Return of the Stolen Souls

Miguel André Santos Antunes
Nº 20181173

Orientador
Paulo Alexandre 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 Informática e multimédia realizada sob a orientação científica do Professor Paulo Alexandre Correia da Silva Neves, do Instituto Politécnico de Castelo Branco.

Junho de 2024

Composição do júri

Presidente do júri
Doutora, Arminda Lopes
Professora Coordenadora ESTCB

Vogais
Doutor, Arlindo Silva
Professor Adjunto ESTCB

Mestre, Paulo Alexandre Neves
Professor Adjunto ESTCB

Resumo

O presente relatório documenta o trabalho realizado no âmbito da unidade curricular de Projeto II, do terceiro ano do curso de Licenciatura em Informática e Multimédia da Escola Superior de Tecnologia de Castelo Branco. Este relatório tem como objetivo descrever, documentar e relatar as atividades desenvolvidas na criação de um jogo intitulado *Return of The Stolen Souls*.

O documento começa com a apresentação do GDD - Game Design Document, elaborado na cadeira de Projeto I, no qual estão descritas todas as funcionalidades que devem ser implementadas no jogo.

Os recursos desenvolvidos para enriquecer a experiência do jogo foram o ponto de partida para toda a implementação. Com a utilização das ferramentas Blender, Unity e Krita, foi possível criar objetos tridimensionais, imagens, texturas, animações e efeitos visuais.

A implementação do jogo foi a fase mais importante do projeto, pois serviu para desenvolver o jogo implementando tudo o que foi desenvolvido anteriormente e aplicar diversos conhecimentos que foram adquiridos ao longo do tempo. Neste capítulo é apresentado a descrição de tudo o que foi implementado através de fluxogramas e código desenvolvido.

São apresentadas também as respostas a um formulário fornecido a um grupo de pessoas que obtiveram acesso ao jogo de modo a testarem o mesmo. Neste capítulo são mostradas as principais opiniões e melhorias fornecidas pelos utilizadores através de vários gráficos e respostas.

Este relatório termina com a uma conclusão, onde é feita a análise do trabalho realizado.

Palavras-chave

Return of Stollen Souls, GDD, Jogo RPG, Jogo 3D

Abstract

This report documents the work carried out within the scope of the Project II curricular unit, of the third year of the Degree in Informatics and Multimedia at the Escola Superior de Tecnologia de Castelo Branco. This report aims to describe, document and report the activities developed in the creation of a game entitled Return of The Stolen Souls.

The document begins with the presentation of the GDD - Game Design Document, prepared in the Project I curricular unit, which describes all the features that must be implemented in the game. Features designed to enrich the game experience were the starting point for this implementation. Using the Blender, Unity and Krita tools, it was possible to create three-dimensional objects, images, textures, animations and visual effects.

The implementation of the game was the most important phase of the project, as it served to develop the game by implementing everything that was previously developed and applying the knowledge that had been acquired over time. This chapter presents a description of everything that was implemented through flowcharts and developed code.

Are also present some answers to a form provided to a group of people who gained access to the game and were able to test it. In this chapter are shown the main opinions and improvements provided by users through various graphs and responses.

This report ends with a conclusion, where the work carried out is analysed.

Keywords

Return of Stollen Souls, GDD, RPG Game, 3D Game

Índice geral

Índice de figuras

Lista de tabelas

Lista de abreviaturas, siglas e acrónimos

GDD – Game Design Document
RPG – Role Playing Game

1 Introdução

Este projeto consiste na criação de um jogo eletrónico 3D que surgiu com a ideia de dar crescimento a um antigo projeto com o mesmo nome, porém uma versão mais simples em 2D desenvolvido em flash. Todo este foi desenvolvido como perspetiva a poder ser distribuído para várias plataformas como um método de entretenimento nos tempos livres.

Para o desenvolvimento do jogo foi utilizado o software Unity como plataforma principal. Este software disponibiliza um ambiente poderoso para a criação de jogos eletrónicos. Além deste, recorreu-se também a dois outros softwares para desenvolvimento de elementos gráficos, o Kryta e Blender.

A base fundamental do projeto encontra-se nas diversas funcionalidades de jogo.

Ao longo deste documento, são descritos todos os elementos criados para poder dar vida ao jogo que vão desde os inimigos até aos componentes desenvolvidos para dar vida aos diferentes níveis presentes dentro do jogo. São também explicadas todas as funcionalidades aplicadas através do seu código desenvolvido.

Este projeto focou-se também em obter opiniões externas através de um formulário desenvolvido e enviado juntamente com o jogo a um determinado grupo de pessoas. As diversas respostas obtidas mostraram-se bastante úteis para descobrir alguns problemas encontrados ao longo da sua experiência de jogo.

1.1 Objetivos

Os objetivos estabelecidos para este projeto concentraram-se na implementação das funcionalidades delineadas em Projeto I, conforme documentado no GDD - Game Design Document. A intenção foi criar um jogo cativante e envolvente, oferecendo ao jogador desafios e momentos de diversão. Para atingir esses objetivos, este projeto abordou vários aspetos, incluindo a mecânica do jogo, a interatividade e o design visual. A realização destas funcionalidades foi essencial para desenvolver uma experiência de jogo rica e atrativa.

1.2 Estrutura do relatório

Este documento encontra-se dividido em cinco capítulos, sendo o presente o primeiro. O segundo capítulo apresenta o GDD - Game Design Document, que descreve as características e funcionalidade que devem ser implementadas no jogo.

O terceiro capítulo apresenta todos os elementos gráficos e visuais que foram desenvolvidos para tornarem a qualidade numa experiência de jogo significativamente boa.

O quarto capítulo destina-se a apresentar todas as mecânicas desenvolvidas para o jogo através de fluxogramas que explicam o código desenvolvido para tornar o jogo mais interativo e cativante.

No quinto e último capítulo foram apresentadas as respostas fornecidas por vários utilizadores que tiveram acesso ao jogo final, os quais puderam dar uma opinião sobre os diversos aspetos do jogo, como inimigos, mapa e mecânicas.

2 GDD – Game Design Document

Este capítulo tem como objetivo apresentar o GDD destinado à descrição de todas as características do jogo a desenvolver. Para isso foi criada uma estrutura própria de modo a apresentar as especificações do jogo. O GDD é composto pela descrição do jogo, a sua história, elementos, *gameplay* e por último a estrutura e dificuldade.

2.1 Descrição do jogo

Este jogo conta a história de um ferreiro que tem como objetivo recuperar todas as almas roubadas da sua aldeia natal por um ser de outro mundo. Cabe ao jogador encarnar a pele desse ferreiro e percorrer um mundo de fantasia repleto de criaturas e reinos onde o mal reina.

Este é um jogo 3D do estilo RPG (*Role Playing Game*) com visão isométrica, onde o jogador terá de construir o seu caminho até chegar ao vilão final para o derrotar. O mesmo terá à sua disponibilidade um vasto mundo para explorar com várias áreas e aldeias/reinos que poderá visitar assim que as desbloquear. Será também livre de voltar às zonas pelas quais já passou e assim poder ganhar mais experiência e moedas para poder evoluir.

Para que o jogador possa evoluir as suas armas, armadura, habilidades ou até mesmo a sua vida máxima, o mesmo terá várias aldeias ou reinos com a presença de vendedores que podem vender poções que o ajudaram a evoluir. Para que possa comprar as poções o jogador poderá de ganhar moedas derrotando as várias criaturas espalhadas pelo mundo inteiro.

Ao percorrer o mundo o jogador irá encontrar várias criaturas diferentes, variando as mesmas consoante a área a explorar. Cada área/fase só é desbloqueada quando o jogador derrotou todas as criaturas da anterior. É, no entanto, possível regressar a uma área já conquistada, pois terá as criaturas de volta para as derrotar outra vez e assim ganhar moedas e experiência ou até mesmo experimentar armas novas. De fase para fase as criaturas também têm um nível de dificuldade diferente, ou seja, quanto mais avançada é a fase, maior a dificuldade que o jogador terá de enfrentar.

2.2 História

Tudo começa em uma pequena aldeia situada no alto de uma montanha denominada de Turim, aldeia esta que parecia ter ficado parada no tempo por ser bastante isolada do mundo inteiro. Nesta aldeia vivia um rapaz de 25 anos chamado de Acolon. Acolon cresceu numa família simples e acolhedora que sempre procurava ajudar toda a população da aldeia da melhor maneira.

O pai de Acolon era um ferreiro respeitado por todos, pois as suas técnicas de ferragem eram únicas no reino inteiro. Ele passava a maior parte do seu tempo a produzir novas ferramentas ou a reparar outras, como consequência, Acolon também acabava por passar quase todo o seu dia junto de seu pai na forja onde com o tempo foi crescendo e aprendendo. Este conhecimento permitiu-lhe não apenas aplicar as

técnicas, como também desenvolvê-las. Os anos foram passando e Acolon acabou por ficar com o lugar de seu pai tal como ele fez quando era mais jovem com o seu avô.

A vida na aldeia era calma e serena, os dias passavam lentamente e parecia que nunca se passava nada de interessante, até que um dia o oposto aconteceu. Era um dia calmo e Acolon estava na sua forja a trabalhar quando começou a sentir um pequeno tremor vindo do chão, não sabendo do que se tratava saiu da sua forja e foi ter com as várias pessoas que se encontravam na rua com a mesma questão. Foi então que por baixo dos seus pés começaram a surgir várias criaturas que começaram a atacar a aldeia e todos os seus habitantes, Acolon e todas as pessoas que viviam nesta aldeia tentaram resistir, mas sem qualquer efeito, pois devido aos vários anos e gerações de paz, ninguém na aldeia sabia como lutar ou defender-se, tornando-se assim alvos fáceis. Acolon ao tentar resistir apercebeu-se que não tinha qualquer hipótese ao lutar contra estas criaturas e assim escondeu-se num lugar onde ninguém o encontraria.

Todos na aldeia foram presos e postos no centro, ninguém sabia o que se passava, pois, nenhuma pessoa da aldeia acabou por ser ferida, foi então que vindo do céu em um raio de luz ofuscante, desceu um humano até eles. Este humano era diferente, era muito maior que um humano normal, usava uma coroa e os seus olhos brilhavam mais que tudo, parecia ser uma criatura de um outro mundo, mas com forma e corpo humano.

Quando este chegou ao pé de todos as pessoas que ali estavam, estendeu a sua mão e em um piscar de olhos todas as almas foram sugadas. A maioria caiu desmaiada no chão depois deste acontecimento, os que ficaram de pé obtiveram uma atenção especial e assim foram todos transformados em cavaleiros negros ou magos que iriam servir este ser para o resto da vida.

Ao fim de um tempo e quando já todos tinham ido embora, Acolon correu para ajudar as pessoas que estavam caídas no chão, que ao levantarem-se, notaram que faltavam vários membros da aldeia, o que fez com que começassem a questionar Acolon sobre o que se tinha acontecido. Depois de Acolon ter contado aos membros restantes da aldeia o que tinha acontecido, ele mesmo jurou vingança a todos os membros raptados

Acolon correu para a sua forja, onde construiu uma espada, uma armadura e quando estava a prestes a construir o escudo o chefe da aldeia, que era a pessoa mais velha e sábia de todos, foi ter com ele e deu-lhe uma pequena pedra roxa e disse lhe para colocar a mesma na sua luva esquerda pois iria ser mais útil do que um escudo.

Esta pedra era um tesouro há muito guardado e protegido por esta aldeia, esta pedra tinha caído do céu há muitas gerações atrás quando ainda nem Turim existia. A pequena pedra era capaz de produzir um campo de forças capaz de proteger Acolon de qualquer ataque, mas também de aumentar as capacidades de luta podendo assim produzir ataques mais fortes.

Cabe ao jogador encarnar na pele de Acolon e percorrer o vasto reino em uma jornada que o irá fazer enfrentar vários desafios para assim no fim poder enfrentar este tal ser de outro mundo e conseguir recuperar todas as almas roubadas e salvar todas as pessoas que foram transformadas em escravos.

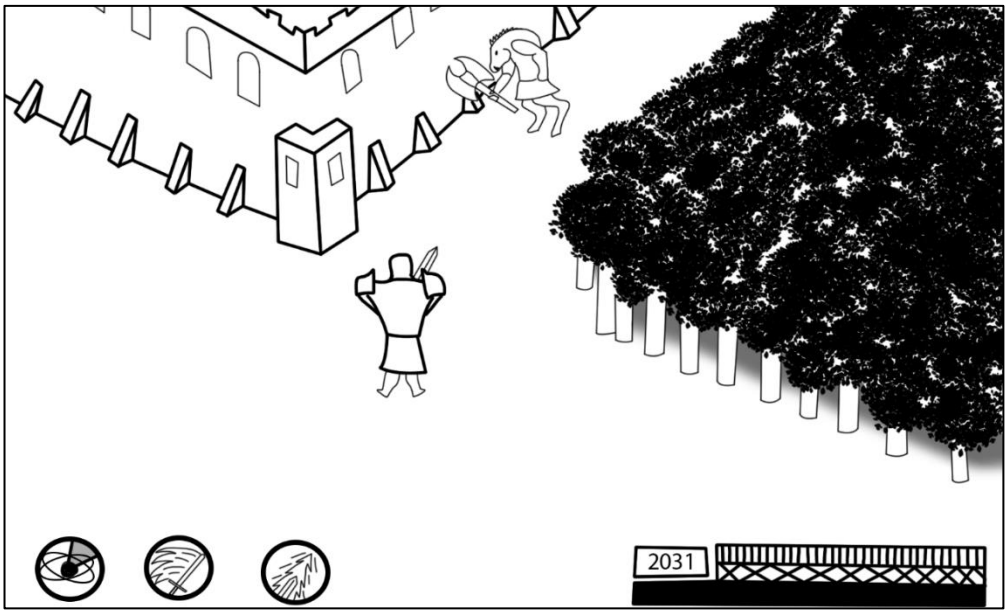


Figura 1 - Representação do jogo

Na Figura 1 pode-se observar uma representação da história do jogo onde o jogador se encontra no controlo da personagem na terceira pessoa num nível de jogo. Este tem à sua disponibilidade as três habilidades especiais no canto inferior esquerdo, contém ainda no seu canto inferior direito as três barras, a de vida representada pela cor preta, a de magia representada pelos losangos e a de escudo representada pelas linhas verticais e ainda um retângulo onde se encontra o número de moedas que tem. Pode se observar também outros elementos como uma floresta existente dentro do nível, ainda muralhas de uma aldeia onde o jogador poderá encontrar as várias lojas do jogo e um dos vários inimigos que o jogador terá que enfrentar.

2.3 Elementos

Nesta secção serão mostrados os elementos que fazem parte do jogo. O personagem principal, todos os inimigos que o jogador irá ter de enfrentar, as armas que poderão ser produzidas ao longo do jogo assim como as suas habilidades especiais são elementos que serão implementados. Além destes elementos será também implementado um sistema de economia que fará com que o jogador possa obter várias poções ou armas para o ajudar na sua jornada.

2.3.1 Personagem principal

Acolon (Figura 2) será o personagem principal do jogo e o qual o jogador poderá controlar. Este terá uma armadura simples construída por ele mesmo e terá uma armadura de malha que servirá para o proteger nas zonas mais expostas. Ele terá uma pedra de cor roxa no seu braço esquerdo que servirá para que o jogador possa invocar um escudo magico para que se possa proteger dos ataques de inimigos.

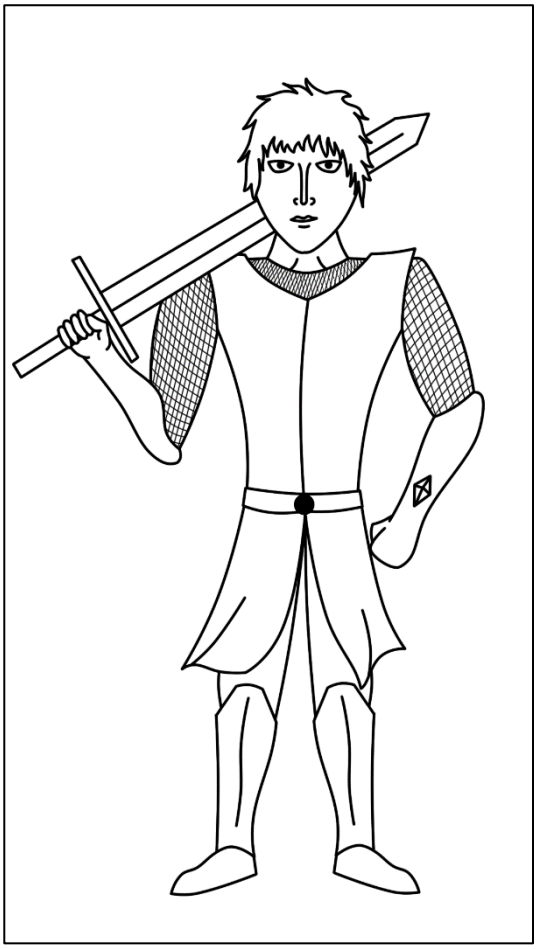


Figura 2 - Personagem Principal

Este escudo terá uma duração/vida limitada, assim sempre que recebe um ataque a vida do mesmo cai e quando chega a zero desaparece por completo. O escudo ao fim de 5 segundos sem sofrer qualquer dano começa a auto regenerar-se até que a barra do mesmo fique completa.

Acolon dependendo da arma que empunha poderá também produzir ataques especiais que o permitirão dar mais dano aos inimigos. Para poder executar essas habilidades ele terá uma barra de magia que poderá encher com poções de magia ou derrotando inimigos, onde cada inimigo derrotado irá contribuir para 10% da barra. Apenas com as armas Espada básica, Machado básico e as Facas básicas não poderá executar qualquer ataque especial.

2.3.2 Armas

Aqui serão mostradas todas as armas que o jogador poderá construir ao longo do jogo. Estas armas podem variar entre básicas ou mais avançadas que permitirão executar diferentes tipos de ataques especiais ou até mesmo armas com poderes elementais como fogo ou gelo.

Espadas

- Espada básica

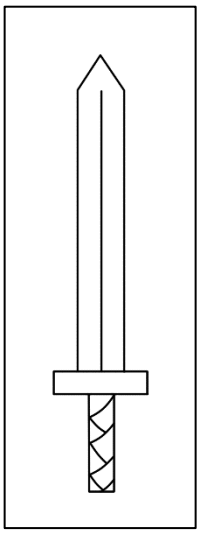


Figura 3 - Espada básica

Esta é uma das armas com que o jogador pode iniciar a sua jornada para recuperar todas as almas roubadas. Esta espada (Figura 3) é feita de ferro simples e por ser uma arma básica ela apenas permite fazer ataques normais, isto é, o jogador não poderá fazer qualquer tipo de ataque especial com esta arma limitando-se a ataques básicos. A velocidade de ataque desta arma é média e assim entre ataques o jogador apenas terá de esperar cerca de meio segundo para poder voltar a atacar.

Esta espada terá um comprimento total de 80 centímetros, 60 centímetros para a lâmina, 2 centímetros para a guarda e ficando apenas com 15 centímetros para o cabo. Devido ao tamanho desta espada o jogador apenas poderá atacar inimigos que estejam próximo dele.

Variantes:

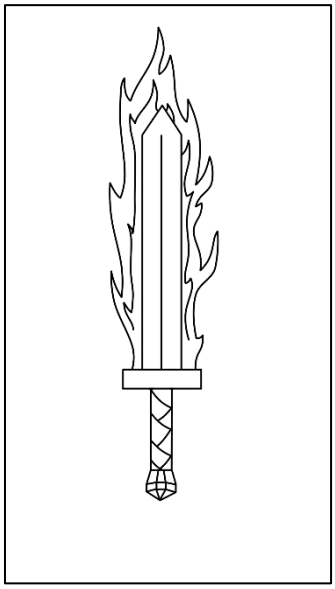


Figura 4 – Espada Básica de Fogo

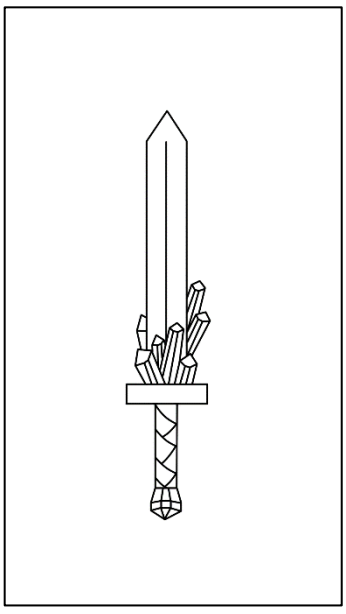


Figura 5 - Espada Básica de Gelo

Fogo: esta arma baseia-se na espada básica (Figura 3), mas com a diferença de que esta contém um cristal de fogo que pode ser obtido nas terras de fogo, o que faz com que a sua lâmina esteja constantemente rodeada por chamas (Figura 4). Com esta arma o jogador terá a mesma forma de ataque que a espada básica normal, mas terá a chance de 30% de colocar os inimigos em chamas produzindo assim dano continuo durante 5 segundos.

Gelo: assim como a espada de fogo (Figura 4) esta espada terá um cristal, mas desta vez é um cristal de gelo que pode ser adquirido nas terras de gelo, assim a sua lâmina fica rodeada de cristais de gelo (Figura 5). Com esta espada o jogador terá 30% de chances de congelar os inimigos que ficaram paralisados durante 3 segundos como também terá todos os ataques iguais à espada básica.

- Espada avançada

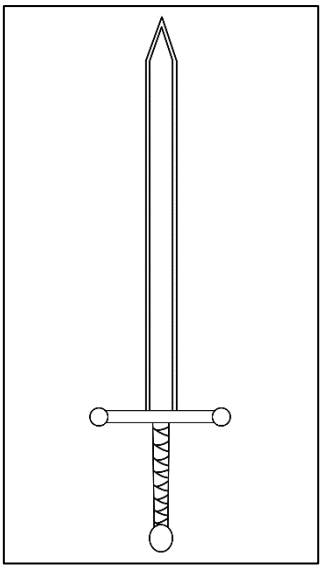


Figura 6 - Espada Avançada

A espada Avançada (Figura 6) pode ser adquirida pelo jogador quando o mesmo conseguir juntar materiais suficientes para construir a mesma ou moedas para a poder comprar. Por ser uma espada avançada esta terá a possibilidade de ser evoluída com o tempo, ou seja, à medida que o jogador vai ganhando mais materiais pode aumentar o dano da espada numa bigorna forjando-a com mais ferro.

Com cerca de 1 metro e 22 centímetros de comprimento esta espada tem a necessidade de ser manejada com as duas mãos, tendo cerca de 90 centímetros de lâmina, 2 centímetros de guarda e ficando com 30 centímetros para o cabo. Devido ao grande comprimento desta arma em relação à espada básica, esta terá a sua velocidade de ataque bastante reduzido e assim o jogador terá de esperar cerca de 1 segundo para poder voltar a atacar, mas em contrapartida terá a vantagem no seu alcance que poderá atingir inimigos que estejam mais afastados do jogador.

Variantes:

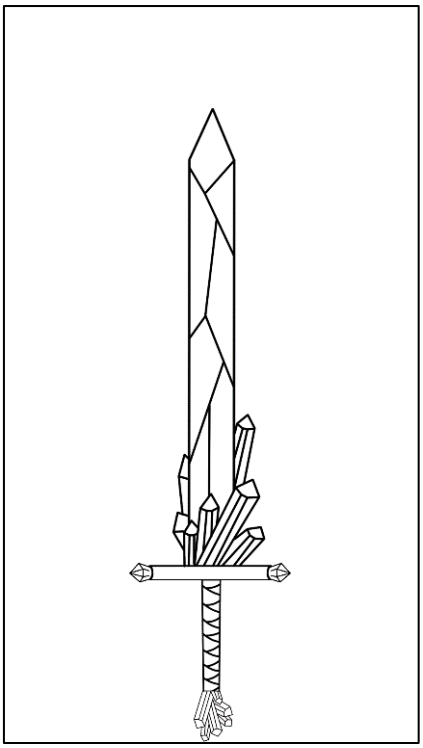


Figura 7 - Espada Avançada de Gelo

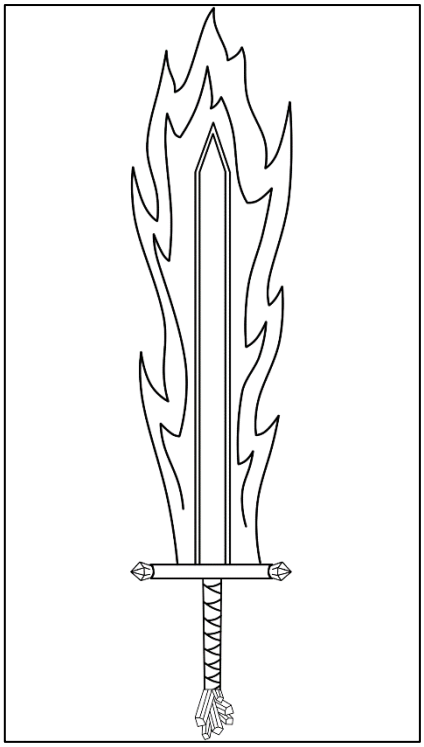


Figura 8 - Espada Avançada de Fogo

Fogo: esta espada baseia-se na espada avançada (Figura 6), mas com a diferença que esta será constituída por três cristais de fogo, fazendo assim com que a sua lâmina seja envolvida em chamas (Figura 8). Contendo cerca de 50 por cento de chance de colocar os inimigos em chamas e assim dando dano continuo durante 5 segundos.

Gelo: assim como a espada de fogo (Figura 8) a espada de gelo irá conter três cristais, mas estes serão de gelo que faram com que a lâmina seja envolvida em cristais de gelo (Figura 7). Quando usada, esta tem 50 por cento de chance de congelar os inimigos e assim paralisando-os durante 5 segundos.

Ataques especiais:

Ataque rotativo: Neste ataque o jogador vai juntar as suas forças e rodar a espada como se fosse um peão e assim podendo dar vários golpes em vários inimigos ao mesmo tempo. Terá um tempo de 4 segundos de duração e o jogador apenas poderá invocar este ataque ao fim de 1 minuto e 30 segundos. Se a espada usada for de a de fogo, esta colocará os inimigos que acertar em chamas durante 5 segundos, se for a espada usada for a de gelo a mesma irá congelar todos os inimigos em que acertar também durante 5 segundos.

Investida: ao selecionar este ataque, o jogador irá dar uma investida como o nome indica, ou seja, ele irá apontar a espada na direção em que se encontra e movimentar-se nessa mesma direção dando dano extra nos inimigos em que este acertar. Este ataque pode servir também para poder fugir eventualmente de um ataque ou inimigo

que o poderá matar. Quando usado, o jogador terá de esperar 2 minutos para poder voltar a invocar esta habilidade. Se usada a espada de fogo com este ataque irá colocar os inimigos em chamas durante 8 segundos, se usada a espada de gelo esta irá congelar os inimigos durante 8 segundos também.

Ataque final: aqui o jogador irá usar a sua espada para aniquilar um inimigo. Ao invocar esta habilidade irá fazer com que Acolon dê um salto para o ar e com a sua espada na direção do inimigo irá fazer com que acabe com a vida do mesmo. Para o jogador voltar a poder invocar este ataque terá de ter derrotado um total de 10 inimigos. Quando usada a espada de fogo com esta habilidade todos os inimigos à volta do jogador iram ser colocados em fogo durante 15 segundos ou então se for usada a espada de gelo todos os inimigos iram ser congelados durante também 15 segundos.

Machados

- Machado básico

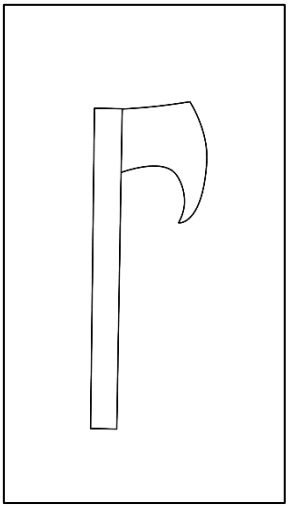


Figura 9 - Machado Básico

O machado básico (Figura 9) irá ser uma das armas que o jogador poderá construir quando iniciar o jogo. Devido ao machado ser uma arma mais pesada o mesmo tem uma velocidade de ataque reduzido o que significa que entre ataques o jogador terá de esperar cerca de 1 segundo, mas o seu dano de ataque será superior à espada básica.

Com cerca de 70 centímetros de comprimento, esta arma não poderá ser evoluída com o tempo por ser uma arma básica, assim o jogador também não terá qualquer possibilidade de executar habilidades especiais com a mesma ficando assim apenas com ataques básicos.

Variantes:

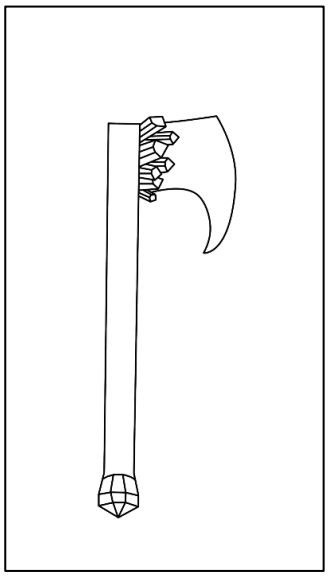


Figura 10 - Machado Básico de Gelo

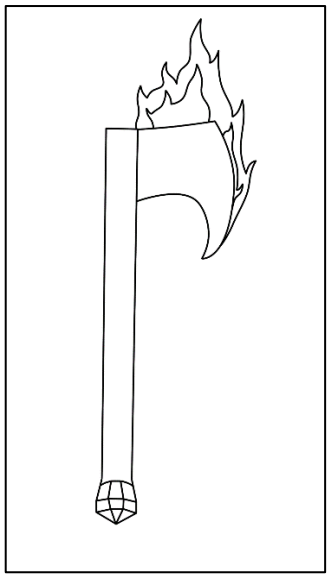


Figura 11 - Machado Básico de Fogo

Fogo: este machado tal como o machado básico, o machado de fogo (Figura 10) não poderá ser evoluído com o tempo nem contém qualquer habilidade especial. Na sua base contém um cristal de fogo que fará com que a sua lâmina seja revestida de chamas, fazendo assim com que tenha cerca de 30 por centro de chances de colocar os inimigos em que acerta em chamas e assim fazendo com que sofram dando continuo.

Gelo: este machado (Figura 11) tem a sua lâmina coberta de cristais de gelo devido ao facto de ter na sua base um cristal de gelo tal como o machado de fogo. Ao usar esta arma o jogador terá 30 por cento de chances de congelar os inimigos que ataca e assim fazendo com que fiquem paralisados durante 3 segundos, mas por ser também uma arma básica não é possível executar qualquer tipo de ataque especial.

- Machado avançado

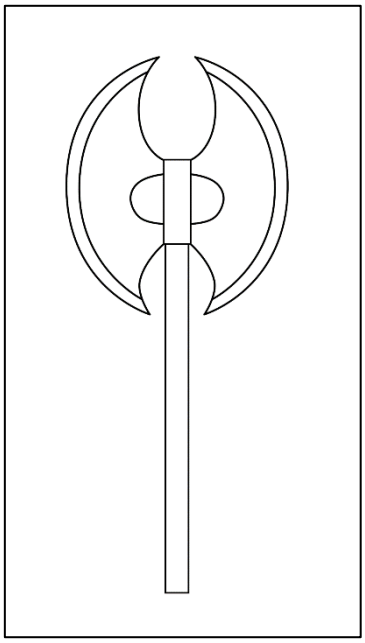


Figura 12 - Machado Avançado

O machado avançado (Figura 12) pode ser adquirido pelo jogador quando este conseguir juntar os materiais suficientes para a construção do mesmo ou quando juntar moedas suficientes para o comprar de uma loja num reino. Ao contrário do machado básico este machado pode ser ter o seu dando aumentado ao longo do tempo, o jogador apenas tem de ir juntando materiais e junto a uma bigorna pode com esses materiais aumentar o dano do machado.

Tendo cerca de 1 metro e 50 centímetros, este machado torna se a maior arma que o jogador pode adquirir, mas devido ao seu tamanho e peso torna-o uma arma bastante lenta de se usar fazendo assim com que o jogador tenha de esperar cerca de 1 segundo e meio entre ataques. Em contrapartida este machado produz ataques que dão bastante dano quando comprado às outras armas.

Variantes:

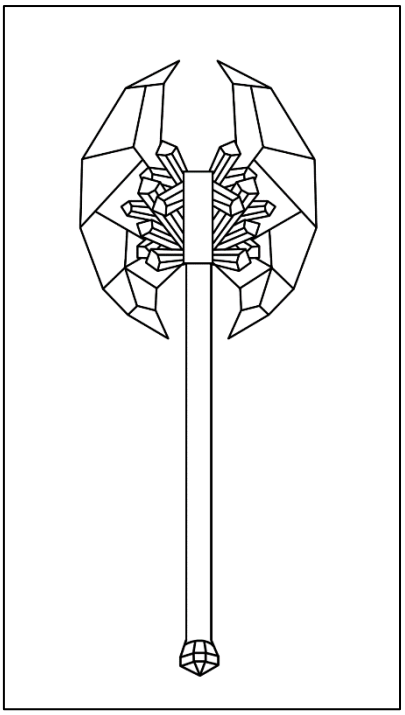


Figura 13 - Machado Avançado de Gelo

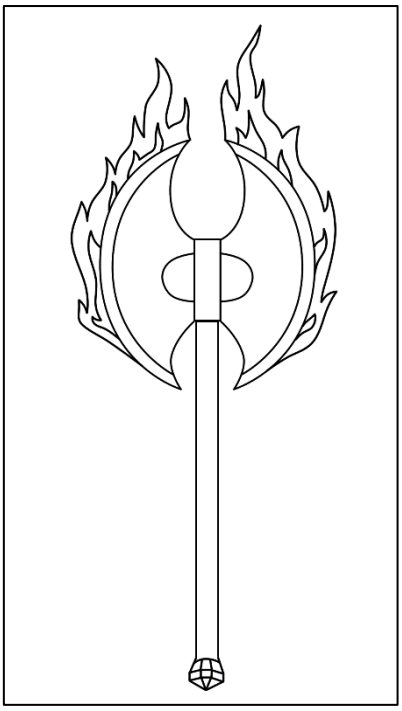


Figura 14 - Machado Avançado de Fogo

Fogo: este machado irá conter na sua constituição cristais de fogo que faram com que as suas laminas sejam revestidas em chamas (Figura 14). Por conter cristais de fogo e assim como o machado básico de fogo, este pode colocar os inimigos em chamas durante 5 segundos tendo cerca de 50 por cento de chances de acontecer.

Gelo: assim como o machado de fogo (Figura 14) o machado de gelo ira ser constituído por cristais de gelo que faram com que as suas laminas sejam revestidas de cristais de gelo (Figura 13). Este machado terá a chance de 50 por cento de congelar os seus inimigos durante 5 segundos.

Ataques especiais

Ataque rotativo: ao invocar este ataque o jogador irá fazer com que o seu personagem Acolon rode o machado e assim dando dano a todos os inimigos em que tiver acertado e fazendo com que esses inimigos sejam empurrados para trás afastando-os. Ao fim de ser usado o jogador terá de esperar cerca de 1 minuto e 30 segundos até poder voltar a invocar esta habilidade. Se o jogador estiver com o machado de fogo equipado, este irá colocar também os seus imigos em chamas durante 5 segundos. Se o machado equipado for o de gelo este irá congelar os inimigos durante 5 segundos.

Terramoto: neste ataque especial Acolon reúne as suas fornas num ataque capaz de abater um inimigo com um único golpe tendo assim uma chance de 30 por cento de chances de acontecer e ainda todos os inimigos que apanhar no seu raio de ataque ficaram atordoados fazendo com que fiquem sem atacar durante 5 segundos. Depois

de usar este ataque o jogador terá de esperar certa de 2 minutos até poder voltar a usar este ataque. Se o machado que estiver equipado for o de fogo este terá 100 por cento de chances de colocar os inimigos em chamas e assim alem de ficar atordoados iram sofrer dano continuo do fogo. Se ao invés do machado de fogo for o machado de gelo que estiver equipado os inimigos apanhados no raio de ataque ficaram congelados durante 10 segundos.

Ataque final: ao selecionar este ataque o jogador irá aniquilar um inimigo por completo. Quando este ataque é invocado a personagem Acolon irá saltar com o seu machado e reunindo todas as forças ele irá bater com o machado no inimigo e assim destruindo o inimigo por completo. Se o jogador tiver o machado de fogo equipado, este irá fazer com que todos os inimigos à volta fiquem em chamas durante 10 segundos. Se o machado equipado for o de gelo todos os inimigos à volta serão congelados durante 10 segundos também.

Facas

- Facas básicas

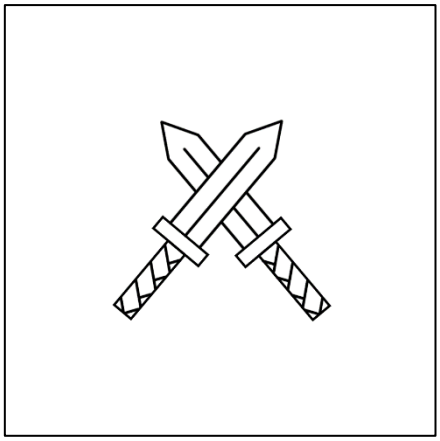


Figura 15 - Facas básicas

Dentro das armas que o jogador pode construir no início do jogo, as facas básicas (Figura 15) são uma delas. As facas básicas medem cerca de 40 centímetros, 13 centímetros de cabo, 2 de guarda e 25 de lâmina, o que torna estas armas relativamente pequenas. Devido ao seu tamanho as facas básicas o jogador irá notar que a velocidade de ataque das mesmas é superior às outras armas e assim podendo atacar a cada meio segundo, mas em contrapartida o dano que as mesmas dão é o mais baixo de todas as armas. Por serem armas básicas o jogador não poderá evoluir as mesmas com o tempo e também não terá acesso a ataques especiais.

Variantes

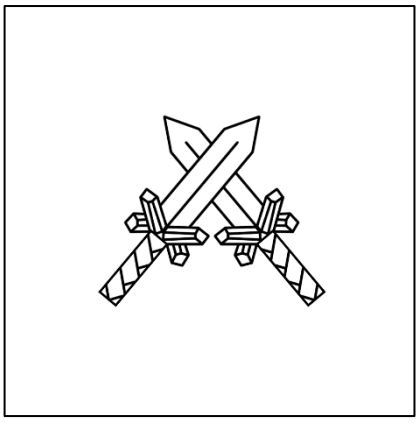


Figura 16 - Facas Básicas de Gelo

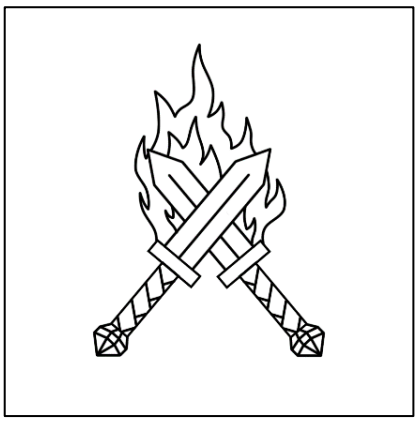


Figura 17 - Facas Básicas de Fogo

Fogo: contendo um cristal de fogo na sua base estas facas terão a sua lâmina revestida por chamas (Figura 17) e assim fazendo com que os ataques acertados em inimigos terão cerca de 30 por cento de chances de os colocar em chamas durante 5 segundos e assim dando dano continuo.

Gelo: assim como a faca básica de fogo a de gelo terá também na sua base um cristal de gelo que fará com que a sua lâmina esteja rodeada por cristais de gelo (Figura 16) o que fará com que o jogador tenha cerca de 30 por cento de chances de congelar os inimigos atacados e assim paralisando-os durante 5 segundos.

- Facas avançadas

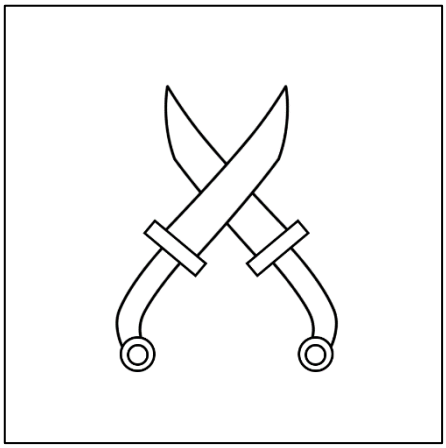


Figura 18 - Facas Avançadas

As facas avançadas (Figura 18) serão possíveis de construir quando o jogador conseguir adquirir os materiais suficientes para construir as mesmas. Estas facas terão o mesmo tamanho e velocidade de ataque que as facas básicas, mas devido a terem uma lâmina mais fina e o cabo sendo mais adaptado ao combate estas laminas serão capazes de produzir mais dano aos inimigos.

Por serem armas avanças estas facas podem ser evoluídas pelo jogador ao longo do tempo e assim podendo aumentar o seu dano. Estas facas terão também a particularidade de poderem executar ataques especiais quando equipadas.

Variantes

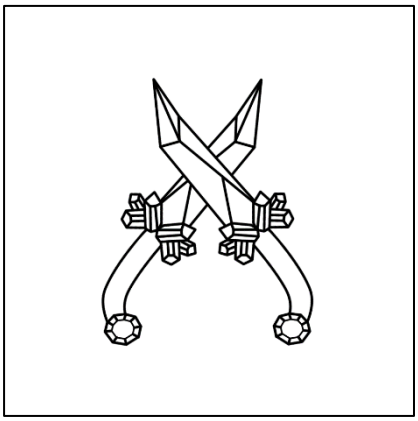


Figura 19 - Facas Avançadas de Gelo

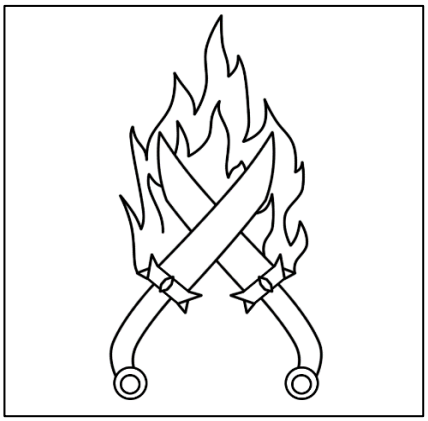


Figura 20 - Facas Avançadas de Fogo

Fogo: estas facas serão constituídas por cristais de gelo fazendo assim com que as suas laminas sejam rodeadas por chamas (Figura 20). Devido às chamas estas armas terão cerca de 50 por cento de chances de colocar os inimigos em chamas durante 8 segundos.

Gelo: por estas armas serem de gelo irão ter na sua constituição cristais de gelo que farão com que a sua lâmina seja rodeada por gelo e cristais (Figura 19). Estas laminas de gelo serão capazes de congelar os inimigos durante 8 segundos, mas tendo apenas 50 por cento de chances de isso acontecer.

Ataques especiais

Giratório: quando o jogador selecionar este ataque fará com que o personagem Acolon gire com as suas facas em direção ao inimigo mais próximo dando vários ataques em simultâneo e dando dano também em inimigos que se aproximem. Ao usar este ataque o jogador terá de esperar cerca de 1 minuto e 30 segundos até que a possa usar outra vez. Se o jogador tiver equipadas as facas de fogo este terá cerca de 80 por cento de chances de colocar os inimigos atingidos em chamas durante 10 segundos. Se as facas equipadas forem as de gelo o jogador terá cerca de 80 por cento de chances de congelar os inimigos durante 10 segundos.

Flash: ao invocar este ataque o jogador fará com que o personagem se mova super-rápido quase como um teletransporte para os 4 inimigos mais próximos e em sequência onde mesmo dará dano extra a esses mesmos inimigos. Depois de ser usado o jogador terá de esperar cerca de 2 minutos até poder voltar a usar este ataque. Se as facas equipadas forem as facas avançadas de fogo, quando o jogador usar este ataque fará também com que os inimigos atacados fiquem em chamas durante 10 segundos dando assim dano continuo. Se ao invés das facas de fogo forem as de gelo os inimigos atacados ficaram congelados durante 10 segundos.

Ataque final: neste ataque o jogador irá eliminar um inimigo num só golpe fazendo com que Acolon corra na direção de um inimigo e acabe com o mesmo. Este ataque depois de ser usado só estará disponível outra vez depois do jogador ter conseguido derrotar cerca de 10 inimigos. Se as facas equipadas forem as de fogo o jogador terá

100 por cento de chances de nos próximos 15 segundos seguintes ao executar este ataque de colocar os inimigos em chamas. Se as armas equipadas forem as de gelo, tal como as facas de fogo o jogador terá 100 por cento de chances de congelar os inimigos nos 15 segundos seguintes ao ataque.

Dano de cada arma e custo de habilidades.

Tabela 1 - Tabela de dano e custo

Armas	Dano					
	Base	Especial 1	Especial 2	Especial 3	Fogo	Gelo
Espada Básica	10	----------	----------	----------	+5	+3
Espada Avançada	20	40	55	100%	+8	+6
Machado Básico	20	----------	----------	----------	+5	+3
Machado Avançado	35	50	70	100%	+8	+6
Facas Básicas	5	----------	----------	----------	+3	+2
Facas Avançadas	10	20	30	100%	+5	+4
Custo Mágico	---------	10	20	40	---------	-------

Na Tabela 1 é representado o dano que cada arma pode dar a um inimigo nos seus ataques básicos e nos seus ataques especiais. Sendo que as armas básicas não contêm qualquer tipo de ataque especial as mesmas não contêm qualquer tipo de dano por via deste meio. No caso das armas avançadas é representado o dano aplicado a um inimigo quando estas habilidades 1 ou 2 são invocadas, a habilidade 3 causa 100% de dano da vida do inimigo atingido, eliminando-o assim por completo.

Devido às armas conterem as variantes de gelo e fogo na tabela está também representado o dano que essas variantes acrescentam às armas. No caso das armas de fogo o dano é dado a cada segundo no tempo em que os inimigos estão em chamas, no caso das armas de gelo o dano extra é todas as vezes que o jogador ataca o inimigo que esteja congelado.

2.3.3 Inimigos

Nesta secção serão mostrados todos os inimigos que o jogador terá de enfrentar durante o jogo, sejam eles mais pequenos ou grandes, criaturas ou humanos. Aqui será também descrito o aspeto de cada um e todos os ataques que os mesmos podem executar assim como as suas fraquezas.

Criaturas

- Javeling

O Javeling (Figura 21) é uma criatura de grande porte e com bastante vida, esta criatura é metade humano metade javali o que faz com que o seu corpo seja coberto por uma pele bastante grossa e por pelos espessos, fazendo assim com que seja difícil penetrar com qualquer arma. Contendo duas pernas bastante fortes, o Javeling é capaz de caminhar bastante rápido apesar do seu tamanho e assim tornando a fuga desta criatura bastante difícil.

Contendo um machado em sua pose, o Javeling é capaz de produzir ataques bastante fortes e eficazes. A maioria dos ataques são apenas feitos baloiçando o seu machado, porem 1 em cada 6 ataques é um ataque pesado capaz de atordoar o jogador durante 5 segundos e assim impossibilitando-o de atacar, mas se o ataque falha existe uma probabilidade de 50% de o machado ficar preso no chão e durante também 5 segundos o Javeling não se consegue mover até retirar o machado do chão.

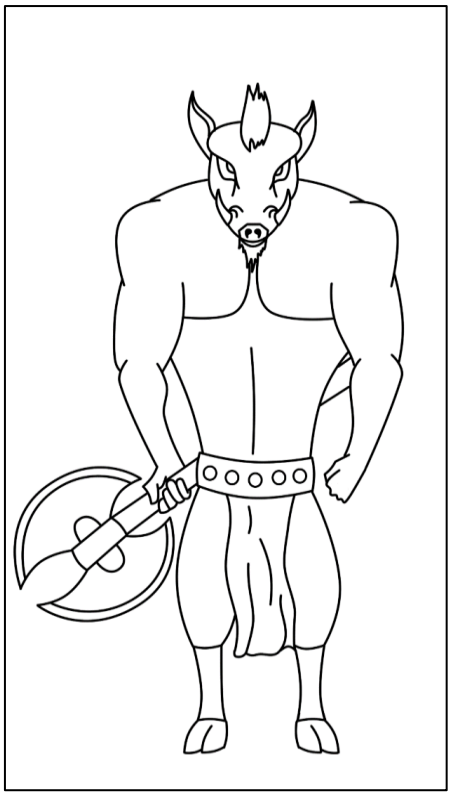


Figura 21 - Javeling

- Esqueleto

O esqueleto (Figura 22) é uma criatura relativamente pequena e pelo que o nome por si já indica é apenas um esqueleto, este sendo apenas um esqueleto anda com uma bomba que transporta sempre consigo, não tem qualquer mentalidade e então vagueia pelo mapa sempre pronto para executar a sua única instrução, quando o jogador se atravessar no campo de visão do Esqueleto o mesmo irá correr atrás do jogador até que o consiga apanhar, quando o alcança ele acende a sua bomba que demora cerca de 3 segundos até rebentar, tendo pouca vida o jogador poderá destrui-lo antes que a bomba rebente.

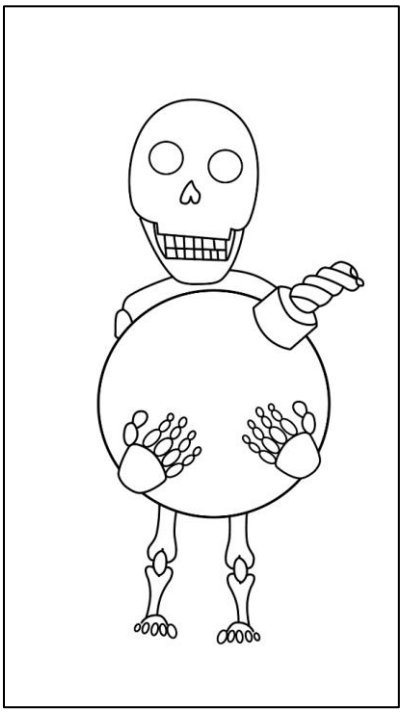
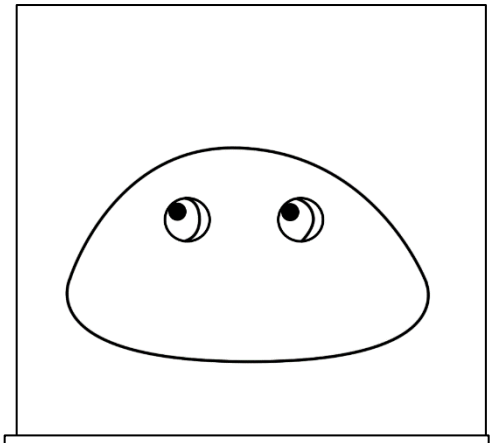


Figura 22 - Esqueleto

- Globs

Os Globs (Figura 23) são uma espécie de gelatina vermelha, verde ou azul com olhos. Este inimigo fica espalhado pelo mapa à espera que o jogador passe por ele para poder atacar. Quando este avista o jogador começa a andar atras dele aos saltos e quando atinge uma certa distancia do jogador este carrega as suas energias e dá um salto para cima do jogador causando dano ao mesmo.



- Crabs

Esta criatura é meia caranguejo meia humana tal como o Javeling. O Crab (Figura 24) tem o corpo revestido por uma carapaça como a de um caranguejo o que o torna imune a qualquer ataque, porem esta armadura torna-o mais pesado e lento. No lugar das suas mãos o Crab tem garras que utiliza para atacar o jogador, o mesmo dá uma espécie de soco com elas e assim provocando dano ao jogador.

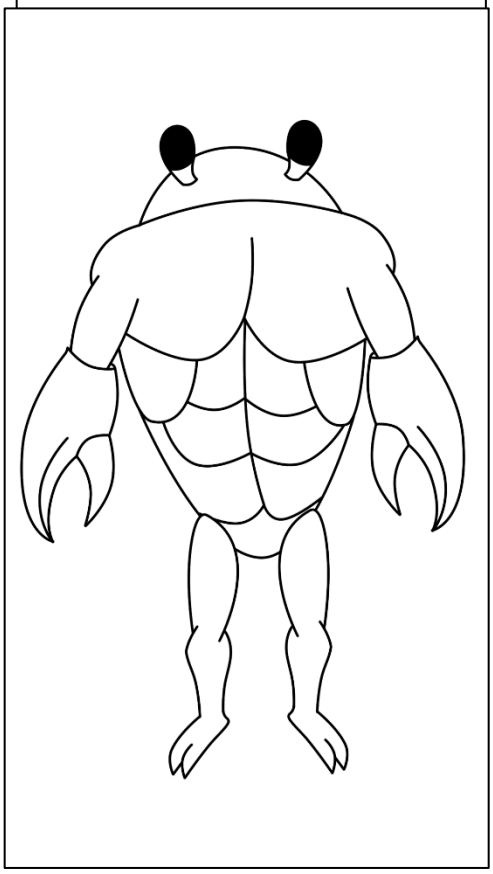


Figura 24 - Crab

Devido ao seu peso o Crab cansa-se bastante rápido e apenas conseguindo dar cerca de 5 ataques de cada vez, assim que faz os 5 ataques ele para descansar, fazendo assim com que fique vulnerável durante cerca de 5 segundos e abrindo a única oportunidade para o jogador poder atacar.

Esta criatura fica a passear em uma certa área do mapa ficando de guarda à espera que o jogador se atravesse na sua área de visão e assim começando a persegui-lo. Por ser uma criatura lenta o mesmo anda muito mais devagar que o jogador e quando atinge uma certa distancia o mesmo volta para a área que estava a patrulhar anteriormente.

Humanos

- Cavaleiros negros

Os cavaleiros negros (Figura 25) são humanos possuídos pelo Baelzor (vilão da história) que protegem as várias aldeias espalhadas pelo reino inteiro, estes nunca andam sozinhos, logo atacam sempre em um grupo mínimo de dois cavaleiros e sem terem número máximo. O cavaleiro possui uma espada que usa para atacar e um escudo que usa para se defender. São o tipo de tropa inimiga mais fraca quando sozinhos, mas em grupo podem produzir bastantes problemas ao jogador.

Este inimigo fica estático e em postos fixos nas entradas das aldeias ou em espaços amplos como se ficassem de guarda. Os cavaleiros negros apenas atacam quando o jogador se atravessa no campo visão dos mesmos na qual perseguem o jogador enquanto não se afastam demasiado do seu posto inicial, quando esta distância é ultrapassada estes voltam à posição inicial.

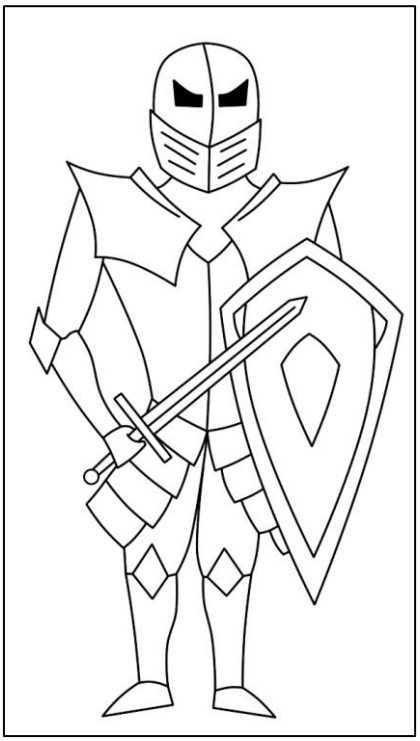


Figura 25 - Cavaleiro Negro

- Magos

Estes são seres humanos raptados pelo vilão do jogo e transformados em servos completamente leais. Os magos (Figura 26) possuem poderes mágicos que foram atribuídos pelo Baelzor (vilão da história).

Os magos ficam espalhados pelo mapa escondidos em áreas de mais difícil acesso e assim poderem atirar as suas bolas de energia que ficam a perseguir o jogador durante cerca de 5 segundos. Quando o jogador se aproxima demasiado de um mago, este para de mandar bolas de energia e com o seu bastão magico começa a produzir uma onde de energia que produz dano ao jogador e o afasta para mais longe.

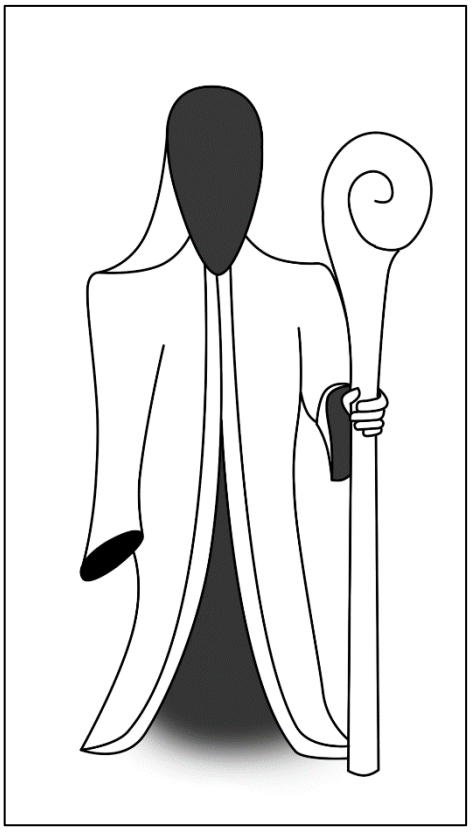


Figura 26 - Mago

- Baelzor

Baelzor ou também conhecido por Sugador de almas (Figura 27) é o vilão da história e é o inimigo mais forte que o jogador terá de enfrentar, o Baelzor permanece sempre dentro do seu castelo sentado no seu trono onde controla o reino inteiro. O Baelzor tem aparência humana, mas com tamanho bastante superior a um humano normal.

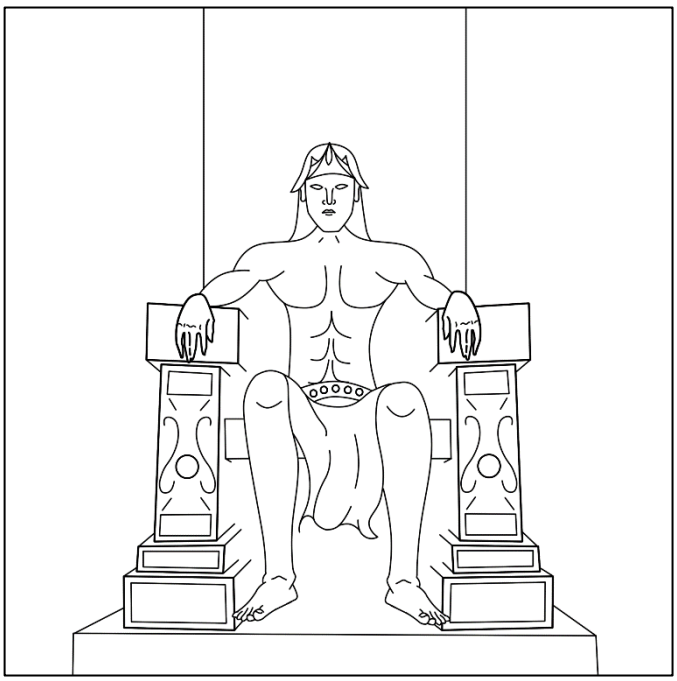


Figura 27 - Baelzor

O Baelzor tem duas fazes de vida, a primeira fase ele fica sentado no seu trono e apenas lançando ataques mágicos na direção do jogador, estes ataques variam entre espinhos que surgem na área em que o jogador se encontra, socos mágicos vindos do céu, raios que percorrem a área inteira e um sopro capaz de afastar o jogador. Na segunda fase de vida o Baelzor levanta-se do seu trono e passa a ataques físicos e mágicos, começa a andar a traz do jogador para conseguir dar

murros e pontapés, consegue também invocar socos vindos do céu e é capaz de produzir uma explosão de energia capaz de roubar vida ao jogador.

2.3.4 Sistema de vidas

O jogador terá uma barra de vida que indicará o estado de saúde do personagem. O jogador terá um máximo de 200 de vida como mostrado na Tabela 2. A vida do jogador diminui progressivamente quando este leva dano de inimigos, o total de dano recebido vai depender da resistência da armadura do jogador, que quando melhor for menos dano recebe de um ataque. Se o jogador não receber qualquer dano num espaço de 10 segundos a mesma começa a regenerar-se, aumentando 1 valor de vida a cada 3 segundos. Quando a barra de vida chega a zero o jogador é derrotado e este é levado para o início do último nível que o jogador tenha estado.

Tabela 2 - Tabela de vida e dano dos personagens

Inimigos	Vida	Dano	Vida do Jogador
Javeling	100	40	200
Esqueleto	20	50	
Glob	50	30	
Crab	100	40	
Cavaleiro Negro	70	30	
Mago	30	20	
Baelzor	1000 x 2	entre 50 e 200	

A vida dos inimigos vai aumentar por nível que o jogador desbloqueia, ou seja, a vida de um inimigo aumenta 10% em relação ao nível anterior.

