Politécnico
Castelo Branco
Escola Superior
de Tecnologia

Desenvolvimento de um jogo em Unity
Projeto I

Filipe José Maurício Pires

20191327

Orientadores

Paulo Alexandre Correia da Silva Neves

Trabalho de Projeto apresentado à Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco para cumprimento dos requisitos necessários à obtenção do grau de Licenciado em licenciatura de Engenharia Informática, realizada sob a orientação científica do Professor Adjunto Paulo Alexandre Correia da Silva Neves, do Instituto Politécnico de Castelo Branco

Janeiro 2025

v. 1.0

Composição do júri

Presidente do júri

Doutor, Eurico Lopes ,Professor Coordenador ESTCB

Vogais

Mestre, Paulo Alexandre Neves,

Professor Adjunto

Doutor, Arlindo Silva

Professor Adjunto ESTCB

Resumo

Este relatório documenta o desenvolvimento do projeto realizado no âmbito da unidade curricular de Projeto I, do terceiro ano da Licenciatura em Engenharia Informática da Escola Superior de Tecnologia de Castelo Branco. O objetivo é apresentar as etapas concluídas até ao momento na criação de um jogo shoot 'em up, destacando as decisões tomadas e as ferramentas utilizadas.

O trabalho inicia-se com o estudo do estado da arte, analisando jogos de referência, como *Galaga*, *Space Hunter* e *Xenon*. Esta análise permitiu identificar mecânicas essenciais do género, como tipos de disparos, movimentações e sistemas de power-ups, adaptando-as ao conceito do jogo.

Com base neste estudo, foi elaborado um GDD – Game Design Document, onde foram definidas as principais mecânicas do jogo, incluindo a movimentação da nave, o sistema de inimigos e os power-ups que modificam a jogabilidade. Este documento serviu como guia para a fase de implementação.

Seguidamente, foi desenvolvido um protótipo funcional, onde foram exploradas as ferramentas e tecnologias escolhidas. O relatório descreve a implementação da movimentação da nave, o sistema de disparo, a criação automática de inimigos e os efeitos dos power-ups, bem como a estrutura do código.

O relatório conclui com uma análise do progresso realizado, e sobre os próximos passos para a continuação do desenvolvimento do jogo.

Palavras-chave

Shoot 'em up, Unity, GDD, Jogo 2D, Jogabilidade

Abstract

This report documents the development of the project carried out within the scope of the Project I curricular unit, of the third year of the Degree in Computer Engineering at the Escola Superior de Tecnologia de Castelo Branco. The objective is to present the steps completed so far in creating a space shoot 'em up, highlighting the decisions made and the tools used.

The work began with a study of the state of the art, analyzing reference games, such as Galaga, Space Hunter and Xenon. This analysis made it possible to identify essential mechanics of the genre, such as automatic shooting, lateral movement and power-up systems, adapting them to the game's concept.

Based on this study, a GDD – Game Design Document was prepared, where the main mechanics of the game were defined, including the movement of the ship, the enemy system and the power-ups that modify the gameplay. This document served as a guide for the implementation phase.

Next, a functional prototype was developed, where the chosen tools and technologies were explored. The report describes the implementation of ship movement, the firing system, automatic enemy creation and the effects of power-ups, as well as the code structure.

The report concludes with an analysis of progress made, and next steps for continued development of the game.

Keywords

Shoot 'em up, Unity, GDD, 2D Game, Gameplay

Índice geral

Índice de figuras

Lista de tabelas

Lista de abreviaturas, siglas e acrónimos

GDD – Game Design Document

1 Introdução

Os videojogos tornaram-se uma das formas mais populares de entretenimento digital, abrangendo diferentes estilos e géneros que atraem milhões de jogadores em todo o mundo. Entre esses géneros, os jogos de nave espacial ganharam destaque ao longo das décadas, caracterizando-se por uma jogabilidade dinâmica, ação intensa e desafios progressivos. Desde os clássicos arcades até os títulos mais modernos, esse tipo de jogo desafia os reflexos dos jogadores e exige uma combinação de estratégia e destreza para superar inimigos e obstáculos[11].

A história dos videojogos remonta às décadas de 1950 e 1960, quando cientistas da computação começaram a desenvolver jogos simples e simulações em minicomputadores e mainframes. Um dos primeiros exemplos é "Spacewar!", criado em 1962 por estudantes do Massachusetts Institute of Technology (MIT), considerado um dos primeiros jogos a utilizar um ecrã de vídeo. Este período inicial lançou as bases para a indústria dos videojogos, que viria a expandir-se rapidamente nas décadas seguintes[12].

Este projeto tem como objetivo desenvolver um jogo 2D de nave espacial inspirado em títulos icónicos do género. O jogador assumirá o controlo de uma nave equipada com disparos automáticos, devendo eliminar ondas de inimigos e evitar colisões. Para tornar a experiência mais envolvente, serão introduzidos power-ups que permitirão ao jogador melhorar temporariamente o desempenho da sua nave, como o aumento da cadência de tiro ou proteção contra ataques. Além disso, os inimigos terão padrões de movimento e ataque variados, proporcionando um desafio progressivo e uma maior complexidade ao longo dos níveis.

1.1 Ferramenta Utilizada

Para a implementação deste jogo, será utilizada a Unity, uma das engines mais versáteis e amplamente adotadas no desenvolvimento de jogos digitais. A Unity oferece um ambiente poderoso para a criação de jogos tanto em 2D ou 3D, contando com um vasto conjunto de ferramentas para animação, física, inteligência artificial, e integração de áudio e gráficos. Além disso, a engine permite uma gestão eficiente dos recursos e

oferece suporte a múltiplas plataformas, o que possibilita um desempenho otimizado do jogo[1].

A programação será realizada em C#, uma linguagem orientada a objetos amplamente utilizada na Unity devido à sua estrutura clara e eficiente. O C# permitirá desenvolver um código modular e reutilizável, facilitando a implementação de funcionalidades essenciais como o movimento da nave, a geração dinâmica de inimigos, a deteção de colisões e a aplicação de efeitos gráficos e sonoros[13].

Outro fator importante no desenvolvimento será a utilização de boas práticas de programação para garantir a escalabilidade e manutenção do código. A organização do projeto seguirá princípios de modularidade e reutilização de código, permitindo uma fácil expansão do jogo no futuro. Além disso, serão exploradas bibliotecas e componentes da Unity para otimizar o desempenho e garantir uma experiência de jogo fluida e envolvente.

Dessa forma, a escolha da Unity como engine e do C# como linguagem de programação garante um ambiente sólido para a concretização do projeto, possibilitando a criação de um jogo desafiante, com mecânicas bem estruturadas e uma jogabilidade cativante[1].

1.2 Objetivos

Os objetivos delineados para este projeto incluem, numa etapa inicial, a realização de uma análise ao estado da arte, com foco em jogos do mesmo género ou relacionados ao tema do projeto. Esta análise tem como objetivo identificar ideias e elementos que possam ser aproveitados e integrados no jogo em desenvolvimento. Além disso, planeia-se criar um GDD (Game Design Document), que servirá como base para a conceção do jogo. Por fim, será desenvolvido um protótipo que implementará as principais mecânicas descritas no GDD.

1.3 Estrutura do relatório

O relatório está organizado em cinco capítulos, sendo o primeiro dedicado à introdução ao projeto. No segundo capítulo, é feita uma análise de jogos relevantes para o desenvolvimento do projeto, abordando o estado da arte. Para cada um desses jogos, será discutida a respetiva plataforma, a forma de interação do jogador, a jogabilidade e os elementos presentes, seguindo-se uma análise que visa identificar funcionalidades que podem ser incorporadas no jogo em desenvolvimento.

No terceiro capítulo será apresentado o Game Design Document (GDD), que aborda de forma detalhada as mecânicas, funcionalidades e objetivos do jogo a desenvolver, fornecendo uma visão clara do planeamento e da estrutura necessários para a sua implementação final.

No quarto capítulo, será apresentado o protótipo desenvolvido, detalhando as funcionalidades já implementadas, as principais dificuldades encontradas e os resultados obtidos até o momento. Este capítulo serve para demonstrar a evolução do projeto e validar algumas das escolhas feitas durante o desenvolvimento.

Por fim, o quinto capítulo apresenta uma conclusão, onde é feita uma análise ao trabalho desenvolvido neste projeto e aos futuros objetivos para a continuação do desenvolvimento do jogo.

2 Estado da Arte

Este capítulo descreve o estudo do estado da arte realizado previamente à definição do jogo a ser desenvolvido neste projeto. O objetivo é analisar jogos que se destacam pela temática. Para isso, inicia-se com a análise de um dos jogos que serviu de base para a ideia, o *Space Hunting*, seguido pelo *Galaga e por fim o Xenon*. Em todos os casos, são examinados a sua origem, os objetivos, a jogabilidade, regras e mecânicas, assim como os elementos que os compõem. Por fim, realiza-se uma análise das características que poderão ser incorporadas no jogo deste projeto.

2.1 Escolha dos jogos para o Estado da Arte

A escolha dos jogos Galaga, Xenon e Space Hunting para o estudo do estado da arte foi motivada pelo objetivo de compreender a evolução do género shoot 'em up ao longo do tempo. Galaga, um clássico dos anos 80, representa as origens do género, com mecânicas simples mas inovadoras para a época. Xenon, lançado no final dos anos 80, introduziu complexidades como a alternância entre modos de nave, refletindo uma fase intermediária de evolução. Por fim, Space Hunting, um jogo mais recente, mostra como o género se adaptou às expectativas modernas, com gráficos atualizados e mecânicas acessíveis, mas estratégicas.

2.2 Space Hunting

Space Hunting é um jogo de tiro arcade desenvolvido e lançado pela empresa MarketJS em 2 de abril de 2020[2]. O título combina a essência dos clássicos do gênero com uma abordagem moderna, oferecendo gráficos atrativos e uma jogabilidade envolvente. No jogo, os jogadores enfrentam ondas de inimigos enquanto progridem por níveis crescentes de dificuldade, testando suas habilidades e reflexos.

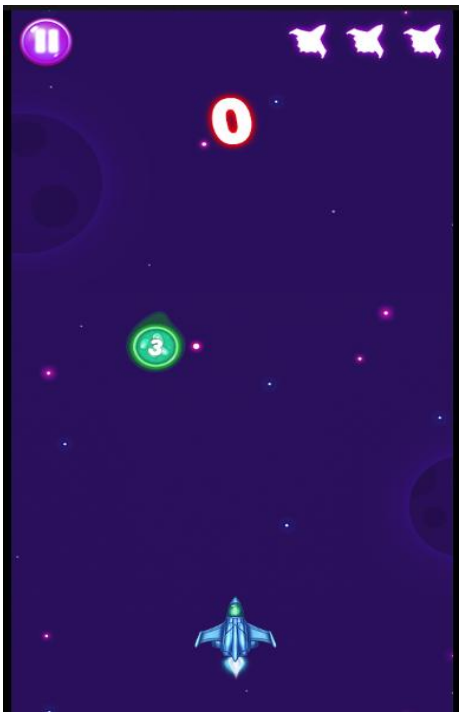

Figura 1 - Space Hunting - Captura de momento de jogo[2]

O objetivo principal de *Space Hunting* é sobreviver o maior tempo possível enquanto se enfrenta ondas de inimigos espaciais acumulando pontos ao longo do caminho. Power-ups podem ser coletados para melhorar as habilidades da nave, ajudando na sobrevivência. O jogo desafia o jogador a obter a maior pontuação possível, superando suas próprias marcas a cada partida.

