Projeto Final - Space Shooters

Disciplina: Sistemas Operacionais I - INE5412

Rafael Luis Sol Veit Vargas

Thiago Augusto Bewiahn

Professores:

Giovani Gracioli e Márcio Bastos Castro

UFSC - Universidade Federal de Santa Catarina

1. Introdução

O software produzido é uma versão do jogo Call of Shooty: Worlds at War. Uma demo do jogo pode ser vista aqui: Space Shooter 2D. Apenas um subconjunto de funcionalidades do jogo original serão obrigatoriamente implementadas. Essas funcionalidades serão descritas na seção de requisitos. Funcionalidades extras podem ser implementadas de acordo com o julgamento e planejamento da equipe de execução do projeto. Por fim, a biblioteca de Threads desenvolvida ao longo do semestre será utilizada para criar threads responsáveis por cada parte do programa, a ser explicado adiante.

2. Requisitos de software

- Código produzido em C++;
- Biblioteca gráfica ALLEGRO;
- Biblioteca de threads desenvolvida durante a disciplina para gerenciamento e criação de threads;
- Espaçonave do jogador deve ser uma thread distinta;
- Criação de espaçonaves inimigas roxas (ver vídeo) deve ser uma thread distinta;
- Classe Window deve ser responsável por desenhar os objetos do jogo na interface gráfica, e também deve ser uma thread distinta;
- Entrada do teclado deve ser processada por uma thread distinta;

- Geração de minas (ver vídeo) deve ser feita a cada 30 segundos por uma thread distinta. Uma mina explode jogando tiros para todas as direções se não for destruída pelo jogador em até 5 segundos;
- Criação de um “Boss” (ver vídeo) com 1 minuto de jogo em uma thread distinta;
- A nave do jogador deve ter 3 vidas;
- Controle de colisões e vidas do jogador deve ser feito em uma thread distinta;
- Documentação deve ser fornecida por meio de diagramas de caso de uso e classes. Além disso, a classe PlayerShip deve ter um diagrama de sequência modelado.

3. Diagramas e estrutura do software

Os casos de uso do sistema são apenas 3: atirar, mover a nave e sair do jogo.

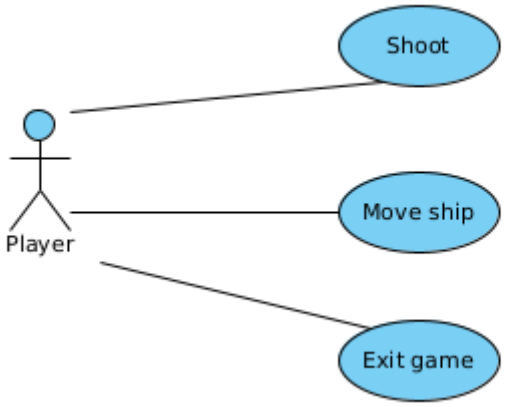

Diagrama 1. Casos de uso

Os diagramas de classes e de sequência por completo se encontram no Apêndice. Nessa seção, serão destacados os trechos mais importantes referentes às decisões de projeto.

SpaceShooters é a “main class” do sistema. Ela é responsável por inicializar todas as threads, aguardar pelo join, e deletá-las. Sendo assim, ela faz a ponte entre as classes do jogo e a biblioteca de threads desenvolvida. Um diagrama de classes à parte foi criado para representar essa ligação, assim como a estrutura da própria biblioteca.

Ao criar uma thread, SpaceShooter passa uma função que instancia o objeto associado à thread e inicia a sua execução. Cada thread fica em loop executando sua ação enquanto o jogo não houver terminado, chamando a função yield da biblioteca de threads após cada execução. Dessa forma, cada thread do sistema pode executar sua ação e ceder a CPU para a próxima thread.

