# 1. City Saver

O jogo denominado como *City Saver* é um remake básico e rudimentar do jogo R-type. A história de City Saver passa em um cenário global prestes a ser destruído, onde cientistas descobriram um grande asteroide vindo para a terra e o destruíram em inúmeros pedaços. O dever do jogador é eliminar cada pequeno asteroide usando as armas da nave, desviando dos obstáculos rumo ao infinito.

# 2. Documentação

## 2.1. Requisitos

* MinG 4.7.0
* Allegro 5.0.10

## 2.2. Bibliotecas

Allegro5 e suas dependências (Imagens, áudio, fontes, etc), stdio, time, stdlib, math.

## 2.3. Lógica de implementação

O corpo principal do jogo foi implementado no arquivo principal, *main.c*, onde é carregado as instancias necessárias para rodar o jogo e a lógica principal. Todas as dependências foram divididas baseadas em tipos e alocadas dentro da pasta src:

* Configurações: configurações principais do jogo e da aplicação, são alocadas na pasta config.
* Data: arquivos para armazenamento de dados da aplicação, são alocadas na pasta data.
* Eventos: bibliotecas pessoais criadas para uso dos eventos do Allegro e do jogo, são alocadas na pasta events.
* Fontes: arquivos de fontes de texto para uso na aplicação, são alocadas na pasta fonts.
* Imagens: todo e qualquer tipo de imagem utilizado no jogo, são alocadas na pasta images.
* Objetos: todo e qualquer coisa na tela, sejam tratativas para o usuário ou objetos em si, são alocadas na pasta objects.
* Sons: todo e qualquer tipo de arquivo relacionado ao áudio do jogo, são alocadas na pasta sounds.
* Utilidades: toda e qualquer biblioteca usada para a manipulação de dados de forma personalizada ao longo da aplicação, são alocadas na pasta utilities.
* Extensões: toda e qualquer extensão de alguma dependência criada, para fins de estender e melhorar o funcionamento da mesma posteriormente, são alocadas dentro das pastas anteriores, quando usadas, dentro de uma pasta denominada extended.

Para carregar todas essas dependências de forma rápida e organizada, foi criado um arquivo dentro da pasta src, sem nenhuma sub-pasta, denominado autoload.h. Este arquivo é o único include dentro da main.c, onde o mesmo faz todos os includes necessários de dependências do projeto de forma automatizada e organizada.

Os includes do autoload são feitos de forma sequencial, onde:

* Inclui as bibliotecas do Allegro.
* Inclui as bibliotecas nativas usadas.
* Inclui as bibliotecas de utilidades.
* Inclui as bibliotecas de configurações.
* Inclui as bibliotecas de objetos e suas respectivas extensões.
* Inclui as bibliotecas de eventos e suas respectivas extensões.

### 2.3.1. Utilidades

#### 2.3.1.1 arquivo.h

Funções relacionadas a manipulação de arquivos.

**bool escreveInteiroArquivo(const char \* localizacao, int valor);** : recebendo a localização do arquivo o valor a ser escrito dentro do mesmo, sobrescreve o conteúdo do arquivo por esse valor, retornando true, se foi feio com sucesso, e false, se deu algo errado.

**int leInteiroArquivo(const char \*localizacao);** : recebendo a localização de um arquivo contendo um inteiro, retornará o valor do inteiro se for lido com sucesso ou zero se algo der errado na leitura.

#### 2.3.1.2. string.h

**char \*concat(const char \*s1, const char \*s2);** : recebendo duas strings, concatena a primeira com a segunda e retorna a string concatenada.

#### 2.3.1.3. graus.h

**double to\_radians(double degrees);** : converte um valor em graus para radianos

#### 2.3.1.3. bitmap.h

**ALLEGRO\_BITMAP \*load\_bitmap\_at\_size(const char \*filename, int w, int h);** : recebendo a localização do arquivo, junto com as dimensões de largura e altura desejadas, retorna um bitmap do alegro no tamanho informado.

#### 2.3.1.4. rand.h

**int randInt(int n);** : Retorna um inteiro aleatório entre 0 e n-1.

**int randIntMinMax(int min, int max);** : Retorna um inteiro aleatório entre o valor mínimo e máximo informado.

**double randDouble();** : retorna um double aleatório entre 0 e 1.

**double randDoubleMinMax(double min, double max);** : retorna um double aleatório entre o valor mínimo e máximo informado.

### 2.3.2. Configurações

Funções e inicializações de configurações primordiais para o funcionamento da aplicação.

#### 2.3.2.1. config.h

**Declaração de variáveis globais**: linha 01 até linha 196.

**int iniciarAllegro();** : função onde instala e inicia todos os módulos do Allegro. Local também onde é instanciado os valores de variáveis globais que usam bitmaps, fontes, etc.

**int finalizaAllegro();** : realiza os procedimentos de fim de jogo do Allegro, como finalizar localizadores de eventos (timer, display, fila de eventos).

### 2.3.3. Objetos

#### 2.3.3.1. background.h

**struct Background** : Struct onde são armazenadas os bitmaps do fundo e seus clones, posições de ambos, tipos, etc.

**void iniciarBackground()**; : inicia um array de struct de backgrounds, definindo suas posições, imagens e bitmaps.

**void defineVelocidadeBaseadoNoTipo();** : define a velocidade do fundo baseado no tipo da instancia da struct. O tipo diz respeito a qual imagem é.

**void finalizaBackground();** finaliza os bitmaps de background.

**void calculaBackground();** : função que calcula a movimentação do fundo.

**void redesenhaBackground();** : função onde redesenha o fundo animado do jogo baseado nas atualizações calculadas.