2.2.1 Plataforma

Space Hunting foi desenvolvido pela empresa MarketJS e é jogado diretamente no navegador utilizando a tecnologia HTML5[3]. O jogo foi projetado para ser acessível em diferentes plataformas, incluindo computadores, tablets e telemóveis. Essa escolha de plataforma possibilita que os gráficos, jogabilidade e mecânicas sejam implementados de forma fluida e otimizada em dispositivos com diversos sistemas operacionais.

2.2.2 Interação

Em *Space Hunting*, a interação no computador é projetada para ser simples e fluida, aproveitando o uso do rato como principal ferramenta de controle. O jogador movimenta a nave ao arrastar o cursor, de forma que a posição do rato na tela determina diretamente o movimento da nave. Além disso, os disparos são automáticos, permitindo que o jogador se concentre exclusivamente em desviar dos inimigos e dos obstáculos enquanto elimina os adversários. Essa abordagem intuitiva reduz a complexidade dos controles, tornando o jogo acessível e garantindo uma jogabilidade dinâmica e responsiva.

Figura 2 - Space Hunting - Movimentos do jogo[2]

Em dispositivos móveis, a interação em *Space Hunting* ocorre através de controles táteis. O jogador movimenta a nave ao deslizar o dedo na tela, seguindo os movimentos de forma direta e precisa. Assim como na versão para computador, os disparos são automáticos, permitindo que o jogador se concentre exclusivamente na movimentação para evitar obstáculos e enfrentar os inimigos. Essa mecânica torna o jogo acessível e prático para partidas rápidas em qualquer lugar, aproveitando plenamente as funcionalidades da interface de toque dos dispositivos móveis.

2.2.3 Gameplay

O jogador assume o controlo de uma nave espacial e tem de eliminar os inimigos presentes no cenário e evitar obstáculos. A principal característica do jogo é a presença de inimigos numerados, representados por orbes que se movem de forma aleatória ou seguindo padrões específicos. Cada orbe tem um valor numérico que indica quantos tiros são necessários para os destruir. À medida que o jogador avança, os inimigos aparecem em maior quantidade e com valores mais elevados, tornando o jogo gradualmente mais difícil.

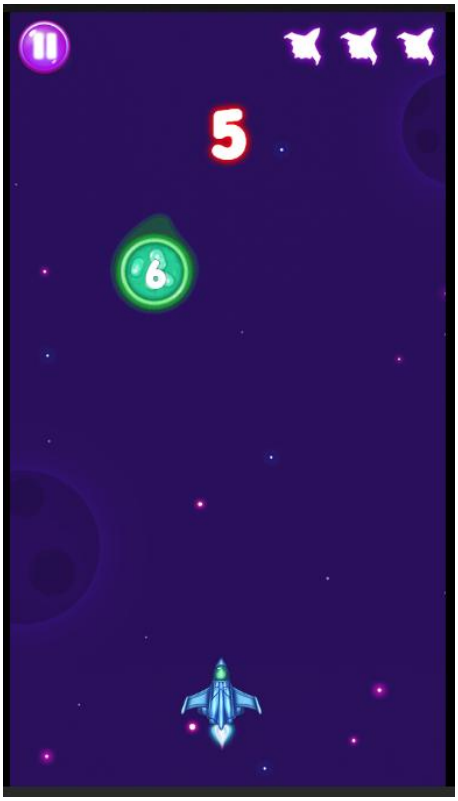

Figura 3 - Space Hunting -Exemplo de um inimigo numerado[2]

2.2.4 Elementos

A nave controlada pelo jogador é o centro de toda a ação, equipada com disparos automáticos que permitem focar exclusivamente na movimentação e no posicionamento estratégico.

Os inimigos são representados por orbes numerados que apresentam um design visual simples, mas funcional. Cada orbe exibe um valor numérico correspondente à quantidade de disparos necessários para destruí-lo, proporcionando ao jogador uma

indicação clara do desafio de cada confronto. Além disso, o aumento gradual na complexidade dos inimigos, tanto em quantidade quanto em valores numéricos, adiciona progressão ao jogo, mantendo o jogador continuamente desafiado.

Outro elemento chave é o cenário dinâmico, que inclui obstáculos e padrões de movimento que o jogador precisa evitar enquanto enfrenta os inimigos. A combinação destes obstáculos com as ondas crescentes de adversários cria um ambiente que testa reflexos e habilidades de decisão em tempo real.

Por fim, os power-ups representam um recurso essencial. Espalhados de forma estratégica durante o jogo, eles fornecem ao jogador melhorias temporárias, como aumento da velocidade, reforço nos disparos ou invulnerabilidade momentânea. Esses elementos tornam-se cruciais para superar níveis difíceis e alcançar altas pontuações, incentivando o jogador a utilizá-los de forma estratégica.

2.2.5 Analise

O *Space Hunting* apresenta mecânicas interessantes que podem ser adaptadas ao projeto, destacando-se os inimigos numerados e os power-ups. Os inimigos numerados oferecem uma forma clara e visual de transmitir a resistência de cada adversário, criando uma dinâmica que incentiva o jogador a planear os seus ataques de forma estratégica. Esta característica promove um desafio progressivo e uma sensação de conquista ao derrotar inimigos mais resistentes à medida que o jogo avança.

Além disso, o uso de power-ups no jogo adiciona variedade e reforça a capacidade do jogador em lidar com ondas crescentes de inimigos. A recolha de melhorias temporárias permite um aumento momentâneo no poder de fogo ou na sobrevivência, o que contribui para uma experiência de jogo dinâmica e divertida. Estas ideias podem ser incorporadas no projeto para enriquecer a jogabilidade.

2.3 Galaga

Galaga é um jogo de tiro arcade fixo desenvolvido pela Namco e lançado em 1981 como sequência direta de Galaxian (1979). Publicado na América do Norte pela Midway Manufacturing, é considerado um dos jogos mais icônicos da Era de Ouro dos Videojogos Arcade. O jogador controla uma nave estelar com o objetivo de destruir as forças inimigas enquanto evita projéteis e ataques[9].

Figura 4 - Galaga - momento de jogo Galaga

Uma das mecânicas inovadoras de Galaga é o sistema de captura de naves: certos inimigos podem usar um feixe trator para capturar a nave do jogador, que pode ser resgatada e convertida em uma "dual fighter", dobrando seu poder de fogo. Essa inovação, junto com as fases desafiantes e progressivamente mais difíceis, trouxe profundidade estratégica ao gênero[16].

Liderado por Shigeru Yokoyama, o desenvolvimento contou com uma pequena equipe e cerca de dois meses de projeto. O sucesso de Galaga nos arcades garantiu sua presença em diversas plataformas, como NES, Atari 7800 e serviços digitais como Xbox Live Arcade. Até hoje, Galaga é reconhecido como um marco dos videojogos e uma referência no gênero de tiro arcade[9].

2.3.1 Plataforma

A Nintendo Entertainment System (NES) (Figura 5) é um console de jogos eletrónicos de 8 bits lançado pela Nintendo em 1983 no Japão como Family Computer (Famicom) e nos anos seguintes na América do Norte, Europa e Austrália. Foi o console mais vendido da sua época, ajudando a revitalizar a indústria de videojogos após o colapso de 1983[9].

Figura 5 -Consola NES [10]

Concebido por Masayuki Uemura, o design inicial previa um sistema avançado, mas acabou simplificado para um console de cartuchos acessível. Após resolver problemas técnicos do lançamento inicial no Japão, o Famicom tornou-se o console mais popular até 1984. Sua entrada no mercado norte-americano enfrentou ceticismo, mas títulos como *Super Mario Bros.* conquistaram o público.

O NES consolidou a Nintendo como líder no setor, definindo padrões para futuros consoles e marcando o início da era de dominação japonesa na indústria de jogos. Apesar de descontinuado em 1995, o NES permanece um marco na história dos videojogos[18].

2.3.2 Interação

A interação com *Galaga* na consola NES era feita através de um comando simples e icónico (Figura 6), projetado com um formato retangular, dois botões de ação, e uma cruz direcional para movimentação. Esse design intuitivo permitia aos jogadores controlar a nave facilmente, com movimentos precisos e rápidos, essenciais para evitar ataques inimigos e disparar[10].

Figura 6 - Comando consola NES[10]

Atualmente, graças aos emuladores disponíveis para computadores, é possível jogar *Galaga* utilizando o teclado, que simula as funções do comando original. Essa alternativa torna o jogo mais acessível, permitindo reviver a experiência clássica, mesmo sem o equipamento da época[10].

2.3.3 Gameplay

Em *Galaga*, o jogador controla uma nave espacial e deve derrotar ondas de inimigos enquanto evita projéteis e ataques diretos. A nave pode mover-se horizontalmente ao longo da parte inferior do ecrã e disparar em linha reta contra os inimigos que surgem a partir do topo do ecrã[14].

A mecânica de jogo inclui inimigos organizados em formações que atacam em padrões variados e coordenados. Alguns inimigos líderes utilizam o "feixe trator" para capturar a nave do jogador. Caso o jogador consiga resgatar a nave capturada, ela é adicionada como uma nave adicional, criando o poderoso *Dual Fighter*, que duplica o poder de fogo e altera a estratégia de combate[17].

À medida que o jogador progride pelos níveis, as ondas de inimigos tornam-se mais numerosas e rápidas, aumentando a complexidade e a exigência de reflexos rápidos. O jogo também inclui níveis bônus, denominados *Challenging Stages*, onde o objetivo é destruir o máximo de inimigos em formação para acumular pontos extras.

2.3.4 Elementos

O jogador assume o controlo de uma nave espacial com mobilidade horizontal limitada na parte inferior da tela, equipada com disparos básicos que podem ser aprimorados caso outra nave seja resgatada dos inimigos. Esta mecânica de resgate ativa a funcionalidade de “dual fighter”, permitindo disparos duplos e aumentando significativamente o poder de ataque[16].

Os inimigos são alienígenas que se destacam pelos seus padrões de voo variados, alternando entre formações organizadas e ataques individuais. Algumas unidades possuem habilidades especiais, como a capacidade de capturar a nave do jogador através de um feixe trator, adicionando um elemento estratégico às batalhas. Apesar de o jogo não incluir power-ups tradicionais, o resgate de uma nave capturada serve como uma forma de melhoria essencial para aumentar as capacidades ofensivas do jogador, recompensando decisões mais arriscadas.