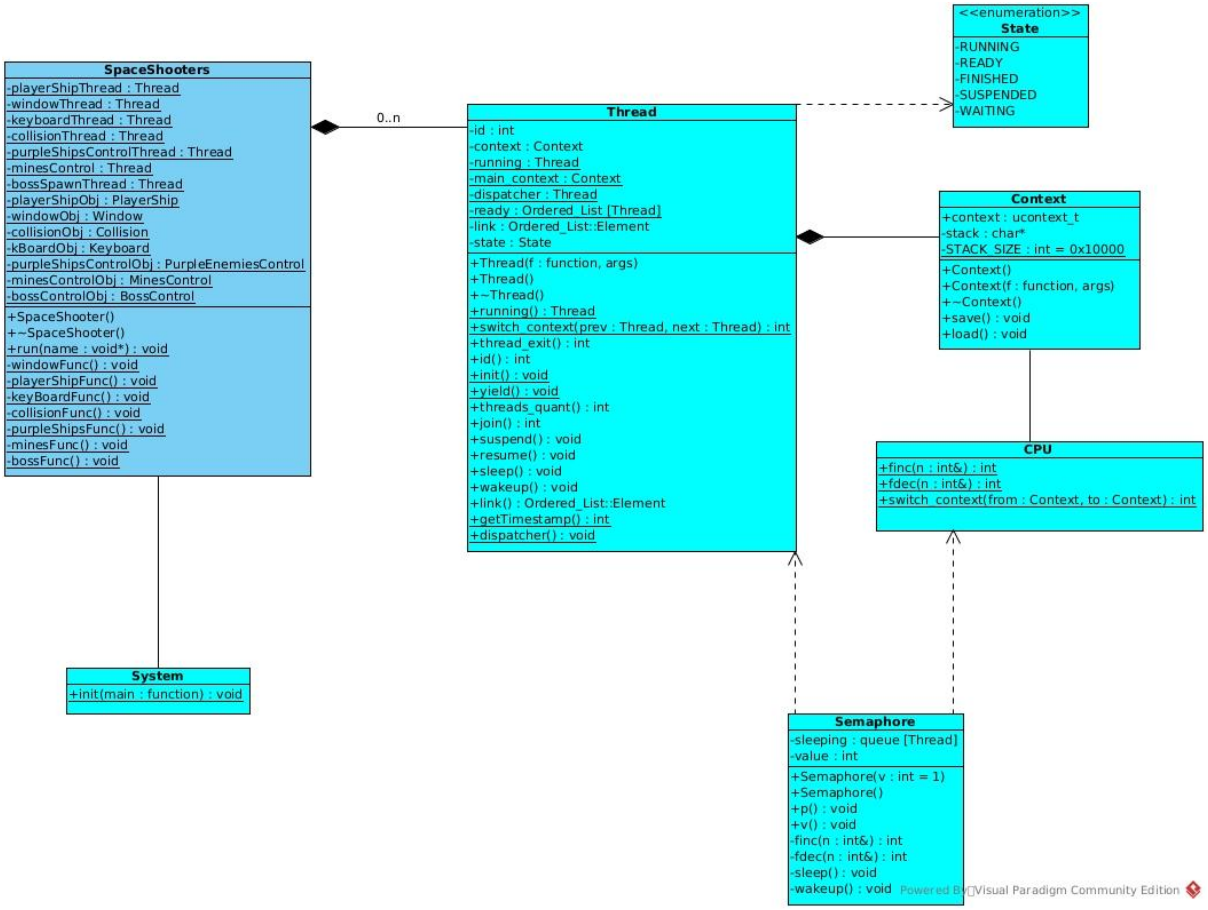

Diagrama 2. Diagrama de classes SpaceShooters com biblioteca de Threads

A classe Window é responsável por desenhar os objetos na tela. Para isso, ela mantém uma lista de objetos Drawable. A superclasse Drawable está no topo de uma hierarquia de herança. Um objeto de uma classe que herda de Drawable precisa ter implementado os métodos update, draw, isOutside, getSize e getPosition, visto que esses são os métodos utilizados por Window para desenhar na e fazer o controle da interface gráfica. Drawable então é subdividido em Hittable e Projectile, visto que objetos Hittable (como Enemy ou PlayerShip) funcionam de forma diferentes de Lasers e Missiles (que são Projectiles), por exemplo.

