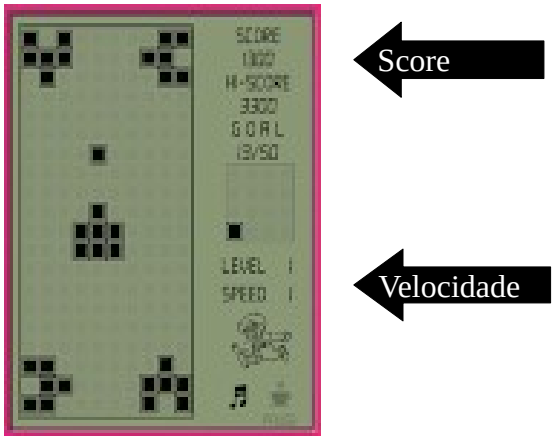
**INE5412 – Sistemas Operacionais I**
**Profs. Giovani Gracioli e Márcio Bastos Castro**
**2023/01**

# Projeto Final
# Jogo de “Naves” utilizando a Biblioteca de Threads Desenvolvida

## 1. Descrição

O projeto consiste em utilizar a biblioteca de Threads no espaço de usuário desenvolvida durante o semestre para implementar o jogo de “Naves” disponível em “vídeo games de mão antigos”:


O jogo consiste em um nave, que inicialmente está no centro da tela, que tem a capacidade de atirar e acertar as naves inimigas. No máximo 4 naves inimigas existem simultâneamente e tais naves vão girando e atirando aleatoriamente. A nave do jogador pode ser atingida 3 vezes antes do “game over” (tem 3 vidas). Se alguma nave inimiga acertar a nave do jogador, a nave do jogador perde uma vida. A cada acerto em uma nave inimiga, são computados 100 pontos no score. A nave inimiga que foi acertada deve esperar 2 segundos até que reapareça na tela. As naves não podem se tocar. Se os tiros da nave do jogador e de uma nave inimiga se acertarem, eles desaparecem (um cancela o outro). O algoritmo de movimento das naves é de livre projeto, mas deve conter pelo menos 2 algoritmos diferentes (ex: 2 naves com o algoritmo X e outras 2 naves com o algoritmo Y). Após a nave do usuário “matar” 4 naves inimigas, as naves inimigas aumentam sua velocidade (nível 2 de velocidade). Isso se repete até atingir o nível de velocidade 3. A velocidade atual (1, 2, ou 3) deve ser mostrada a direita, bem como a pontuação (score) atual.

O projeto deve implementar a descrição do jogo acima utilizando a biblioteca de Threads no espaço de usuário desenvolvida durante o semestre. Neste sentido, **cada nave inimiga deve ser uma thread, a nave do jogador deve ser outra Thread, assim como uma Thread responsável pelo desenho da tela**

**e outra para o tratamento da entrada do teclado (mínimo de 7 Threads)**. Os alunos têm a liberdade para criar outras Threads como acharem necessário.

As teclas de entrada são as seguintes:

- Teclas para cima, para baixo, lado esquerdo e lado direto para controlar a nave do jogador.
- P para pausar e despausar o jogo.
- Q para sair do jogo.
- R para reiniciar o jogo quando acabar.
- Espaço para atirar.

A biblioteca gráfica utilizada será a SFML (https://www.sfml-dev.org/). Para instalar a biblioteca no Ubuntu digite: **sudo apt-get install libsfml-dev**. A documentação da biblioteca encontra-se em https://www.sfml-dev.org/learn.php.

Cada grupo terá a liberdade para desenvolver o software seguindo o paradigma de orientação a objetos do C++ obrigatoriamente. Isso signifca que o uso de programação estruturada e elementos da linguagem C terão descontos na nota (veja avaliação abaixo). A estruturação do código em classes, subclasses (hierarquia) e reaproveitamento de código a partir disso será avaliada.

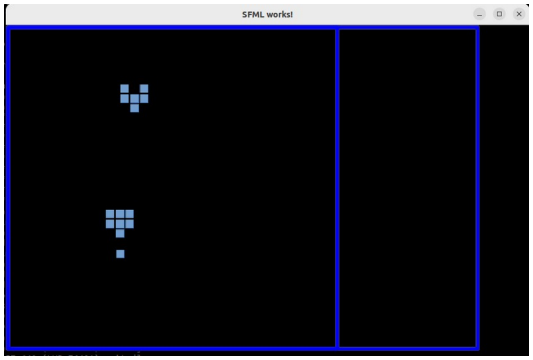
**Dica**: vocês podem manter os elementos da tela (como posição das naves e tiro) em uma matriz que será compartilhada pelas Threads e atualizada a cada "tick" do jogo. Lembrem-se de proteger a matriz e qualquer outra estrutura de dados compartilhada com Semáforos. Não utilizem variáveis globais, utilizem classes e atributos estáticos.

## 2. Arquivos Disponibilizados

Foram disponibilizados no moodle arquivos para criação de uma janela e desenho das estruturas básicas do jogo na tela utilizando a bilbioteca SFML. Dentro do diretório "sprites" são disponibilizados alguns arquivos para desenhar na tela. Por exemplo, dentro de sprites/space_ships, existem arquivos que representam as naves em suas 4 posições.

O exemplo disponibilizado também já mostra o tratamento das teclas pela biblioteca:

```
./main
Keyboard para baixo!
Keyboard para cima!
Keyboard direita!
Keyboard esquerda!
Keyboard esquerda!
```




## 3. Formato de Entrega

Todos os arquivos utilizados na implementação do trabalho devem ser entregues em um único arquivo .zip ou .tar.gz na atividade do moodle. Deve ser anexado um arquivo Makefile para compilar o

código e um relatório descrevendo o projeto do software (diagramas UML – casos de uso, classes e sequência da Thread do Pacman pelo menos).

## 4. Data de Entrega

Data e horário da entrega estipulados na tarefa do moodle.

## 5. Avaliação

A avaliação se dará em 3 fases

1. Avaliação de compilação: compilar o código enviado. Caso haja erros de compilação, a nota do trabalho será automaticamente zerada.

2. Avaliação de execução: para validar que a solução executa corretamente sem falhas de segmentação. Caso haja falhas de segmentação, a nota é zerada. Será também avaliado o uso de variáveis globais (-5 pontos) e vazamentos de memória (-20%). **Variáveis estáticas de classes não são consideradas globais.**

3. Avaliação da organização do código: busca-se nesta fase avaliar a organização do código orientado a objetos. Deve-se usar classes e objetos e não estilo de programação baseado em procedimentos (como na linguagem C). Uso de programação estruturada (-30%). Uso da linguagem C ao invés de linguagem C++ (-30%)

Este trabalho precisará ser apresentado aos professores nos dias 23/06 e 27/06/2023 em sala de aula. Plágio não será tolerado em nenhuma hipótese ao longo dos trabalhos, acarretando em nota 0 a todos os envolvidos. Cada grupo terá entre 10 e 15 minutos para apresentar, devendo apresentar primeiramente a estrutura do software projetado para o jogo (1 figura) e em seguida apresentar o jogo funcionando. Haverá votação para a escolha do melhor jogo, sendo este premiado com 1 ponto na nota final do projeto.

Os pesos da avaliação serão os seguintes:

- 30% para o relatório
- 10% para a apresentação
- 60% para a avaliação funcional e organização do código (seguindo as regras acima)