Software I, foi necessário que alunos criassem um jogo utilizando a biblioteca Allegro. Através desta biblioteca, foi possível expandir os conhecimentos aprendidos em sala para um uso prático - a criação de um protótipo do clássico jogo de Atari Keystone Kapers. Essa documentação visa explicar o funcionamento e a implementação do jogo.

2 Jogabilidade

No jogo, o usuário controla um policial que possui como objetivo capturar um ladrão no menor tempo possível.

O policial é controlado pelas teclas A (movimento para esquerda), D (movimento para direita) e espaço (pulo). Ele deve atravessar salas, subir andares e pular sobre poças de lama para conseguir capturar o ladrão antes que esse chegue ao final do shopping, onde o jogo é ambientado.

O shopping possui elevadores em suas extremidades. Caso o policial esteja em um andar par, o elevador a direita sobe um andar e o da esquerda desce um andar. O contrário acontece quando o policial está em um andar ímpar. A contagem dos andares começa em 0.

O policial possui um pulo duplo que o capacita a evitar as poças de lama que o desaceleram consideravelmente. Se ele não evitar nenhuma poça de lama, irá perder o jogo.

O ladrão move-se automaticamente a partir do início do jogo, começando a se mover do andar do meio e da sala do meio do shopping. Ele se movimenta com metade da velocidade do policial e caso chegue ao final do shopping sem que o policial o capture, o policial perde o jogo.

Assim que o jogo começa, o contador exibe o número de segundos decorridos, representando a pontuação do jogador. Caso ele ganhe o jogo no menor tempo já registrado, o jogo registra um novo recorde para ser exibido na tela de fim de jogo.

3 Estruturas Principais

O jogo conta com duas estruturas de dados para facilitar a implementação do jogo:

- *Pessoa:* é a estrutura usada para implementar tanto o policial quanto o ladrão. Essa estrutura registra as coordenadas dos personagens na tela e no shopping, a direção para qual estão se movimentando, quem o personagem é (policia ou ladrão), se o personagem está pulando e quantos pulos ele possui.

- *Mud:* é a estrutura usada para implementar as poças de lama. Essa estrutura registra as coordenadas da lama na tela e no shopping, assim como a sua largura.

4 Definições e Variáveis Globais

Para que certos parâmetros do jogo, como por exemplo o número de frames por segundo, o número de salas e o tamanho da tela sejam facilmente alterados, o código do TP conta com as seguintes definições e variáveis globais:

- *define NUM_ANDARES:* número de andares do shopping (default: 3);

- *define NUM_SALAS:* número de salas do shopping (default: 3);

- *define COP_VEL:* velocidade da movimentação lateral do policial, ou seja, quantos pixels ele se move para direita ou para esquerda quando o jogador aperta as teclas de movimentação lateral (default: 3);

- *define JUMP_VEL:* velocidade do pulo do policial, ou seja, quantos pixels ele se move quando o jogador aperta a tecla espaço (default: 3);

- *define GRAVIDADE*: taxa em pixels com a qual a velocidade do pulo do policial descresce (default: 0.1);

- *define DESCONTO_VEL_LAMA:* número pelo qual a velocidade do policial é multiplicada quando ele está na lama (default: 0.15);

- *define TRUE:* número definido apenas para tornar o código mais legível (default: 1);

- *define FALSE:* número definido apenas para tornar o código mais legível (default: 0);

- *define MAX_JUMPS*: número de pulos permitidos ao policial (default: 2);

- *const int PESSOA_W:* largura do ladrão e do policial em pixels (default: 30);
- *const int PESSOA_H:* altura do ladrão e do policial em pixels (default: 50);
- *const float FPS:* taxa de frames por segundo (default: 100);
- *const int SCREEN_W:* largura da tela em pixels (default: 960);
- *const int SCREEN_H:* altura da tela em pixels (default: 540);
- *const int PISO_H:* altura do piso de cada andar em pixels (default: 30);
- *const int MAX_POCAS:* número máximo de poças de lama no jogo (default: 9);
- *int score:* pontuação inicial do jogador (default: 0);
- *ALLEGRO_COLOR BKG_COLOR:* cor do plano de fundo do jogo (default: preto);
- *ALLEGRO_FONT *size_32:* ponteiro para as características da fonte do jogo (default: arial, tamanho 32);
- *int FLOOR_H:* coordenada y onde o piso se encontra (default: SCREEN_H / NUM_ANDARES);
- *int MALL_W:* largura do shopping em pixels (default: NUM_SALAS * SCREEN_W).

5 Funções e Procedimentos

- *void init_global_vars():* esse é um procedimento simples que inicializa as variáveis globais, sendo elas a cor do plano de fundo, a fonte a ser usada no jogo, a largura do shopping e a altura de cada andar.
- *void initCop(Pessoa *cop):* esse procedimento recebe um ponteiro para Pessoa e inicializa as variáveis do policial, definindo sua cor, suas coordenadas na tela e no shopping, as características do seu pulo e a direção do seu movimento.