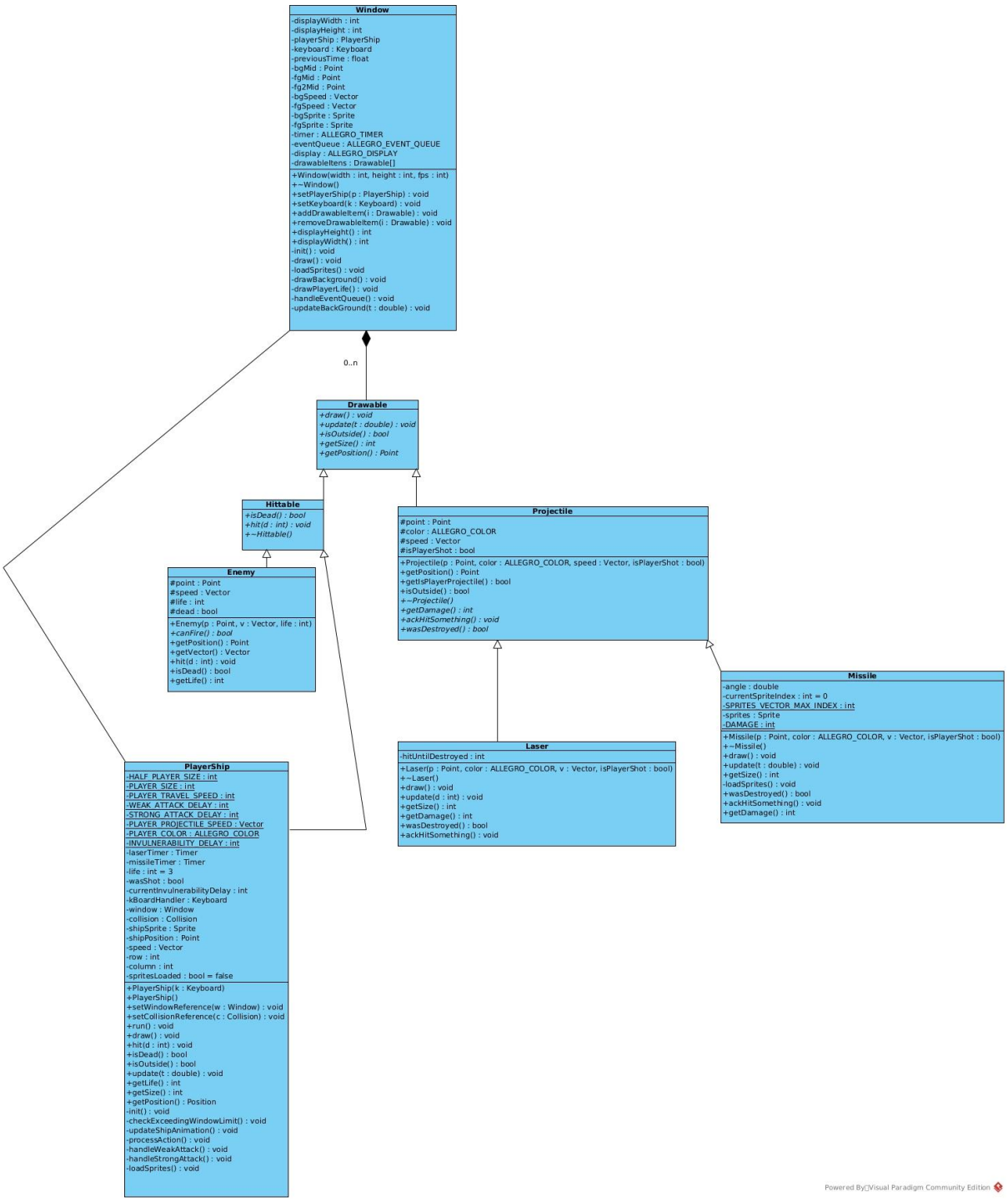

Diagrama 3. Interação Window/Drawable

A classe Collision é responsável por controlar as colisões do jogo. Para tal, ela possui 3 listas: de inimigos, de tiros inimigos, e de tiros do jogador, assim como uma referência para PlayerShip. A cada loop de

