**Universidade Federal de MInas Gerais**

**Matemática Computacional**

Roger Souza Brandão

**Trabalho Prático**

Space Invaders

[**1. Introdução**](#_54z948j70xq3) **2**

[**2. Funcionalidades do Jogo**](#_k91amomer7oi) **2**

[**3. Como Jogar**](#_wu0zyofkl1x0) **4**

[**4. Estrutura do Projeto**](#_u83i9ccwqbo8) **4**

[**5. Configuração, Compilação e Execução**](#_wm6seh2y8gej) **6**

[5.1 Configuração](#_9vq1x8xa9ipw) 6

[5.2 Compilação e Execução](#_wqoz5wkcggob) 6

[**6. Visão Geral do Código**](#_u2zyzsr136ci) **7**

[6.1 main.c](#_lsm0hsacmvl2) 7

[6.2 model.h / model.c](#_gaj71l7vu44e) 7

[6.3 view.h / view.c](#_cl62llb9b5bc) 8

[6.4 controller.h / controller.c](#_qpicv332y8fd) 9

[**7. Lógica Chave do Jogo**](#_q19lsqa8i6l0) **10**

**Documentação do Jogo Space Invaders**

Este documento fornece uma visão geral do jogo Space Invaders, sua estrutura de projeto, como compilar e executar o jogo, e uma explicação detalhada dos principais componentes do código. Dessa forma, esta documentação deve fornecer uma base sólida para entender o jogo Space Invaders.

# 1. Introdução

O Space Invaders é um jogo clássico de arcade recriado em C utilizando a biblioteca Allegro 5. O objetivo do jogo é controlar uma nave espacial, destruir ondas de alienígenas invasores e evitar seus tiros e colisões. O jogo inclui funcionalidades como pontuação, vidas e um sistema de recorde.

# 2. Funcionalidades do Jogo

Controle da Nave: Movimentação lateral para esquerda e direita.

Tiro da Nave: A nave pode disparar projéteis para destruir os alienígenas.

Inimigos (Aliens): Múltiplas fileiras de alienígenas com diferentes tipos e valores de pontuação.

Movimento dos Aliens: Os alienígenas se movem horizontalmente e descem ao atingir as bordas da tela.

Tiros dos Aliens: Os alienígenas disparam projéteis aleatoriamente em direção à nave do jogador.

Colisões: Detecção de colisão entre:

Tiro da nave e alienígenas.

Tiro dos alienígenas e a nave.

Nave e alienígenas.

Alienígenas e o solo (resultando em "Game Over").

Pontuação e Vidas: Sistema de pontuação que aumenta ao destruir alienígenas, e vidas que diminuem ao ser atingido.

Recorde (High Score): A pontuação mais alta é salva e carregada para ser exibida.

Estados do Jogo: Gerenciamento dos estados "Rodando", "Game Over" e "Você Venceu".

Tela Final: Exibe a pontuação final e o recorde, permitindo reiniciar o jogo ou sair.

Sons: Efeitos sonoros para tiros da nave, tiros dos aliens, alienígenas sendo destruídos e movimento dos aliens.

Animações: Animação simples para os alienígenas.

# 3. Como Jogar

Movimentar a Nave: Use as teclas 'A' ou Seta Esquerda para mover para a esquerda, e 'D' ou Seta Direita para mover para a direita.

Atirar: Pressione a tecla Espaço para atirar.

Reiniciar (Tela Final): Pressione 'R' para reiniciar o jogo quando na tela final.

Sair (Tela Final): Pressione 'ESC' para sair do jogo, quando na tela final.

Fechar Janela: Clicar no botão 'X' da janela também encerra o jogo.

# 4. Estrutura do Projeto

O projeto está organizado nos seguintes diretórios e arquivos:

*Space-Invaders/*

*├── Assets/*

*│ ├── audio/*

*│ │ ├── fase.wav*

*│ │ ├── hit.wav*

*│ │ ├── morto.wav*

*│ │ └── tiro.wav*

*│ ├── fonts/*

*│ │ ├── arial.ttf*

*│ │ └── spaceage.ttf*

*│ └── images/*

*│ ├── alienp.bmp*

*│ ├── fundo.bmp*

*│ └── navep.bmp*

*├── bin/ (Gerado após compilação)*

*│ ├── controller.o*

*│ ├── main.o*

*│ ├── model.o*

*│ ├── spaceinvaders.exe*

*│ └── view.o*

*├── include/*

*│ ├── controller.h*

*│ ├── model.h*

*│ └── view.h*

*├── src/*

*│ ├── controller.c*

*│ ├── main.c*

*│ ├── model.c*

*│ └── view.c*

*└── Makefile*

*Assets/:* Contém todos os recursos do jogo (áudios, fontes e imagens).