2.3.5 Lojas

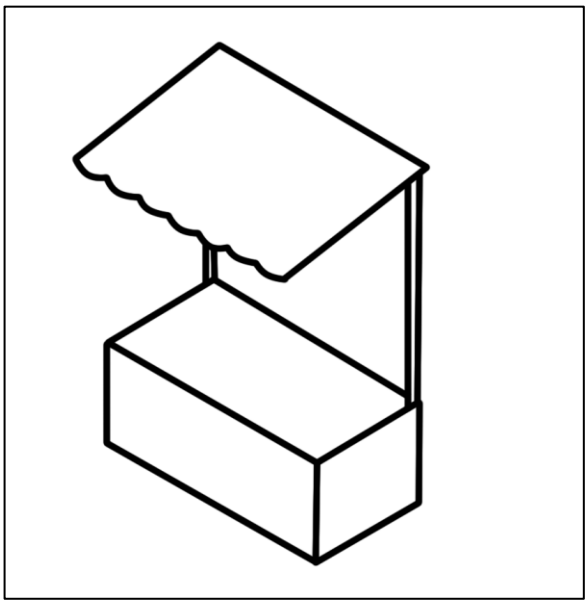


Figura 28 - Representação da loja

As lojas são lugares presentes em aldeias espalhadas pelo reino que podem ser identificadas no mapa de jogo, estas lojas são locais na qual o jogador pode interagir para poder comprar várias coisas como armas, poções e materiais que poderá usar para evoluir. Na Figura 28 está representada a loja que irá estar presente no jogo.

- **Armas**

A loja de armas é uma loja com teto de cor avermelhada e com duas espadas cruzadas em cima para que o jogador a possa identificar a mesma, nesta loja o jogador poderá comprar armas já feitas e prontas a ser usadas.

- **Poções**

Esta loja terá uma cor verde na cortina que faz de teto, e terá um frasco por cima para a identificar como loja de poções. Nesta loja o jogador poderá obter poções para aumentar recuperar vida, recuperar escudo, recuperar magia, ou poções que aumentem as suas estatísticas temporariamente.

- **Materiais**

A loja de materiais vai ter uma cor laranja e terá um pedaço de ferro e um de couro por cima da mesma para que o jogador a possa identificar. Esta loja irá vender materiais que o jogador necessitará de ter para construir ou evoluir as suas armas ou armadura.

2.3.6 Poções

As poções serviram para que o jogador possa aumentar ou regenerar as suas estatísticas, este pode aceder ao seu inventário e clicar na poção que deseja tomar e assim aplica os efeitos da poção. As poções podem ser tomadas em qualquer altura do jogo, seja numa vila sem problemas ou a meio de um combate, mas apenas podem ser adquiridas quando compradas numa loja de poções situadas em vilas.

- **Vida**

A poção de vida irá ser um frasco de vidro com um líquido vermelho identificando que será a poção que regenera vida do jogador. Esta poção irá regenerar 30% da vida total do jogador.

Escudo

A poção de escudo irá ser identificada por um líquido de cor roxa dentro de um frasco de vidro. Esta poção irá regenerar 30% do escudo total do jogador.

- **Magia**

A poção de magia irá ser um frasco com um líquido de cor azul que servira para o jogador identificar que se trata de uma poção para regenerar magia. Esta irá regenerar 30% da magia total do jogador.

- **Upgrades**

As poções de upgrades servirão para melhorar as estatísticas do jogador para que poça ter vantagem durante uma batalha, mas ao contrário das poções anteriores que o efeito é permanente, estas apenas duram um certo tempo até que percam qualquer tipo de efeito.

- **Dano**

A poção de dano terá um líquido de cor laranja para a identificar e adicionara 40% do dano total do jogador, este efeito terá uma duração de 3 minutos até o efeito acabar.

- **Escudo**

A poção de escudo terá um líquido de cor roxa para a identificar e adicionara 50% do escudo total do jogador, o efeito desta poção dura 3 minutos.

- **Velocidade**

Esta poção terá um líquido de cor cinza para que o jogador a poça identificar, esta poção fará com que a velocidade do jogador aumente em 30% durante 3 minutos.

2.3.7 Mapa

- **Mapa de nível**

O jogador durante o seu jogo terá à sua disponibilidade o mapa do nível que se encontra, este pode ser aberto em qualquer altura do jogo, dando ao jogador a informação de onde se encontram as lojas do nível em que se encontra, informação dos inimigos, a entrada e saída do nível e os seus objetivos.

- **Mapa do mundo**

o mapa mundo mostrará ao jogador a localização de todos os níveis que o jogador terá acesso e também mostrará o que já se encontra desbloqueado para o jogador visitar. Este terá a informação de quais os níveis que já visitou ou não e ainda identificará os níveis que são de ação/combate e os que são apenas de vilas de comercio.

2.3.8 Bigorna

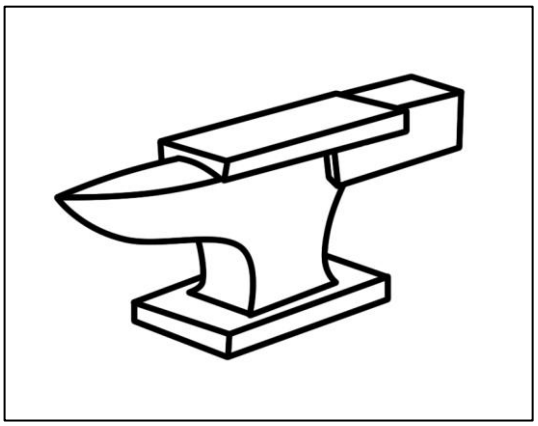


Figura 29 – Bigorna

A bigorna é o elemento pela qual o jogador poderá evoluir ou construir as armas do jogo. Na Figura 29 está representada a bigorna pela qual o jogador poderá interagir e aceder ao menu da mesma. A bigorna estará presente também nas aldeias como as lojas do jogo e por vezes também poderá ser possível encontrá-la escondida em alguns níveis.

2.3.9 Portas e portões

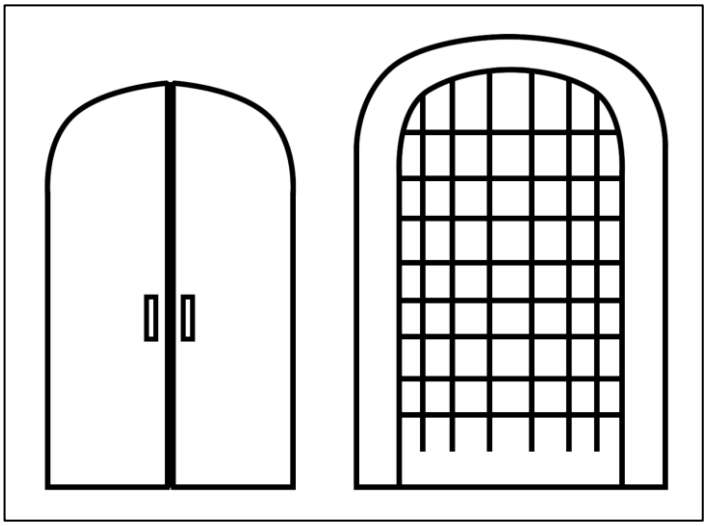


Figura 30 - Representação das portas e portões

As portas ou portões são elementos onde jogador poderá interagir para conseguir desbloquear níveis ou até mesmo outras zonas dentro do nível. Para conseguir abrir a porta ou o portão o jogador na maioria das vezes terá de concluir alguma ação dentro do jogo para que assim a porta/portão fique disponível para abrir. Na Figura 30 é possível observar a representação de uma porta e um portão que estarão presentes dentro do jogo.

2.4 Gameplay

O jogo é iniciado quando o jogador é levado para a primeira fase/nível e o objetivo é percorrer o reino e evoluir com o tempo até chegar à fase final e consiga derrotar o Baelzor e assim libertar todas as almas roubadas do reino.

2.4.1 Perspetiva do jogador

O jogador irá conseguir controlar o seu cavaleiro através das teclas "WASD" do seu teclado e assim podendo se mover livremente pelo mapa em que se encontra, se o jogador quiser se mover para outra área de jogo anterior terá que ir ao menu de pausa e aceder ao mapa do reino e clicar no nível para onde se quer transportar, mas se o jogador quiser ir para um nível seguinte terá que completar o nível que se aproxime mais do fim do mapa e quando este esteja completo bastava avançar para a saída do nível e assim é automaticamente transportado para o nível seguinte ficando assim disponível para o jogador.

O jogador terá acesso a várias armas ao longo do jogo e para este poder alterar entre elas terá de apertar a tecla "I" do teclado para poder aceder ao seu inventário e assim poder escolher a arma que desenha usar. Para o jogador poder desbloquear novas armas este terá de reunir materiais necessários para construir a arma que deseja junto a uma bigorna ou deslocar-se a uma loja de armas para poder comprar uma que esteja disponível.

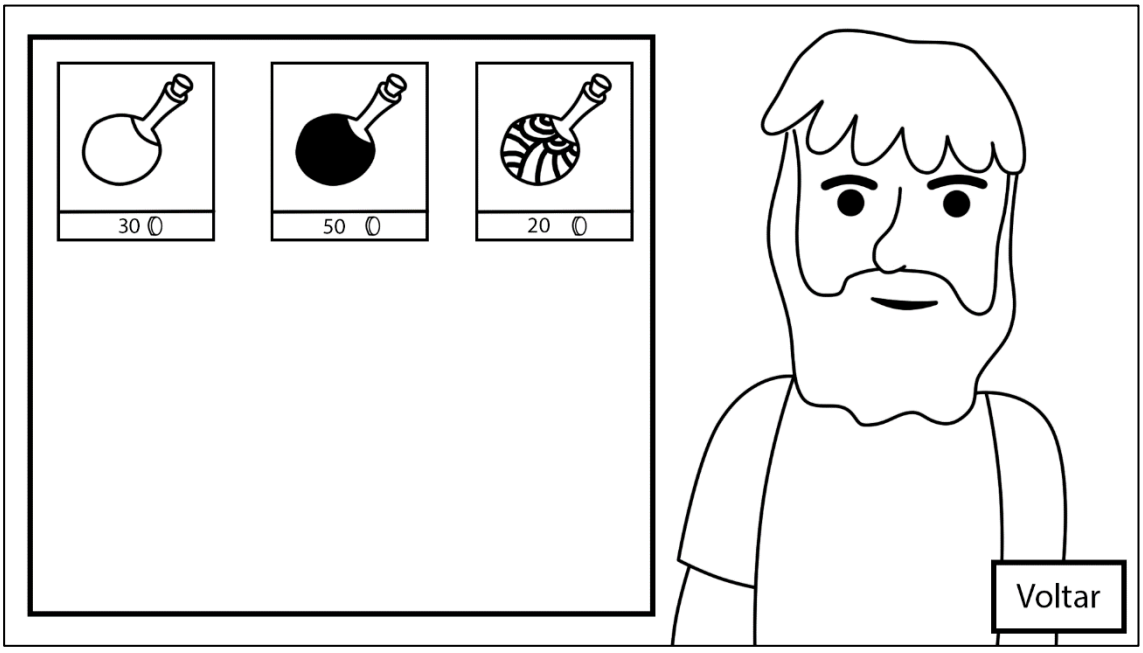


Figura 31 - Menu de compra

Ao logo do jogo o jogador irá encontra vilas nas quais terão lojas disponíveis para este aceder e poder comprar o que pretende ou necessite (Figura 31), para poder aceder a uma loja terá que se deslocar até à mesma que deseja e ao clicar na tecla "E" do seu teclado terá acesso ao menu da loja onde se encontraram os itens disponíveis para compra assim como o seu preço, se o jogador tiver o número de moedas

necessário para a compra, com o rato é só clicar com o botão esquerdo no item e assim o mesmo irá automaticamente para o seu inventário.

Para poder atacar os inimigos que irá encontrar ao longo da sua jornada o jogador poderá fazer ao clicar no botão esquerdo do rato e assim o seu personagem irá fazer um ataque, para poder fazer combos com o personagem é só ficar a clicar repetidamente no botão esquerdo do seu rato que o personagem irá fazer uma sequência de ataques. O jogador também terá acesso a um conjunto de habilidades especiais consoante a sua arma e para poder utilizar esses ataques o jogador poderá clicar no teclado numérico do seu teclado e assim usar as habilidades especiais, sendo a tecla "1" para a primeira habilidade, a tecla "2" para a segunda habilidade e a tecla "3" para a terceira habilidade. Estas habilidades só poderão ser usadas enquanto a barra de magia do jogador tiver suficiente magia para poder usar as habilidades.

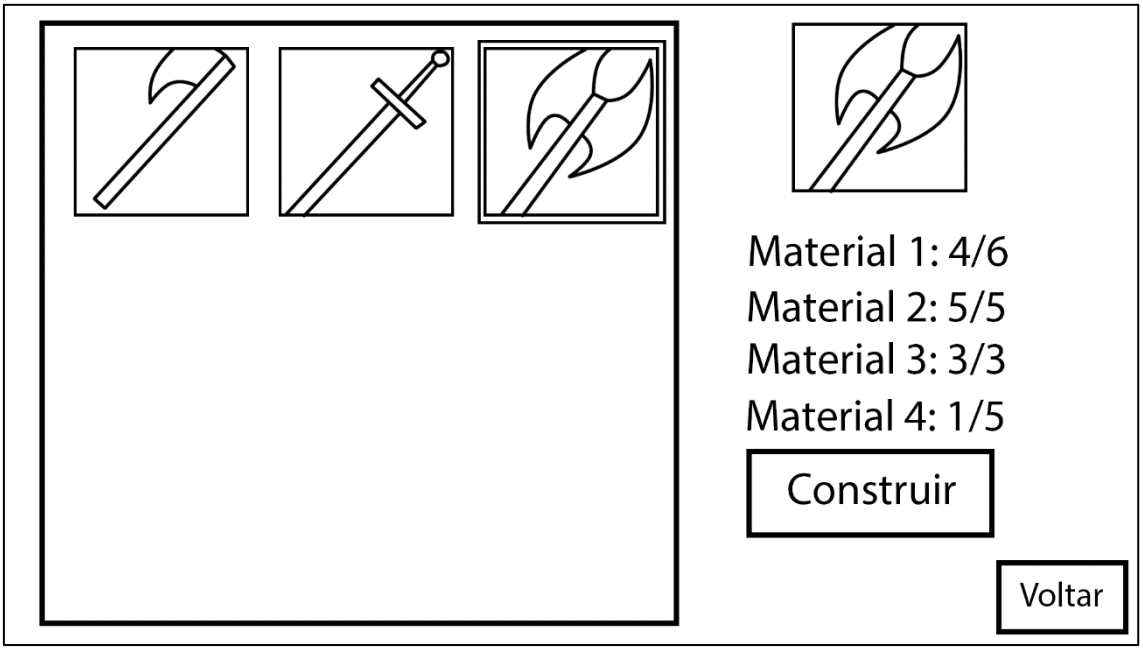


Figura 32 - Menu de construção

Para que o jogador possa construir as suas próprias armas o mesmo terá então de juntar os materiais necessários e deslocar-se até uma bigorna que se encontrará em ferreiros nas aldeias e clicar na tecla "E" do seu teclado para poder aceder à bigorna, quando acedida aparecerá um menu com uma lista de armas (Figura 34) onde aparecerão desbloqueadas apenas as armas para as quais o jogador tenha materiais suficientes para a sua construção. Quando o jogador tiver escolhido a arma que deseja construir basta clicar com o botão esquerdo do rato em cima da arma que queira e esteja desbloqueada e a mesma irá automaticamente para o inventário do jogador.

2.4.2 Perspetiva dos inimigos

Os inimigos situação espalhados pelo mapa do nível, as posições iniciais destes são sempre as mesmas sempre que o jogador entra nível, isto é, quando o jogador entra num nível e o completa derrotando todos os inimigos presentes, se este sair do nível e

voltar os inimigos voltaram a aparecer na mesma posição como estavam na primeira vez que o jogador acedeu a esse nível.

Quando um inimigo é atingido por vários golpes do jogador e estes sejam suficientes para levar a vida deste a 0, o mesmo é morto e dará ao jogador uma quantia aleatória de moedas consoante o seu tipo e também uma quantia de materiais.

Os inimigos iram atrás do jogador para o atacar e derrotar quando este atinge uma certa distancia dos mesmos, apenas os magos não perseguem o jogador e ficam em cantos escondidos a disparar as suas bolas de energia.

2.4.3 Interface do utilizador

O jogo irá contar com vários menus que permitiram ao jogador ter a possibilidade de começar um novo jogo, continuar o jogo anterior, poder também viajar pelo mapa para níveis ou vilas anteriores ou poder comprar ou construir novos itens.

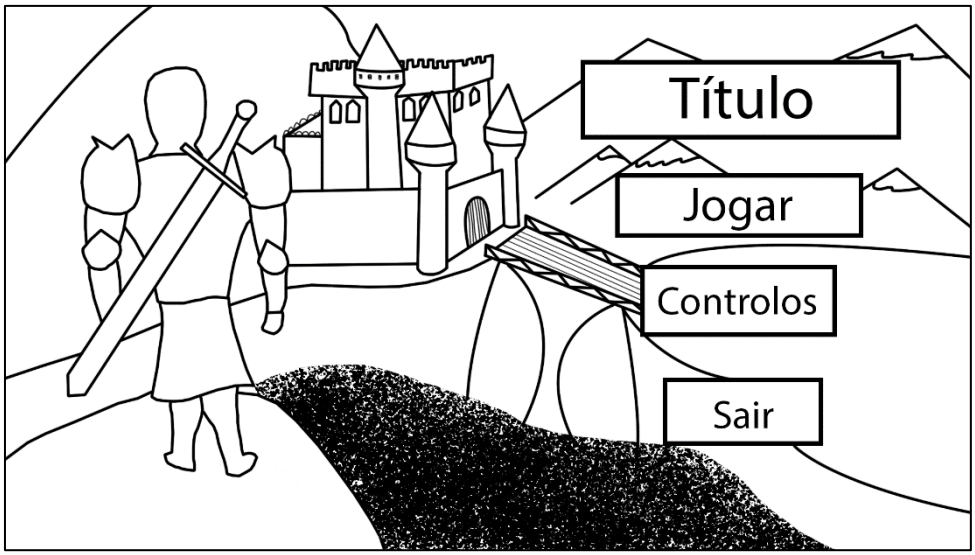


Figura 33 - Menu inicial do jogo

Como demostrado na Figura 33 é possível ver o menu inicial do jogo que irá ser implementado, este é constituído pelo titulo do jogo e por mais três botões na qual o jogador pode interagir, são estes o botão de jogar que irá levar o jogador ao mapa do jogo que contém os níveis, o botão de controlos que levará o jogador para um menu que permitirá visualizar as teclas que poderá usar e a sua função, tem também o botão de sair que que irá fechar o jogo, mas se o jogador eventualmente clicar na tecla “Esc” doi seu teclado neste menu o mesmo também irá fechar o jogo.

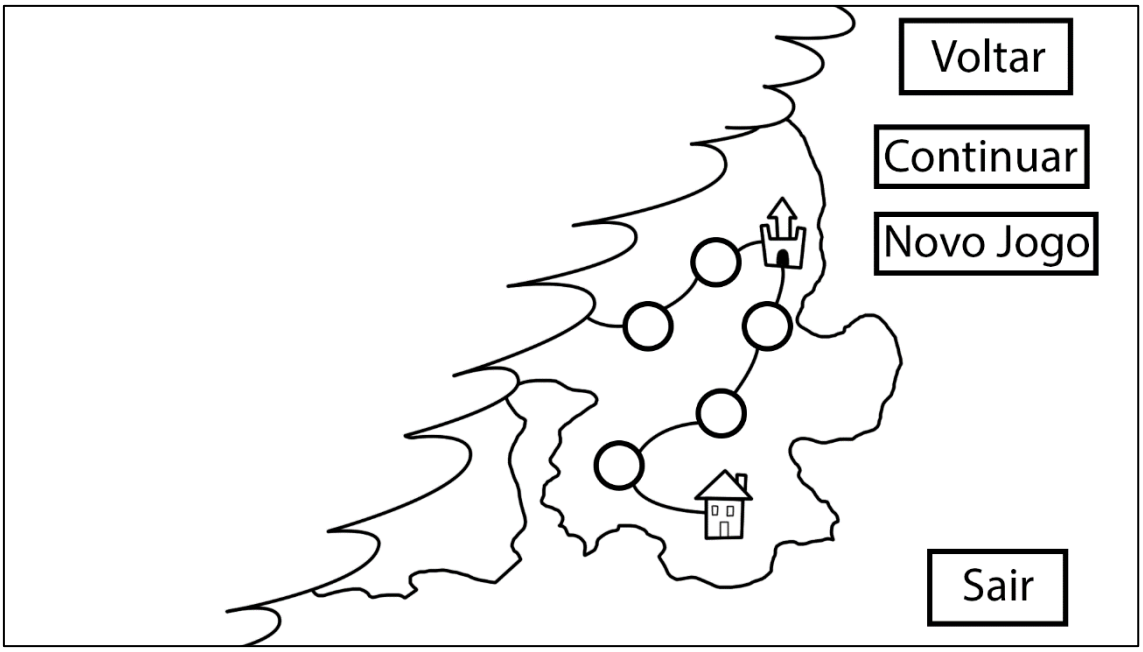


Figura 34 - Mapa do jogo

A Figura 34 mostra o menu que o jogador entra quando clica no botão no menu inicial, este menu mostra o mapa do jogo e os níveis disponíveis que o jogador pode entrar, os níveis bloqueados encontram-se tapados por uma nuvem que vai desaparecendo à medida que o jogador vai avançando no jogo e desbloqueando os níveis, este menu é também constituído por mais quatro botões sendo estes o botão Voltar para se o jogador quiser voltar ao menu inicial, o botão de continuar que levara o jogador ao novo nível desbloqueado da *última* vez que jogou, o botão de Novo Jogo que irá fazer com que elimine o progresso do jogo e inicie um jogo do zero e também o botão Sair que ira fazer com que o jogador saia do Jogo.

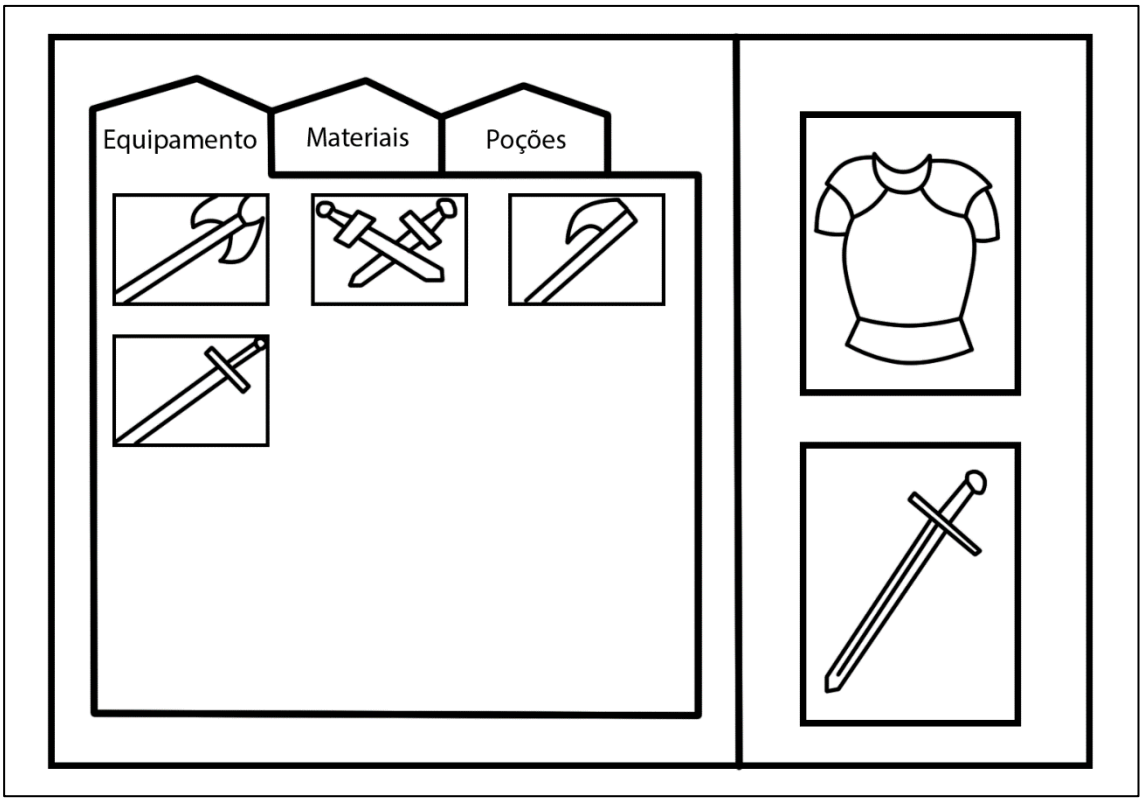


Figura 35 - Inventário do jogador

O jogador terá acesso a um inventário (Figura 35) onde terá acesso a todos os itens que o mesmo possui, mostrando assim todas as suas armas e armaduras, todos os materiais que possui e todas as poções que tem. Para mudar de arma o jogador apenas terá de clicar em cima do ícone da arma que deseja e a mesma irá aparecer no ícone da direita. Para usar poções é só clicar na aba que diz Poções onde aparecerão todas as poções que que o mesmo tem e é só clicar no ícone da poção que deseja e a mesma irá ser consumida automaticamente e o seu efeito aplicado.

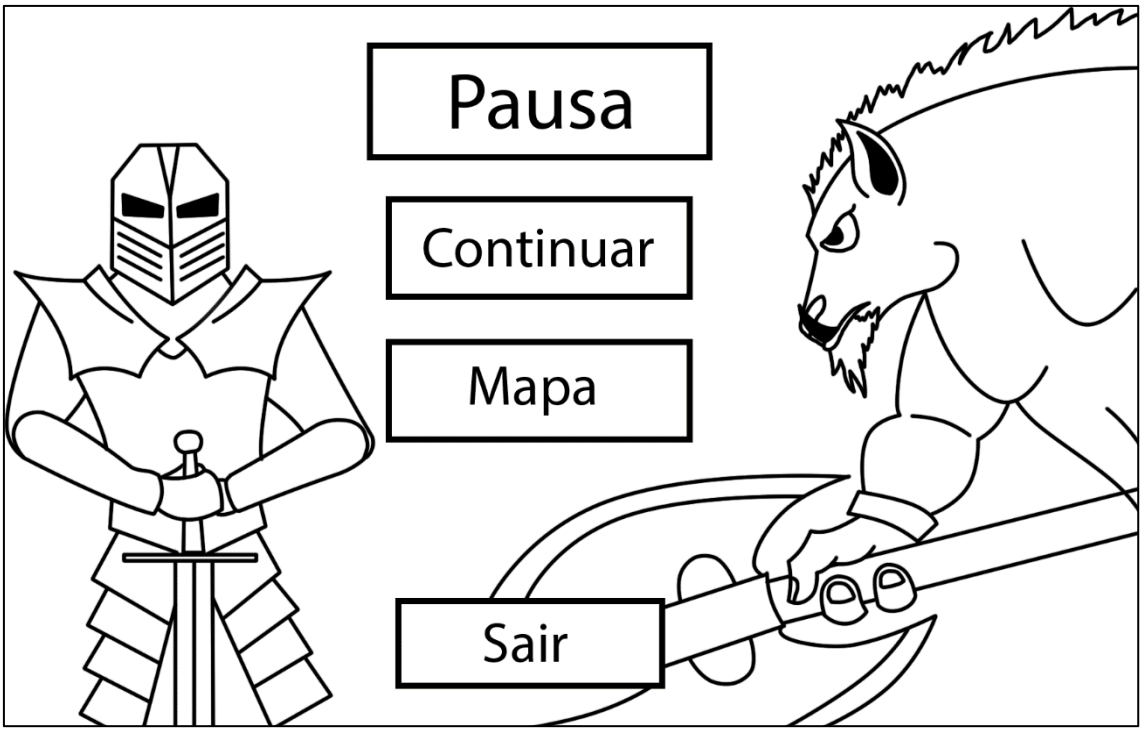


Figura 36 - Menu de pausa

O jogador poderá também por o jogo em pausa clicando na tecla "P" do seu teclado e assim ter acesso ao menu de pausa do jogo (Figura 36), este menu será constituído por três botões, o botão de Continuar que fará com que o jogador volte ao jogo, o botão Mapa que irá transportar o jogador para o menu do mapa e o botão de Sair que fará com que jogador saia do jogo.

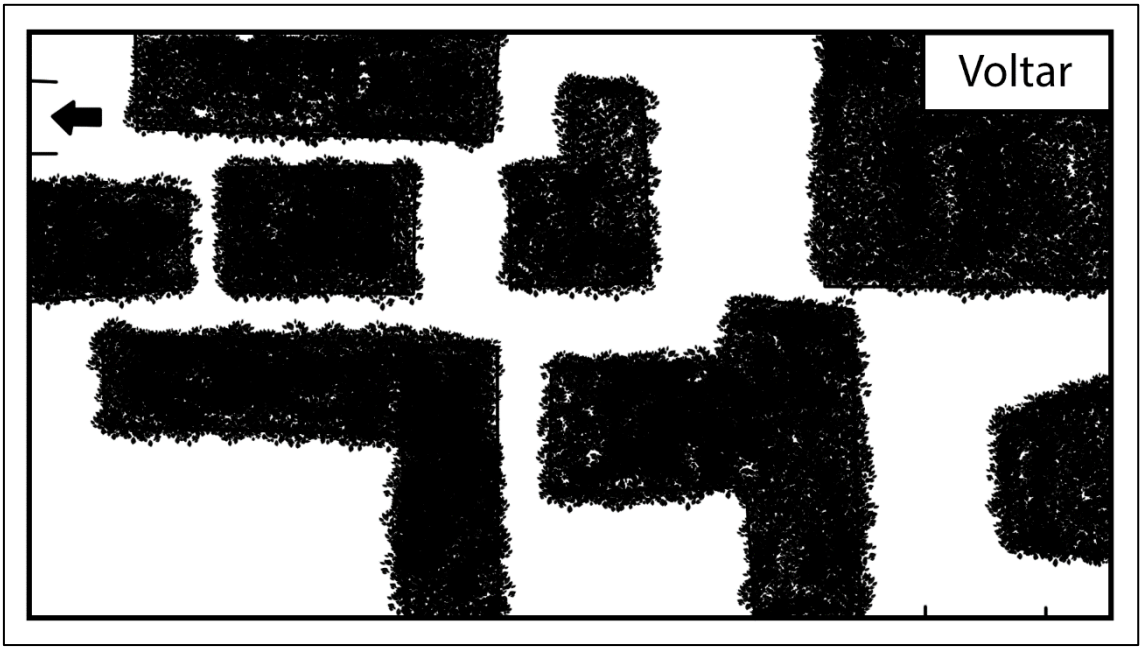


Figura 37 - Mapa de nível

Se o jogador quiser visualizar o mapa do nível basta clicar na tecla "M" do seu teclado quando estiver dentro do nível desejado e aparecerá o menu presente na Figura

37, neste menu o jogador poderá ver as áreas já exploradas e por explorar do nível, assim facilitando saber onde poderão faltar ainda inimigos por eliminar para poder passar o nível, terá acesso também a um botão Voltar para poder voltar ao jogo, ao aceder a este menu o jogo é colocado também em pausa.

2.5 Estrutura de dificuldade

O jogo está dividido por níveis podendo assim oferecer um início e um fim ao jogador. O jogo será constituído por 20 níveis sendo a dificuldade aumentada de forma crescente, ou seja, o primeiro nível será o mais fácil e o último o mais difícil. A dificuldade de cada nível vai depender do número de inimigos presentes em cada nível assim como o seu tipo, podendo também alterar características físicas nos mesmos de um nível para o outro, como aumentar a vida ou dano dos mesmos. A dificuldade depende também de como o jogador evolui o seu personagem, por exemplo se apenas evoluir o seu dano e as suas armas quando este num nível superior receber dano de um inimigo terá mais possibilidade de perder pois tem menos vida para poder suportar os ataques. Se o jogador eventualmente perder o mesmo é recolocado no início do nível e tudo é colocado onde estava inicialmente, isto é, o nível reinicia por completo e o jogador fica com as estatísticas e itens que tinha quando iniciou o nível.

2.5.1 Sistema de pontuação

Para tornar o jogo mais interativo e conseguir proporcionar uma maior dificuldade ao jogador trazendo um sistema de evolução baseado em *Grind*, isto é, o jogador terá de eliminar várias vezes os mesmos inimigos para que consiga obter recursos suficientes para construir os itens que deseja. O jogador terá duas alternativas para obter os itens, a primeira será recorrendo a matérias que os inimigos fornecem ao serem eliminados, a segunda opção será compra de vendedores com dinheiro do jogo que será também adquirido eliminando inimigos e concluindo níveis novos.

3 Implementação dos *Assets*

Neste capítulo serão mostrados todos os *assets* desenvolvidos para serem implementados no jogo e assim ser possível melhorar a experiência de jogo do jogador. Com a utilização das ferramentas Blender, Unity e Adobe Photoshop foi possível desenvolver todos os elementos gráficos para o jogo. Estas ferramentas permitiram a modelação de objetos tridimensionais, de efeitos visuais e a criação de animações.

3.1 Modelação e Animação

Neste subcapítulo serão descritos todos os elementos desenvolvidos e modelados no software Blender. Primeiramente será apresentado o modo como os elementos foram desenvolvidos e modelados, seguido de uma descrição do elemento e das suas funções dentro do jogo. Por último será apresentado uma descrição das animações desenvolvidas para os elementos que as contêm.

3.1.1 Processo de modelação

Para que fosse possível proceder à modelagem tridimensional dos personagens procedeu-se à utilização de diversas imagens de referência. Na Figura 38 é possível observar o processo de modelação de um personagem através da utilização de imagens como referência, com vista de frente e lateral. Para realizar esta modelação, começou-se por utilizar apenas metade do objeto em questão para que fosse possível torná-lo simétrico com a utilização do modificador de espelho do Blender, como mostrado no retângulo vermelho da Figura 38, obtendo assim o resultado apresentado no retângulo amarelo da Figura 38.

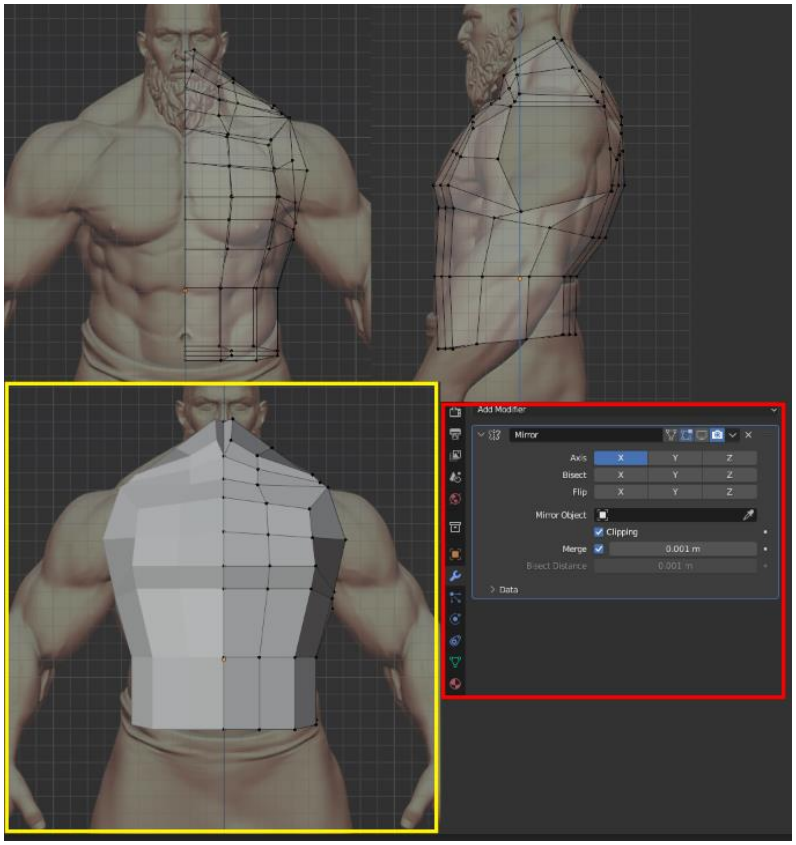


Figura 38 - Modelação do tronco

Após a conclusão da modelação do tronco seguiu-se a modelação dos braços. Para a modelação destes foram também usadas imagens de braços com a vista de frente e lateral. Na Figura 39 é possível observar a construção de um braço através do auxílio de imagens como referência, utilizando também o modificador de espelho como abordado anteriormente.

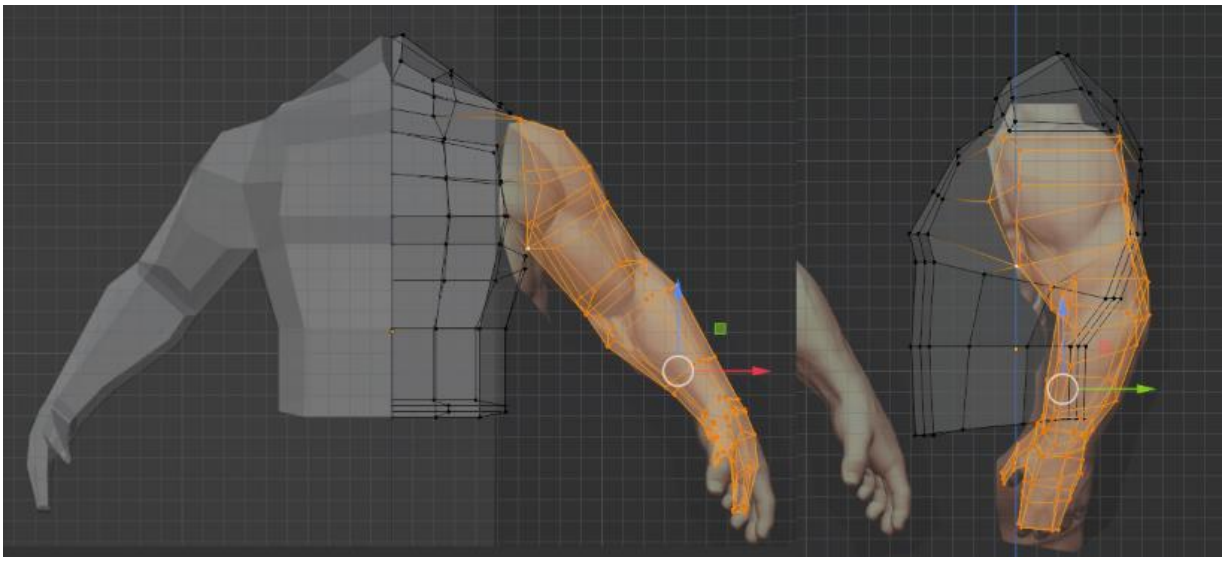


Figura 39 - Modelação dos braços

Em seguida procedeu-se à modelação das pernas do personagem. Na Figura 40 possível observar o processo de modelação das pernas usando tanto imagens de referência, como o modificador de espelho.

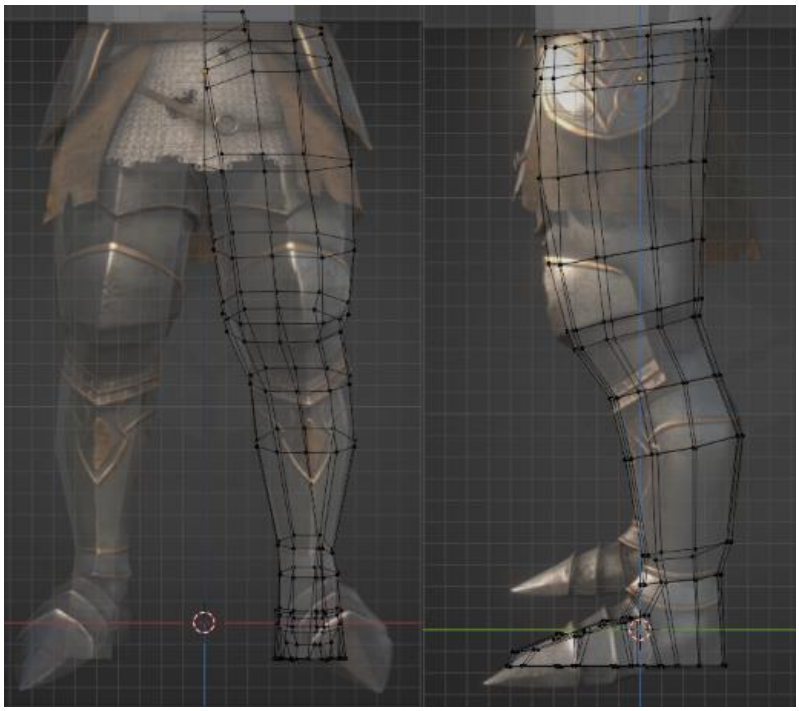


Figura 40 - Modelação das pernas

Para terminar o corpo na sua totalidade foi desenvolvido e modelado a cabeça do personagem. Na Figura 41 é possível observar a imagem usada como referência para o desenvolvimento da cabeça, usando a vista frontal e lateral para ser possível obter uma

forma tridimensional. Foi usado também o modificador de espelho do Blender para que o a cabeça fosse simétrica.

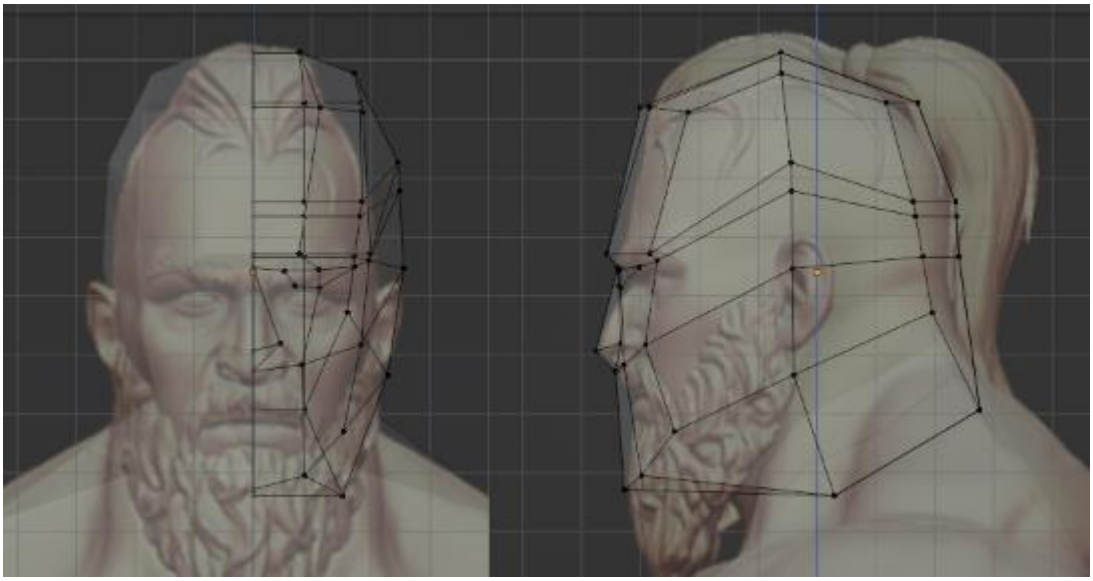


Figura 41 - modelação da cabeça

Por último foram modelados elementos e características únicas ao personagem, que o caracterizassem, para que este não fique apenas com um corpo vazio. Os adereços modelados variam entre peças de roupa, acessórios, capacetes, armadura e até mesmo armas. Na Figura 42 é possível observar alguns dos elementos modelados para uma personagem, sendo estes um capacete e algumas peças de armadura.

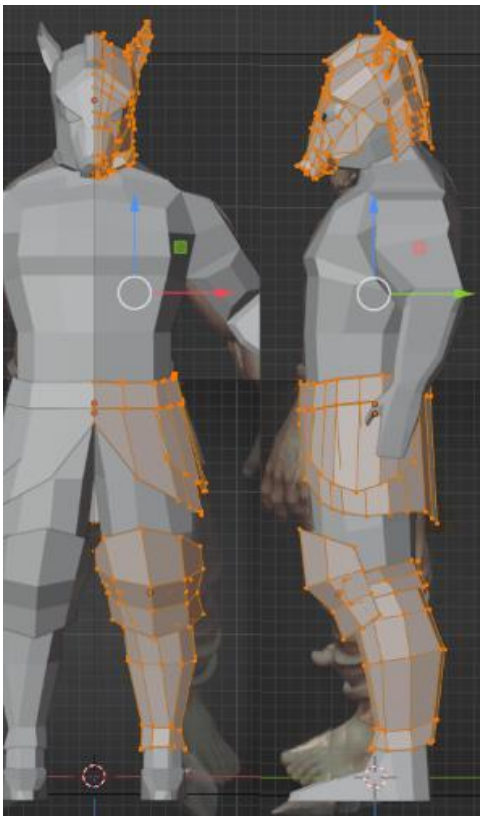


Figura 42 - Exemplo modelação de adereços

Para as texturas foram atribuídas cores solidas com base nas características de cada personagem. Na Figura 43 é possível observar o processo de aplicação de texturas a determinada parte da armadura de um personagem. Para isso é selecionado o elemento desejado (A na Figura 43), em seguida foi criado um novo material (B na Figura 43) e por fim selecionada a cor desejada (C na Figura 43).

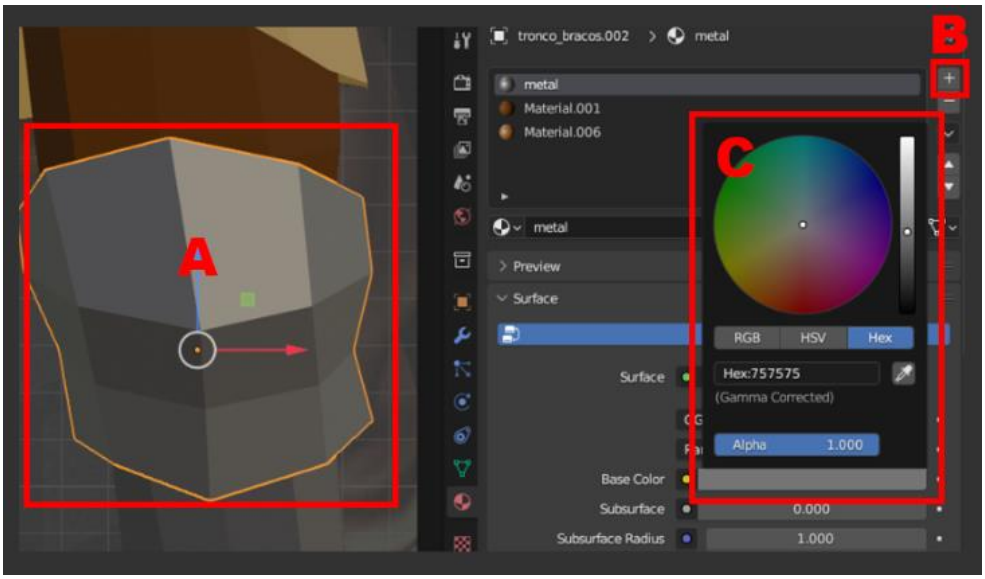


Figura 43 - Atribuição de texturas aos modelos

3.1.2 Processo de animação

Para gerar as animações foi preciso criar um esqueleto e atribuí-lo ao modelo desejado, fazendo com que cada parte do esqueleto esteja diretamente ligado à respetiva parte do corpo do modelo. Na Figura 44 é possível observar o processo de atribuição de um esqueleto a um modelo. Primeiramente é adicionado um esqueleto à cena clicando em "*Add*" (A na Figura 44), em seguida em "*Armature*" (B na Figura 44), o que vai fazer com que seja criado um único osso na cena. Posteriormente um osso é colocado no centro do modelo e com a ferramenta de *Extrude* (G na Figura 44) do Blender são criados novos ossos nas posições desejadas, formando assim o esqueleto completo. (D na Figura 44). Após colocar todos os ossos na posição desejada e ser criado um esqueleto, este é selecionado em conjunto com o modelo. Clicasse no esqueleto com botão direito do rato e seleciona-se "*Parent*" (E na Figura 44) e em seguida "*with Automatic Weights*" (F na Figura 44) para que o modelo fique preso ao esqueleto e assim seja possível movê-los em conjunto, tornando possível criar as diversas animações.

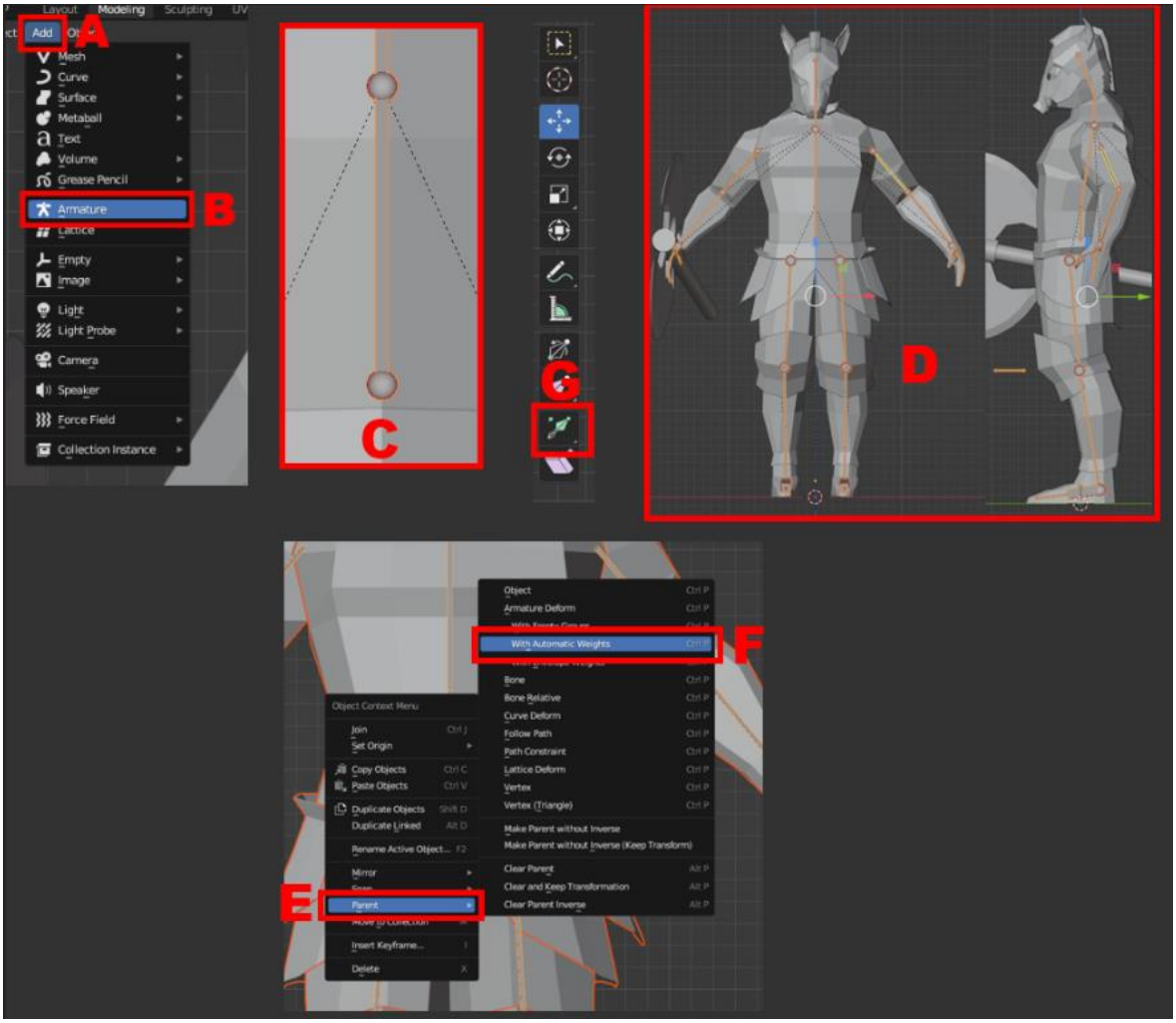


Figura 44 - Atribuição de um esqueleto a uma modelo

Todas as animações presentes foram também criadas no *software* Blender na janela de animação "Animation", como é possível observar em A na Figura 45. Para ser possível a criação destas animações, primeiro foi selecionada a armadura do personagem e colocado em modo de *pose* (B na Figura 45). Em seguida foi selecionado o osso que se deseja mover (C na Figura 45) e depois movido para a posição desejada. Após mover o osso é colocada a linha de tempo (D na Figura 45) na posição para guardar a posição do osso e, em seguida, com o botão direto do rato na cena é selecionado "Insert KeyFrame" na janela que aparece (E na Figura 45) e por fim na próxima janela que aparece é selecionado "Location, Rotation, Scale & Custom Propeties" (F na Figura 45) fazendo assim com que na linha do tempo fiquem guardadas todas as propriedades do osso no *frame* criado. Este conjunto de *frames* desenvolvidos na linha de tempo em D da Figura 45, quando colocados em play (G na Figura 45) formam assim uma animação.

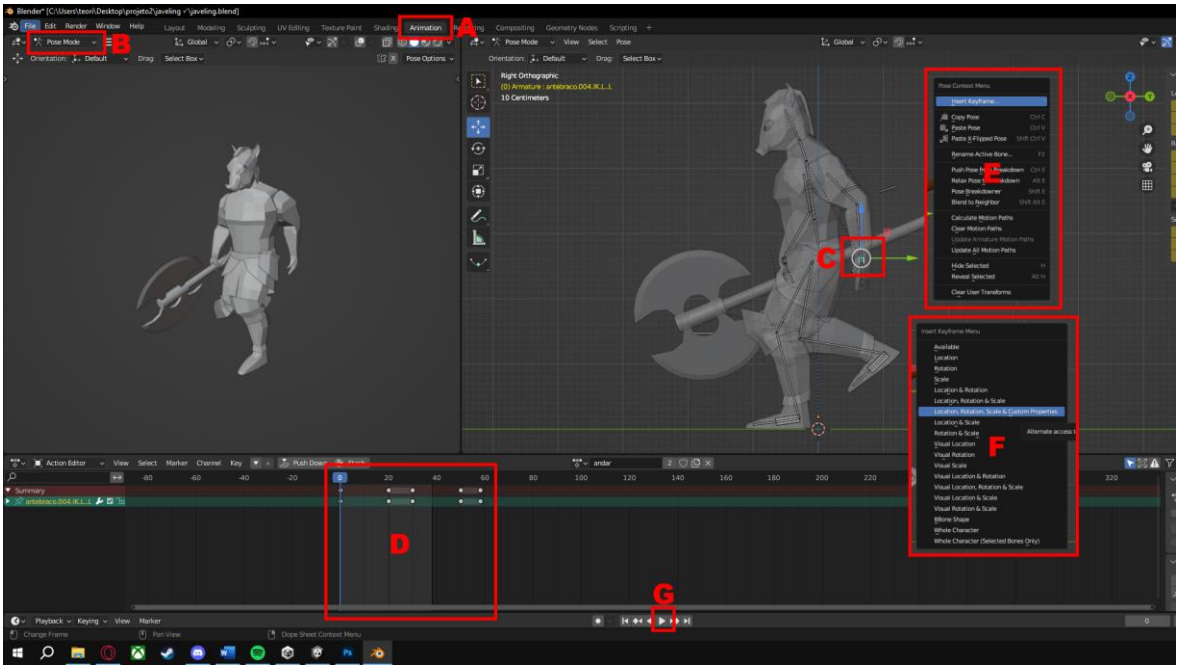


Figura 45 - Criação de uma animação

3.2 Personagens

O primeiro personagem a ser modelado foi o personagem principal, ou seja, Acolon. Como o personagem principal é um simples ferreiro que construiu a própria armadura, o mesmo tem uma armadura simples que apenas protege as zonas mais expostas a receber ferimentos, ou seja, tronco, braços e pernas. As restantes zonas do corpo permanecem expostas e apenas cobertas com roupa como é possível observar na Figura 46. Sendo então um ferreiro/cavaleiro da idade média foi dado ao personagem a cor cinza para a armadura, castanho para cabelo e roupa para simbolizar o couro. Para a sua saia foi lhe dado uma cor em tons amarelos para poder realçar em relação aos restantes elementos do corpo. No seu braço esquerdo encontra-se a pedra magica, de cor roxa de modo a simbolizar a magia dela. Por fim foi dado à pele do personagem uma cor bege. Acolon sendo o personagem principal do jogo é também o personagem que o jogador irá controlar durante todo o tempo que estiver a jogar.

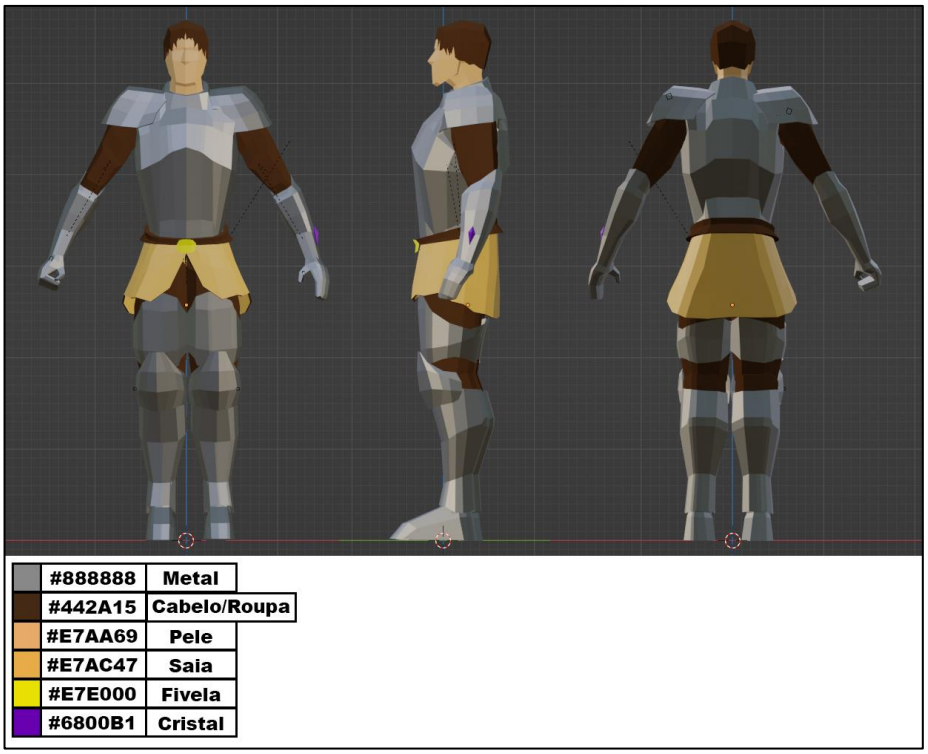


Figura 46 - Personagem principal Acolon

O personagem que foi modelado em seguida foi o inimigo Javeling como é possível observar na Figura 47. No GDD desenvolvido foi descrito que este personagem seria uma criatura metade javali metade humano, porém foi repensada a sua ideia e optou-se por humanizar todas as criaturas descritas no GDD, como é o caso do Javeling e as restantes que irão ser descritas mais à frente. Para criação do Javeling, foi lhe dado uma estrutura humana e um capacete em forma de javali para manter a sua identidade original (Figura 47). O Javaling tem na sua posse um machado grande e robusto para que possa produzir ataques que causem grande impacto, porém mais lentos em conformidade com o tamanho da arma. Visto ter uma arma bastante grande e pesada foi apenas desenvolvido uma armadura na zona inferior do corpo para se conseguir movimentar melhor e não ter que suportar mais peso extra além do machado, compensando a falta de armadura com ligeiro aumento sua vida total. Para o metal foi usada a cor cinza, para a roupa, madeira e couro foi dada a cor castanha, os olhos de cor vermelha para representá-lo como inimigo e com estado de espírito possuído, para a saia foi dado um amarelo-escuro e por último para a sua pele é usado um tom bege como representado na Figura 47.

Figura 47 - Inimigo Javeling

Uma das personagens que está mais presente ao longo do jogo é o *Dark-Knight* ou Cavaleiro Negro. Este por ser totalmente um cavaleiro é revestido por uma armadura tendo as cores à base de cinza representando metal, como é possível observar na Figura 48. Outra característica deste personagem é o facto deste andar sempre com uma espada e um escudo dando apenas a este personagem a habilidade de se defender. As cores usadas para a espada e escudo dividem-se em castanho para a madeira, cinza para metal e amarelo para os detalhes (representando ouro). A roupa interior deste personagem é feita de couro usando também o castanho para o representar. Por último é usada a cor preta para transmitir a ideia de profundidade e escuridão nos olhos e zona da boca presentes no capacete do personagem (Figura 48).