Os cenários do jogo recriam a imensidão do espaço com fundos estrelados e estáticos, criando uma atmosfera envolvente enquanto os inimigos ganham destaque visual. À medida que o jogador avança nos níveis, a dificuldade aumenta gradualmente, com os inimigos surgindo em maior quantidade e apresentando padrões de ataque mais rápidos e complexos. Esta progressão dinâmica exige reflexos apurados e tomadas de decisão estratégicas, garantindo que o desafio permaneça constante e recompensador[17].

2.3.5 Analise

O *Galaga* apresenta mecânicas de jogo inovadoras para a sua época, muitas das quais podem ser adaptadas ou servir de inspiração para novos projetos. Entre os aspetos mais notáveis, destaca-se a mecânica de "dual fighter", que não só aumenta o poder de ataque da nave do jogador, mas também introduz uma camada estratégica ao jogo. A decisão de arriscar o resgate de uma nave capturada adiciona tensão e recompensa o jogador com um aumento significativo das suas capacidades ofensivas, criando um equilíbrio interessante entre risco e benefício.

Outro elemento digno de análise é o comportamento dos inimigos. Os padrões de voo diversificados e os ataques organizados em formações contribuem para uma jogabilidade dinâmica que exige reflexos rápidos e planeamento estratégico. Além disso, a inclusão de inimigos com habilidades especiais, como o feixe trator que captura a nave do jogador, proporciona variedade e um desafio adicional à experiência.

2.4 Xenon

Xenon é um jogo de tiro vertical desenvolvido pela The Bitmap Brothers e lançado em 1988 para o Atari ST. Considerado um dos primeiros sucessos da empresa, Xenon rapidamente se espalhou por outras plataformas, incluindo Amiga, Amstrad CPC, Commodore 64, DOS, MSX e ZX Spectrum. O jogo introduz uma mecânica única, permitindo ao jogador alternar entre uma nave espacial e um tanque terrestre, oferecendo uma jogabilidade versátil dependendo da situação enfrentada no jogo. Além da versão original para computadores, Xenon também foi adaptado para arcade através da divisão Arcadia da Mastertronic, utilizando hardware do Commodore Amiga para oferecer uma experiência fluida e visualmente impressionante para a época[19].

Figura 7 - Xenon - Captura de momento de jogo[22]

A inovação de Xenon não se limitou apenas às suas versões multiplataforma, mas também se destacou pelas suas fases desafiantes e por oferecer power-ups que enriqueciam a experiência de jogo, como naves que podiam ser melhoradas ao coletar itens durante a batalha. Com uma recepção positiva da crítica, Xenon permaneceu como um título icônico dentro do gênero de atiradores verticais, sendo lembrado pelo seu design inovador e pela diversão imersiva que proporcionou aos jogadores na época[22].

2.4.1 Plataforma

O jogo *Xenon* foi inicialmente desenvolvido para a plataforma Atari ST(Figura 8), um computador pessoal lançado em 1985. Esse sistema foi uma das principais plataformas de jogos nos anos 80 e 90, graças ao seu processador Motorola 68000 e gráficos avançados para a época, que permitiam jogos com uma boa qualidade visual e sonora. O Atari ST tornou-se uma das plataformas ideais para desenvolvedores de jogos, pois oferecia grande capacidade de processamento de gráficos, algo fundamental para jogos como *Xenon[23]*.

Figura 8 - Atari ST[23]

Além do Atari ST, *Xenon* foi rapidamente portado para outras plataformas populares da época, como o Amiga, uma das principais concorrentes do Atari ST no mercado de computadores pessoais. O Amiga era conhecido por seus gráficos e sons superiores, o que oferecia uma experiência ainda mais aprimorada em comparação com o Atari ST, especialmente em jogos com grande exigência gráfica. *Xenon* também foi disponibilizado para plataformas como o Commodore 64, ZX Spectrum, MSX e Amstrad CPC, garantindo uma ampla distribuição e acessibilidade ao jogo, permitindo que os jogadores de diferentes sistemas de então tivessem a oportunidade de vivenciar a experiência de jogar *Xenon*.

Xenon aproveitou as tecnologias avançadas de diferentes plataformas, o que garantiu uma experiência consistente e de alta qualidade ao longo das versões em que foi lançado, destacando-se como um dos jogos mais memoráveis da época[24].

2.4.2 Interação

Em Xenon, a interação do jogador é caracterizada por controles simples e diretos, típicos dos jogos de computador da década de 1980. O jogo, sendo um shoot 'em up de rolagem vertical, permite ao jogador controlar uma nave espacial em sua missão de eliminar inimigos enquanto evita obstáculos e projéteis[20].

O jogador utiliza o teclado para controlar a nave, normalmente atribuindo as teclas de direção para mover a nave para os lados, para cima e para baixo na tela. Este controle de movimento é fundamental, pois o jogador deve esquivar-se dos tiros inimigos e, simultaneamente, atacar.

Dessa forma, a interação em *Xenon* é simples, porém estratégica, permitindo uma jogabilidade intensa e dinâmica. O controle é intuitivo, mas a alternância entre as formas de nave adiciona uma camada extra de decisão que eleva a experiência do jogador, garantindo que cada fase seja mais desafiadora e estimulante[22].

2.4.3 Gameplay

Xenon oferece uma gameplay dinâmica característica dos shooters verticais dos anos 80, desafiando os reflexos e a estratégia do jogador em igual medida. O jogador controla Darrian, um piloto espacial da Federação, enquanto enfrenta os Xenites, uma raça alienígena hostil[24].

Figura 9 -Xenon - momento de jogo[25]

A nave possui um sistema de controle manual para disparos, o que significa que o jogador deve pressionar um botão para disparar projéteis contra os inimigos. Diferentemente de outros jogos de tiro da época, onde o disparo poderia ser automático, em *Xenon*, o controle manual oferece maior profundidade à gameplay, exigindo que o jogador administre suas ações com mais cuidado, pensando em quando disparar enquanto executa manobras evasivas[25].

A nave de *Xenon* tem duas formas distintas: uma forma de avião e uma forma de tanque, o que proporciona ao jogador um controle versátil ao longo do jogo. A transição entre essas formas pode ser feita a qualquer momento durante o jogo (exceto em algumas fases específicas ou durante lutas contra chefes), permitindo uma adaptação tática a diferentes tipos de ameaças. A forma aérea é mais ágil e eficaz contra inimigos

voadores, enquanto a forma de tanque é mais robusta e útil contra obstáculos e inimigos de terreno.

Além disso, o jogo conta com power-ups que são liberados por alguns inimigos ao serem destruídos. Esses itens oferecem melhorias temporárias para a nave, como maior velocidade de movimento, melhorias no disparo ou até escudos de proteção, o que permite ao jogador enfrentar ondas mais difíceis de inimigos e obstáculos[21].

Em resumo, a gameplay de *Xenon* é uma mistura de ação e decisões rápidas e estratégicas. O controle manual do disparo, a possibilidade de alternar formas de nave e a coleta de power-ups geram uma experiência fluida, mas desafiadora, que mantém o jogador constantemente focado e motivado a progredir pelas fases.

2.4.4 Elementos

O jogo Xenon conta com diversos elementos essenciais que ajudam a criar a sua jogabilidade e criam uma experiência única ao jogar este jogo. A nave do jogador pode alternar entre dois modos distintos: o modo de voo e o modo tanque, o que torna a mecânica de combate mais estratégica. O jogador deve usar essa alternativa com sabedoria, adaptando-se às situações de cada fase e ajustando sua nave conforme o tipo de ameaça que enfrenta. Isso faz com que o jogo exija uma habilidade maior de adaptação, além de ser uma grande inovação em relação aos jogos de tiro tradicionais da época.

Os inimigos, variando em forma e comportamento, são outro elemento crucial no jogo. Ao longo das fases, o jogador enfrenta diferentes tipos de adversários, e cada um oferece um tipo distinto de desafio. Além disso, em determinadas fases, o jogador encontra chefes com padrões de ataque complexos e maior resistência, proporcionando um pico de dificuldade que recompensa a vitória com grande satisfação. Esses chefes ajudam a marcar a progressão do jogador, deixando cada nível mais envolvente e desafiador[20].

Outro elemento fundamental em Xenon são os power-ups, que se tornam vitais para a sobrevivência e o sucesso do jogador. Ao destruir inimigos, power-ups são liberados e podem ser coletados. Esses itens são extremamente importantes, pois aumentam temporariamente as capacidades da nave, como aumentar a velocidade de disparo ou

oferecer escudos de proteção. Esse sistema de upgrades traz um ritmo de jogo onde o jogador deve tomar decisões rápidas sobre quais power-ups aproveitar e quando usá-los, criando uma camada adicional de estratégia ao longo da experiência.

Os níveis, que variam em design e dificuldade, também contribuem para a dinâmica do jogo. Cada fase de Xenon possui um cenário espacial distinto, com novos desafios e inimigos. As fases são desenhadas de forma a exigir uma combinação de reflexos rápidos, habilidades estratégicas e uma compreensão do comportamento dos inimigos. O design das fases, aliado ao aumento da dificuldade, garante que o jogador sinta que está progredindo e, ao mesmo tempo, o desafia com novas mecânicas e obstáculos à medida que avança no jogo.

Por fim, a mecânica de dano em Xenon reflete a necessidade de os jogadores aprenderem os padrões de ataque e defesa dos inimigos. À medida que o jogo avança, os inimigos se tornam mais fortes e resistentes, desafiando ainda mais os reflexos e a habilidade do jogador. A constante mudança no comportamento dos inimigos e o uso de diferentes tipos de dano, como tiros múltiplos ou ataques de laser, tornam o jogo intenso e empolgante[24].

2.4.5 Analise

Xenon destacou-se no gênero de shooters verticais ao incluir a mecânica inovadora de alternar entre uma nave aérea e um tanque terrestre, tornando a jogabilidade mais estratégica. Esta abordagem diferenciou o jogo de muitos outros títulos da época, exigindo adaptação constante do jogador ao ambiente e aos padrões de ataque dos inimigos.

Além da jogabilidade inovadora, o jogo impressionou pelo seu alto nível técnico, especialmente nas versões para Atari ST e Amiga, com gráficos detalhados e efeitos visuais avançados para a época. A trilha sonora e os efeitos sonoros também foram bem trabalhados, aumentando a imersão.

2.4.6 Analise Geral

A análise feita a Space Hunting, Galaga e Xenon revela três abordagens diferentes dentro do género shoot 'em up, cada um com características únicas que mostram a evolução deste género de jogo ao longo dos tempos. Galaga, um clássico dos arcades, destaca-se pela simplicidade e pela mecânica inovadora do "dual fighter", que adiciona uma camada estratégica ao combate. Xenon, por sua vez, introduziu uma complexidade maior com a possibilidade de alternar entre modos de nave aéreo e terrestre, oferecendo uma jogabilidade mais desafiadora. Já Space Hunting, um jogo mais moderno, combina gráficos atualizados com mecânicas acessíveis, como inimigos numerados e power-ups, que incentivam a estratégia sem sobrecarregar o jogador.

