UNIVERSIDADE FEDERAL DO PARANÁ

JONAS RIGAMONTI GIRARDI
OELSON THIBES DE CAMPOS SEGUNDO
SIDARTA NAKATANI
THIAGO MIGUEL LOPES SEQUEIRA SANTOS

SHMUP EDITOR: UM JOGO SHOOT 'EM UP COM EDITOR DE FASES

CURITIBA
2016

JONAS RIGAMONTI GIRARDI
OELSON THIBES DE CAMPOS SEGUNDO
SIDARTA NAKATANI
THIAGO MIGUEL LOPES SEQUEIRA SANTOS

SHMUP EDITOR: UM JOGO SHOOT 'EM UP COM EDITOR DE FASES

Trabalho apresentado à disciplina de Trabalho de Conclusão de Curso, do curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Setor Escola Técnica da Universidade Federal do Paraná.

Orientador: Professor João Eugênio Marynowski

CURITIBA
2016

RESUMO

Jogos eletrônicos existem em estilos variados, dentre eles um que tem mantido fãs fiéis desde a década 70 é o *shoot 'em up*, ou *shmup*, como também é conhecido. Shmup é um jogo em que o jogador atua em terceira pessoa como piloto de um veículo e tem como objetivo destruir o máximo de inimigos possíveis antes que suas vidas se esgotem ou que a fase chegue ao fim. A elaboração deste trabalho dá-se pelo desenvolvimento de um jogo eletrônico do estilo shmup com um editor de fases disponível para usuários (*level editor*). Mesmo sem conhecimentos de programação e de desenvolvimento de jogos os usuários poderão criar e alterar as fases do jogo. O compartilhamento das fases é feito através da importação e exportação de arquivos gerados pelo editor. Metodologias embasadas em métodos ágeis foram adotadas para o gerenciamento por serem mais comuns na área de desenvolvimento de jogos. Também foram utilizadas apenas ferramentas de acesso livre (*open source)*, como as bibliotecas e *frameworks* específicos para o desenvolvimento, assim como o jogo, Shmup Editor, se encontra disponível também com acesso livre tanto para jogadores quanto desenvolvedores.

Palavras-chave: jogos eletrônicos, *indie games*, *shoot 'em up, shmup, level editor, editor de fases, level maker*.

ABSTRACT

There are many styles of eletronic games, among them the one that has kept loyal fans since the 70's is the shoot 'em up, or shmup, as it is known. Shmup is a game where the player acts in the third person as a driver of a vehicle and aims to destroy as many enemies as possible before its life is exhausted or the phase's end. The formulation of this project is about the development of an electronic game shmup style with a level editor tool available for its users. Even with their lack of programming knowledge users can create and edit phases in the game. The sharing of phases is done through the import and export of files generated by the editor. For management methodologies were adopted grounded agile methods because they are more common in game development environments. For its coding were used open source tools, libraries and specific frameworks for development. The Shmup Editor is also available with free access for players and programmers.

Key-words: eletronic games, *indie games*, *shoot 'em up, shmup, level editor, level maker* .

SUMÁRIO

1. INTRODUÇÃO

Quando falamos de jogos eletrônicos sempre nos vem à cabeça jogos com campanhas e objetivos a serem alcançados, mas existe também um nicho de jogos que têm como diferencial e, pode-se dizer, ponto principal um editor de fases. Neste estilo de jogo o jogador pode desenvolver os próprios cenários que irá interagir durante jogo, desde a paisagem até o comportamento de seus inimigos. Para o sucesso dos jogos desse estilo as empresas distribuidoras de jogos oferecem ferramentas simples e intuitivas para que o jogador possa construir seu próprio mundo sem grandes dificuldades. Trata-se de um software com recursos específicos para criação de fases, também conhecido por *level editor*.

Grandes empresas de desenvolvimento de jogos, normalmente, não desenvolvem editores de fases para seus produtos. Algumas franquias famosas, como o DOOM, tornam seus arquivos “modificáveis” e disponibilizam sua documentação para que entusiastas possam editá-los e transformar as fases ou até criar uma nova (HALL, 1992). Essas modificações tornaram-se populares entre fãs dessas franquias, conhecidas como *Modification* ou *Mod*, é possível baixá-las para tornar o jogo customizado. Criar desafios para compartilhar com outros jogadores também se torna parte da diversão.

Com os recursos cada vez mais acessíveis e poderosos das ferramentas de compartilhamento de arquivos online, este tipo de jogo, onde as fases criadas por você podem ser compartilhadas, tende a chamar mais atenção.

A competitividade e o interesse aumentam quando, além de criar o nível, é possível que outros usuários possam jogar nessa fase. Muitas vezes, jogos simples, como jogos de celular, por exemplo, fazem sucesso pelo fato do objetivo ser apenas fazer mais pontos que os outros.

Com essa ideia de jogos simples de aprender, mas difíceis de dominar, um estilo de jogo que existe em várias gerações de vídeo games é o *shoot 'em up (ou shmup)*, onde o protagonista deve combater uma grande quantidade de inimigos atirando neles e desviando de seus ataques. Em muitos jogos desse gênero, o desafio é apenas fazer mais pontos.

O primeiro jogo shmup a fazer um grande sucesso foi *Space Invaders* (1978) e, desde então, jogos com essa temática vem acompanhando a evolução das

tecnologias que envolvem o mercado de jogos eletrônicos e, ao mesmo tempo, mantendo suas tradições de desafios e jogabilidade.

1.1. MOTIVAÇÃO

Apesar de ainda existirem grandes projetos que lançam jogos shmup, como Ikaruga e Touhou, a grande maioria, são com fases pré-definidas. Não é comum encontrar empresas deste ramo que lancem jogos já com uma ferramenta para criar fases.

Também seria interessante aproveitar a grande quantidade de recursos disponíveis para pequenos desenvolvedores e o crescente interesse em *indie games* - jogos desenvolvidos por uma única pessoa ou por pequenas equipes.

1.2. OBJETIVO GERAL

O Shmup Editor visa atender fiéis “adoradores” do estilo shmup e conquistar novos fãs com recursos para que os próprios jogadores possam desenvolver fases e compartilhá-las.

1.3. OBJETIVOS DO PROJETO

Este projeto objetiva desenvolver um software que possibilite a criação de fases para um jogo no estilo shmup sem necessidade de conhecimento em programação ou desenvolvimento de jogos. Pode-se ainda listar os seguintes objetivos:

- Aprofundar os conhecimentos no desenvolvimento de jogos;
- Utilizar metodologias ágeis para gerenciamento das tarefas durante o desenvolvimento;
- Identificar características dos jogos desse estilo;
- Utilizar recursos atuais para dar um diferencial à *engine* - é uma biblioteca ou um pacote de funcionalidades que são disponibilizadas para facilitar o desenvolvimento de um jogo e impedir que sua criação tenha que ser feita do zero (BOYER 2016);

- Desenvolvimento de uma *game engine* compatível com várias plataformas.

1.4. ORGANIZAÇÃO DO TEXTO

O texto deste trabalho está organizado da seguinte maneira. O Capítulo 2 (Fundamentação Teórica) apresenta os conceitos e questões relacionados ao estilo e desenvolvimento do jogo (*level editor*, *game design document* e *indie games)*. O capítulo 3 (Metodologia) descreve as técnicas de modelagem utilizadas na arquitetura do sistema, a seleção de ferramentas e recursos computacionais. Neste capítulo também constam as ferramentas de controle de projeto, como cronogramas, *backlogs, sprints* e mapa de atividades; seguidos pelo mapeamento das etapas de desenvolvimento do Shmup Editor. O capítulo 4 (Apresentação do Software), apresenta o sistema, abordando o processo de utilização, descrição das funcionalidades, telas e respectivos comportamentos. O capítulo 5 (Considerações Finais) lista as conclusões sobre os resultados obtidos neste trabalho e as expectativas para o futuro sistema. Os apêndices contêm todos os artefatos gerados durante o projeto, análise e homologação do Shmup Editor.

2. FUNDAMENTAÇÃO TEÓRICA

Ao se desenvolver um jogo deve-se fixar um público alvo para o mesmo. Hoje o misticismo de que os jogos são apenas para crianças já foi quebrado. Os jogos já alcançaram todas as idades e gêneros. A chegada dos dispositivos móveis com hardwares e softwares mais robustos também serviu como estopim para disseminar os jogos digitais entre as diversas faixas etárias.

De acordo com pesquisas realizadas as mulheres já são maioria no segmento com 52,6%. O perfil multi-plataforma – jogo que pode ser executado em diferentes dispositivos – se consolidou no Brasil e hoje alcançou 70,8% do mercado brasileiro. Os smartphones continuam sendo os mais populares com 77,2%, seguidos de computadores 66,9% e consoles 45,7% (Pesquisa Game Brasil 2016).

Neste capítulo serão abordados temas comuns ao desenvolvimento de jogos eletrônicos e importantes para a compreensão do escopo, funcionamento e finalidade do software proposto nesse trabalho. Serão apresentados os conceitos sobre o gênero do jogo, sobre a ferramenta para criação de níveis e sobre as metodologias utilizadas.

2.1. SHOOT 'EM UP

Também conhecido por shmup ou STG, é um subgênero de jogos de tiro (BUCHANAN, 2016). Em um jogo *shmup* o personagem embarca em um ataque solitário, normalmente em uma espaçonave ou aeronave. São jogos em que, normalmente, a visão é de cima pra baixo (*top-down*) ou lateral (*side-view*) e o jogador deve utilizar armas de longo alcance para que possa atacar a distância. O objetivo é atirar o mais rápido possível em qualquer ameaça enquanto desvia dos ataques inimigos, que são repetitivos e massivos (BIELBY, 1990). Em alguns jogos o personagem possuí uma barra de energia, podendo sofrer alguns danos antes de ser destruído, em outros, apenas um ataque (dano) basta. Esse jogo é conhecido por não ter um roteiro bem elaborado como os jogos da atualidade.

Basicamente, tem-se como enredo uma aeronave que precisa abater os seus inimigos até que nenhum mais sobre. Nenhum personagem complexo é apresentado e nenhum vídeo entre níveis se faz necessário para a criação de um objetivo para cada nível que o jogador alcança. Este estilo de jogo simplista conquistou muitos

jovens na década de 90. Alguns jogos dos jogos mais populares da época são Gaiares (1990), Thunder Force IV (1992), Phelios (1990).

As habilidades mais exigidas nesse gênero são reflexos rápidos e capacidade de memorizar os padrões de ataques dos inimigos (ASHCRAFT, 2009). Alguns jogos apresentam uma quantidade imensa de projéteis inimigos e o jogador tem que memorizar seus padrões para sobreviver.

2.2. EDITOR DE FASES

Editor de fases ou, como é mais conhecido, *level editor*, é uma ferramenta desenvolvida por programadores, que permite que o usuário/jogador crie níveis, mapas, campanhas personalizadas. Como por exemplo, o jogador pode definir o comportamento dos inimigos ou o cenário em que as batalhas se contextualizam. O atrativo de se ter um *level editor* é de que nenhum conhecimento em linguagens de programação ou em ferramentas avançadas é necessário para que o usuário customize seu jogo. O *level editor* é uma ferramenta de fácil utilização, o usuário apenas precisa 'arrastar e soltar' – do inglês *drag and drop* – durante o processo de desenvolvimento de uma nova fase.

Esse software pode estar integrado ao jogo como, por exemplo, um editor de circuito em um jogo de corrida, como também pode ser um software complementar e oferecido de forma separada. Os recursos são limitados para criar níveis apenas para um determinado jogo.

O *level editor* tem o intuito de incrementar e dinamizar o jogo, mas, em alguns casos, são os fãs que desenvolvem ferramentas com este propósito ao invés da empresa oficial.

Um dos primeiros jogos 3D que parte de sua fama se deve aos editores de níveis feitos por fãs, foi o Doom. A criação de vários editores independentes levou ao nascimento de uma comunidade online para troca desses mapas "não-oficiais". (SANCHES, 2016)

2.3. JOGOS ELETRÔNICOS INDEPENDENTES

Jogos eletrônicos independentes, também conhecidos como *indie games* devido ao termo em inglês (*independent video games*), são jogos criados por uma única pessoa ou uma pequena equipe, normalmente, sem qualquer patrocínio (GRIL, 2016). Por causa da crescente evolução das ferramentas de desenvolvimento de jogos e dos novos meios de distribuição online, os jogos independentes vêm ganhando cada vez mais destaque e atenção nos últimos anos.

A vantagem nesse tipo de projeto é que os desenvolvedores não precisam de aprovação do patrocinador e, com isso, não tem limites para criatividade (GRIL, 2016). Além disso, a equipe, por ser pequena, se envolve mais profundamente em todo o projeto. Tais características tornam os jogos independentes conhecidos por sua inovação, criatividade e experimentalismos (MCGUIRE, 2008).

Foi no final do século XXI, com a popularização da internet, que os jogos *indies* tiverem grande destaque. A facilidade de distribuição destes jogos através da rede foi um grande aliado e motivador para que vários desenvolvedores investissem nos jogos independentes. Sem a necessidade de arranjar um patrocinador para suporte, capital e, principalmente, divulgação dos seus jogos, os desenvolvedores *indies* começaram a se multiplicar no mercado global.

Um dos jogos mais conhecidos no cenário de jogos independentes é o Minecraft. Inicialmente criado por apenas uma pessoa, o sueco Markus Persson, o jogo se tornou febre entre os internautas. Markus começou a distribuir o jogo na internet antes mesmo do jogo estar na versão final. Como consequência ele foi adaptando as futuras atualizações do jogo de acordo com algumas sugestões dos próprios jogadores.

2.4. GAME DESIGN DOCUMENT

O *game design document* (normalmente abreviado como GDD) é o documento onde será registrado detalhadamente todas as atividades do projeto. Ele é criado e editado pela própria equipe de desenvolvimento, envolvendo designers, artistas e programadores, como um guia usado durante o projeto (OXLAND, 2004).

O documento deve ser produzido no estágio de pré-produção do desenvolvimento e, assim que o projeto for aprovado, alguns complementos são

adicionados para que possa servir de guia para os desenvolvedores. Devido ao dinamismo recorrente no desenvolvimento do jogo, esse documento sofre constantes alterações, revisões e expansões ao longo do projeto. Ele deve começar apenas com os conceitos básicos e, no final do projeto, estar completo com detalhes de cada aspecto do jogo (MOORE, 2009).

Um GDD pode ser feito de texto, imagens, diagramas, artes conceito, ou qualquer mídia que possa ilustrar os conceitos aplicados no jogo. Alguns documentos podem até incluir protótipos funcionais de algumas partes do jogo ou até tornar-se um recurso online para confecção colaborativa.

Para atender essas características, as abordagens mais coerentes de desenvolvimento são as metodologias ágeis, que tem como premissa entregar software adaptativo e de alta qualidade que pode ser construído por pequenas equipes, com uso de princípios como melhoria contínua do projeto e o desenvolvimento e testes baseados no rápido feedback e na aceitação de mudanças (SCRUM ALLIANCE, 2016).

Uma das metodologias mais populares é o SCRUM (SCRUM ALLIANCE, 2016), cuja aplicação envolve entregas ao final de cada ciclo que, geralmente, são curtos (ciclos semanais, por exemplo). Nessa abordagem, esses ciclos chamados de *Sprints*. As funcionalidades a serem implementadas são mantidas em uma lista que é conhecida como *Product Backlog*. No início de cada Sprint, faz-se uma reunião de planejamento na qual se prioriza os itens do *Product Backlog* e a equipe seleciona as atividades que ela será capaz de implementar durante o *Sprint* que se inicia. As tarefas alocadas em um *Sprint* são transferidas do *Product Backlog* para o *Sprint Backlog*.

2.5. DESENVOLVIMENTO DE JOGOS

O processo de desenvolvimento, produção ou design de um jogo inicia a partir de uma ideia ou um conceito. Frequentemente essa ideia é baseada em uma modificação ou combinação de conceitos já existentes (CHANDLER, 2008).

Na fase inicial, chamada de pré-produção, é preparado o GDD com a proposta do jogo que contenha informações sobre os seus conceitos básicos. O GDD contém informações como:

- Jogabilidade: a maneira pela qual o usuário irá interagir com o jogo, incluindo quais periféricos serão necessários (mouse, teclado, joystick).
- Funcionalidades básicas: como por exemplo, definir em quais momentos o jogo poderá ser salvo, se o jogo será *singleplayer* (apenas um jogador) ou *multiplayer* (mais de um jogador).
- Enredo: a história do mundo no qual o jogador se encontra.
- Público alvo: quais as necessidades básicas que o jogo deve atender para alcançar seu público.
- Requisitos não-funcionais: os pré-requisitos que são objetivos ou restrições estabelecidas por clientes e usuários do sistema que definem suas diversas propriedades.
- Estimativas: tempo e recurso.
- Arquitetura: plataforma, linguagem de programação, banco de dados e ferramentas em geral que serão utilizadas para o desenvolvimento do software.
- Orçamento.

Além do documento da pré-produção, é comum também que haja o desenvolvimento de um protótipo. O protótipo é útil para reproduzir visualmente um conceito empregado no jogo ou para testar algumas ideias e funcionalidades para todos os *stakeholders* - pessoa ou grupo que possui participação, investimento ou ações e que possui interesse em uma determinada empresa ou negócio (FREEMAN 2010).

A fase de produção inicia-se após a aprovação do projeto e pode ser dividida em cinco áreas:

- Design: consiste na elaboração dos elementos visuais do jogo e suas configurações, tais como telas de menu, layout dos itens apresentados nas telas do jogo, layout das fases pré-prontas e seus artefatos (veículos, tiros, inimigos, cenários).
- Programação: codificação dos requisitos levantados no início do projeto e criação dos fluxos lógicos. Nesta etapa o desenvolvimento fica voltado a *engine* do jogo, assim criando a base para que o jogo possa ser executado e testado.
- Criação de níveis: desenvolvimento focado na ferramenta de construção de fases (*level editor*), que é a parte mais complexa e trabalhosa deste projeto.

- Áudio: configuração dos áudios de som produzidos pelos objetos do jogo e trilhas sonoras.
- Testes: mesmo os testes sendo realizados no decorrer do projeto, após o desenvolvimento completo do jogo novos testes de qualidade são realizados para afirmar o correto funcionamento de todas as funcionalidades e fluxos lógicos do jogo.

3. METODOLOGIA

Este capítulo apresenta detalhes do gerenciamento e desenvolvimento do projeto.

Para que o trabalho ficasse mais próximo de um cenário de desenvolvimento de jogos, a equipe utilizou conceitos e práticas do GDD como um guia para o andamento do projeto. Alguns artefatos gerados durante as fases de elaboração deste software, como rascunhos de imagens de tela, podem ser consultados no APÊNDICE E. A criação iniciou-se com os conceitos macros e, conforme as definições ocorreram, foram registradas neste documento tornando-o assim, mais dinâmico, sofrendo alterações, revisões e expansões ao longo do projeto.

As entregas de protótipos funcionais ocorreram como forma de provas de conceito e testes de funcionalidades.