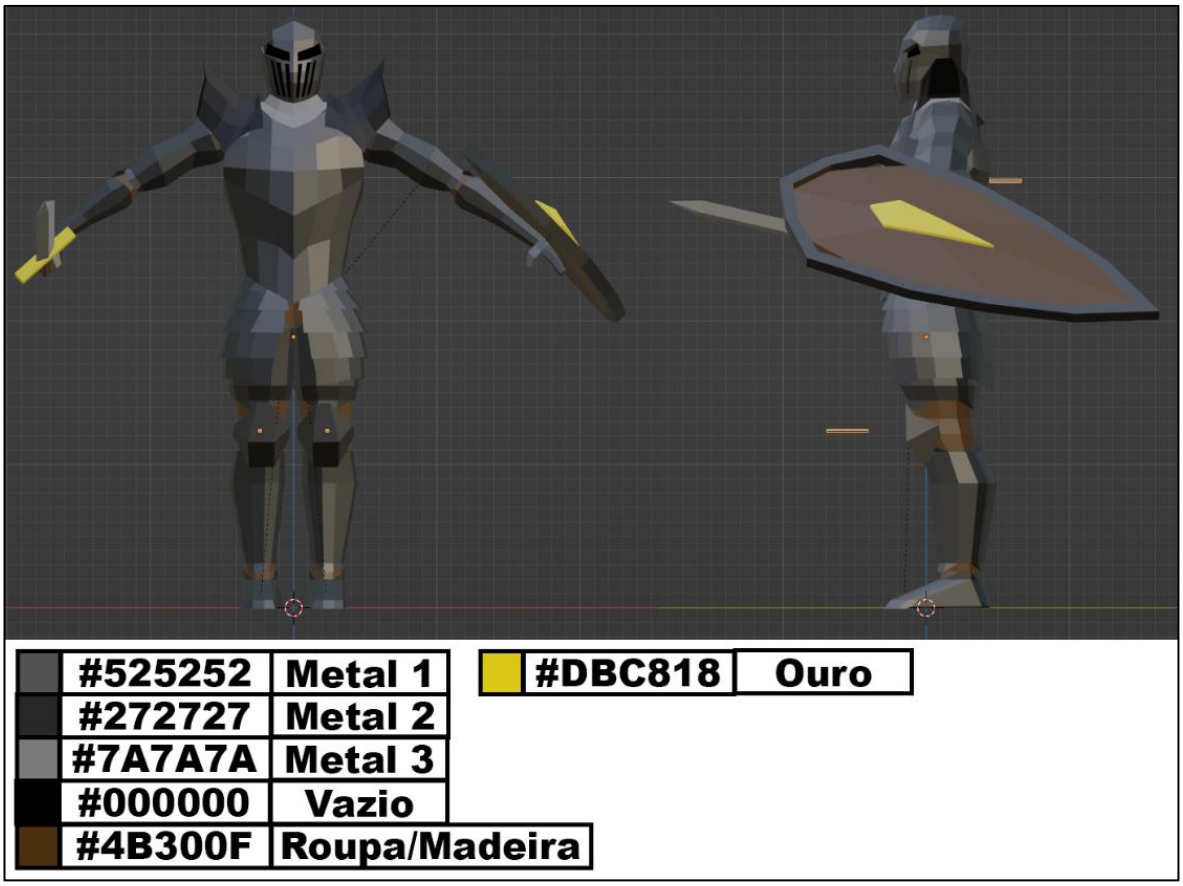


Figura 48 - Inimigo Dark-Knight (Cavaleiro Negro)

Outro dos inimigos descritos no CDD que era considerado uma criatura é o Crab. O Crab era metade caranguejo metade humano, fazendo com que fosse uma criatura de grande porte e com o corpo revestido por uma carapaça quase impenetrável com duas grandes garras capazes de produzir bastante dano ao personagem principal.

Porém com a humanização das criaturas o Crab passou a ter uma armadura grande e espessa capaz do proteger dos ataques assim como a antiga carapaça. Outra característica que foi alterada na sua estética foram as garras que por sua vez mudaram para dois grandes martelos para que possa dar o mesmo dano pensado anteriormente. Estas características são possíveis de observar na Figura 49.

As cores escolhidas para o Crab foram os tons de cinza para as zonas de metal, como a armadura e os martelos. Para a roupa foi-lhe dado a cor castanha para representar o couro. Para o acabamento do martelo foi lhe dado a cor amarela para representar ouro. Por último e assim como o Cavaleiro Negro (Figura 48) foi lhe dado a cor preta na zona do capacete para produzir a mesma ideia de profundidade e escuridão (Figura 49).

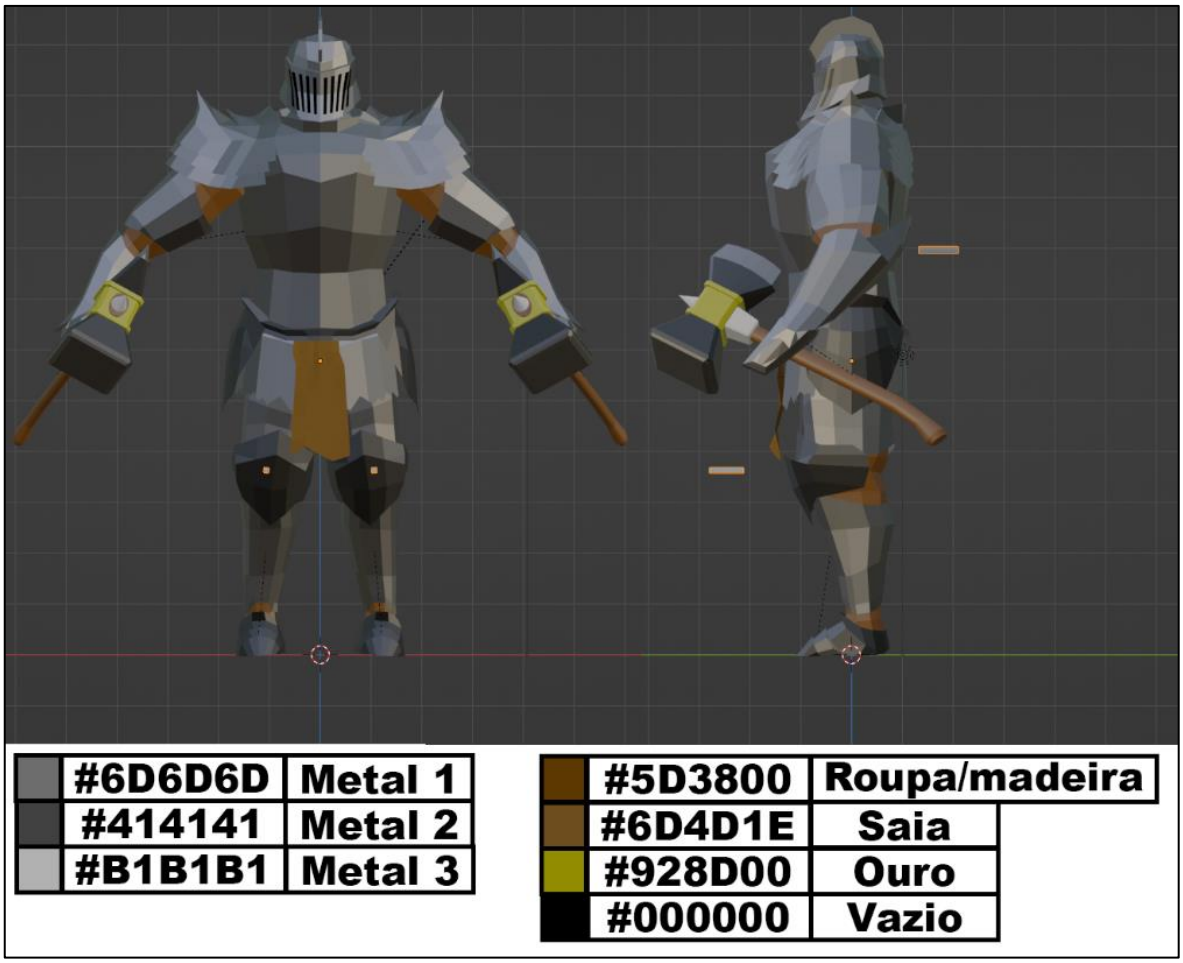


Figura 49 - Inimigo Crab

O *Bombist* também era um dos personagens representado por um a criatura composta por apenas um esqueleto no GDD desenvolvido, porém passou a ser humanizado e também teve o seu nome alterado de Esqueleto-Bomba para *Bombist* devido à sua humanização. O *bombist* era um pequeno esqueleto que corria pelo mapa com uma bomba gigante no seu colo e para manter essa característica foi colocada a bomba numa mochila nas costas do pequeno homem como é possível observar na Figura 50.

Este personagem não tem qualquer tipo de proteção pelo que lhe foi dado apenas uma t-shirt, umas calças normais e umas luvas, ombreiras e um capacete para o incluir no tema medieval. A bomba como dito anteriormente foi colocada numa mochila que o personagem traz sempre consigo às costas (Figura 50).

As cores usadas neste personagem são cores simples e neutras, dando destaque apenas na t-shirt que é vermelha para ser possível identificar melhor o inimigo e azul para os seus óculos. O cinza é usado para as partes de metal e nas calças. A bomba é composta pela cor preta para a tornar mais discreta. Por último o couro usado nas botas, cinto e mochila tem uma cor castanha (Figura 50).

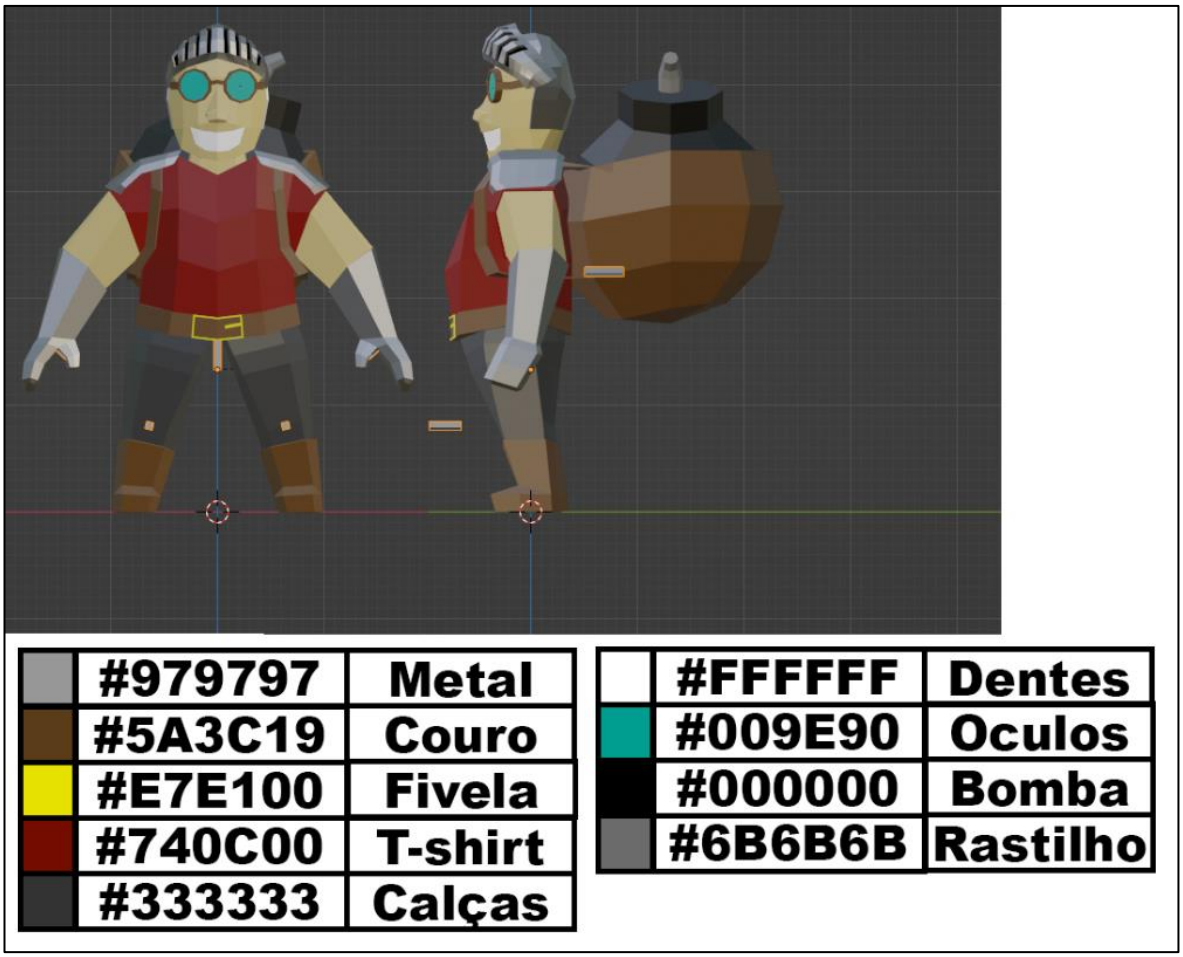


Figura 50 - Inimigo Bombist

O inimigo Mage ou Mago é um personagem que usa magia como método de ataques, pelo que lhe foi dado um manto de cor azul com detalhes a dourado para o representar (Figura 51). Este personagem contém um cajado de madeira com cor castanha, que usa para lançar os seus poderes, contendo uma bola de energia azul na sua ponta. Para representar o mistério e a escuridão do personagem foi lhe dado a cor preta para o seu interior.

Figura 51 - Inimigo Mage

Um dos inimigos descritos no GDD desenvolvido era uma gosma que iria andar pelo mapa a vaguear à procura do jogador para o atacar, porém com a humanização dos inimigos levou a que esse personagem tivesse de ser completamente repensado. Foi então descartada a ideia da gosma e dada uma nova cara ao personagem criando algo que represente melhor a idade medieval, um cavaleiro montado a cavalo como é possível observar na Figura 52. Com a alteração do personagem foi alterado também o nome de Glob para Horse-Knight ou Cavaleiro com Cavalo.

As cores escolhidas para este personagem baseiam-se nas cores de cavaleiros da idade média. Destacando as cores cinza para a armadura do cavaleiro, castanho para a a pele do cavalo e o vermelho para o seu manto para assim o identificar como inimigo (Figura 52).

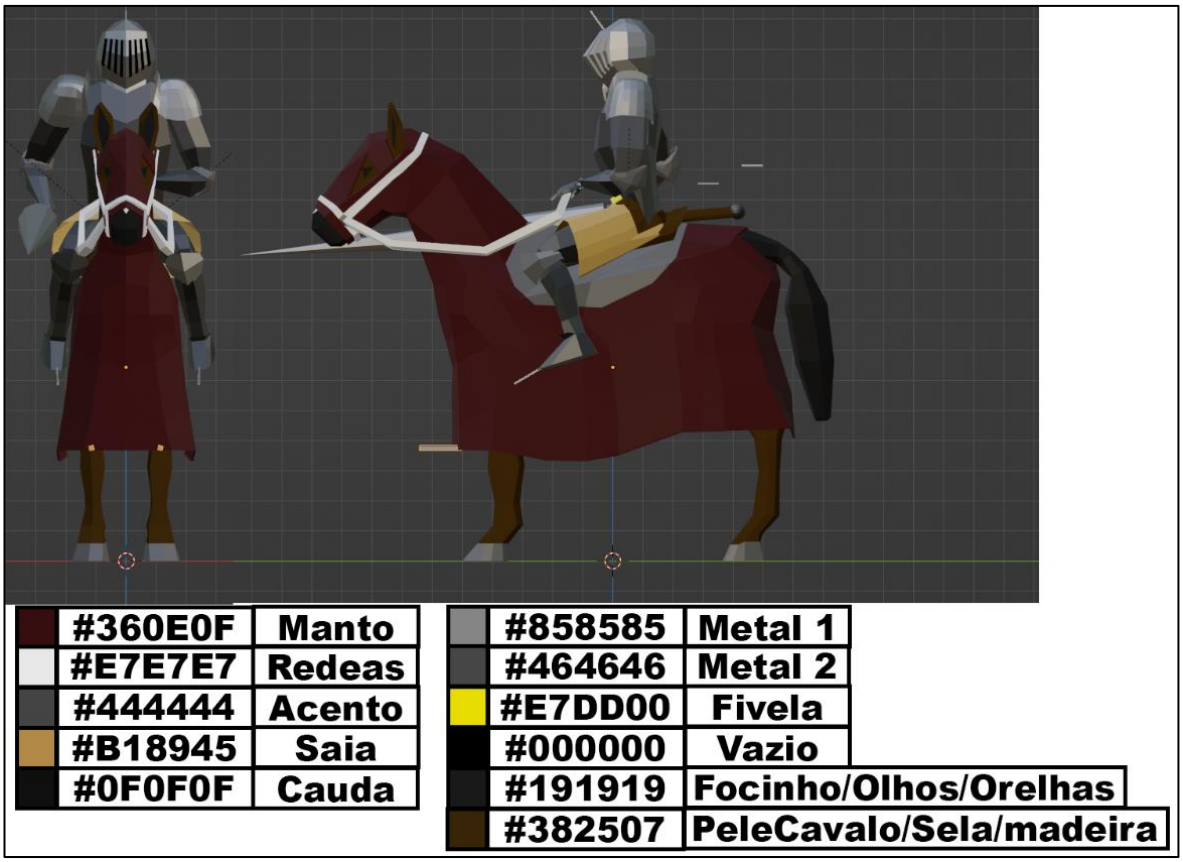


Figura 52 - Inimigo Horse-Knight

O último inimigo a ser desenvolvido foi o inimigo final, ou seja, Baelzor (Figura 53). Este personagem tem o objetivo de ter uma imagem simples pelo que lhe foi dado poucos adereços, destacando-se apenas alguns adornos como cristais de cor azul e umas pulseiras em ouro e prata. A sua principal cor é a cor da sua pele por ser a zona mais exposta deste personagem como é possível observar na Figura 53.

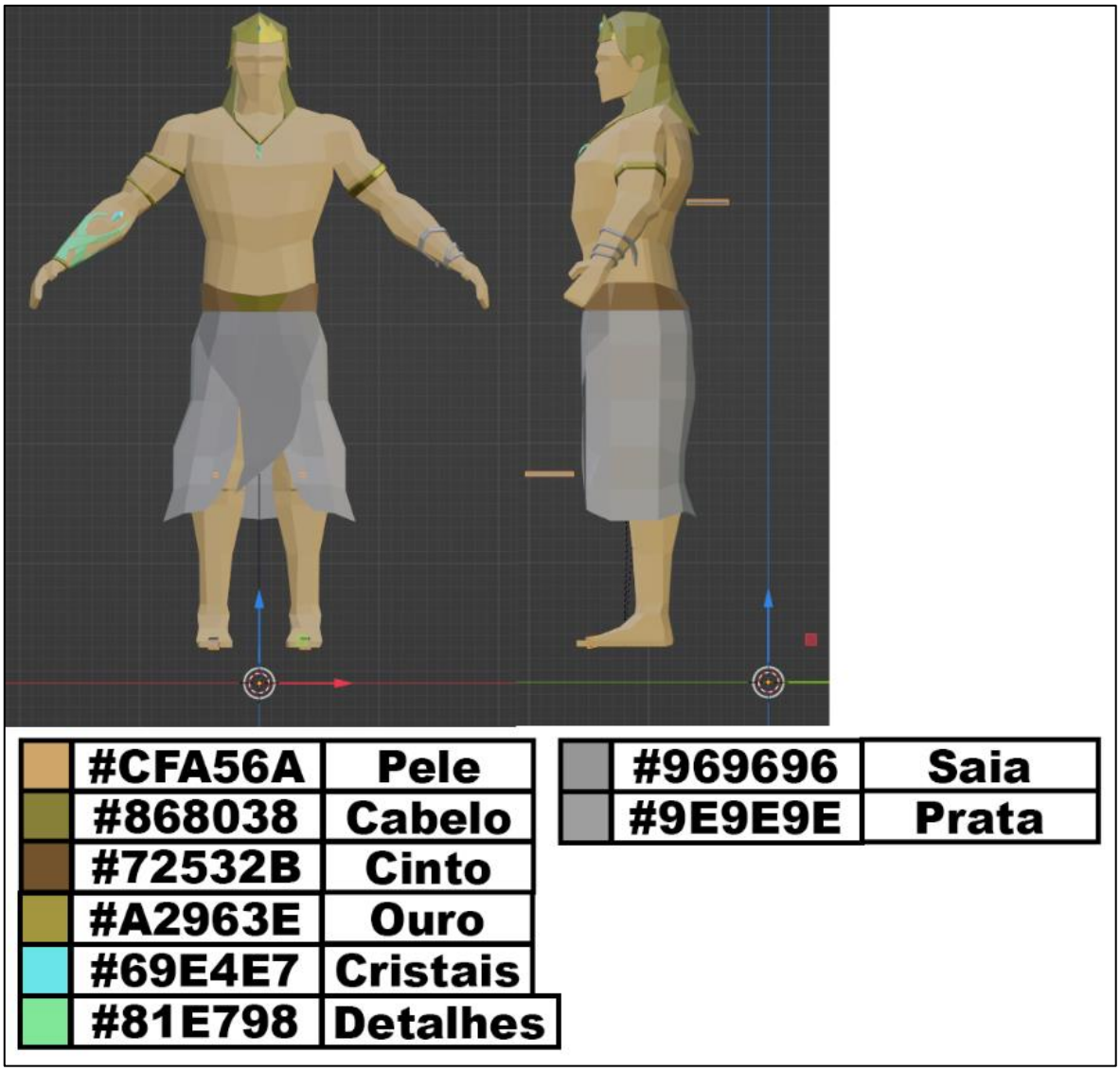


Figura 53 - Inimigo Baelzor

3.3 Cenário

Neste subcapítulo serão descritos todos os *assets* desenvolvidos para criar os diversos cenários dos níveis. Aqui serão mostrados todos os elementos usados na construção do terreno dos níveis. Será também apresentado todas as estruturas (muralhas e casas) criadas para tornar os níveis mais interativos.

O jogo tem quatro ambientes diferentes, sendo eles os campos verdes, as planícies de fogo, as montanhas de gelo, e por fim um castelo. . Será apresentado as diferenças de todas as estruturas inseridas no seu respetivo ambiente, bem como as diversas cores presentes.

Na Figura 54 é possível observar os elementos desenvolvidos para serem usados como elementos caminháveis onde o jogador terá que caminhar para percorrer os níveis e encontrar os vários inimigos e objetivos. Estes elementos foram construídos com de forma a encaixarem uns nos outros para construir caminhos que nos levam em

várias direções dentro de cada nível, onde podemos encontrar inimigos, muralhas, casas, árvores e também nos levam do início ao fim dos níveis.

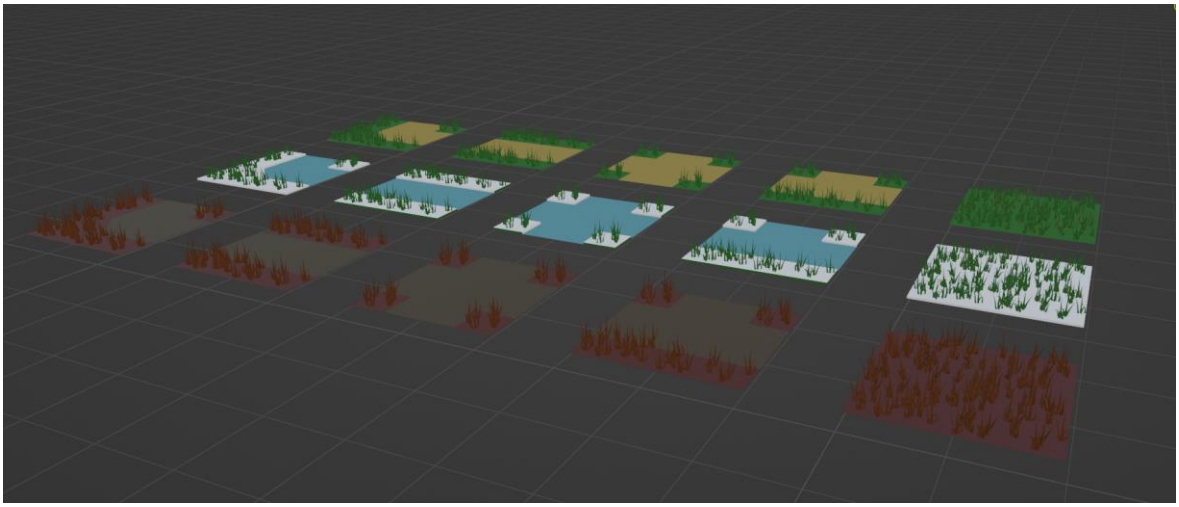


Figura 54 - Elementos caminháveis

Na Figura 55 é possível observar os elementos desenvolvidos para determinar os limites de cada nível. Visto que os níveis são todos ao ar livre com tema de floresta, à exceção do último nível, surgiu a ideia de dar aos limites de cada nível a forma de árvores pela qual o jogador não pudesse passar.

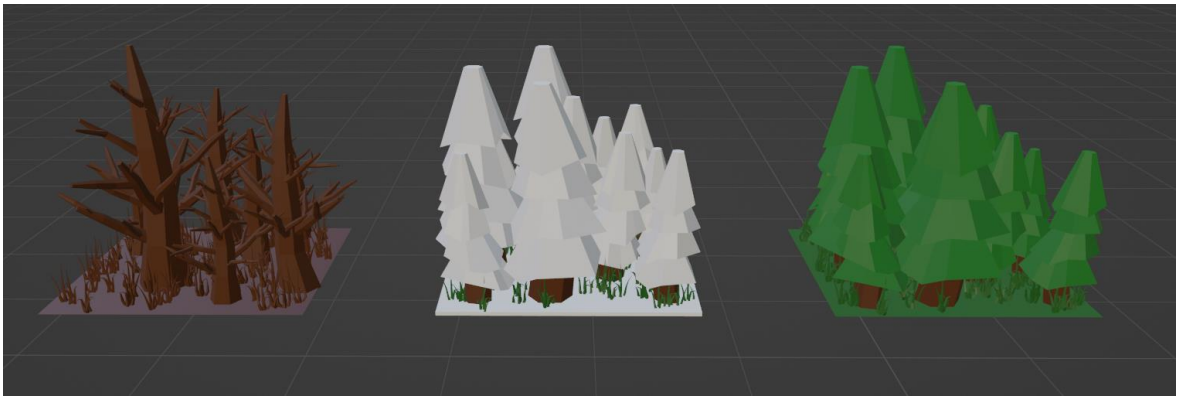


Figura 55 - Elementos delimitadores de nível

Devido à utilização do tema medieval foram desenvolvidas muralhas para dar mais vida aos níveis do jogo, muralhas essas que é possível observar na Figura 56. Estas muralhas têm como objetivo determinar também certos limites dentro do nível, mas também dar ao jogador a oportunidade de poder desbloquear novas zonas dentro do nível para assim obter mais recompensas. Estas foram desenvolvidas de maneira que fosse possível juntar várias e assim estruturas com infinitas formas e tamanhos.

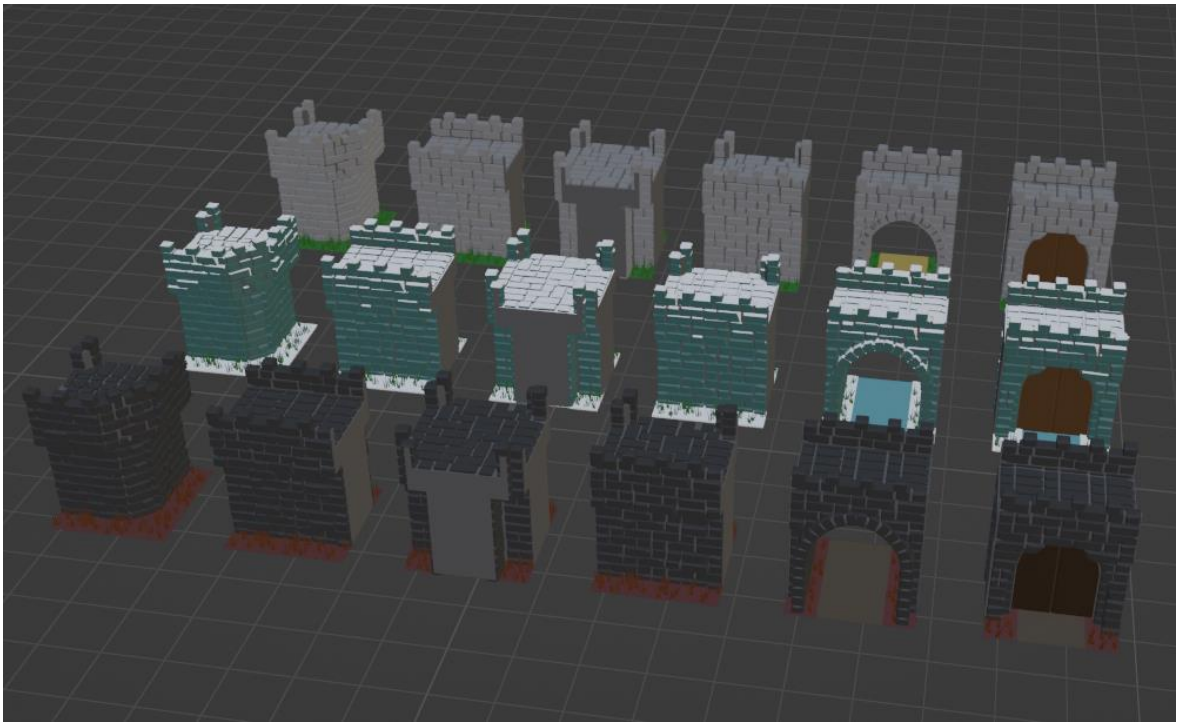


Figura 56 – Muralhas

Para que cada nível não fosse apenas floresta e inimigos foram desenvolvidas casas para dar mais diversidade de estruturas, casas essas que podem ser observadas na Figura 57. Estes elementos foram desenvolvidos de maneira que possam ser usados em conjunto com os elementos anteriores de forma a se encaixarem na perfeição uns com os outros podendo assim formar, por exemplo, pequenas aldeias ou vilas. A cada casa foi dada uma forma e cor de telhado única, adaptável consoante o ambiente em questão, para haver diversidade dentro dos níveis.

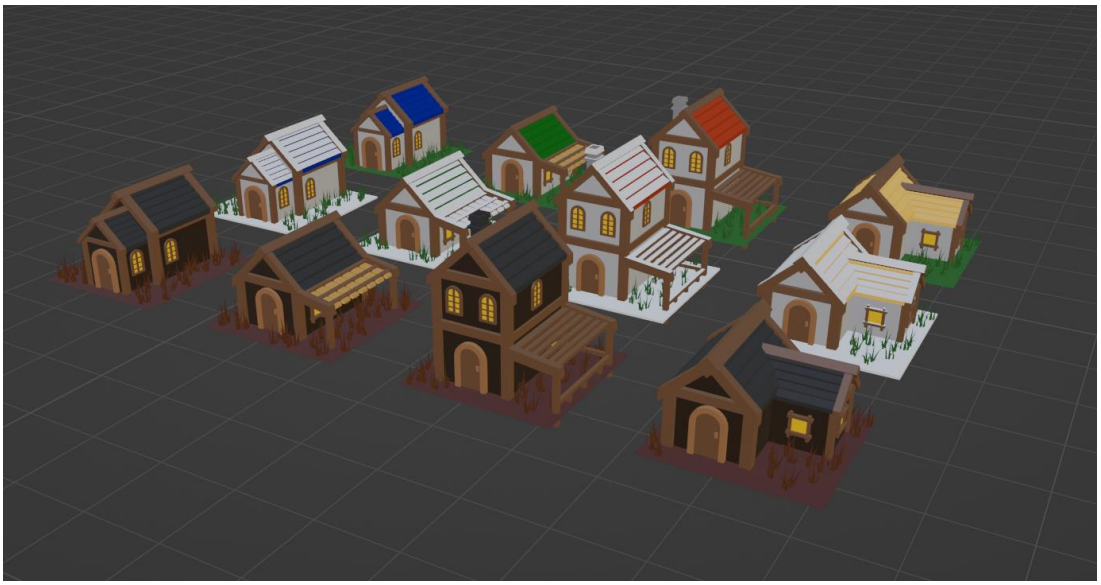


Figura 57 – Casas

Outro tipo de casa que o jogador poderá encontrar é a casa do ferreiro (Figura 58). Esta é uma casa onde o jogador poderá interagir para fabricar novas armas. O ferreiro

poderá ser identificado pela sua chaminé e pela sua bigorna na entrada da casa, e encontra-se sempre situado em vilas.

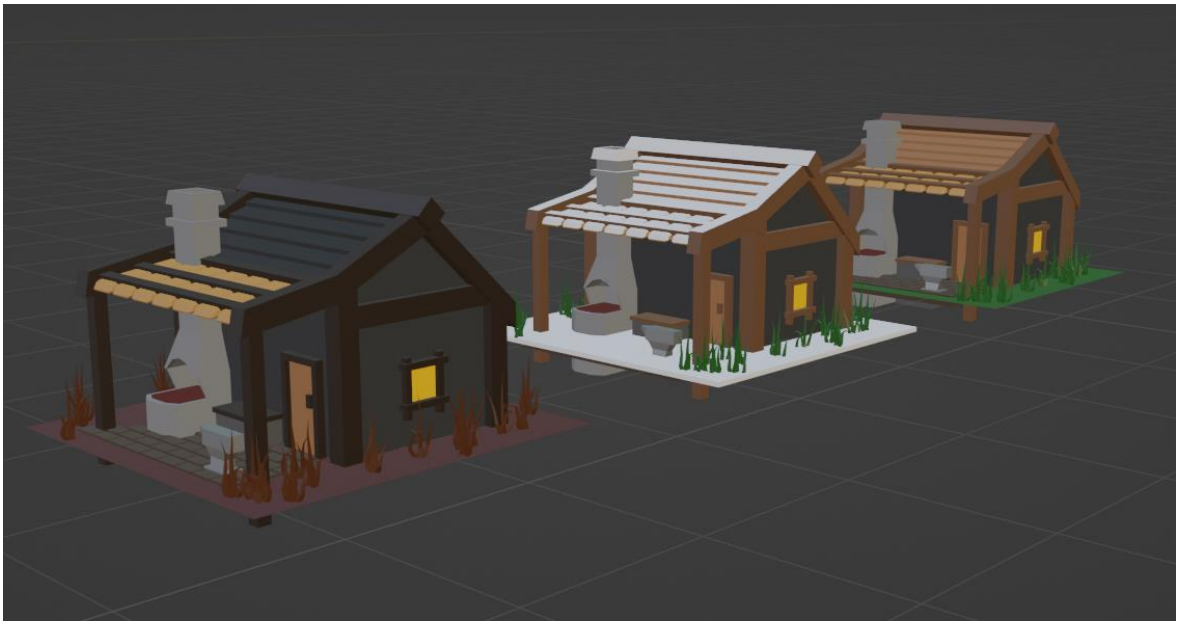


Figura 58 – Ferreiro

Outro tipo de casas interativas onde o jogador poderá aceder são as lojas. Existem dois tipos de lojas, a de poções e a de materiais. Em cada aldeia apenas existe 1 loja de cada e um ferreiro, porém se a vila tiver grandes dimensões é possível que aparecem múltiplas lojas na mesma vila. Em caso de dificuldade em encontrar, também é possível identificar as lojas através de pequenos objetos encontrados na frente de cada loja, na loja de poções é possível encontrar pequenos frascos de poções como os que estão representados no lado esquerdo da Figura 59 e a loja de materiais é possível encontrar uma placa com um martelo e um serrote (lado direto da Figura 59).

Figura 59 - Identificadores de lojas

Para o nível final foi escolhido um tema mais sombrio de forma a intimidar o jogador, com tema de castelo como é possível observar na Figura 60, onde se encontra o chefe final. Estes elementos foram construídos com o intuito de se puderem juntar uns aos outros de forma a formarem uma estrutura única e completamente diferente dos restantes elementos dos níveis anteriores destacando o nível final do jogo.

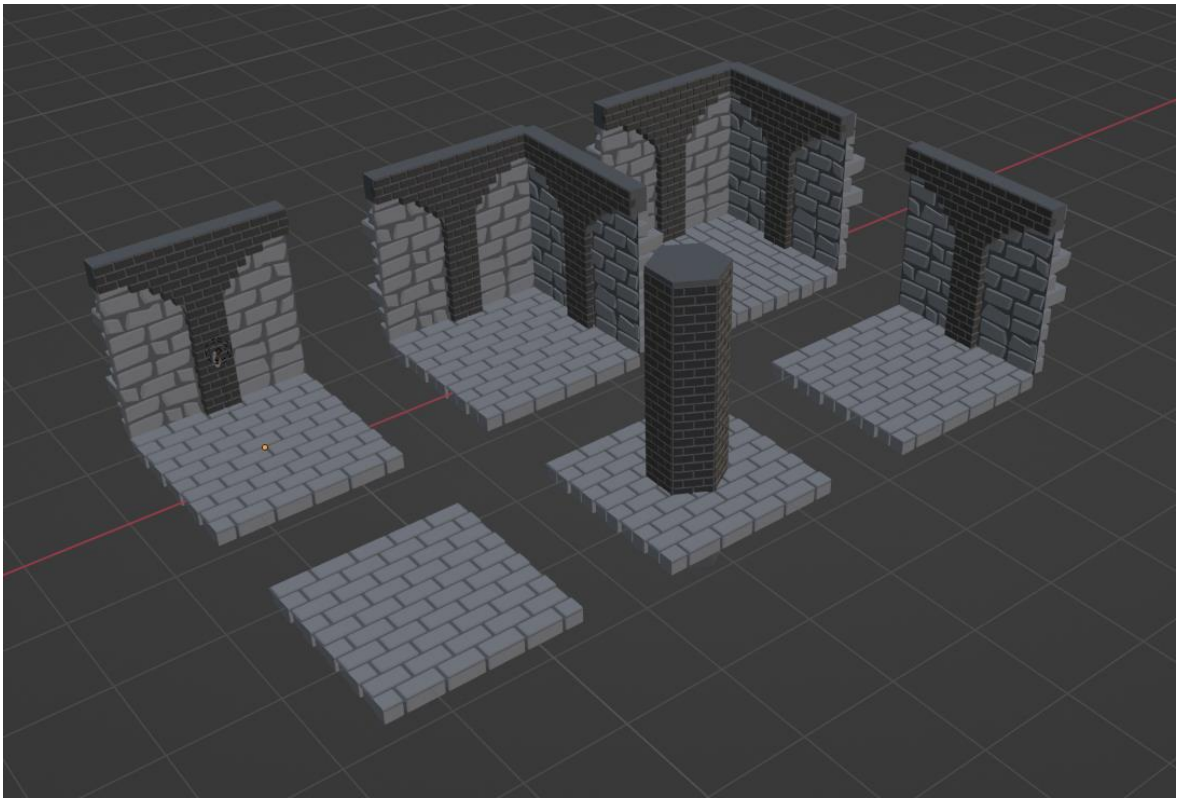


Figura 60 - Elementos do nível final, Castelo

Nos elementos do cenário de Campos Verdes podemos observar na Figura 61 que a cor predominante é o verde. Os muros contêm uma cor mais acinzentada para poder representar as pedras usadas para construir os mesmos. As casas contêm também uma cor semelhantes umas às outras sendo a maior diferença o seu formato e a cor do telhado.

Nos elementos do cenário de Planícies de fogo existe uma ausência de cores, como é possível observar na Figura 61, pois pretendia-se que este cenário transmitisse a ideia de terras sombrias e com falta de vida, optando por utilizar como cor predominante os tons em castanho e cinza-escuro.

No cenário das Montanhas de Gelo as cores predominantes são o azul e branco representando o gelo e neve. Nos muros deste cenário as pedras usam a cor azul também para representar o gelo. As casas contêm a mesma lista de cores que as casas do cenário de Campos Verdes porem cobertas de cor branca representando neve. Por último o cenário do nível final no interior do castelo, as cores que predominam são os tons de cina para representar as várias pedras da construção do castelo (Figura 61).

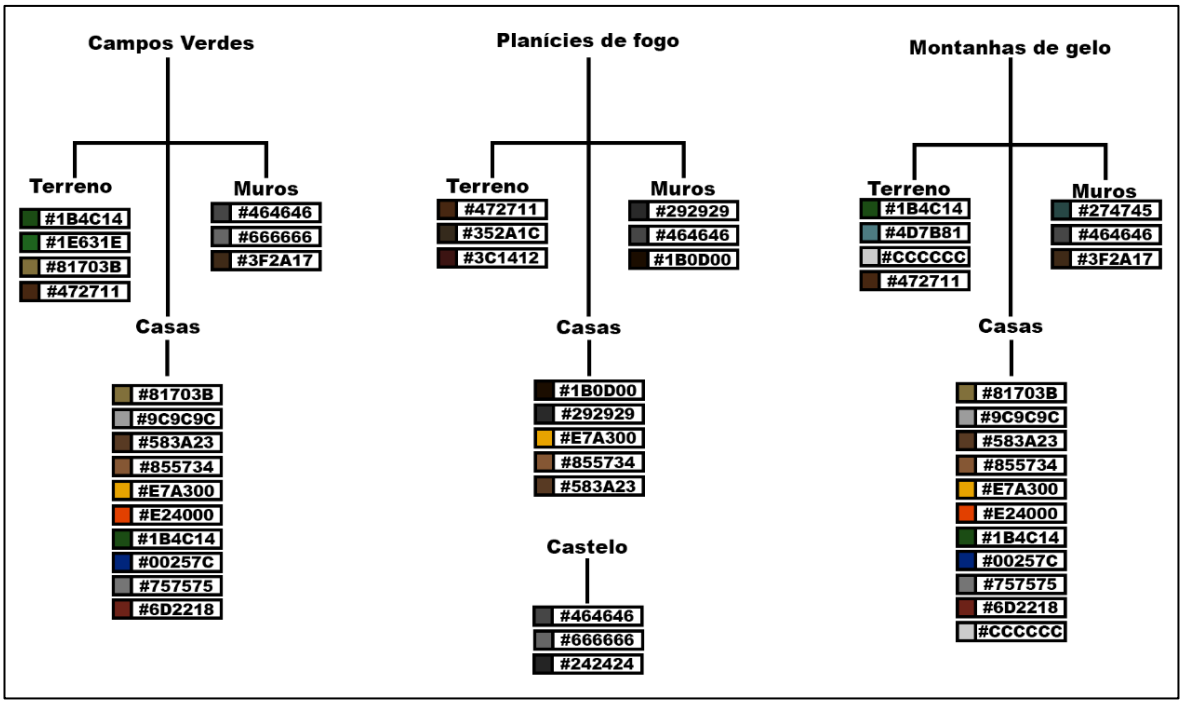


Figura 61 - Cores dos elementos do Cenário

3.4 Armas

Neste subcapítulo serão mostradas as armas do jogo. Para cada tipo de arma haverá uma descrição e explicação das suas variações. Serão também explicadas as cores usadas para o desenvolvimento de cada arma.

A primeira arma a ser desenvolvida foi a espada. A espada tendo inspiração nas espadas usadas na idade média, tendo um comprimento de lâmina bastante longa em comparação com o seu punho como pode ser observado na Figura 62. As cores usadas na espada são também elas inspiradas nas espadas da idade média, ou seja, lâmina de cor cinza para representar o metal usado, pequenos detalhes em ouro, guarda de cor escura e por último o castanho no punho para representar o couro usado.

A espada divide-se em duas variações, sendo estas a espada de fogo e gelo podendo ser observadas na Figura 62. Estas espadas têm forma e tamanho idêntico à espada normal, diferenciando apenas na guarda e algumas das cores. A espada de fogo contém uma lâmina vermelha para poder representar o fogo e mais à frente será mostrado o efeito de fogo da mesma. Sendo necessário utilizar um cristal de fogo para ser construída, a mesma tem um cristal de fogo no seu fundo e pequenos detalhes em vermelho para o representar. A espada de gelo contém a lâmina igual à espada normal, porém é revestida por uma camada de gelo e alguns cristais de cor azul. Sendo esta construída com um cristal, no fundo contém também um cristal de gelo de cor azul.

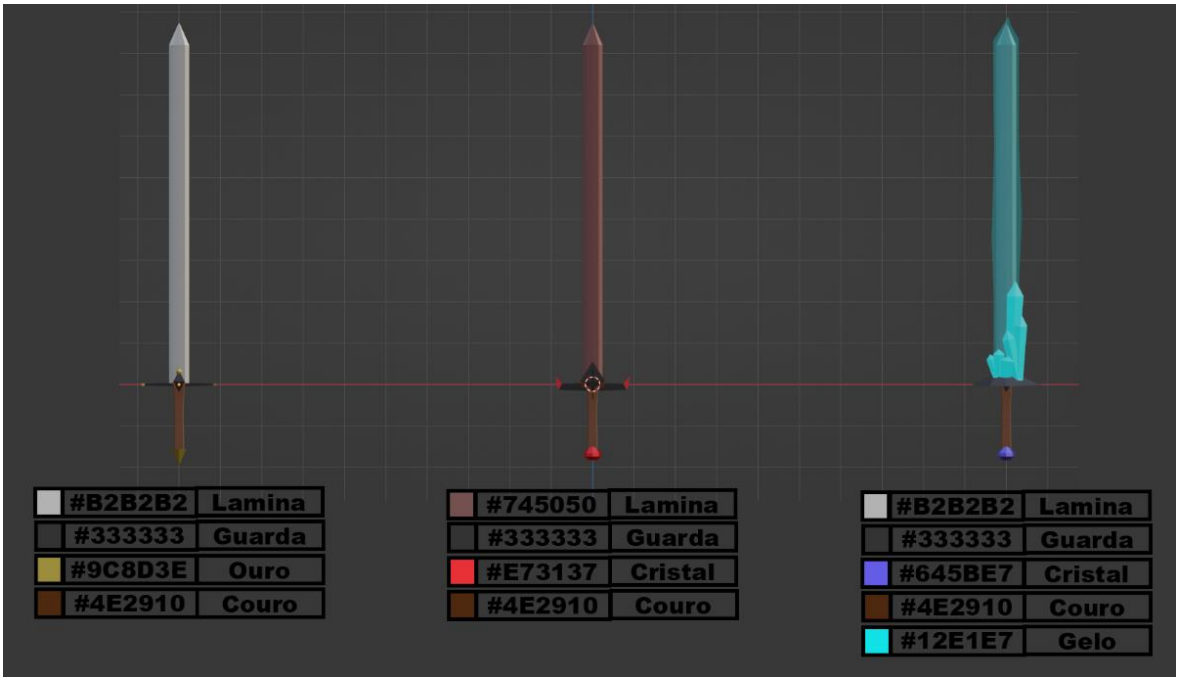


Figura 62 – Espadas

A arma que foi desenvolvida em seguida foi o machado. O machado tem a sua forma e tamanho inspirados nos machados da idade média como é possível observar na Figura 63. O machado tem duas laminas em cada um dos seus lados, preso a um longo cabo de madeira. O seu tamanho, forma e peso proporcionam ataques bastante poderosos, porém mais lentos que as restantes armas. As cores usadas são também inspiradas na idade média, predominando o cinza no metal da lamina, ouro para representar o centro e castanho para representar a madeira do cabo. Tal como a espada o machado também contém duas variações quase idênticas ao machado normal, mudando apenas a forma do centro e algumas cores (Figura 63). O machado de fogo tem pequenos detalhes a vermelho na sua lâmina para representar o fogo e sendo que é construído também com cristais, o mesmo contém um cristal de fogo no seu topo. O seu efeito de fogo será mostrado mais à frente. O machado de gelo tem as mesmas cores que o machado normal, porém a sua lâmina é revestida de gelo e cristais de gelo. Por ser construído com cristais, o mesmo contém também um cristal de gelo no seu topo. Estas variações e as suas cores são possíveis de observar na Figura 63.

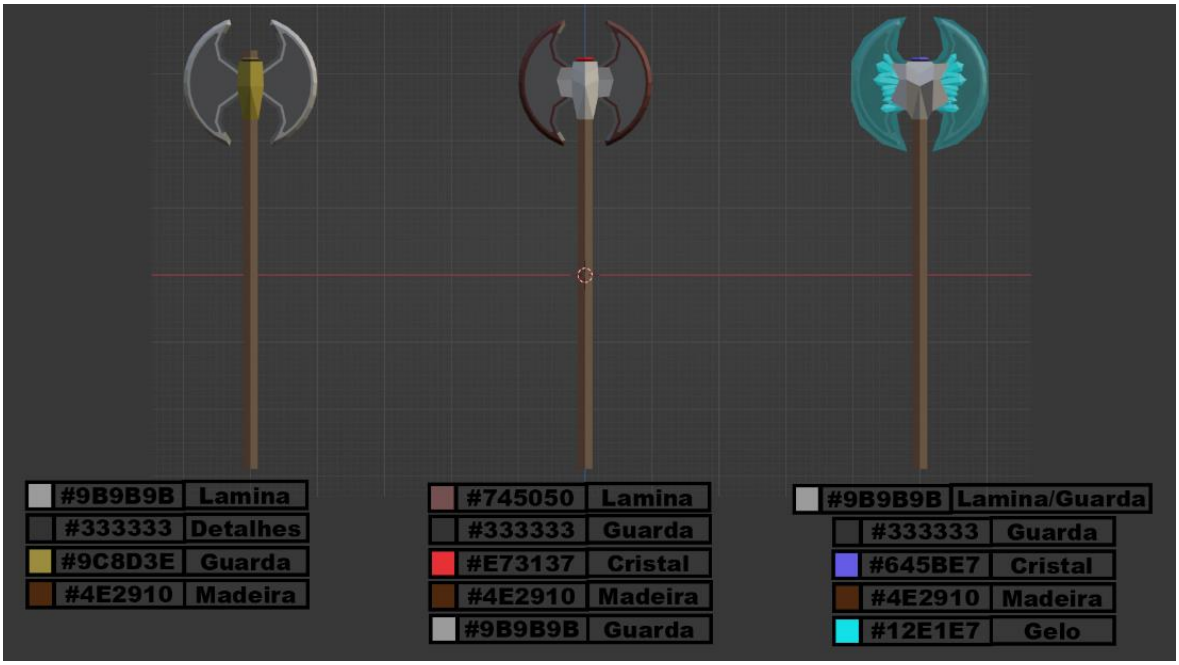


Figura 63 – Machados

Por último foram desenvolvidas as facas. As facas são inspiradas nas facas da idade média, tanto em tamanho como em cores usadas para as desenvolver. As facas têm um tamanho reduzido o que resulta em ataques com menos dano, porém a sua velocidade de ataque é bastante superior às restantes armas. As cores desta arma são o cinza para a lâmina, castanho para o punho em representação do couro usado e dourado para a sua guarda como é possível observar na Figura 64. Tal como as armas anteriormente descritas, também estas contêm duas variações, sendo as facas de fogo e gelo. Estas variações têm o mesmo tamanho e formato que as facas normais, diferenciando apenas em algumas cores. As facas de fogo têm a sua lâmina com uma cor vermelha para representar o fogo e alguns detalhes em vermelho mais brilhante para representar os cristais usadas para construir a mesma (Figura 64). As facas de gelo contêm a sua lâmina revestida por gelo com uma cor azul e a sua guarda é feita também de cristais de gelo. Tal como a faca de fogo, esta arma contém alguns detalhes em azul mais escuro para representar os cristais de gelo usados para construir a mesma (Figura 64).

Figura 64 – Facas

3.5 Efeitos visuais

Neste subcapítulo serão descritos todos os efeitos visuais usados dentro do jogo. Primeiramente será feita uma explicação de como funcionam os efeitos visuais e de que forma foram feitos. Em seguida será mostrado todos os efeitos visuais desenvolvidos para o jogo juntamente com uma descrição dos mesmos.

Primeiramente um efeito visual é na sua base um conjunto de partículas com várias formas e formatos que dão forma a um efeito visual. Para poder criar essas partículas o software Unity já contém os objetos para ser possível criar um efeito visual. Para poder criar um efeito de partículas basta clicar com o botão direto do rato na área onde se se encontram os objetos presentes na cena (zona A da Figura 65). Em seguida podemos encontrar uma opção chamada “effects” (zona B da Figura 65), depois de selecionada essa opção é possível observar que existem cerca de 4 efeitos, porém o único usado dentro do projeto para criar os vários efeitos foi apenas o “Particle System” (zona C da Figura 65). Ao clicar na opção “Particle System” o Unity vai criar um objeto 3D base para poder ser editado para formar um efeito visual, este objeto é possível de observar na zona D da Figura 65

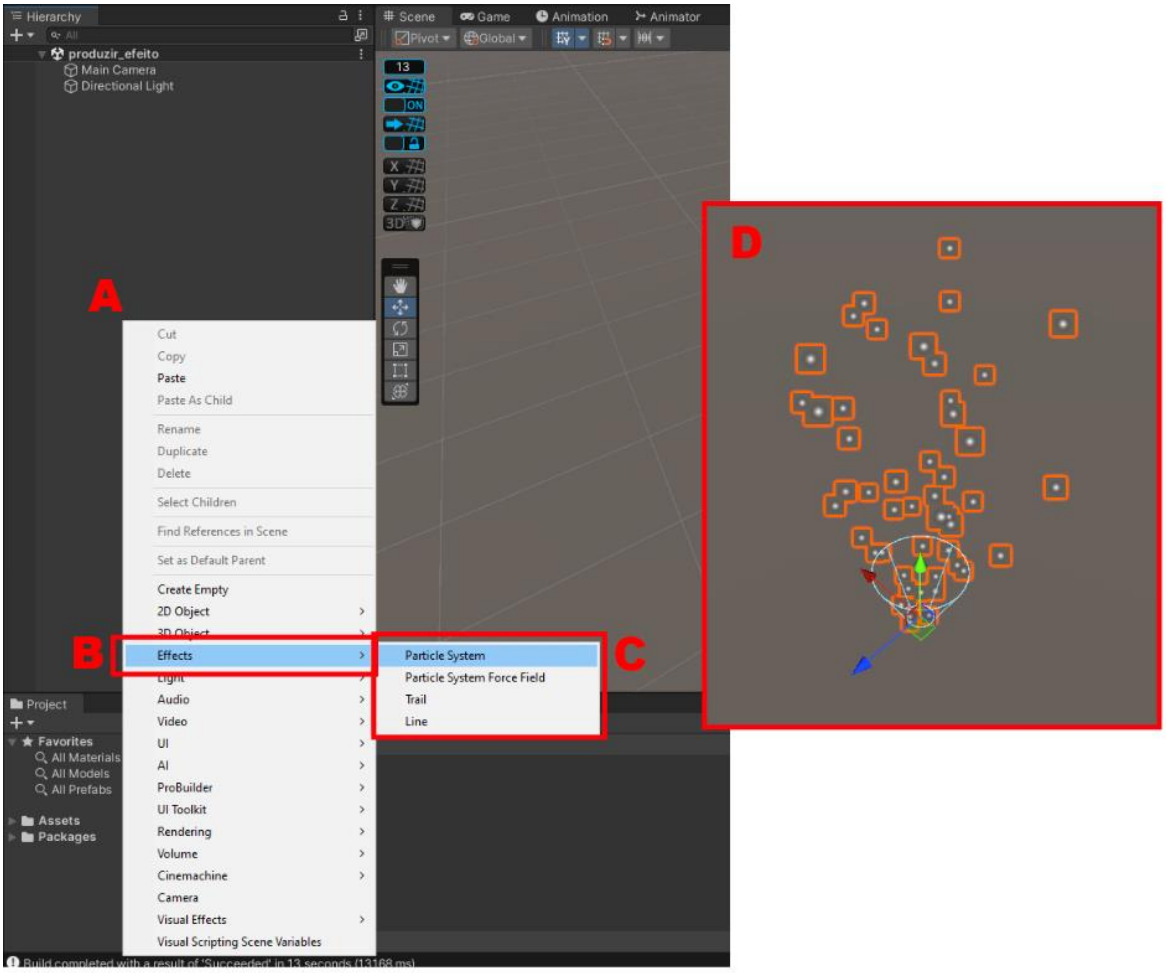


Figura 65 - Criar efeito de partículas do Unity

Após ser criado o objeto, ao clicar no mesmo é possível observar no "*inspector*" as definições base do objeto como é podemos observar na Figura 66. Estas definições são o que dão aspeto e forma ao sistema de partículas do Unity, ou seja, tudo que é alterado, adicionado ou retirado destas definições vai afetar na forma de como as partículas se comportam. Na Figura 66 é possível observar que o objeto criado já vem com algumas definições selecionadas, sendo elas "Emission" (zona B), "Shape" (zona C) e "Renderer" (zona G). Porém para a criação dos efeitos deste projeto foram usadas opções adicionais, sendo elas "Color over Lifetime" (zona D), "Size over Lifetime" (zona E) e por último "Trails" (zona F). A zona A da Figura 66 trata-se das propriedades principais da partícula quando estas são geradas.

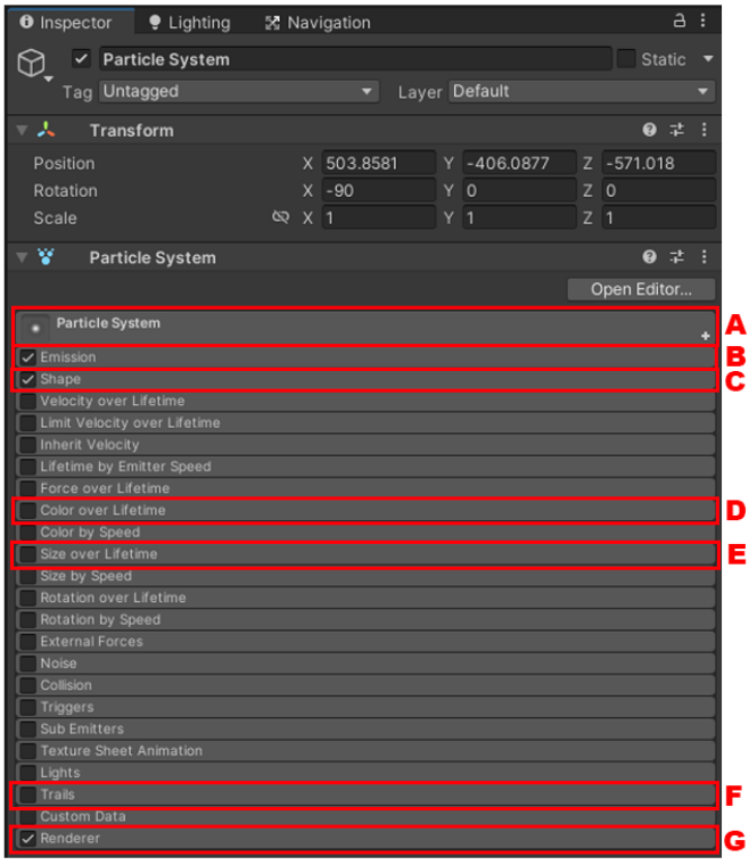


Figura 66 - Definições base do objeto de partículas

Quando clicado na zona A da Figura 66 é possível observar o submenu mostrado na Figura 67. Este mostra todas as propriedades que são possíveis de alterar para modificar as partículas geradas pelo sistema de partículas. As propriedades podem ser separadas em quarto grupos sendo estes: Grupo A, que trata do tempo da partícula, ou seja, quanto tempo vai ter o efeito, se é um *loop*, se tem algum *delay* e quanto tempo de vida vai ter cada partícula; Grupo B trata das propriedades físicas da partícula gerada, ou seja, trata da sua velocidade, tamanho, rotação e cor; Grupo C trata das propriedades de simulação das partículas, ou seja, trata da sua gravidade quando é gerada ou se é simulada em relação ao objeto ou ao mundo de jogo. Por fim grupo D que trata das propriedades do sistema de partículas, , por exemplo, se o objeto é destruído quando este acaba, se é só desativado ou se não faz nada, este grupo pode também limitar o número de partículas total que podem estar presentes no mundo de jogo ou até mesmo se o efeito é ativado ou não quando este é criado.

Figura 67 - "Particle System" propriedades

Ao clicar na zona B da Figura 66 aparece o menu de propriedades presente na Figura 68. Este menu diz respeito ao número de partículas que é criado dentro do ciclo de vida do sistema de partículas. A zona A da Figura 68 altera o número de partículas que é criada a cada 1 segundo, ou seja, a cada 1 segundo vai libertar cerca de 10 partículas em sequência. A zona B da Figura 68 altera o número de partículas que são libertadas num determinado ponto no ciclo de vida do sistema de partículas, neste exemplo o sistema de partículas vai libertar cerca de 30 partículas no segundo 0. Para poder adicionar mais ciclos ou retirar basta clicar o mais ou menos na zona C da Figura 68.

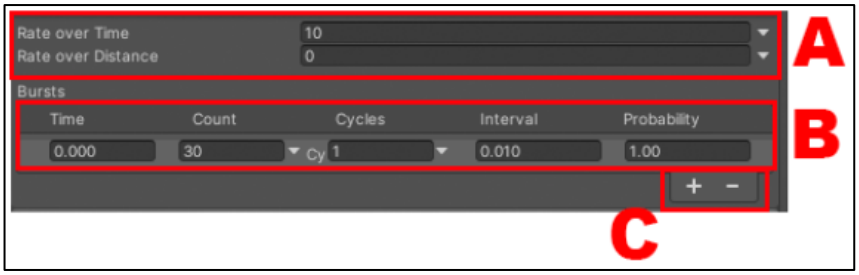


Figura 68 - "Shape" propriedades

Ao clicar na zona C da Figura 66 irá a aparecer o menu de propriedades da Figura 69. Este menu altera a forma em que as partículas vão ser direcionadas. Na zona A da Figura 69 é possível observar o submenu que aparece quando clicamos e é possível ver as diferentes formas que as partículas podem ser direcionadas. Quando o objeto é criado o objeto que está selecionado é o cone. As restantes propriedades alteram o ângulo, tamanho ou a rotação da forma. O submenu B altera a forma de como as partículas são criadas em torno da forma e o submenu C altera a localização de onde são criadas as partículas.