Enquanto Galaga e Xenon focam em padrões de movimento e combate mais tradicionais, Space Hunting traz uma abordagem mais dinâmica e visualmente atraente, refletindo as expectativas dos jogadores contemporâneos. Essas diferenças mostram como o género evoluiu, mantendo a essência de ação rápida e desafios progressivos, mas adaptando-se às novas tecnologias e preferências dos jogadores.

3 GDD – Game Design Document

Este capítulo apresenta o Game Design Document (GDD), que descreve as características do jogo a ser desenvolvido. O objetivo é fornecer uma visão geral do projeto, destacando sua história, mecânicas e gameplay. O GDD é constituído pela descrição do jogo, historia, elementos, gameplay e dificuldade.

3.1 Descrição do jogo

O jogador assume o papel de um piloto de elite que controla uma nave espacial altamente avançada, encarregado de enfrentar uma ameaça alienígena que coloca a galáxia em perigo. Após a descoberta de uma frota inimiga invadindo colônias humanas, o jogador deve enfrentar ondas de inimigos e derrotar poderosos líderes alienígenas para recuperar territórios e salvar a civilização.

Este é um jogo 2D no estilo shoot 'em up, com visão lateral e progressão linear, que combina ação intensa e estratégia. O jogador terá a liberdade de aprimorar sua nave, adquirindo armas mais poderosas, escudos reforçados e habilidades especiais ao longo do jogo. Cada fase ocorre em um ambiente único, como campos de asteroides, nebulosas eletrificadas e bases espaciais em ruínas, trazendo variações visuais e novos desafios a serem superados.

Conforme o jogador avança, os inimigos tornam-se mais variados e letais, apresentando padrões de ataque mais complexos e habilidades específicas. Os chefes de fase representam um teste significativo de habilidade e estratégia, com ataques devastadores e fraquezas que o jogador deve descobrir e explorar.

3.2 Historia

A humanidade já foi próspera, dominando os céus e expandindo-se por incontáveis mundos sob a bandeira da Ordem Celestial, um governo interplanetário que trouxe estabilidade e progresso à galáxia. Durante séculos, acreditou-se que nada poderia ameaçar esta civilização. Estávamos errados.

Das profundezas do espaço, as Fendas do Abismo abriram-se, libertando horrores indescritíveis. Criaturas conhecidas como Devoradores de Almas emergiram, atacando colónias inteiras e drenando as essências vitais de milhões. Os mais fracos caíram como folhas ao vento. Os mais fortes? Corrompidos e transformados em servos espectrais ao comando de uma única entidade: O Profeta Vazio.

A Ordem Celestial retaliou com tudo o que tinha. Tropas de choque, frotas inteiras, armas capazes de pulverizar planetas—tudo inútil. As criaturas não eram deste mundo. As suas forças aumentavam a cada batalha perdida, e a humanidade começou a ruir. A guerra não estava apenas a ser perdida. Já estava perdida.

Mas havia esperança.

Num pequeno planeta isolado, Turim-7, um único sobrevivente escapou ao massacre: Acolon, descendente de uma linhagem esquecida, este herdou um segredo que nem ele próprio conhecia: um fragmento celestial, caído há milénios e selado no seu planeta. Com a ajuda dos últimos anciãos, Acolon fundiu esse fragmento no seu próprio corpo, criando uma tecnologia híbrida capaz de resistir aos Devoradores e transformar a própria essência do Abismo contra eles.

Agora, armado com a única arma capaz de reverter a maré da guerra, Acolon tem uma missão, libertar os mundos caídos. Resgatar as almas roubadas e destruir o Profeta Vazio**.**

3.3 Elementos

Nesta secção serão apresentados os elementos que compõem o jogo, incluindo o personagem principal, os inimigos que o jogador enfrentará, as armas disponíveis para uso ao longo da aventura e as habilidades especiais que poderão ser desbloqueadas. Esses elementos serão fundamentais para criar uma experiência desafiadora e envolvente, permitindo ao jogador personalizar a sua abordagem e melhorar o desempenho durante a jornada.

3.3.1 Personagem Principal

O protagonista do jogo será um piloto experiente, que controla uma nave espacial (Figura 10) , projetada para enfrentar os desafios do universo. A nave possui um sistema de resistência especial que permite suportar até dois ataques antes de ser completamente destruída no terceiro impacto. Este sistema fornece uma margem de segurança ao jogador para reagir estrategicamente durante os combates mais intensos.

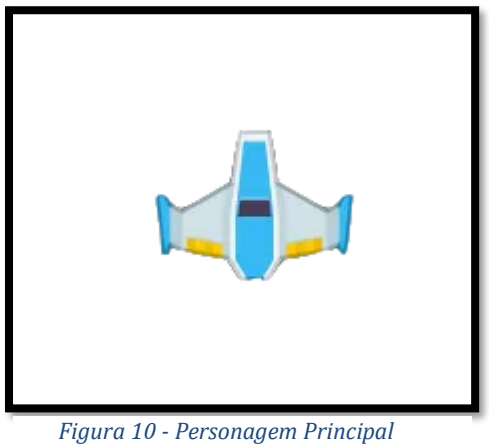

Figura 10 - Personagem Principal

Além disso, a nave possui um sistema de energia especial que permite ativar habilidades avançadas, que podem atingir múltiplos inimigos ao mesmo tempo. Esta energia é recarregada gradualmente conforme o jogador derrota os adversários, sendo que cada inimigo abatido contribui para a reposição parcial da barra de energia.

As armas do jogo serão cuidadosamente projetadas para atender diferentes estilos de combate. À medida que o jogador avança, poderá obter novas opções que permitirão melhorar as suas capacidades ofensivas e enfrentar inimigos mais difíceis com estratégias variadas, mantendo a progressão divertida e equilibrada.

3.3.2 Armas

A nave está equipada com um sistema de disparo automático, que constitui a sua principal forma de ataque. Esses disparos são contínuos, garantindo que o jogador esteja sempre preparado para enfrentar inimigos que surjam de qualquer direção.

Cada disparo retira uma parte da vida dos inimigos ou toda, sendo essencial para progredir pelas fases e superar os desafios do jogo.

Além do disparo padrão, o jogo também conta com melhorias que aumentam a eficácia dos ataques. Um exemplo notável é o Tiro Duplo, um upgrade temporário que proporciona uma capacidade ofensiva superior. Quando ativado, a nave passa a disparar dois projéteis simultaneamente, em paralelo, aumentando a área de impacto e duplicando o dano causado no mesmo intervalo de tempo. Esse aprimoramento permite ao jogador eliminar múltiplos inimigos com maior eficiência, tornando-se crucial para momentos de elevada dificuldade.

Outro elemento essencial é o Escudo Temporário, um power-up defensivo que concede uma proteção extra à nave. Enquanto ativo, o escudo impede que a nave perca uma vida ao ser atingida, absorvendo o impacto do ataque e desaparecendo imediatamente após sofrer dano.

O funcionamento automático do sistema de disparo permite que o jogador se concentre na movimentação e estratégia, tornando a experiência de combate fluida e intuitiva. Com o avanço das fases, o jogador terá a oportunidade de usufruir de diferentes melhorias que reforçam a jogabilidade, garantindo uma experiência dinâmica e desafiadora.

3.3.3 Inimigos

Nesta secção serão apresentados todos os inimigos que o jogador enfrentará ao longo do jogo. Tratam-se de diferentes tipos de naves inimigas, cada uma com características específicas que desafiam a estratégia do jogador. Entre elas estão naves com comportamentos variados: rápidas, mas mais frágeis; lentas, mas mais resistentes; e outras de equilíbrio padrão.

3.3.4 Nave normal

A Nave Normal (Figura 11) é o adversário padrão do jogo e aparece com maior frequência em todas as fases. Este tipo de nave caracteriza-se por seu equilíbrio, com uma velocidade mediana e uma resistência moderada.

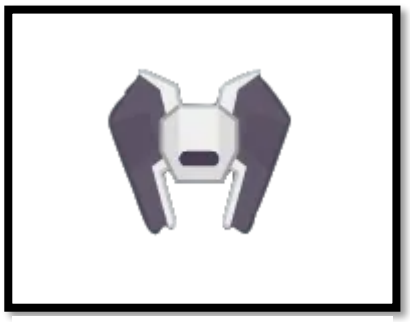

Figura 11 - Nave normal

Essas naves representam os inimigos mais comuns nas primeiras fases, ajudando o jogador a se familiarizar com os controles e com a dinâmica de combate. Com o progresso no jogo, as Naves Normais continuam a aparecer em conjunto com outros tipos de inimigos, formando grupos desafiantes que exigem estratégia e reflexos para lidar com diferentes comportamentos ao mesmo tempo.

A simplicidade da Nave Normal faz dela uma oponente ideal para preparar o jogador para os desafios mais complexos que virão, garantindo uma progressão natural na dificuldade sem sobrecarregar o jogador nos momentos iniciais.

3.3.5 Nave rápida

A Nave Rápida (Figura 12) destaca-se pela sua alta velocidade e mobilidade, sendo um dos inimigos mais desafiantes no jogo. Distintivamente da Nave Normal, que se desloca apenas em linha reta, a Nave Rápida tem a capacidade de se mover também na diagonal, tornando-a ainda mais imprevisível e difícil de acertar.

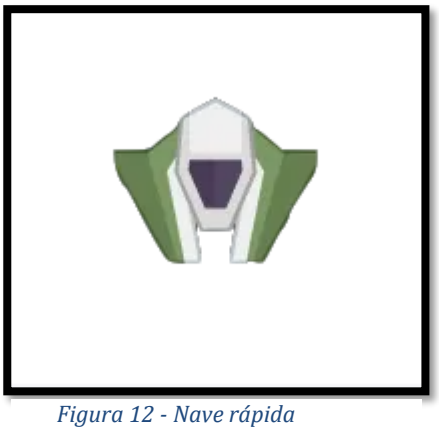

Figura 12 - Nave rápida

Sua fragilidade em termos de resistência é compensada pela velocidade aumentada e pela capacidade de flanquear o jogador com movimentos rápidos e inesperados. Essas características obrigam o jogador a ajustar sua estratégia e a manter atenção total ao campo de batalha, já que a Nave Rápida pode facilmente atravessar zonas de disparo desprotegidas.

Este inimigo surge em maior quantidade nas fases intermediárias e combina-se frequentemente com outros tipos de naves para criar desafios variados e intensos. Apesar de sua resistência reduzida, a movimentação ágil permite que a Nave Rápida represente uma ameaça constante, exigindo reflexos apurados para ser derrotada.

A introdução de movimentos diagonais neste inimigo não só diversifica os padrões de combate, como também aumenta significativamente a complexidade e o dinamismo das batalhas.