3.1. MODELO DE PROCESSO DE ENGENHARIA DE SOFTWARE

O processo de engenharia escolhido para o acompanhamento e evolução deste projeto foi uma metodologia ágil utilizada com a ferramenta Taiga, que permite gerenciar uma *backlog* de atividades e *sprints* de entregas.

As complexidades destas atividades foram definidas em um sistema de pontuação Fibonnaci, sendo a pontuação de complexidade mais baixa utilizada neste projeto a de número um (1) e de maior complexidade a de número vinte e um (21). (WATERS, 2012)

Para cada *sprint* de entrega são adicionadas pelos integrantes do grupo várias atividades que devem ser completadas em um curto período de tempo. Estas entregas são periódicas. Cada integrante é responsável por adicionar e completar suas próprias atividades. As atividades de codificação têm complexidade crescente, ou seja, a *sprint* atual sempre terá uma entrega mais complexa de funcionalidades do sistema que a *sprint* anterior.

3.2. PLANO DE ATIVIDADE

O plano de atividades para o desenvolvimento deste projeto foi dividido em seis etapas que foram executadas cronologicamente ao longo do desenvolvimento deste projeto. Estas etapas são descritas a seguir.

Na etapa de Análise ocorreu o levantamento de requisitos, que é fator primordial para o sucesso de qualquer software, foi discutido e fundamentado. Após definirmos o escopo do projeto e seus requisitos funcionais e não funcionais, escolhemos quais tecnologias iriamos usar para o desenvolvimento do projeto, comunicação entre os membros da equipe em reuniões não presenciais, ferramentas de compartilhamento de documentos, frameworks e controle de versão. Ao término da escolha das tecnologias envolvidas, todos os membros da equipe configuraram seus ambientes de desenvolvimento de acordo com as estruturas previamente escolhidas.

Gerenciamento do projeto: esta etapa de gerenciamento contempla o fator organizacional do projeto, por isso vários artefatos de suporte foram elaborados para um melhor entendimento da progressão do projeto. Em diversas reuniões a discussão de quais metodologias seriam usadas foi abordada até que todos os integrantes e o professor orientador entrassem em um consenso. O cronograma foi estipulado levando em conta a WBS construída, o tempo hábil que tínhamos para desenvolver o projeto, os cálculos do plano de risco e a divisão de trabalho entre a equipe.

Plano do Jogo: definimos qual o gênero e as características do jogo. Como o projeto é mais voltado para o desenvolvimento de fases e não sua jogabillidade, focamos na ferramenta de construção das fases e de quais ações os usuários teriam sob ela.

Design: os fluxos de telas e esboços de menus foram definidos para concretizar as ideias até então discutidas.

Desenvolvimento: primeiro o desenvolvimento foi da estrutura. Contendo os menus para a navegação dentro do jogo e suas funcionalidades. Em seguida a *game engine* foi desenvolvida, porque este é responsável por executar as fases e, assim, rodar o jogo. Depois o trabalho tomou outro foco que foi o de desenvolver a proposta ‘ambiciosa’ do nosso projeto, o desenvolvimento do *level editor*. Este último

é a ferramenta mais complexa e com maior interação com o usuário, por isso foi a fase do desenvolvimento que tomou mais tempo.

Testes: todas as funções do jogo e do construtor de fases (*level editor*) foram testadas em diversos cenários pelos integrantes da equipe a fim de encontrar e corrigir eventuais erros de funcionalidades, e atestar a qualidade do jogo.

3.3. PLANO DE RISCOS

Durante o processo de análise de riscos, a equipe identificou vários riscos e fez-se um julgamento sobre a probabilidade e o impacto dos mesmos, classificando-os conforme a Tabela 1.

		Impacto				
		Muito Baixo	Baixo	Moderado	Alto	Muito Alto
Probabilidade	Muito Alto	5	6	7	8	9
	Alto	4	5	6	7	8
	Moderado	3	4	5	6	7
	Baixo	2	3	4	5	6
	Muito Baixo	1	2	3	4	5

Tabela 1 - Matriz de Impacto x Probabilidade (MARTINS, 2007)

Baseado nessa classificação, foi possível identificar quais riscos eram necessários apenas controle (classificação entre 1 e 3), quais necessitavam um monitoramento mais ativo e, se possível, redução do risco (classificação entre 4 e 6) e quais eram necessários uma nova abordagem para o projeto (classificação maior ou igual a 7). Os riscos identificados pela equipe e suas classificações podem ser verificados na Tabela 2.

No	Condição	Consequência	Ação	Probabilidade	Impacto	Classificação
1	Pouco conhecimento das tecnologias envolvidas.	Atrasos no desenvolvimento do projeto.	Realizar estudos. Desenvolver protótipos.	Baixo	Moderado	4
2	Problemas de comunicação.	Acompanhamento ruim do andamento do projeto. Demora na detecção de impedimentos.	Reuniões. Softwares de gerenciamento online.	Alto	Alto	7
3	Desistência de algum membro da equipe.	Atraso na conclusão do projeto	Revisar o projeto (escopo) e divisão de tarefas.	Muito baixo	Alto	4
4	Não cumprimento dos prazos.	Atraso nas entregas.	Alocar recursos dedicados para as tarefas atrasadas.	Baixo	Baixo	3
5	Mudanças de tecnologias.	Atraso no desenvolvimento	Utilizar tecnologias já conhecidas.	Moderado	Moderado	5
6	Falta de precisão nos requisitos.	Ajustes / Mudanças de requisitos do jogo.	Reavaliar os requisitos para que contemplem todo o projeto.	Baixo	Alto	5
7	Excesso de mudanças dos requisitos.	Atraso nas entregas.	Refazer as atividades de analise e revisar todo o planejamento do projeto.	Baixo	Muito Alto	6
8	Má divisão de tarefas.	Sobrecarga de alguns membros.	Discutir as tarefas durante o planejamento do sprint e manter a ferramenta de gerenciamento atualizada.	Moderado	Moderado	5

Tabela 2 - Plano de Risco

3.4. RESPONSABILIDADES

O atual projeto é composto de quatro integrantes e foi decidido dividir o trabalho em duas competências principais, sendo elas desenvolvimento e documentação. Porém, todos os integrantes colaboraram igualmente na fase da análise de requisitos, testes, revisão geral da documentação e participação das tomadas de decisão no decorrer do projeto.

Como a documentação tende a ser uma parte extensa e um tanto quanto cansativa, e o desenvolvimento requer concentração e foco, optamos em enfatizar cada área e suas dificuldades, assim otimizando o uso do nosso tempo.

- Oelson: desenvolvimento da interface e codificação das funcionalidades, prototipação, elaboração dos diagramas, analise de requisitos e testes.
- Jonas: elaboração e organização da documentação, prototipação, elaboração dos diagramas, analise de requisitos e testes.
- Sidarta: desenvolvimento da interface e codificação das funcionalidades, prototipação, elaboração dos diagramas, analise de requisitos e testes.
- Thiago: elaboração e organização da documentação, prototipação, elaboração dos diagramas, analise de requisitos e testes.

3.5. RECURSOS DE HARDWARE

Para desenvolvimento do projeto foram utilizados apenas os computadores pessoais de cada integrante. Não houve necessidade de equipamentos ou ambientes especializados.

Seguem as configurações de cada equipamento:

Computador 1
Processador: Intel Core i5 2410 2.3GHz
Memória RAM: 6GB
Sistema Operacional: Windows 10

Computador 2
Processador: Intel Core i7 4770 3.5GHz
Memória RAM: 16GB
Sistema Operacional: Windows 10

Computador 3
Processador: Intel Core i7 2600K 3.40GHz
Memória RAM: 8GB
Sistema Operacional: Windows 10

Computador 4

Processador: Intel Core i5 5200U 2.2GHz

Memória RAM: 8GB

Sistema Operacional: Windows 8

3.6. RECURSOS DE SOFTWARE

Os softwares utilizados para desenvolvimento do Shmup Editor foram todos de código aberto. Para gerenciamento, foram utilizadas, em sua maioria, ferramentas gratuitas e online.

- **FlashDevelop [http://www.flashdevelop.org] -** Editor de código *open source* ideal para desenvolvimento em ActionScript (2 & 3) e Haxe.
- **Haxe [http://haxe.org] -** Ferramenta *open source*, compilador nativo multiplataforma com linguagem de programação de alto nível própria, Haxe (similar a Java, ActionScript3).
- **OpenFl [http://www.openfl.org] -** Open Flash Library, é uma implementação *open source* da API do Flash. Ao contrário da implementação da Adobe, OpenFl usa renderização de hardware, compila C++ nativo, alcança mais plataformas que o Adobe AIR, e usa a linguagem Haxe.
- **Flixel [http://flixel.org] -** Biblioteca *open source* específica para jogos. Para uso pessoal ou comercial.
- **HaxeFlixel [http://haxeflixel.com] -** União da linguagem Haxe e das bibliotecas OpenFL e Flixel. Permite criar jogos de forma fácil e grátis, além de compilar nativamente para várias plataformas a partir de uma única base de códigos.
- **Taiga [https://tree.taiga.io] -** Plataforma online de gerenciamento de projetos voltada ao desenvolvimento ágil.
- **Balsamiq [https://balsamiq.com] -** Ferramenta proprietária utilizada para modelar as interfaces. Foi utilizado a versão *trial* (30 dias) para criação dos layouts e a versão Web Demo para pequenos ajustes.
- **Astah Professional [http://astah.net/student-license-request] -** Ferramenta proprietária utilizada para criação dos diagramas de classe, casos de uso, e diagramas de sequência. Foi utilizado uma licença especial para estudantes válida por um ano (*Free Student Academic License*).

- **GitHub [https://github.com] -** Serviço de *Web Hosting* Compartilhado utilizado para controle de versão do código.
- **Textcraft [https://textcraft.net] -** Website gratuito para criação de textos customizados. Foi utilizado para criar a logotipo do jogo e alguns textos do jogo.
- **Millionthvector [http://millionthvector.blogspot.com.br/p/free-sprites.html] -** Blog com imagens de *sprites* compartilhadas para uso gratuito. Foi utilizado para definir a aparência dos veículos do jogo.

3.7. DESENVOLVIMENTO DO PROJETO

A ideia de desenvolver um jogo eletrônico surgiu do interesse da equipe em expandir o conhecimento nessa área. Como apenas um integrante tinha experiência profissional com isso, para os demais seria também um desafio.

Inicialmente, focou-se em um jogo online multi-plataforma. O estilo escolhido foi o *shmup*, devido ao sucesso que faz desde o início da indústria de jogos. Durante as discussões de funcionalidades, pensava-se em uma forma de envolver mais o jogador, além de completar missões. Assim, a equipe decidiu adicionar ao projeto um recurso em que a fase poderia ser criada. Tal opção possuía tanto potencial, que acabou tornando-se o próprio projeto.

De acordo com o escopo e as características principais do estilo do jogo, foram levantados os requisitos e analisados algumas ferramentas de desenvolvimento. Para aproveitar a experiência com desenvolvimento de jogos de um dos integrantes, a plataforma escolhida foi a HaxeFlixel pois, além da linguagem similar a ActionScript3 e Java, com uma única base de códigos é possível compilar para várias plataformas.

Para o gerenciamento, inicialmente dividiu-se o projeto em componentes menores utilizando a WBS. As estimativas iniciais, priorizações e divisões de tarefas foram planejadas utilizando o diagrama de Gantt. Porém, a atualização e manutenção se mostrou muito custosa e, com isso, a equipe decidiu adotar técnicas mais ágeis para o acompanhamento do desenvolvimento. Com intuito de facilitar esse gerenciamento, optou-se por uma plataforma online de gerenciamento de projetos (Taiga).

As reuniões com o orientador ocorreram quinzenalmente. Nelas, a evolução do trabalho era apresentada e os tópicos para a próxima reunião discutidos. Nesse momento a equipe aproveitava também para tirar as dúvidas referente ao andamento do projeto. No último mês, por considerarmos o período crítico, as reuniões foram semanais. Essas reuniões periódicas e a utilização do Taiga permitiram a equipe perceber durante o *sprint* quando alguma tarefa sofria atraso e, com isso, rediscutir prioridades, alocação de recursos e até escopo, conforme planejado no plano de risco.

3.8. CONSIDERAÇÕES SOBRE O DESENVOLVIMENTO

A falta de experiência da maioria da equipe com desenvolvimento de jogos ocasionou tarefas não previstas e falta de precisão na estimativa dos *backlogs.* Por ser uma área pouco aborda no curso, tarefas simples como a criação de diagramas, exigiram mais esforço do que o previsto, devido principalmente ao interesse da equipe em tornar o projeto o mais próximo possível de um ambiente de desenvolvimento de jogos.

Para a codificação, algumas bibliotecas prontas facilitam o trabalho, porém, alguns módulos precisavam ser bem customizados e, com isso, era necessário desenvolvê-los do zero. O módulo mais complexo e que demandou mais tempo de desenvolvimento neste projeto foi o editor de fases (*level editor*). Nele podemos citar as configurações de ondas de inimigos, que demandaram cálculos matemáticos para que seu posicionamento fosse proporcional independentemente do tamanho da tela de jogo usado pelo usuário.

Esses elementos refletiram em dificuldades com relação a arquitetura do jogo, fazendo com que a equipe precisasse rediscuti-la algumas vezes durante o desenvolvimento.

4. APRESENTAÇÃO DO SOFTWARE

Esse capítulo apresentará o manual de utilização do software contendo pré-requisitos para instalação, passo-a-passo para executar o jogo, orientações de como editar uma fase e de como jogá-la.

4.1. PRÉ-REQUISITOS

- Sistema Operacional: Sem restrições
- Leitor de CD (apenas para copiar a pasta do jogo para um disco local)
- Adobe Flash Player versão 22 ou superior
- Navegador compatível com flash (não é necessário caso utilize qualquer outro software que execute arquivos SWF)

4.2. INSTALAÇÃO

Não é necessário instalar o jogo, apenas copiá-lo para o disco local. A pasta com nome ShmupEditor, com todo seu conteúdo, deve ser inteiramente copiada para o disco local. Após a cópia, o CD pode ser retirado do drive.

Para iniciar o jogo, basta acessar a pasta no local em que foi copiada e executar o arquivo "ShmupSandbox.swf". Caso o computador não tenha um programa padrão associado para esse tipo de arquivo, o jogo não iniciará automaticamente. Nesse caso, é necessário abrir um programa compatível com flash (como por exemplo, o navegador Google Chrome) para abrir o arquivo. Assim que iniciado, o jogo carrega a tela inicial, cuja navegação será explicada adiante.

4.3. ENREDO

Você é um piloto da Galactic Federation Police, responsável por manter a ordem e a paz da galáxia e foi designado para uma missão em Tallon IV, um planeta dominado pelos Space Pirates, uma raça conhecida por saquear pequenos planetas eliminando todas formas de vida do local. Seu último ataque, o mais brutal que se tem notícia, foi no planeta Coruscant, uma zona neutra da Organização das Galáxias Unidas (OGU). Essa foi o estopim para a Galactic Federation Police lhe designar o

objetivo de matar todos os Space Pirates possíveis. Para isso, você dispõe de um veículo com a mais alta tecnologia. Capaz de absorver recursos de veículos inimigos, melhorando, assim, seu escudo e potência do tiro.

4.4. MANUAL

Ao iniciar o Shmup Editor a tela inicial é carregada, conforme a Figura 1, permitindo ao usuário selecionar qual módulo deseja acessar:

- *Play*: Carrega a tela que permite a seleção de uma fase ou importação de uma nova.
- *Stage Editor*: Acessa o editor de fase.
- *Options*: Apresenta as configurações do jogo, como volume e tamanho de tela.

Figura 1 - Tela Inicial

Após clicar em *Play*, na tela de seleção de fases, o usuário poderá selecionar uma fase que já está importada para iniciar o jogo (Figura 2) ou clicar em importar para adicionar uma nova fase na lista.

Figura 2 - Tela para selecionar / importar fase para jogar

Ao selecionar uma fase, o jogo é iniciado. A pontuação é exibida para que o usuário perceba em tempo real sua evolução.

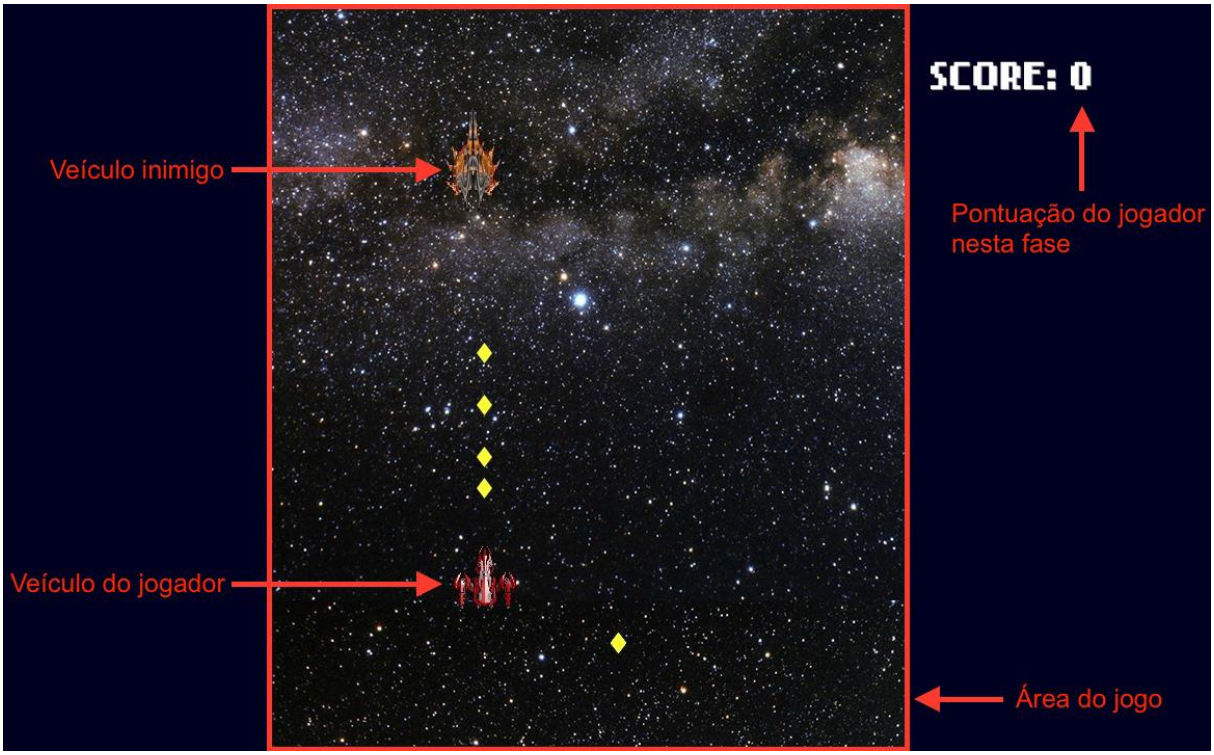

Figura 3 - Tela de uma fase em execução

Os controles utilizados para movimentar o veículo são as setas do teclado e, para atirar, a barra de espaço.