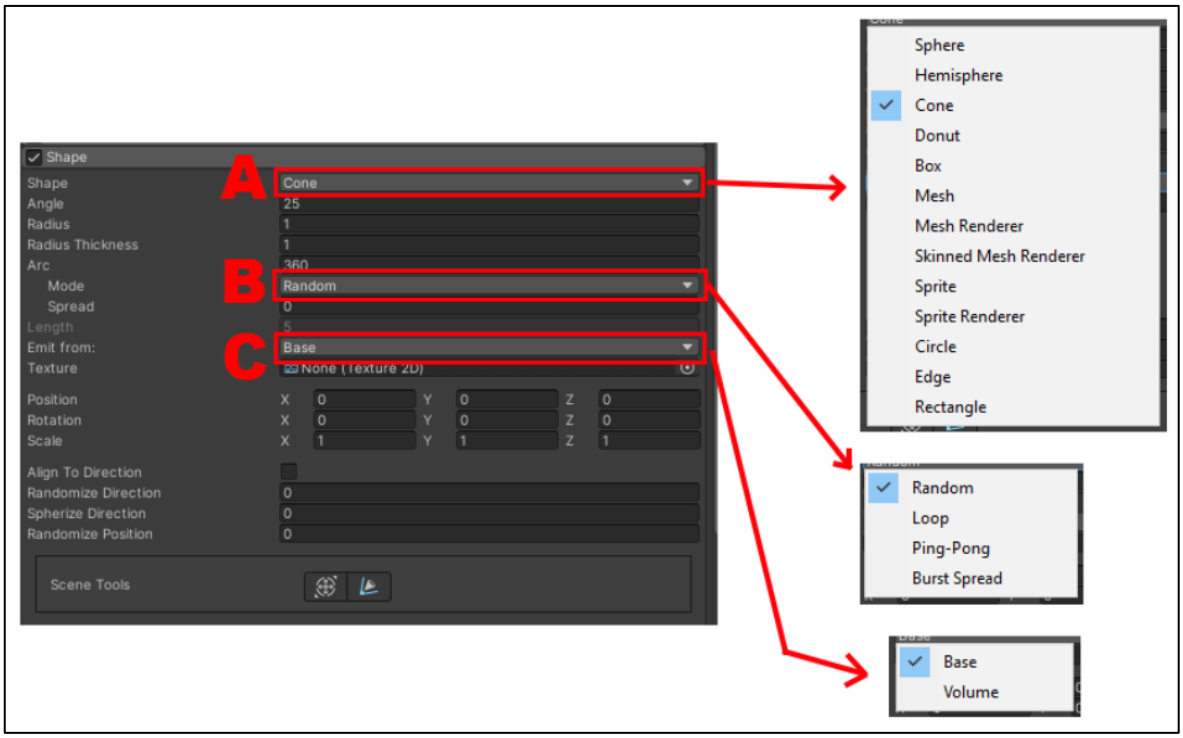


Figura 69 - "Shape" propriedades

Quando clicado na zona D da Figura 66 é mostrado o menu representado na Figura 70. Este menu vai alterar a cor e transparência da partícula ao longo do seu tempo de vida. Ao clicar na zona A da Figura 70 irá aparecer a janela representada por um F. Nesta janela é possível alterar a cor e transparência desejada para as partículas, clicando em B será possível alterar a transparência através das *keys*, e para alterar a cor é possível através das *keys* em C alterando a cor em D, para adicionar mais keys é possível fazê-lo através do botão esquerdo do rato e para eliminar basta clicar com o botão direto. A barra representada a branco diz respeito ao ciclo de vida da partícula sendo o início à esquerda e o fim à direta. A zona E representa algumas configurações já salvas para poderem usadas em outros efeitos futuros.

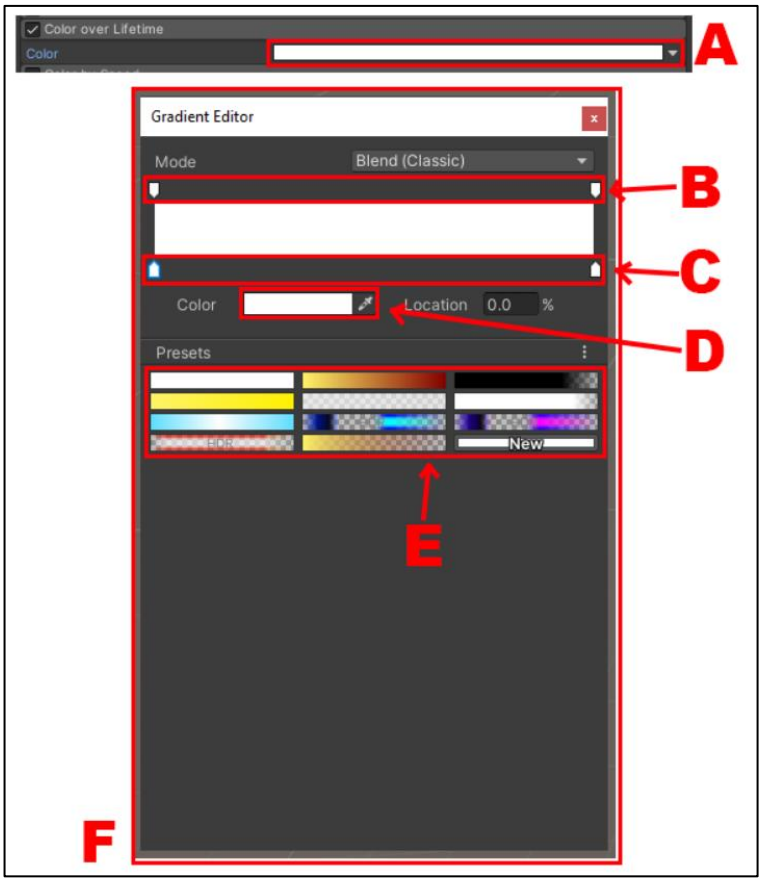


Figura 70 - "Color over Lifetime" propriedades

Quando clicado na zona E da Figura 66 aparece o menu para poder alterar o tamanho das partículas durante o seu ciclo de vida representado na Figura 71. Ao clicar na zona A da Figura 71 aparece o submenu (B) que altera o tamanho da partícula durante o seu ciclo de vida e para isso acontecer basta mover as *keys* presentes em C para a posição desejada, sendo que a esquerda representa o início do ciclo de vida e a direita o seu fim. Para adicionar mais *keys* basta clicar com o botão esquerdo em cima da linha vermelha da zona C. Na zona D é possível observar já algumas configurações salvas para poderem ser usadas futuramente.

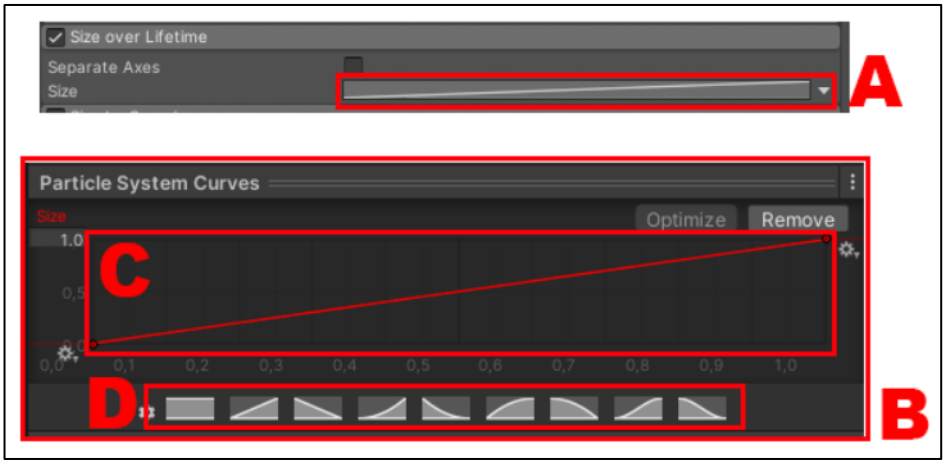


Figura 71 - "Size over LifeTime" propriedades

Ao clicar na zona F da Figura 66 é possível observar o menu presenta na Figura 72. Este menu ao ser selecionado adiciona rastos às partículas. No menu da Figura 72 é possível observar duas propriedades destacadas, sendo esta a "Lifetime" (A) que representa o ciclo de vida do rasto e por fim a "Color over Lifetime" (B) que determina a cor do rasto ao longo do seu ciclo de vida, esta propriedade exibe uma janela igual à imagem Figura 70 zona F. Estas duas propriedades foram as únicas usadas e editadas para o presente projeto.

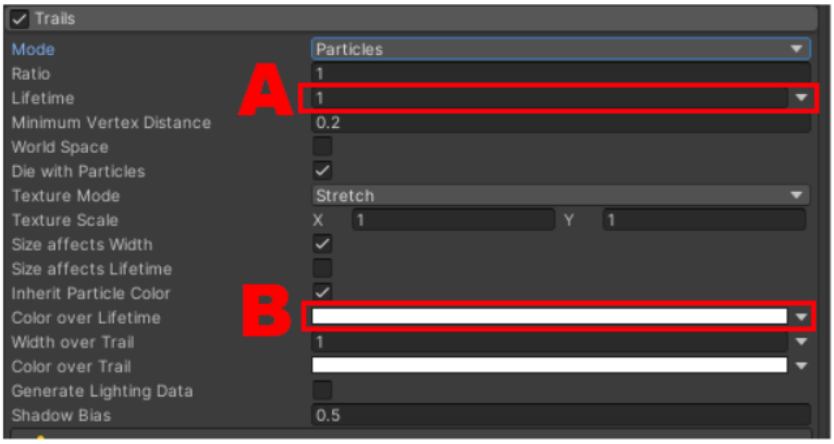


Figura 72 - "Trails" propriedades

Ao clicar na zona G da Figura 66 é possível observar o menu presenta na Figura 73. As principais propriedades usadas no projeto são as destacadas, sendo a primeira representada em A, que ao ser clicada aparece o submenu representado em B, este submenu apresenta as diferentes opções para renderizar a forma de como as partículas geradas são apresentadas, ou seja, se são apresentadas consoante a visão do jogador, se são na posição horizontal, esticadas, estes são alguns exemplos das diversas opções. A propriedade destacada em C trata-se do local onde se coloca a textura desejada para as partículas geradas. A propriedade destacada em D é o local onde se coloca o material desejado para os rastos quando estes estão ativos, caso não estejam selecionados, esta propriedade não aparece. Por último a propriedade destacada em E determina o tamanho máximo das partículas que são geradas.

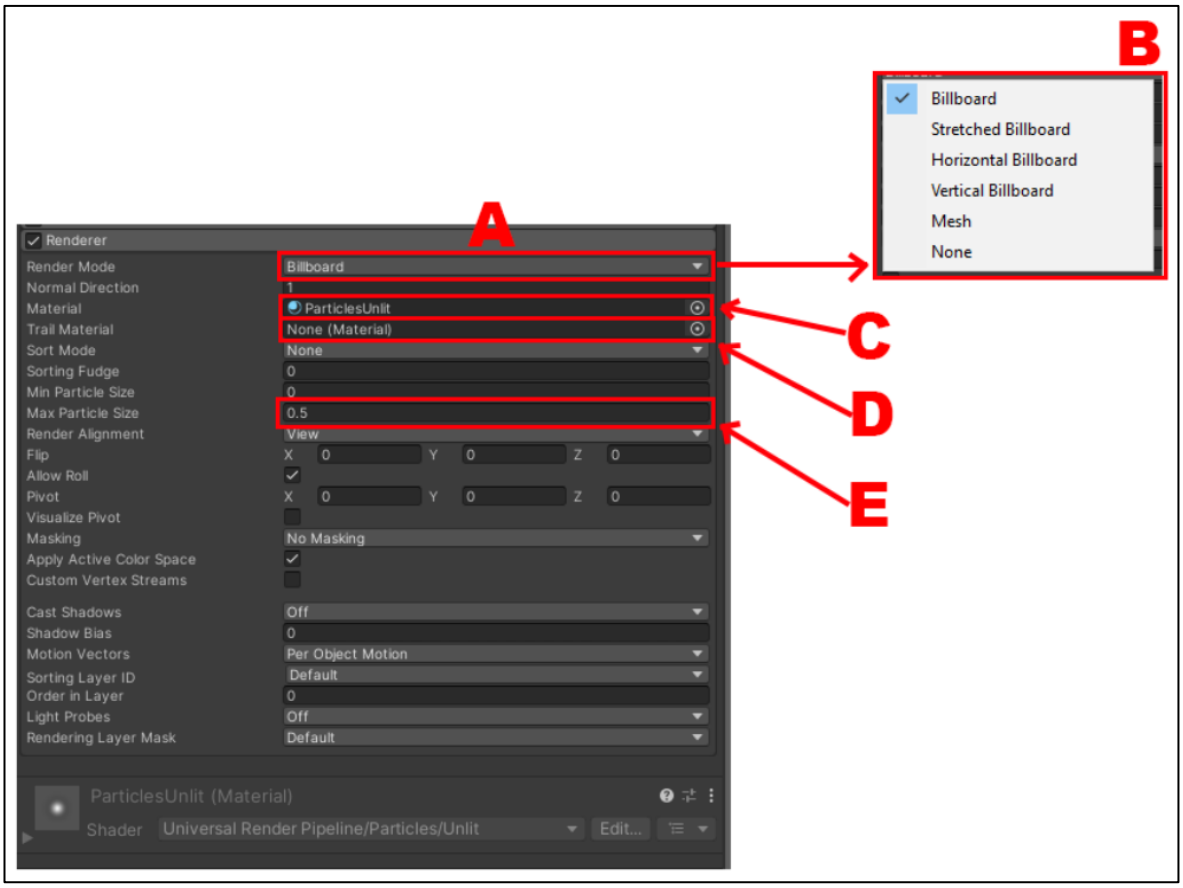


Figura 73 - "Renderer" propriedades

Na Figura 74 é possível observar o efeito visual desenvolvido para quando um inimigo bate no personagem principal com a sua arma. Porém este efeito só aparece quando o inimigo inflige dano no personagem, caso contrário, não acontece nada. Este efeito pode ser visto nos inimigos Crab, Dark Knight e Javeling (em ambos no ataque 1)

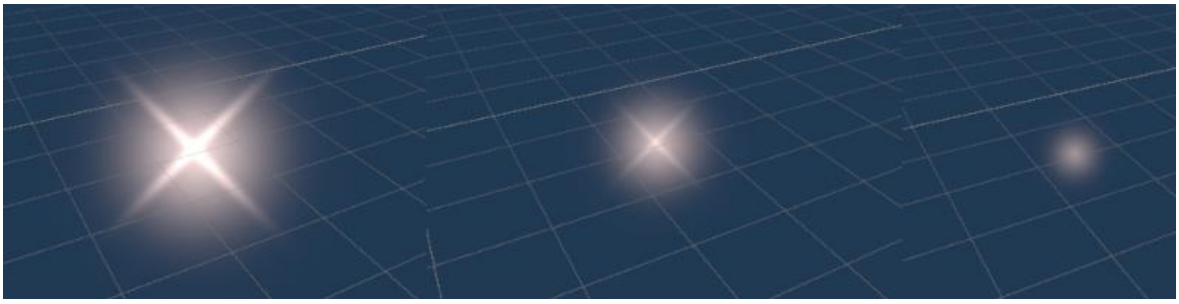


Figura 74 - Efeito de bater

Quando um inimigo morre, este liberta um efeito de morte, sendo possível observar na Figura 75. Este efeito está presente em todos os inimigos que o jogador tem de enfrentar, à exceção do inimigo final que contém um efeito de morte diferente.

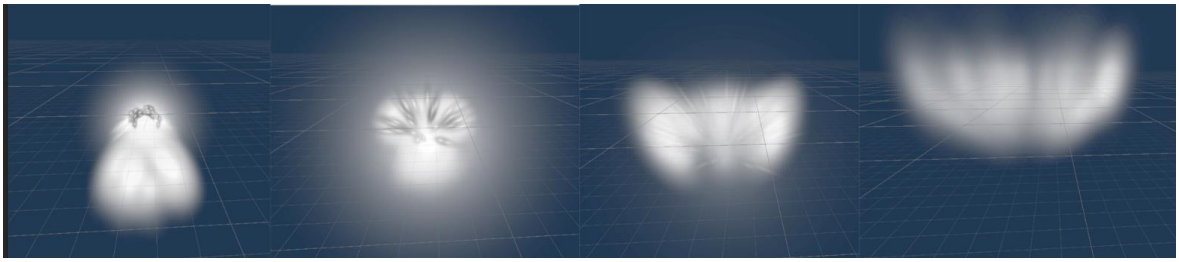


Figura 75 - Efeito de morte de um inimigo

Um dos inimigos que mais difere dos outros é o Bombist com a sua bomba gigante que traz na sua mochila. Esta bomba quando ativada rebenta ao fim de um tempo (nomeadamente 15 segundos) o que faz formar uma enorme explosão. Esta explosão é possível observar na Figura 76

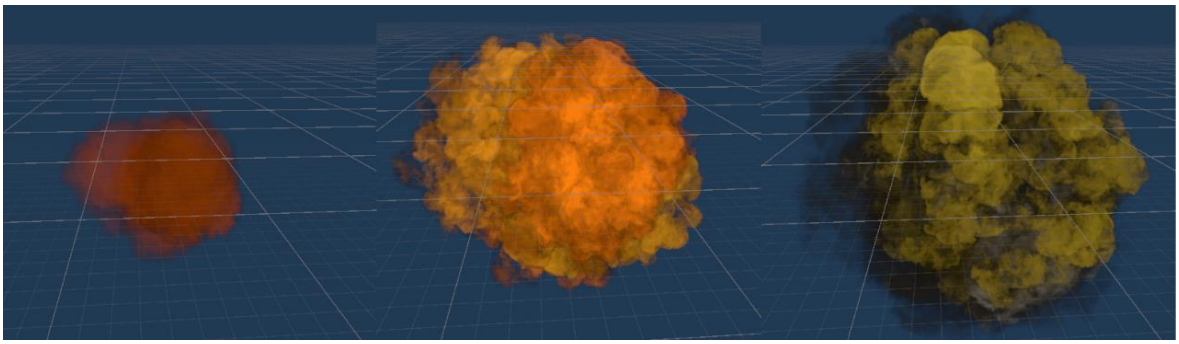


Figura 76 - Efeito de explosão

O efeito usado no ataque 2 do inimigo Horse Knight é possível observar na parte superior da Figura 77 em cor laranja. Este efeito tem mais duas variações sendo esta a rosa, pertencente ao ataque 3 da segunda fase do Inimigo final e a variação azul que pertence ao impacto da bola de energia do inimigo Mage.

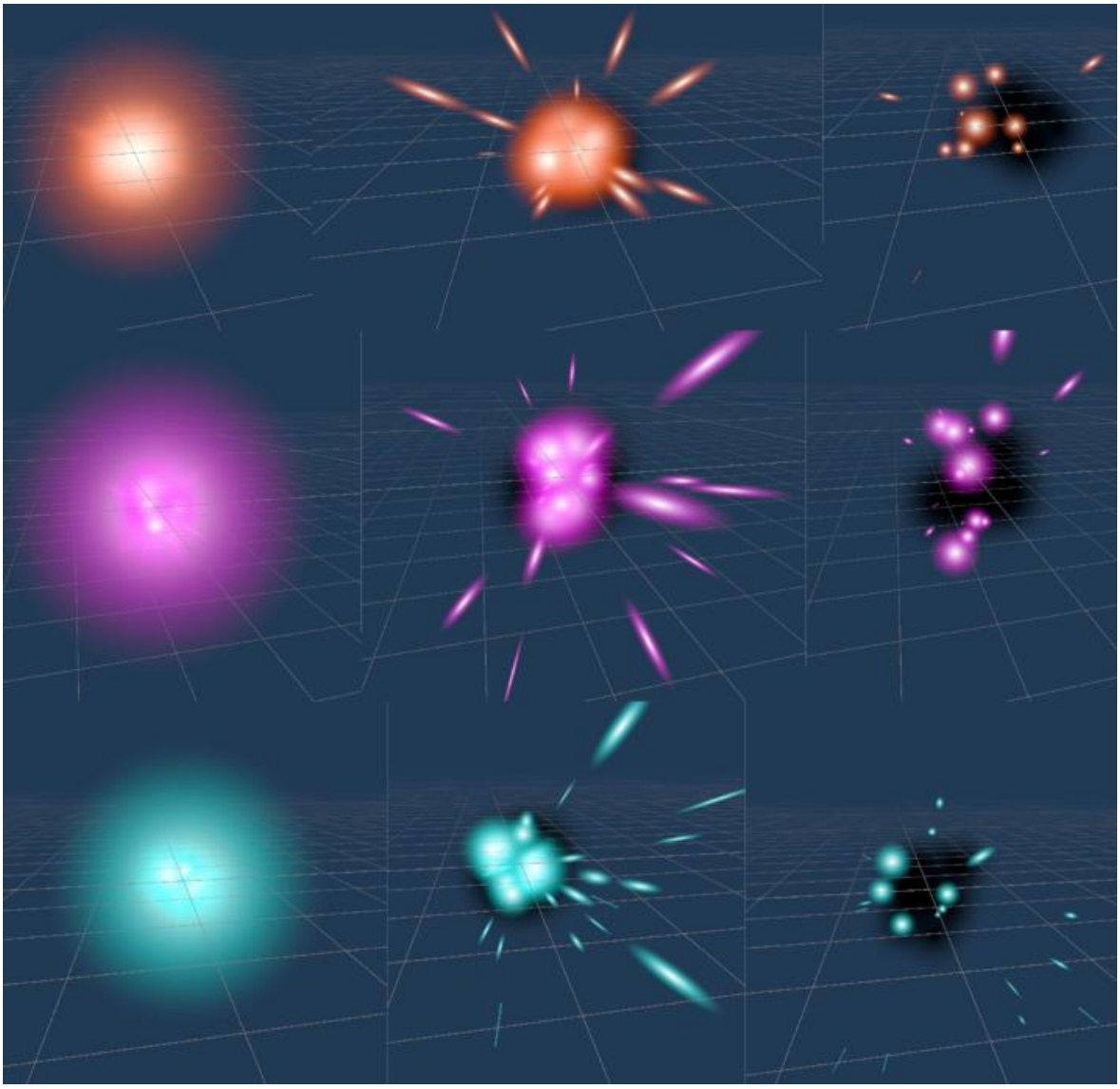


Figura 77 - Efeito de impacto

Dentro das diversas armas de jogo, algumas delas têm o poder de fogo o que faz com que após atingidos, coloque os inimigos em fogo a dar ideia que estão a arder levando dano. Este efeito é possível de observar na Figura 78.

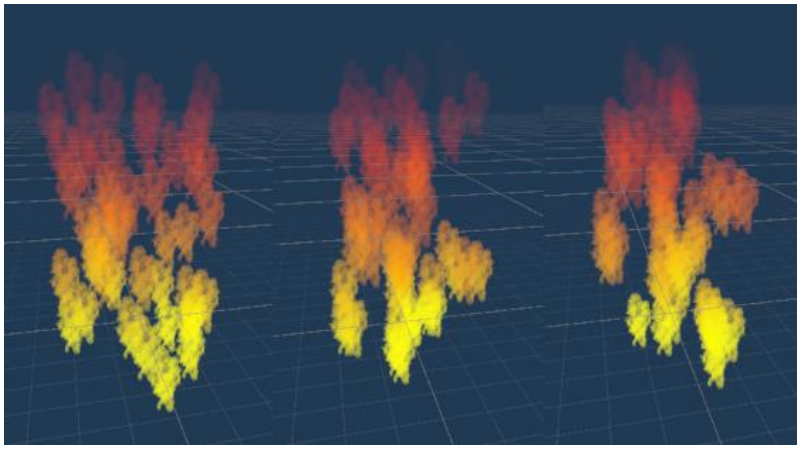


Figura 78 - Efeito de fogo

O inimigo Javeling contém uma arma poderosa, um machado imenso capaz de produzir bastante dano. Quando este inimigo produz o seu segundo ataque bate com o seu machado no chão fazendo com que causa um grande impacto. O efeito visual de impacto no chão é possível de observar na Figura 79.

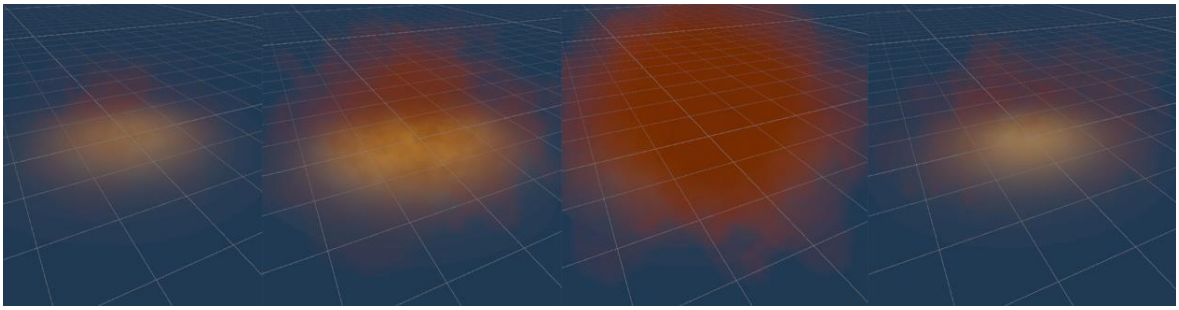


Figura 79 - Efeito de impacto no chão

Na Figura 80 é possível observar o efeito visual do impacto causado pelo inimigo Crab quando este produz o seu segundo ataque. Quando este inimigo bate com os seus martelos um no outro produz uma onda de choque.

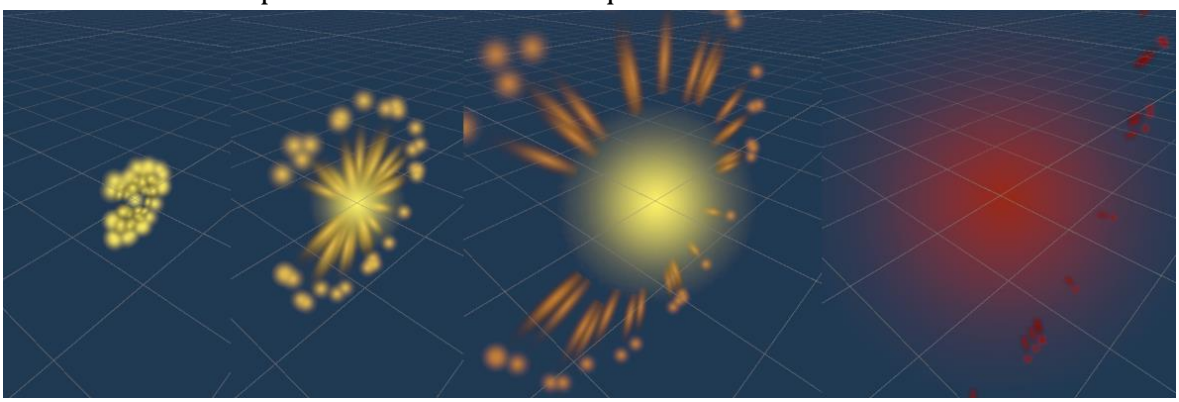


Figura 80 - Efeito de impacto 2

O inimigo Mage quando vê que o personagem principal se aproxima demasiado dele, o mesmo começa lança um poder magico à sua volta. O poder magico lançado pelo ataque 2 deste inimigo é possível de observar na Figura 81.

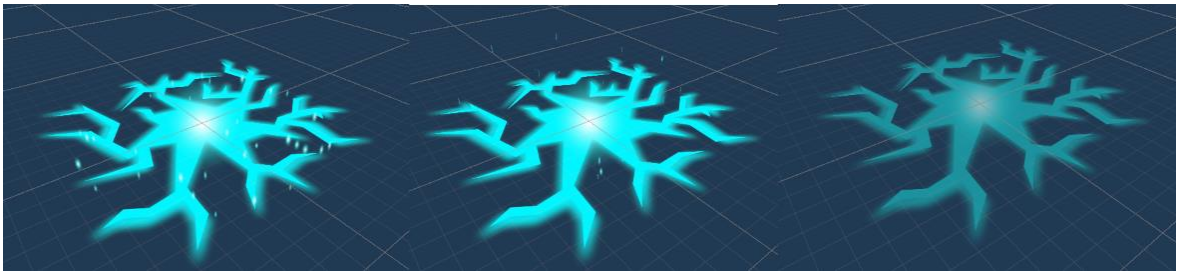


Figura 81 - Efeito magico do Mage

Além da explosão criada pela bomba do Bombista, antes disso existe outro efeito presente na animação, efeito esse que inicia quando a bomba é ativada, onde começam a sair faíscas do rastilho da bomba, este efeito de faíscas é possível de observar na Figura 82.

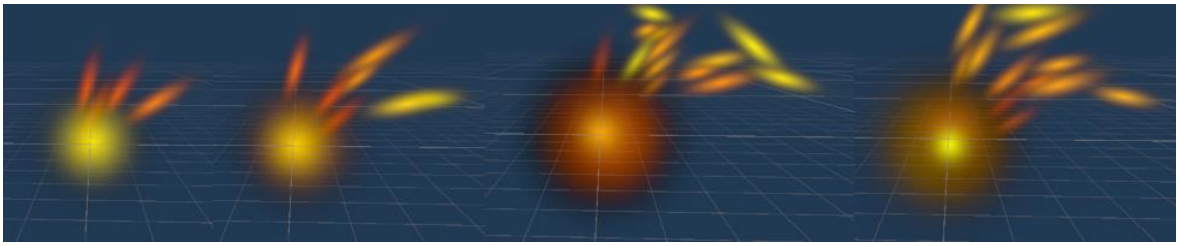


Figura 82 - Efeito de faíscas

O inimigo Horse knight utiliza a sua arma que consiste numa lança bastante comprida para poder executar os seus ataques. Um dos ataques consiste em, com a ajuda do seu cavalo, ganhar balanço e produzir um ataque poderoso capaz de infligir bastante dano no personagem principal. Este ataque produz também um efeito especial na sua lança, este efeito é possível de observar na Figura 83.

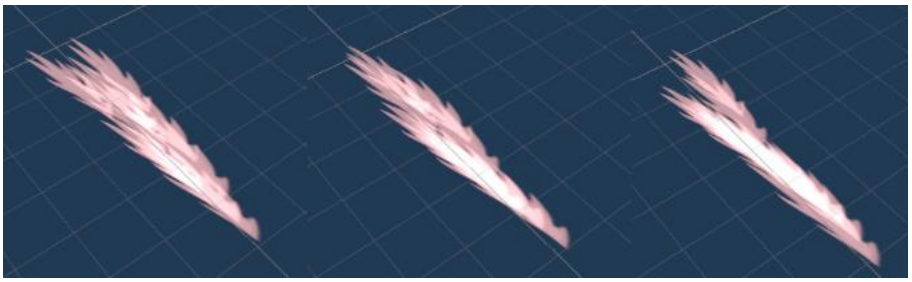


Figura 83 - Efeito lança

O inimigo final contém um total de 6 ataques para poder atacar o personagem principal. Um destes ataques acontece durante a sua segunda fase onde o mesmo emite um laser poderoso. Este efeito visual é possível de ser observado na Figura 84

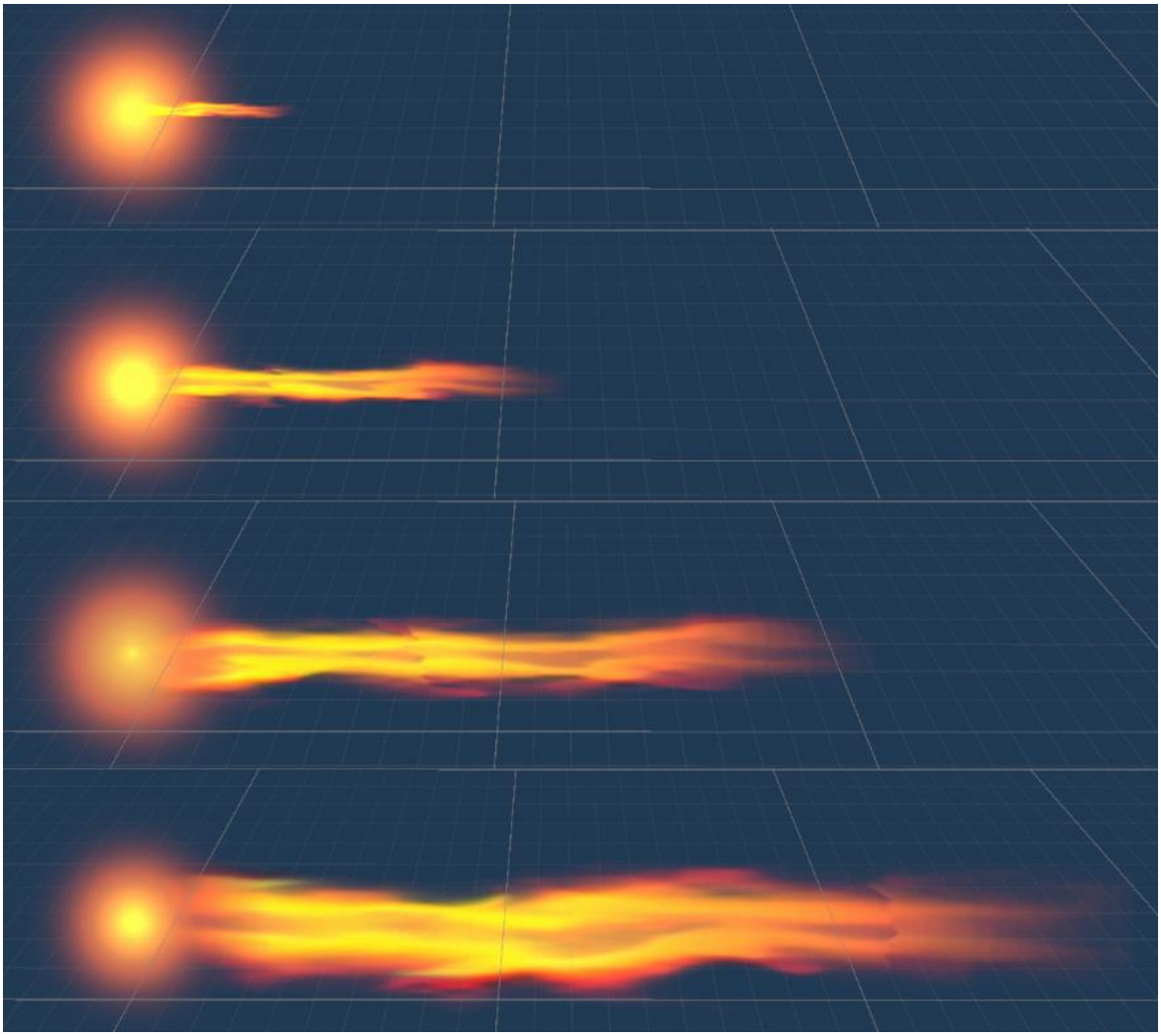


Figura 84 - Efeito de Laser

Quando o laser acima mencionado produz dano ao personagem principal, é ativado um efeito que é possível observar na Figura 85. Este efeito indica que o personagem principal está a receber dano do laser, caso este saía do alcance do laser, este efeito desaparece.

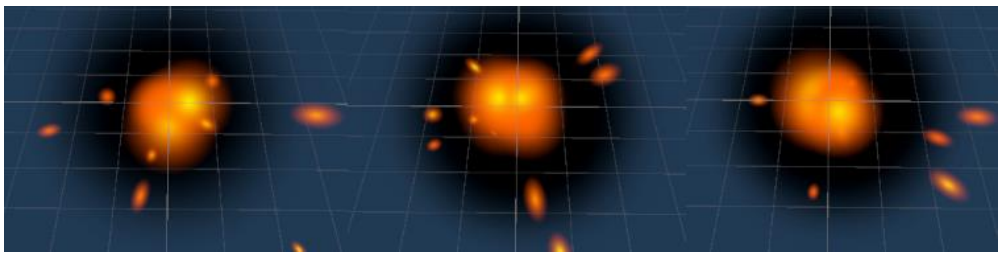


Figura 85 - Efeito dano do laser

Um dos maiores ataques que o inimigo final tem, é o ataque de fogo que ele é capaz de produzir. Este ataque acontece durante a primeira fase de luta quando o mesmo está sentado no seu trono. Este efeito de ataque de fogo é possível de observar na Figura 86.

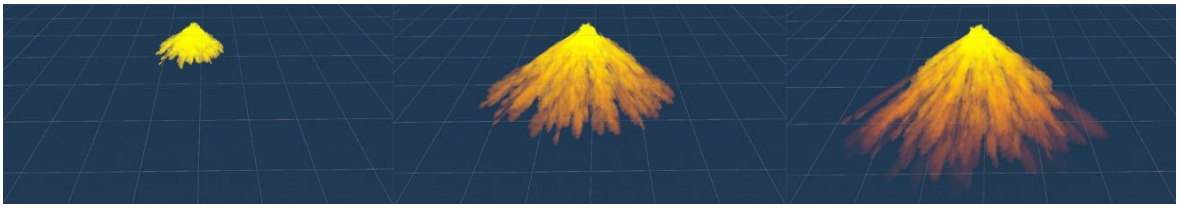


Figura 86 - Efeito ataque de fogo

Outro dos ataques capaz de produzir também bastante dano, é o ataque do punho de força durante a segunda fase de luta contra o inimigo final. Este ataque é possível de ser identificado pois o punho direto de Baelzor fica coberto de um fogo roxo. Este fogo roxo é possível de observar na Figura 87.

Figura 87 - Efeito de fogo roxo

Durante a primeira fase de luta contra o inimigo final existe um ataque que faz com que surja um espinho de gelo. Este ataque emite pedaços de gelo quando o espinho surge do chão. Estas partículas são possíveis de observar na Figura 88.

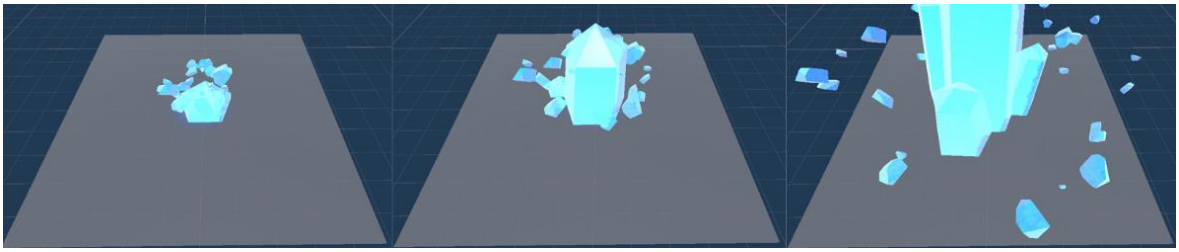


Figura 88 - Efeito de partículas de gelo

Também durante a primeira fase de luta contra o inimigo final, o mesmo lança um ataque em área produzindo diversos relâmpagos que caiem de forma aleatória. O efeito de relâmpago é possível de observar na Figura 89.

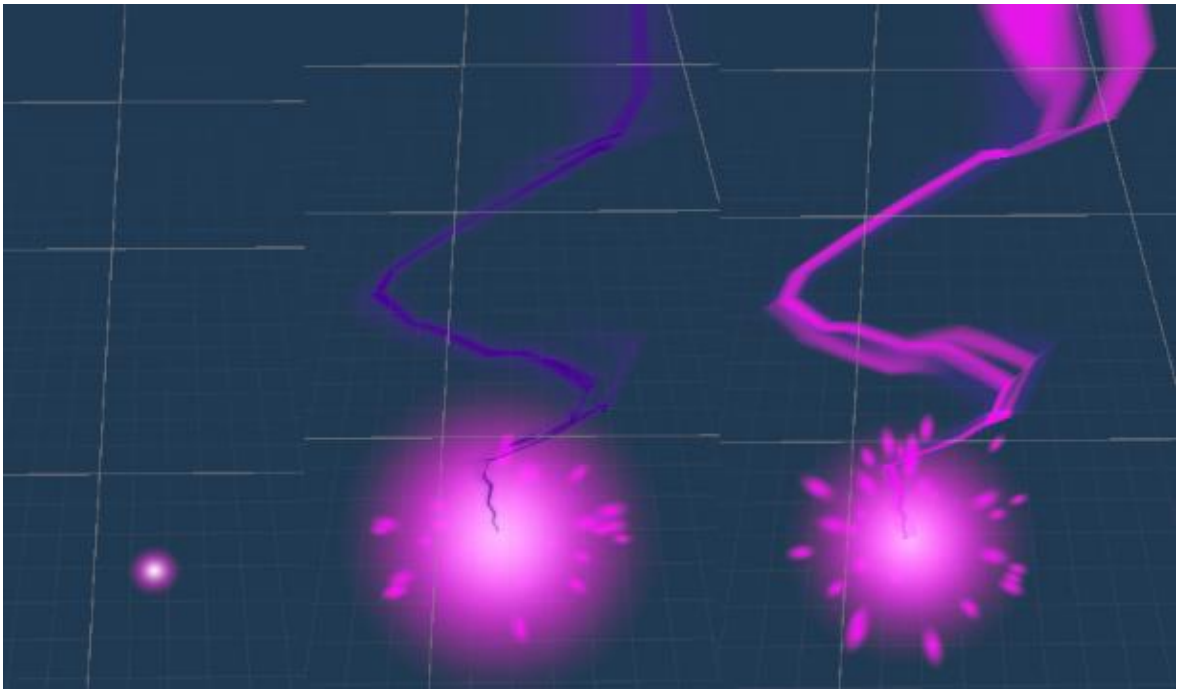


Figura 89 - Efeito de relâmpago

O último dos seis ataques presentes durante a luta contra o inimigo final é possível observar na Figura 90. Este ataque acontece durante segunda fase de luta onde o inimigo lança um campo de força na direção do personagem principal.

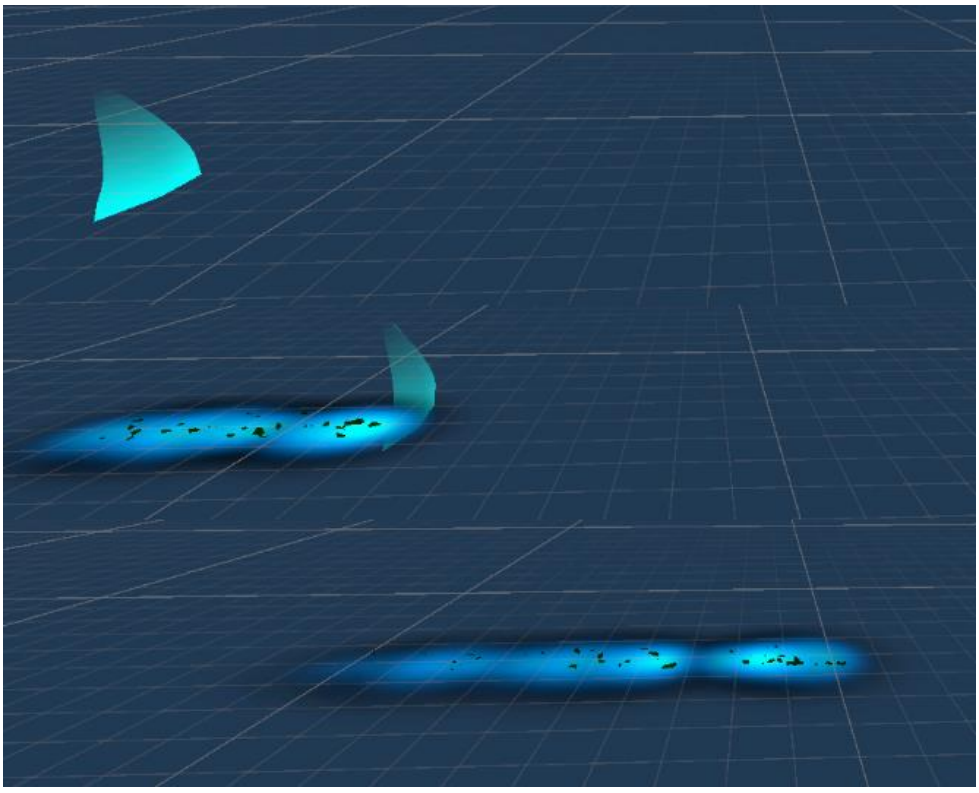


Figura 90 - Efeito de campo de força

No final da luta contra o Bealzor, quando este é derrotado, o mesmo liberta um efeito do seu peito. Este efeito é possível observar na Figura 91 e representa as almas

que foram roubadas às pessoas do reino a serem todas libertadas, concluindo assim o grande objetivo do jogo

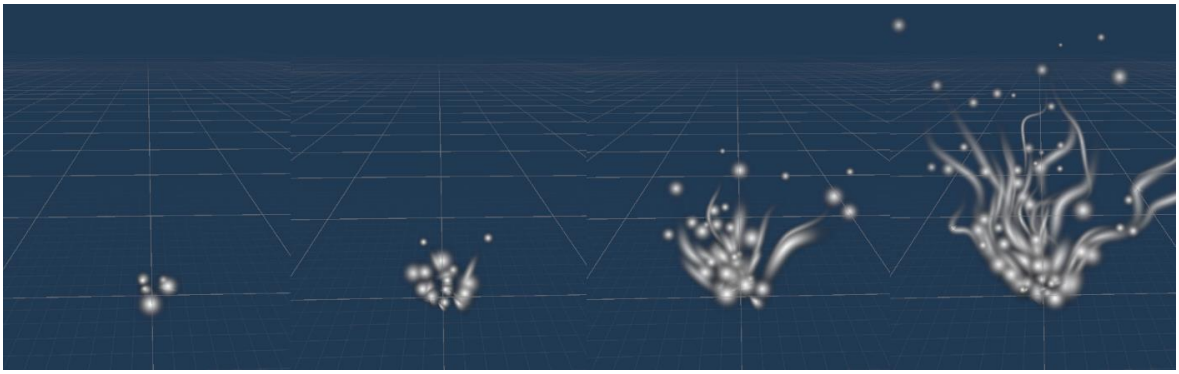


Figura 91 - Efeito de almas

O personagem principal consegue executar vários ataques especiais com as suas armas. Um desses ataques é o ataque giratório presente em todas as armas, este ataque produz uma espiral do poder da arma em uso, ou seja, se for a arma normal produz um efeito branco, se for arma de fogo produz uma espiral de fogo e se tiver a de gelo produz uma espiral de gelo. Este efeito é possível observar na Figura 92 onde estão presentes os três tipos de poderes, normal, fogo e gelo.

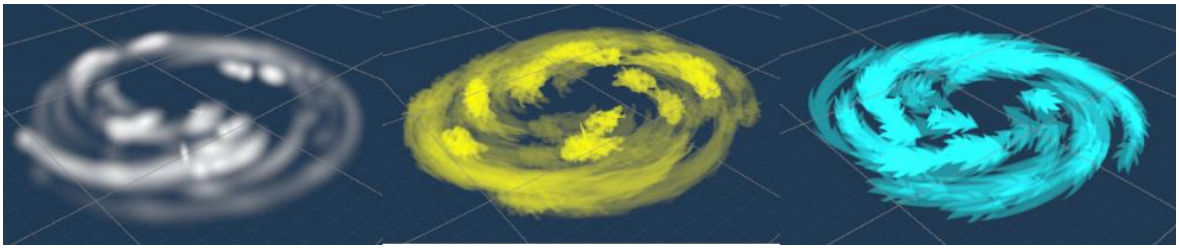


Figura 92 - Efeito de espiral

Outro ataque que é comum ao todas as armas é o ataque final que produz um efeito de explosão quando o ataque atinge o inimigo. Este ataque tal como o ataque de espiral também depende do poder da arma que o personagem estiver a usar. Este efeito é possível de observar na Figura 93 com os seus três tipos de poderes.

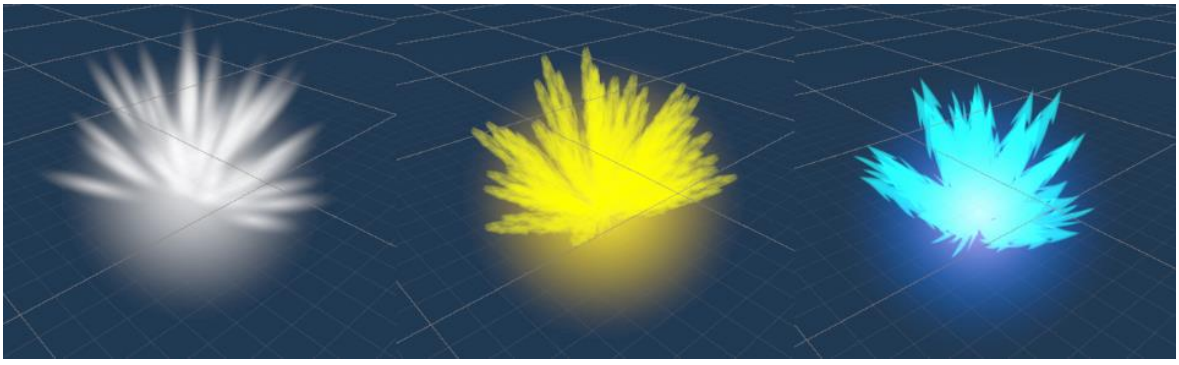


Figura 93 - Efeito do ataque final

Apesar de dois dos ataques especiais serem iguais em todas as armas, existe um que é único em cada arma, o segundo ataque das 3 armas.

O segundo ataque da espada consiste numa espiral que poder que emite dano e empurra os inimigos. Este contém também três tipos de poder dependendo da arma que o personagem esteja a usar (Figura 94).

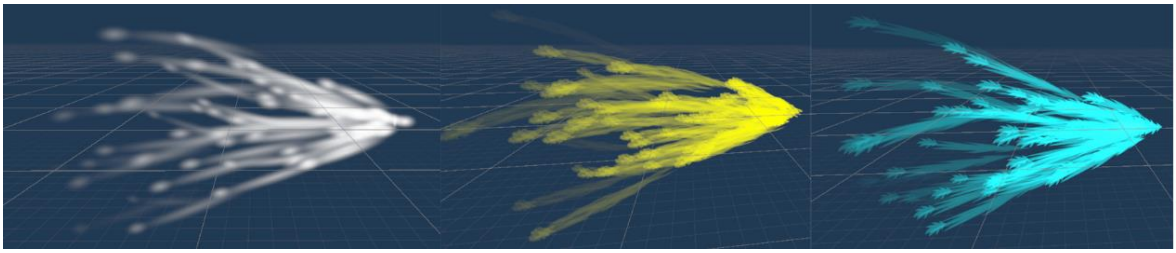


Figura 94 - Efeito de espiral

O segundo ataque das facas, consiste em teletransportar-se para os inimigos mais próximos. Ao executar este ataque é possível observar um efeito de flash/rasto deixado pelo personagem. Este efeito é possível de observar na Figura 95

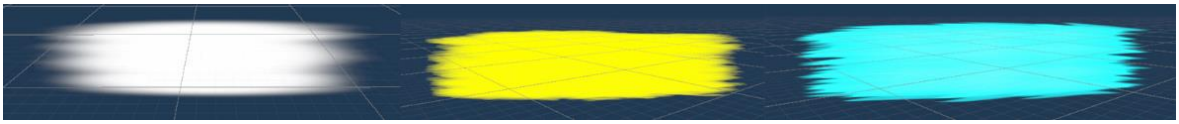


Figura 95 - Efeito de flash

Por fim o segundo ataque do machado consiste na lâmina do machado brilhar com o poder da mesma, este brilho ainda deixa um rasto quando o machado é movido como se pode ver no lado esquerdo da Figura 96. Neste ataque o personagem atinge o chão com o seu machado o que faz com que cause um impacto com o poder do mesmo. Cada impacto é possível de observar do lado direto da Figura 96 com cada tipo de poder.

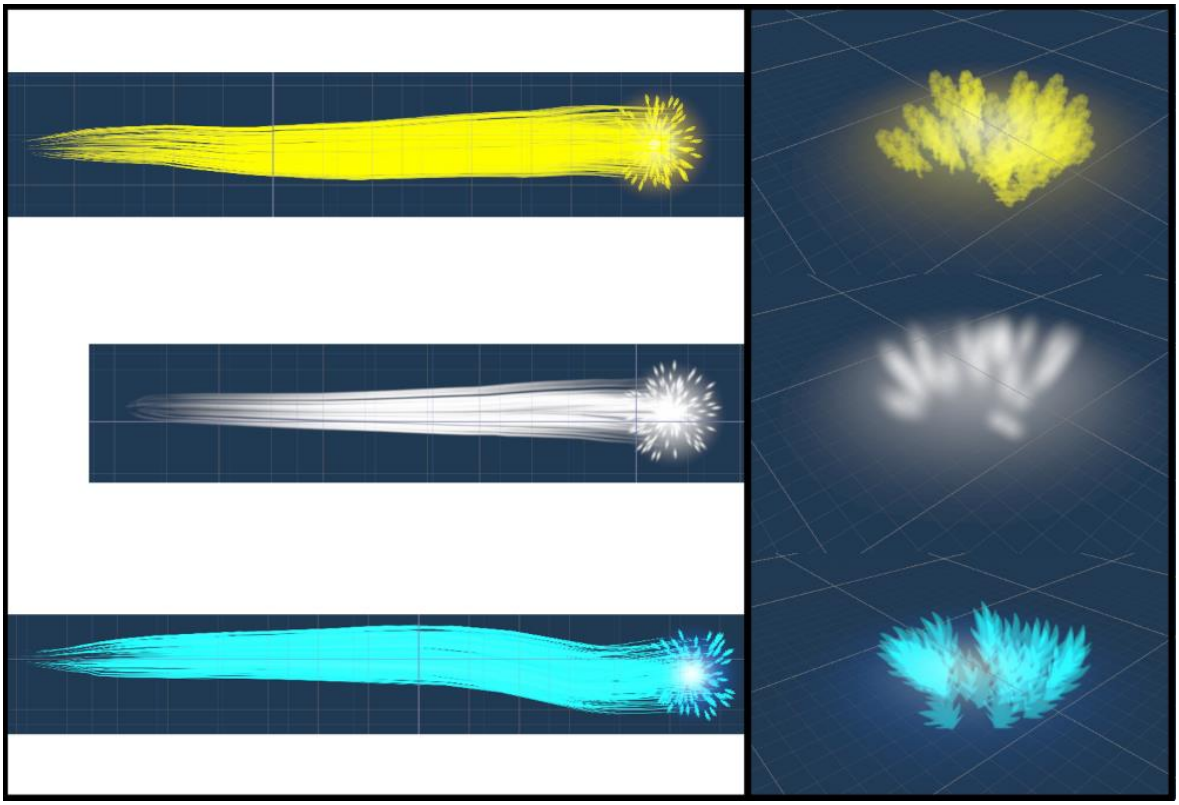


Figura 96 - Efeito impacto no chão 2

Em relação aos efeitos das poções, existem duas poções que causam efeitos no *player* sendo estas a de velocidade e a de dano.

A poção de velocidade deixa o personagem com partículas brilhantes de cor verde e quando este anda emite um rasto iguais às partículas. Este efeito é possível observar na Figura 97

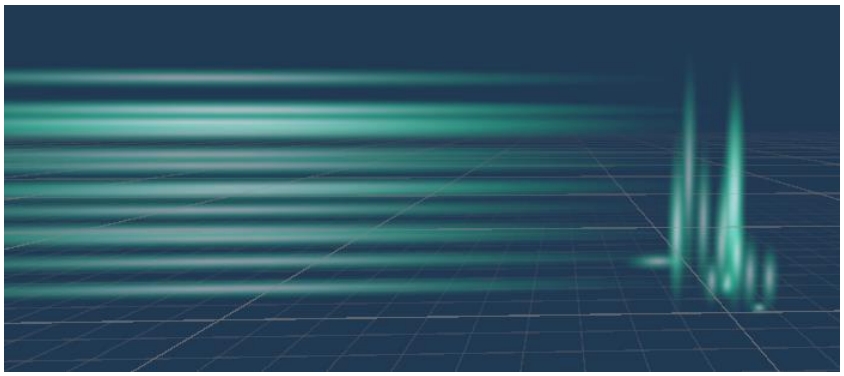


Figura 97 - Efeito de velocidade

Quando tomada a poção de dano, o personagem principal fica rodeado de partículas de cor laranja brilhantes assim como a poção de velocidade. Este efeito é possível observar na Figura 98.

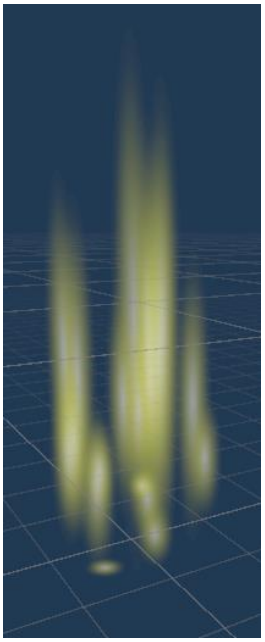


Figura 98 - Efeito e dano extra

Dentro das armas existem algumas que contêm poderes especiais, estes poderes dão às armas propriedades especiais. Nas armas de fogo, estas estão rodeadas de chamas e emitem um rasto de fogo quando movidas. Este efeito de fogo é possível de observar na Figura 99.

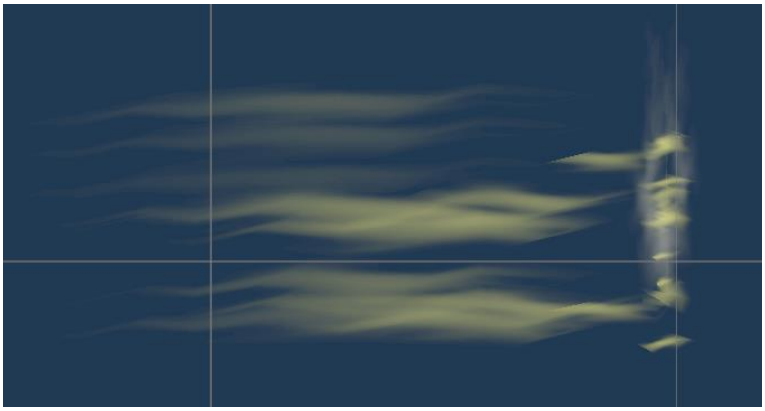


Figura 99 - Efeito de fogo das armas

As armas de gelo por serem revestidas de gelo emitem pequenas partículas brilhantes de gelo. Estas partículas são deixadas para trás sempre que a arma se move deixando um rasto de brilhantes. Este efeito é possível observar na Figura 100.

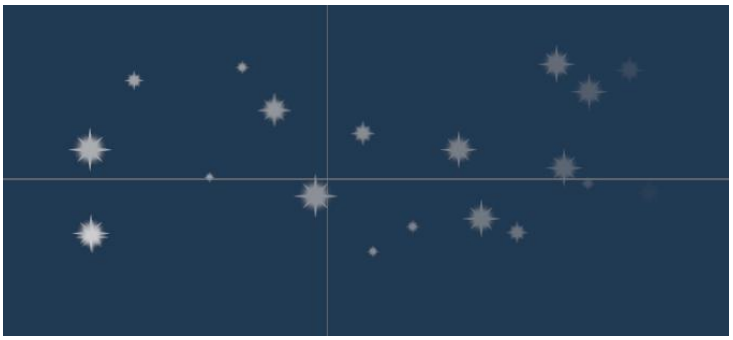


Figura 100 - Efeito de partículas de gelo das armas

3.6 Interface do Utilizador

Neste subcapítulo serão mostradas todas as imagens desenvolvidas para a interface do utilizador. Através do software Krita foi possível desenvolver as imagens presentes dentro do jogo em todos os mapas e menus.

Todos os menus de jogo foram desenvolvidos de forma a serem únicos e diferentes, mas seguindo todos o mesmo tema. Primeiramente foi desenvolvido o título do jogo dando a impressão de que o título foi escrito em um pergaminho. Foi usado um estilo de texto retirado da internet chamado de "Deutsch Gothic" [1]. O título que foi desenhado à mão no software Krita pode ser observado na Figura 101.

Figura 101 - Título do jogo

No mesmo estilo do título de jogo foram desenvolvidos também os botões presentes dentro do jogo. Todos os botões foram desenhados com cerca de três diferentes aspetos, normal, quando passa o rato por cima e para quando o botão é pressionado. Quando está em modo normal o botão tem um fundo preto, quando o rato passa por cima fica destacado com um fundo azul o que lhe traz destaque e por fim quando é clicado o botão fica com um tom mais escuro na totalidade. Estes botões são possíveis de observar na Figura 102.