*bin/:* Diretório onde os arquivos objeto (.o) e o executável (.exe) são gerados após a compilação.

*include/:* Contém os arquivos de cabeçalho (.h) que declaram as funções e estruturas de dados usadas em múltiplos arquivos .c.

*src/:* Contém os arquivos de código-fonte (.c) que implementam a lógica do jogo.

*Makefile:* Arquivo de script para automatizar o processo de compilação e limpeza do projeto.

# 5. Configuração, Compilação e Execução

## 5.1 Configuração

Para compilar e executar o jogo, você precisará de um compilador GCC (como o MinGW-w64 no Windows) e a biblioteca Allegro 5 instalada e configurada corretamente. Todo o processo de configuração está detalhado em: <https://github.com/rogersoub/Space_Invaders/blob/main/README.md> .

## 5.2 Compilação e Execução

* Navegue até o Diretório Raiz: Abra seu terminal (MSYS2 no Windows, ou terminal Linux/macOS) e navegue até a pasta Space-Invaders/ onde o Makefile está localizado.

*cd R:\UFMG\PDS\Space-Invaders*

Limpar o Projeto (Opcional, mas recomendado): Para remover quaisquer arquivos de compilações anteriores, execute:

*make clean*

* Compilar o Projeto: Para compilar todos os arquivos .c e linká-los no executável, execute:

*make*

Isso criará os arquivos .o em bin/ e o executável spaceinvaders.exe também em bin/.

* Copiar DLLs do Allegro (Somente Windows): Se você estiver no Windows e as DLLs do Allegro não estiverem no seu PATH, você precisará copiá-las para a pasta bin/ para que o executável possa encontrá-las.

*make copy-dll*

Nota: Verifique o Makefile para o caminho correto do seu diretório ALLEGRO\_DLL\_DIR.

* Executar o Jogo: Após a compilação bem-sucedida, execute o jogo:

*bin/spaceinvaders.exe*

# 6. Visão Geral do Código

O projeto segue uma arquitetura básica de Modelo-Visão-Controle (MVC), embora não estritamente formal.

## 6.1 main.c

Ponto de entrada do programa: Inicializa a biblioteca Allegro e seus addons (primitivas, imagens, fontes, áudio).

* Cria o display (janela do jogo), o temporizador e a fila de eventos.
* Registra as fontes de eventos (display, teclado, mouse, timer).
* Chama a função *game\_loop* (o loop principal do jogo).
* Destrói os recursos do Allegro ao finalizar.

## 6.2 *model.h / model.c*

Modelo - Dados e Lógica do Jogo

*model.h:*

Declara as constantes globais (tamanhos de tela, objetos, velocidades), enums para estados do jogo (*GameState*), e as estruturas de dados para Nave, *Alien, Shot e AlienShot*. Também declara todas as funções relacionadas à lógica do jogo, como inicialização de entidades, atualização de posições, detecção de colisões e gerenciamento de pontuação.

*model.c:*

Implementa as funções declaradas em *model.h.*

* *initNave, initAlien, initShot, initAlienShots:* Funções para inicializar a posição e estado das entidades.
* *update\_nave, update\_alien, updateShot, updateAlienShot:* Funções para atualizar a posição e estado das entidades a cada tick do jogo.
* *fireShot, fireAlienShot:* Lógica para disparar os tiros.
* *colisao\_alien\_solo, check\_ship\_alien\_collision, check\_shot\_alien\_collision, check\_alien\_shot\_nave\_collision:* Funções para detectar diferentes tipos de colisões.
* *all\_aliens\_eliminated:* Verifica a condição de vitória.
* *save\_high\_score, load\_high\_score:* Gerenciamento do arquivo de recorde (*record.txt*).

## 6.3 view.h / view.c

Visão - Desenho e Carregamento de Recursos Visuais/Sonoros

*view.h:*

Declara os ponteiros para os bitmaps (sprites) e samples (sons) usados no jogo, além das variáveis para as fontes. Declara as funções para carregar e destruir esses recursos, e as funções para desenhar os elementos do jogo na tela.