Figura 4 - Tela de informação dos controles (disponível em Options > Controls)

Caso o usuário queira criar ou editar uma fase, na tela inicial (Figura 1) deverá clicar na opção *Stage Editor*. O editor inicia conforme Figura 5 e o usuário já pode iniciar a criação. Para limpar as configurações feitas e iniciar a criação de uma nova fase, basta clicar em *New* (toda informação não salva será perdida).

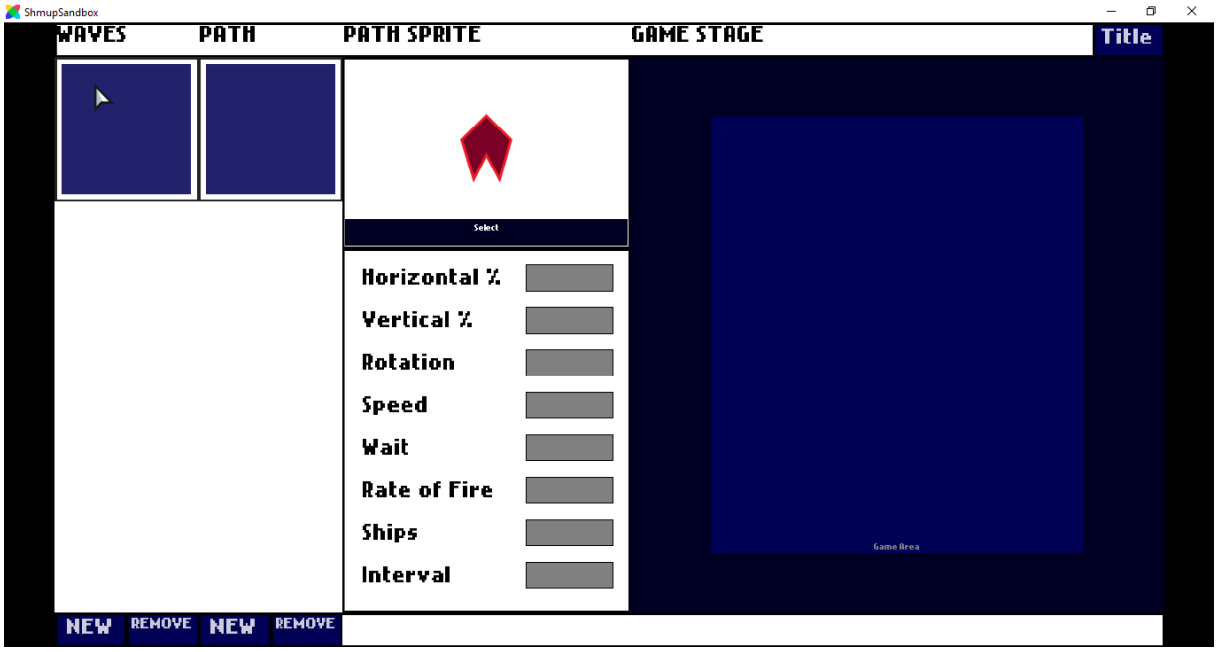

Figura 5 - Tela inicial do editor

Para editar uma fase existente é necessário primeiro carregá-la. Isso pode ser feito através do botão *Import*. Já para salvar a fase basta clicar em *Export* e escolher o local de destino do arquivo.

O usuário precisa conhecer os elementos que a compõem uma fase para começar a editá-la ou criar uma nova:

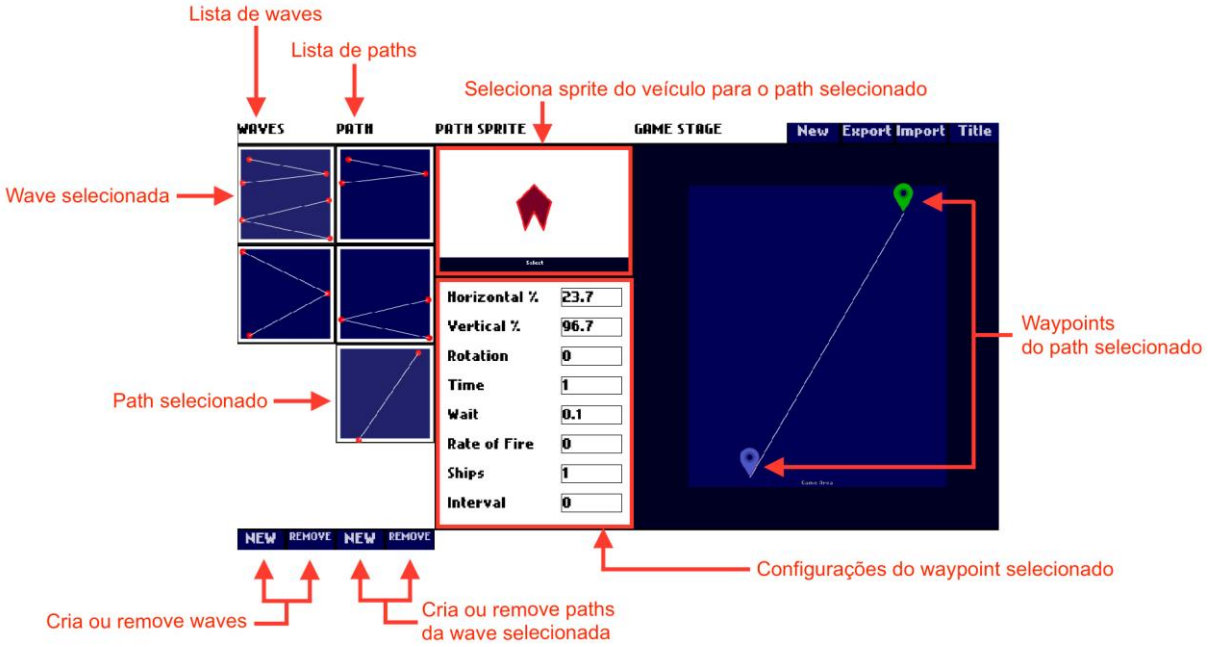

Figura 6 - Conceitos básicos para criação da fase

As fases do Shmup Editor são compostas por ***waves***. A ideia é que cada uma delas seja, realmente, uma onda de inimigos. Cada *wave* possuí vários ***paths*** e é em cada um deles que serão criados os ***waypoints*** para indicar o caminho pelo qual o inimigo irá seguir.

Cada *waypoint* contém informações que definirão o comportamento dos inimigos. Essa configuração é feita através dos parâmetros disponíveis na área de "Configurações do waypoint selecionado" (Figura 6). Os parâmetros *Horizontal %* e *Vertical %* indicam a posição do *waypoint*. *Rotation* é o ângulo que o inimigo irá rotacionar ao passar pelo *waypoint*. *Time* é o tempo que ele levará entre o atual *waypoint* e o próximo. *Wait* é o tempo de espera antes de ir para o próximo *waypoint*. *Rate of Fire* é a intensidade de tiros por segundo que será disparado a partir do *waypoint*. Cada parâmetro pode ser editado ao criar um novo *waypoint* ou ao selecionar um existente (clicando em cima). O *waypoint* selecionado é indicado pela cor azul.

O primeiro *waypoint* de um *path*, indicado na cor verde, deve conter também a quantidade de inimigos daquele *path* (*Ships*) e o intervalo em que eles serão inseridos na fase (*Interval*).

É possível criar um *waypoint*, para o path selecionado, apenas clicando na área "Game Stage" (Figura 6) e, para apagá-lo, é necessário clicar com o botão direito do mouse em cima do mesmo.

4.5. AMBIENTE DE DESENVOLVIMENTO

Para seguir o conceito de código livre, conforme todas as ferramentas utilizadas, fica disponível todo o código fonte do jogo no Github (https://github.com/Eniacc/TCC). Assim qualquer desenvolvedor interessado tem a possibilidade de fazer contribuições no código atual, ou até mesmo criar um novo jogo utilizando esta base como exemplo.

Para a criação do ambiente de desenvolvimento basta utilizar os softwares indicados na Seção 3.6.

4.6. CARACTERÍSTICAS TÉCNICAS

Nesta seção são apresentadas algumas características técnicas relacionadas ao desenvolvimento e codificação desde projeto.

4.6.1. ARQUIVO

Para armazenar dados referentes a uma determinada fase, ao invés da utilização de um banco de dados, o software foi implementado para armazenar, executar e interpretar arquivos JSON (*JavaScript Object Notation*). Essa notação é uma formatação leve de troca de dados e está baseado em um subconjunto da linguagem de programação JavaScript.

4.6.2. PROPORCIONALIDADE DA TELA

Para o posicionamento dos *waypoints* no editor não foi utilizada a posição fixa em *pixels* e sim a posição relativa a área do jogo em porcentagem. Isso permite que *waypoints* sejam criados fora da tela do jogo (para que naves saiam do campo de visão do jogador) além de executar a fase em resoluções diferente sem perder suas características.

5. CONSIDERAÇÕES FINAIS

Para o desenvolvimento deste projeto a equipe se reuniu para juntos fazer o levantamento de requisitos funcionais e não funcionais, escolher as tecnologias, traçar um planejamento de acompanhamento e desenvolvimento do projeto e, assim, executar as tarefas ao longo do projeto.

Ao término do desenvolvimento do projeto a nossa proposta de criar um jogo no estilo *shmup* com editor de fases foi alcançado. A equipe satisfez também seu interesse em tornar o andamento do projeto o mais próximo possível de um cenário comercial de desenvolvimento de jogos eletrônicos. Elementos como descritos no GDD foram utilizados como base para o trabalho, desde a concepção do produto até a criação de um manual de usuário.

A equipe se propôs a outros desafios, como utilizar o mínimo possível de tecnologias proprietárias com possibilidade de compilar para várias plataformas e, o mais ambicioso, criar um editor de fases ao invés de apenas o jogo, possibilitando ao usuário customizar toda a fase e compartilhá-la.

A utilização do Taiga permitiu que o gerenciamento ocorresse com características ágeis, cuja abordagem está presente no dia-a-dia dos integrantes. Mesmo com esse acompanhamento, nos deparamos com algumas dificuldades decorrentes de erros de planejamento feito pela equipe no início do projeto. A divisão do trabalho se mostrou ineficiente, pois no final do desenvolvimento um dos integrantes ficou sobrecarregado com a codificação do trabalho enquanto outros ficaram inertes à espera da conclusão da codificação para documentá-la. Uma melhor maneira de contornar esse erro seria ter envolvido todos os integrantes em todas as áreas do projeto, assim, tendo uma distribuição homogênea de tarefas.

Devido à sobrecarga de um integrante somada ao tempo perdido a equipe teve que optar pelo corte de algumas funcionalidades do jogo. Optamos por funcionalidades de baixo impacto na performance do jogo:

- Edição do veículo: posicionamento das armas e os tipos de tiro de cada arma.
- Importação de *sprites*: diferentes imagens para o veículo.
- Alteração do background: escolher, dentro de uma lista de imagens, um cenário para o jogo.

- Upgrades do veículo: diferentes níveis de tiros, velocidade e escudos de acordo com bônus temporários adquiridos durante o jogo.
- Inimigos de diferentes classes: inimigos com resistências mais altas e tiros com danos causados mais altos como chefes e subchefes.

Por isso citamos como melhorias futuras a implementação destas funcionalidades que constavam inicialmente no escopo do projeto. Além de atingir os objetivos do escopo original, também citamos como melhorias o banco de dados online que permite o compartilhamento das fases criadas pelos jogadores, e o *deploy* do jogo para outras plataformas, principalmente dispositivos móveis.

REFERÊNCIAS

ASHCRAFT, Brian. **Arcade Mania**: The Turbo-Charged World of Japan's Game Centers. Nova Iorque: Editora Kodansha USA, 2009

BATES, Bob. **Game Design**: Second Edition. Independence: Editora Cengage Learning, 2004

BIELBY, Matt. **The Complete YS Guide to Shoot 'Em Ups**. Londres: Revista Your Sinclair (edição 55), 1990

BOYER, Brandon. **Serious Game Engine Shootout**. Disponível em <http://www.gamasutra.com/view/news/12772/SGS_Feature_Serious_Game_Engine_Shootout.php>. Acesso em 16/06/2016

BUCHANAN, Levi. **Top 10 Classic Shoot 'Em Ups**. Disponível em <http://uk.ign.com/articles/2008/04/08/top-10-classic-shoot-em-ups >. Acesso em 10/04/2016

CHANDLER, Heather Maxwell. **The Game Production Handbook**: Second Edition. Burlington: Editora Jones & Bartlett Learning, 2008

FREEMAN, R. Edward. **Strategic Management:** A Stakeholder Approach. Cambridge: Editora Cambridge University Press, 2010

GRIL, Juan. **The State of Indie Gaming**. Disponível em <http://www.gamasutra.com/view/feature/132041/the_state_of_indie_gaming.php>. Acesso em 12/04/2016

MARTINS, J. C. C. **Gerenciando projetos de desenvolvimento de software com PMI, RUP e UML**. Rio de Janeiro: Brasport, 2007.

MCGUIRE, Morgan; JENKINS, Odest Chadwicke. **Creating Games**: Mechanics, Content, and Technology. Natick: Editora A K Peters, 2008

MOORE, Michael E.; NOVAK, Jeannie.
Game Development Essentials: Game Industry Career Guide. Independence: Editora Cengage Learning, 2009

OXLAND, Kevin. **Gameplay and design**. Boston: Editora Addison-Wesley, 2004

Pesquisa Game Brasil. Disponível em <http://www.pesquisagamebrasil.com.br>. Acesso em 18/04/2016

SANCHES, Bruno. **Os softwares de um jogo**. Disponível em <http://www.pontov.com.br/site/arquitetura/51-programacao/108-os-softwares-de-um-jogo>. Acesso em 18/06/2016

Scrum Alliance. Disponível em <http://www.scrumalliance.org/>. Acesso em 07/07/2016.

Shmups 101: A Beginner's Guide to 2D Shooters. Disponível em <http://www.racketboy.com/retro/shooters/shmups-101-a-beginners-guide-to-2d-shooters>. Acesso em 18/06/2016

WATERS, Kelly. **All About Agile:** Agile Management Made Easy.Editora: CreateSpace Independent Publishing Platform, 2012

WHITEHEAD, Jim. **Foundations of Interactive Game Design**. Santa Cruz: University of California, 2007