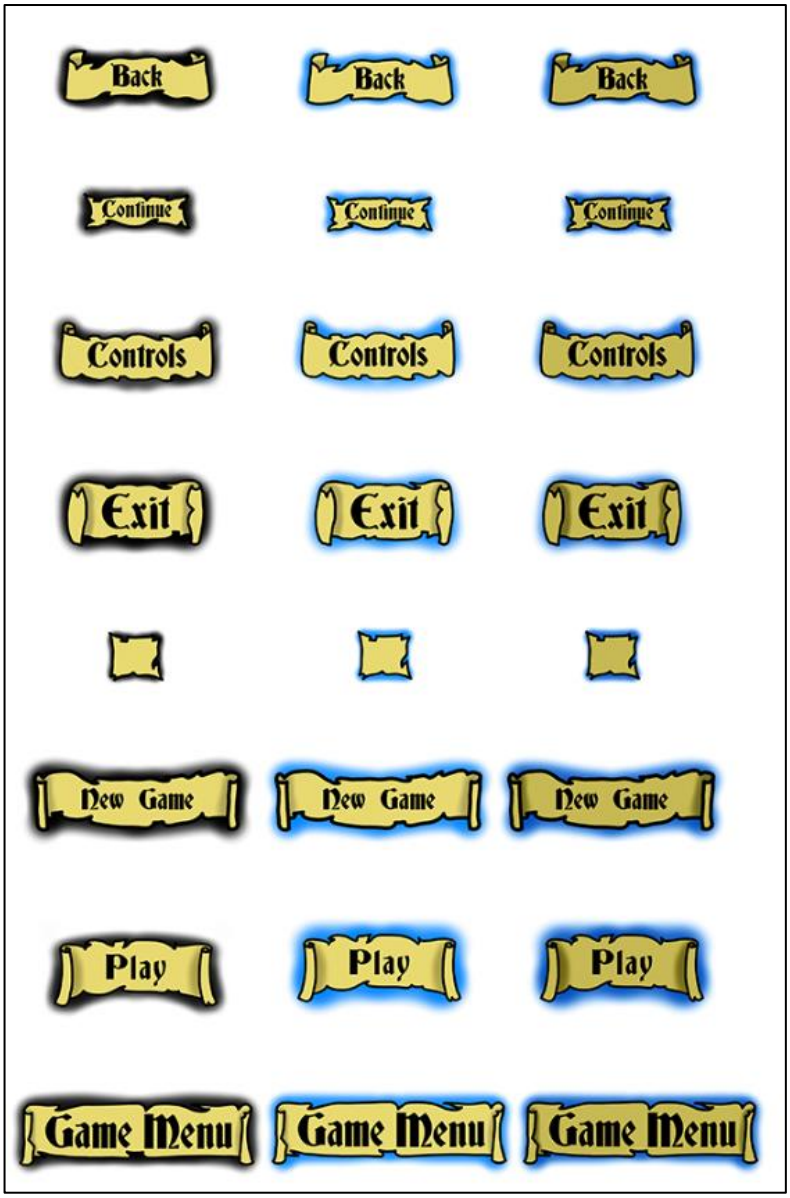


Figura 102 - Botões de jogo

Durante o jogo o jogador tem acesso a várias habilidades especiais. Para ter informação sobre estas habilidades, como quando estão disponíveis ou não, existem ícones no ecrã de jogo para saber quais as habilidades a que o jogador tem acesso mostrando a imagem do ataque e o seu tipo de poder. Quando uma habilidade não está disponível o ícone fica escuso indicando assim ao utilizador que a habilidade está indisponível. Todos os ícones desenvolvidos e usados no jogo são possíveis de observar na Figura 103.

Figura 103 - Ícones das habilidades especiais

Também no ecrã de jogo é possível tem acesso a várias outras informações como a barra de vida do jogador, a barra de escudo, a barra de magia e e barra de rolar. As informações da vida contêm uma moldura amarela com o símbolo de um coração e a sua barra é vermelha. A barra de magia também tem a mesma moldura que a vida, porém tem um símbolo de um fogo azul simbolizando a sua magia e uma cor azul. A barra de escudo também contém a mesma moldura que as outras barras, porém tem como símbolo um escudo roxo e cor roxa. A barra de rolar encontra-se em cima da barra de vida de forma discreta, contém apenas uma linha preta como moldura e a sua cor é verde.

No último nível do jogo o jogador tem informação de uma outra barra de vida, a barra de vida do inimigo final que se encontra no topo centro do ecrã. Esta barra contém também uma moldura amarela e um cristal azul para simbolizar o cristal onde se encontram todas as almas roubadas e a sua barra tem uma cor vermelha para simbolizar a sua vida, barra esta que retorna a encher na fase 2 do Bealzor. Todas as barras contêm um fundo de cor cinzento-escura.

Na Figura 104 pode-se observar as barras de vida, magia e escudo na zona superior, no centro encontra-se a barra de rolar e por último a barra de vida do inimigo final.

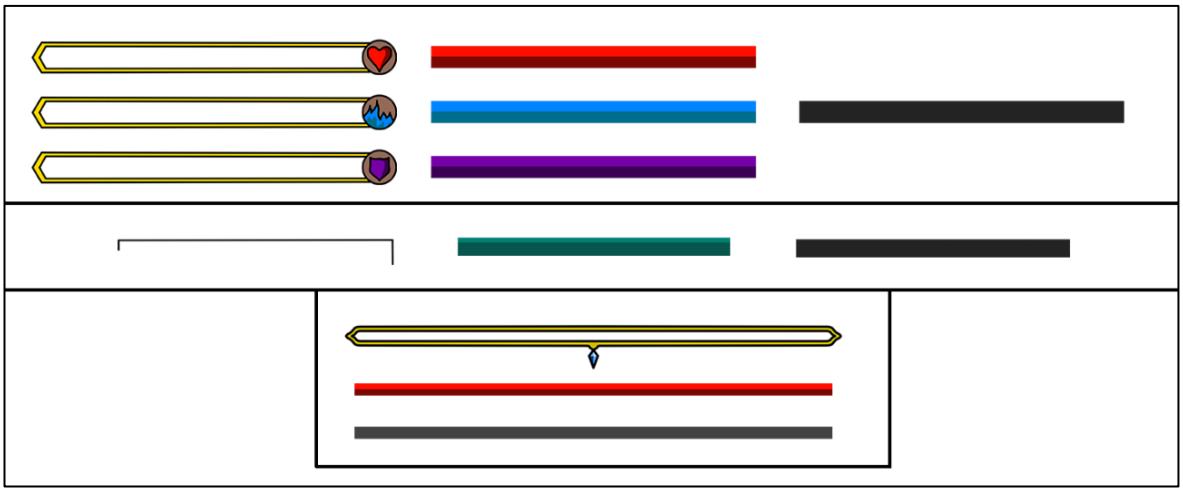


Figura 104 - Barras de vida e outras informações

Dentro do jogo existem também vários menus com uma imagem de fundo única em cada um deles. O primeiro menu que o jogador terá uma interação é o menu de jogo inicial, tendo sido desenvolvida uma imagem que desse destaque a esse menu. Neste menu podemos encontrar desenhado um castelo para simbolizar o objetivo final do jogo onde iremos derrotar o inimigo final, uma floresta para representar o caminho até esse objetivo e por último o personagem principal a olhar para o que teria que enfrentar ao longo da história. Esta imagem é possível de observar na Figura 105.

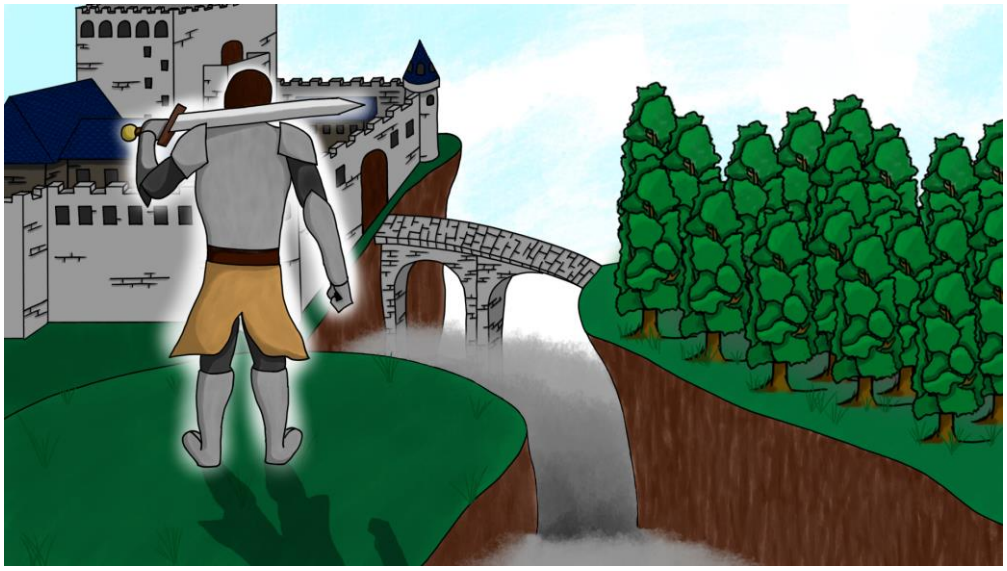


Figura 105 - Imagem de fundo do menu de jogo

Na Figura 106 é possível observar o resultado final do menu inicial de jogo. onde foram adicionados a imagem o título de jogo e os vários botões.

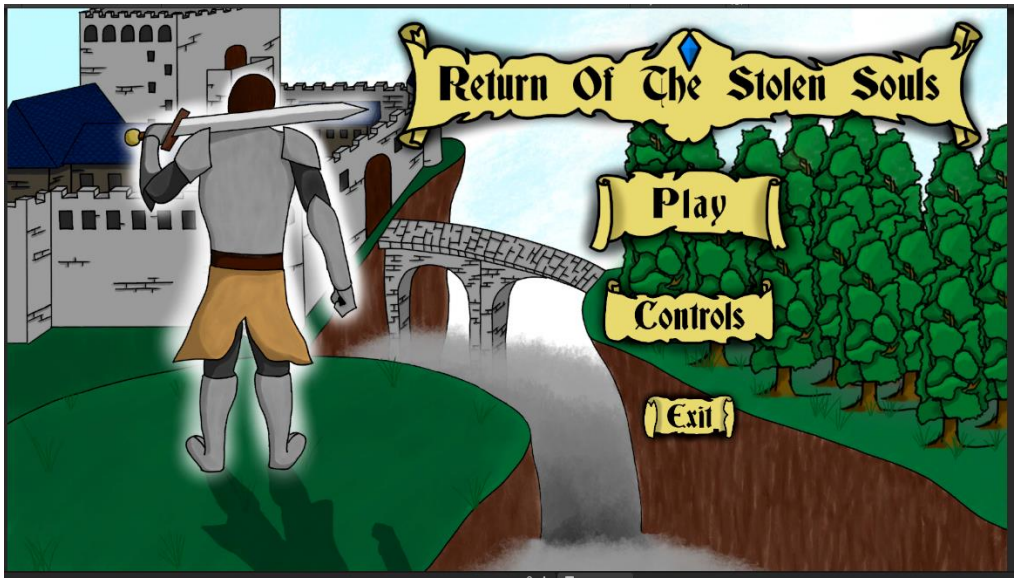


Figura 106 - Resultado final do menu de jogo

Um dos menus onde o jogador passará mais tempo durante o seu tempo a jogar é o menu do mapa de jogo, onde pode observar o seu progresso. A imagem desenvolvida para o mapa contém a representação de uma montanha onde se encontram os vários biomas apresentados ao longo do jogo e por último o castelo mesmo no topo, representando o nível final. Na Figura 107 é possível observar a imagem desenvolvida para o mapa de jogo.

Figura 107 - Imagem de fundo do mapa de jogo

Na Figura 108 é possível observar o resultado final do mapa de jogo onde foram adicionados os vários botões para a interação do jogador. Foram adicionados o botão de “New Game”, “Continue” e “Exit”, bem como os vários botões com os níveis que estarão disponíveis à medida que o jogador desbloqueia o seu progresso. Para o jogador poder aceder ao último nível dentro do castelo, foi transformado o castelo num botão também que quando se encontra o rato por cima dele, o mesmo fica destacado com um fundo azul e uma cor mais clara.

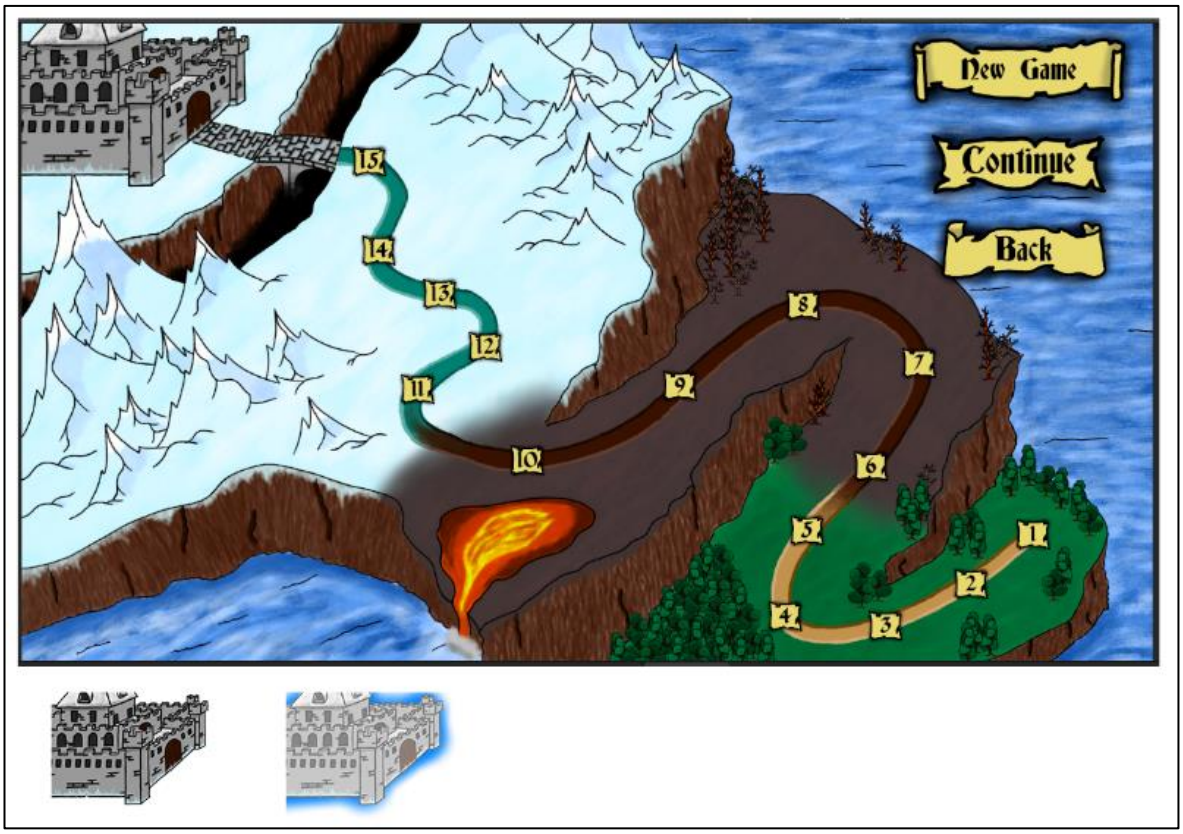


Figura 108 - Resultado final do mapa de jogo

Um dos menus mais importantes do jogo é o menu de controlos. Na Figura 109 é possível observar o menu desenvolvido com toda a informação dos controlos que o jogador tem acesso dentro do jogo.

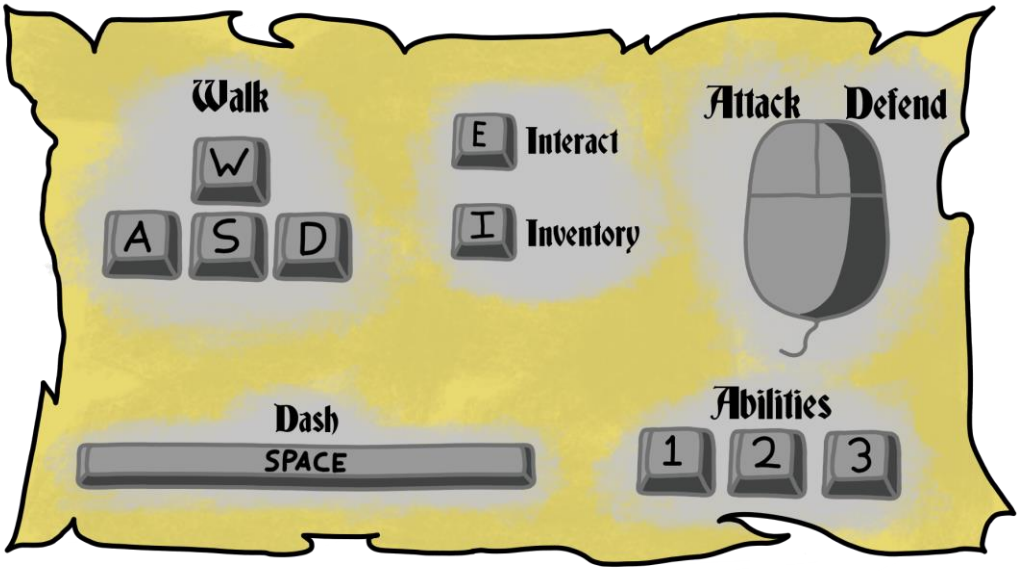


Figura 109 - Menu de controlos de jogo

Na Figura 110 é possível observar todos os ícones usados para representar objetos. Os primeiros cinco com moldura azul representam todos os materiais presentes no jogo (madeira, couro, metal, cristais de gelo e fogo), os nove ícones com moldura castanha representam todas as armas presentes no jogo (espada, facas e machado, com

as suas respetivas variações de fogo e gelo) e por fim os últimos 5 ícones com moldura roxa representam todas as poções do jogo (velocidade, escudo, magia, vida e dano). Estes ícones podem ser encontrados no inventário de jogo ou nos três menus interativos das casas, ou seja, na loja de materiais, na forja e na loja de poções.

Figura 110 - Ícones de materiais, armas e poções

O Inventário do jogador contém cerca de três menus onde cada um apresenta um nome diferente, sendo estes, "Weapons", "Potions" e "Materials" que podem ser observados na Figura 111. Estes menus contêm a informação de todos os materiais, armas e poções que o jogador possui. Na Figura 111 é possível observar na parte superior os três menus antes de obter os objetos, enquanto que na zona inferior é possível observar uma representação onde se encontram os objetos organizados consoante o seu conteúdo e disponibilidade.

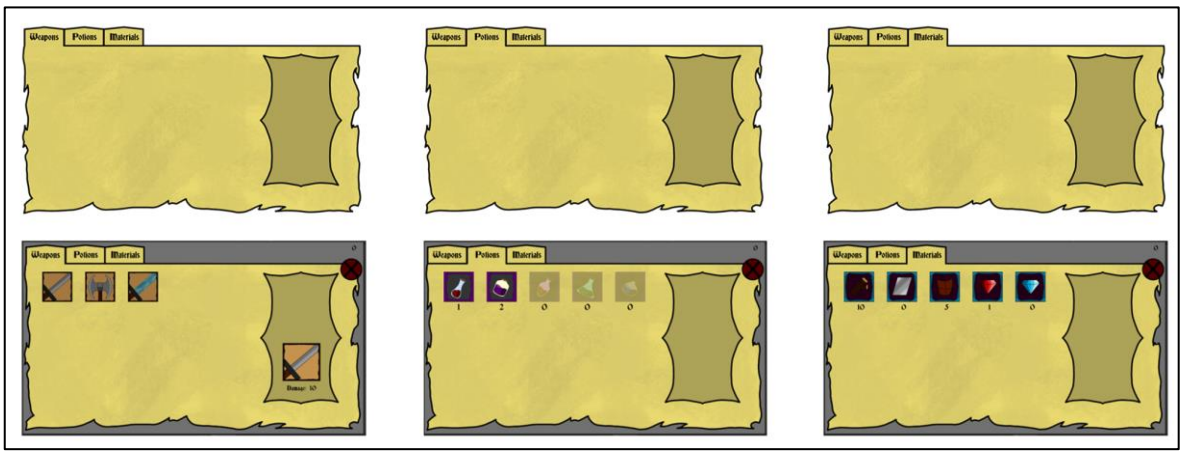


Figura 111 – Inventário

Um dos principais menus do jogo é a forja onde o jogador poderá construir as suas armas. Na Figura 112 é possível observar a imagem de fundo desenvolvida para este menu onde se pretende transmitir a ideia de semelhança a uma forja real. Do lado esquerdo encontram-se as opções de armas e do lado direito (que se encontra-se realçado com uma luz laranja para simular o fogo existente na forja) a informação do materiais que são necessários para construir a arma selecionada. No fundo da Figura 112 é possível observar um botão desenvolvido especialmente para a forja, com as suas variações.

Todos os botões presentes na forja, loja de materiais e poções têm a mesma variação de botões, apenas alterando a escrita entre *Construct* e *Buy*, sendo estas variações: Botão Normal (pronto para construir/comprar), botão translucido (impossível adquirir por falta de recursos) e botão mais escuro (quando clicamos para construir/comprar).

Figura 112 – Forja

Na Figura 113 é possível observar imagem desenvolvida para a loja de materiais. Esta imagem contém uma zona do lado esquerdo onde se encontram os ícones com todos os materiais disponíveis para compra dentro da loja. Do lado direito é possível observar toda a informação do objeto selecionado assim como a informação para poder comprá-lo.

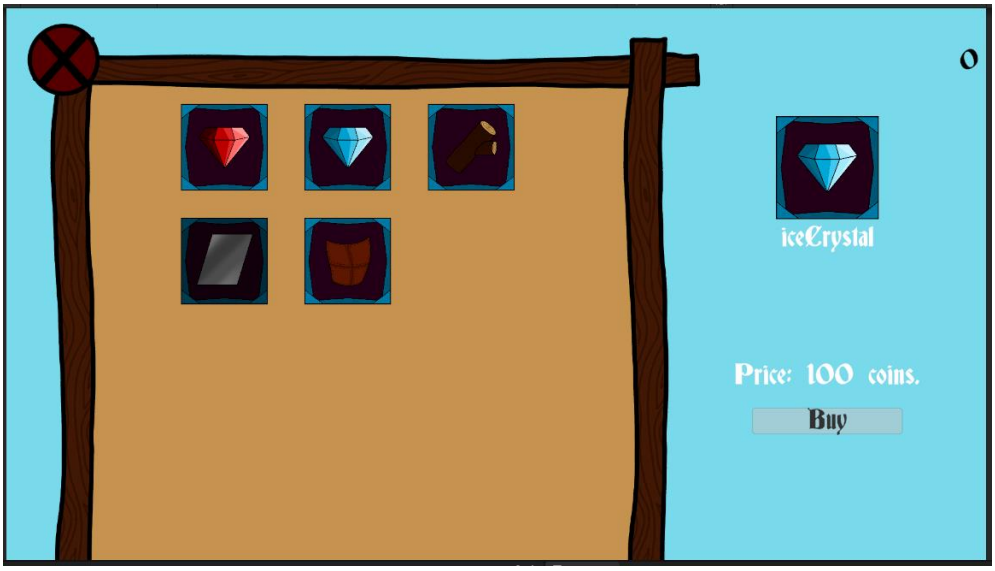


Figura 113 - Loja de materiais

Outra loja muito importante dentro do jogo é a loja de poções, que é possível de observar na Figura 114. Para esta loja foi desenvolvido uma imagem de fundo escura repleta de cortinas, para transmitir a ideia de uma loja misteriosa. Do lado esquerdo encontra-se uma estante com todas as poções possíveis de obter dentro o jogo e do lado direto toda a informação da poção selecionada assim como o botão para a adquirir.

Figura 114 - Loja de poções

Ao longo do jogo existe a possibilidade do jogador perder (quando a vida chega ao zero) e para isso foi também desenvolvido um menu dedicado para esse efeito. O menu desenvolvido na Figura 115 foi desenhado para poder transmitir a ideia de derrota para quando o personagem principal morre durante uma luta.

Figura 115 - Menu de Game Over

Mas tal como o jogador pode perder o jogo, também o pode ganhar, pelo que foi desenvolvido o menu de vitória que se encontra na Figura 116. Para este menu foi desenhado como fundo um cristal partido para representar as almas que foram libertadas quando o jogador derrota o inimigo final.

Figura 116 - Menu de vitória

4 Implementação do jogo

Neste capítulo será apresentado todo o código desenvolvido e todas as funcionalidades presentes dentro do jogo. Será apresentado a forma como foram controladas as animações presentes no jogo e o seu funcionamento, e em seguida será explicado todo o código desenvolvido através de fluxogramas. Controlo de animações

Foram produzidas cerca de trinta e três animações diferentes para que fosse possível dar ao personagem a capacidade de produzir movimentos e ataques diferentes de arma para arma. As animações podem-se dividir em três grupos, sendo eles espada, machado e facas. Cada grupo conta com a animação do personagem parado, de caminhar, três ataques e três habilidades especiais, conta também com a animação de rolar, defender e de morrer.

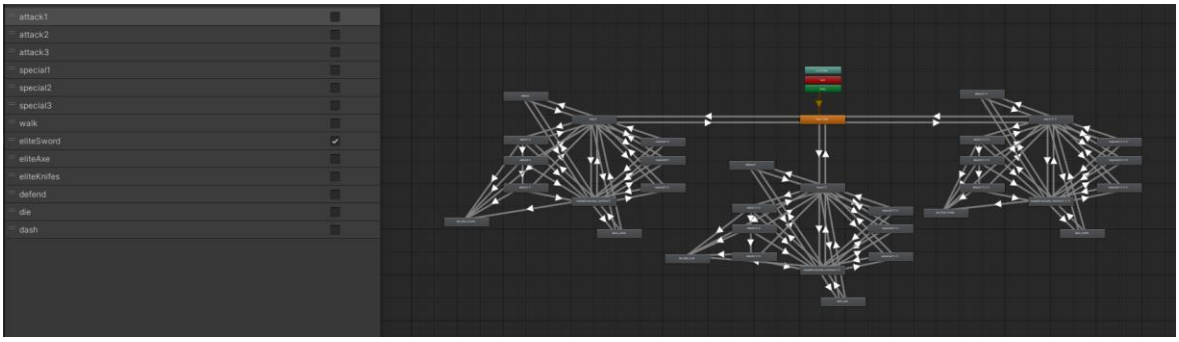


Figura 117 - Animator para controlo de animações

Para ser possível que o personagem usasse todas as animações foi construído um *animator* para poder controlar as mesmas ao longo do jogo. No lado direto da Figura 117 é possível observar o *animator* construído dentro do software *Unity,* do lado esquerdo todas as variáveis utilizadas para poder ativar ou desativar as animações. O *animator* pode ser separado também em três grupos, sendo esteso grupo de animações para a espada, para o machado e por último para as facas. Cada grupo tem o mesmo número de animações e são exatamente iguais na maneira como funcionam, apenas mudando as animações dentro de cada controlador.

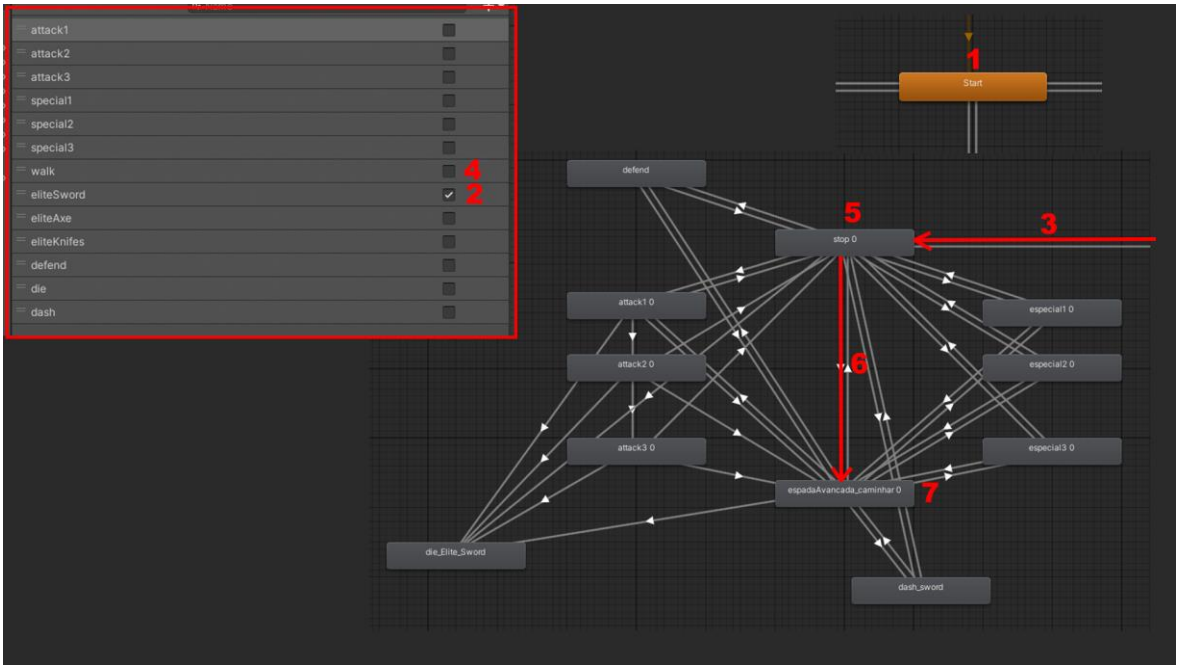


Figura 118 - Controlo do Animator

Na Figura 118 é possível observar o controlo do *animator.* Quando o jogo é iniciado o personagem começa com o primeiro estado representado com o número 1 e por consequência da variável "eliteSword" (número 2) estar já ativa o estado muda para o controlo de animações pertencentes à espada (número 3), sendo a primeira a de parada (número 5). Se o jogador decidir fazer o personagem caminhar, a variável "walk" (número 4) é ativada e o estado do *animator* é mudado através do número 6 para o estado de caminhar representado pelo número 7 por exemplo. Caso o jogador deixe de pressionar as teclas para caminhar a variável "walk" (número 4) é desativada e o estado do *animator* volta a passar pelo número 6 voltando ao estado de parado (número 5). Para os restantes estados o processo é o mesmo, sempre que uma variável é ativa o estado do *animator* muda. Caso o jogador mude de arma, é também alterada a variável responsável pelo controlo das animações de cada arma, passando para a respetiva arma selecionada.

4.1 Personagem principal Acolon

Neste subcapítulo será descrito todo o código desenvolvido para o jogador poder controlar o personagem principal. Primeiramente será feita uma descrição do movimento, ação ou mecânica que se pretende apresentar, seguido de um fluxograma para explicar o código desenvolvido.

- **Descrição de movimentos**

Como dito anteriormente o personagem conta com uma série de movimentos possíveis de serem executados dentro o jogo. Os principais movimentos que o personagem é capaz de produzir são o movimento de caminhar, rolar, defender, executar combos de ataques e invocar habilidades/ataques especiais.

- **Caminhar**

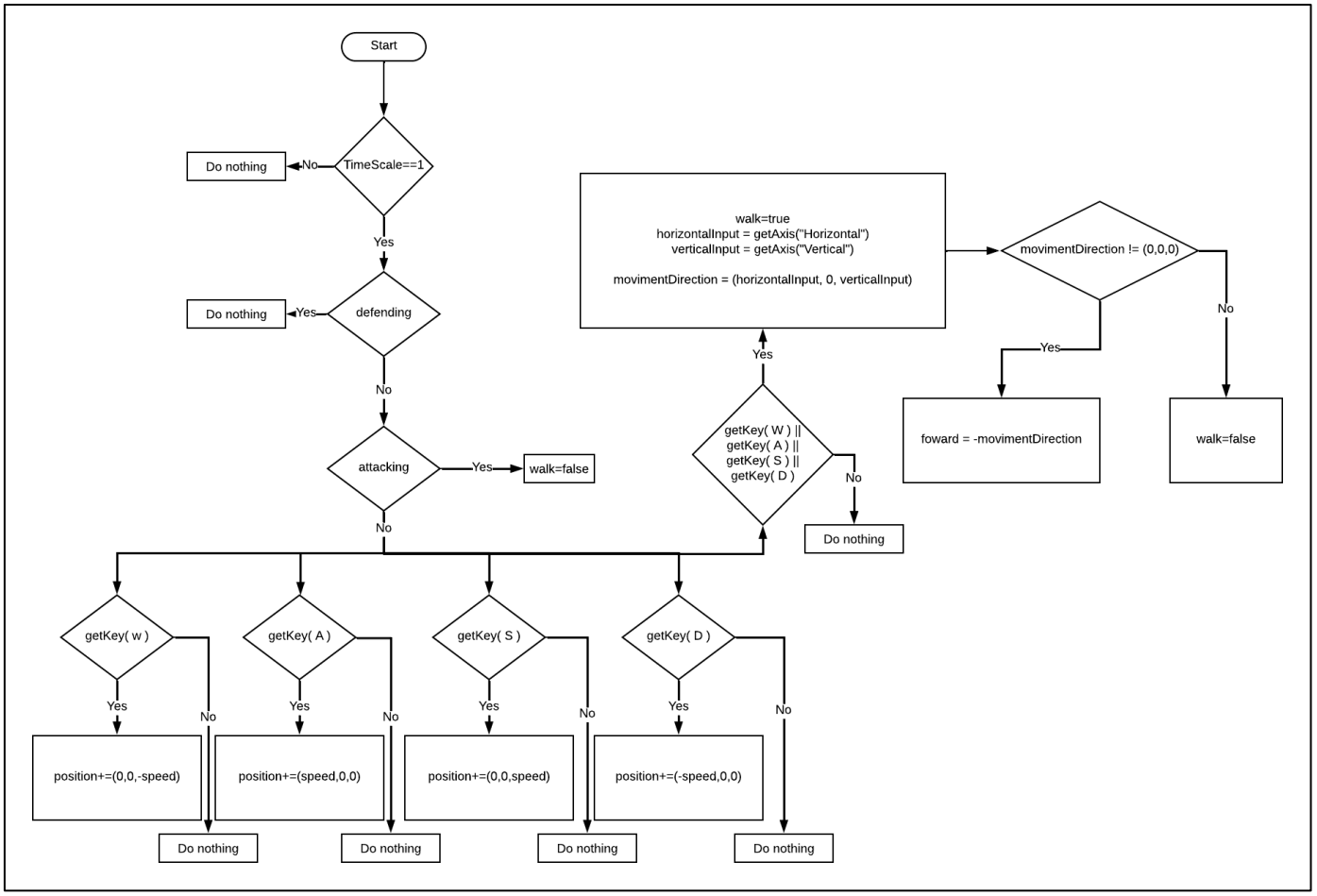


Figura 119 - Fluxograma Caminhar

Na Figura 120 é possível observar o código que foi desenvolvido para fazer com que o personagem possa caminhar. Seguindo o fluxograma presente na Figura 119, jogador pode fazer o personagem caminhar através das teclas "WASD" do seu teclado caso este não esteja a atacar, defender ou em modo pausa (TimeScale = 0). Para que o personagem fique virado na direção que o jogador deseja são usados os eixos do *Unity* (Horizontal e Vertical) para calcular a direção para onde o personagem tem que se virar. Caso o jogador pare de clicar numa das quatro teclas o personagem fica virado na direção em que estava. Ao caminhar é ativada a animação de caminhar e quando parado a animação de caminhar para e passa para a animação de parado.

```
if(Time.timeScale == 1){
    if(!Stats.defending){
        if(!Stats.attacking){
            if(Input.GetKey(KeyCode.W)){
                transform.position += new Vector3(0f,0f,-Stats.speed*Time.deltaTime);
            }

            if(Input.GetKey(KeyCode.A)){
                transform.position += new Vector3(Stats.speed*Time.deltaTime,0f,0f);
            }

            if(Input.GetKey(KeyCode.S)){
                transform.position += new Vector3(0f,0f,Stats.speed*Time.deltaTime);
            }

            if(Input.GetKey(KeyCode.D)){
                transform.position += new Vector3(-Stats.speed*Time.deltaTime,0f,0f);
            }

            if(Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.S) || Input.GetKey(KeyCode.D)){
                actions.SetBool("walk",true);

                float horizontalInput = Input.GetAxis("Horizontal");
                float verticalInput = Input.GetAxis("Vertical");

                Vector3 movementDirection = new Vector3(horizontalInput, 0, verticalInput);

                if (movementDirection != Vector3.zero)
                {
                    Player.transform.forward = -movementDirection;
                }
            }else{
                actions.SetBool("walk",false);
            }

        }else{
            actions.SetBool("walk",false);
        }
    }
```

Figura 120 - Código de caminhar

- **Defender**

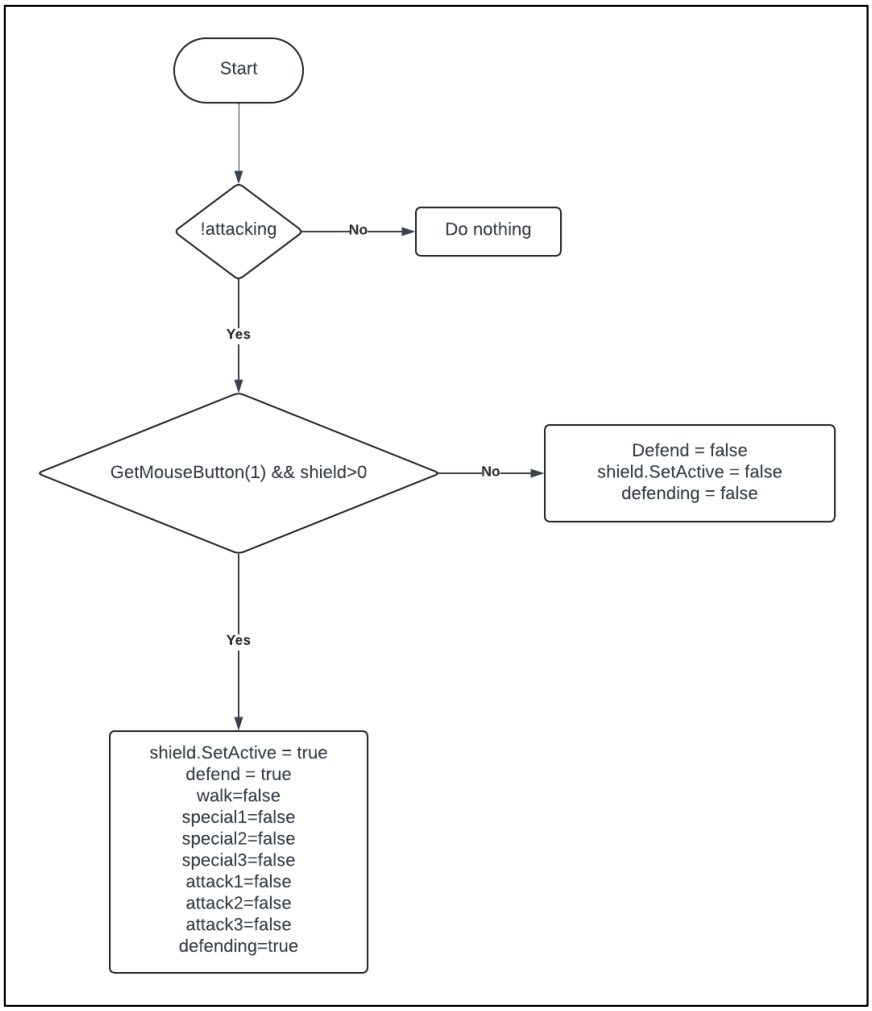


Figura 121 - Fluxograma defender

Para fazer o personagem defender, caso não esteja em modo ataque, o jogador deve pressionar o botão direto do rato ou botão 1 do mouse como é possível observar no fluxograma da Figura 121. Ao ativar o modo de defender o personagem ativa a animação de defender, desativando todas as outras, passa o "*defending*" para *true*, e ativa o efeito especial de escudo fazendo assim com que não seja possível executar qualquer movimento enquanto estiver a defender. Caso não seja mais preciso defender ou o escudo chegue a zero, o estado "*defending*" passa a *false* e desativa a animação de defender assim como o seu efeito especial.

```
if(!Stats.attacking){
    if(Input.GetMouseButton(1) && Stats.shield>0){
        shield.SetActive(true);
        actions.SetBool("defend",true);
        actions.SetBool("walk",false);
        actions.SetBool("special1",false);
        actions.SetBool("special2",false);
        actions.SetBool("special3",false);
        actions.SetBool("attack1",false);
        actions.SetBool("attack2",false);
        actions.SetBool("attack3",false);
        Stats.defending=true;
    }else{
        actions.SetBool("defend",false);
        shield.SetActive(false);
        Stats.defending=false;
    }
```

Figura 122 – Código para defesa

- **Abrir inventário, menu de pausa ou interagir**

Dentro do jogo é possível abrir diversos menus que o jogador pode aceder durante o seu tempo de jogo. Estes menus podem ser o próprio inventário, o menu de pausa e os menus das lojas ou forja dentro de alguns níveis.

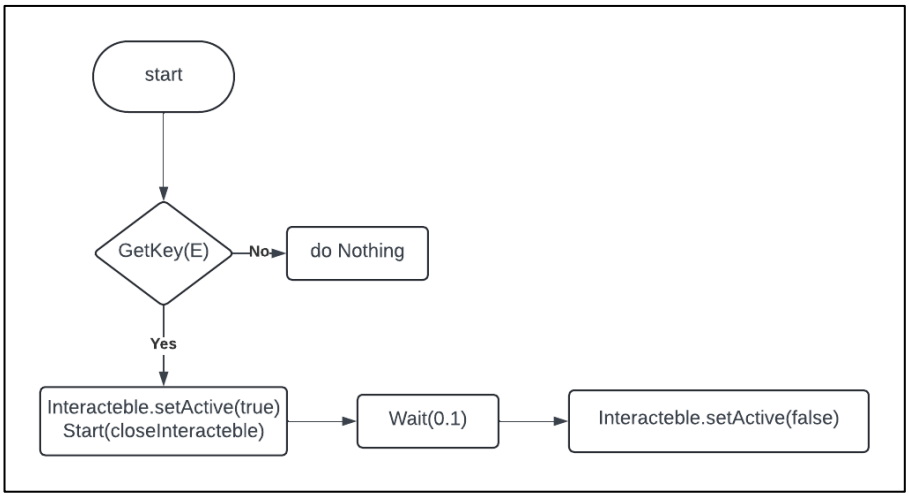


Figura 123 - Fluxograma Interagir

Na Figura 123 é possível observar o fluxograma pertencente ao excerto de código presente na zona A da Figura 126 que serve para abrir a forja e os menus das lojas. Para o jogador poder abrir um menu dentro duma loja terá de se deslocar até a loja em questão e clicar na tecla "E" do seu teclado na qual ativa um *collider* e invoca uma função de tempo (retângulo vermelho da zona A da Figura 126) que espera cerca de 0.1 segundos e volta a desativar o *collider* anteriormente ativado.

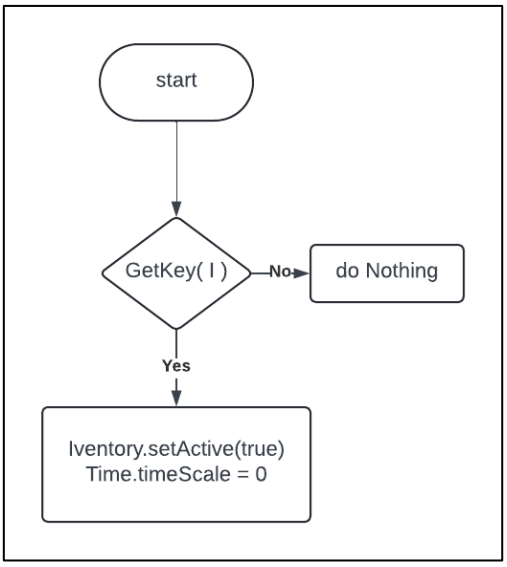


Figura 124 - Fluxograma abrir inventário

Na Figura 124 é possível observar o fluxograma pertencente ao excerto de código na zona B da Figura 126. Este pedaço de código faz com que quando o jogador clique na tecla "I" do seu teclado, o menu do inventário fique ativo e coloque o jogo em pausa (Time.timeScale=0).

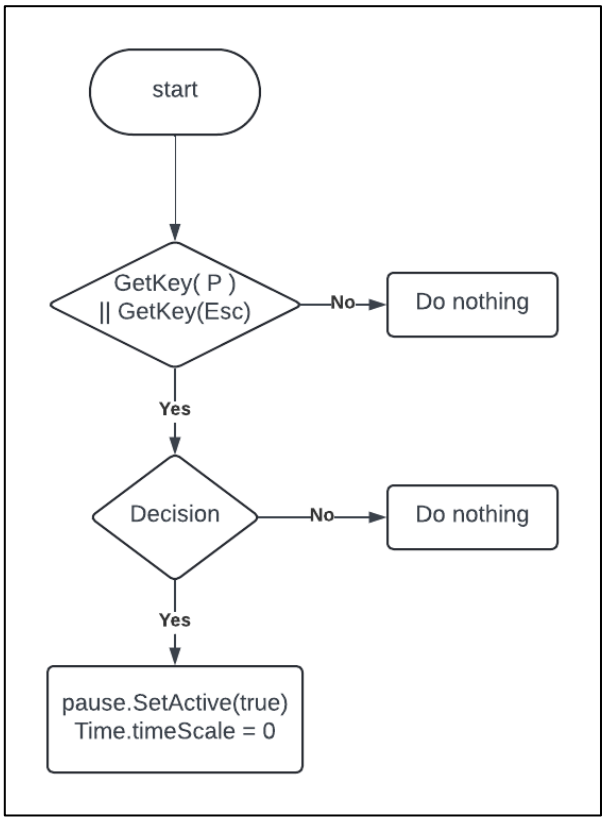


Figura 125 - Fluxograma pausa

Na Figura 125 pode ser observado o fluxograma para explicar como o jogador pode colocar o jogo em pausa através do excerto de código presente na zona C da Figura 126. Para o jogador colocar o jogo em pausa é verificado primeiramente se o jogo já está colocado em pausa (por exemplo com o inventário aberto) quando este clica na tecla

"P" ou "Esc" para assim poder evitar menus sobrepostos, do seu, caso o jogo já não esteja em pausa é ativado o menu de pausa e colocado o jogo em pausa.

```
if(Input.GetKeyDown(KeyCode.E)){
    Interecteble.SetActive(true);
    StartCoroutine(closeInteracteble());
}
```

A

```
IEnumerator closeInteracteble(){
    yield return new WaitForSeconds(0.1f);
    Interecteble.SetActive(false);
}
```

```
if(Input.GetKeyDown(KeyCode.I)){
    Iventory.SetActive(true);
    Time.timeScale = 0;
}
```

B

```
if(Input.GetKeyDown(KeyCode.P) || Input.GetKeyDown(KeyCode.Escape)){
    if(Time.timeScale == 1){
        pause.SetActive(true);
        Time.timeScale = 0;
    }
}
```

C

Figura 126 - Código para abrir os menus de jogo

- **Ataques**

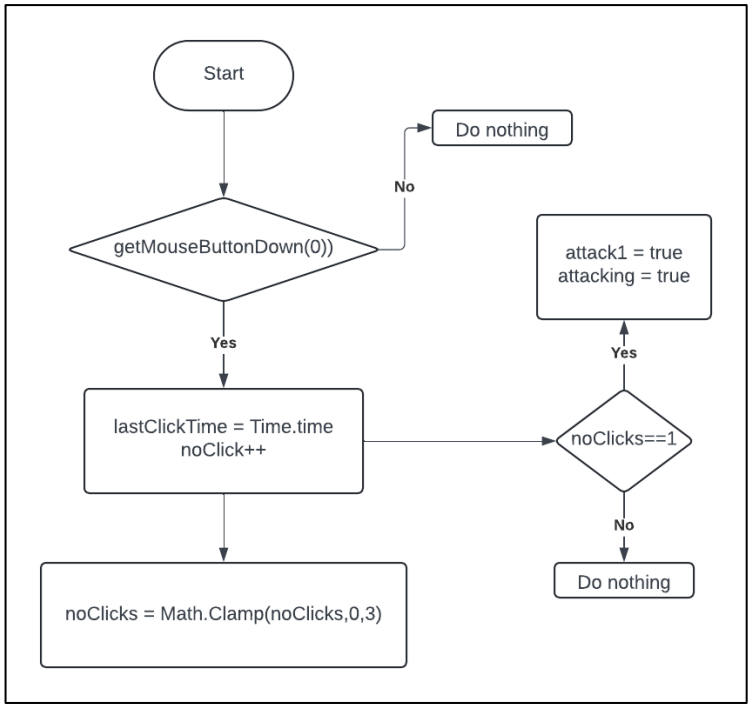


Figura 127 - Fluxograma início de combo de ataques

Para que o jogador possa atacar os inimigos durante o jogo o mesmo terá de clicar no seu botão esquerdo do rato para começar o combo de ataques como é possível observar no fluxograma da Figura 127 pertencente ao excerto de código da Figura 128. Ao começar o combo de ataques é possível apenas produzir 1 ataque, 2 ataques ou no máximo3 ataques em sequência. Quando o combo começa é contado o número de cliques o jogador faz e guarda o tempo em que este começou a clicar, ativa a animação

de ataque 1 e é colocado o personagem em modo de ataque. O número de cliques que o jogo guarda é no mínimo zero e no máximo três.

```
if(Input.GetMouseButtonDown(0)){
    lastClickTime=Time.time;
    noClicks++;
    if(noClicks==1){
        attack.SetBool("attack1",true);
        Stats.attacking=true;
    }
    noClicks = Mathf.Clamp(noClicks,0,3);
}
```

Figura 128 - Código para início do combo de ataques

Na Figura 129 é possível observar que dentro da animação são invocadas várias funções para que seja possível executar os ataques e produzir um combo. No número 1 e 2 da figura da Figura 129 são invocadas as funções "*attack*" que vai fazer com que detete se colidiu com um inimigo para que possa produzir dano e "*activeCollider*" que vai fazer com que ative o *collider* da arma ativa para produzir o dano, estas funções são invocadas no *frame* número 13 da animação da espada. No *frame* seguinte (14) é desativado o *collider* da arma com a invocação da função "*desactiveCollider*". Por último é invocada a função "*return1*" no *frame* 15 da animação para que possa ser verificado se o número de cliques que o jogador deu é maior ou igual a 1, para que assim seja possível continuar o combo ativando a segunda animação ou caso o número de cliques seja igual a 1 a animação do ataque é desativada e o ciclo recomeça quando o jogador volta a clicar no botão esquerdo do seu rato.

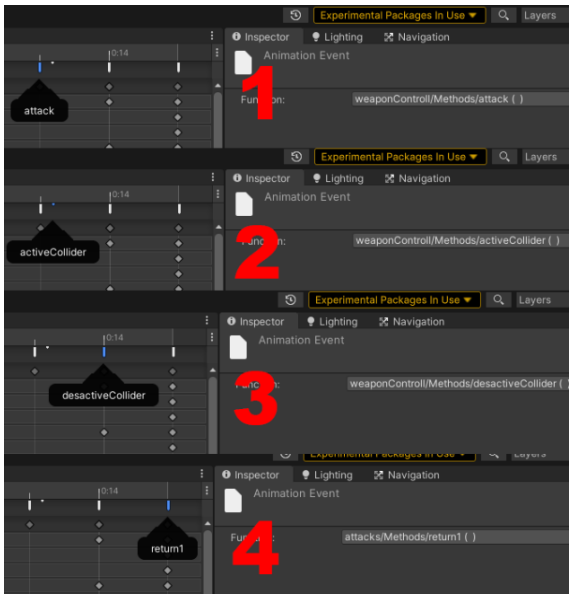


Figura 129 - Controlo do ataque dentro da animação

Observação: Cada arma tem as suas animações, logo a posição onde são invocadas as funções pode variar dentro da animação, porém a ordem de invocação é igual em qualquer situação.

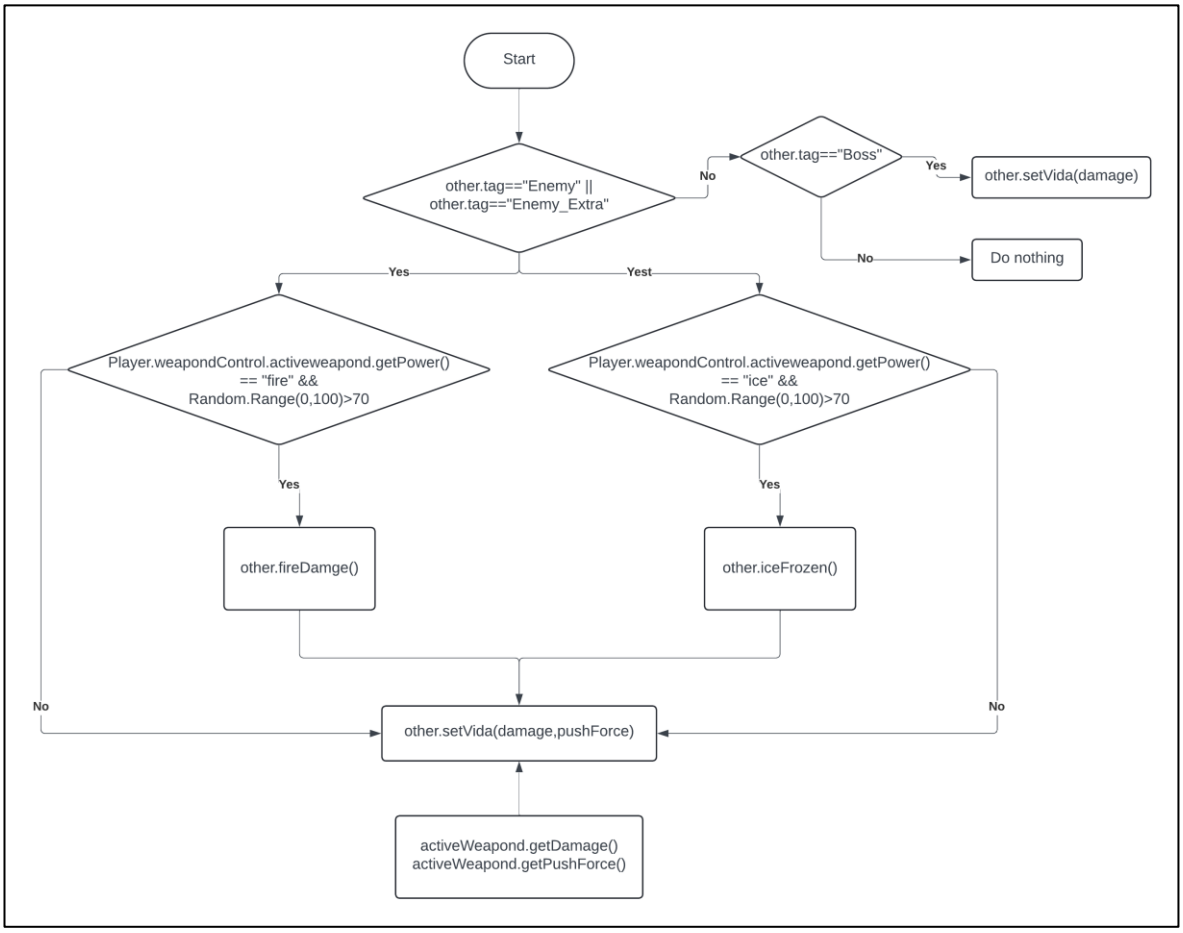


Figura 130 - Fluxograma dar dano

Na Figura 132 é possível observar o código desenvolvido para controlar o ataque do jogador. No número 1 da Figura 132 encontra-se a função "*attack*" que é a primeira e ser invocada dentro da animação (Figura 129), esta função vai passar o dano e o valor de empurrar da arma ativa para dentro do seu range com a função "*setDamage*" presente no número 2 da Figura 132. Em seguida é invocada a função "*activeCollider*" (número 1 da Figura 132, número 2 da Figura 129) onde vai ativar o range da arma ativa e vai verificar se ouve colisão com um inimigo.

Através do fluxograma da Figura 130, que representa o código presente no número 2 da Figura 132, é possível observar que caso haja colisão com um inimigo ("other" com *tag* "Enemy" ou "Enemy_Extra") é verificada se a arma ativa contém algum poder e tendo cerca de trinta por cento de hipóteses de ativar o poder da arma, ou seja, colocar o inimigo em fogo ou congelá-lo, em seguida é dado o dano e valor de empurrar ao inimigo atingido através dos valores adquiridos da função "attack" do número 1 da Figura 132. Caso o inimigo atingido seja o chefe final (*tag* igual a "Boss") é apenas dado o dano da arma pois no último nível não é possível usar os poderes especiais das armas.

A função que é invocada em seguida é a função "*desactiveCollider*" onde vai desativar o *collider* da arma ativa (número 3 da Figura 132).

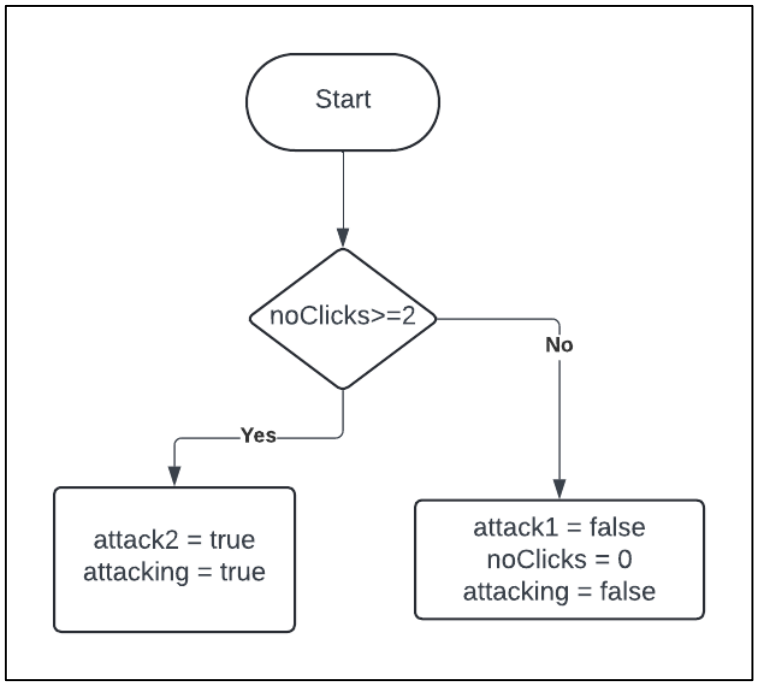


Figura 131 - Fluxograma da função return1

Por último é invocada a função "*return1*" presente no número 4 da Figura 132. Através do fluxograma da Figura 131 é possível observar que vai ser verificado se o número de cliques é maior ou igual a 2, caso seja ativa a animação de ataque 2 mantém o personagem em modo ataque, caso seja menor a 2, a animação de ataque é desativada assim com o modo de ataque do personagem e o número de cliques é colocado a 0.

Figura 132 - Código de controlo de ataque

Quando a segunda animação de ataque é ativada, as mesmas funções da Figura 129 são invocadas pela mesma ordem. Porém a última função invocada é a função "*return2*" presente na Figura 134 e através do fluxograma da Figura 133 é possível observar que irá ser verificado se o número de cliques é maior ou igual a 3 e, caso seja é ativada a aminação de ataque 3 mantem-se o modo de ataque do personagem ativo, caso não seja verdade, a animação de ataque é desativada assim como o modo de ataque e é colocado número de cliques a zero.

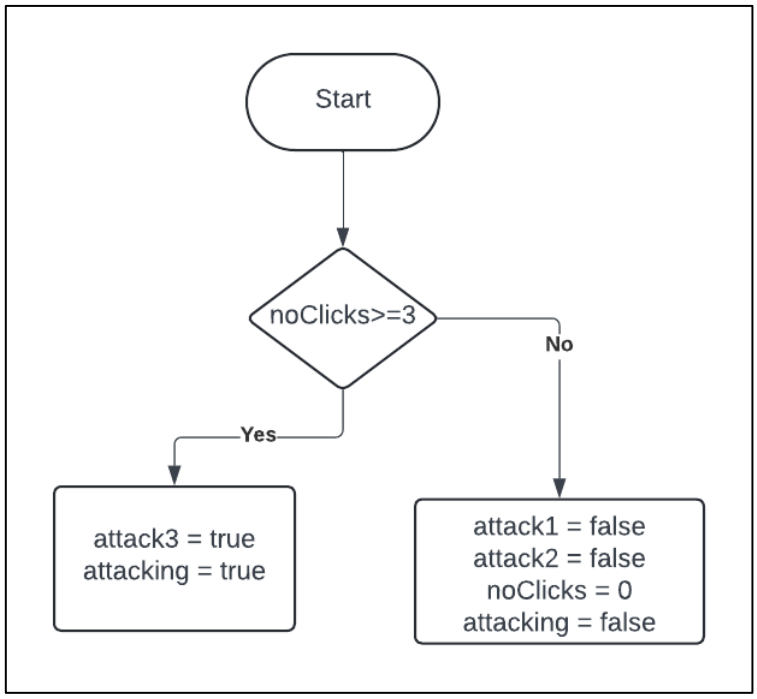


Figura 133 – Fluxograma da função return2

Na última animação de ataque é feita a mesma sequência de invocação de funções da Figura 129, porém assim como na segunda animação ao invés da última função ser "*return1*", a função invocada é "*return3*" (Figura 134), esta função vai dar término à sequência de ataques do personagem com a desativação das animações de ataques, colocando o número de cliques a zero e desativando o modo de ataque do personagem.

```
private void return2 (){
    if(noClicks>=3){
        attack.SetBool("attack3",true);
        Stats.attacking=true;
    }else{
        attack.SetBool("attack1",false);
        attack.SetBool("attack2",false);
        noClicks=0;
        Stats.attacking=false;
    }
}

0 references
private void return3 (){
    attack.SetBool("attack1",false);
    attack.SetBool("attack2",false);
    attack.SetBool("attack3",false);
    noClicks=0;
    Stats.attacking=false;
}
```

Figura 134 - Continuação do código de controlo de ataque

- ***Rolar***

```
if(Input.GetKeyDown(KeyCode.Space) && Stats.dash==100){
    actions.SetBool("dash",true);
    Stats.dash = 0;
    if(Stats.speed == 7f)
        Stats.speed = 14f;
}
```

Figura 135 - Código para rolar

Na Figura 135 é possível observar o código desenvolvido para fazer com que o personagem possa rolar para se esquivar. Para que o jogador possa fazer rolar o personagem terá de clicar na tecla "espaço" do seu teclado para que possa ativar o movimento. Ao clicar nessa tecla é verificado se a barra de rolar está cheia e caso esteja a animação de rolar é ativada e a barra passa a 0. A velocidade é comparada ao valor base da velocidade e caso seja igual a velocidade atual é alterada para 14. Se a barra de rolar não estiver a 100, não acontece nada, se a velocidade atual seja diferente do valor base apenas faz a animação de rolar.