3.3.6 Nave lenta

A Nave Lenta (Figura 13) é um adversário resistente, projetado para testar a capacidade do jogador de lidar com inimigos que demoram mais tempo para serem derrotados. Apesar de sua baixa velocidade, esta nave compensa sua lentidão com uma grande resistência, sendo capaz de suportar mais dano antes de ser destruída.

Figura 13 - Nave lenta

Este tipo de nave move-se exclusivamente em linha reta, tornando seus movimentos previsíveis. No entanto, sua presença no campo de batalha cria uma pressão constante, especialmente em situações onde surgem em grande número ou em conjunto com outros tipos de naves, como as Naves Rápidas. A Nave Lenta é mais comum em fases avançadas, onde sua resistência e movimentos deliberados são estrategicamente combinados com ataques de outras naves, forçando o jogador a equilibrar suas prioridades entre eliminar ameaças rápidas e lidar com alvos mais resistentes.

Esse inimigo serve como um elemento tático no jogo, exigindo atenção para ser eliminado. Embora seus ataques não sejam rápidos, a resistência da Nave Lenta representa uma ameaça significativa, especialmente quando alinhada com outras forças no campo de combate.

3.3.7 Boss

O Boss Final (Figura 14) é a maior e mais temível nave inimiga do jogo, projetada para ser o auge da dificuldade e um verdadeiro teste das habilidades do jogador. Dotado de grande resistência, movimentos imprevisíveis e ataques perigosos, o boss apresenta múltiplas fases de combate, cada uma mais desafiadora que a anterior.

Figura 14 - Nave (Boss)

Ao contrário das outras naves inimigas, o Boss Final possui a capacidade de disparar projéteis diretamente contra o jogador, forçando-o a desviar-se continuamente enquanto tenta atacar. Esses projéteis variam em intensidade e frequência, criando padrões complexos de ataque que aumentam conforme o combate avança.

Além de seus próprios ataques, o boss pode invocar qualquer um dos três tipos de naves inimigas – Normais, Rápidas ou Lentas – para auxiliá-lo durante a luta. Essa habilidade adiciona um nível extra de desafio, já que o jogador precisará lidar simultaneamente com as táticas do boss e com as naves auxiliares, que surgem para criar confusão e dificultar os ataques diretos.

3.3.8 Sistema de vidas

O jogador terá um sistemas de vidas que indicará o estado da nave durante o jogo. A nave possui um máximo de 3 pontos de vida. Cada vez que a nave é atingida, perde um ponto de vida, e no terceiro ataque, a nave é destruída, resultando no fim de jogo.

Além disso, a integridade das naves inimigas varia de acordo com o tipo, sendo que as Naves Normais possuem vida moderada, enquanto as Naves Rápidas têm menor resistência devido à sua agilidade, e as Naves Lentas destacam-se por sua alta durabilidade.

Tabela 1 - Tabela de vida dos personagens

Inimigos	Vida	Vida do jogador
Nave normal	100	3
Nave rápida	50	
Nave lenta	200	
Nave Boss	1000	

3.3.9 Power-ups

Os power-ups desempenham um papel essencial no progresso e na experiência do jogador, proporcionando habilidades temporárias ou melhorias específicas que facilitam o combate e aumentam as chances de sobrevivência. Espalhados pelo campo de batalha ou liberados ao derrotar certos inimigos, esses itens incentivam uma exploração estratégica e adicionam algum dinamismo ao gameplay.

Neste capítulo, serão apresentados os diferentes tipos de power-ups disponíveis no jogo, destacando suas funções e impacto nas mecânicas. Cada power-up foi projetado para oferecer uma vantagem única ao jogador, complementando sua nave e permitindo-lhe superar desafios ainda mais difíceis.

- Tiro duplo

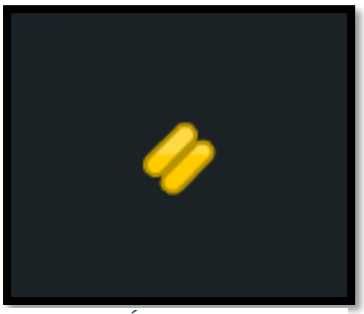

Figura 15 - Ícone de tiro-duplo

O Tiro Duplo é uma das melhorias mais impactantes no combate, aumentando significativamente a capacidade ofensiva da nave. Ao recolher este power-up, representado por dois projéteis dourados juntos (Figura 15), a nave passa a disparar dois tiros paralelos em vez de um único. Isso permite cobrir uma área maior e duplicar o dano infligido aos inimigos. O efeito do Tiro Duplo é ativado imediatamente após a recolha e tem uma duração de 20 segundos. Durante esse período, o jogador pode utilizar essa vantagem para eliminar grandes grupos de inimigos de forma mais rápida e eficiente. No entanto, como o efeito é temporário, o jogador deve utilizá-lo estrategicamente para maximizar sua eficácia nos momentos mais críticos do combate.

Ao permitir ataques mais amplos e eficazes, o Tiro Duplo torna-se uma vantagem essencial para lidar com grandes quantidades de inimigos, aumentando as chances de eliminar os alvos de forma mais rápida. Apesar das suas vantagens, o efeito deste power-up é temporário, exigindo que o jogador aproveite o período ativo de forma estratégica para maximizar o impacto nos momentos mais críticos do combate.

- Escudo temporário

Figura 16 - ícone de escudo temporário

O Escudo Temporário é um power-up defensivo que concede uma proteção extra à nave durante 30 segundos. Quando ativado, este power-up impede que a nave perca uma vida ao ser atingida, absorvendo o impacto do ataque e desaparecendo imediatamente após sofrer dano. Caso o jogador não seja atingido nesse intervalo, o escudo expira naturalmente ao fim do tempo estipulado. Este power-up é representado por um ícone de escudo prateado (Figura 16), e, enquanto estiver ativo, a nave exibirá um brilho cinzento ao seu redor, sinalizando a sua proteção temporária.

O Escudo Temporário é especialmente útil em situações de alta dificuldade, como ondas intensas de inimigos ou batalhas contra chefes, oferecendo uma margem de segurança extra para o jogador. Utilizado estrategicamente, pode ser a chave para superar desafios críticos e prolongar a sobrevivência na partida.

- Aumento de velocidade

O Aumento de Velocidade é um power-up que melhora a mobilidade da nave, permitindo que esta se mova mais rapidamente pelos lados durante 20 segundos. Representado por um ícone de setas azuis apontando para a direita e para a esquerda (Figura 17), este power-up é ativado imediatamente após a recolha.

Figura 17 - ícone de velocidade de ataque

Com um deslocamento mais ágil, o jogador pode desviar-se com maior eficácia dos ataques inimigos e reposicionar-se rapidamente para situações estratégicas. O Aumento de Velocidade é especialmente vantajoso em momentos de combate intenso, permitindo escapar de ataques sucessivos ou reposicionar-se rapidamente para melhor aproveitar power-ups e evitar zonas de perigo.

- Escudo dourado

O Escudo Dourado funciona de forma semelhante ao Escudo Temporário, proporcionando uma defesa adicional contra um ataque inimigo. No entanto, ao contrário da versão temporária, este escudo não desaparece ao fim de 30 segundos. Ele permanece ativo até que a nave sofra um impacto, bloqueando o primeiro ataque recebido e desaparecendo em seguida.

Figura 18 - ícone de escudo dourado

Representado por um ícone de escudo dourado (Figura 18), este power-up é uma ferramenta valiosa para jogadores que desejam uma segurança adicional ao enfrentar inimigos mais desafiadores ou momentos críticos do jogo. Ao permitir que o jogador se concentre no combate sem se preocupar com um golpe imediato, o Escudo Dourado

proporciona uma camada extra de proteção que pode ser decisiva para a sobrevivência em fases mais avançadas.

3.4 Gameplay

O jogo começa na área de batalha, onde o jogador assume o controlo da sua nave com o objetivo de destruir todas as naves inimigas sem que estas atinjam o jogador fazendo este ficar sem vidas e perder o jogo.

3.4.1 Perspetiva do jogador

O jogador controla a nave espacial utilizando as teclas direcionais do teclado ou o rato, dependendo das configurações previamente definidas. O movimento é limitado ao eixo horizontal, permitindo ao jogador deslocar-se para a esquerda e para a direita, o que simplifica os controlos e garante precisão nas ações. A nave dispara automaticamente projéteis contra os inimigos, retirando do jogador a necessidade de gerir os disparos manualmente, permitindo-lhe focar-se em desviar-se de ataques inimigos e recolher power-ups que aparecem ao longo do cenário.

O sistema de vidas da nave é visualmente representado no ecrã através de um indicador numérico ou gráfico no canto superior esquerdo. O jogador começa o jogo com três vidas e perde uma sempre que a nave é atingida por um disparo inimigo ou colide com outro objeto perigoso, como uma nave alienígena. Sempre que o jogador perde uma vida, é apresentado um breve feedback visual, como o piscar da nave ou uma mudança temporária de cor, para sinalizar o impacto. Quando todas as vidas são perdidas, o jogo termina automaticamente, e o jogador pode optar por reiniciar a partida ou regressar ao menu principal.

Ao longo do jogo, surgem power-ups no campo de batalha, os quais fornecem vantagens temporárias para ajudar o jogador a lidar com as crescentes dificuldades. Estes itens incluem o Escudo Temporário, que protege a nave contra danos durante 30 segundos e é representado por um brilho em torno da nave, indicando que os disparos inimigos não causarão dano durante este período. Outro exemplo é o Tiro Duplo, um power-up que permite que a nave dispare projéteis ao mesmo tempo 2 projeteis ao

invés de um aumentando a sua eficiência ofensiva. Estes itens aparecem em posições estratégicas e seguem um padrão aleatório de movimento.

O jogador deve utilizar os seus reflexos para desviar-se de projéteis inimigos e posicionar-se corretamente para recolher power-ups ou eliminar inimigos prioritários. O design das mecânicas, que combina movimentação limitada com disparos automáticos e uma gestão de recursos básica através dos power-ups, exige que o jogador desenvolva uma estratégia eficaz e reaja rapidamente às mudanças na dinâmica do jogo. Essa combinação de elementos mantém a experiência envolvente e progressivamente desafiante.

3.4.2 Perspetiva do inimigo

No que diz respeito às naves inimigas, estas são geradas de forma contínua ao redor da área de batalha, criando um ambiente de pressão constante para o jogador. Os inimigos variam em termos de comportamento, com alguns tipos a moverem-se diretamente em direção à nave do jogador, enquanto outros adotam padrões de voo mais imprevisíveis para dificultar os ataques. Todas as naves alienígenas procuram invadir a zona ocupada pelo jogador, sendo a principal ameaça a sua capacidade de provocar colisões diretas. Importa destacar que os inimigos atacam em grupo, sendo gerados em vagas de intensidade crescente, à medida que o jogador avança nos níveis.