APÊNDICE A – DIAGRAMAS DE CASOS DE USO

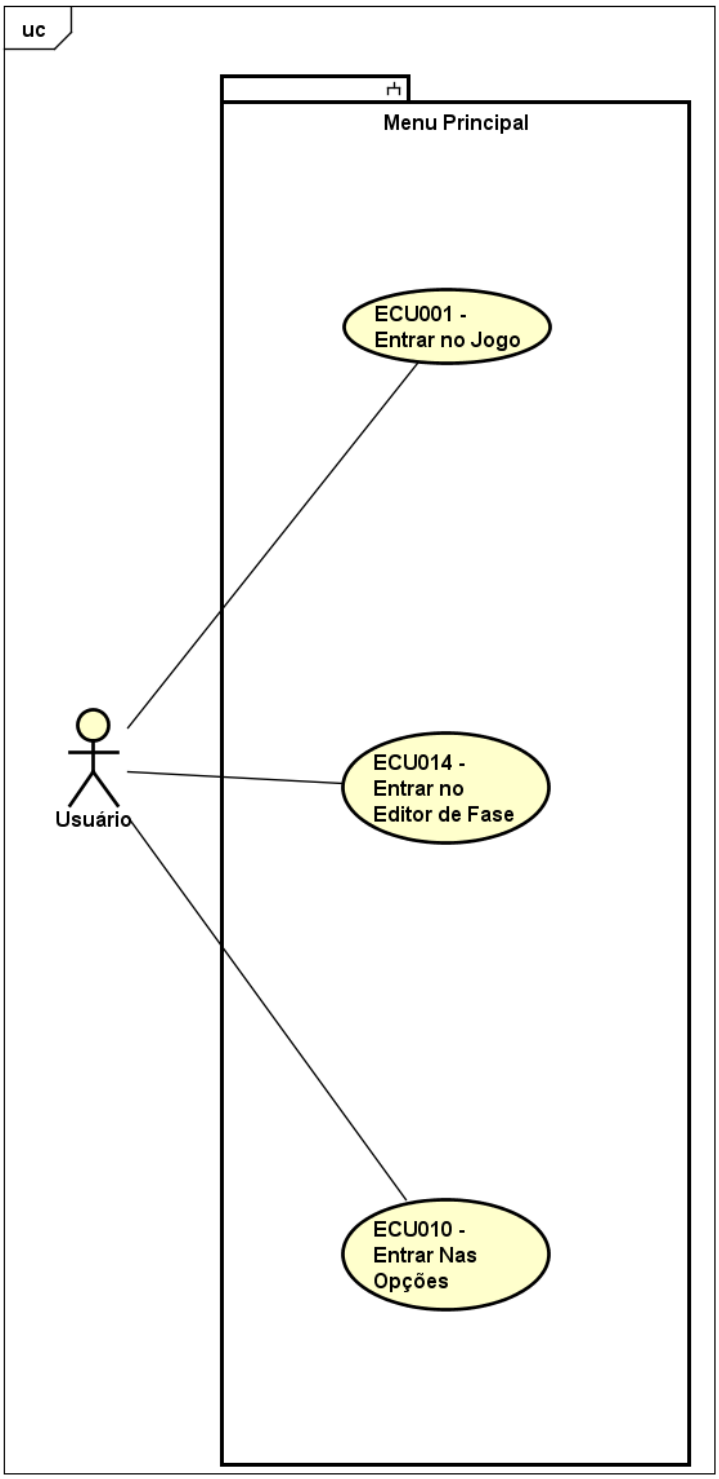

Figura 7 - Casos de Uso do menu principal

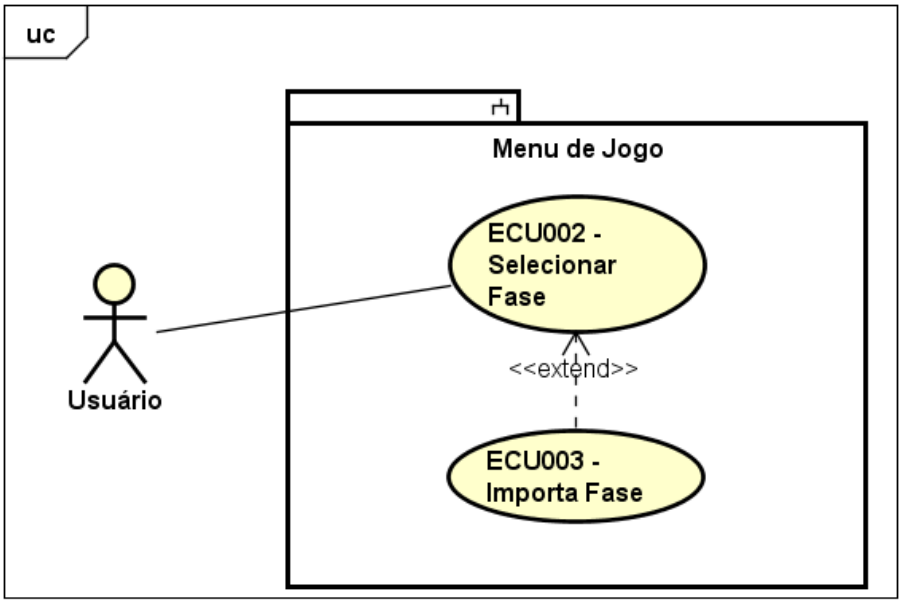

Figura 8 - Casos de Uso do menu de fases

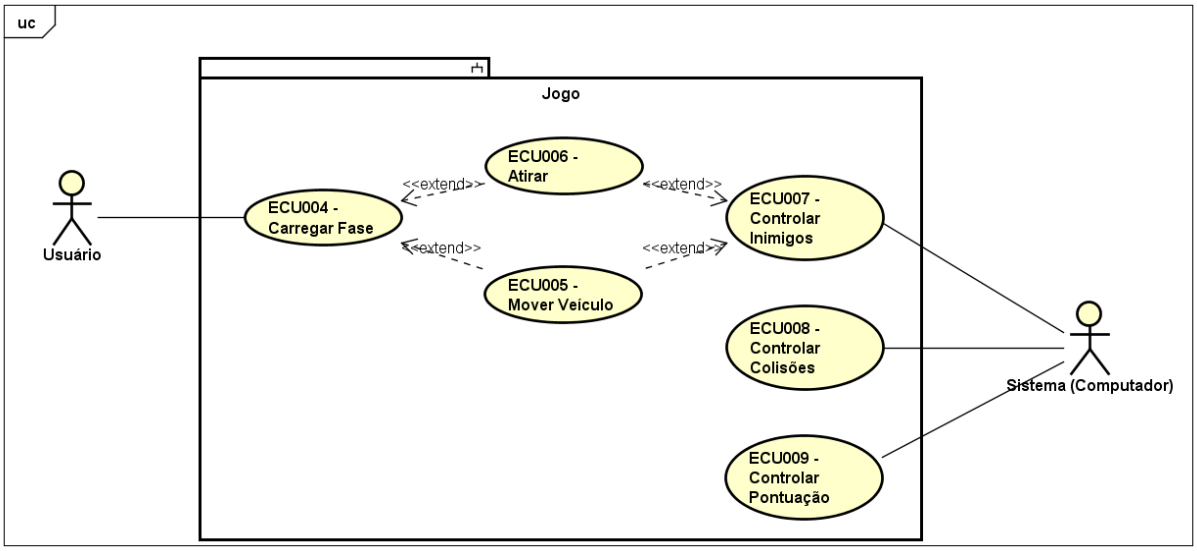

Figura 9 - Casos de uso do jogo

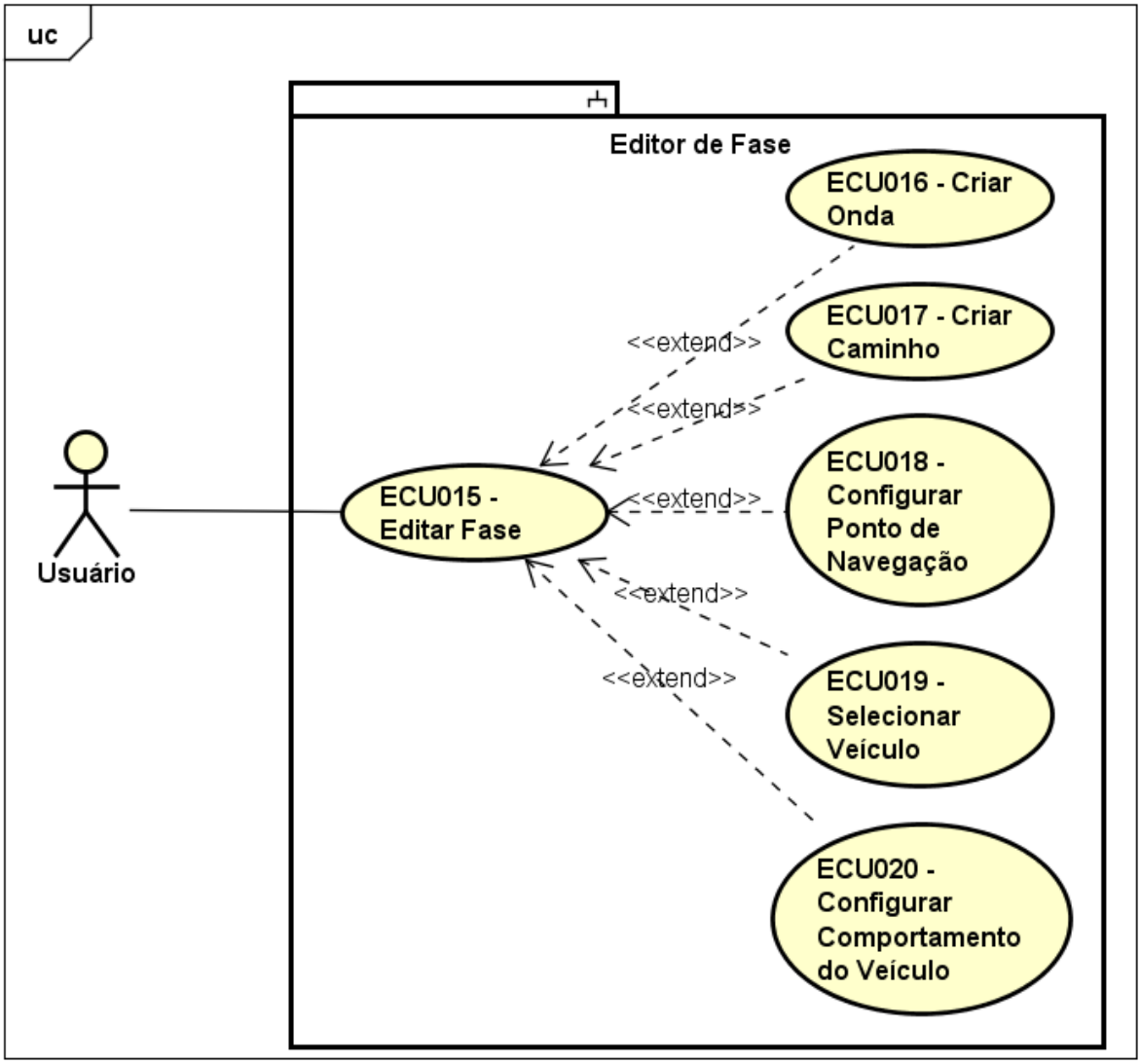

Figura 10 - Casos de Uso do editor de fases

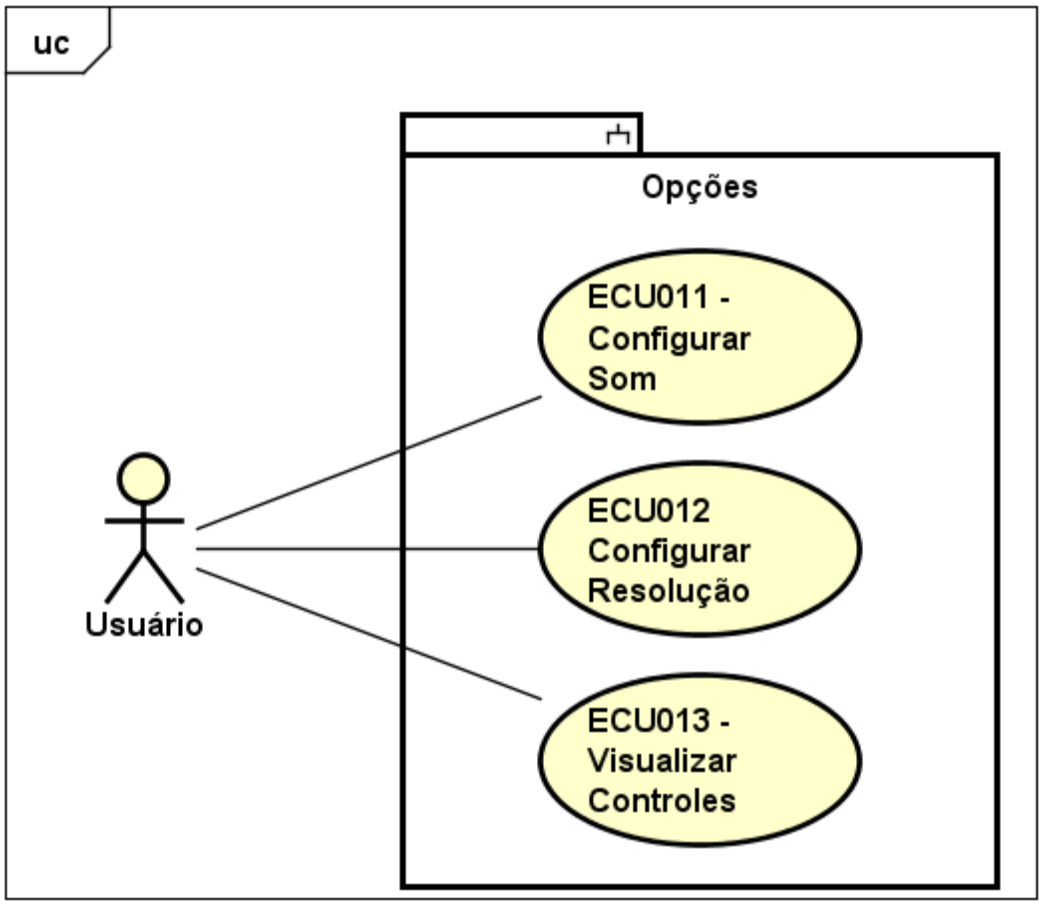

Figura 11 - Casos de uso das opções

APÊNDICE B – ESPECIFICAÇÃO DE CASO DE USO

ECU001 Entrar no Jogo

Descrição:

Usuário acessa o modo de jogo.

Data View

DV1 - Menu Principal

Figura 12 - Menu principal

Ator:

Usuário

Pré-Condição:

Nenhuma.

Pós-Condição:

Carregar lista de fases disponíveis.

Fluxo Principal

P1

1. Usuário inicia a aplicação.
2. O usuário clica no botão “Play”.
3. O sistema carrega o caso de uso Selecionar Fase.
4. O caso de uso é encerrado.

ECU002 Selecionar Fase

Descrição:

Lista de fases já importadas fica disponível para o usuário selecionar.

Data View

DV1 - Lista de fases

Figura 13- Lista de fases

Ator:

Usuário

Pré-Condição:

Uma importação já deve ter ocorrido.

Pós-Condição:

Carregar a fase selecionada pelo usuário.

Fluxo Principal

P1

1. Usuário acessa a lista através do menu.
2. Usuário navega pelas fases disponíveis.
3. Usuário escolhe uma fase. (A1)
4. Sistema carrega a fase.
5. Caso de uso é encerrado.

Fluxo Alternativo

A1 - Botão Importar Fase

1. Sistema carrega caso de uso Importar Fase.

ECU003 Importar Fase

Descrição:

Usuário seleciona arquivo de fase para importar para a aplicação.

Data View

DV1 - Importa Fase

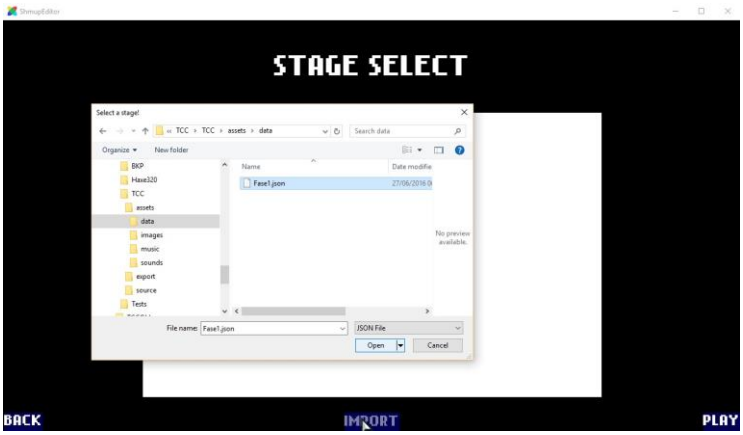

Figura 14 - Importa fase

Ator:

Usuário

Pré-Condição:

Nenhuma.

Pós-Condição:

Importar a fase para a aplicação e torná-la disponível na lista de fases.

Fluxo Principal

P1

1. Usuário clica no botão “Importar”.
2. Usuário navega pelos arquivos do seu dispositivo.
3. Usuário seleciona um arquivo.
4. Sistema valida o tipo do arquivo. (E1)
5. Sistema inicia a importação do arquivo. (A1)
6. Sistema informa o usuário que a importação foi feita.
7. Sistema atualiza a lista de fases disponíveis.
8. Caso de uso é encerrado.

Fluxo Alternativo

A1 - Fase já importada

1. Sistema informa ao usuário que fase já foi importada.
2. Sistema retorna para o passo 6.

Fluxo de Exceção

E1 - Arquivo Inválido

1. Sistema informa que tipo de arquivo é inválido.

ECU004 Carregar Fase

Descrição:

Aplicação inicia o jogo com a fase selecionada.

Data View

DV1 - Fase carregada

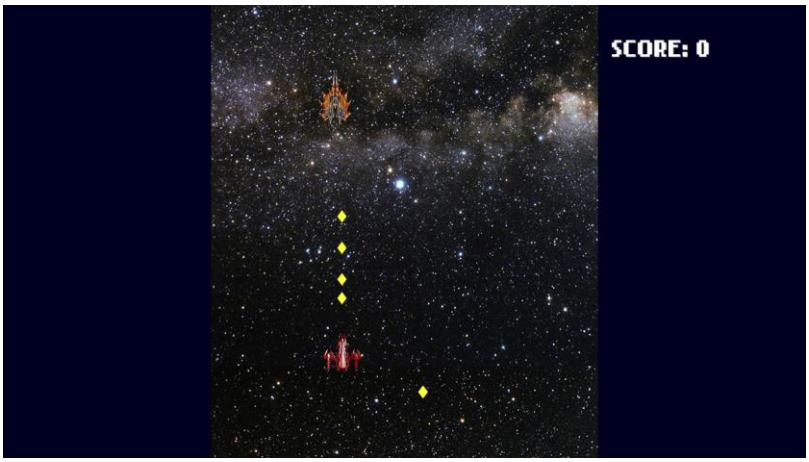

Figura 15 - Fase carregada

Ator:

Sistema

Pré-Condição:

Uma fase foi selecionada.

Pós-Condição:

Iniciar o jogo.

Fluxo Principal

P1

1. Usuário seleciona uma fase.
2. Sistema efetua a leitura dos componentes da fase.
3. Sistema carrega todos os componentes necessários.
4. Sistema inicia a fase. (E1)
5. Caso de uso é encerrado.

Fluxo de Exceção

E1 - Falha ao carregar

1. Sistema exibe mensagem de falha.

ECU005 Mover Veículo

Descrição:

Movimenta veículo conforme input.

Ator:

Usuário

Sistema

Pré-Condição:

Fase estar em execução.

Veículo não estar destruído.

Pós-Condição:

Veículo movimentado.

Fluxo Principal

P1

1. Usuário ou Sistema comanda movimentação do veículo.
2. Veículo executa o movimento comandado. (A1)
3. Caso de uso é encerrado.

Fluxo Alternativo

A1 - Limite do cenário.

1. Veículo não executa movimento caso tenha atingido o limite do cenário.

ECU006 Atirar

Descrição:

Veículo atira conforme input.

Ator:

Usuário

Sistema

Pré-Condição:

Fase estar em execução.

Veículo não estar destruída.

Pós-Condição:

Tiro é disparado.

Fluxo Principal

P1

1. Usuário ou sistema comanda um disparo.
2. Veículo executa o disparo.
3. Caso de uso é encerrado.

ECU007 Controlar Inimigos

Descrição:

Sistema executa ações dos veículos inimigos conforme definido no arquivo de fase.

Ator:

Sistema

Pré-Condição:

Fase estar em execução.

Pós-Condição:

Veículo inimigo executa ação definida.

Fluxo Principal

P1

1. Sistema comanda ação que a veículo inimiga deve executar.
2. Veículo inimigo executa ação. (A1)
3. Caso de uso é encerrado

Fluxo Alternativo

A1 - Veículo é destruído

1. Veículo inimigo está destruído.

ECU008 Controlar Colisões

Descrição:

Calcula se houve colisão entre objetos (projéteis e veículos).

Ator:

Sistema

Pré-Condição:

Fase estar em execução.

Pós-Condição:

Colisão detectada.

Fluxo Principal

P1

1. Sistema calcula colisão.
2. Colisão é detectada.
3. Sistema determina interação entre objetos
4. Caso de uso é encerrado.

ECU009 Controlar Pontuação

Descrição:

Sistema calcula pontuação do usuário.

Ator:

Sistema

Pré-Condição:

Fase estar em execução.

Pós-Condição:

Pontuação registrada.

Fluxo Principal

P1

1. Sistema calcula pontuação do usuário.
2. Pontuação é registrada.
3. Caso de uso é encerrado.

ECU010 Entrar nas Opções

Descrição:

Usuário tem as opções de jogo a serem configuradas na tela.

Data View

DV1 – Menu Principal

Figura 16 - Menu principal

Ator:

Usuário

Pré-Condição:

Usuário estar na tela de do menu principal.

Pós-Condição:

Sistema carregar dados da tela de opções.

Fluxo Principal

P1

1. Usuário clina no botão ”Options”.
2. Sistema carrega todos os componentes necessários.
3. Caso de uso é encerrado.

ECU011 Configurar Som

Descrição:

Usuário efetua as configurações de som.

Data View

DV1 – Opções

Figura 17 - Tela de opções

Ator:

Usuário

Pré-Condição:

Estar na tela de opções.

Pós-Condição:

Configurações de som alteradas.

Fluxo Principal

P1

1. Usuário clica no botão “+” ou “-” para ajustar o som. (A1)
2. Sistema grava e atualiza configurações de som.
3. Caso de uso é encerrado.

Fluxo Alternativo

A1 - Usuário clica no botão “Back”

1. Sistema volta para o menu principal.
2. Caso de uso é encerrado.

ECU012 Configurar Resolução

Descrição:

Usuário efetua as configurações de resolução.

Data View

DV1 – Opções

Figura 18 - Tela de opções

Ator:

Usuário

Pré-Condição:

Estar na tela de opções.

Pós-Condição:

Configurações de resolução alteradas.

Fluxo Principal

P1

1. Usuário clica no botão “Yes/No” para mudar a tela de ou para “full screen”. (A1)
2. Sistema altera configurações do sistema.
3. Caso de uso encerrado.

Fluxo Alternativo

A1 - Usuário clica no botão “Back”

1. Sistema volta para o menu principal
2. Caso de uso é encerrado.

ECU013 Visualizar Controles

Descrição:

Usuário visualiza as configurações do controle.