- **Controlo de barras de vida e informação**

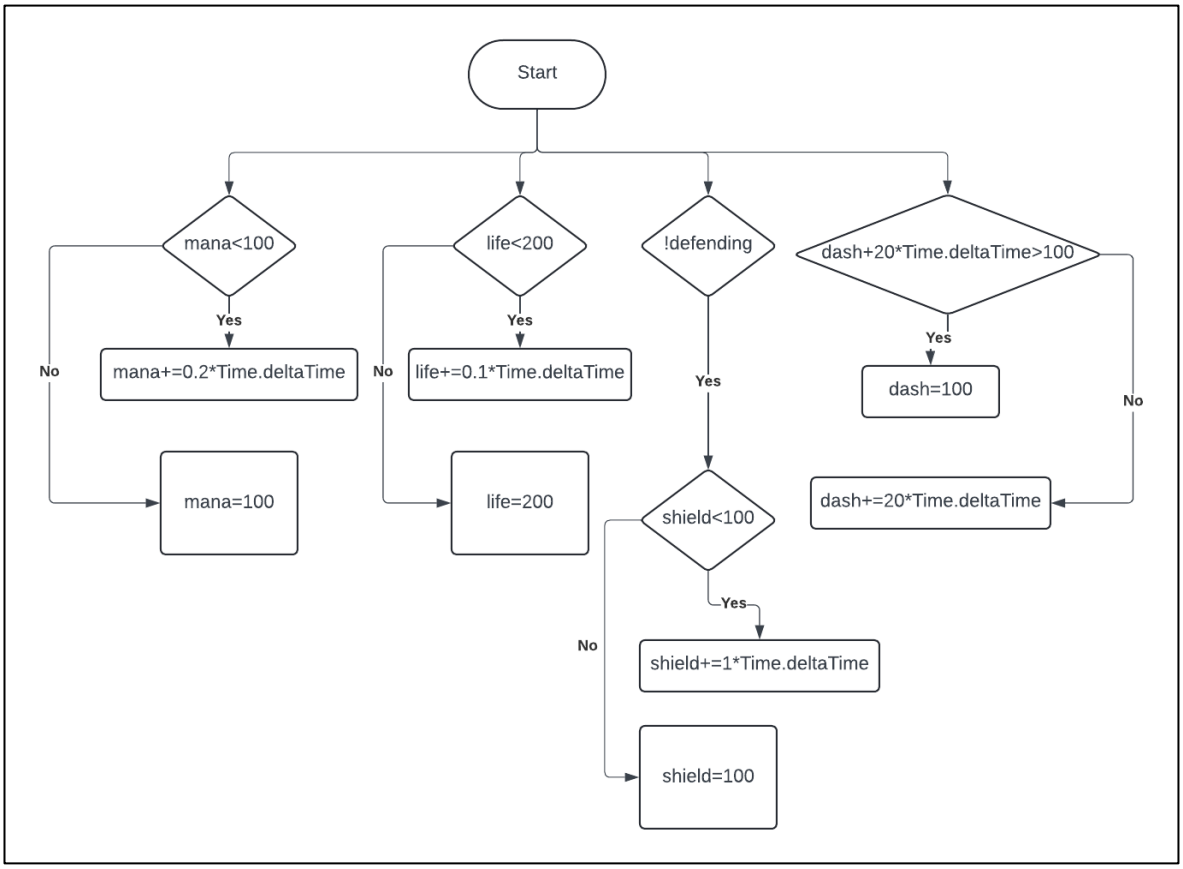


Figura 136 - Fluxograma das barras de informação

Através do fluxograma da Figura 136 é possível examinar o código representado na Figura 137 onde é possível observar o controlo das barras de estatísticas. A barra de vida (2 da Figura 137) irá recarregar cerca de 0.1 ao longo do tempo caso esta esteja menor que 200. A barra de magia (1 da Figura 137) recarrega cerca de 0.2 ao longo do

tempo caso esta se encontre menor que 100. A barra de escudo (3 da Figura 137) recarrega cerca de 1 ao longo tempo, caso esta esteja com um valor menor que 100 e o personagem não esteja a modo de defesa. Por último a barra de rolar (4 da Figura 137) é recarregada com o valor 20 ao longo do tempo logo depois de ser usada, para que possa recarregar rapidamente.

```
//controlo da barra de magia
if(Stats.mana<100)
    Stats.mana += 0.2f*Time.deltaTime;
else if(Stats.mana>100)
    Stats.mana=100;

//controlo da barra de vida
if(Stats.life<200)
    Stats.life += 0.1f*Time.deltaTime;
else if(Stats.life>200)
    Stats.life=200;

//controlo da barra de escudo
if(!Stats.defending){
    if(Stats.shield<100)
        Stats.shield += 1f*Time.deltaTime;
    else if(Stats.shield>100)
        Stats.shield=100;
}

//controo da barra de dash
if(Stats.dash+20*Time.deltaTime >100)
    Stats.dash=100;
else
    Stats.dash+=20*Time.deltaTime;
```

Figura 137 - Código de controlo das barras de informação das estatísticas

Observação: Quando o personagem recebe dano a barra de vida desce, porém caso o personagem esteja em modo de defesa a barra de escudo desce em primeiro lugar em vez da barra de vida, contudo essa informação é tratada através dos inimigos quando estes dão dano. A barra de magia é reduzida quando o personagem usa uma habilidade especial que será apresentada seguidamente.

- **Ataques especiais**

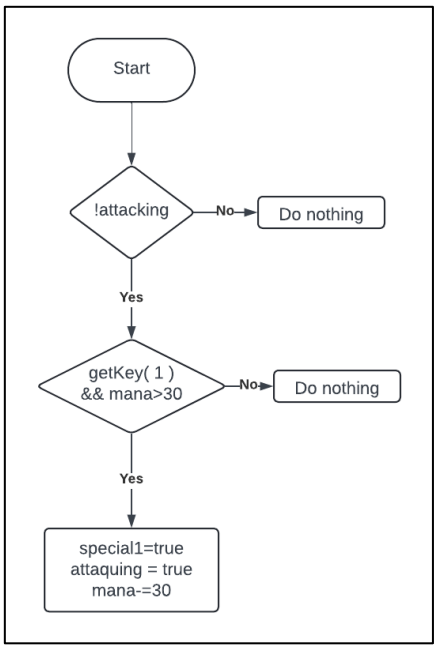


Figura 138 - Fluxograma da invocação da primeira habilidade especial

Dentro do jogo o personagem principal é capaz de invocar cerca de três habilidades especiais para o poder ajudar a derrotar os inimigos que encontra ao longo do tempo. Para invocar uma habilidade especial o jogador terá de clicar no número "1" do seu teclado para poder invocar a primeira habilidade especial. Através do fluxograma da Figura 138 é possível examinar o código da Figura 139 que, caso o personagem não esteja a atacar e o jogador clique no número "1" do seu teclado e caso a barra de magia esteja maior a 30 é iniciada a animação da habilidade especial 1, passa o personagem para modo de ataque e diminui a barra de magia com o valor 30, sendo este o custo da habilidade. Sendo que a primeira habilidade especial é igual para todas as armas é apenas invocada a animação de habilidade especial 1 da arma que o personagem tenha ativa.

```
if(!Stats.attacking){
    if(Input.GetKey(KeyCode.Alpha1) && Stats.mana>=30){
        attack.SetBool("special1",true);
        Stats.attacking=true;
        Stats.mana-=30;
    }
```

Figura 139 - Código invocação da primeira habilidade especial

Quando é invocada a animação da habilidade especial 1 é também invocada a função "CircleEffect" que recebe um valor *float*. Quando a animação percorre o *frame* em A da Figura 140 é invocada a função (B da Figura 140) que irá ativar o efeito da espiral, que irá receber um valor para verificar se a espiral é invocada com rotação horária ou anti horária.

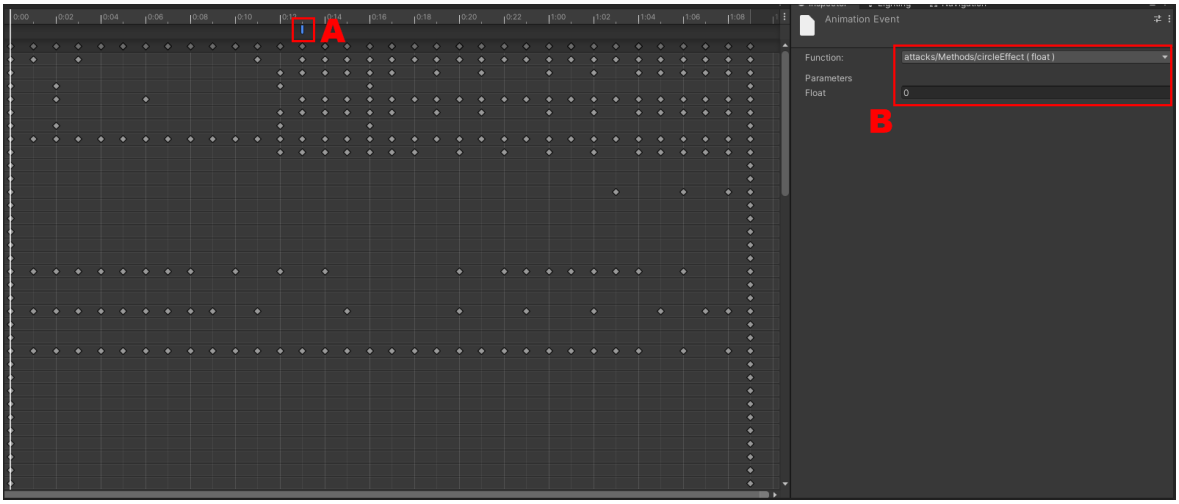


Figura 140 - Invocação de uma função dentro da animação especial 1

Quando invocada a função "circleEffect" é possível observar no fluxograma da Figura 141 que é criado um objeto com o valor *null*. Em seguida é verificado o tipo de poder da arma, se é nenhum ("none"), fogo ("fire") ou gelo ("ice") e em todos os casos é instanciado um efeito consoante o tipo de poder, por exemplo, se o poder da arma for de fogo, é instanciado um efeito de fogo na posição do jogador mais 3.7 em y e é dada a rotação recebida pela função. Por último é invocada a função de tempo "stopCircleEffect" presente na Figura 142.

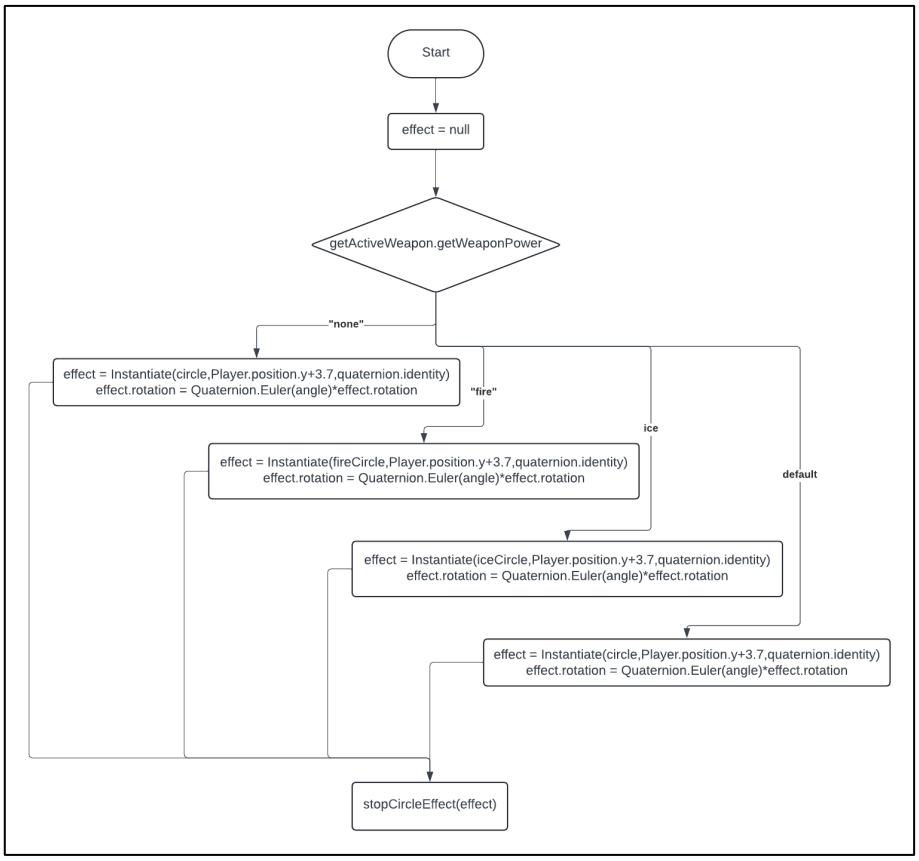


Figura 141 - Fluxograma criação do efeito especial 1

A função de tempo "stopCircleEffect" (Figura 142) tem como função destruir o efeito visual no fim da animação para que o objeto não fique dentro do jogo por tempo ilimitado. Quando é invocada esta função, a mesma recebe o objeto referente ao efeito e espera cerca de 0.8 segundos e, em seguida, destrói o objeto, desativa a animação de habilidade especial 1 e termina o modo de ataque.

```
public void circleEffect(float angle){
    GameObject effect = null;
    switch (transform.gameObject.GetComponent<weaponControll>().getActiveWeapon().GetComponent<weaponInfo>().getWeaponPower()){
        case "none":
            effect = Instantiate(circle.transform.gameObject,new Vector3(Player.transform.position.x,Player.transform.position.y+3.70f,Player.transform.position.z) ,quaternion.identity);
            effect.transform.rotation = Quaternion.Euler(angle, 0f, 0) * effect.transform.rotation;
        break;
        case "fire":
            effect = Instantiate(fireCircle.transform.gameObject,new Vector3(Player.transform.position.x,Player.transform.position.y+3.70f,Player.transform.position.z),quaternion.identity);
            effect.transform.rotation = Quaternion.Euler(angle, 0f, 0) * effect.transform.rotation;
        break;
        case "ice":
            effect = Instantiate(iceCircle.transform.gameObject,new Vector3(Player.transform.position.x,Player.transform.position.y+3.70f,Player.transform.position.z),quaternion.identity);
            effect.transform.rotation = Quaternion.Euler(angle, 0f, 0) * effect.transform.rotation ;
        break;
        default:
            effect = Instantiate(circle.transform.gameObject,new Vector3(Player.transform.position.x,Player.transform.position.y+3.70f,Player.transform.position.z),quaternion.identity);
            effect.transform.rotation = Quaternion.Euler(angle, 0f, 0) * effect.transform.rotation;
        break;
    }
    StartCoroutine(stopCircleEffect(effect));
}

1 reference
IEnumerator stopCircleEffect(GameObject effect){
    yield return new WaitForSeconds(0.8f);
    Destroy(effect);
    attack.SetBool("special1",false);
    Stats.attacking=false;
}
```

Figura 142 – Código de criação do efeito especial da habilidade especial 1

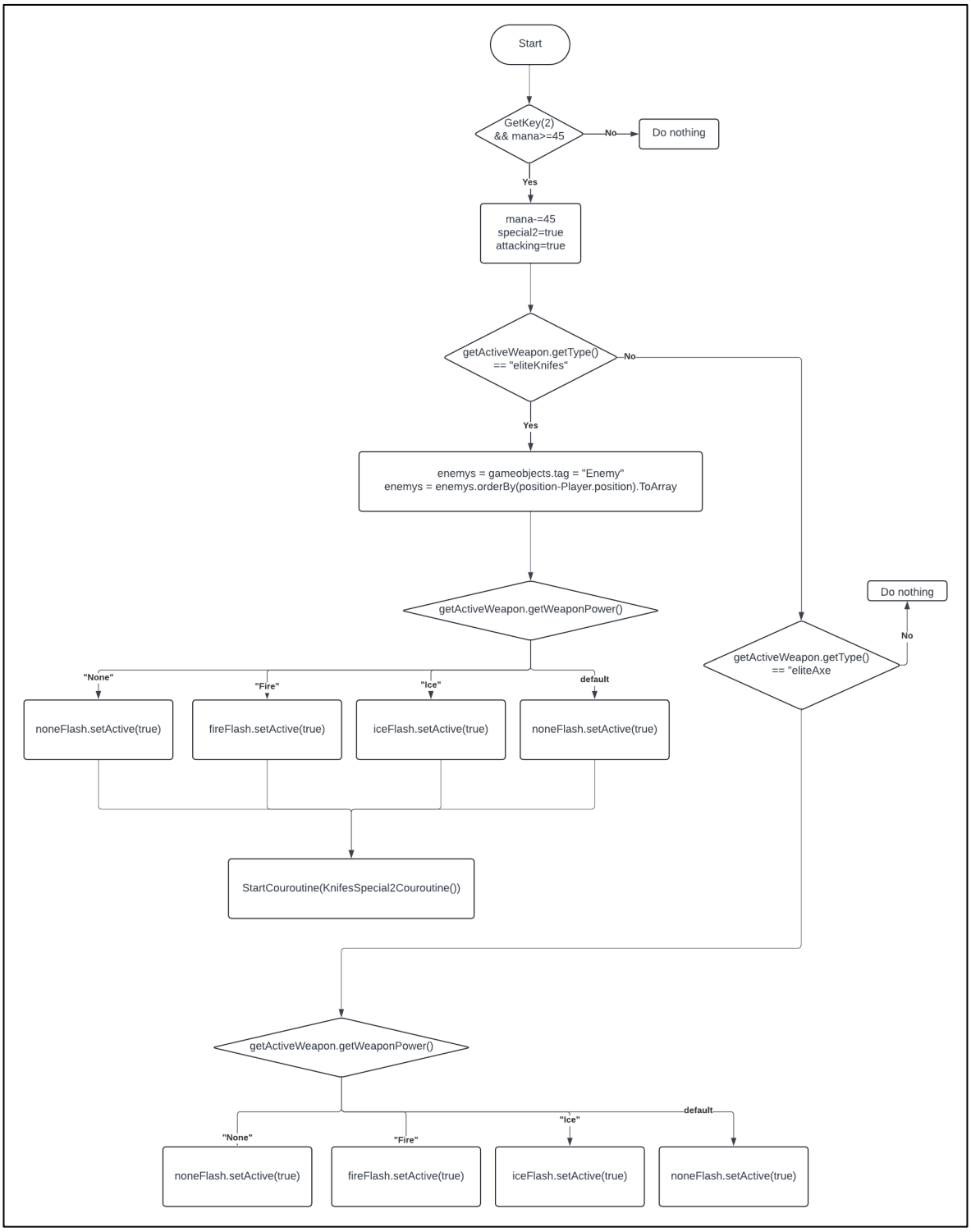


Figura 143 - Fluxograma do controlo da habilidade especial 2

Na Figura 143 é possível observar o fluxograma que representa o código presente na Figura 144 que demonstra o controlo da habilidade especial 2 de todos três tipos de armas do jogo. Primeiramente é verificado se o jogador clica no número 1 do seu teclado e se a barra de magia (“mana”) é maior ou igual a 45. Caso se verifique os requisitos é reduzida da barra de magia o valor 45 (custo da habilidade especial 2) e é ativa a animação da habilidade especial 2 pondo o personagem em modo de ataque. Em seguida é verificado se a arma ativa são as facas e caso se verifique é criado um *array*

com todos os inimigos presentes no nível em questão e, em seguida, é ordenado o *array* por ordem da sua distância em relação ao personagem. Posteriormente é verificado o tipo de poder da arma e ativado o efeito visual correspondente ao seu poder. Por fim é invocada a função de tempo "knifesSpecial2Couroutine".

Caso a arma ativa não seja as facas é verificado se é um machado. Caso a arma ativa seja um machado, verifica-se o seu tipo de poder e por fim é ativado o efeito especial correspondente ao seu tipo de poder.

```
if(Input.GetKey(KeyCode.Alpha2) && Stats.mana>=45){
    Stats.mana-=45;
    attack.SetBool("special2",true);
    Stats.attacking=true;
    if(GetComponent<weaponControll>().getActiveWeapon().GetComponent<weaponInfo>().getType() == "eliteKnifes"){
        enemys = GameObject.FindGameObjectsWithTag("Enemy");
        enemys = enemys.OrderBy(x => Vector3.Distance(x.transform.position,Player.transform.position)).ToArray();
        switch (transform.gameObject.GetComponent<weaponControll>().getActiveWeapon().GetComponent<weaponInfo>().getWeaponPower()){
            case "none":
                noneFlash.SetActive(true);
            break;
            case "fire":
                fireFlash.SetActive(true);
            break;
            case "ice":
                iceFlash.SetActive(true);
            break;
            default:
                noneFlash.SetActive(true);
            break;
        }
        StartCoroutine(knifesSpecial2Couroutine());
    }else if(GetComponent<weaponControll>().getActiveWeapon().GetComponent<weaponInfo>().getType() == "eliteAxe"){
        switch (transform.gameObject.GetComponent<weaponControll>().getActiveWeapon().GetComponent<weaponInfo>().getWeaponPower()){
            case "none":
                noneAxe2.SetActive(true);
            break;
            case "fire":
                fireAxe2.SetActive(true);
            break;
            case "ice":
                iceAxe2.SetActive(true);
            break;
            default:
                noneAxe2.SetActive(true);
            break;
        }
    }
}
```

Figura 144 - código controlo da habilidade especial 2

Caso a arma ativa seja uma espada apenas é ativada a animação de habilidade especial 2. Dentro da animação é invocada a função "spiralEffect" que verifica o poder da arma e ativa o efeito especial correspondente ao poder da arma ativa e, por fim, invoca a função de tempo "stopSpiralEffect".

```
public void spiralEffect(){
    switch (transform.gameObject.GetComponent<weaponControll>().getActiveWeapon().GetComponent<weaponInfo>().getWeaponPower()){
        case "none":
            noneSpiral.SetActive(true);
        break;
        case "fire":
            fireSpiral.SetActive(true);
        break;
        case "ice":
            iceSpiral.SetActive(true);
        break;
        default:
            noneSpiral.SetActive(true);
        break;
    }
    StartCoroutine(stopspiralEffect());
}
```

Figura 145 - Criação do efeito da habilidade especial 2 da espada

A função de tempo “stopSpiralEffect” ao ser invocada, primeiramente é dado ao personagem principal o destino que se encontra mais a posição frontal vezes 40 e é começado o movimento. O movimento é efetuado durante 2 segundos e em seguida são desativados todos os objetos de efeitos visuais. É desativada também a animação de habilidade especial 2, o modo de ataque do personagem principal e por fim desativado o movimento do personagem (Figura 146).

```
IEnumerator stopspiralEffect(){
    Player.GetComponent<NavMeshAgent>().destination = Player.transform.position + transform.forward*40;
    Player.transform.gameObject.GetComponent<NavMeshAgent>().isStopped=false;
    yield return new WaitForSeconds(2);
    noneSpiral.SetActive(false);
    fireSpiral.SetActive(false);
    iceSpiral.SetActive(false);
    attack.SetBool("special2",false);
    Stats.attacking=false;
    Player.GetComponent<NavMeshAgent>().destination = Player.transform.position;
    Player.transform.gameObject.GetComponent<NavMeshAgent>().isStopped=true;
}
```

Figura 146 - Função de tempo do efeito especial 2 da espada

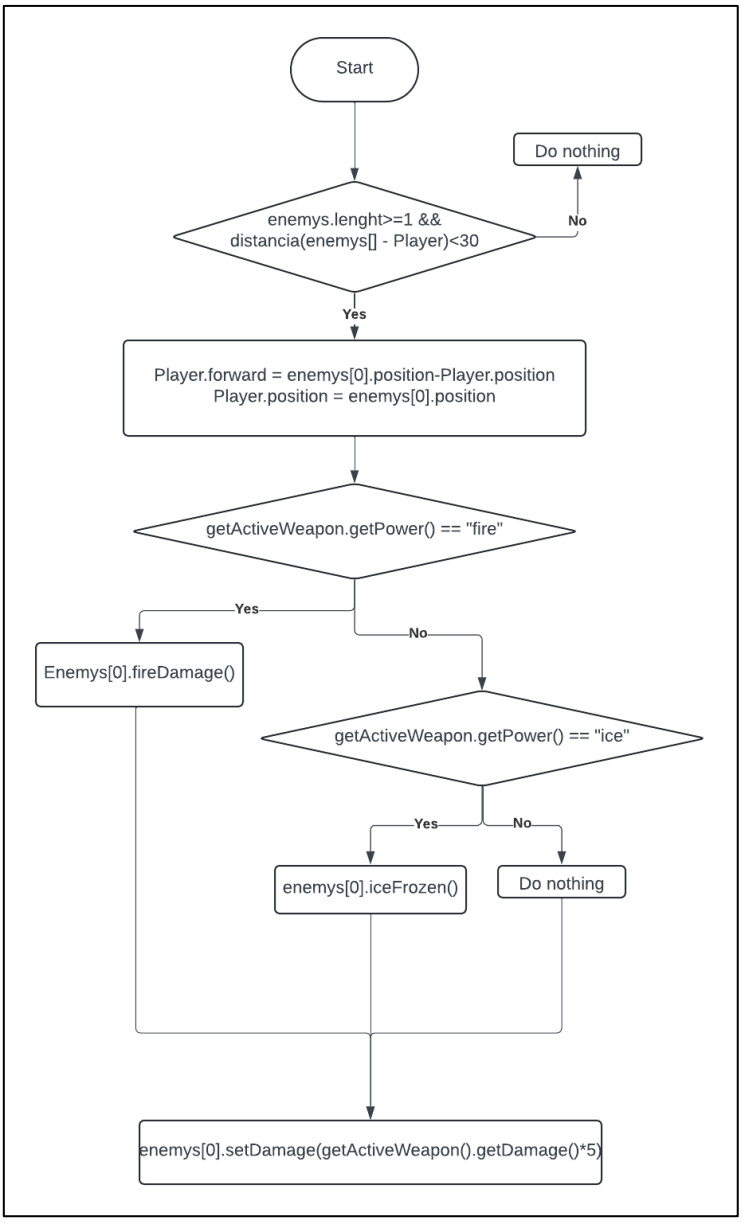


Figura 147 - Fluxograma do controlo da habilidade especial 2

Caso a arma ativa sejam as facas é invocada a função de tempo "knifesSpecial2Couroutine". O fluxograma presente na Figura 147 representa o código em A da Figura 148. A função começa por verificar se o *array* "enemys" tem tamanho igual ou superior a 1 e se a distancia entre o personagem principal e o inimigo na posição 0 do array "enemys" é menor que 30. Caso se comprove, é verificado se a arma ativa contém algum poder e é ativado o efeito no inimigo correspondente ao poder da arma. Caso não tenha qualquer tipo de poder é apenas dado o dano da arma vezes cinco.

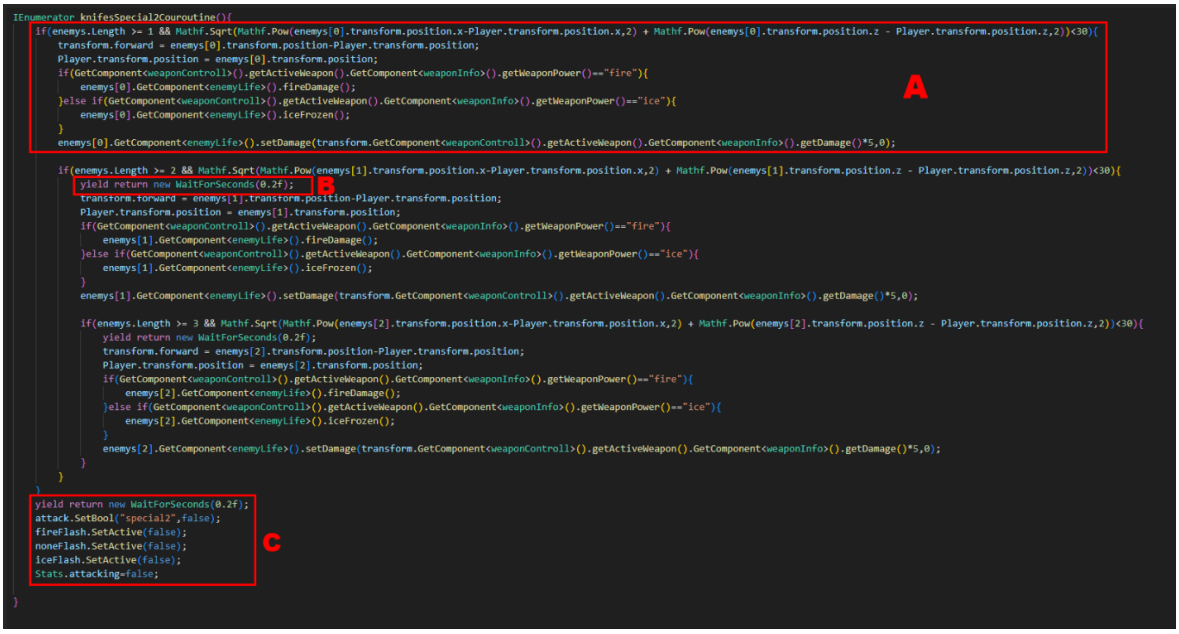


Figura 148 – Função de tempo da habilidade especial 2 das facas

O código presente em A da Figura 148 e explicado através do fluxograma na Figura 147 é repetido mais duas vezes, porém são feitas as verificações para o segundo e terceiro inimigo presente no *array* "enemys". É acrescentada uma linha de código que faz com que espere cerca de 0.2 segundos (B na Figura 148) e só depois ative os efeitos dos poderes e aplica o dano da arma. Por fim desativa a animação da habilidade especial 2 e são desativados os efeitos especiais da habilidade especial 2 e o modo de ataque (C na Figura 148).

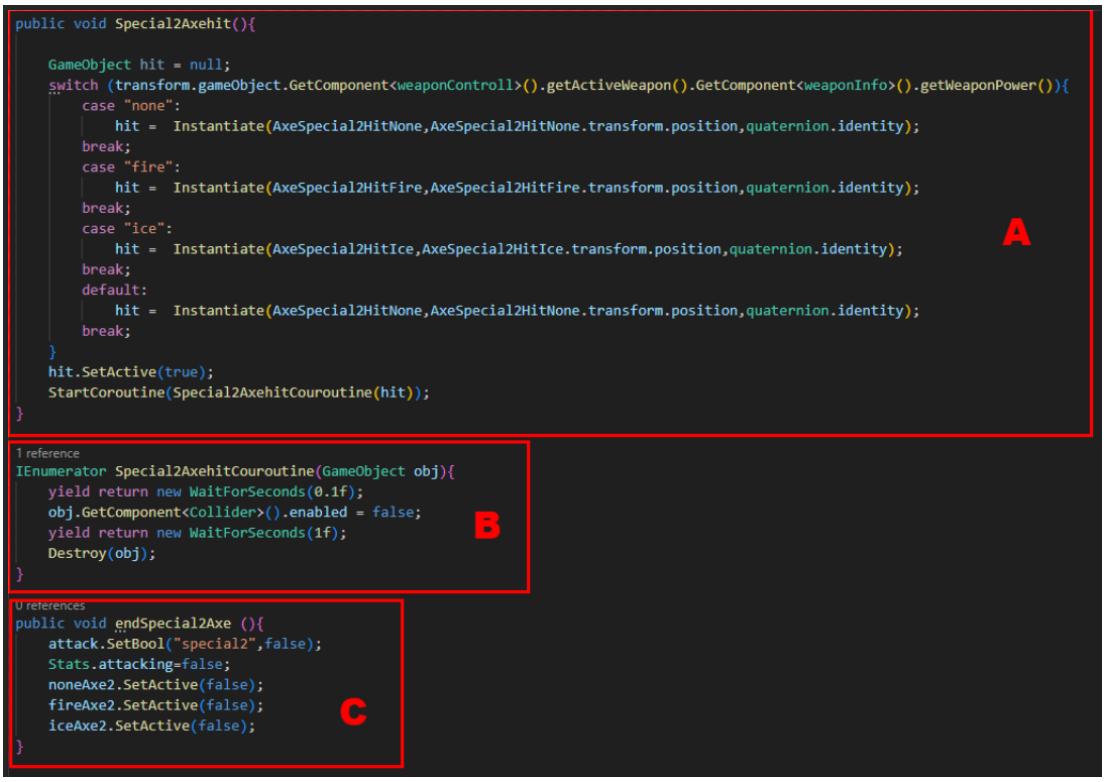


Figura 149 - Funções da habilidade especial 2 do machado

Por fim caso a arma ativa seja o machado, no fim da animação da habilidade é invocada a função "Special2Axehit" presente em A na Figura 149. Esta função tem como objetivo instanciar um objeto na posição onde o machado bate para poder dar dano aos inimigos que foram acertados por esta habilidade especial com o efeito visual correspondente ao poder da arma e, no final, invoca uma função de tempo "special2AxehitCouroutine" presente em B da Figura 149, que recebe o objeto anteriormente criado e ao fim de 0.1 segundos desativa o *collider* do efeito e após de mais 1 segundo destrói o objeto. Por fim ainda no término da animação é invocada a função "endSpecial2Axe" presente em C da Figura 149, que tem como objetivo terminar a animação da habilidade especial 2, desativar o modo de ataque e desativar os efeitos visuais anteriormente ativos.

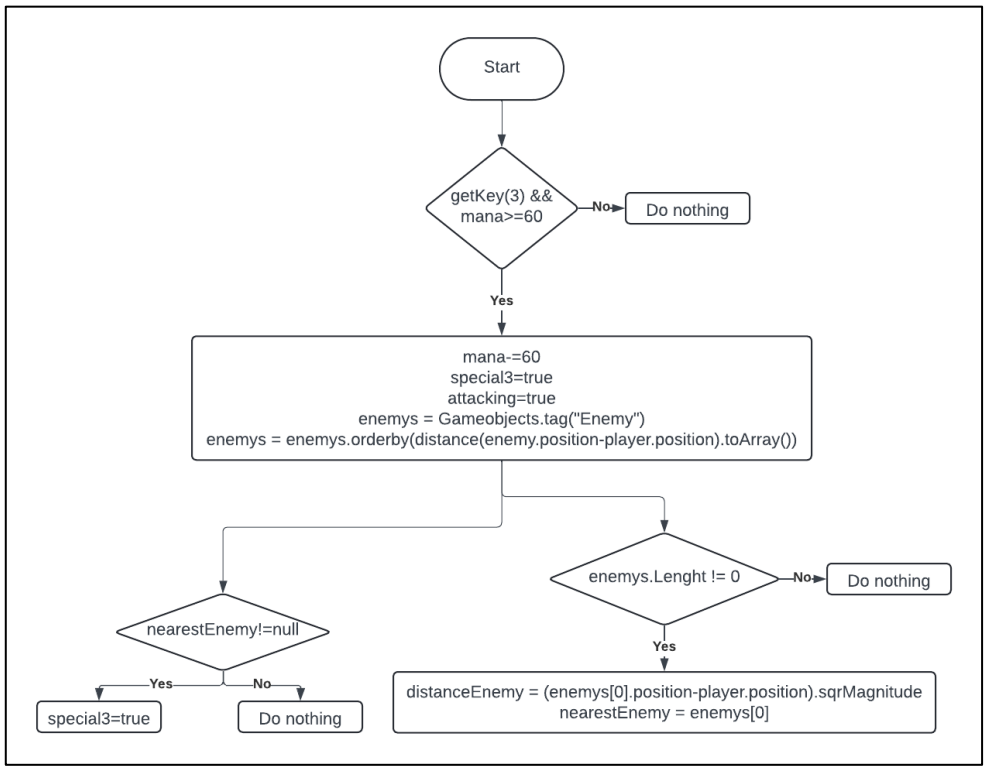


Figura 150 - Fluxograma do controlo da habilidade especial 3

Quando o jogador quiser executar a habilidade especial 3 do personagem basta clicar na tecla 3 do seu teclado que esta é executada caso a barra de magia (mana) seja 60 ou mais. No fluxograma representado na Figura 150 é possível observar essa verificação e caso seja concretizado esses requisitos é reduzida da barra de mana o valor 60, e ativada a animação da habilidade especial 3, colocando o personagem principal em modo de ataque. É criado também um *array* com todos os inimigos dentro do nível em que o personagem se encontra e, em seguida, é ordenado pela distância dos inimigos até ao personagem principal. Após isto é verificado se o tamanho do *array* é maior que 0 e caso se confirme, é calculada a distância do primeiro inimigo mesmo *array*. Por fim é verificado se o inimigo guardado não é *null* para poder ativar o modo de habilidade especial 3. O fluxograma representa o código apresentado na Figura 151.

```
if(Input.GetKey(KeyCode.Alpha3) && Stats.mana>=60){
    Stats.mana-=60;
    attack.SetBool("special3",true);
    Stats.attacking=true;
    enemys = GameObject.FindGameObjectsWithTag("Enemy");
    enemys = enemys.OrderBy(x => Vector3.Distance(x.transform.position,Player.transform.position)).ToArray();
    if(enemys.Length!=0){
        distanceEnemy = (enemys[0].transform.position - transform.position).sqrMagnitude;
        nearestEnemy = enemys[0];
    }
    if(nearestEnemy!=null){
        special3=true;
    }
}
```

Figura 151 - Código de controlo da habilidade especial 3

No decorrer da animação da habilidade especial 3 é ativada a função "explosionEffect" que irá instanciar o efeito visual do dano dado ao inimigo quando este é atingido. Em seguida é invocada a função de tempo "special3Couroutine" que recebe o efeito visual criado. Estas duas funções são possíveis de observar na Figura 152.

```
public void explosioneffect(){
    GameObject effect = null;
    switch (transform.gameObject.GetComponent<weaponControll>().getActiveWeapon().GetComponent<weaponInfo>().getWeaponPower()){
        case "none":
            effect = Instantiate(explosion,Player.transform.position + Player.transform.forward*2 ,quaternion.identity);
        break;
        case "fire":
            effect = Instantiate(fire_explosion,Player.transform.position + Player.transform.forward*2 ,quaternion.identity);
        break;
        case "ice":
            effect = Instantiate(ice_explosion,Player.transform.position + Player.transform.forward*2 ,quaternion.identity);
        break;
        default:
            effect = Instantiate(explosion,Player.transform.position + Player.transform.forward*2 ,quaternion.identity);
        break;
    }

    StartCoroutine(Special3Couroutine(effect));
}


1 reference
IEnumerator Special3Couroutine( GameObject effect){
    special3=false;
    Player.transform.gameObject.GetComponent<NavMeshAgent>().destination=Player.transform.position;
    Player.transform.gameObject.GetComponent<NavMeshAgent>().isStopped=true;
    if(nearestEnemy !=null && distanceEnemy< 200){
        nearestEnemy.GetComponent<enemyLife>().setDamage(2000,0);
        nearestEnemy = null;
    }
    yield return new WaitForSeconds(1.5f);
    Destroy(effect);
    attack.SetBool("special3",false);
    Stats.attacking=false;
}
```

Figura 152 - Funções da habilidade especial 3

Quando a variável "special3" é colocada em *true* e o inimigo mais próximo encontra-se a uma distância menor a 200 é possível observar no código presente na Figura 153 que o personagem principal fica virado sempre para o inimigo mais próximo e segue na sua direção pois é dado como destino a posição do inimigo mais próximo.

```
if(special3 && distanceEnemy<200){
    transform.forward = enemys[0].transform.position-Player.transform.position;
    Player.transform.gameObject.GetComponent<NavMeshAgent>().isStopped=false;
    Player.transform.gameObject.GetComponent<NavMeshAgent>().destination = nearestEnemy.transform.position;
}
```

Figura 153 - Controlo de distância da habilidade especial 3

No fluxograma da Figura 154 é possível observar a representação da função de tempo "special3Couroutine" presente na Figura 152. Esta função começa por colocar a variável "special3" como *false* dá ao personagem o seu mesmo destino para prevenir que se mova mais sozinho. Em seguida é verificado se o inimigo mais próximo não é *null* e se ele se encontra a uma distância inferior a 200, caso se aplique é dado ao inimigo um dano com o valor 2000 e colocada a variável que guarda o inimigo mais próximo com *null*. Por fim destrói o objeto criado com o efeito visual, desativa a animação de habilidade especial 3 e desativa o modo de ataque do personagem principal.

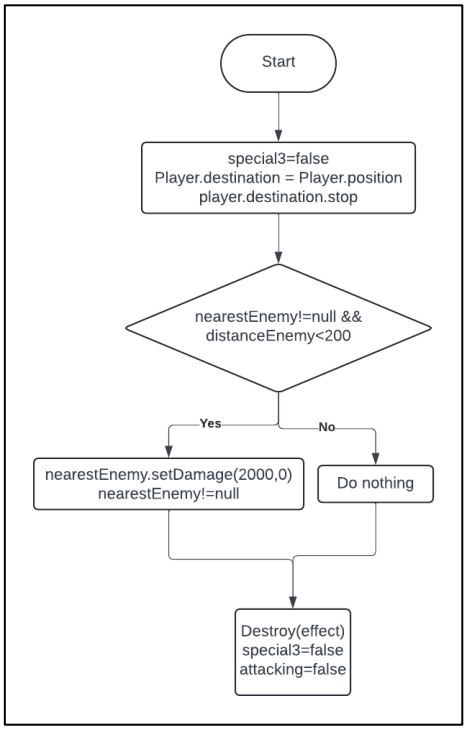


Figura 154 - Fluxograma da função de tempo "special3Couroutine"

4.2 Inimigos

Dentro do jogo existem cerca de sete inimigos e cada um com a suas características únicas que os torna diferentes, seja em tamanho, aspeto ou comportamento. Dentro deste subcapítulo será apresentado o código desenvolvido para cada um deles, apresentando um fluxograma seguido de uma descrição e por fim apresentando o código desenvolvido.

4.2.1 Controlo de vida dos inimigos

Dentro de cada inimigo, à exceção do inimigo final, foi desenvolvido um script que controla a vida dos inimigos com o nome "enemyLife". Este script é igual em todos os inimigos que o contêm o que faz com que seja possível aplicar dano de igual forma a todos os inimigos e controlar a sua vida.

Na Figura 155 é possível observar o início do script "enemyLife". Ao ser iniciado é guardado o objeto do personagem principal dentro e uma variável e o número do nível presente do nome da cena.

```
void Start()
{
    Player = GameObject.FindWithTag("Player");
    string sceneName;
    sceneName = SceneManager.GetActiveScene().name;
    level = int.Parse(sceneName[sceneName.Length-2]+""+sceneName[sceneName.Length-1]);
}
```

Figura 155 - Inicio do script "enemyLife"

Na função *update* presente na Figura 156 é possível observar que é verificada a vida do inimigo e caso esta chega a 0 ativa a animação de morte, caso contrário é feita a verificação de duas variáveis, sendo a primeira a variável "fire" que caso seja *true* é dado ao inimigo dano enquanto esta permaneça em *true*. A variável seguinte controla se o inimigo é empurrado ou não, caso a variável "push" seja *true* o inimigo é empurrado.

```
void Update()
{
    if(life<=0f){
        animator.SetBool("die",true);
    }else{
        if(fire){
            life-=2*Time.deltaTime;
        }
        if(push){
            transform.position += direction * pushForce*Time.deltaTime;
        }
    }
}
```

Figura 156 - Função update do script "enemyLife"

No fluxograma presente na Figura 157 representa o código da Figura 158. É possível observar que é verificado se o inimigo está em modo defesa e caso não esteja é aplicado o dano recebido à vida do inimigo e em seguida é calculada a direção na qual

o inimigo irá ser empurrado, dando o valor *true* à variável "push" e por fim é invocada a função de tempo "pushCouroutine";

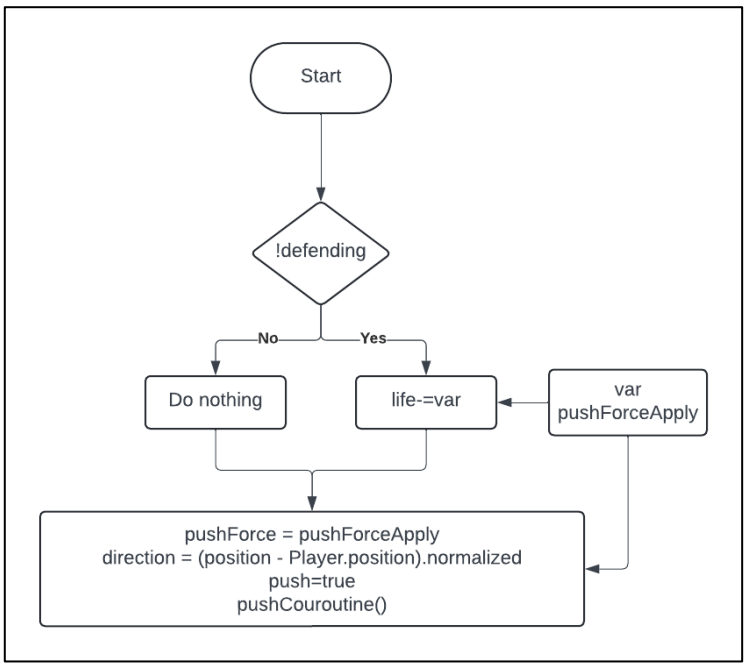


Figura 157 - Fluxograma do controlo de vida dos inimigos

Quando é invocada a função "pushCouroutine" presente no excerto de código da Figura 158 a função faz com que ao fim de 0.5 segundos a variável "push" volte ao valor *false*, ou seja, o inimigo permanece em movimento apenas durante 0.5 simulando o empurrão.

```
public void setDamage(float var,int pushForceApply){
    if(!defending){
        life-=var;
    }
    pushForce = pushForceApply;
    direction = (transform.position - Player.transform.position).normalized;
    push = true;
    StartCoroutine(pushCouroutine());
}

IEnumerator pushCouroutine(){
    yield return new WaitForSeconds(0.5f);
    push=false;
}
```

Figura 158 - Controlo de vida dos inimigos

Quando a vida do inimigo chega a 0 é ativada a animação de morte do inimigo. Esta animação serve para identificar que o inimigo está a morrer e no fim da mesma é invocada a função "spawnMaterial" presente na Figura 160.

Na Figura 159 é possível observar o fluxograma que representa essa função. A função começa por colocar o raio do inimigo a 0 para que o jogador não possa interagir com o mesmo durante a sua morte e é desativado o seu *collider*. É determino o destino do inimigo como a que se encontra nesse exato momento, colocando o inimigo em

modo stop e por fim instanciado o efeito de morte na posição do inimigo. Após terminar o efeito de morte, ou seja, esperar 2 segundos, são instanciados os materiais que o inimigo pode conter através da função “instantiateMaterial” que vai criar os materiais com base na probabilidade que lhe é atribuída. Em seguida é verificado se o nível está entre 5 e 10 para assim poder instanciar os cristais de fogo, pois esses 5 níveis pertencem ao bioma de fogo, e depois é verificado se o nível pertence aos 5 do bioma de gelo para poder instanciar os cristais de gelo em vez dos de fogo. Por fim é adicionado ao inventário do personagem principal o número de moedas por ter morto o inimigo e é adicionado também à barra de magia a quantia de magia proveniente do inimigo morto , por fim é destruído o objeto do efeito de morte assim como o inimigo.

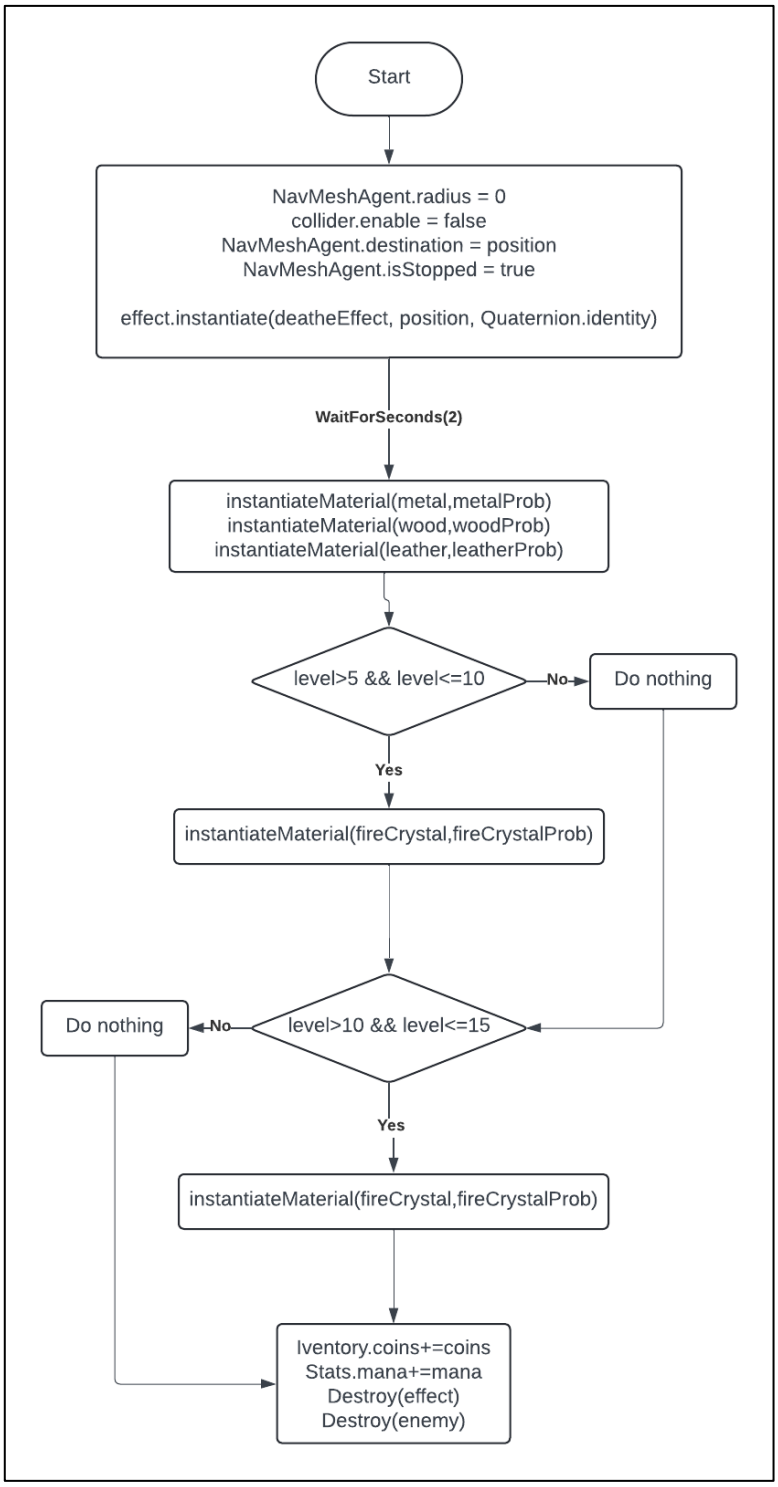


Figura 159 - Fluxograma do controlo de morte do inimigo

A função "instantiateMaterial" serve para instanciar os materiais com base na probabilidade que recebe, ou seja, é repetido entre 0 e a probabilidade recebida para assim instanciar os objetos dos materiais. Os materiais são instanciados em torno da posição do inimigo com um raio entre 0 e 2. É dado também a rotação de 90 graus em torno de x para que os objetos fiquem na posição vertical.

```
IEnumerator spawnMaterials(){
        GetComponent<NavMeshAgent>().radius = 0f;
        GetComponent<Collider>().enabled = false;
        GetComponent<NavMeshAgent>().destination = transform.position;
        GetComponent<NavMeshAgent>().isStopped = true;
    effect = Instantiate(deathEffect,transform.position,Quaternion.identity);
    yield return new WaitForSeconds(2);
    instantiateMaterial(metal,metalProb);
    instantiateMaterial(wood,woodProb);
    instantiateMaterial(leather,leatherProb);
    if(level>5 && level<=10)
        instantiateMaterial(fireCrystal,fireCrystalProb);

    if(level>10 && level<=15)
        instantiateMaterial(iceCrystal,iceCrystalProb);
    Inventory.coins+=coins;
    Stats.mana+=mana;
    Destroy(effect);
    Destroy(transform.gameObject);
}
private void instantiateMaterial(GameObject obj , int prob){
    for(int i = 0; i<Random.Range(0,prob);i++){
        Instantiate(obj,new Vector3(
            transform.position.x + Random.Range(-2,2),
            transform.position.y+2f,
            transform.position.z + Random.Range(-2,2)
        ),obj.transform.rotation = new Quaternion(-90f,0f,0f,0f));
    }
}
```

Figura 160 - Controlo da morte do inimigo

Quando o inimigo é colocado em fogo devido à arma do personagem principal ter esse mesmo poder, é invocada a função “fireDamage” que, caso o inimigo não esteja já sobre o efeito de fogo, é invocada a função de tempo “fireDamageCouroutine”. Ao invocar esta função de tempo a mesma ativa o efeito de fogo e coloca a variável “fire” com o valor *true* para definir que o inimigo está sobre o efeito de fogo para assim receber dano do mesmo e, ao fim de 5 segundos, é desativado este efeito e colocada a variável de fogo com o valor *false*. Quando a arma do personagem tem o poder de gelo, esta pode congelar os inimigos, ou seja, faz com que eles parem de se mexer por completo. Quando este efeito é ativado invoca a função “iceFrozen” que caso o inimigo não este já congelado, é invocada a função de tempo “iceFrozenCouroutine”. Esta função ativa o objeto que dá o efeito de congelado, coloca o inimigo neste modo dando o valor *true* para a variável “frozen”, desativa o *animator* para assim poder parar a animação e desativa a “NavMeshAgent”. Ao fim de 5 segundos volta a ativar tudo, desativa o objeto com o efeito de congelado e desativa o modo de congelado. Estas funções e o seu funcionamento são possíveis de observar na Figura 161.

```
public void fireDamage(){
    if(!fire)
        StartCoroutine(fireDamageCouroutine());
}

7 references
public void iceFrozen(){
    if(!frozen)
        StartCoroutine(iceFrozenCouroutine());
}

1 reference
public IEnumerator fireDamageCouroutine(){
    fireDamageEffect.SetActive(true);
    fire = true;
    yield return new WaitForSeconds(5);
    fireDamageEffect.SetActive(false);
    fire = false;
}
1 reference
public IEnumerator iceFrozenCouroutine(){
    frozenEffect.SetActive(true);
    frozen=true;
    animator.enabled=false;
    GetComponent<NavMeshAgent>().enabled = false;
    yield return new WaitForSeconds(5);
    animator.enabled=true;
    frozen=false;
    frozenEffect.SetActive(false);
    GetComponent<NavMeshAgent>().enabled = true;
}
```

Figura 161 - Controlo de efeitos aplicados por gelo e fogo

4.3 Controlo do range dos inimigos

Outro Script comum a todos os inimigos é o script “attackRange” que controla o range de ataque de cada inimigo. Este script consiste em três principais funções (“OnCollisionEnter”, ”OnCollisionStay”, ”OnCollisionExit”) que controlam se o personagem principal está dentro do range de ataque ou não.

A primeira função presente na Figura 162 verifica se o range de ataque colidiu com o personagem principal fazendo assim com que a variável “PlayerHit” adquira o valor *true*. A segunda função tem como objetivo verificar se o personagem principal se mantém dentro do range de ataque dando à variável “inside” o valor *true*. Por último a terceira função verifica se o personagem principal sai de contato com o range de ataque dando o valor *false* às duas variáveis anteriores. As quatro últimas funções apenas são usadas dentro de cada script pertencente aos movimentos de cada inimigo para poder adquirir o valor das variáveis anteriores ou atribuir-lhes valores.

```
0 references | Unity Message
void OnCollisionEnter(Collision other)
{
    if(other.gameObject.tag == "Player"){
        PlayerHit = true;
    }
}

0 references | Unity Message
void OnCollisionStay(Collision other)
{
    if(other.gameObject.tag == "Player"){
        Inside = true;
    }
}

0 references | Unity Message
void OnCollisionExit(Collision other)
{
    if(other.gameObject.tag == "Player"){
        PlayerHit = false;
        Inside=false;
    }
}

4 references
public bool getPlayerHit(){
    return PlayerHit;
}

7 references
public bool getPlayerInside(){
    return Inside;
}

4 references
public void setPlayerHit(bool var){
    PlayerHit=var;
}

4 references
public void setPlayerInside(bool var){
    Inside=var;
}
```

Figura 162 - Controlo do range dos inimigos

4.4 Comportamento dos inimigos

Quando se trata do comportamento que os inimigos seguem pode-se afirmar que todos têm o mesmo comportamento, salvo algumas exceções que serão vistas mais à

frente. Através do fluxograma presente na Figura 163 é possível observar que um inimigo começa por verificar se a sua vida se encontra com o valor maior que zero e se não se encontra em modo congelado. Caso estes requisitos se confirmem, é calculada a distância entre ele e o personagem principal e é guardado o valor numa variável. Em seguida é verificado se o personagem principal colide com o seu range de ataque, caso não se confirme é verificado se a distancia é menor que 30.

Se a distancia for maior que 30 não faz nada, apenas espera, caso seja maior que 30 é ativada a animação de caminhar, desativada qualquer animação de ataque é dado ao inimigo as coordenadas do personagem principal. Enquanto este caminha é constantemente atualizada a distância entre o personagem principal e o inimigo e simultaneamente verificado se o personagem principal colide com o range de ataque.

Caso o personagem colida com o range de ataque é desativada a animação de caminhar, o inimigo para na posição atual e é posto em modo de ataque passando o valor *true* para a variável "attaking". Em seguida é calculado um valor aleatório entre 0 e 100 e caso seja maior que 50 é ativada a animação de ataque 1 e dado à variável "damage" o valor de dano desse ataque, caso seja menor que 50 é ativada a animação de ataque 2 e dado à variável "damage" o valor de dano desse ataque.

No fim da animação de ataque é invocada a função "attack" que verifica primeiramente se o personagem principal se encontra dentro do range de ataque. Caso se encontre dentro do range é aplicado o dano da variável "damage" ao personagem principal, caso não esteja não aplica nada. Em seguida desativa o *collider* do inimigo, passa o valor false para a variável "inside" dentro do range de ataque e desativa qualquer animação de ataque ou caminhar. Ao fim de alguns segundos é desativado o modo de ataque e ativado o seu *Collider*. Por fim volta a verificar a distância e se o seu range colide com o personagem ou não, criando assim um ciclo até que a vida do inimigo chegue a 0 e este seja derrotado.

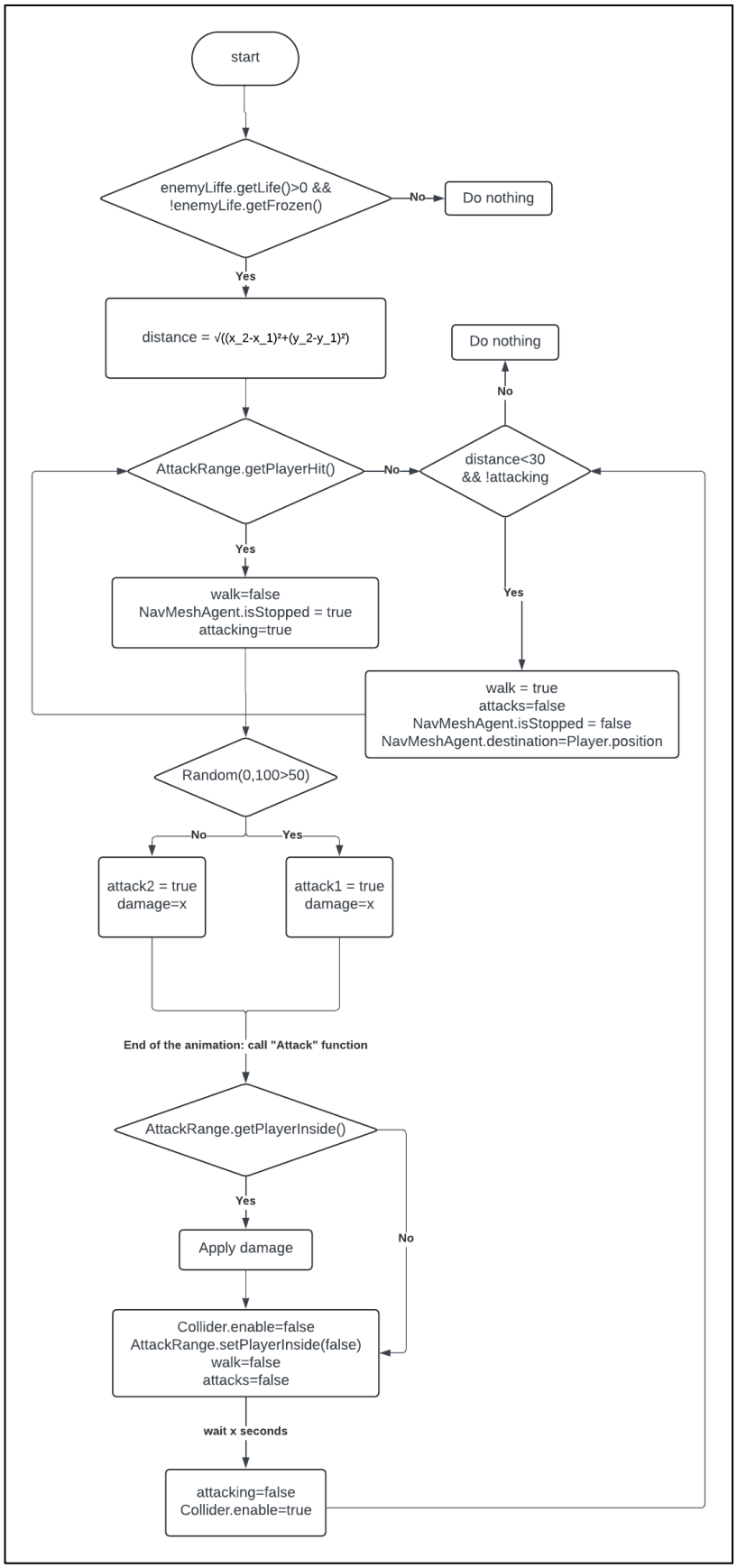


Figura 163 - Comportamento de um inimigo

Ao contrário dos outros inimigos, o *Bombist* quando verifica que a distância entre ele e o personagem principal é menor que 30, ativa a sua animação de correr e uma função de tempo com o nome "ExplosionCouroutine" (Figura 164) que irá criar a explosão da bomba. Quando invocada a função de tempo da explosão é ativado o efeito das faíscas do rastilho da bomba e ao fim de 15 segundos é instanciada a explosão na posição do inimigo. Por fim é invocada a função de ataque (igual aos outros inimigos) e destruído o objeto pertencente ao inimigo Bombist.

```
IEnumerator ExplodingCoroutine(){
    sparks[0].SetActive(true);
    sparks[1].SetActive(true);
    yield return new WaitForSeconds(15);
    Instantiate(explosion,new Vector3(
        transform.position.x,
        transform.position.y+2.75f,
        transform.position.z),
        Quaternion.identity
    );
    Attack();
    Destroy(gameObject);
}
```

Figura 164 - Ataque do inimigo Bombist

Ao contrário dos outros inimigos o inimigo Dark Knight apenas tem um ataque. Porém tem a particularidade que se poder defender sempre que que executa um ataque. Este comportamento pode ser observado através do fluxograma presente na Figura 165.

