

# Relatório - Fase II

Germano Huning Neuenfeld 9298340

Lucas Moreira Santos 9345064

Victor Wichmann Raposo 9298020

Maio 2016

## 1 Introdução

O documento visa descrever o projeto do jogo Spacewar. Allegro ideia e como baixar e como rodar Modelo ideia

## 2 Classes

Fizemos o uso de orientação a objetos usando structs. Definimos algumas classes que serão explicadas abaixo cuja documentação de cada função está especificada no cabeçalho das respectiva classe.

### 2.1 Body

O Body funciona como base para outros corpos. Ele possui propriedades básicas que todo corpo (nave, planeta, projétil) possui. Os atributos implementados nele são: raio (double), peso (double), ângulo em radianos (double), posição (Vector), força (Vector) e velocidade (Vector).

### 2.2 Projectile

Essa é a representação dos projéteis. Os atributos que eles possuem são: corpo (Body) e duração (double). Como não há orientação a objetos em C e portanto não há hierarquia, inserimos o corpo como atributo da struct.

### 2.3 Ship

Essa é a representação das naves. Os atributos que eles possuem são: corpo (Body) e nome (array de chars). Como não há orientação a objetos em C e portanto não há hierarquia, inserimos o corpo como atributo da struct.

## 2.4 Vector

Essa é a representação de um vetor no  $V^2$ . Os atributos que eles possuem são: x (double) e y (double).

## 3 MVC

Utilizamos a arquitetura de software Model-View-Controller para desenvolvimento do projeto por facilitar o desenvolvimento em módulos.

### 3.1 Modelos

Procuramos implementar uma orientação a objetos no nosso projeto pois acreditamos que fosse simplificar o desenvolvimento e manutenção do código.

O objeto Vector que criamos simplificou muito o manuseio e cálculo de velocidades, acelerações e posições.

O arquivo "simulation.c" é responsável por aplicar a modelagem física, computando as forças e movimentos.

#### 3.1.1 Gravidade

A cada interação zeramos os vetores de força de todos os corpos e depois aplicamos as forças gravitacionais entre:

- Planeta e nave 1;
- Planeta e nave 2;
- Nave 1 e nave 2;

Propositalmente, não computamos a gravidade nos projeteis pois queremos que seu movimento seja retilíneo.

#### 3.1.2 Movimento

Assumimos que, para o dt dado, o movimento dos corpos é um movimento uniformemente variado. A partir disso calculamos o espaço e a velocidade dos corpos em relação ao eixo x e ao eixo y;

#### 3.1.3 Toróide

A implementação do toróide não interfere na modelagem. // TO DO

### 3.2 Visão

O framework que usamos para o jogo foi o Allegro que utiliza por trás o OpenGL. Escolhemos essa engine por alguns motivos:

- Ainda é utilizado nos dias de hoje pela comunidade, o que implica em mais suporte;
- Há bastante material didático na internet;
- Usa OpenGL;

A parte do código que trata a apresentação na tela foi desenvolvida no arquivo "draw.c", visando a modularização.

### 3.3 Allegro

#### 3.3.1 Instalação

Para instalar o Allegro, siga as instruções da página oficial:

*[https://wiki.allegro.cc/index.php?title=Getting\\_started#Installing\\_Allegro](https://wiki.allegro.cc/index.php?title=Getting_started#Installing_Allegro)*

#### 3.3.2 Memory Leak

Detectamos que o Allegro apresenta um memory leak constante e pequeno, natural da sua implementação e que não interfere na execução do programa.

#### 3.3.3 Funcionamento

// TO DO

### 3.4 Controladores

Por ora, não fizemos nenhum trabalho que envolva controladores.

## 4 Makefile

Criamos um Makefile para o projeto tal que apenas os arquivos que sofreram alguma mudança sejam recompilados. Além disso, fizemos uma especificação para deletar os arquivos .o e outra para executar o programa.

Como em nosso grupo haviam usuários de Mac OS e de Linux, fizemos um Makefile que detecta em conta o sistema operacional e compila de acordo. Incluímos três flags para importar as bibliotecas do Allegro.

Para compilar, utilize o Makefile com:

> *make*

Para excluir os arquivos .o:

> *make clear*

Para executar uma entrada de teste pré-definida:

> *make test*

## 5 Testes

Criamos uma pasta "Samples" que contém diversos casos de testes distintos e verificamos, para cada um deles, se houve qualquer vazamento de memória com o Valgrind

Para usar o Valgrind:

```
> valgrind --leak-check = yes --track-origins = yes ./Spacewar 50000 <  
Teste.txt
```

## 6 Considerações Finais