execução, ela percorre cada lista, verificando por colisões dos tiros do jogador com inimigos e dos tiros dos inimigos com o jogador e decrementando a vida do objeto atingido. Além disso, verifica se os inimigos e projéteis saíram da tela do jogo ou estão mortos para removê-los do sistema.

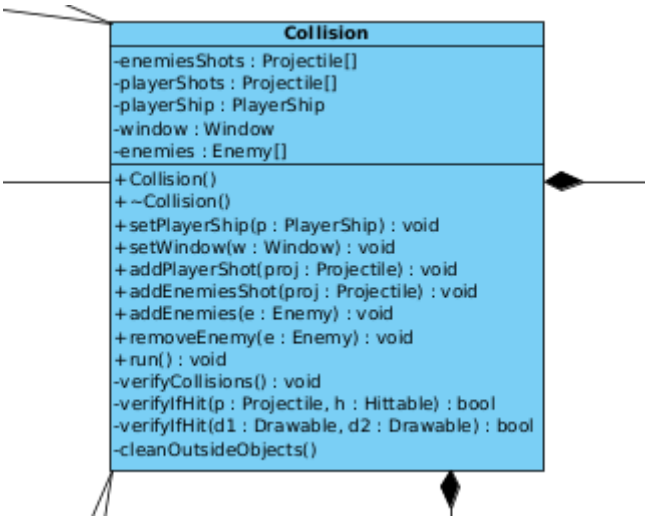

Diagrama 4. Classe Collision

Para a criação de inimigos, classes de controle foram utilizadas, responsáveis por instanciar os objetos no momento correto do jogo, colocá-los nas listas de Window e Collision, e mandar cada inimigo atirar de acordo com a sua lógica. Isso foi utilizado para PurpleEnemy (naves roxas), Mine (minas) e para o “Boss”. Essas 3 classes herdam de Enemy, classe já mostrada no diagrama 3. Como exemplo, classes PurpleEnemiesControl e PurpleEnemy:

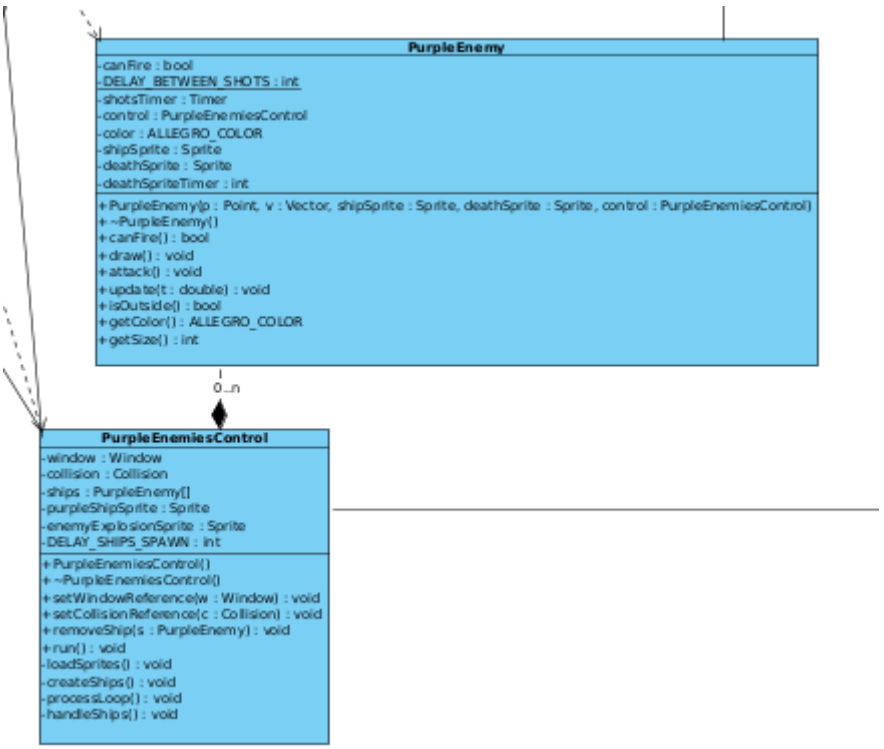

Diagrama 5. Classes PurpleEnemiesControl e PurpleEnemy

4. Apêndice