Data View

DV1 – Controles

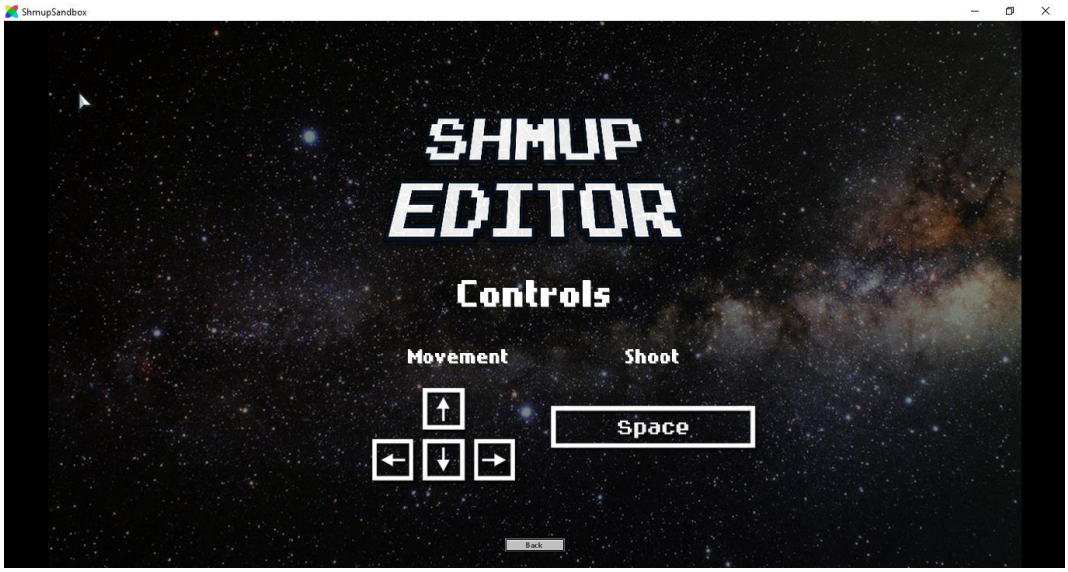

Figura 19 - Tela de controles

Ator:

Usuário

Pré-Condição:

Usuário ter clicado no botão “Controls”.

Pós-Condição:

Visualização dos controles do jogo.

Fluxo Principal

P1.

1. Usuário clica no botão “Controls”
2. Sistema carrega dados.
3. Caso de uso encerrado.

ECU014 Entrar no editor de fase

Descrição:

Usuário inicia a criação de fases customizadas.

Data View

DV1 – Editor de fases.

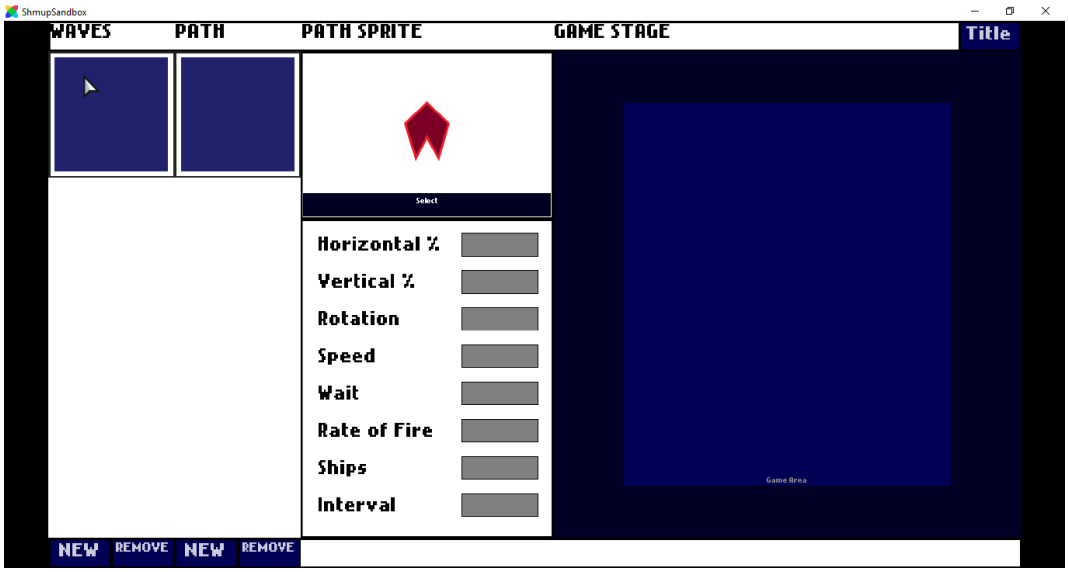

Figura 20 - Editor de fases

Ator:

Usuário.

Pré-Condição:

Usuário estar na tela de do menu principal.

Pós-Condição:

Sistema carregar dados da tela de edição de fase.

Fluxo Principal

P1.

1. Usuário clica no botão “Stage Editor”.
2. Sistema carrega todos os componentes necessários.
3. Caso de uso é encerrado.

Fluxo alternativo

A1.

1. Usuário clica no botão “Exit”.
2. Sistema é encerrado.
3. Caso de uso é encerrado.

ECU015 Editar fase

Descrição:

Usuário inicia a criação de fases customizadas.

Data View

DV1 – Editor de fases.

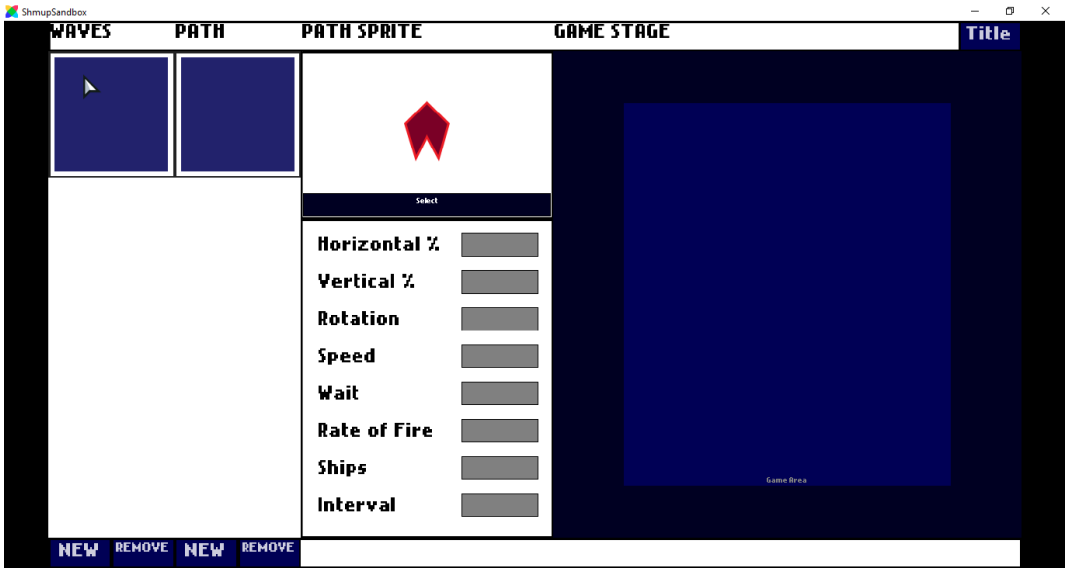

Figura 21 - Editor de fases

Ator:

Usuário.

Pré-Condição:

Usuário ter clicado no botão “Stage Editor”.

Pós-Condição:

Salvar o arquivo contendo a fase editada.

Fluxo Principal

P1.

1. Caso de uso é iniciado quando o usuário clica no botão “Stage Editor”.
2. Sistema carrega todos os componentes necessários.
3. Usuário edita sua fase como desejado.

   a. Caso de uso Criar Onda.
   b. Caso de uso Criar Caminho.
   c. Caso de uso Configurar Ponto de Navegação.
   d. Caso de uso Selecionar Veículo.
   e. Caso de uso Configurar comportamento do Veículo.
   f. Caso de uso Configurar opções gerais da fase.
4. Usuário clica no botão “Salvar” (E1)(A1)(A2)(A3)(A4).
5. Caso de uso é encerrado.

Fluxo alternativo

A1.

1. Usuário clica no botão “Cancelar”.
2. Caso de uso é encerrado.

A2.

1. Usuário clica no botão “Play”.
2. Sistema deixa a fase em modo de execução.
3. Usuário clica em ”Stop”.
4. Sistema retorna para o passo 3 do P1.

A3.

1. Usuário clica no botão “Exportar”.
2. Sistema abre uma janela para escolher o caminho e o nome do arquivo a ser exportado. (E1).
3. Caso de uso é encerrado.

A4.

1. Sistema avisa caso alguma wave esteja sem inimigos.
2. Usuário clica no botão “Salvar” mesmo assim (A5).
3. Caso de uso é encerrado.

Fluxo de Exceção

E1 – Erro ao salvar fase

1. Sistema exibe mensagem de falha.

ECU016 Criar Onda

Descrição:

Usuário cria onda de inimigos

Data View

DV1 – Editor de Fases com uma onda criada

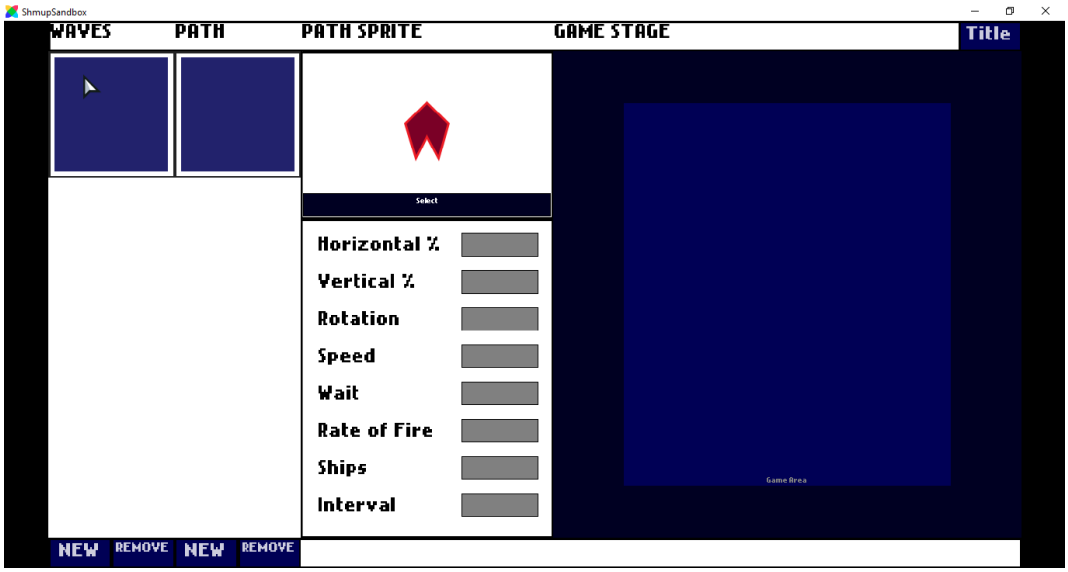

Figura 22 - Editor de fases com uma onda criada

DV2 – Editor de Fases com mais ondas criadas

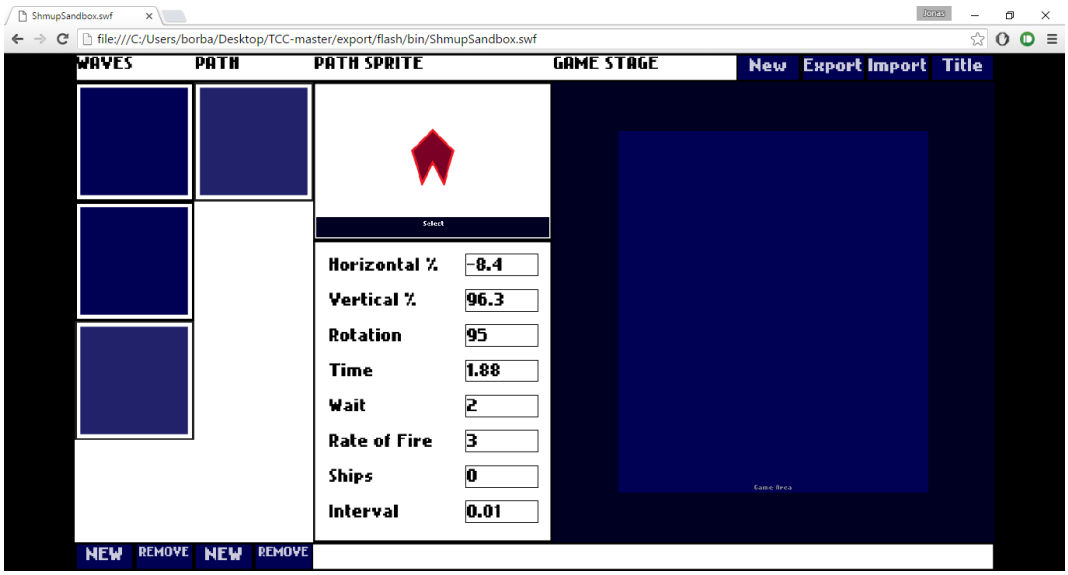

Figura 23 - Editor de fases com mais ondas criadas

Ator:

Usuário.

Pré-Condição:

Estar em modo de edição de fase.

Pós-Condição:

Salvar onda de inimigos na fase editada.

Fluxo Principal

P1.

1. Usuário clica no botão “New” para adicionar nova onda na fase
2. Usuário adiciona onda na fase. (A1) (A2)
3. Caso de uso é encerrado.

Fluxo alternativo

A1.

1. Usuário clica no botão “Cancelar”.
2. Caso de uso é encerrado.

A2.

1. Usuário seleciona ‘wave’ para ser removida.
2. Usuário clica no botão “Remove”.
3. Wave é removida.
4. Caso de uso é encerrado.

ECU017 Criar Caminho

Descrição:

Usuário caminho de inimigos

Data View

DV1 – Editor de Fases com uma onda e um caminho

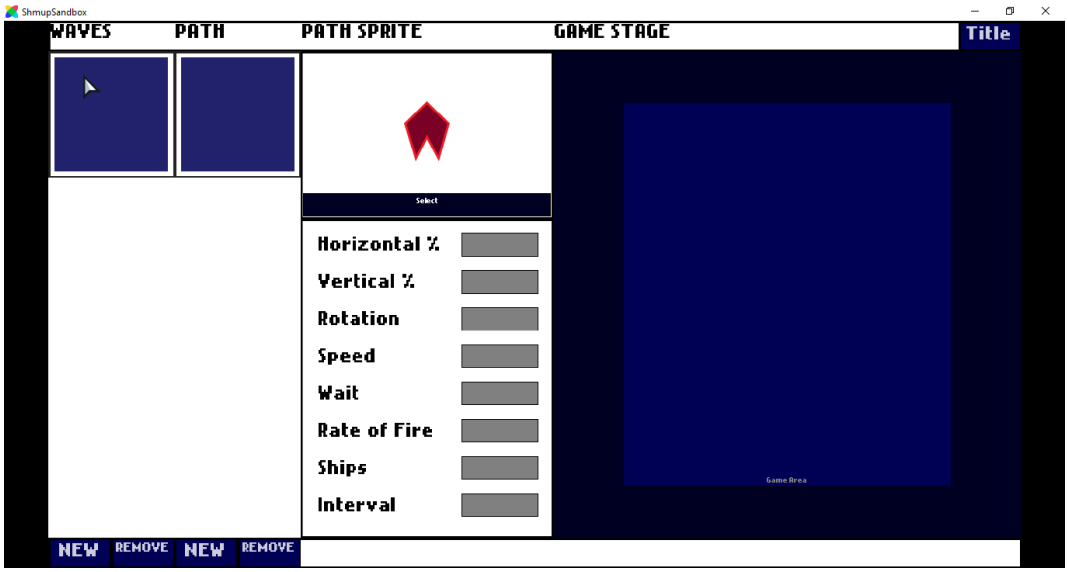

Figura 24 - Editor de fases com uma onda e um caminho

DV2 – Editor de Fases com novos caminhos criados em uma onda

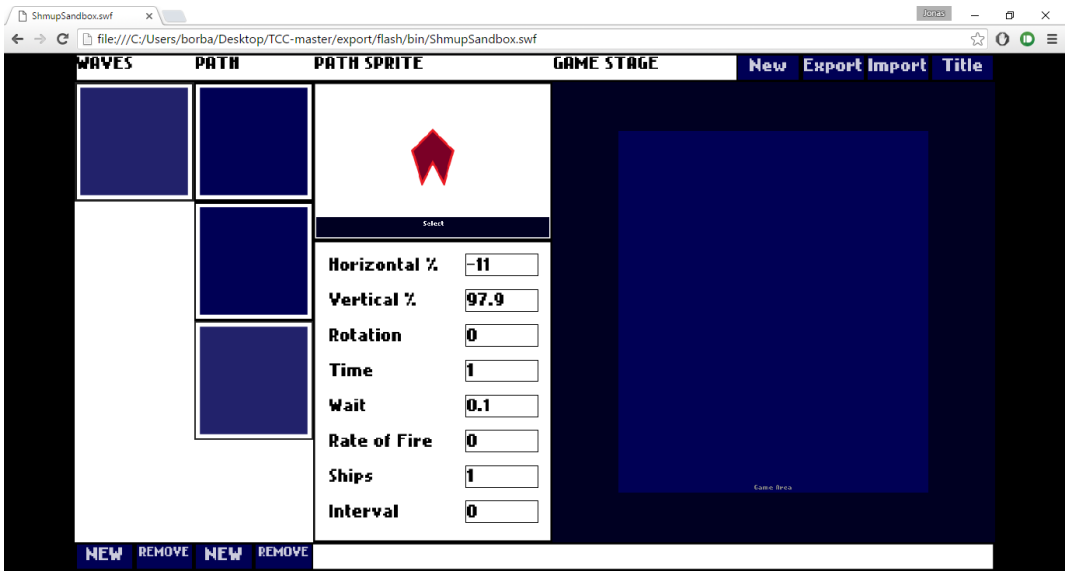

Figura 25 - Editor de fases com novos caminhos criados em uma onda

Ator:

Usuário.

Pré-Condição:

Estar em modo de edição de fase.

Pós-Condição:

Salvar caminho de inimigos na fase editada.

Fluxo Principal

P1.

1. Usuário clica em “New” para adicionar novos pontos de caminho de inimigos.
2. Usuário adiciona caminho na fase. (A1) (A2)
3. Caso de uso é encerrado.

Fluxo alternativo

A1.

1. Usuário clica no botão “Cancelar”.
2. Caso de uso é encerrado.

A2.

1. Usuário seleciona ‘Path’ para ser removida.
2. Usuário clica no botão “Remove”.
3. Path é removido.
4. Caso de uso é encerrado.