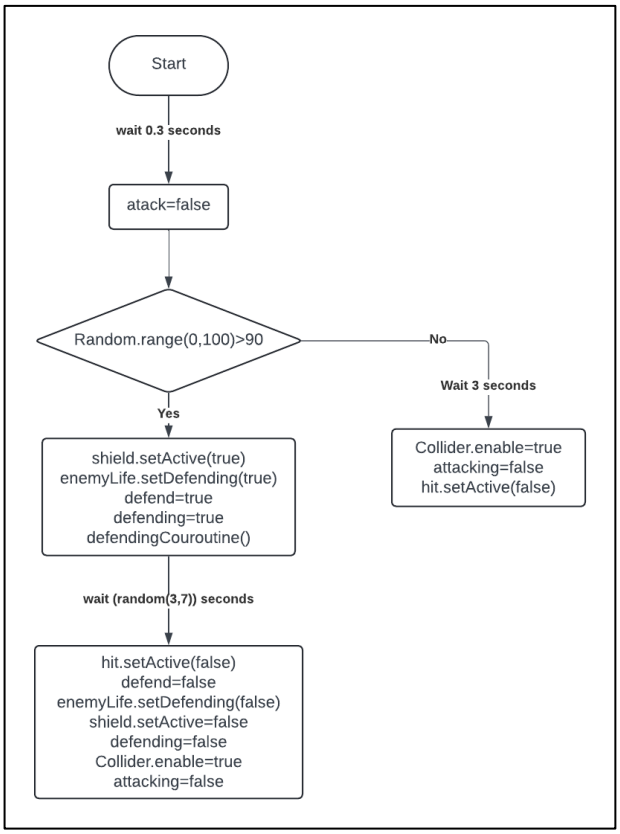


Figura 165 - Fluxograma do modo de defesa do DarkKnight

Quando é executado um ataque o Dark Knight desativa a animação de ataque e em seguida calcula um valor aleatório entre 0 e 100. Caso o valor seja maior que 90 é ativado o efeito de escudo, é passado o valor *true* para a variável "defending" dentro do script "enemyLife", é ativado a animação de defender e por fim é invocado a função de tempo "defendingCouroutine". Ao fim de passar entre 3 ou 7 segundos é desativo o efeito de *hit* caso este tenha sido ativo anteriormente, é desativada a animação de defender e a variável "defending" dentro do script "enemyLife", é ativado o *Collider* de novo e desativado o modo de ataque. Caso a defesa não seja ativada, o inimigo apenas espera 3 segundos no seu modo de parado, e passados esses três segundos é ativado o *Collider*, desativando o modo de ataque assim como o efeito de hit. O fluxograma presente na Figura 165 é uma representação do código presente na Figura 166.

```
IEnumerator waitNextAttack(){
    yield return new WaitForSeconds(0.3f);
    animator.SetBool("attack",false);
    if(Random.Range(0,100)>90f){
        shield.SetActive(true);
        transform.GetComponent<enemyLife>().setDefending(true);
        animator.SetBool("defend",true);
        defending=true;
        StartCoroutine(defendingCouroutine());
    }else{
        yield return new WaitForSeconds(3);
        rangeAttack.gameObject.GetComponent<Collider>().enabled = true;
        attacking=false;
        hit.SetActive(false);
    }
}

1 reference
IEnumerator defendingCouroutine(){
    yield return new WaitForSeconds(Random.Range(3,7));
    hit.SetActive(false);
    animator.SetBool("defend",false);
    transform.GetComponent<enemyLife>().setDefending(false);
    shield.SetActive(false);
    defending=false;
    rangeAttack.gameObject.GetComponent<Collider>().enabled = true;
    attacking=false;
}
```

Figura 166 - Modo defender do inimigo DarkKnight

Outro inimigo que é bastante único é o inimigo Mage (mago) que ao contrário dos outros inimigos não se move e fica parado a lançar os seus poderes. Através do fluxograma da Figura 167 é possível observar que também foi desenvolvido de forma diferente dos restantes inimigos.

Primeiramente começa por verificar se a sua vida é maior que 0 e se o efeito de congelado não está ativo. Caso os requisitos se verifiquem é calculada a distância e a sua diferença entre o personagem principal e a sua posição para assim ficar sempre virado na posição do personagem principal. Em seguida é verificado se o personagem principal se encontra dentro do range e caso se verifique é averiguado se este já está em modo de ataque 2 ou não para assim executar o mesmo. Caso o personagem principal não esteja dentro do seu range é verificado se ele se encontra a uma distância menor a 120 e caso esteja é apurado se já está em modo de ataque 1 para assim poder iniciar o seu primeiro ataque. O fluxograma presente na Figura 167 representa o excerto de código presente na zona da Figura 168.

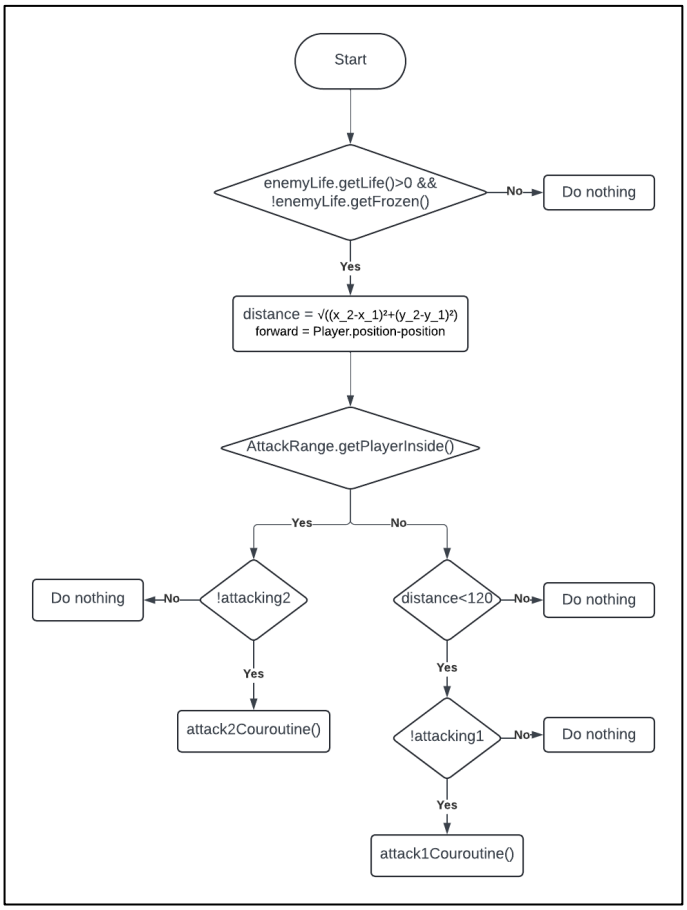


Figura 167 - Fluxograma de controlo de ataques do mago

Ao invocar uma das funções de tempo (“attack1Couroutine”, “attack2Couroutine”) irá fazer com que ative a animação de ataque correspondente, ativando o modo de ataque e ao fim de quatro segundos este é desativado. Estas funções podem ser observadas na zona B da Figura 168.

No final de cada animação é ativada uma função que irá criar o efeito correspondente ao ataque. No ataque 1 vai invocar a função “spawnEnergyBall” (zona C da Figura 168), esta função vai instanciar o objeto correspondente à bola de energia que irá seguir o personagem principal assim como os seus efeitos visuais. No final da animação do ataque 2 vai invocar a função “attack2Effect” (zona C da Figura 168) que irá ativar o efeito visual correspondente ao ataque 2.

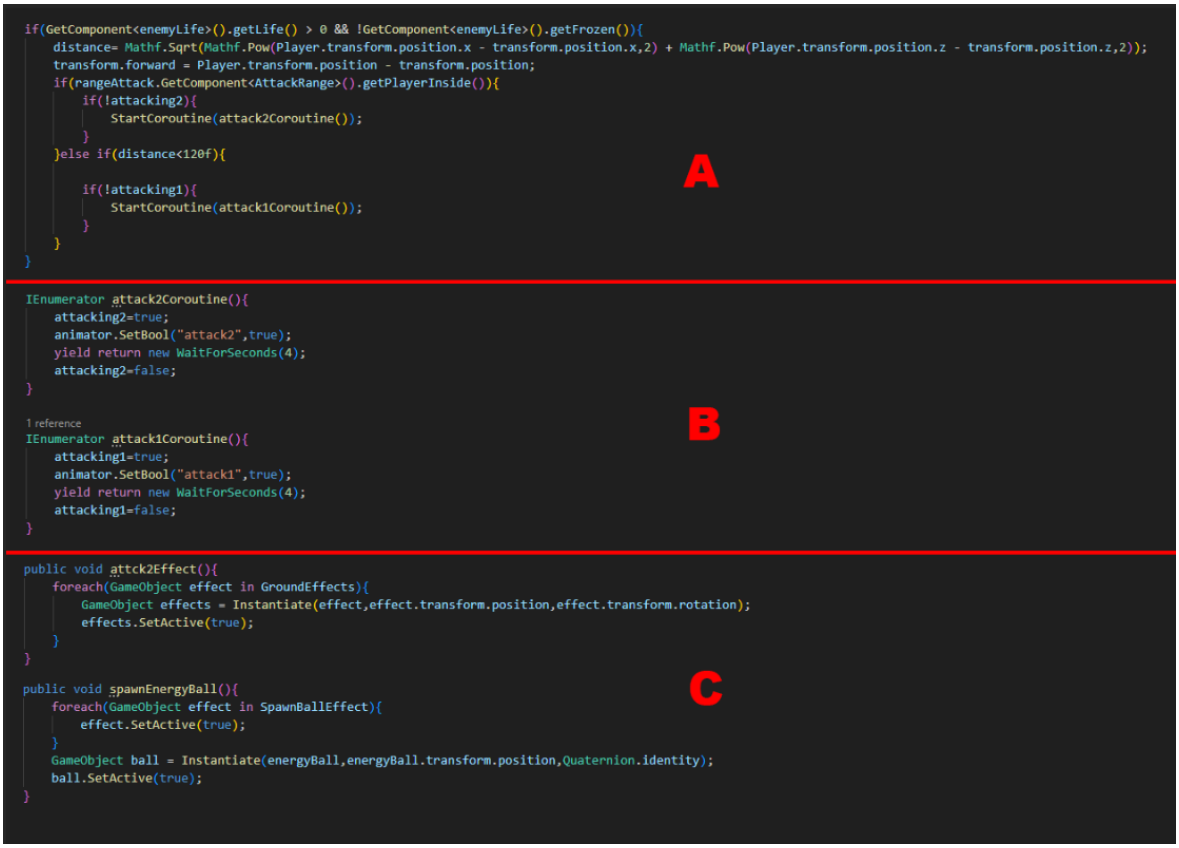


Figura 168 - Comportamento do mago

Dentro da bola de energia do mago foi desenvolvido um script que faz com que a bola siga o personagem principal até esta colidir com o mesmo. Quando colide com o personagem faz com que o mesmo receba dano e ative o efeito de explosão.

4.5 Inimigo final (Baelzor)

Para o jogo poder ser finalizado pelo jogador, o mesmo terá de entrar no castelo final e derrotar Baelzor para poder libertar todas as almas roubadas. Neste subcapítulo será explicado o funcionamento do inimigo final através de um fluxograma. Será também apresentado o código principal para o funcionamento de algumas mecânicas, como o começo da segunda fase de vida e o controlo da sua vida.

4.5.1 Funcionamento

Seguindo o fluxograma presente na Figura 169 é possível observar o funcionamento do inimigo final. Primeiramente é verificado se a sua primeira vida é maior que 0. Caso se confirme são ativadas as animações pertencentes ao estado de sentado. Em seguida é verificado se o inimigo está em modo de ataque que, caso se confirme negativo, é iniciado o modo de ataque verificando o ataque escolhido através da função *start* que escolhe de imediato o primeiro ataque a ser executado. O inimigo final pode executar cerca de 3 ataques diferentes pelo que dependendo do valor da variável é executado o ataque correspondente, caso a variável não se encontre entre 0 e 4 (não incluídos) é dado à variável "attackChose" um novo valor entre 0 e 4. Assim que é executado o ataque volta ao estado de verificar a sua primeira vida e por aí em diante.

Caso a primeira vida chegue a 0 ou abaixo, é começado a segunda fase de ataque que começa por verificar se a vida um se encontra menor ou igual a 0 e se a segunda vida se encontra maior que 0. Caso se verifique estas condições é começada a transição para a segunda fase que faz com que o inimigo final diminua e encha de novo a sua barra de vida. Esta transição só acontece uma única vez durante o jogo quando a variável "standingUp" se encontra com o valor *true*, depois de acabar a transição para a segunda fase, esta variável recebe o valor false e nunca mais volta atrás. Durante a segunda fase em seguida à transição é ativado o conjunto de animações que pertencem à segunda fase. Em seguida é verificado o ataque escolhido e assim como na primeira fase, o inimigo final poderá executar cerca de três ataques diferentes. *Após* executar um ataque volta-se a verificar a segunda vida e é repetido o ciclo até a segunda vida ser igual ou menor que 0. Quando a segunda vida chega a 0 é ativada a animação de morte e desativado o movimento do inimigo.

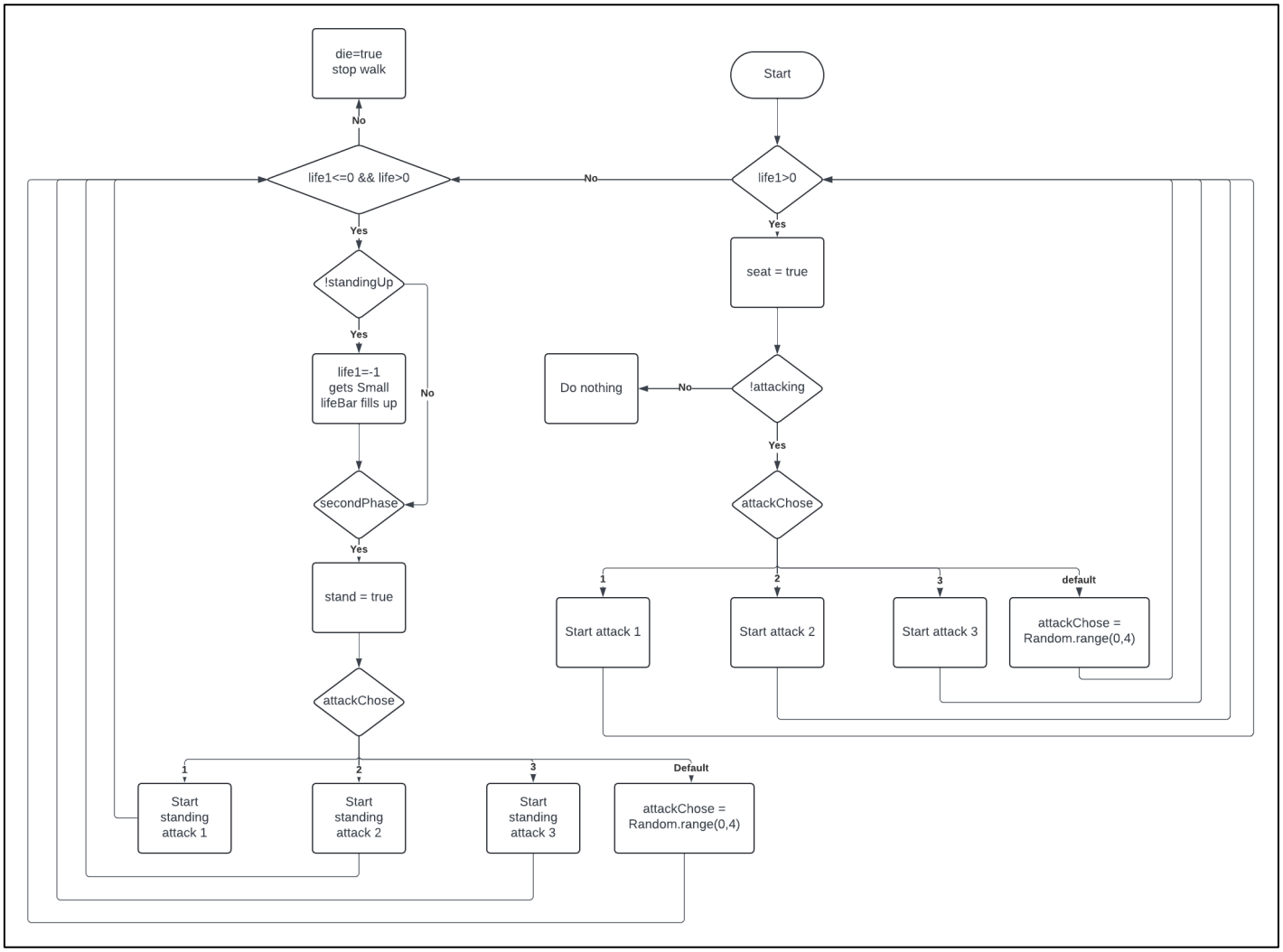


Figura 169 - Fluxograma sobre o funcionamento do inimigo final

4.5.2 Controlo da vida

Através do código presente na Figura 170 é possível observar que sempre que a função "setLife" é invocada, recebe o dano que irá ser aplicado ao inimigo final. Em seguida é verificado se a primeira vida é maior que 0 e caso seja o dano é aplicado à primeira vida. Logo de seguida é verificado se a primeira vida continua maior que 0 e caso não seja após ter sido aplicado o dano é dado início à segunda fase desativando todas as animações pertencentes à fase enquanto está sentado e desativado qualquer efeito dos ataques. Caso a primeira vida já se encontre abaixo de 0, o dano recebido é aplicado à segunda vida.

```
public void setLife(float damage){
    if(life1>0){
        life1-=damage;
        if(life1<=0){
            throneCollider.GetComponent<Collider>().enabled = false;
            phase1Collider.SetActive(false);
            phase2Collider.SetActive(true);
            animator.SetBool("attack1",false);
            animator.SetBool("attack2",false);
            animator.SetBool("attack3",false);
            fire.SetActive(false);
            StartCoroutine(startSecondPhase());
        }
    }else{
        life2-=damage;
    }
}
```

Figura 170 - Código de controlo de vida do inimigo final

Após a primeira vida do inimigo ter chegado a 0 é invocada uma função de tempo "startSecondPhase" presente no código da Figura 170. Na Figura 171 é possível observar essa função que controla a transição para a segunda fase de batalha do inimigo final. Esta função começa por desativar o grupo de animações pertencente à primeira fase de batalha e ativa animação de levantar, *após* dois segundos é dado o valor false para a variável "standingUp" para assim parar a diminuição do tamanho do inimigo final. Após quatro segundos e meio, que é o tempo para a barra de vida encher por completo, é dado inimigo a segunda fase de batalha, onde é ativado o grupo de animações pertencentes a esta segunda fasee desativado o modo de ataque caso não tivesse sido desativado anteriormente, e por fim é escolhido o primeiro ataque a ser executado.

```
IEnumerator startSecondPhase(){
    animator.SetBool("seat",false);
    animator.SetBool("standingUp",true);
    yield return new WaitForSeconds(2);
    standingUp=false;
    yield return new WaitForSeconds(4.5f);
    attacking = false;
    animator.SetBool("stand",true);
    secondPhase=true;
    attackChose = Random.Range(1,4);
}
```

Figura 171 transição para a segunda fase

Quando o inimigo final é derrotado, este começa a sua animação de morte o que faz com que sejam invocadas duas funções. A meio da animação é invocada o efeito das

almas (A da Figura 172) e por fim no final da animação é invocada a função que ativa o menu de vitória de jogo (B da Figura 172).

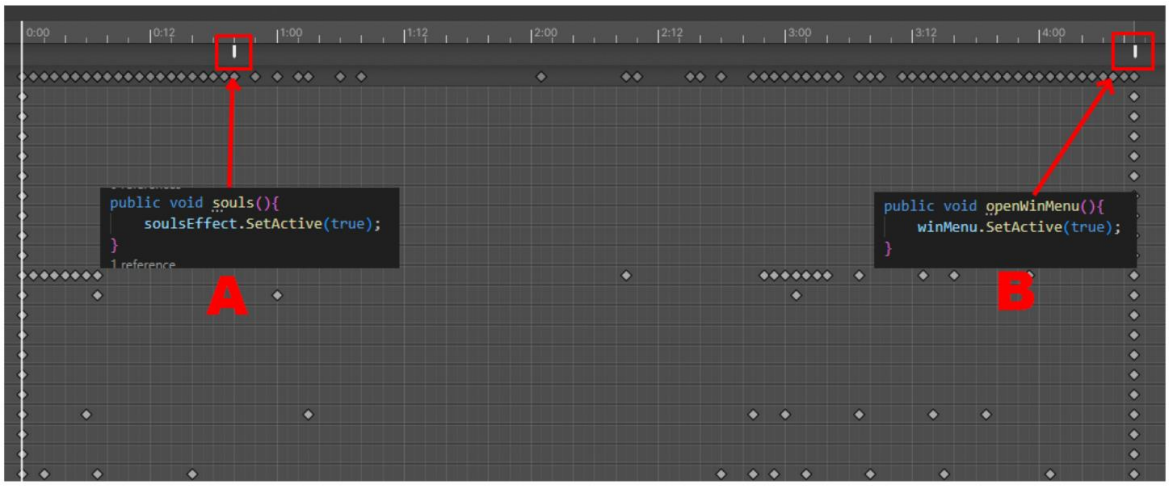


Figura 172 - Funções invocadas pela animação de morte

4.6 Controlo de armas

Neste subcapítulo será apresentado como é feita a gestão das armas dentro do jogo. Esta gestão é feita desde a loja onde são contruídas até ao inventário onde são controladas. Para essa gestão foi criado um script com o nome "weaponControll" que faz toda a gestão das armas.

Na Figura 173 é possível observar o código principal pelo controlo das armas dentro do jogo

Quando é iniciado o jogo ou cena é verificado o tamanho do *array* presente no script de inventário, é colocada a primeira arma como ativa assim como o grupo de animações pertencente à espada. Em seguida é percorrido o *array* que contém todas as armas do jogo e é verificado se algum ID de arma existe dentro do *array* das armas dentro do script do inventário. Caso alguma arma exista é criado um botão com a imagem da espada e é adicionado um evento ao botão criado para quando clicado ative a arma com o id correspondente.

```
weapondsSize = Inventory.weaponsId.Length;
weapons[0].SetActive(true);
animator.SetBool("eliteSword",true);
ActiveWeapond = weapons[0];
Player = GameObject.FindGameObjectWithTag("Player");

foreach(GameObject weapon in weapons){
    if(Inventory.weaponsId.Contains(weapon.GetComponent<weaponInfo>().getId())){
        GameObject go = Instantiate(button);
        go.transform.SetParent(panel.transform);
        go.transform.localScale = new Vector3(1,1,1);
        Button tempButton = go.GetComponent<Button>();
        //go.GetComponentInChildren<TextMeshProUGUI>().text = weapon.GetComponent<weaponInfo>().getName();
        go.GetComponent<Image>().sprite = weapon.GetComponent<weaponInfo>().getImage();
        tempButton.onClick.AddListener(() => setActiveWeapon(weapon.GetComponent<weaponInfo>().getId()));
    }
}
```

Figura 173 - Controlo das armas

Dentro de cada arma existe um script que recebe vários tipos de informações. Na Figura 174 é possível observar todos as informações que uma arma recebe para assim ser possível ter acesso às informações e ser possível ter um controlo das armas. As informações que cada arma são o seu tipo, tipo de poder, várias imagens pertencentes à arma, id, dano, a sua força de empurrão, o seu range e por fim o número de materiais necessários para ser possíveis construir a arma.

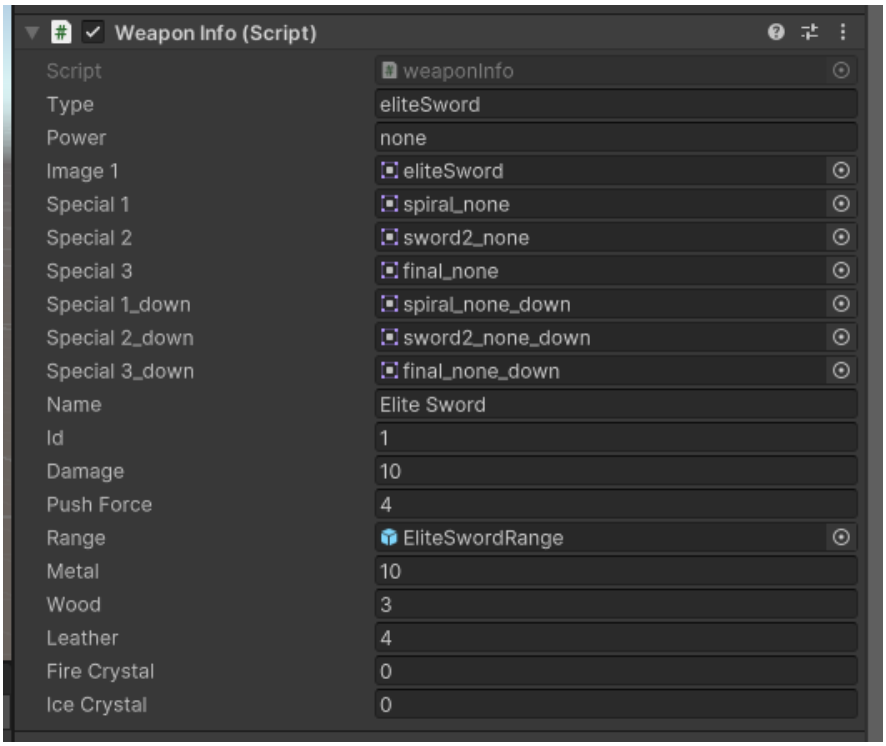


Figura 174 - Informação de cada arma

Visto que cada arma recebe as suas imagens, algumas destas imagens são as imagens pertencentes às habilidades especiais que aparecem no ecrã do jogador. Então dentro do script "weaponControll" também é feita essa gestão que verifica essas mesmas imagens dentro da arma ativa e coloca cada imagem das habilidades ativa ou

desativa consoante o valor da barra de magia. Essa gestão é possível de observar na Figura 175.

```
if(Stats.mana>=30){
    special1.sprite = ActiveWeapond.GetComponent<weaponInfo>().special1Image()[0];
}else{
    special1.sprite = ActiveWeapond.GetComponent<weaponInfo>().special1Image()[1];
}
if(Stats.mana>=45){
    special2.sprite = ActiveWeapond.GetComponent<weaponInfo>().special2Image()[0];
}else{
    special2.sprite = ActiveWeapond.GetComponent<weaponInfo>().special2Image()[1];
}
if(Stats.mana>=60){
    special3.sprite = ActiveWeapond.GetComponent<weaponInfo>().special3Image()[0];
}else{
    special3.sprite = ActiveWeapond.GetComponent<weaponInfo>().special3Image()[1];
}
```

Figura 175 - Controlo das imagens de habilidades especiais

Sempre que é criado um botão das armas construídas é adicionado um evento ao botão, esse evento permite ativar a arma que o jogador deseja usar. A função na Figura 176 recebe o id da arma que irá ficar ativa e percorre o *array* das armas e coloca ativa a arma com o mesmo id recebido. Esta função também ativa o grupo de animações correspondente à arma ativa.

```
public void setActiveWeapon(int id){
    foreach(GameObject weapon in weapons){
        if(weapon.GetComponent<weaponInfo>().getId() != id){
            weapon.SetActive(false);
        }else{
            weapon.SetActive(true);
                ActiveWeapond = weapon;
                animator.SetBool("attack1",false);
                animator.SetBool("attack2",false);
                animator.SetBool("attack3",false);
                animator.SetBool("eliteSword",false);
                animator.SetBool("eliteAxe",false);
                animator.SetBool("eliteKnifes",false);
                animator.SetBool(ActiveWeapond.GetComponent<weaponInfo>().getType(),true);
        }
    }
}
```

Figura 176 – Função de ativar arma

4.7 Controlo de poções

Para ser possível gerir as poções dentro do jogo foi criado um script com o nome “inventoryPotionControll”. Este script gere o número de poções dentro do inventário, mas também como aplica os efeitos das mesmas ao personagem principal.

Dentro do inventário existem os vários botões com as poções disponíveis ou indisponíveis (A na Figura 177) para serem usadas. Cada botão contém o script de controlo de poções que recebe vários parâmetros, sendo estes, o objeto que corresponde à imagem em G da Figura 177 que vai ser alterada dependendo de qual

poção seja clicada, recebe a imagem da poção(B na Figura 177)e a caixa de texto (C na Figura 177) onde vai aparecer o nome da poção, o botão que vai ser usado para o personagem principal consumir a poção (D na Figura 177) e por fim vai receber o seu nome assim como os efeitos que serão usados apenas na poção de velocidade e dano extra (E e F na Figura 177).

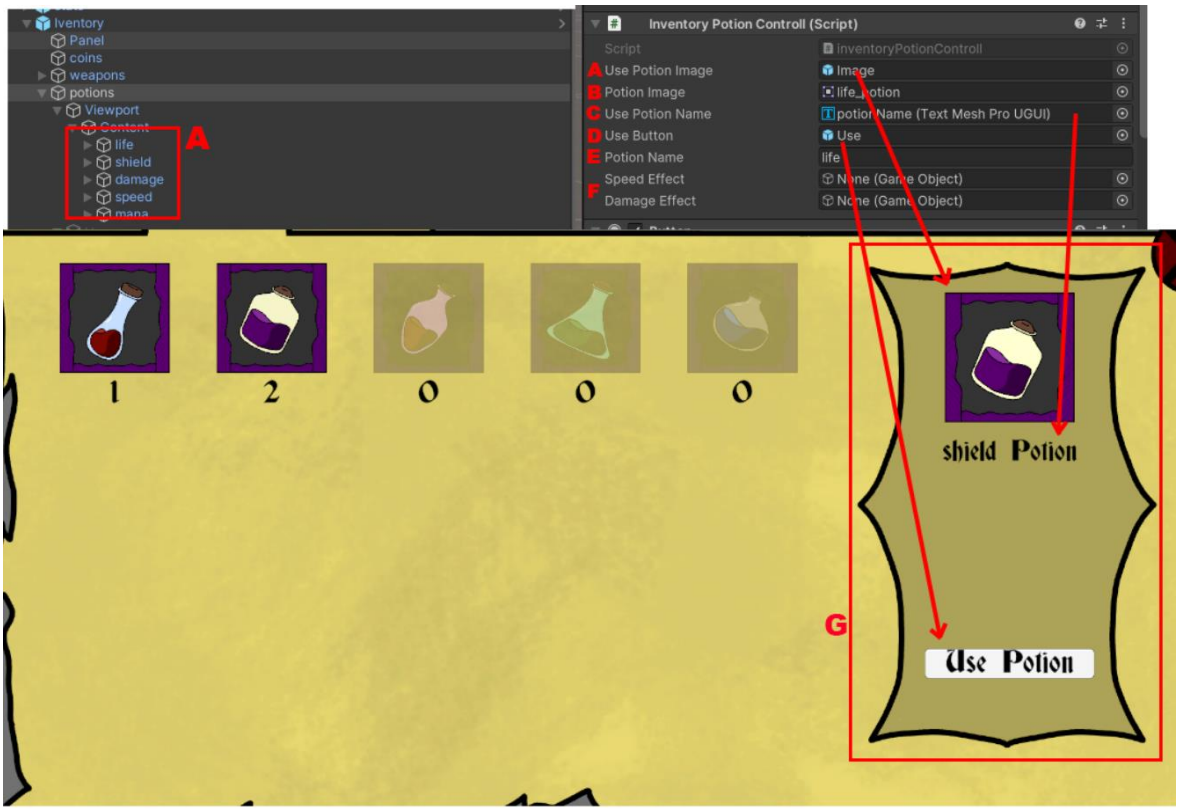


Figura 177 – Parâmetros do script " inventoryPotionControll "

Dentro do script "inventoryPotionControll" foi desenvolvida a função "atualizaPotion" que atualiza os botões sempre que que estes são clicados, ou seja, ativa os botões caso estes tenham no inventário do personagem principal uma ou mais poções correspondentes. No excerto de código da Figura 178 é possível observar que quando é pressionado uma poção é alterada a imagem, nome e é removido todos os eventos do botão de usar poção. Em seguida é feita uma verificação para ver qual o nome da poção que foi pressionada, no exemplo da Figura 178 é mostrado quando o nome da poção pressionada é a poção de vida ("life"). Na poção de vida é criado dois eventos para o botão de usar poção, o primeiro subtrai uma poção ao inventário e a segunda aplica o efeito da poção. Por último é passado uma *string* para o script "usePotion" que será visto mais à frente. Para a poção de magia e escudo o processo é o mesmo.

```
public void atualizaPotion(){
    usePotionImage.GetComponent<Image>().sprite = potionImage;
    usePotionName.text = potionName + " Potion";
    useButton.GetComponent<Button>().onClick.RemoveAllListeners();
    useButton.SetActive(true);
    usePotionImage.SetActive(true);
    usePotionName.gameObject.SetActive(true);
    switch(potionName){
        case "life":
            useButton.GetComponent<Button>().onClick.AddListener(() => Inventory.lifePotion--);
            useButton.GetComponent<Button>().onClick.AddListener(() => Stats.life += 100f);
            useButton.GetComponent<Button>().GetComponent<usePotion>().setPotionName("");
        break;
```

Figura 178 - Gestão de poções

Para as poções de dano extra e velocidade o processo de uso da poção é diferente. Na Figura 179 é possível observar que ao ser pressionada a poção de dano ("damage") é adicionado um evento de subtração de poção ao inventário como as outras poções, porém em seguida é invocada uma função de tempo que faz com que o efeito seja aplicado durante os segundos recebidos. É também adicionado um evento onde é desativado o botão de usar poção para assim não ser possível usar mais que uma poção com os mesmos efeitos ao mesmo tempo. Caso o personagem esteja com dano extra é passado o nome da poção para dentro do script "usePotion".

```
case "damage":
    useButton.GetComponent<Button>().onClick.AddListener(() => Inventory.damagePotion--);
    useButton.GetComponent<Button>().onClick.AddListener(() => StartCoroutine(damagePotionDurationCouroutine(45f)));
    useButton.GetComponent<Button>().onClick.AddListener(() => desativeButton("damage"));
    if(Stats.boostDamage)
        useButton.GetComponent<Button>().GetComponent<usePotion>().setPotionName("damage");
break;
```

Figura 179 - Gestão de poções com efeito

Em todos os casos é adicionado um último evento que vai invocar uma função pela qual vai receber o nome da poção (Figura 180) para que possa ser verificado se a poção clicada tem o valor maior que 0 dentro do inventário.

```
useButton.GetComponent<Button>().onClick.AddListener(() => verifyInventoryStock(potionName));
```

Figura 180 - Evento de verificação de quantidade de poções

Como dito anteriormente, quando é invocada a função "verifyIventoryStock" sempre que uma poção é usada, esta vai verificar se no inventário a quantidade da poção usada é maior que 0 e caso seja vai desativar todos os objetos presentes em G da Figura 177. Esta função pode ser observada na Figura 181 que contém um excerto da função apenas com a poção de vida ("life"), as restantes poções funcionam exatamente da mesma maneira.

```
public void verifyInventoryStock(string potionName){
    switch(potionName){
        case "life":
            if(Inventory.lifePotion == 0){
                useButton.SetActive(false);
                usePotionImage.SetActive(false);
                usePotionName.gameObject.SetActive(false);
            }
        break;
```

Figura 181 - Função de verificação de quantidade de poções

Como indicado anteriormente, sempre uma poção é pressionada é passado o nome da poção para dentro do script "usePotion". Dentro desse script é verificado se o nome corresponde a um dos dois presentes e caso se verifique o botão para usar a poção é desativado para prevenir o uso de várias poções com os mesmos efeitos. O código presente no script é possível de ser observado na Figura 182.

```
void Update()
{
    switch (potionName){
        case "damage":
            GetComponent<Button>().interactable = false;
        break;
        case "speed":
            GetComponent<Button>().interactable = false;
        break;
        default:
            GetComponent<Button>().interactable = true;
        break;
    }
}
```

Figura 182 - Script "usePotion"

4.8 Controlo de materiais

Para ser possível construir as armas dentro do jogo são necessários vários materiais como madeira, ferro e couro. Assim para o jogador saber a quantidade de materiais que o personagem principal contém no seu inventário foi criado uma aba dentro do inventário para esse efeito. Para controlar a quantidade de materiais existentes no inventário foi criado um script com o nome "materialInfo".

Dentro do jogo é possível observar através da Figura 183 que o jogador tem acesso à informação de todos os materiais existentes com a sua quantidade por baixo de cada imagem. Para ser possível obter a quantidade de cada material foi adicionado o script "materialInfo" às caixas de texto (retângulo vermelho da Figura 183).

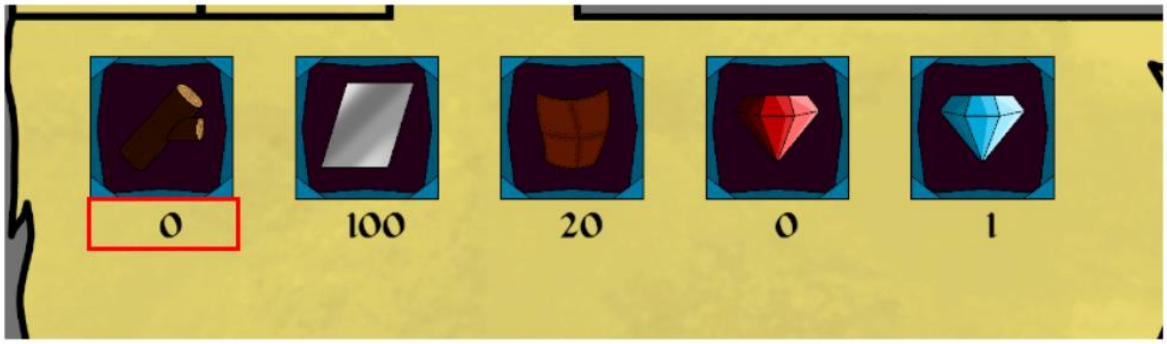


Figura 183 - Representação do inventário

Dentro do script é possível encontrar o código presente na Figura 184. Este script recebe um id pertencente ao material correspondente e é alterado o texto da caixa de texto consoante a quantidade do material. Este código está presente dentro da função "update" pelo que é atualizada em tempo real.

```
switch(id){
    case 1:
        text.text = "" + Inventory.metal;
    break;
    case 2:
        text.text = "" + Inventory.wood;
    break;
    case 3:
        text.text = "" + Inventory.leather;
    break;
    case 4:
        text.text = "" + Inventory.FireChristal;
    break;
    case 5:
        text.text = "" + Inventory.IceChristal;
    break;
}
```

Figura 184 - Código presente no script "materialInfo"

4.9 Lojas

Dentro do jogo existem duas lojas onde o jogador pode comprar diversos objetos. Estas duas lojas são a loja de poções e de materiais, presentes em vários níveis dentro do jogo. Estas lojas têm um funcionamento idêntico, e o código presente nos scripts também é idêntico em ambas as lojas. Neste subcapítulo será descrito o funcionamento de uma loja assim como o código desenvolvido para o funcionamento da mesma através de fluxogramas.

Na Figura 185 é possível observar que uma loja tem três principais zonas. A primeira zona (A na Figura 185) contém os botões que iram dar ao jogador a informação sobre o material escolhido. A segunda zona (B na Figura 185) representa o número de moedas que o jogador tem acumuladas durante o jogo. Por último a *última*

zona (C na Figura 185) representa o local onde vai aparecer a informação sobre o material que o jogador clicou assim como o botão para poder comprar.

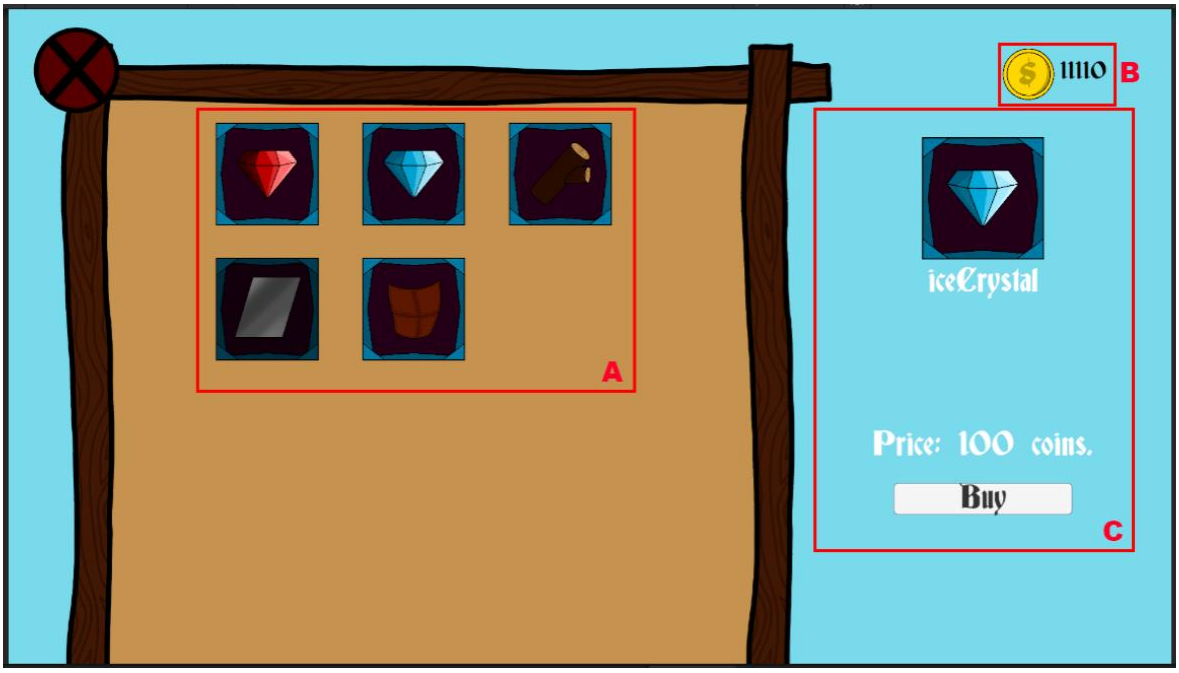


Figura 185 – Representação de uma loja

Cada botão dentro da zona A da Figura 185 recebe o script pertencente à loja respetiva que recebe como parâmetros os objetos presentes na zona C da Figura 185 como é possível observar nos retângulos vermelhos da Figura 186. Os restantes parâmetros pertencem à informação do objeto, como a sua imagem, preço e nome.

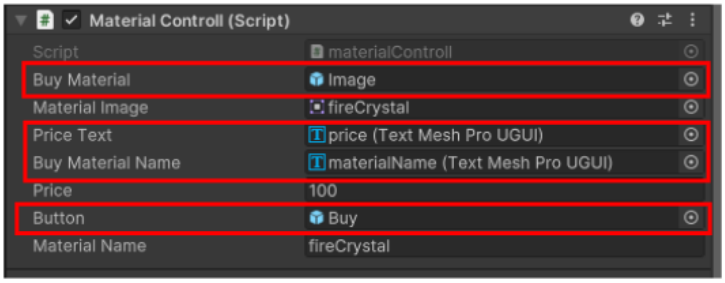


Figura 186 - Parâmetros do script de uma loja

Como dito anteriormente cada botão presente na zona A da Figura 185 recebe um script e dentro desse script existe uma função que é invocada sempre que o jogador pressiona uma das opções para comprar. A função é possível observar na Figura 187 com o nome “atualizaMaterial” (este script por pertencer à loja de materiais, o nome da função remete a essa loja, na loja de poções o nome da função chama-se “atualizaPotion”).

```
public void atualizaMaterial(){
    buyMaterial.GetComponent<Image>().sprite = materialImage;
    priceText.text = "Price: " + price + " coins.";
    buyMaterialName.text = materialName ;
    button.GetComponent<Button>().onClick.RemoveAllListeners();
    button.SetActive(true);
    buyMaterial.SetActive(true);
    priceText.gameObject.SetActive(true);
    buyMaterialName.gameObject.SetActive(true);
    if(Inventory.coins<price){
        button.GetComponent<Button>().interactable = false;
    }else{
        button.GetComponent<Button>().interactable = true;
        button.GetComponent<Button>().onClick.AddListener(() => button.SetActive(false));
        button.GetComponent<Button>().onClick.AddListener(() => buyMaterial.SetActive(false));
        button.GetComponent<Button>().onClick.AddListener(() => priceText.gameObject.SetActive(false));
        button.GetComponent<Button>().onClick.AddListener(() => buyMaterialName.gameObject.SetActive(false));
        switch(name){
            case "wood":
                button.GetComponent<Button>().onClick.AddListener(() => Inventory.wood++);
                button.GetComponent<Button>().onClick.AddListener(() => Inventory.coins-=price);
            break;
```

Figura 187 - Código de controlo de loja

Através do fluxograma presente na Figura 188 é possível observar que quando é invocada a função são ativados os objetos da zona C da Figura 185 e recebem a informação pertencente ao objeto pressionado pelo jogador. Em seguida é verificado se o número de moedas é suficiente para comprar o objeto e caso o jogador tenha moedas insuficientes o botão de comprar é desativado, caso seja suficiente é adicionado vários eventos ao botão que fazem com que os objetos da C da Figura 185 sejam desativados quando o jogador compra um material. Em seguida é verificado o nome do objeto pressionado (retângulo vermelho da Figura 187) e é dado ao botão mais dois eventos, um que adiciona o objeto ao inventário e outro que subtrai o preço do objeto às moedas do jogador.

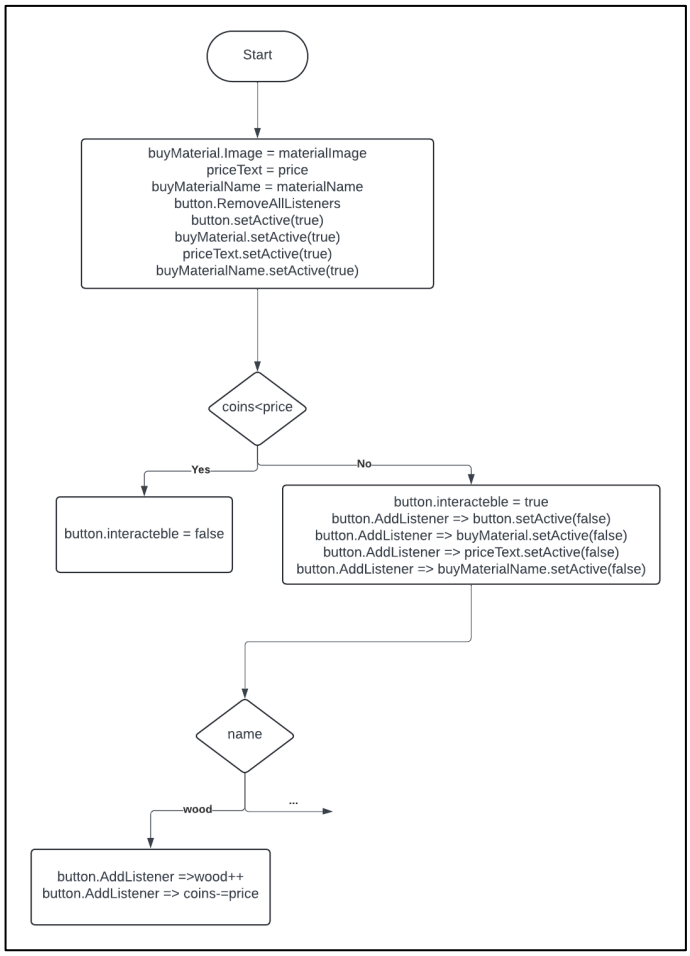


Figura 188 - Fluxograma controlo de uma loja

4.10 Forja

Dentro do jogo o jogador tem acesso a várias armas para poder usar para derrotar os inimigos, porém para conseguir obter essas armas o jogador tem de coletar vários materiais e dirigir-se à uma forja para as conseguir construir. Neste subcapítulo é descrito o controlo da forja assim como o seu funcionamento através do código desenvolvido.

Na Figura 189 é possível observar que a forja é constituída por dois principais grupos. O primeiro grupo constituído pelos botões pertencentes às armas disponíveis para construção, que se encontra na zona A da Figura 189 e o segundo grupo (zona B da Figura 189) que é constituído pelos objetos que dão informação sobre a arma escolhida, ou seja, os materiais necessários para o efeito, é também constituído também pelo botão de construção de arma.

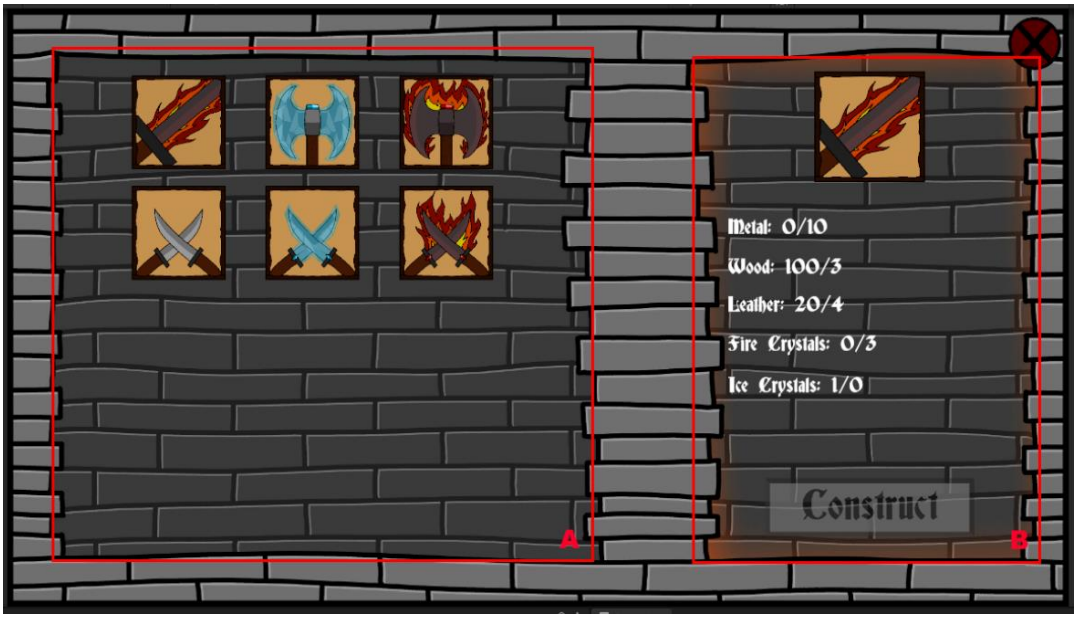


Figura 189 - Elementos da forja

Dentro de cada botão da zona A da Figura 189 contém o script "construckInfo" que recebe vários objetos como parâmetros. Na Figura 190 é possível observar os vários objetos que o script recebe sendo que o primeiro (A da Figura 190) pertence à arma que se pretende construir e os restantes (B da Figura 190) pertencem aos objetos da zona B da Figura 189.

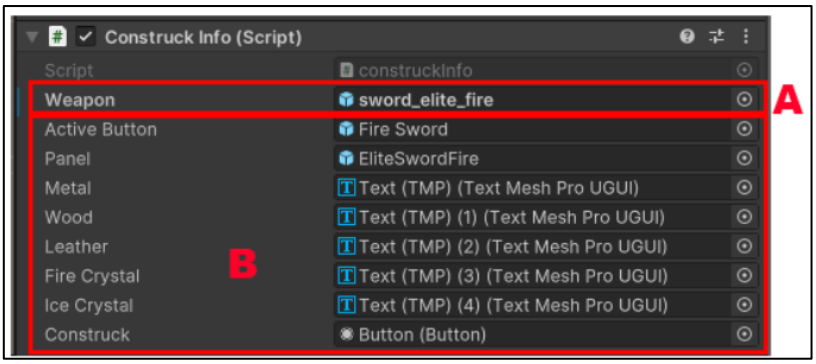


Figura 190 - Parâmetros do script "construckInfo"

Para ser possível mostrar os objetos respetivos à informação de construção das armas é usado a função "OnClick" dos botões das armas para ativar ou desativar esses mesmos objetos. A zona A da Figura 191 corresponde aos botões presentes na zona A da Figura 189, que, quando pressionados, (B da Figura 191) ativam um conjunto de objetos com informação (D da Figura 191).

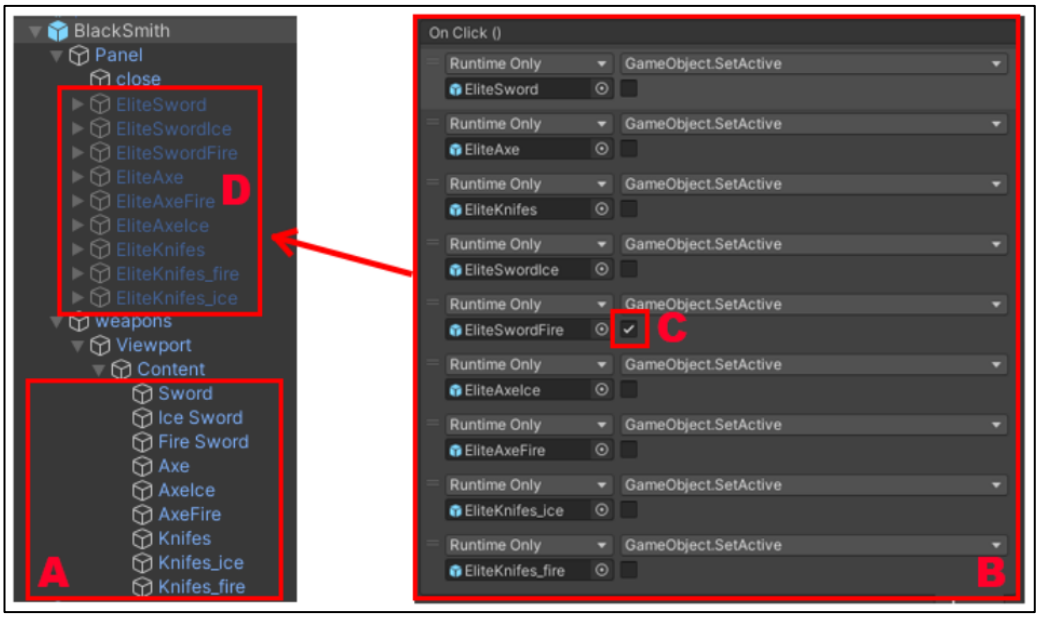


Figura 191 - Controlo de armas na forja

Na Figura 192 é possível observar a função "Update" que irá atualizar as caixas de texto com a informação correspondente para construção da arma pressionada através da zona A. Na zona B da Figura 192 é feita a verificação se os materiais presentes no inventário são suficientes para a construção da arma e, caso sejam, o botão de construção fica ativo, caso não sejam o botão de construção fica desativo não sendo possível clicar no mesmo. Por último é feita a verificação se a arma já existe ou não dentro do inventário do personagem principal e caso já esteja o botão é destruído o que impossibilita a construção de armas repetidas.

```
void Update()
{
    metal.text = "Metal: " + Inventory.metal + "/" + weapon.GetComponent<weaponInfo>().getMetal();

    wood.text = "Wood: " + Inventory.wood + "/" + weapon.GetComponent<weaponInfo>().getWood();

    leather.text = "Leather: " + Inventory.leather + "/" + weapon.GetComponent<weaponInfo>().getLeather();

    fireCrystal.text = "Fire Crystals: " + Inventory.FireChristal + "/" + weapon.GetComponent<weaponInfo>().getFireCrystal();

    iceCrystal.text = "Ice Crystals: " + Inventory.IceChristal + "/" + weapon.GetComponent<weaponInfo>().getIceCrystal();
    A
    if(Inventory.metal >= weapon.GetComponent<weaponInfo>().getMetal() &&
        Inventory.wood >= weapon.GetComponent<weaponInfo>().getWood() &&
        Inventory.leather >= weapon.GetComponent<weaponInfo>().getLeather() &&
        Inventory.FireChristal >= weapon.GetComponent<weaponInfo>().getFireCrystal() &&
        Inventory.IceChristal >= weapon.GetComponent<weaponInfo>().getIceCrystal()){
        construck.interactable = true;
    }else{
        construck.interactable = false;
    }
    B
    if(Inventory.weaponsId.Contains(weapon.GetComponent<weaponInfo>().getId())){
        Destroy(transform.gameObject);
    }
    C
}
```

Figura 192 - Código de controlo da informação das armas para construção

Dentro do botão de construir é invocada a função "ConstruckWeapon" que faz com que a arma seja construída. Quando a função é invocada é subtraído o número de materiais usados na construção do inventário do personagem principal, e é adicionado

o id da arma construída ao inventário, por fim são desativados os objetos de informação de construção.

```
public void constructWeapon(){
    Inventory.metal -= weapon.GetComponent<weaponInfo>().getMetal();

    Inventory.wood -= weapon.GetComponent<weaponInfo>().getWood();

    Inventory.leather -= weapon.GetComponent<weaponInfo>().getLeather();

    Inventory.FireChristal -= weapon.GetComponent<weaponInfo>().getFireCrystal();

    Inventory.IceChristal -= weapon.GetComponent<weaponInfo>().getIceCrystal();

    int[] Array = {weapon.GetComponent<weaponInfo>().getId()};
    int[] newArray = Inventory.weaponsId.Concat(Array).ToArray();

    Inventory.weaponsId = newArray;

    panel.SetActive(false);
}
```

Figura 193 - Código de construção de arma

4.11 Controlo de níveis

Dentro do jogo existem diversos níveis para o jogador poder explorar e coletar recursos para a construção das suas armas e obtenção de moedas para comprar poções. Para ser possível passar de nível e ter acesso a níveis anteriores foram desenvolvidos vários *scripts* que permitem ao jogador navegar pelos níveis livremente e sem qualquer restrição. Neste subcapítulo será apresentado tudo o que foi desenvolvido para tornar o acesso aos níveis livres.

Para ser possível a navegação entre níveis primeiramente foi preciso construí-los um por um e cada um com um design único. Para isso foi instalado um *package* que permite fixar os objetos a uma grelha invisível para tornar a construção de níveis mais fácil. Na Figura 194 é possível ver o package instalado com o nome “ProGrids” (A na Figura 194). Este package cria uma grelha invisível no mundo 3d que permite focar os objetos em coordenadas fixas. Para controlar o tamanho da grelha é possível através dos diversos botões que aparecem na tela como é possível observar na zona B da Figura 194. Exemplificando, para desenvolvimento dos níveis foram usados *assets* que têm a forma de um quadrado e cada lado tem o valor 26 e sendo que a origem do quadrado é no seu centro e o primeiro quadrado situa-se nas coordenadas (0,0). O seguinte teria de ficar separado pelo valor 26, ficando nas coordenadas (0,26). Ao colocar a grelha com tamanho 26 permite que os objetos sejam colocados na coordenada certa sem ser preciso ajustar manualmente cada coordenada de cada *asset*. As restantes opções apenas fazem com que essa grelha seja visível ou não, e decidir se a função de prender os *assets* automaticamente à coordenada esteja ativa ou não.