Cada nave inimiga possui um número específico de pontos de vida, e a destruição destas depende diretamente da força dos disparos automáticos da nave do jogador. Por exemplo, naves mais pequenas podem ser destruídas com um único disparo, enquanto naves de maior porte ou mais avançadas requerem vários impactos antes de serem eliminadas. Algumas destas naves possuem habilidades especiais, como desviar-se dos projéteis disparados pelo jogador, tornando-as uma ameaça mais desafiante.

Além disso, há a possibilidade de certos elementos do cenário, como meteoros ou destroços espaciais, interferirem nos combates. Em situações onde os disparos inimigos colidem com esses objetos, o comportamento das naves pode ser afetado, como uma aceleração repentina em direção ao jogador ou uma alteração no padrão de movimento. Esta dinâmica aumenta o grau de imprevisibilidade no jogo e exige que o jogador esteja atento a múltiplas ameaças em simultâneo.

A destruição de uma nave inimiga pode ainda beneficiar o jogador, pois existe uma probabilidade de estas largarem power-ups valiosos após serem eliminadas. Esta mecânica incentiva o jogador a manter uma postura ofensiva estratégica, considerando não só a sobrevivência, mas também a otimização do desempenho através da recolha de melhorias durante as batalhas. Este equilíbrio entre ataque e defesa define o ritmo acelerado e estratégico do gameplay.

3.5 Estrutura de dificuldade

O jogo é dividido em cinco níveis distintos, cada um projetado com um grau de dificuldade crescente, de forma a oferecer uma experiência progressiva e desafiadora ao jogador. No início do jogo, o jogador é apresentado ao nível 1, que funciona como uma introdução às mecânicas principais e aos inimigos básicos. À medida que os níveis avançam, novos tipos de inimigos, com padrões de ataque mais complexos e maior resistência, começam a surgir, aumentando a dificuldade gradualmente.

Cada nível introduz elementos novos para diversificar o gameplay, como diferentes formações de inimigos, maior velocidade de movimento das naves alienígenas, e até mesmo uma maior frequência de disparos dos adversários. Desta forma, os jogadores são desafiados a adaptar continuamente as suas estratégias para sobreviver. A progressão para o próximo nível só é permitida após o jogador derrotar todos os inimigos do nível atual. O jogo também incentiva a recolha de power-ups durante os combates, ajudando o jogador a lidar com o aumento da dificuldade.

O quinto e último nível apresenta o maior desafio para o jogador, sendo este uma batalha épica contra um Boss. Este inimigo especial distingue-se dos restantes por ser significativamente mais resistente e por apresentar ataques únicos, como disparos, mudanças de padrão de movimento.

3.5.1 Sistema de pontuação

O sistema de pontuação no jogo é projetado para recompensar o jogador de forma simples e proporcional ao desafio enfrentado. Ao destruir naves inimigas, o jogador acumula pontos que refletem o tipo e a dificuldade dos alvos. As naves normais, que são mais fáceis de abater, oferecem 10 pontos por cada eliminação, constituindo o tipo mais comum de pontuação. Já as naves rápidas e lentas, que apresentam desafios

adicionais devido à sua velocidade elevada ou maior resistência, proporcionam 20 pontos por cada destruição, recompensando o esforço extra necessário para as atingir.

No nível final, o jogador enfrentará o boss, uma ameaça significativa que, ao ser derrotado, garante 1000 pontos, representando a maior recompensa individual do jogo. Este sistema de pontuação incentiva o jogador a melhorar a sua precisão e estratégia, ao mesmo tempo que torna cada progresso mais recompensador e motivador.

4 Protótipo

Neste capitulo será apresentado o protótipo desenvolvido para demonstrar a jogabilidade e os principais conceitos do jogo final. Sendo um protótipo, foram implementadas apenas as mecânicas essenciais, de forma simplificada. Como o jogo foi desenvolvido em 2D utilizando a engine Unity, serão descritas as ferramentas utilizadas no seu desenvolvimento, bem como o código implementado para cada funcionalidade essencial. Além disso, serão abordados os principais elementos do protótipo, como o jogador, os inimigos, e os disparos, destacando a sua importância na experiência de jogo.

4.1 Unity

O Unity é uma plataforma para o desenvolvimento de jogos, reconhecida pela sua versatilidade e facilidade de uso. Desde seu lançamento em 2005, essa engine passou por uma evolução notável, apresentando um leque completo de ferramentas que facilitam a criação de jogos em 2D e 3D de maneira eficiente. Com um compromisso contínuo com a inovação, o Unity se firmou como uma das escolhas mais procuradas por desenvolvedores, desde os novatos até os experientes, sendo amplamente empregado na elaboração de experiências interativas e envolventes.

A interface do Unity foi projetada para ser de fácil compreensão, constituída por diversas janelas e ferramentas que oferecem acesso imediato às funcionalidades fundamentais do motor de jogo. Cada componente tem um papel significativo no processo de desenvolvimento, possibilitando a organização, manipulação e visualização prévia de recursos e cenários.

Figura 19 - Interface do Unity[6]

(A) Toolbar: A Barra de Ferramentas encontra-se na parte superior da interface e proporciona acesso à conta Unity e aos serviços em nuvem. Inclui também comandos fundamentais, como os controlos para o modo Play, que ativa a simulação do jogo, histórico de desfazer, pesquisa no Unity, definições de visibilidade de camadas e opções para personalizar o layout do editor[6].

(B) Janela Hierarchy: A Hierarquia organiza todos os GameObjects de forma hierárquica. É uma representação textual que reflete a estrutura do cenário e mostra como os diferentes objetos estão interligados, facilitando a visualização e gestão das relações entre eles[6].

(C) Game View: A Vista do Jogo simula a aparência final do jogo tal como é renderizado pelas câmaras da cena. Quando o botão Play é ativado, esta janela exibe a jogabilidade em tempo real, permitindo ao desenvolvedor testar e ajustar o funcionamento do jogo[6].

(D) Scene View: A Vista da Cena é o espaço onde se navega e edita visualmente o conteúdo da cena. Permite alternar entre visões 2D e 3D, conforme as necessidades do projeto, sendo essencial para posicionar e ajustar os GameObjects com precisão[6].

(E) Overlays: Localizados na Scene View, os Overlays incluem ferramentas básicas para manipular objetos e o cenário. Além disso, permitem a personalização com

ferramentas adicionais, otimizando o fluxo de trabalho de acordo com as preferências do utilizador[6].

(F) Inspector: A janela Inspetor exibe todas as propriedades do GameObject selecionado, possibilitando edições detalhadas. É uma ferramenta versátil, cujo conteúdo se adapta automaticamente ao tipo de objeto escolhido, assegurando que o utilizador tem sempre acesso às configurações relevantes[6].

(G) Project Window: A janela Projeto funciona como a biblioteca de ativos (Assets) do projeto, exibindo todos os recursos importados, organizados de forma a facilitar o acesso e utilização durante o desenvolvimento[6].

(H) Status Bar: Situada na parte inferior, a Barra de Status fornece notificações sobre os processos em execução e oferece atalhos para ferramentas e configurações relacionadas, promovendo uma visão geral do estado do projeto[6].

Com esta divisão clara de funções, a interface do Unity demonstra-se eficiente, intuitiva e personalizável, sendo essencial para desenvolvedores que procuram um ambiente de trabalho funcional e produtivo[6].

Para a criação de um script em Unity basta clicar com o botão direito na zona G da Figura 17 e assim fica visível o menu da figura 18 (zona A) e de seguida clicar em "C# Script" e o script é criado automaticamente. Para adicionar este script a um objeto basta arrastar para o objeto pretendido na zona B da Figura 16.

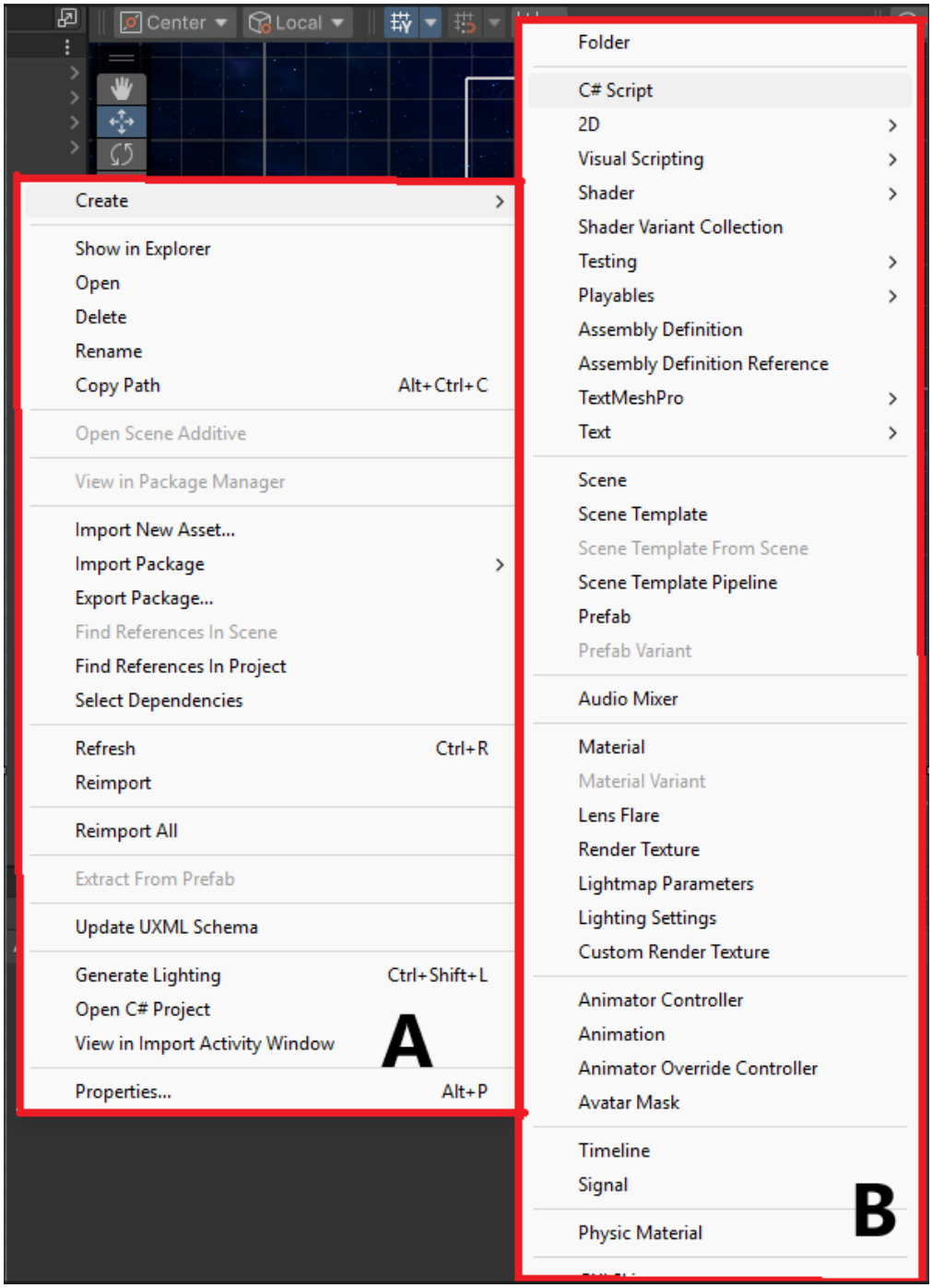

Figura 20 - Criação de um script em Unity

```
0 scene usages | 0 prefab usages
public class teste : MonoBehaviour
{
    // Start is called before the first frame update
    Unity Message
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message
    void Update()
    {

    }
}
```

Figura 21 - Script em branco

Uma vez criado em script ele vem já com duas funções inicializadas, como podemos observar na Figura 21, exatamente as funções "Start" e "Update". A função *Start* é chamada uma única vez quando o jogo é iniciado. Já a função *Update* é executada a cada fotograma, o que significa que, por padrão, o Unity a invoca 60 vezes por segundo, considerando uma taxa de 60 FPS.

