UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE I

Desenvolvimento de Jogo de plataforma na linguagem C

Professor: Pedro Olmo

Francielle Aparecida da Paz

Belo Horizonte

30/01/2025

1. DESCRIÇÃO DO JOGO ORIGINAL

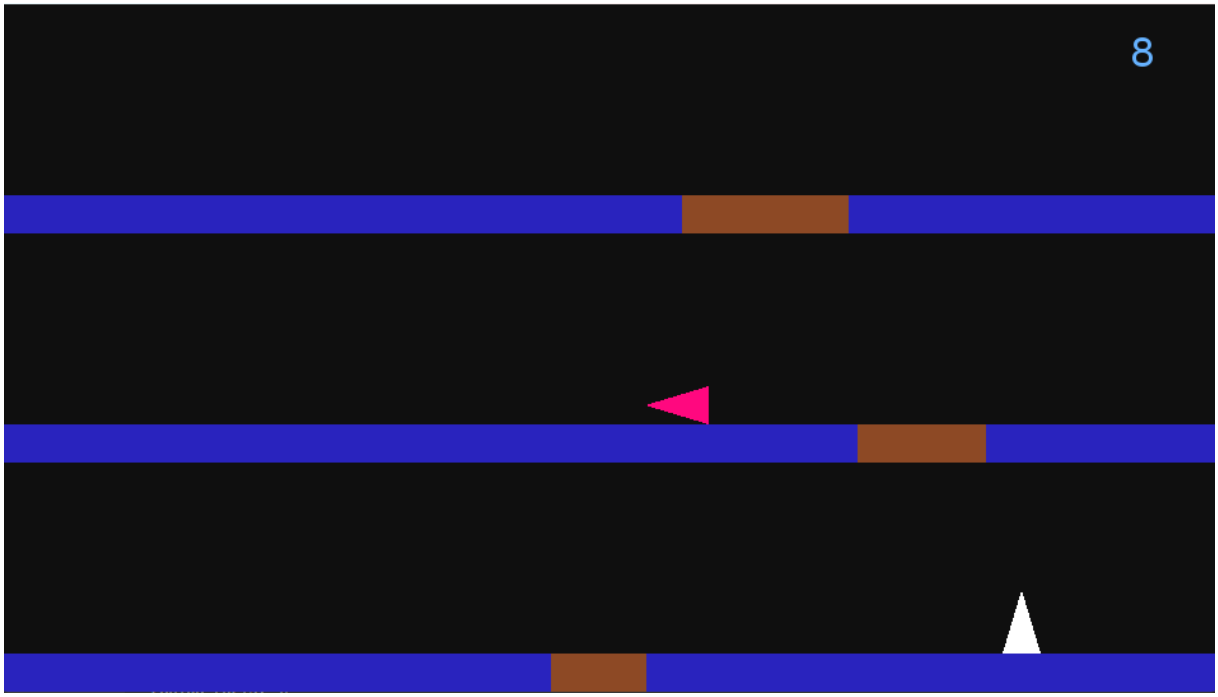
Keystone Kapers é um jogo de sobrevivência para Atari 2960 lançado em 1983. O jogador assume o papel de um segurança de shopping que precisa capturar um ladrão que acabou roubar uma lógica, a captura acontece quando o jogador encosta no ladrão e imediatamente outra fase se inicia dessa vez mais difícil que a anterior, o jogo se encerra quando o ladrão consegue escapar. O objetivo é conseguir chegar o mais longe possível, sendo ágil, desviando de diversos objetos que aparecem na tela como aviões, bolas quicando, baús e etcs, e somar a maior quantidade de pontos possível.

2. OBJETIVO

O objetivo do trabalho é desenvolver uma fase do jogo inspirada em Keystone Kapers, utilizando linguagem C e a biblioteca para programação de jogos multiplataforma Allegro.

3. TELA E CONTROLES

A tela da fase de jogo inspirada no jogo de Atari, está mostrado abaixo. O policial é o avatar branco e o ladrão o avatar rosa. Os espaços marrons nos andares são “lamas” caso o jogador não pule com antecedência ela irá reduzir sua velocidade. A pontuação do jogo é mostrada no canto superior direito.



Os comandos do jogo são:

Tecla D: move o policial para direita.

Tecla A: move o policial para a esquerda

Espaço: Pula

Para sair basta clicar no X no canto superior direito da tela.

4. Descrição do código

Linhas 1 a 8: Includes de bibliotecas utilizadas no desenvolvimento.

Linhas 10 a 25: Definição das constantes do jogo.

Linhas 27 a 38: Criação da struct Pessoa, que será utilizada para definir o avatar do policial e do ladrão. As variáveis do avatar são comprimento e altura, cor, posição em x no espaço do shopping, sala e andar na qual se encontra e variáveis de rastreio de movimentação.

Linhas 40 a 42: Criação da struct Mud, que é utilizada para definir os obstáculos na tela. As variáveis dela são posição inicial, comprimento e atual.

Linhas 45 a 50: Declaração das variáveis globais.

Linhas 52 a 115: São as funções de inicialização dos elementos do jogo. *init_Global_var()* inicializa as variáveis globais, *init_Policial()* define os parâmetros iniciais associados ao avatar do policial, *init_Ladrao()* define os parâmetros iniciais associados ao avatar do ladrão, *init_Lama()* define os parâmetros dos obstáculos na tela.