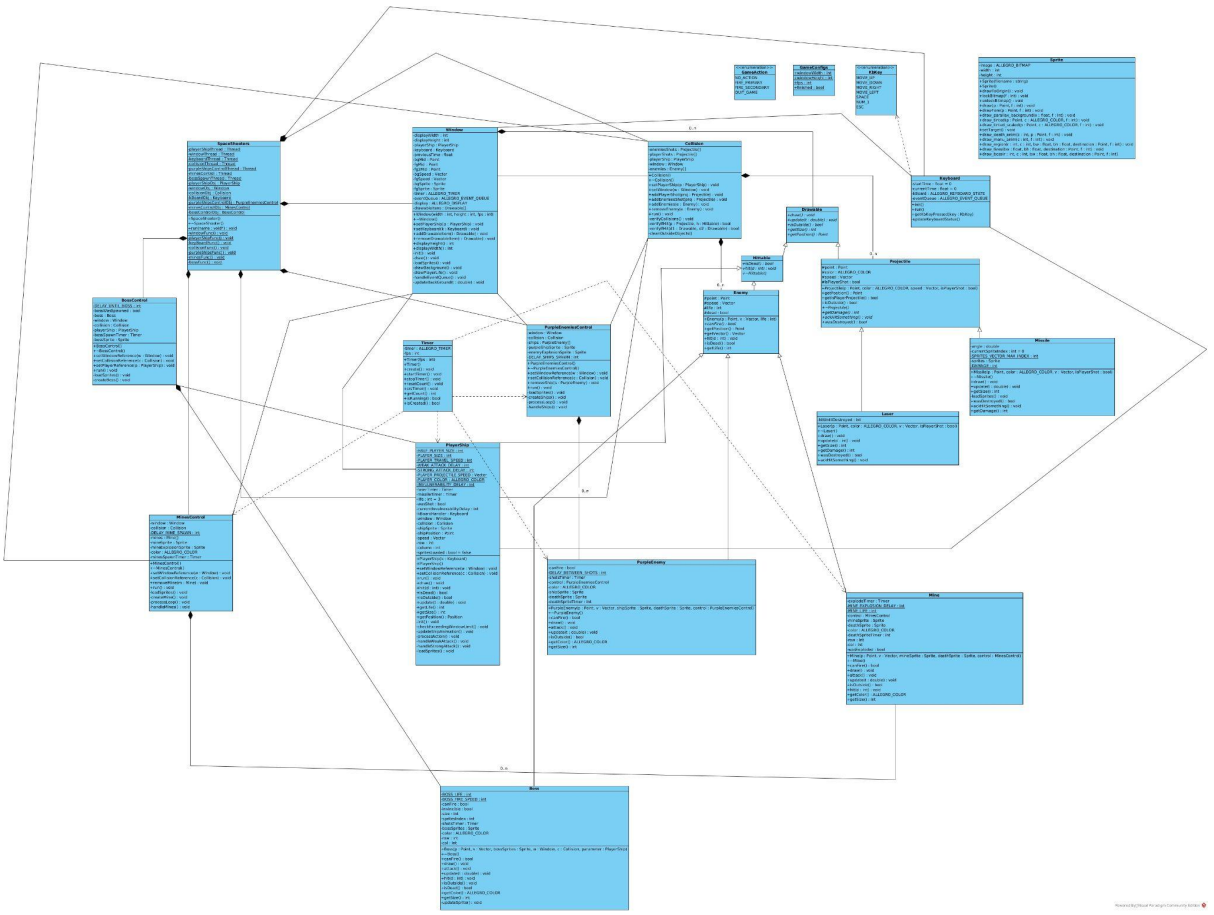

Diagrama de classes completo

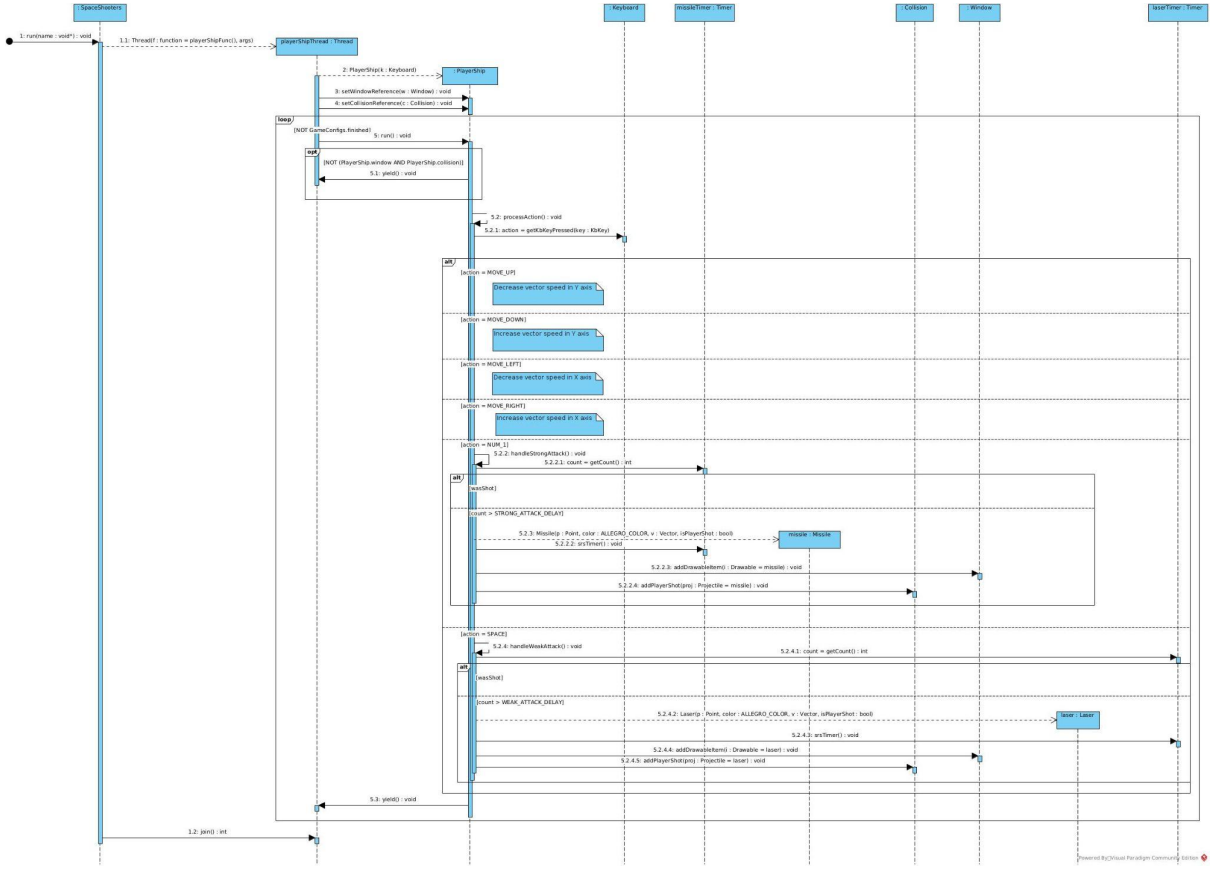

Diagrama de sequência para PlayerShip