4.1.1 Implementação do player

O primeiro elemento a ser criado foi o player pois este é a personagem principal do nosso jogo logo o que terá mais funcionalidades, e também o que se manifesta mais dentro do jogo. Comecei por criar o movimento do player, este tendo um movimento horizontal, para a esquerda e para a direita.

```
0 references
void Update()
{
    Vector3 movimento = transform.position;

    // Movimento horizontal
    movimento.x += Input.GetAxis("Horizontal") * velocidade * Time.deltaTime;

    // Limitar o movimento do jogador dentro dos limites
    movimento.x = Mathf.Clamp(movimento.x, limiteEsquerdo, limiteDireito);

    transform.position = movimento;


}
```

Figura 22 - Código para movimento do player

Na figura 22 pode-se observar o excerto de código que fazer parte do script “Player”, onde esta implementado o movimento pelas teclas de setas do teclado, onde estas são usadas para controlar a nave dentro de jogo. Por exemplo se for pressionada a tecla “seta para a direita” a nave ira andar sobre o eixo do X para a esquerda com o valor da velocidade definida. Fio também implementado um limite de movimento da nave para esta n sair dos limites do ecrã do jogador.

```
// Método para disparar tiros
0 references
void Disparar()
{
    Instantiate(meuTiro, transform.position, transform.rotation);
}
```

Figura 23 - código para disparar

Na Figura 23, apresenta-se o método "Disparar()", responsável por criar os projéteis da nave. Este método utiliza a função "Instantiate()" para gerar uma nova instância do objeto correspondente ao tiro, na mesma posição e rotação da nave, garantindo que os disparos sejam efetuados corretamente no jogo.

```
0 references
void Start()
{
    // Dispara automaticamente a cada 'intervaloTiro' segundos
    InvokeRepeating("Disparar", 0.4f, intervaloTiro);
}
```

Figura 24 - código de disparo automático

Na Figura 24, demonstra-se a forma como o método "Disparar()" é invocado automaticamente dentro da função "Start()", através do "InvokeRepeating()". Este comando executa o método de forma repetida em intervalos de tempo predefinidos, assegurando que a nave efetue disparos contínuos sem necessidade de intervenção do jogador. Dessa forma, a mecânica de disparo automático é implementada de maneira eficiente, proporcionando uma jogabilidade mais fluida e dinâmica.

```
// Update is called once per frame
0 references
void Update()
{
    if (transform.position.y > Camera.main.orthographicSize + 1)
    {
        Destroy(gameObject);
    }
}
```

Figura 25 - código para atribuir velocidade ao tiro

Na Figura 25, é apresentada o método “Start()” da classe “tiro”, que é responsável por inicializar a velocidade do projétil no momento em que ele é criado. Dentro desse método, a variável meuRB.velocity é configurada para que o projétil se mova para cima, na direção positiva do eixo Y, com uma velocidade definida pela variável pública velocidade. A física de movimento do projétil é controlada pelo componente Rigidbody2D associado a ele, fazendo com que o projétil se mova de maneira consistente ao longo do tempo.

O uso da função Vector2.up permite que o projétil se mova para cima da tela com uma velocidade constante, o que garante que o disparo tenha um movimento fluido e controlado. Esse comportamento é essencial para a jogabilidade, já que o projétil deve se mover rapidamente para alcançar os inimigos e proporcionar uma experiência dinâmica ao jogador.

```
// Update is called once per frame
0 references
void Update()
{
    if (transform.position.y > Camera.main.orthographicSize + 1)
    {
        Destroy(gameObject);
    }
}
```

Figura 26 - código para destruir o tiro quando ele sai do ecrã

Na Figura 26, vemos o método Update(), que é executado a cada quadro do jogo e tem como objetivo verificar a posição do projétil em relação aos limites da tela. Se o projétil ultrapassar o limite superior da câmera (ou seja, se sua posição Y for maior que o tamanho da câmera), o projétil é destruído para evitar que objetos fora da tela permaneçam no jogo e consumam recursos desnecessários.

Aqui, Camera.main.orthographicSize é utilizado para determinar o limite superior da câmera em um jogo 2D, e o valor "+1" é adicionado para garantir que o projétil seja destruído apenas depois de sair totalmente da tela. Esse código permite manter a cena limpa e otimizar o desempenho do jogo.

```
//tiro acerta no inimigo
0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    Debug.Log("acertei num inimigo");
    Destroy(collision.gameObject);
    //destruir o tiro depois de acertar no inimigo
    Destroy(gameObject);
}
```

Figura 27 - código se o tiro acertar um inimigo

Por fim, na Figura 27, da classe Tiro implemento o método OnTriggerEnter2D(), responsável pela deteção de colisões entre o projétil e outros objetos do jogo. Quando o projétil colide com um inimigo, o método destrói tanto o projétil quanto o inimigo.

Dentro do método, Destroy(collision.gameObject) destrói o inimigo atingido, enquanto Destroy(gameObject) destrói o projétil.

4.1.2 Implementação do inimigo

O segundo elemento a ser implementado foi o inimigo, que tem como objetivo proporcionar ações básicas de interação para o jogo. Para isso, foi criada uma classe para o inimigo, a qual permite que ele se mova para baixo automaticamente e seja destruído ao sair da tela ou ao colidir com um tiro.

```
// Start is called before the first frame update
0 references
void Start()
{
    //quando inicio dar uma velocidade ao rigidbody
    meuRB.velocity = Vector2.down * velocidade;
    //valor do limite inferior
    limiteInferior = -Camera.main.orthographicSize - 1;
}
```

Figura 28 - código para controlar o movimento do inimigo

Como mostra a Figura 28 o método Start da classe inimigo é atribuído ao Rigidbody2D do inimigo uma velocidade inicial, fazendo com que ele se mova para baixo na tela de forma contínua.

Para que o inimigo se mova para baixo, utilizei o Vector2.down multiplicada pela velocidade definida. Além disso, o limite inferior do inimigo é calculado com base na altura da câmera, para garantir que o inimigo seja destruído quando sair da tela.

```
// Update is called once per frame
0 references
void Update()
{
    // Verifica se o inimigo saiu do ecrã e destrói
    if (transform.position.y < limiteInferior)
    {
        Destroy(gameObject);
    }
```

Figura 29 - código para destruir o inimigo quando este sai do ecrã

A cada quadro do jogo, no método Update como podemos ver na Figura 29, o código verifica se o inimigo passou do limite inferior da tela, destruindo-o caso tenha saído da área visível do jogador.

Este método garante que, assim que o inimigo sair da tela ele será destruído para liberar memória e evitar que o jogo tente renderizar um objeto fora da área visível.

```
    // Quando o inimigo colide com um tiro
0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    // Verifica se a colisão foi com o tiro
    if (collision.CompareTag("Tiro"))
    {
        Destroy(collision.gameObject);  // Destroi o tiro
        Destroy(gameObject);           // Destroi o inimigo
    }
}
```

Figura 30 - código de quando o inimigo colide com um tiro

O inimigo também interage com os tiros disparados pela nave do jogador. No método OnTriggerEnter2D como podemos ver na Figura 30, o código verifica se o inimigo colidiu com um objeto do tipo "Tiro". Caso sim, tanto o inimigo quanto o tiro são destruídos.

Dentro do método, Destroy(collision.gameObject) destrói o tiro, enquanto Destroy(gameObject) destrói a nave atingida.

```
0 references
void Start()
{
    // Calcula os limites visíveis com base na câmera
    Camera cam = Camera.main;
    float altura = cam.orthographicSize;
    float largura = altura * cam.aspect;

    xMin = -largura + 0.5f; // Pequeno ajuste para evitar spawn fora da tela
    xMax = largura - 0.5f;
    yMin = cam.transform.position.y + altura + 1f;  // Acima da câmera
    yMax = yMin + 2f;  // Pequena faixa acima do limite da câmera

    timer = tempospawn;
}
```

Figura 31 - código para definir limites onde crio inimigos

Quando o jogo é iniciado, o método Start é executado. Nele, os limites da tela são calculados com base na posição da câmera (Camera.main). A altura e largura da tela são obtidas através das propriedades orthographicSize da câmera e cam.aspect

(proporção da tela). Isso nos permite definir os valores de xMin, xMax, yMin e yMax, que determinam a área dentro da qual os inimigos aparecerão

A variável (timer) é iniciada com o valor de tempospawn que esta com o valor de 2 segundos, o que significa que o primeiro inimigo será gerado após esse tempo.

```
0 references
void Update()
{
    // Reduz o temporizador
    timer -= Time.deltaTime;

    // Dá spawn ao inimigo
    if (timer <= 0)
    {
        // Cria uma posição random dentro dos limites visíveis da câmera
        Vector3 posicao = new Vector3(Random.Range(xMin, xMax), Random.Range(yMin, yMax), 0);
        Instantiate(inimigo, posicao, Quaternion.identity);

        // Reinicia o temporizador
        timer = tempospawn;
    }
}
```

Figura 32 - código para criar o inimigo

No método Update como podemos ver na Figura 32, o temporizador é constantemente atualizado com base no tempo que passa. Quando o timer chega a zero, o método cria um novo inimigo. A posição do inimigo é gerada aleatoriamente dentro dos limites definidos pelos cálculos anteriores, usando o método Random.Range para as coordenadas x e y.

Após a criação do inimigo, o temporizador é reiniciado com o valor de tempospawn, fazendo com que o próximo inimigo seja gerado após o mesmo intervalo.