ECU018 Configurar Ponto de Navegação

Descrição:

Usuário configure pontos de navegação

Data View

DV1 – Editor de Fases com pontos de navegação criados no caminho selecionado

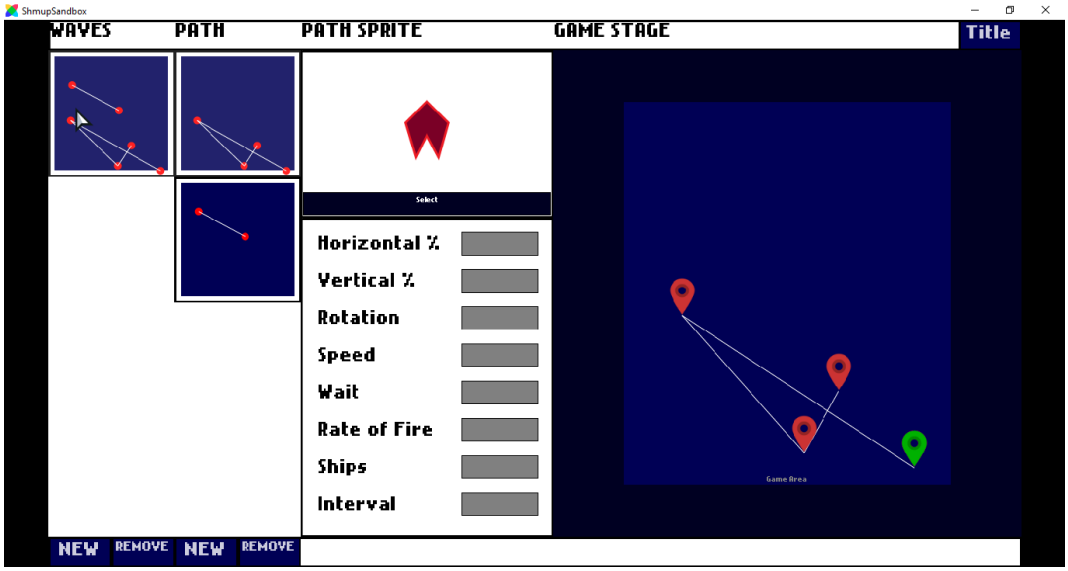

Figura 26 - Editor de Fases com pontos de navegação criados no caminho selecionado

Ator:

Usuário.

Pré-Condição:

Estar em modo de edição de fase.

Ter selecionado um path.

Pós-Condição:

Salvar pontos de navegação na fase editada.

Fluxo Principal

P1.

1. Usuário adiciona os pontos navegação. (A1) (A2)
2. Caso de uso é encerrado.

Fluxo alternativo

A1.

1. Usuário clica no botão “Cancelar”.
2. Caso de uso é encerrado.

A2.

1. Usuário seleciona ponto de navegação para ser removida.
2. Usuário clica com o botão direito do mouse.
3. Ponto de navegação é removido.
4. Caso de uso é encerrado.

ECU019 Selecionar Veículo

Descrição:

Usuário seleciona veículos do jogo

Data View

DV1 – Level Editor.

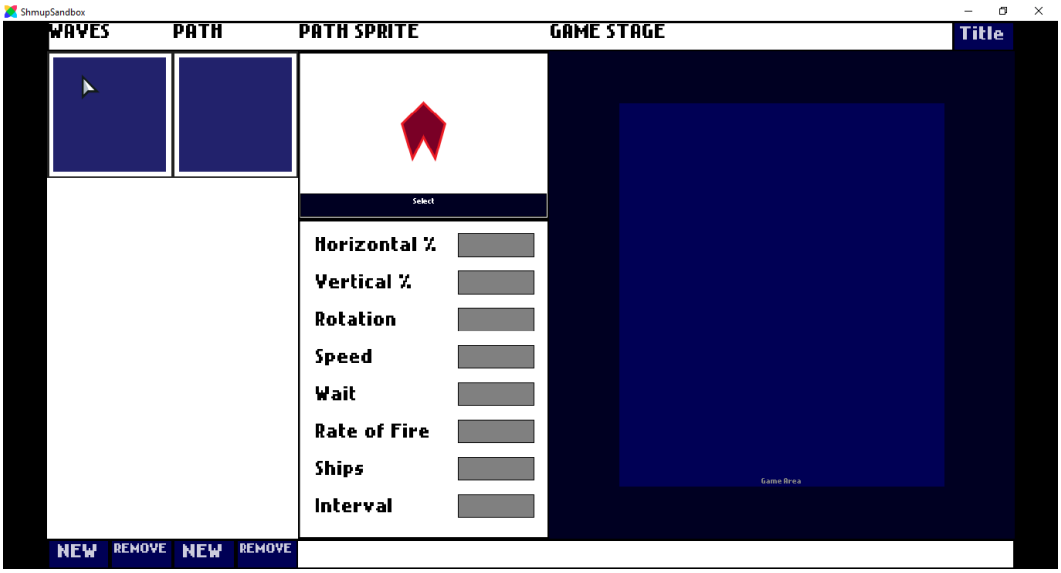

Figura 27 Editor de fases

Ator:

Usuário.

Pré-Condição:

Estar em modo de edição de fase.

Pós-Condição:

Salvar veículos na fase editada.

Fluxo Principal

P1.

1. Usuário clica no botão ‘Select’.
2. Sistema carrega uma janela de navegação para selecionar as sprites. (A1) (A2)

3. Usuário seleciona imagem e clica no botão ‘Incluir’.
4. Sprite é adicionada na fase.
5. Caso de uso é encerrado.

Fluxo alternativo

A1.

1. Usuário clica no botão “Cancelar”.
2. Caso de uso é encerrado.

ECU020 Configurar Comportamento do Veículo

Descrição:

Usuário configure comportamento dos veículos

Data View

DV1 – Level Editor.

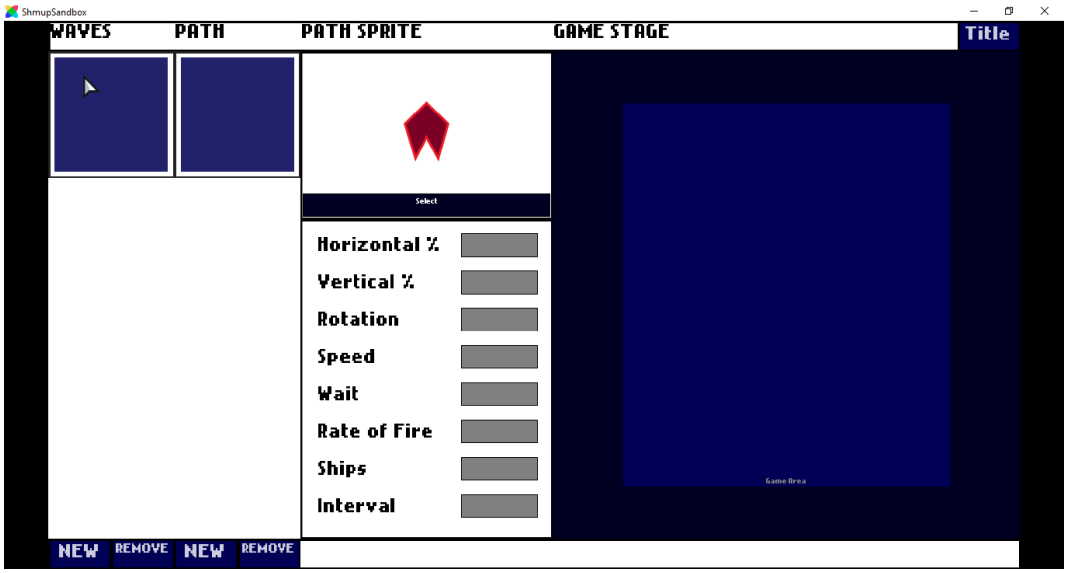

Figura 28 - Editor de fases

Ator:

Usuário.

Pré-Condição:

Estar em modo de edição de fase.

Pós-Condição:

Salvar comportamento dos veículos.

Fluxo Principal

P1.

1. Usuário seleciona o ‘way point’ e adiciona seu comportamento.(A1) (A2)
2. Usuário configura campo de texto “Horizontal%”.

3. Usuário configura campo de texto “Vertical%”.
4. Usuário configura campo de texto “Rotation”.
5. Usuário configura campo de texto “Speed”.
6. Usuário configura campo de texto “Wait”.
7. Usuário configura campo de texto “Rate of Fire”.
8. Usuário configura campo de texto “Ships”.
9. Usuário configura campo de texto “Interval”.
10. Caso de uso é encerrado.

Fluxo alternativo

A1.

1. Usuário clica no botão “Cancelar”.
2. Caso de uso é encerrado.

APÊNDICE C – DIAGRAMA DE CLASSES

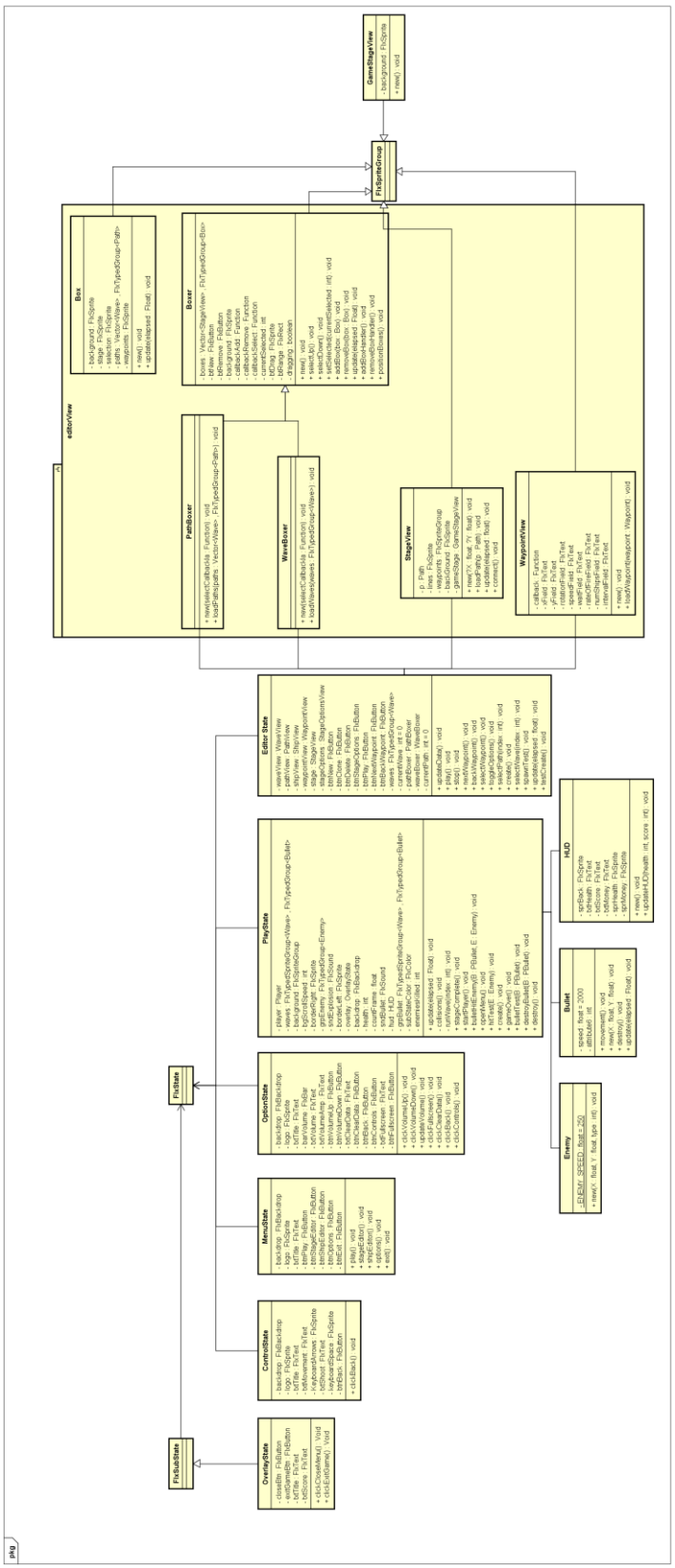

Figura 29 - Diagrama de Classes

APÊNDICE D – DIAGRAMAS DE SEQUÊNCIA

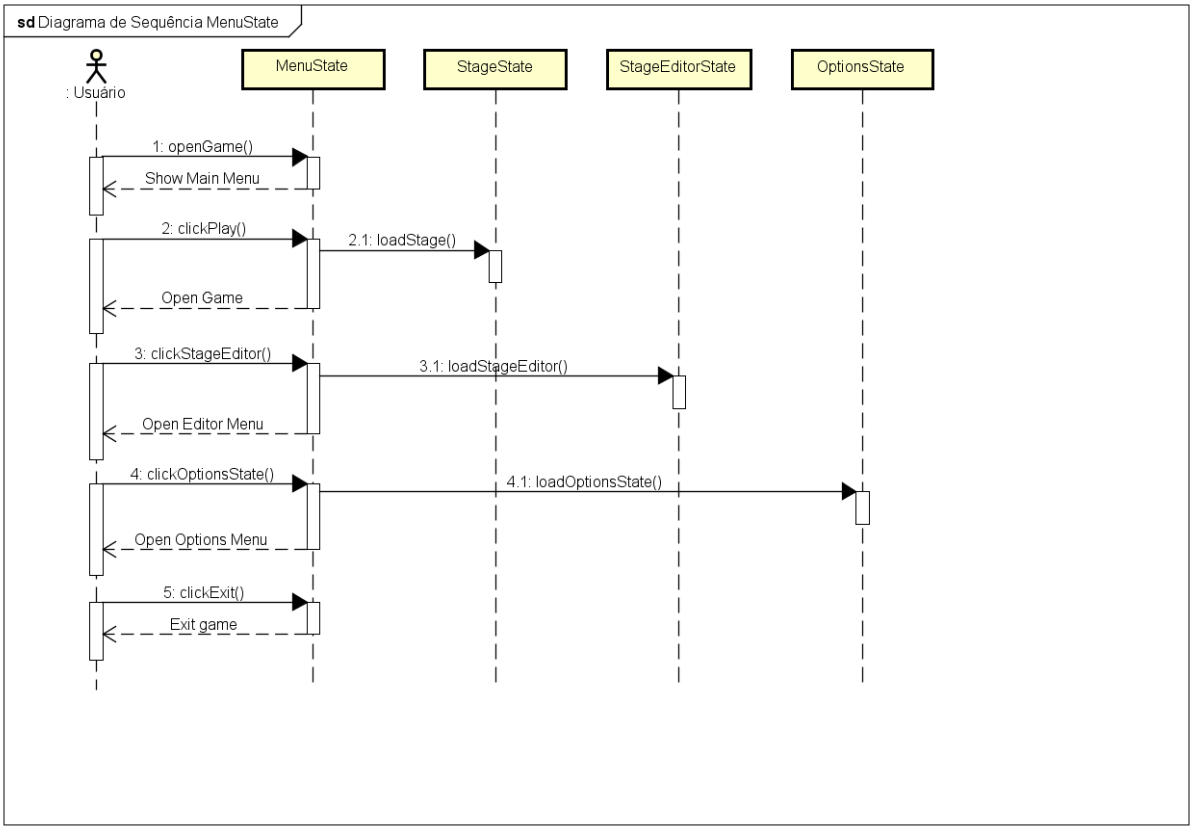

Figura 30 - Diagrama de sequência da navegação do meu

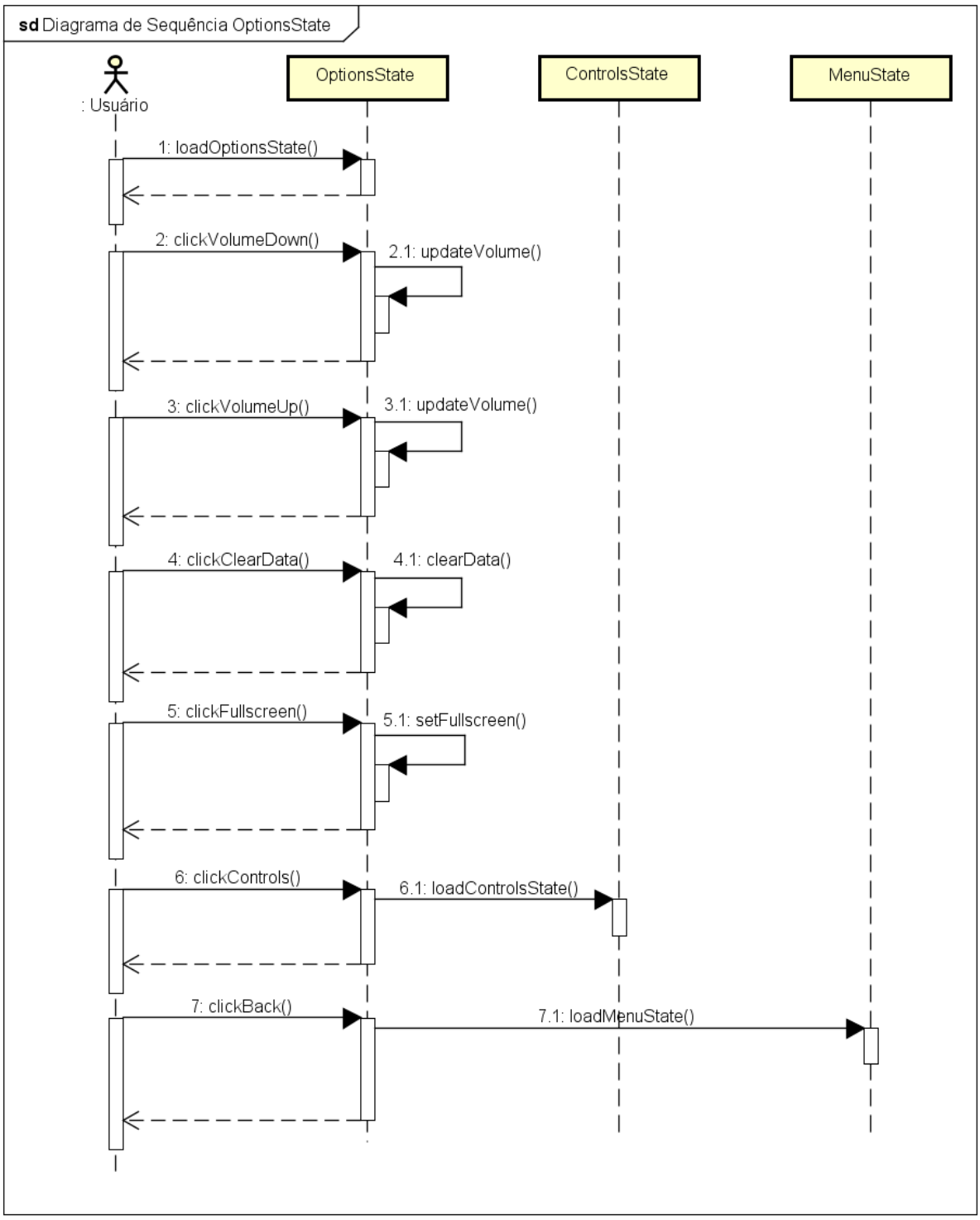

Figura 31 - Diagrama de sequência das configurações disponíveis em "Options"