Linha 116 a 133: Função *desenha_Cenario(Pessoa cop, Mud lama[][NUM_ANDARES])*. Essa função define a cor do fundo da tela, cria os andares e cria os obstáculos nos andares. Ela utiliza os parâmetros para definir posição dos obstáculos na tela e a cor dos andares.

Linha 135 a 175: Funções *desenha_policial(Pessoa p)* e *desenha_ladrao(Pessoa p, Pessoa cop).* Para representar o avatar tanto do ladrão quanto do policial foi escolhido um triângulo que muda sua orientação de acordo com a movimentação dos personagens, quando parado sua ponta aponta para cima quando em movimento aponta para a direção de movimentação. O policial é desenhado em todos os momentos, já o ladrão só é desenha quando ele está na mesma sala que o policial.

Linha 177 a 208: Função *elevador(Pessoa p).* Essa função move verticalmente os avatares entre os andares. Ela verifica se o avatar está próximo às bordas esquerda ou direita do shopping e, dependendo do andar em que ela se encontra (ímpar ou par), move para o andar superior ou inferior. A posição do avatar é ajustada para garantir que ele não ultrapasse os limites do shopping. Além disso, a função atualiza as coordenadas visíveis da pessoa na tela com base na sua posição no shopping e no andar atual, considerando a altura de cada andar e a altura do piso.

Linha 210 a 231: Função *obstáculo_lama(Pessoa *p, Mud obs [][NUM_ANDARES]*

Linha 233 a 284: Função *update_pessoa(Pessoa *p)* atualiza a posição do personagem com base no movimento, colisões e gravidade. Primeiro, verifica se o

personagem está na lama para aplicar redução de velocidade e impedir pulos. Depois, processa os movimentos horizontais para esquerda e direita, garantindo que ele não ultrapasse os limites do shopping e chamando o elevador se necessário. Caso esteja pulando, ajusta a velocidade e altura até atingir o limite, iniciando a queda. A gravidade é aplicada durante a queda até que o personagem atinja o chão. Por fim, a posição horizontal na tela e a sala atual são calculadas a partir da posição global no shopping.

Linha 286 a 316: Função *update_ladrao(Pessoa *p)* Faz a mesma coisa que a função *update_pessoa* mas para o Ladrão, a diferença é que nela o ladrão se move sozinho, não é verificado colisão com obstáculos nem há implementação de pulo.

Linha 318 a 338: Função *colisaoPolicialLadrao(Pessoa p, Pessoa l)* Verifica se os avatares do ladrão e do policial estão na mesma sala e mesmo andar, calcula os limites esquerdo, direita, topo e base dos avatares levando em consideração sua posição x e y na tela, e compara se a posição x e y de ambos se sobrepõem. Se se houver interseção há colisão a função retorna 0. Se nenhuma dessas verificações for verdade a função retorna 1.

Linha 340 a 350: Função *ladraoVenceu(Pessoa ladrao)* Verifica se o avatar do ladrão está na ultima sala e no ultimo andar e se ele estiver se sua posição em x ultrapassa o limite direito da tela, se ultrapassar significa que o ladrão ganhou e função retorna 1, se nenhuma dessas afirmações for verdade a função retorna 0.

Linha 351 a 427: Parte inicial do código main do jogo. O código inicializa a biblioteca Allegro e seus addons necessários, como fontes, primitivas gráficas e imagens. Em seguida, define variáveis globais e configura elementos do jogo, incluindo a tela, a fila de eventos e o temporizador. As estruturas dos personagens (policial e ladrão) são inicializadas com suas propriedades, assim como os obstáculos (lama). O sistema de eventos é configurado para capturar interações da tela, teclado e temporizador. Por fim, o temporizador é iniciado e as variáveis de controle do jogo, como playing e ladrao_venceu, são definidas antes do loop principal.

Linha 428 a 505: Looping principal do jogo, enquanto a variável playling for 0 ele é executado. O loop principal aguarda eventos e executa ações conforme necessário. Se um evento de temporizador for detectado, o cenário é atualizado, os personagens são redesenhados e seus estados são atualizados, verificando colisões e condições de vitória. Se o evento for de fechamento da tela, o jogo termina. Se uma tecla for pressionada ou solta, as ações do policial são ajustadas, como movimento e pulo. A cada segundo, o placar é incrementado e a tela é atualizada. O jogo continua até que o ladrão vença ou ocorra uma colisão com o policial.

Linha 507 a 569: É iniciado os procedimentos de fim de jogo. Eles tratam do encerramento do jogo e da exibição da tela final. Se o ladrão venceu, a tela é limpa, uma imagem de fim de jogo é carregada e exibida, e o jogo pausa por alguns segundos antes de fechar. Caso contrário, o jogo verifica e atualiza o recorde, exibindo uma mensagem de vitória, a pontuação final e o recorde atual. Após a exibição da tela

final, os recursos como o temporizador, a tela e a fila de eventos são liberados antes do encerramento do programa.

5. CONCLUSÃO

Utilizando a biblioteca Allegro e linguagem de programação C foi construído uma fase de jogo inspirado no Keystone Kapers graficamente mais simples que o jogo original consolidando os conhecimentos adquiridos em sala de aula.