4.1.3 Cenário

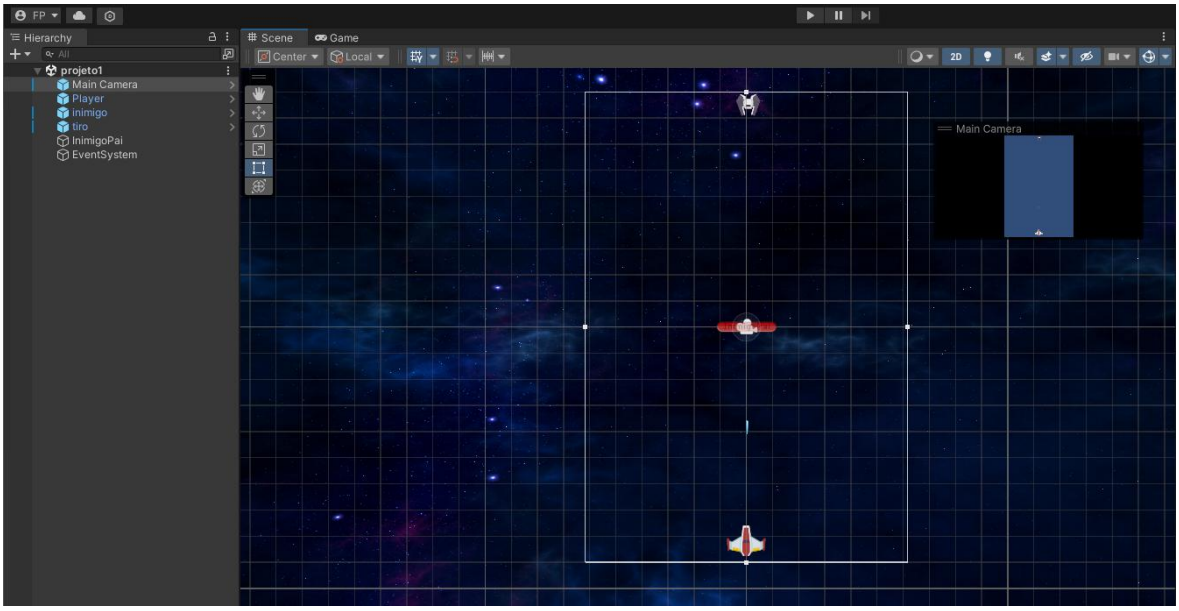

Figura 33 - Main câmera do Unity

O cenário do protótipo foi desenhado para representar uma versão simples do jogo final. No caso deste projeto, foi criado um ambiente espacial, onde o jogador controla uma nave que deve enfrentar inimigos. O fundo do cenário é uma imagem de galáxia, criando a atmosfera desejada para o jogo como podemos ver na Figura 33.

A cena foi montada utilizando objetos 2D fornecidos pelo Unity, e os principais elementos incluem a nave controlada pelo jogador, os inimigos e os projéteis. O jogo utiliza um sistema de movimentação horinzontal dentro da área delimitada e permite que o player ataque os inimigos utilizando tiros que se deslocam na vertical em direção aos alvos.

Figura 34 - Print da consola

Na Figura 34, é possível observar um print da consola do Unity exibindo mensagens que confirmam o acerto de um tiro em um inimigo.

Sempre que um tiro acerta com um inimigo, um evento é ativado no script responsável pelo comportamento do projétil, registando a colisão e imprimindo a mensagem na consola. Estes mensagens permitem validar que o sistema de combate está a funcionar de forma correta, garantindo que os tiros estão a ser detetados e aplicando os efeitos desejados, como a redução de vida do inimigo ou sua destruição.

Essa funcionalidade é essencial para o desenvolvimento do jogo, pois fornece feedback imediato sobre as interações entre os elementos do cenário.

5 Conclusão

A partir da análise detalhada de jogos como *Galaga*, *Space Hunter* e *Xenon*, foi possível identificar mecânicas essenciais para a criação do jogo *Galactic Defender*. Essa pesquisa forneceu uma visão aprofundada sobre os elementos que tornam esses títulos atrativos e desafiadores, permitindo-nos entender melhor as dinâmicas de combate, movimentação e progressão de níveis. Ao examinar esses jogos, foi possível perceber como a simplicidade nas mecânicas de disparo e movimentação pode ser combinada com desafios crescentes, criando uma experiência de jogo envolvente e dinâmica. As referências obtidas durante essa análise foram decisivas na criação do protótipo inicial, permitindo a implementação de mecânicas chave como o movimento da nave e o disparo automático, fundamentais para a jogabilidade de *Galactic Defender*.

No desenvolvimento do protótipo, a utilização de ferramentas como o Unity foi essencial não apenas para a criação dos elementos visuais, mas também para a implementação das funcionalidades do jogo. O Unity permitiu-nos integrar eficientemente os diferentes sistemas do jogo, desde a física do movimento até o gerenciamento de eventos de disparo e colisão. Essa fase inicial do projeto foi crucial para validar as ideias de design e entender como unir as mecânicas de jogo com os aspetos técnicos, proporcionando uma experiência coesa e divertida.

Os objetivos do Projeto I foram alcançados com sucesso, incluindo a análise do estado da arte, a criação do Game Design Document (GDD) e a implementação das mecânicas básicas do jogo. Entre os pontos fortes, destacam-se a análise detalhada do género, e a jogabilidade simples e acessível. No entanto, o protótipo ainda apresenta limitações, como a falta de variedade de inimigos e power-ups, a necessidade de ajustes no balanceamento de dificuldade e a ausência de uma identidade visual e sonora mais definida. No próximo passo do desenvolvimento, o foco será expandir o jogo com novas funcionalidades, como inimigos variados, power-ups e sistemas de progressão, além de refinar as mecânicas existentes para garantir uma experiência mais imersiva e desafiadora. Com base nas lições aprendidas, o projeto avançará para a fase final, visando criar um jogo completo e recompensador para os jogadores.

5.1 Trabalho Futuro

O trabalho realizado até o momento serve como base para a evolução do jogo, funcionando como um protótipo essencial para a próxima etapa. No Projeto II, a proposta é ampliar as mecânicas que já foram estabelecidas, com a intenção de proporcionar uma experiência de jogo mais envolvente e desafiadora. Nesta fase, o foco será a continuidade do desenvolvimento, incluindo a implementação de novos elementos, como níveis de dificuldade variados, melhorias na interatividade e a introdução de novas mecânicas, tudo para elevar a qualidade da jogabilidade. A meta final é criar um jogo completo e otimizado, apto para ser lançado e compartilhado com os jogadores, permitindo que eles vivenciem e se divirtam com "Galactic Defender".

6 Referencias

[1] «Unity»,Wikipédia, a enciclopédia livre. Acedido:13 de dezembro de 2024.[Em linha]. Disponível em: https://pt.wikipedia.org/wiki/Unity

[2] «Space Hunting», Addicting Games. Acedido:13 de dezembro de 2024.[Em linha]. Disponível em: https://www.addictinggames.com/shooting/space-hunting

[3] «MarketJS», Market JS. Acedido:13 de dezembro de 2024.[Em linha].Disponível em: https://www.marketjs.com/

[4] «What is Unity?», GameDev Academy. Acedido: 23 de dezembro de 2024. [Em linha]. Disponível em: https://gamedevacademy.org/what-is-unity/

[5] «Benefits of Unity Game Engine – Why it's Still Number One», N-iX Game Studio. Acedido: 23 de dezembro de 2024. [Em linha]. Disponível em: https://gamestudio.n-ix.com/benefits-of-unity-game-engine-why-its-still-number-one/

[6] «Unity (game engine)», Wikipedia. [Em linha]. Disponível em: https://en.wikipedia.org/wiki/Unity_(game_engine). Acedido em: 26 de dezembro de 2024.

[7] «Unity Documentation: Using the Editor», Unity Technologies. Acedido: 23 de dezembro de 2024. [Em linha]. Disponível em: https://docs.unity3d.com/Manual/UsingTheEditor.html

[8] «Galaga», RetroGames. Acedido: 2 de janeiro de 2025. [Em linha]. Disponível em: https://www.retrogames.cz/play_018-NES.php

[9] «Galaga», Wikipedia. [Em linha]. Disponível em: https://en.wikipedia.org/wiki/Galaga . Acedido em: 2 de dezembro de 2025.

[10] «RetroGames.cz», [Em linha]. Disponível em: https://www.retrogames.cz/konzole_NES.php. Acedido em: 2 de janeiro de 2025

[11] «História dos jogos eletrônicos», Wikipédia, a enciclopédia livre, [Em linha]. Disponível em: https://pt.wikipedia.org/wiki/Hist%C3%B3ria_dos_jogos_eletr%C3%B4nicos. Acedido em: 3 de janeiro de 2025.

[12] «History of video games», Wikipedia, The Free Encyclopedia, [Em linha]. Disponível em: https://en.wikipedia.org/wiki/History_of_video_games. Acedido em: 3 de janeiro de 2025.

[13] «O que é uma engine de jogos?», Tecnoblog, [Em linha]. Disponível em: https://tecnoblog.net/responde/o-que-e-uma-engine-de-jogos/. Acedido em: 3 de janeiro de 2025.

[14] "History of Galaga," *Galaga.com*, [Em linha]. Disponível em: https://galaga.com/en/history/galaga.php. Acedido em: 4 de janeiro de 2025.

[15] A. Oliveira, "Galaga: arcade, 40 anos de história," *GameBlast*, 9 Nov. 2021. [Em linha]. Disponível em: https://www.gameblast.com.br/2021/11/galaga-arcade-40-anos.html. Acedido em: 4 de janeiro de 2025.

[16] M. McNamara, "Galaga," *Gaming History 101*, 22 Mar. 2012. [Em linha]. Disponível em: https://gaminghistory101.com/2012/03/22/galaga/. Acedido em: 4 de janeiro de 2025.

[17] J. McNew, "Galaga Review," *HubPages*, [Em linha]. Disponível em: https://discover.hubpages.com/games-hobbies/Galaga-Review. Acedido em: 5 de janeiro de 2025.

[18] "Nintendo Entertainment System," *Wikipedia*, [Em linha]. Disponível em: https://pt.wikipedia.org/wiki/Nintendo_Entertainment_System. Acedido em: 5 de janeiro de 2025.

[19] "Gameplay - Xenon", *Lemon Amiga*, disponível em: https://www.lemonamiga.com/games/docs.php?id=1835, acessado em: 8 Jan. 2025.

[20] "Xenon (video game)", *Wikipedia*, disponível em: https://en.wikipedia.org/wiki/Xenon_(video_game), acessado em: 8 Jan. 2025.

[21] "Xenon - DOS Game", *RetroGames.cz*, disponível em: https://www.retrogames.cz/play_698-DOS.php, acessado em: 9 Jan. 2025.

[22] "Xenon", *MobyGames*, disponível em: https://www.mobygames.com/game/1553/xenon/, acessado em: 9 Jan. 2025.

[23] "Atari ST", *Wikipedia*, disponível em: https://pt.wikipedia.org/wiki/Atari_ST, acessado em: 30 Jan. 2025.

[24] "The Atari ST - Nostalgia Nerd", disponível em: https://www.nostalgianerd.com/the-atari-st/, acessado em: 10 Jan. 2025.

[25] "Xenon - AtariCrypt", disponível em: https://ataricrypt.blogspot.com/2020/01/xenon.html, acessado em: 10 Jan. 2025.