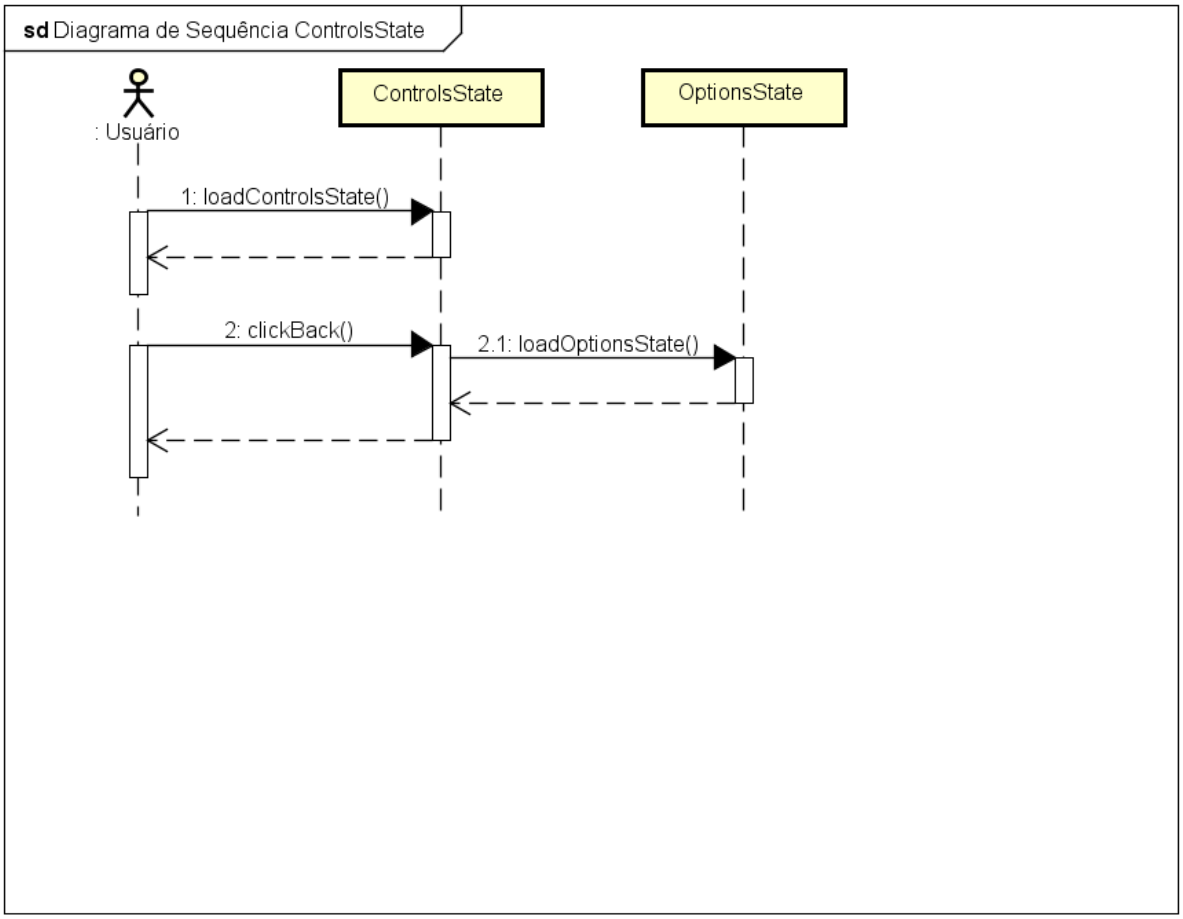

Figura 32 - Diagrama de sequência da visualização dos controles

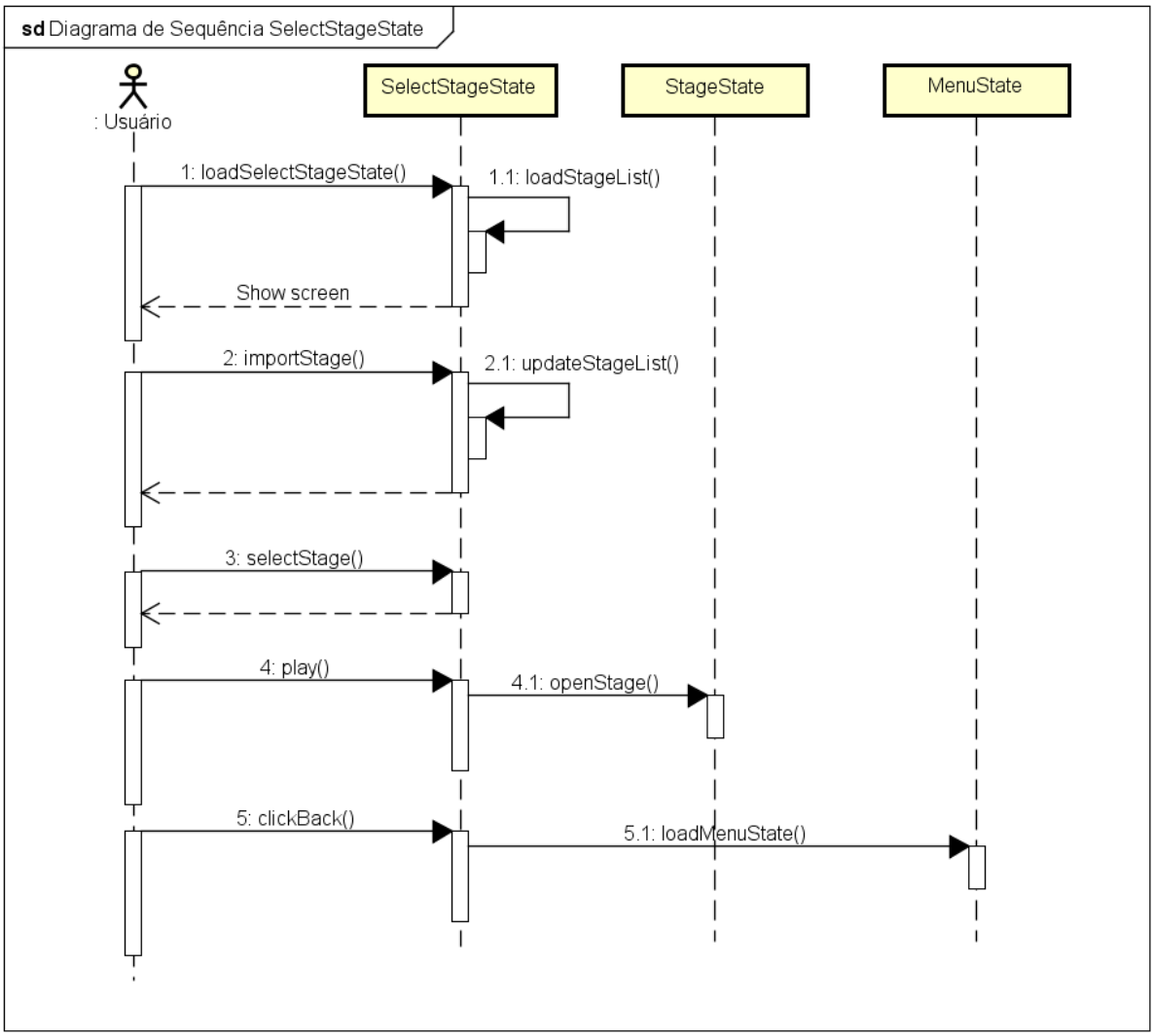

Figura 33 - Diagrama de sequência da seleção / importação de fase para jogar

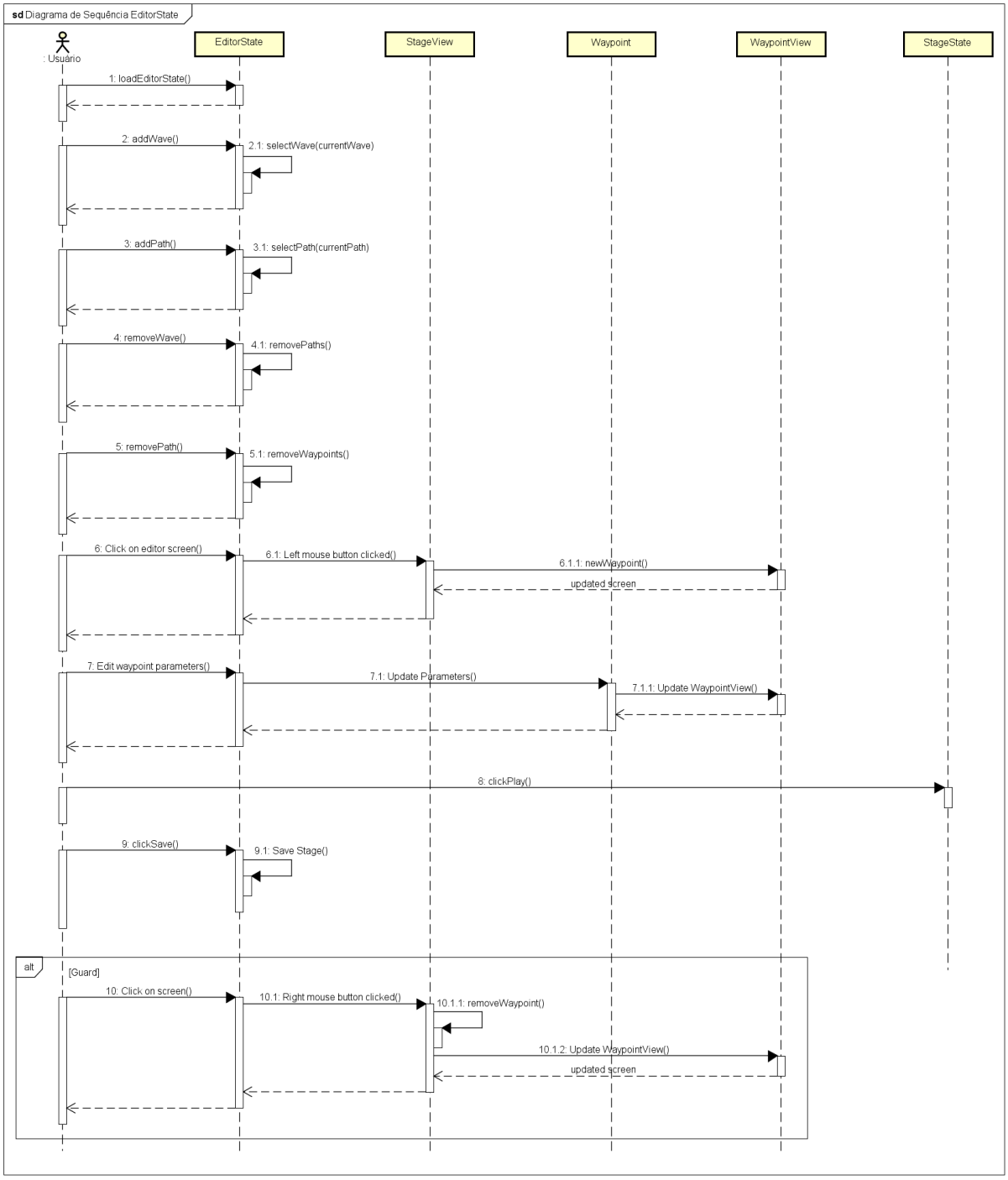

Figura 34 - Diagrama de sequência do editor de fases

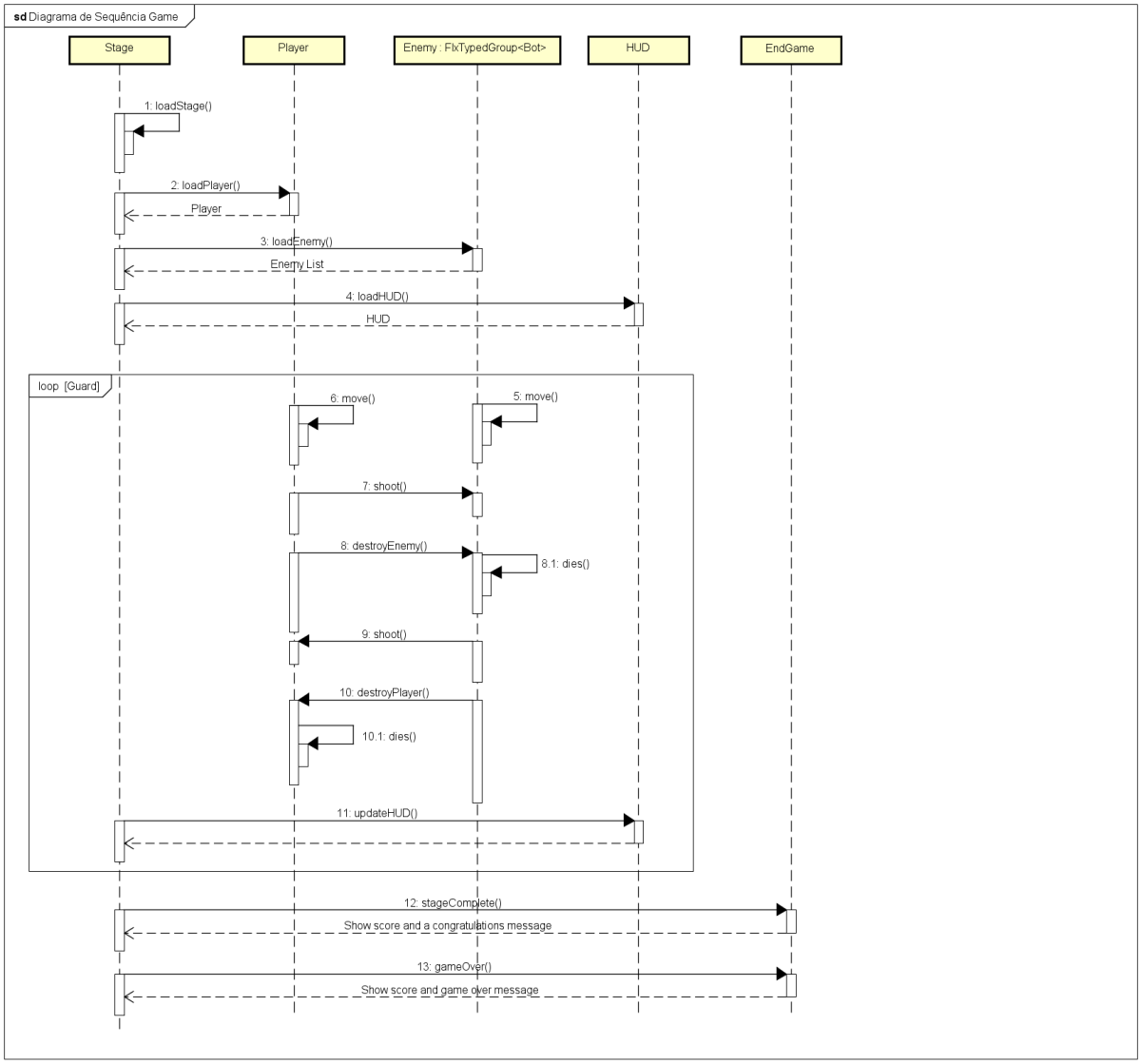

Figura 35 - Diagrama de sequência de uma fase em execução

APÊNDICE E – ARTEFATOS GDD

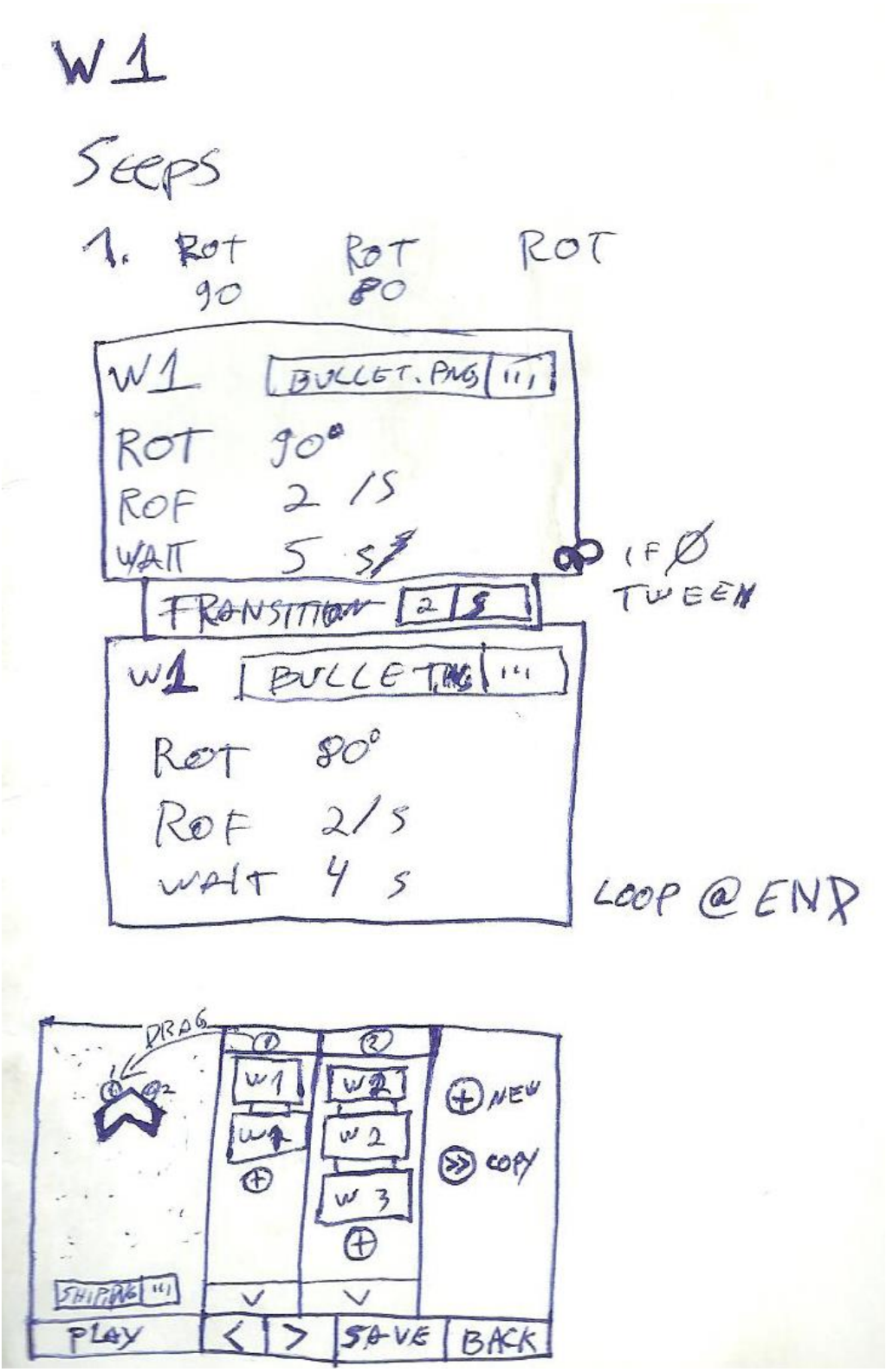