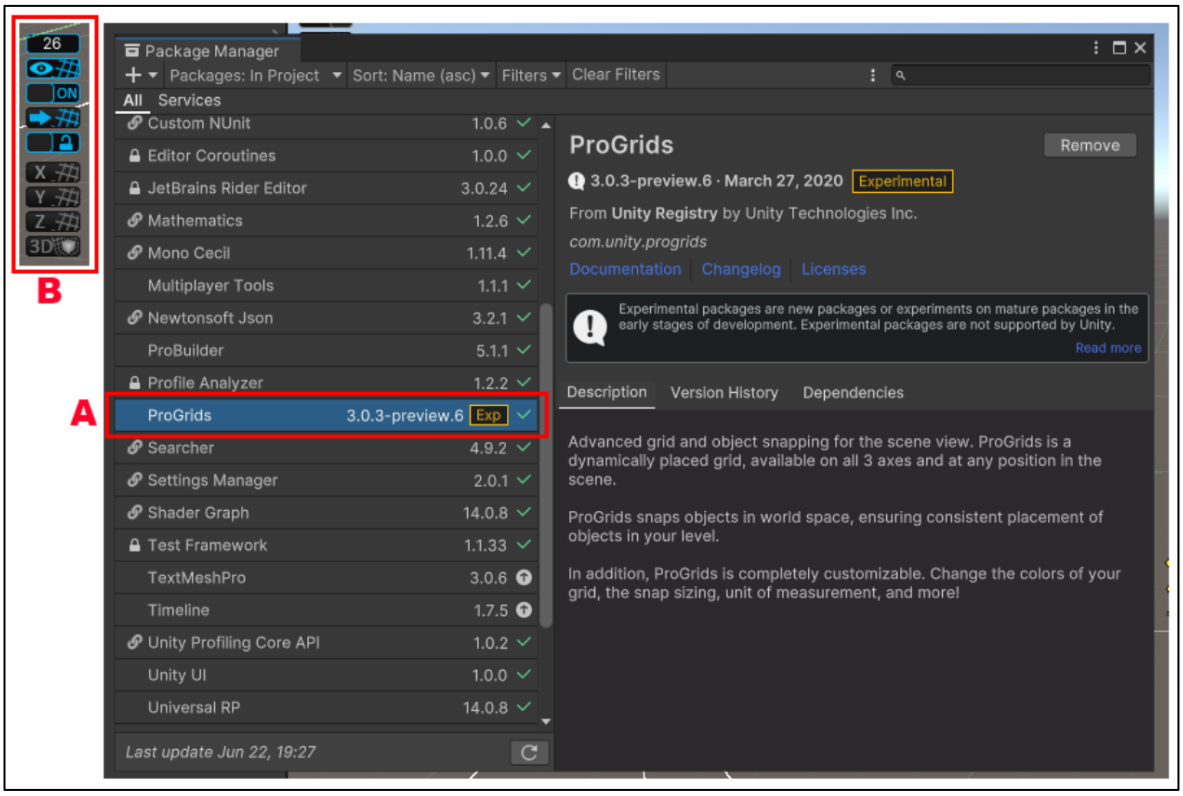


Figura 194 – ProGrids

Dentro de cada nível existe um objeto que faz com que seja possível passar de nível. Este objeto é invisível e apenas tem um *collider* para poder detetar colisões com o personagem principal. Na Figura 195 é possível observar esse objeto presente em cena com o nome "nextLevel". Este objeto contém um script que recebe um objeto que bloqueia a passagem do jogador impossibilitando-o de passar de nível.

Figura 195 - Objetos para a passagem de nível

Dentro do script "nextLevel" é possível observar o seu código presente na Figura 196. Cada cena dentro do jogo corresponde a um nível, pelo que o nome de cada cena começa por "Level_" seguido do número do nível, assim sendo é possível verificar o nível em que o jogador se encontra ou guardá-lo. O script "nextLevel" começa por guardar o nível em que se encontra numa variável como é possível observar na zona A da Figura 196. Em seguida na função "Update" é guardada e é atualizado o número de

inimigos presente dentro do nível, caso seja 0 o objeto que impede o jogador de progredir é desativado para assim então permitir o jogador de progredir (zona B da Figura 196). Por fim quando é detetada a colisão com o personagem principal, este verifica se o nível atual é maior ou menor que 9 para assim, quando existir a colisão, o jogador avançar um nível em frente. Por fim é verificado se o nível atual é igual ao nível mais alto atingido pelo jogador e caso seja é adicionado 1 ao nível mais alto para assim desbloquear um nível novo, caso não seja igual não é feito nada para não haver desbloqueio de níveis enquanto o jogador se encontra em níveis anteriores (zona C da Figura 196).

Figura 196 - Script "nextLevel"

Dentro do mapa de jogo existem diversos níveis (A da Figura 197) que ficam disponíveis consoante o jogador vai progredindo dentro do jogo. O controlo destes níveis é feito através do script "changeScene" e cada botão de nível contém esse mesmo script. Dentro desse script é utilizado a função "Update" para verificar se cada nível é menor que o nível mais alto desbloqueado pelo jogador e caso se verifique o mesmo fica disponível para o jogador usar, senão o mesmo fica desativo até o jogador desbloquear o nível correspondente (B da Figura 197). Ao clicar num botão de nível é carregada a cena/nível correspondente através da invocação da função "ChangeScene" presente na zona B da Figura 197.

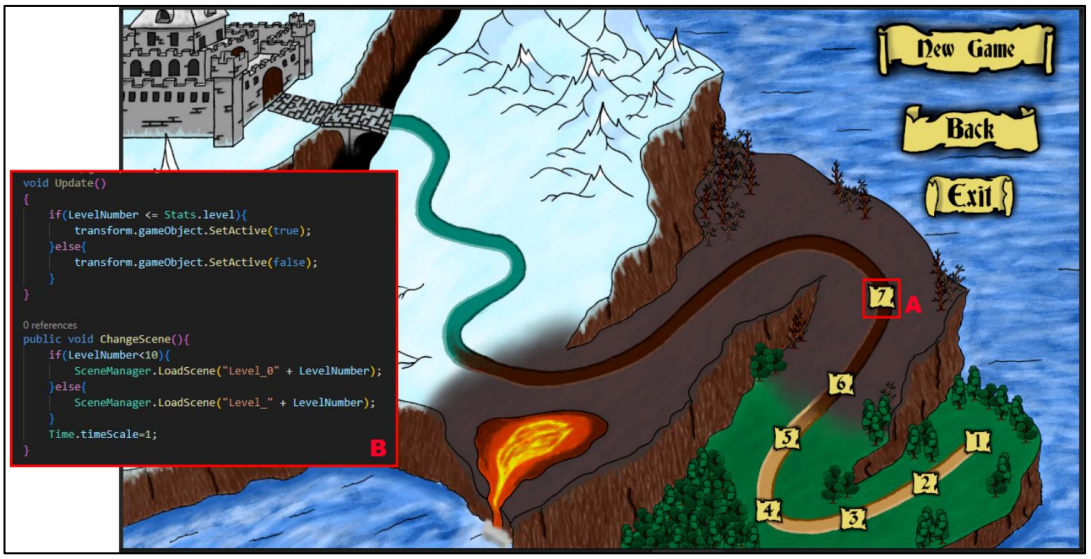


Figura 197 - Controlo de níveis no mapa

4.12 Save Game

O jogo desenvolvido é constituído por vários níveis e que levam tempo para serem concluídos e só depois é possível progredir. Para não perder o progresso feito dentro do jogo, foi desenvolvido um método de modo a ser possível guardar o jogo sempre que o jogador sai do mesmo. Foi então desenvolvido um script com o nome "dataControll" que permite guardar o progresso do jogador, assim como continuar o jogo através da informação guardada.

Para ser possível guardar o progresso do jogo foi adicionado ao botão de *exit* presente no menu de pausa (zona B da Figura 198) o script "dataControll" (zona C da Figura 198). Este botão ao ser pressionado é invocado a função "saveDataToFile" (zona D da Figura 198). Na zona A da Figura 198 é possível observar a função invocada pelo botão quando este é pressionado com o nome "saveDataToFile". O início desta função começa por criar uma variável com a localização desejada onde irá ser guardado o ficheiro de *save game* e em seguida cria um objeto, dentro do qual serão criadas várias variáveis onde são armazenadas todas as informações necessárias para guardar o progresso do jogo. Em seguida é criado um ficheiro txt onde é armazenada toda a informação desejada com as variáveis dentro do objeto criado anteriormente. Por fim é executado o comando para assim sair da aplicação.

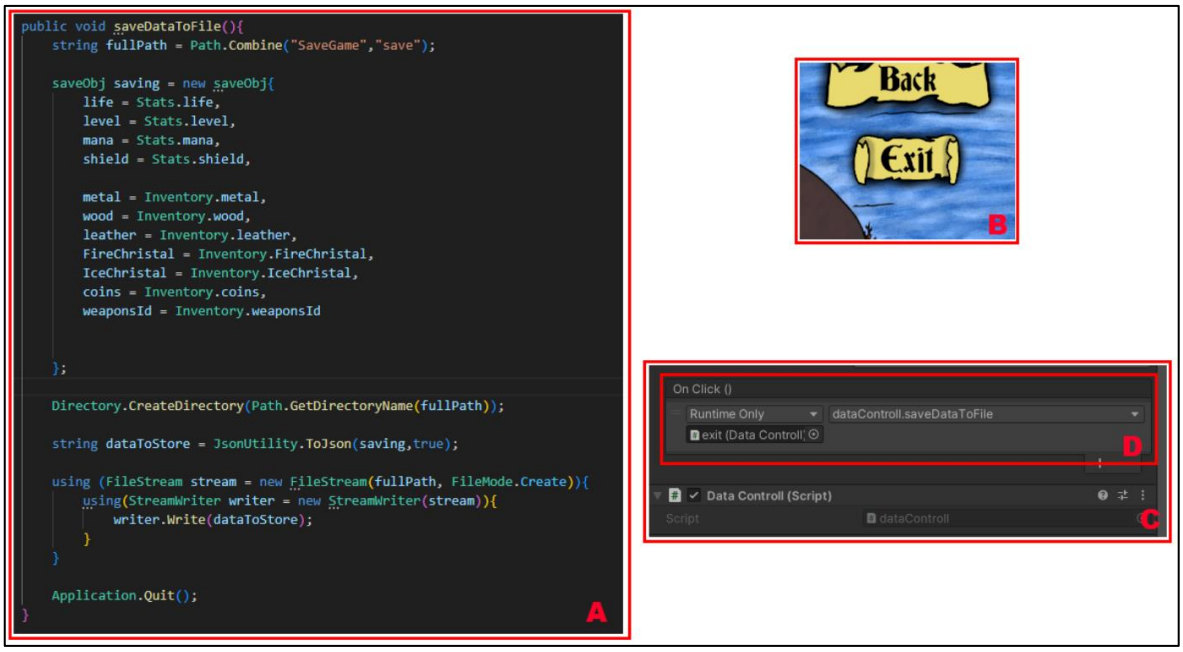


Figura 198 – Guardar progresso do jogo

Na Figura 199 é possível observar o que é guardado dentro do ficheiro criado como *save game.* É possível também observar as várias variáveis armazenadas dentro de aspas seguidas pelo valor guardado dentro da variável, todas elas separadas por virgulas.

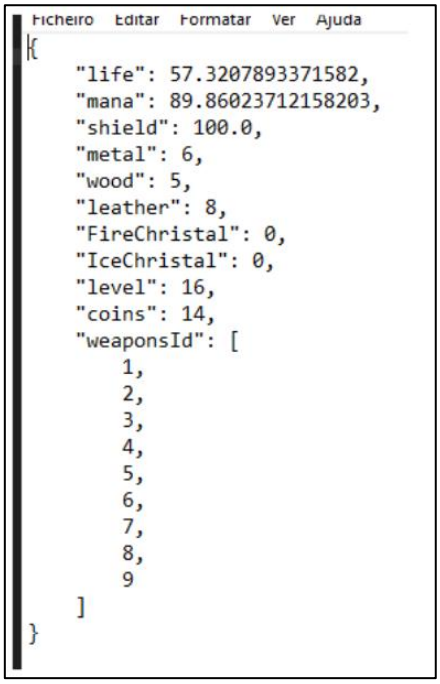


Figura 199 - Ficheiro de save game

Ao iniciar o jogo no menu inicial existe o botão de "Play" (zona B da Figura 200) que permite carregar as informações para o jogo salvo caso o ficheiro de *save game* exista. Dentro deste botão foi inserido também o script "dataControll" (zona C da Figura 200) que ao ser pressionado invoca a função "loadData" (zona D da Figura 200). Esta função ao ser invocada verifica se o ficheiro existe e caso exista é carregada a informação

presente nas variáveis para dentro das variáveis do jogo guardando assim a informação da última vez que o jogo foi jogado.

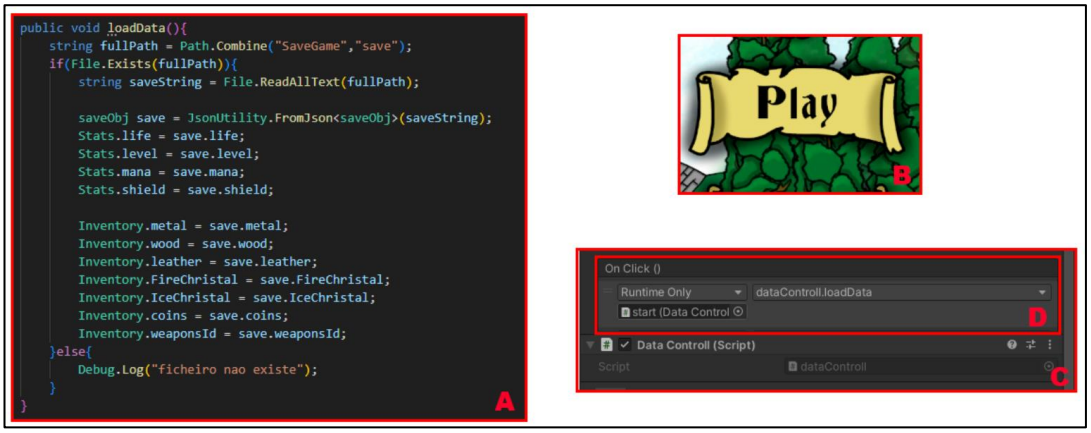


Figura 200 - Carregar progresso do jogo

Depois de clicar em "Play" o jogador é levado para o menu de jogo onde terá a opção de clicar em "Continue" (zona B da Figura 201) para poder continuar o jogo no sítio onde o deixou na última vez que jogou. Este botão contém o script "changeScene" (zona C da Figura 201) que ao ser pressionado invoca a função "autoChangeScene" (zona A da Figura 201). Esta função carrega a cena correspondente ao nível mais alto desbloqueado pelo jogador.

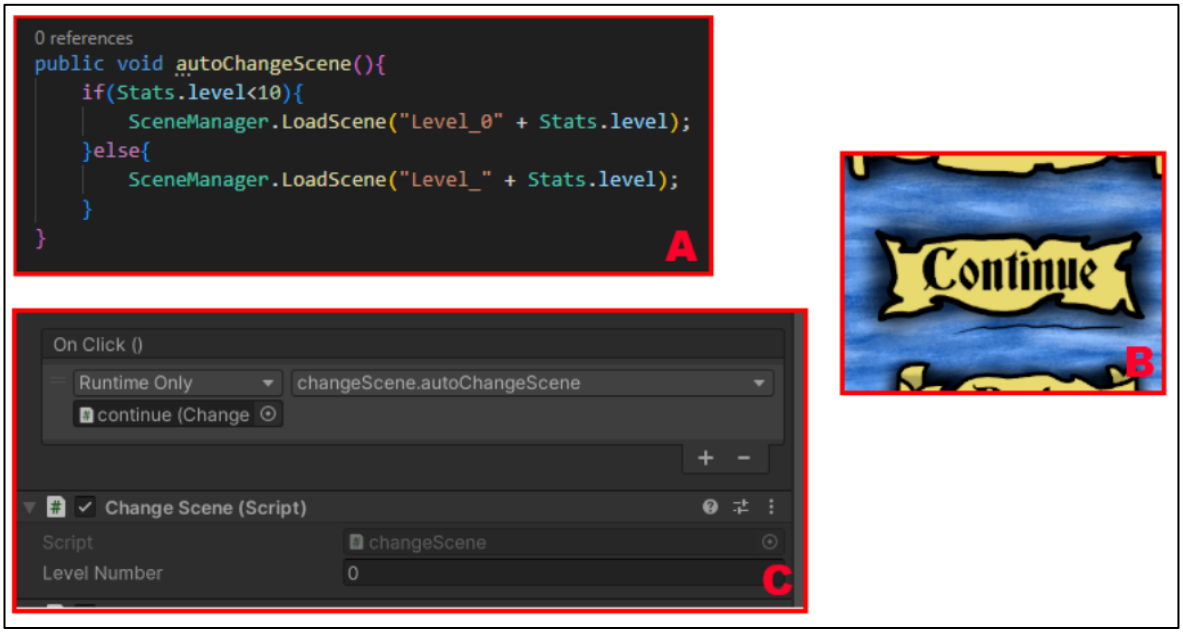


Figura 201 - Continuar jogo

5 Opiniões e melhorias

Após terminar o desenvolvimento do jogo, foi enviado o mesmo para um grupo de utilizadores para ser possível testarem e darem uma opinião geral. Foi também enviado com o jogo um formulário para os jogadores preencherem e darem as suas opiniões sobre o que foi desenvolvido, os erros que podem ter encontrado, e as sugestões de alterações que achassem que ia melhorar o mesmo. Este grupo de utilizadores foi constituído por dois tipos de utilizadores, os que jogam regularmente no seu dia a dia e outros que mal jogaram ao longo da sua vida. Durante uma semana foram recebidas várias opiniões sobre o jogo, sendo que várias foram respostas em comum e outras bastante distintas. Neste capítulo será apresentado um resumo destas opiniões recebidas pelos vários utilizadores, acompanhadas dos gráficos correspondentes.

5.1 Inimigos

Neste subcapítulo será apresentado as opiniões relacionadas aos inimigos do jogo. Será apresentado um gráfico com as várias notas dadas a diversos aspetos dos inimigos, seguindo as principais opiniões dadas por cada um.

5.1.1 Crab

Na Figura 202 é possível observar o gráfico gerado automaticamente com as notas dadas ao inimigo Crab. Através deste gráfico é possível observar que os resultados deste inimigo são bastante positivos, especialmente em relação ao seu design, variando maioritariamente as opiniões na sua movimentação e dificuldade. Em geral este inimigo obteve bons resultados obtendo uma nota média de 4.04 de 0 a 5.

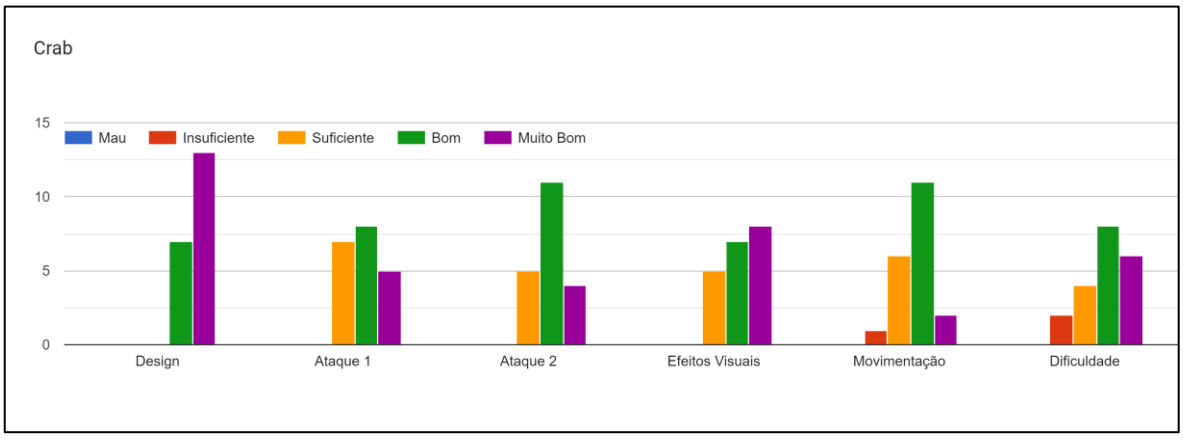


Figura 202 - Notas dadas ao inimigo Crab

As opiniões extra para este inimigo foram poucas, destacando-se apenas duas que dizem que o efeito do segundo ataque devia ser diferente pois confunde-se com sangue, o que pode fazer o jogador pensar que o personagem tivesse sido atingido, mas também é destacando também que o inimigo devia ser mais forte devido às suas dimensões.

5.1.2 Bombist

Obtendo também resultados muito positivos como é mostrado no gráfico da Figura 203, o inimigo Bombist ficou com uma nota média de 4,25. Este resultado demonstra que os utilizadores gostaram deste inimigo no geral.

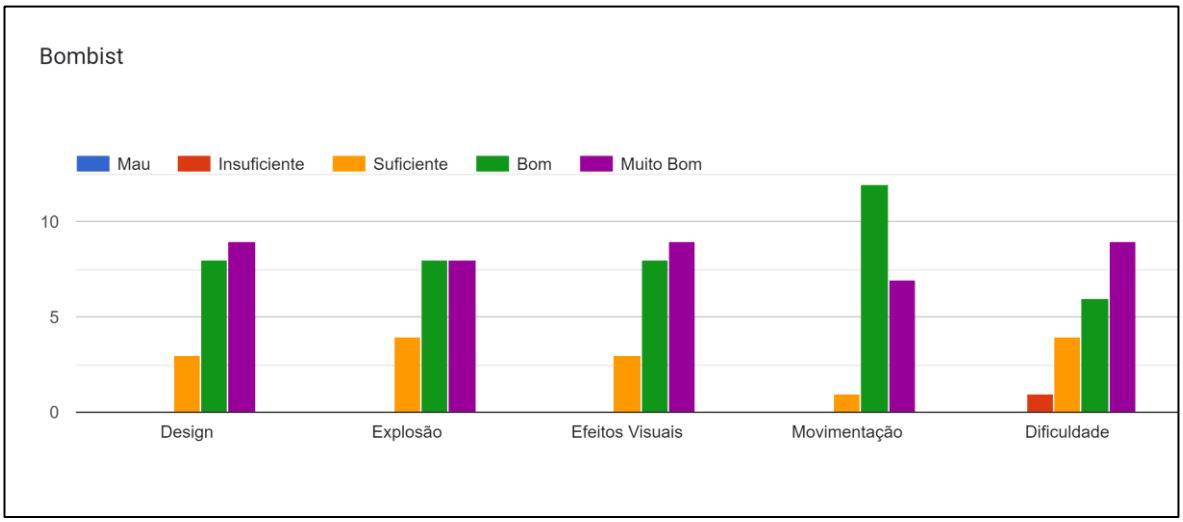


Figura 203 - Notas dadas ao inimigo Bombist

Este inimigo obteve também poucas opiniões, destacando-se o facto de no geral achar, que é um inimigo bastante fácil de derrotar, bastando apenas fugir dele depois de ser ativada a sua bomba. Foi também apresentado uma melhoria para o inimigo onde sugerem que o inimigo depois de morto poderia deixar a sua bomba no local e assim ter a possibilidade de utilizar a bomba em seu favor.

5.1.3 Dark Knight

O inimigo Dark knight obteve uma média bastante positiva, obtendo uma nota de 4,06. As notas obtidas por este inimigo são possíveis de observar através do gráfico na Figura 204. Este inimigo não obteve grandes opiniões destacando-se apenas o comentário de ser demasiado fácil de derrotar.

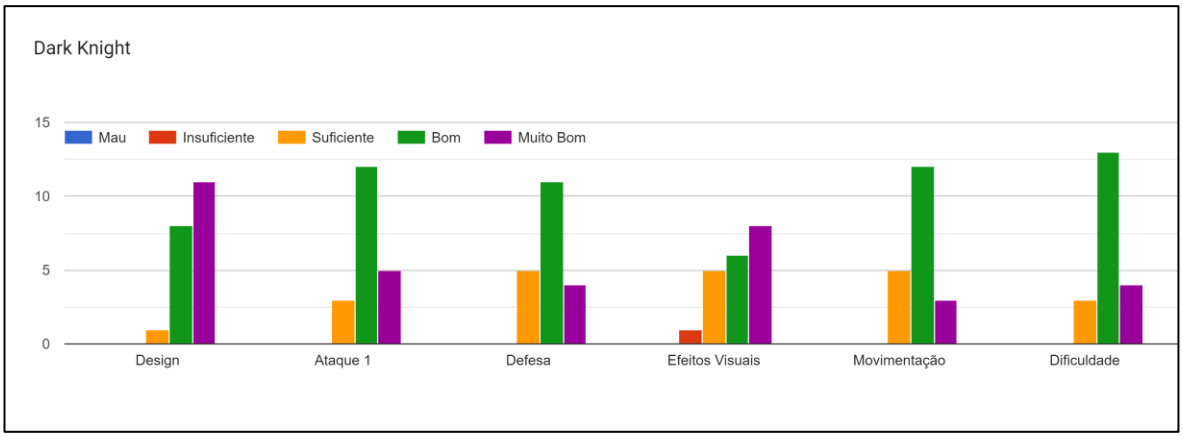


Figura 204 - Notas dadas ao inimigo Dark Knight

5.1.4 Horse Knight

Através do gráfico da Figura 205 é possível observar que o inimigo Horse Knight obteve igualmente resultados bastante positivos obtendo uma nota média de 4,2 , o que mostra que este inimigo foi bem recebido pelos utilizadores, destacando o seu design que obteve um excelente resultado.

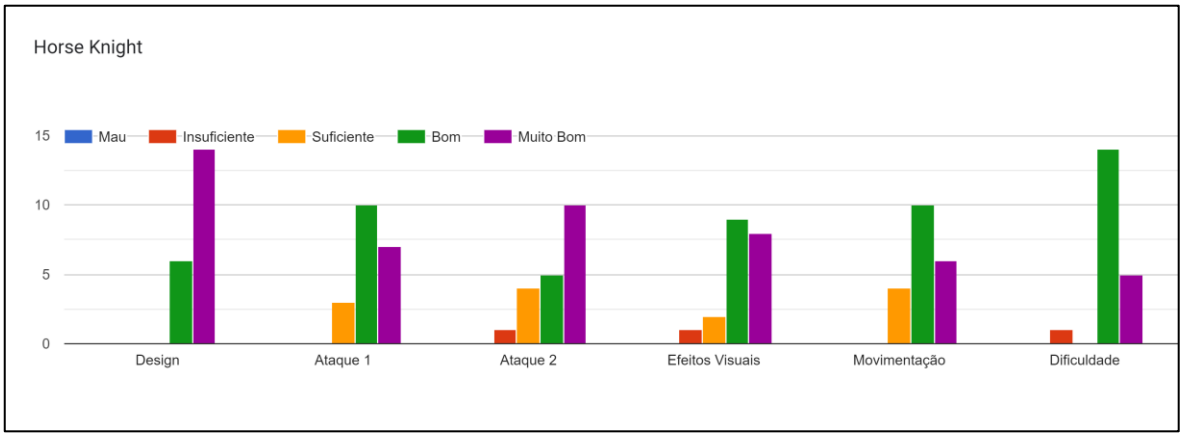


Figura 205 - Notas dadas ao inimigo Horse Knight

As opiniões para este inimigo foram menores, mas no geral positivas destacando apenas o facto da sugestão de que o segundo ataque deveria ser mais longo pois é bastante fácil desviar-se, e que o mesmo ataque deveria fazer o cavalo correr para a frente.

5.1.5 Javeling

O inimigo Javeling obteve uma nota média de 4,05, o que o torna também um inimigo bastante bem recebido.

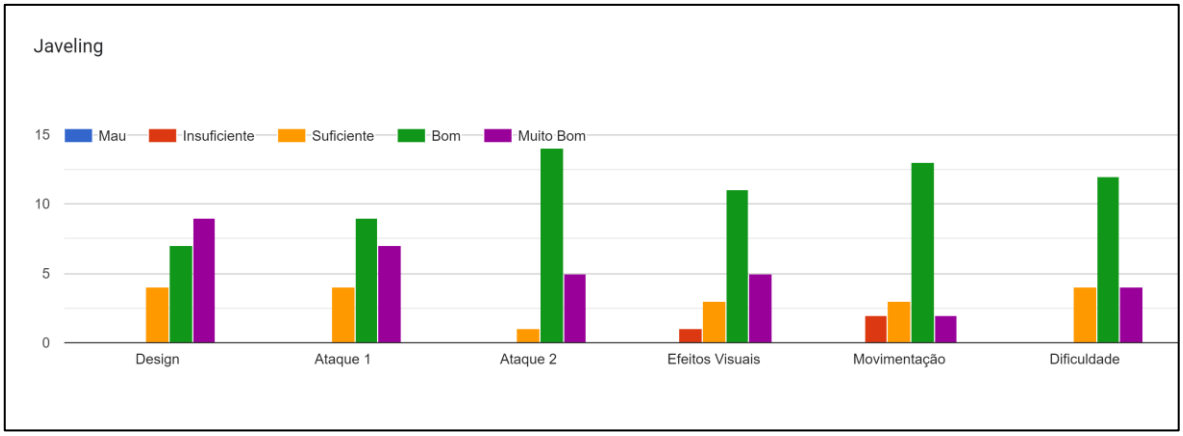


Figura 206 - Notas dadas ao inimigo Javeling

As opiniões deste inimigo mostraram que os utilizadores acharam que o inimigo deveria ser mais forte tendo em conta o seu tamanho e a arma que tem. É indicado também que o primeiro ataque não é bem claro, tornando difícil prever quando o vai executar.

5.1.6 Mage

O inimigo Mage obteve um bom resultado no que diz respeito ao seu design e efeitos visuais, porém a sua dificuldade obteve um resultado pior com base no gráfico presente na Figura 207. O Mage obteve uma nota média de 4,23.

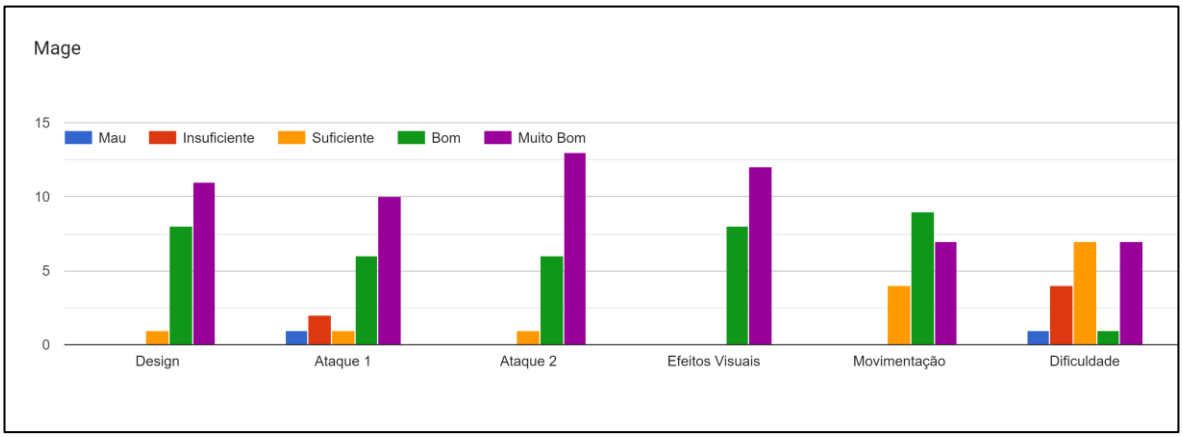


Figura 207 - Notas dadas ao inimigo Mage

As opiniões deste inimigo foram inúmeras em comparação aos restantes inimigos. No geral referem quase todas que o inimigo é bastante difícil de derrotar devido à frequência que executa o seu primeiro ataque, especialmente quando estão vários inimigos destes juntos, o que torna impossível conseguir defender e atacar outros inimigos ao mesmo tempo. Resumindo, é dito que o inimigo Mage é muito difícil e que a sua velocidade de ataque é muito alta.

5.1.7 Bealzor

O inimigo Bealzor obteve notas bastante divididas como é possível observar no gráfico da Figura 208. Porém a sua dificuldade teve um resultado bastante negativo quando comparado com os restantes elementos. A sua movimentação teve também as notas bastante divididas. O inimigo Bealzor obteve uma nota média de 3,85 tornando-o o inimigo com a nota mais baixa de todos.

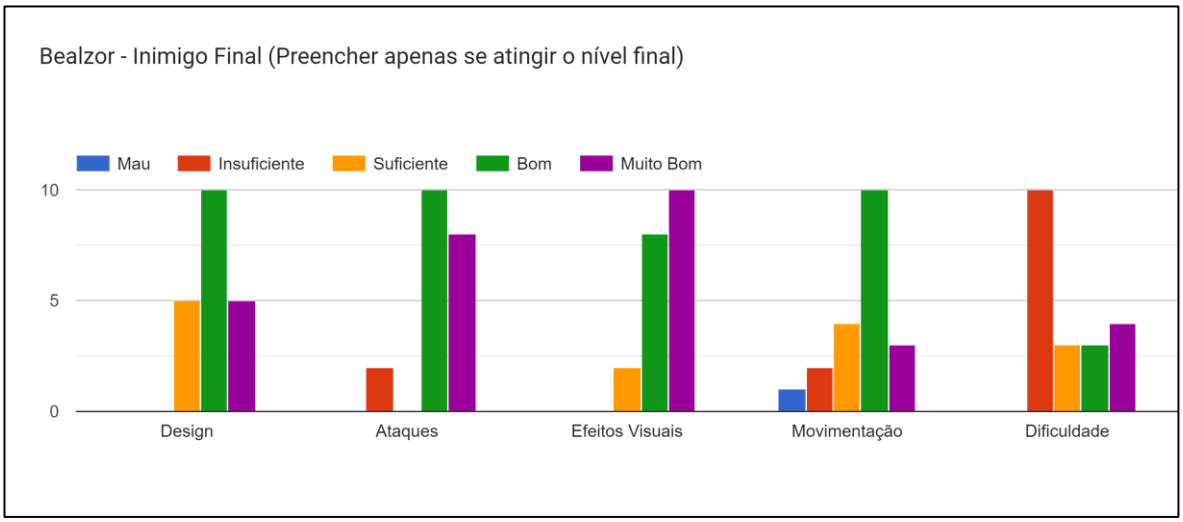


Figura 208 - Notas dadas ao inimigo Bealzor

As opiniões para este inimigo foram bastante distintas umas das outras, dando ênfase ao design que, no geral foi dito que está simples demais tendo em conta que é o

inimigo final, deveria ser mais chamativo. Foi dito que na segunda fase de vida deste inimigo deviam estar presentes também os 3 ataques iniciais, tendo assim um total de seis ataques para ser mais complexo. Foi também dito que a sua dificuldade é bastante baixa em comparação aos restantes inimigos, o que não deveria acontecer visto que é o inimigo final, deveria ser o pior.

Sobre este inimigo foram questionados também quais os ataques que os utilizadores gostaram mais e menos, onde se obteve a conclusão de que o ataque que foi melhor recebido foi o lazer de fogo lançado na segunda fase e o menos apreciado foi o ataque de gelo na primeira fase. Porém também o ataque de raios da primeira fase foi mencionado diversas vezes tendo um bom design.

5.2 Armas

Neste subcapítulo será mostrado um resumo das opiniões deixadas sobre as nove armas. Será apresentado um gráfico com a avaliação obtida em vários aspetos, seguido dos principais pontos das opiniões recebidas.

5.2.1 Espadas

A espada foi bem recebida pelos utilizadores no geral sendo que foi a preferência em relação às restantes, porém foi bastante criticado o seu dano por grande parte dos utilizadores, achando que era pouco, tendo que atingir demasiadas vezes os inimigos para assim os conseguir derrotar. Foi destacado o seu visual apreciando todo o seu design e os seus efeitos visuais. Na Figura 209 é possível observar às várias notas recebidas pelos utilizadores aos vários aspetos da espada.

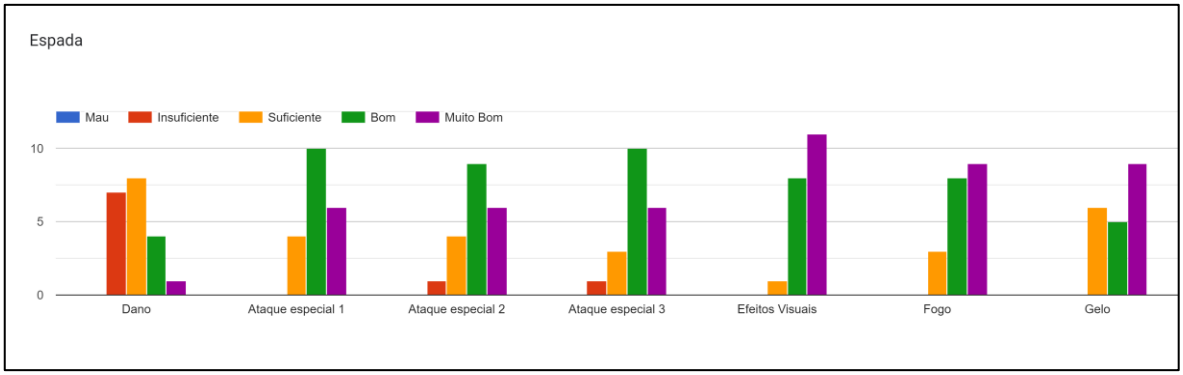


Figura 209 - Notas dadas à arma espada

5.2.2 Machados

O machado teve o seu terceiro ataque maioritariamente criticado, visto que empurrava demasiado os inimigos fazendo com que fosse bastante difícil os atingir com o terceiro ataque. Tal como a espada, foi apreciado no geral os seus efeitos visuais, porém teve alguns utilizadores que destacaram o dano do macho como sendo pouco. As notas dadas aos diversos aspetos do machado são possíveis de observar na Figura 210.

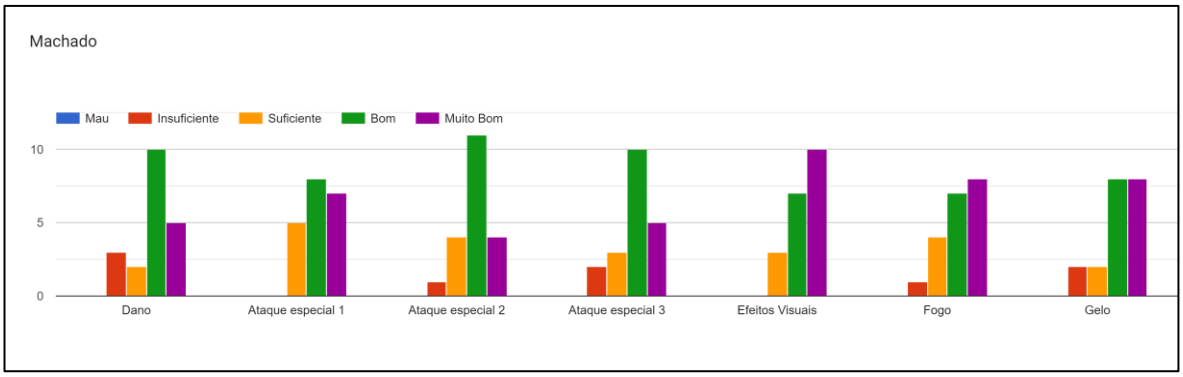


Figura 210 - Notas dadas à arma machado

5.2.3 Facas

As facas num geral tiveram os seus efeitos visuais e ataques bem recebidos, tendo aspetos positivos acerca dos mesmos. Porém foi bastante criticado o facto de darem muito pouco dano em relação as outras armas, o que as tornou pouco ou nada usadas dentro do jogo. Na Figura 211 é possível observar as notas obtidas nos diversos aspetos relacionados às facas.

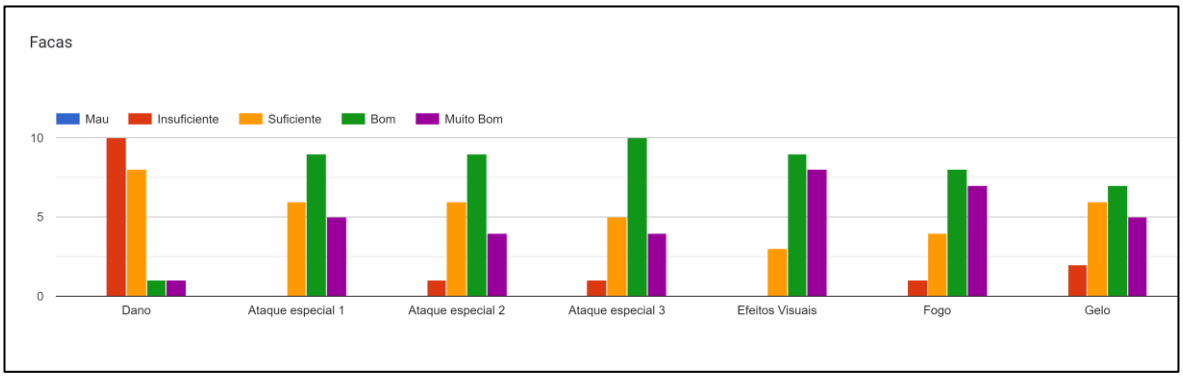


Figura 211 - Notas dadas à arma facas

5.3 Opiniões sobre jogabilidade

Foram feitas diversas perguntas sobre o jogo para perceber o que deveria ser mudado ou melhorado, sendo que se obteve diversas opiniões em relação ao mesmo. Neste subcapítulo serão apresentadas as diversas conclusões retiradas a partir das opiniões deixadas sobre o jogo.

Quando perguntado a dificuldade do jogo, foi referido que a dificuldade se encontra bastante alta ao longo do jogo devido à quantidade de inimigos presentes nos diversos níveis. Através do gráfico presente na Figura 212 é possível observar que 40% dos utilizadores acharam bastante difícil o jogo e 55% acharam difícil o jogo. As opiniões dadas seguem o mesmo padrão dizendo que nível é difícil apenas devido ao número de inimigos por nível, destacando o inimigo Mage devido ao seu Primeiro ataque. É falado também que a ausência de um mapa dentro do nível torna bastante difícil a navegação pelos mesmos.

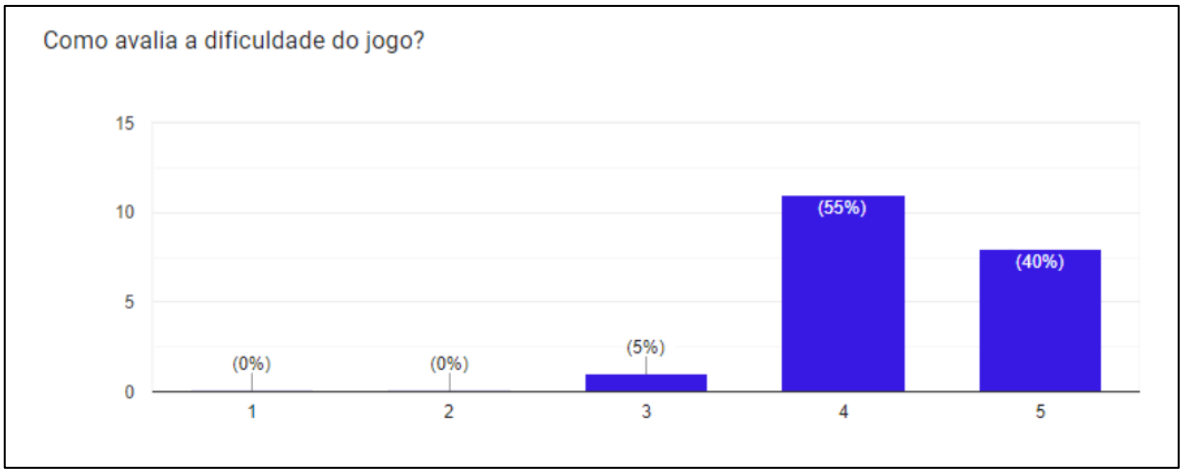


Figura 212 - Dificuldade do jogo

Os diversos temas dos níveis foram bastante apreciados tendo sido destacando o nível com tema de castelo respetivo ao nível final, que obteve bastantes opiniões positivas devido à diferença dos restantes níveis, tendo um ótimo design para uma batalha final contra o inimigo final. Os restantes biomas tiveram as suas opiniões bastante parecidas devido ao facto de serem semelhantes, porém foram também positivamente avaliados por ser um conceito interessante dividirem-se os vários biomas ao longo do jogo.

Quando questionados os utilizadores sobre a quantidade de níveis que o jogo deveria ter, foram obtidas diversas opiniões, porém a maioria dos utilizadores refere que o jogo tem um bom número de níveis como é possível observar no gráfico da Figura 213. Uma grande parte dos utilizadores refere também que devido à dificuldade do jogo, este poderia ter menos níveis, pois acabam por perder muito tempo em certos níveis, o que torna mais difícil progredir. Nenhum utilizador quis que o jogo tivesse mais níveis, mas foi referido que poderia ter mais níveis se fosse por exemplo para conseguir bónus apenas se não interferissem com a progressão do jogo.

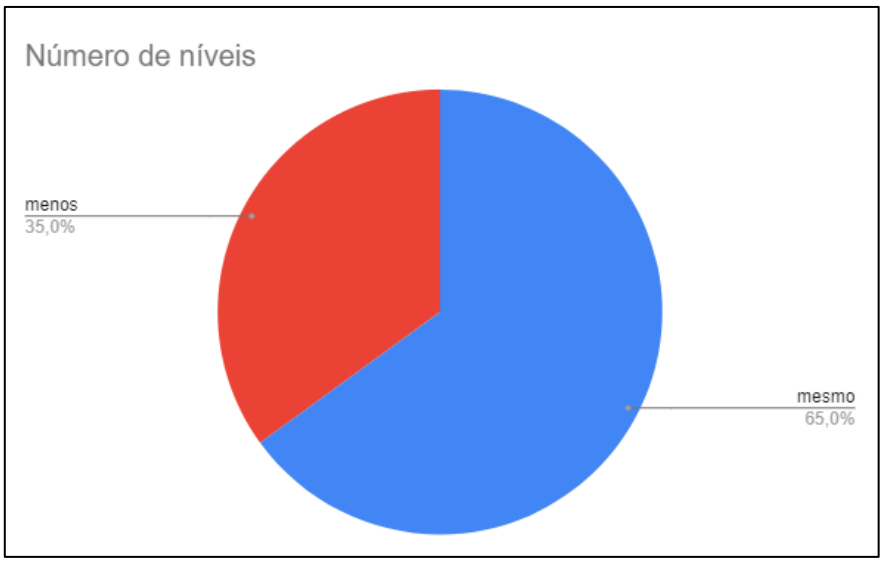


Figura 213 - Quantidade de níveis

As opiniões sobre a construção das armas foram bastante divididas entre os utilizadores como é possível observar através do gráfico da Figura 214. Apesar da maioria achar o método de construção foi fácil, porém foi destacado que era bastante difícil obter certos materiais para a sua construção.

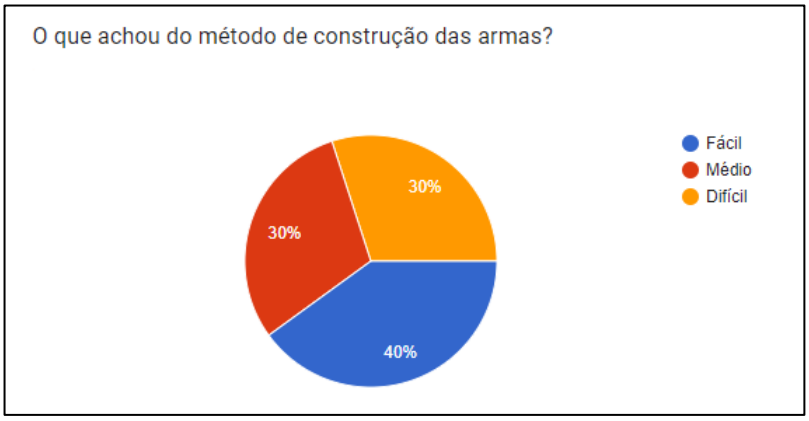


Figura 214 - Construção de armas

Quanto à loja de poções foi perguntado aos utilizadores se acharam fácil encontrar a loja de poções, pelo que a grande maioria achou fácil, como é possível observar no gráfico da Figura 215. Foi também perguntado a facilidade em obter as poções da loja e as opiniões mudaram, mostrando que a maioria achou difícil adquirir poções devido aos preços elevados comparado com o número de moedas que são recebidas por eliminar inimigos.

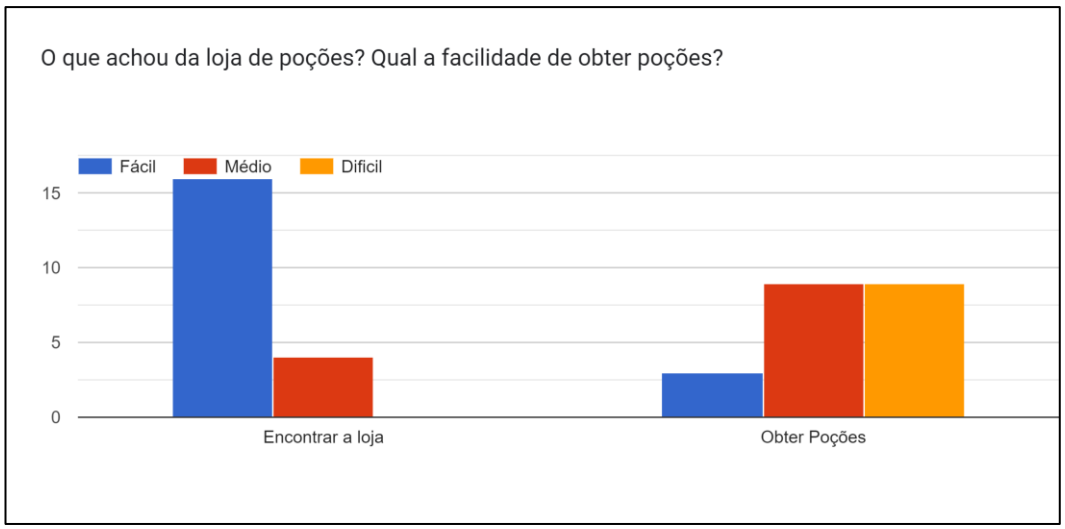


Figura 215 - Dificuldade em encontrar/comprar poções

No que toca aos menus do jogo as opiniões foram bastante semelhantes, dizendo todos que os menus contêm um bom design e eram bastante fáceis de usar.

Foi apontado também que deveria ser possível aceder o menu principal através do menu de pausa.

O menu de controlos demostrou-se útil e fácil de entender assim como o menu do mapa de jogo. Perguntou-se aos utilizadores o que acharam do mapa de jogo devido a ser o menu que mais vezes foi acedido, onde as opiniões foram bastante positivas obtendo 100% classificação na pergunta se gostaram do mapa como é possível

observar no gráfico da Figura 216. A única alteração possível que foi referida, foi que o botão de novo jogo não deveria estar na posição atual, pois acontecia muitas vezes carregar no mesmo em vez de continuar, fazendo com que perdessem o progresso já feito.

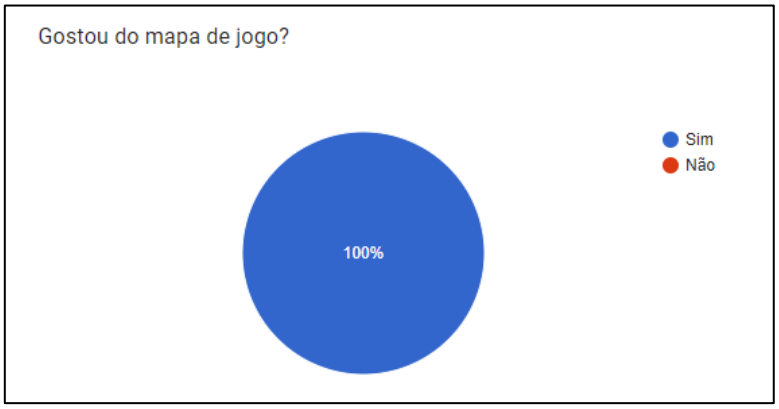


Figura 216 - Avaliação do mapa de jogo

Foi perguntado também aos utilizadores como foi o seu interesse do jogo ao longo do mesmo, concluindo que o interesse pelo jogo se manteve o mesmo do início ao fim como podemos verificar através do gráfico presente na Figura 217. Foi obtido também bastantes respostas que o interesse foi diminuindo devido a ser bastante difícil progredir no jogo tornando-se monótono.

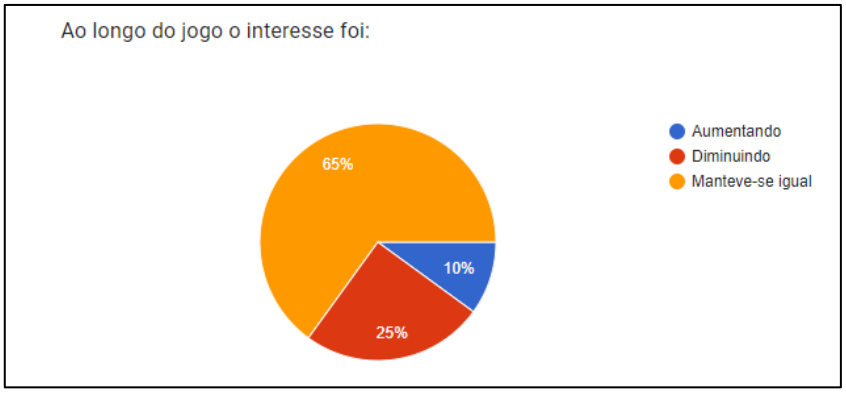


Figura 217 - Interesse no jogo

Por fim foi pedido aos utilizadores para avaliarem o jogo na totalidade pelo que a maior resposta dada foi 6 de 0 a 10 como é possível observar no gráfico da Figura 218. Apesar de também existirem bastantes respostas em que os utilizadores avaliaram o jogo com 7, o jogo obteve uma nota média de 6,4 de nota final.

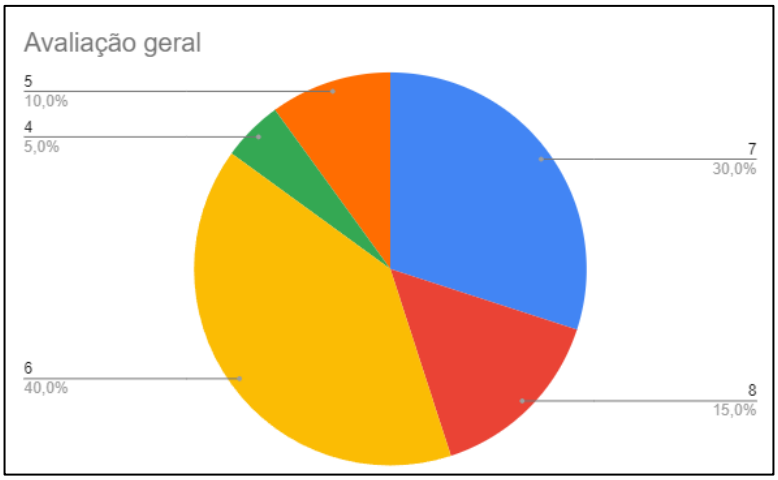


Figura 218 - Avaliação geral do jogo

5.4 Bugs encontrados

Os utilizadores ao longo do jogo foram encontrados diversos bugs dentro do mesmo tendo sido pedido que descrevessem o bug encontrado e o que fizeram para erro acontecer. Neste subcapítulo será descrito os principais bugs encontrados pelos utilizadores durante o seu tempo de jogo.

- **Bloqueio da personagem principal**

Este bug acontece durante o ataque do personagem principal que bloqueia na primeira animação do combo de ataque. Ao bloquear torna-se impossível executar alguma ação dentro do jogo, tendo apenas acesso aos menus sendo que a única solução para desbloquear o personagem é reiniciar o nível, porém mantem-se o risco de voltar a acontecer. Este bug destacou-se o pior de todos pois foi detetado por todos os utilizadores e acontecendo diversas vezes, perdendo o progresso que tinham dentro do nível em que estavam.

- **Velocidade infinita**

Este bug acontece quando se ativa o movimento de rolar e se muda de nível enquanto este movimento acontece. Por alterar a velocidade de movimento do jogador este movimento mantém-se infinitamente até que seja outra vez ativado o movimento de rolar.

- **Armas duplicadas**

Este bug acontece sempre que se constrói as facas, por serem dois objetos com o mesmo id, dentro do inventário aparecem dois botões pertencentes às facas. Este é um bug apenas visual, não trazendo qualquer tipo de problema ao jogo.

- **Inimigos estáticos**

Ao longo do jogo foram colocados diversos inimigos espalhados pelos diferentes níveis, porém em alguns níveis foram encontrados inimigos estáticos sem qualquer tipo de comportamento ou física. Este bug acontece devido ao ter sido utilizado um objeto do inimigo sem qualquer tipo de script ao invés do *prefab* criado do mesmo inimigo.

- **Flash de botões no mapa de jogo**

Este problema foi detetado por muitos poucos utilizadores, descrevendo terem visto todos os níveis do jogo desbloqueados quando abrem pela primeira vez o mapa de jogo. Os botões aparecem todos durante uma fração de segundo e acabam por desaparecer, apenas permanecendo os níveis que já tenham sido desbloqueados.

- **Bloqueio jogo ao sair de uma loja**

Este bug aconteceu apenas nos primeiros utilizadores pois foi um bug que rapidamente foi detetando e resolvido. Este bug fazia com que o jogo bloqueasse sempre que se abria e fechava um menu de uma loja.

- **Não atualização de animações**

Durante o jogo vários utilizadores detetaram que as animações não mudavam quando estes mudavam de arma sem parar de clicar nas teclas de caminhar, fazendo com que as animações da arma anterior permanecessem.

- **Paredes invisíveis**

No último nível foram encontradas algumas paredes pela qual a camara do personagem consegue atravessar fazendo com que se fique a ver a zona de fora do mapa e deixando de ser possível observar o jogo. Este bug não causa qualquer tipo de problema, pois basta o jogador andar para o lado ou na direção contaria que a câmara volta à sua posição normal.

- **Bloqueio do inimigo final**

Por último foi detetado também por diversos utilizadores que ao usarem o machado, o inimigo final bloqueava na primeira fase. No decorrer da batalha, quando a vida do inimigo final chega a zero e, caso o jogador esteja a usar o machado, o inimigo bloqueia ficando sem qualquer tipo de animação ou ataque, fazendo com que seja necessário reiniciar o nível, fazendo com que a batalha tenha que ocorrer novamente.

5.5 Possíveis melhorias

Para terminar o formulário, foi questionado aos utilizadores quais as possíveis melhorias a acrescentar numa futura atualização do jogo, onde foram também obtidas diversas opiniões. Neste subcapítulo serão descritas quais foram as principais opiniões dadas para o jogo.

Uma das principais ideias foi a adição de som ao jogo, criando uma nova forma de dar feedback ao utilizador quando, por exemplo, o personagem principal recebe ou emite dano. Outra opinião foi a adição de barras de vida dos inimigos, informação bastante útil durante todo o jogo.

Diversos utilizadores sugeriram a diminuição da dificuldade do jogo, baixando o número de inimigos na maioria dos níveis. Ainda na dificuldade foi mencionado também a sugestão de diminuição da dificuldade do inimigo Mage, por exemplo, aumentando o tempo entre ataques ou dando um tempo limite de duração das bolas de energia.

Outra das principais características comentadas para adicionar seria um mini mapa por cada nível, devido a ser difícil encontrar a zona onde termina o nível e se passa ao seguinte sem se perderem diversas vezes, especialmente nos níveis com vários

caminhos possíveis. Outra alternativa ao mapa que poderia ajudar na orientação dentro do nível seria uma bússola que apontasse para a saída do mesmo.

Por fim uma das coisas que foi bastante destacado pelos utilizadores foi a falta de uma história que pudesse dar um sentido ao jogo, sugerindo adicionar ao início do jogo uma série de imagens que contassem a história do mesmo, tornando tudo mais interessante devido a percebermos o porquê de tudo o que se passa no jogo. Outra sugestão dentro da ideia do mapa, foi no lugar de uma simples história no início do jogo, tornar possível adquirir livros antigos, ou existirem pedras com escrituras antigas, aparecendo quando progredíamos pelos níveis e, deste modo ir desbloqueando memórias sobre a história do jogo e personagens.

6 Conclusão

Com o desenvolvimento deste projeto foi possível adquirir diversos conhecimentos novos e experiência na área de desenvolvimento de jogos. Ao longo deste processo, foram exploradas diversas ferramentas e técnicas, desde a utilização do Unity como plataforma principal de desenvolvimento, até à criação de recursos gráficos com ferramentas como o Blender e Kryta.

As diversas funcionalidades que foram planeadas no GDD foram elementos cruciais para alcançar os objetivos do projeto. Aplicar todos os conhecimentos adquiridos foi um grande passo para poder desenvolver todo o raciocínio por detrás do jogo desenvolvido. A interface do jogador, composta pelos diversos menus de jogo e informações presentes no jogo principal, foram também fundamentais para proporcionar uma boa experiência ao utilizador.

É importante destacar também os diversos utilizadores que deram a sua opinião acerca dos vários aspetos do jogo, que serviram para poder perceber o que poderia ser alterado dentro do mesmo. Foi fundamental para perceber o que estava errado e o que estava bom para poder encontrar todos os problemas e erros do código desenvolvido que resultavam em bugs dentro do jogo.

6.1 Trabalho Futuro

Para o futuro pretende-se executar diversas funcionalidades presentes no GDD que não foram completamente implementadas, assim como diversos elementos descritos que não tiveram lugar no presente jogo final.

Dentro dessas funcionalidades destaca-se um sistema para poder melhorar as armas construídas dentro do jogo, bem como a armadura do jogador. A implementação de um sistema de dificuldade melhorado, que aumenta a dificuldade dos inimigos ao invés de aumentar o número dos mesmos por níveis. Destaca-se também a implementação de uma história dentro do jogo que trará outro significado ao jogo dando mais enfase ao objetivo final. Por fim a criação de mais armas dentro do jogo dando mais oportunidades e variedade ao jogador, bem como a implementação de um mini mapa para todos os níveis presentes dentro do jogo.

Devido a limitações de tempo e complicações encontradas durante todo o desenvolvimento, todas estas funcionalidades e melhorias mencionadas foram adiadas para futuras iterações do projeto. Estas serão exploradas e desenvolvidas com o objetivo de proporcionar uma experiência de jogo ainda mais enriquecedora.

7 Referências

[1] J. Fordyce, “DaFont,” [Online]. Available: https://www.dafont.com/pt/deutsch-gothic.font.