*view.c:*

Implementa as funções declaradas em *view.h.*

* l*oad\_fundo, load\_nave, load\_alien, load\_shot, load\_alien\_shot, load\_sounds, load\_game\_font, load\_score\_font:* Funções para carregar os recursos (imagens, sons, fontes) dos arquivos. Nota-se a conversão de máscara para alfa (*al\_convert\_mask\_to\_alpha*) para sprites com fundo magenta, tornando essa cor ignorada.
* *destroy\_fundo, destroy\_nave, destroy\_alien, destroy\_shot, destroy\_alien\_shot, destroy\_sounds, destroy\_game\_font, destroy\_score\_font:* Funções para liberar a memória alocada pelos recursos.
* *draw\_scenario, draw\_nave, draw\_alien, draw\_shot, draw\_alien\_shot:* Funções para desenhar o fundo, a nave, os alienígenas (com animação), e os tiros na tela.
* *draw\_score\_lives:* Desenha a pontuação atual, vidas e recorde.
* *draw\_game\_over\_screen:* Desenha a tela final de "Game Over" ou "Você Venceu".

## 6.4 controller.h / controller.c

Controlador - Gerenciamento de Eventos e Loop do Jogo

*controller.h*

Declara as variáveis globais de controle do jogo (*score, lives, high\_score, current\_alien\_animation\_frame*) e as funções principais do controlador: *process\_event, game\_loop e reset\_game*.

*controller.c:* Implementa as funções declaradas em *controller.h.*

* *score, lives, high\_score, current\_alien\_animation\_frame*: Variáveis globais para o estado do jogo.
* *alien\_shot\_timer, ALIEN\_SHOT\_INTERVAL\_MIN, ALIEN\_SHOT\_INTERVAL\_MAX:* Constantes e variáveis para controlar a frequência dos tiros dos alienígenas.
* *reset\_game*: Reinicializa todas as entidades do jogo, a pontuação, vidas, timers e carrega o recorde. É fundamental para iniciar um novo jogo.
* *process\_event*: Lida com a entrada do usuário (teclado, fechar janela). Atualiza o estado da nave e do jogo (current\_game\_state) com base nas teclas pressionadas.
* *game\_loop*: É o coração do jogo, contém o loop principal que mantém o jogo em execução. Aguarda por eventos na fila. Chama *process\_event* para lidar com as entradas.
* No estado *GAME\_RUNNING*, atualiza a lógica de todas as entidades (update\_nave, update\_alien, etc.), gerencia os tiros dos alienígenas e verifica todas as condições de colisão e fim de jogo.
* No estado *GAME\_OVER ou GAME\_WON*, chama *draw\_game\_over\_screen* para exibir a tela final e entra em um loop interno que aguarda por 'R' (reiniciar) ou 'ESC' (sair). Se 'R' for pressionado, chama *reset\_game* e muda *current\_game\_state* para *GAME\_RUNNING*. Se 'ESC' for pressionado ou a janela for fechada, define *current\_game\_state* para *GAME\_OVER* para encerrar o loop principal.
* Gerencia a lógica de redesenho da tela (redraw flag) para evitar "tearing".
* Gerencia a reprodução do som dos passos dos alienígenas.
* Libera todos os recursos do jogo ao sair do loop principal.

# 7. Lógica Chave do Jogo

Loop do Jogo (*game\_loop*): A estrutura *while(current\_game\_state != GAME\_OVER)* garante que o jogo continue rodando até que o estado final de saída seja alcançado. O loop interno na tela final permite que o jogador decida reiniciar ou sair sem fechar o aplicativo imediatamente.

Gerenciamento de Recursos: O carregamento e a destruição de bitmaps, fontes e sons são cruciais para evitar vazamentos de memória. As funções *load\_ e destroy\_* em *view.c* são responsáveis por isso.

Animação de Aliens: A animação é controlada por *current\_alien\_animation\_frame* e *ALIEN\_ANIMATION\_SPEED*, alternando entre dois frames de sprite a cada *ALIEN\_ANIMATION\_SPEED* ticks.

Movimento de Grupo de Aliens: A variável estática global\_alien\_x\_vel em model.c controla o movimento horizontal de todo o grupo de alienígenas. Quando qualquer alien atinge a borda da tela, a direção global\_alien\_x\_vel é invertida e todos os alienígenas descem um alien\_drop\_amount.

Timer de Tiro dos Aliens: *alien\_shot\_timer* e os intervalos *ALIEN\_SHOT\_INTERVAL\_MIN/MAX* controlam a frequência aleatória dos tiros dos alienígenas.

Colisões (AABB): As funções *check*\_ no *model.c* implementam a detecção de colisão por meio de Axis-Aligned Bounding Box (AABB), verificando se os retângulos que envolvem os objetos se sobrepõem.

High Score: O recorde é salvo em um arquivo *record.txt* e carregado a cada início de jogo, garantindo persistência da maior pontuação.

Controle de Áudio: O uso de *al\_get\_sample\_instance\_playing* para o som de fase em controller.c evita que o som se sobreponha repetidamente, tocando apenas quando a instância anterior terminou.