- *void initThief(Pessoa *thief):* esse procedimento recebe um ponteiro para Pessoa e inicializa as variáveis do ladrão, definindo sua cor, suas coordenadas na tela e no shopping e a direção do seu movimento.

- *int colisaoCopThief(Pessoa cop, Pessoa thief):* essa função recebe duas Pessoas, o policial e o ladrão, e verifica se uma colisão ocorreu. Caso a colisão tenha ocorrido, a função retorna 1 e, caso contrário, retorna 0.

- *int fuga_ladrao(Pessoa thief, int *ladrao_venceu):* essa função recebe uma Pessoa que representa o ladrão e verifica se ele chegou ao final do shopping. Se o ladrão tiver chegado ao final do shopping, a função retorna 1. Caso contrário, a função retorna 0. Além disso, a função recebe um ponteiro para inteiro que armazena 1 em seu conteúdo, caso o ladrão tenha chegado ao final do jogo, e 0, caso contrário.

- *int random(int min, int max):* essa função recebe dois inteiros, um mínimo e um máximo e retorna um número inteiro aleatório entre o mínimo e máximo.

- *void desenhaCenario(Pessoa cop, Mud obst[][NUM_ANDARES]):* esse procedimento desenha o plano de fundo, os andares, as salas e as poças de lama. Para isso, ele recebe uma pessoa representando o policial e uma matriz de Muds que armazena todas as poças de lama do jogo.

- *void desenha_pessoa(Pessoa p, Pessoa *cop):* esse procedimento recebe uma Pessoa e a desenha na tela. Além disso, ele recebe um ponteiro para Pessoa que é usado para descobrir em qual sala o policial está e desenhar o ladrão apenas se ele estiver na mesma sala que o policial.

- *void elevador(Pessoa *p):* esse procedimento é o responsável por controlar a mecânica do elevador no jogo. Ele recebe um ponteiro para Pessoa e altera o andar do personagem no conteúdo desse ponteiro.

- *int colisaoLama(pessoa *p, Mud[][NUM_ANDARES]):* essa função recebe um ponteiro para Pessoa e uma matriz de Muds e verifica se houve uma colisão entre a pessoa e alguma lama do jogo. Caso a colisão tenha ocorrido, a função retorna 1 e, caso contrário, retorna 0.

- *void pulo(Pessoa *p, float y_base):* esse procedimento é responsável pela mecânica do pulo do policial. Para seu funcionamento, ele recebe um ponteiro para Pessoa que será usado para mudar a coordenada y do jogador e um float que é usado para saber onde é o “chão” no qual o policial deve aterrissar após o fim do pulo.

- *void updatePerson(Pessoa *p, Mud obst[][NUM_ANDARES]):* esse procedimento recebe um ponteiro para Pessoa e uma matriz de Muds e atualiza as variáveis da pessoa. Esse procedimento é responsável por diversas mecânicas do jogo, sendo elas:
    1. mudança das coordenadas do policial na tela e no shopping de acordo com as teclas apertadas pelo jogador;
    2. aciona a função pulo() quando o jogador aperta a tecla espaço;
    3. muda a velocidade do policial caso ocorra colisão com alguma poça de lama;
    4. aciona a função elevador() caso o policial chegue a algum elevador;
    5. controla toda a movimentação do ladrão.

6 Programa Principal

O programa principal é um int main() clássico em que cada linha faz as ações descritas abaixo:

- *Linha 383 à 443:* começam as rotinas básicas de inicialização do jogo;

- *Linha 446 à 472:* inicializam as variáveis globais, o policial e o ladrão e cria as poças de lama;

- *Linha 477 à 492:* começam o loop que controla o jogo, registrando os eventos em uma fila de eventos.

- *Linha 495 à 515:* se o evento registrado for um evento do temporizador, desenham o cenário, atualizam as variáveis do policial e do ladrão, terminam o jogo caso o policial ou o ladrão vençam e reinicializam a tela.

- *Linha 516 à 546:* se o evento registrado for o jogador ter apertado uma tecla do teclado, alteram as variáveis do policial de acordo com a tecla apertada.

- *Linha 547 à 560:* se o evento registrado for o jogador ter soltado uma tecla do teclado, alteram as variáveis do policial de acordo com a tecla soltada.

- *Linha 563 à 568:* se o evento for o jogador ter fechado o jogo, terminam o jogo. Além disso, registram o evento final da fila de eventos.

- *Linha 574 à 649:* lidam com todo o sistema de recorde do jogo e exibem a tela de fim de jogo. Além disso, realizam os procedimentos de fim de jogo (fecha a tela, limpa a memória, etc).