Figura 36 - Propriedades do waypoints e editor

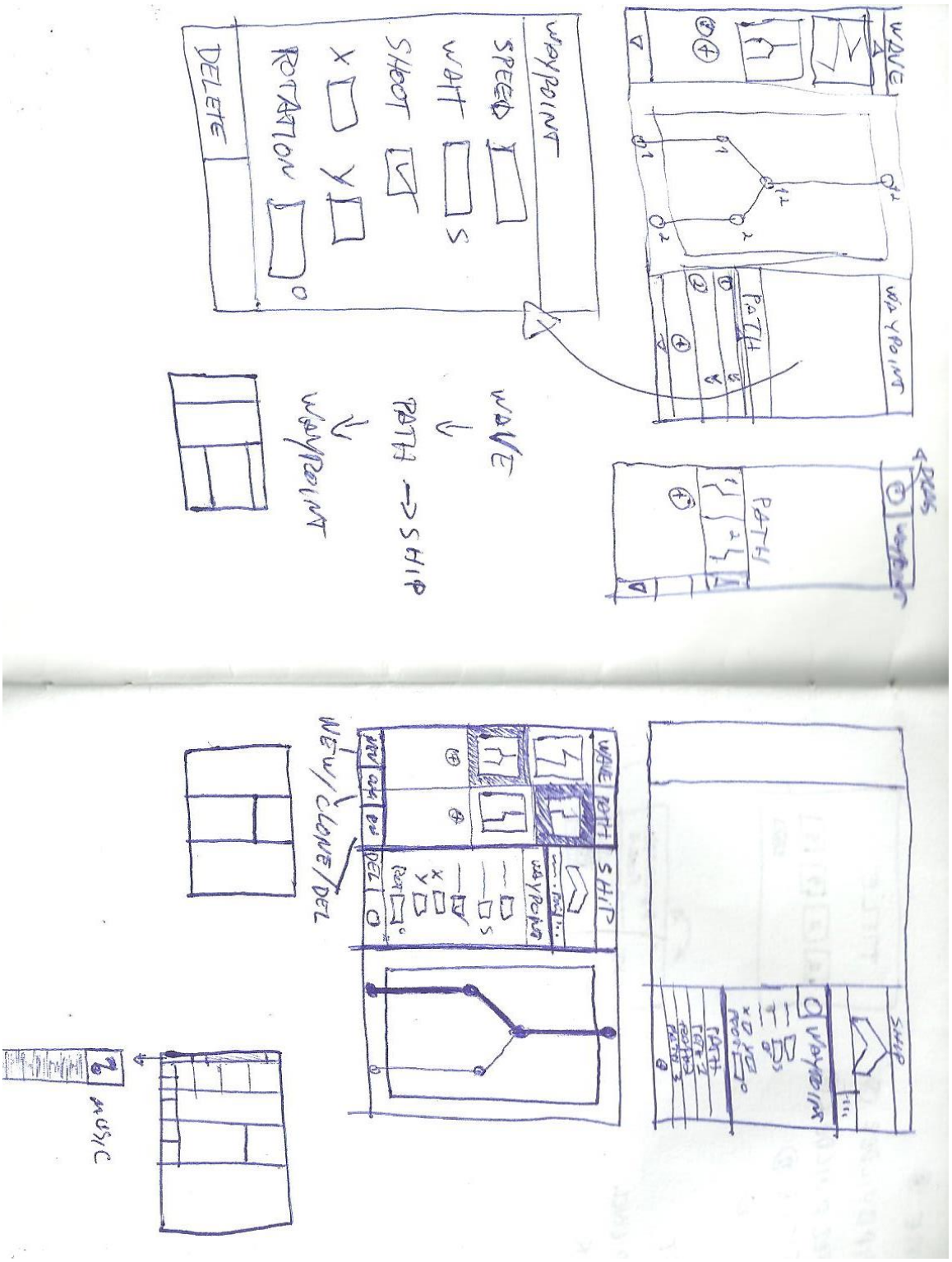

Figura 37 - Fluxo do editor de fases e waypoints

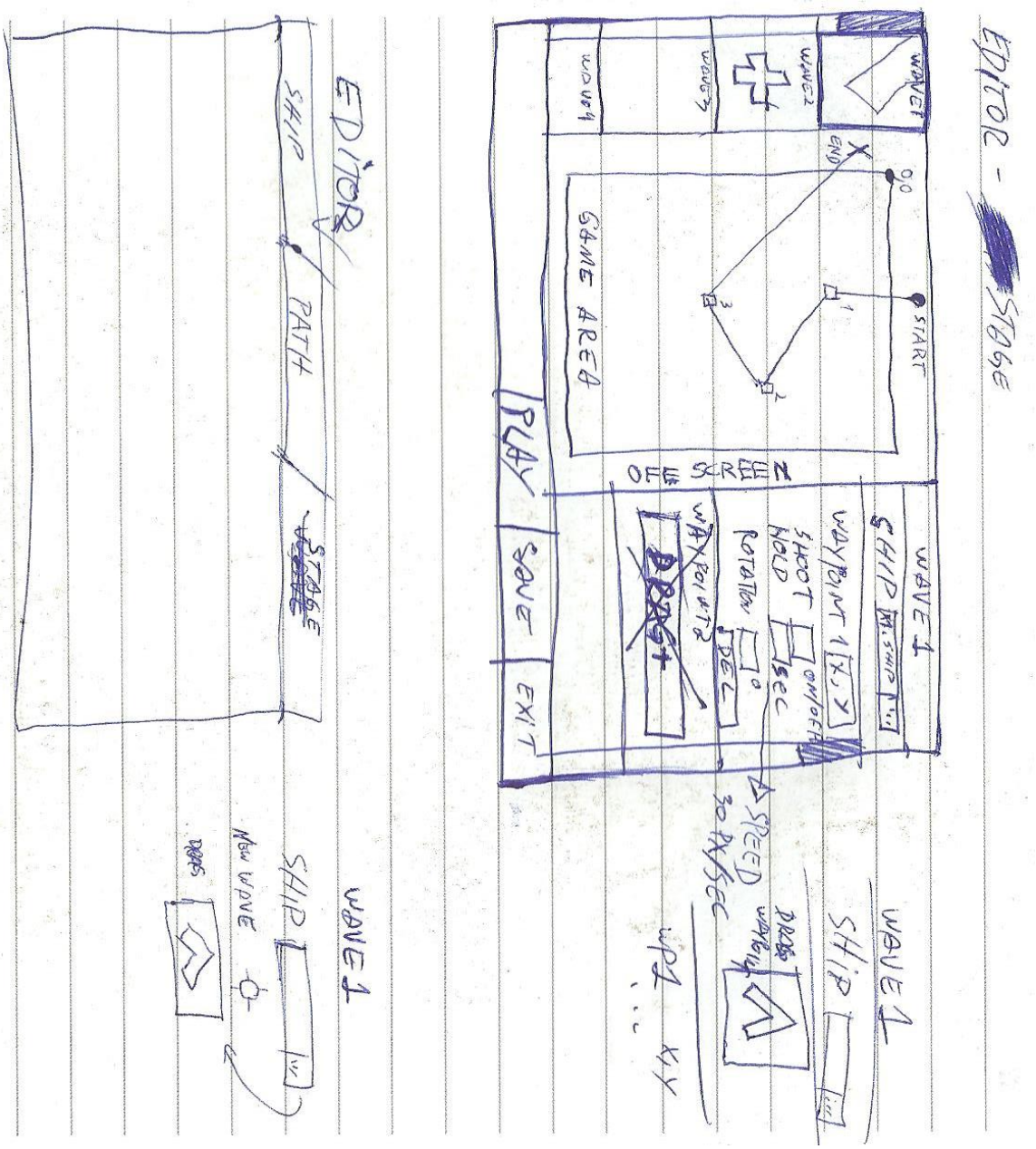

Figura 38 - Rascunho do editor de fases

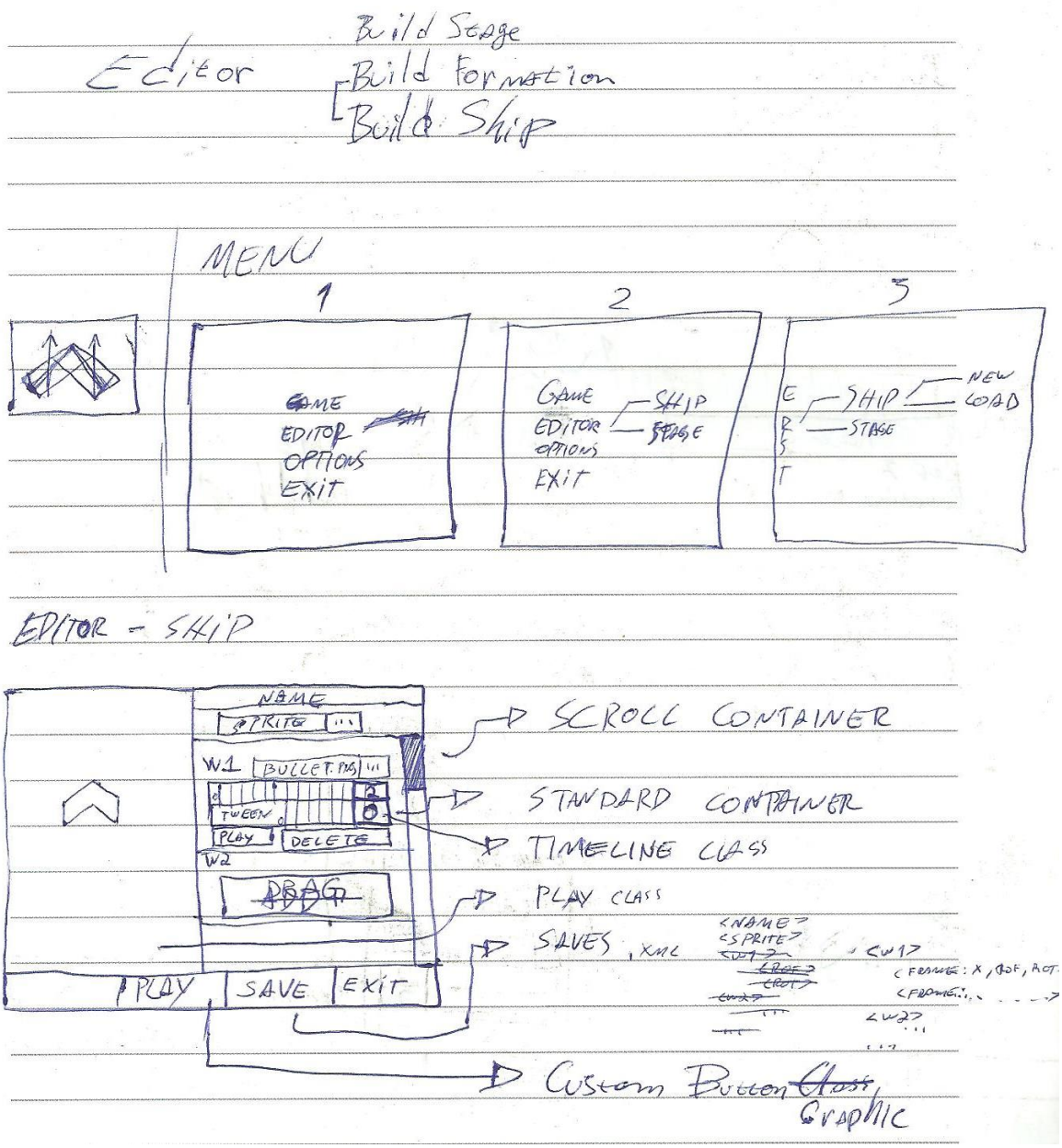

Figura 39 - Menus e editor de fases

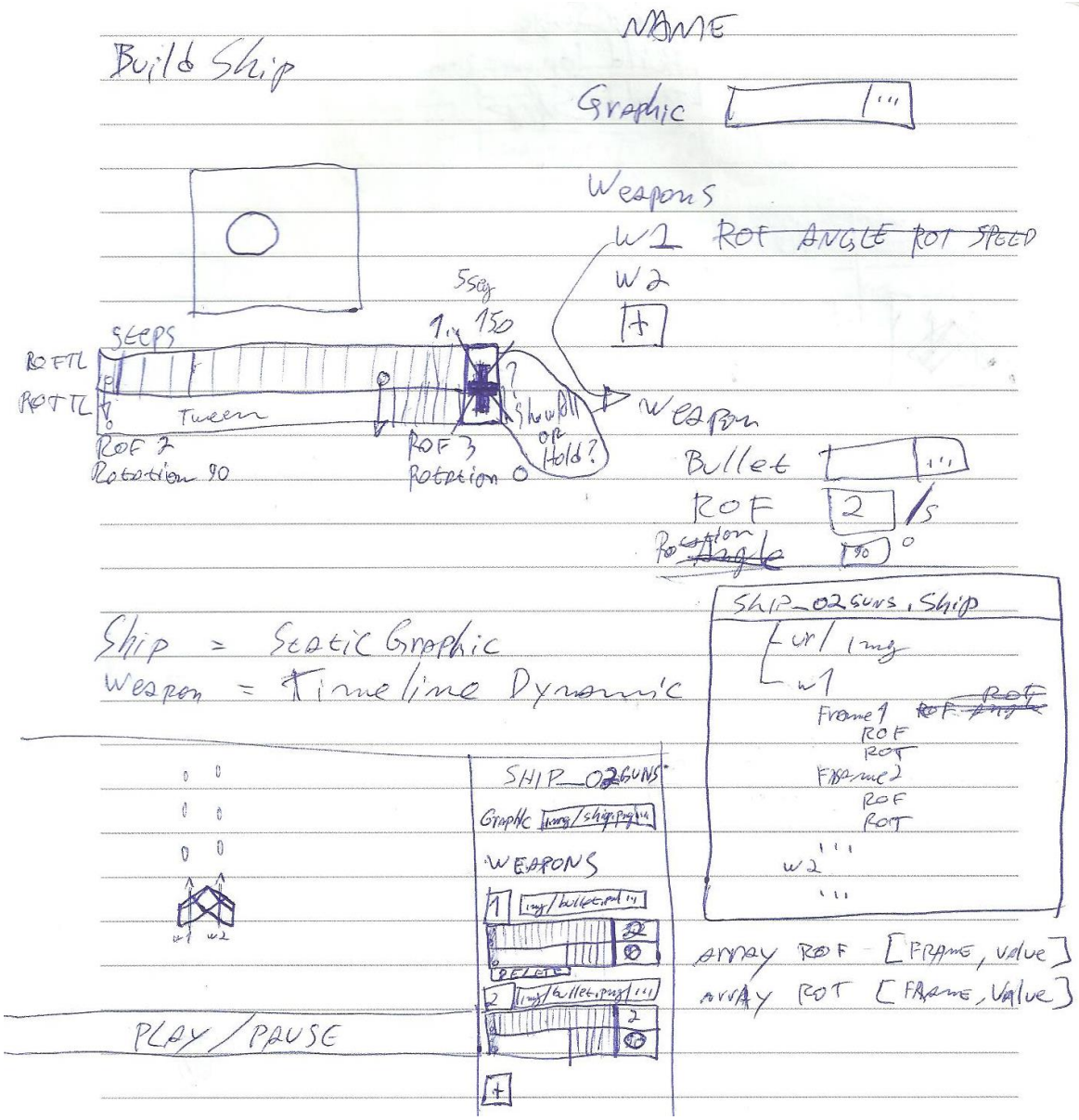

Figura 40 - Editor de naves

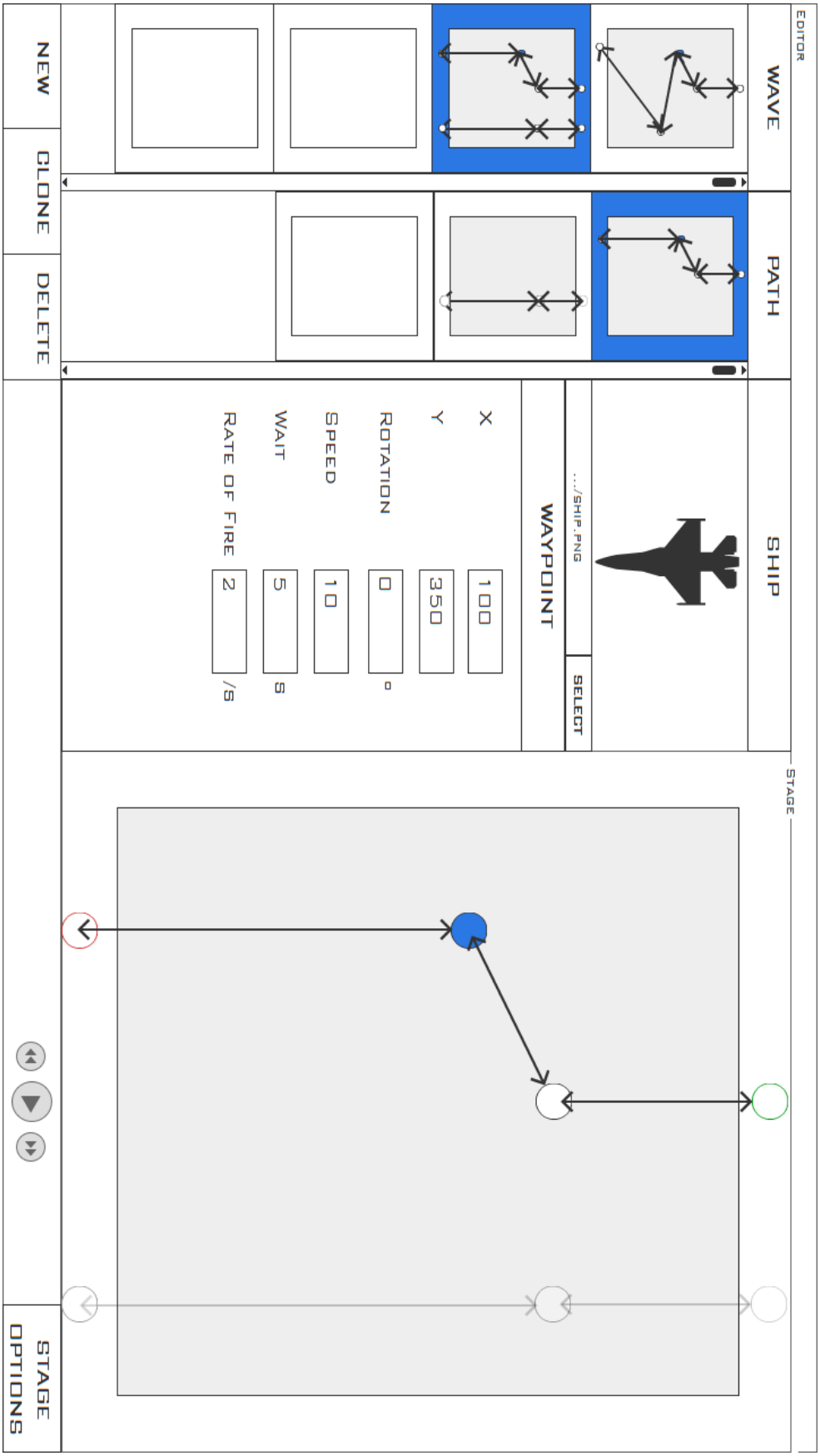

Figura 41 - Balsamic do editor de fases