Tutorial para desenvolvimento de jogos eletrônicos 3D na Unity

Glauber Ferreira Angelo

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, PB
Dezembro – 2022

Glauber Ferreira Angelo

Tutorial para desenvolvimento de jogos eletrônicos 3D na Unity

Monografia apresentada ao curso de Ciência da Computação do Centro de Informática, da Universidade Federal da Paraíba, como requisito para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Gustavo Henrique Matos Bezerra Motta

João Pessoa, PB
Dezembro – 2022

Catalogação na publicação
Seção de Catalogação e Classificação

A584t Angelo, Glauber Ferreira.
Tutorial para desenvolvimento de jogos eletrônicos 3D na Unity / Glauber Ferreira Angelo. - João Pessoa, 2022.
72 f. : il.

Orientação: Gustavo Henrique Matos Bezerra Motta.
Monografia (Graduação) - UFPB/CI.

1. GDD. 2. Unity. 3. Jogos. 4. UML. I. Motta, Gustavo Henrique Matos Bezerra. II. Título.

UFPB/CI CDU 004.4

Elaborado por Michelle de Kássia Fonseca Barbosa - CRB-738

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
COORDENAÇÃO DO CURSO

Centro de Informática
UFPB

Ata da Sessão Pública de Defesa de Trabalho de Conclusão de Curso de **Glauber Ferreira Angelo**, realizada em **07** de **dezembro** de **2022**.

Aos **07** dias do mês de **dezembro**, do ano de **2022**, às **10:30** horas, por meio de videoconferência, reuniram-se os membros da Banca Examinadora constituída para julgar o Trabalho de Conclusão de Curso do Sr. **Glauber Ferreira Angelo**, matrícula nº **20160144357**, aluno do Curso de Bacharelado em Ciência da Computação da Universidade Federal da Paraíba. A comissão examinadora foi composta pelo professor Gustavo Henrique Matos Bezerra Motta (UFPB), orientador e presidente da banca, e pelos professores Danielle Rousy Dias Ricarte (UFPB), examinador interno e Vitor Meneghetti Ugulino de Araújo (UFPB), examinador interno. Iniciando os trabalhos, o presidente da banca cumprimentou os presentes, comunicou-os da finalidade da reunião e passou a palavra ao candidato para que fizesse a exposição oral da monografia intitulada **"Tutorial para desenvolvimento de jogos eletrônicos 3D na UNITY"**. Concluída a exposição, o candidato foi arguido pela Banca Examinadora que, em seguida, emitiu o seguinte parecer: "***aprovado***", com conceito __9,0__ (0,0 a 10,0). Do ocorrido, eu, Leandro Carlos de Souza, Coordenador do Curso de Bacharelado em Ciência da Computação, lavrei a presente ata que vai assinada por mim e pelos membros da banca examinadora. João Pessoa, 07 de dezembro de 2022.

Prof. Leandro Carlos de Souza

gov.br Documento assinado digitalmente
GUSTAVO HENRIQUE MATOS BEZERRA MOT
Data: 07/12/2022 15:16:29-0300
Verifique em https://verificador.iti.br

Professor Gustavo Henrique Matos Bezerra Motta

Orientador (UFPB)

gov.br Documento assinado digitalmente
DANIELLE ROUSY DIAS RICARTE
Data: 09/12/2022 16:14:30-0300
Verifique em https://verificador.iti.br

Professora Danielle Rousy Dias Ricarte

Examinador interno (UFPB)

gov.br Documento assinado digitalmente
VITOR MENEGHETTI UGULINO DE ARAUJO
Data: 09/12/2022 09:10:55-0300
Verifique em https://verificador.iti.br

Professor Vitor Meneghetti Ugulino de Araújo

Examinador interno (UFPB)

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DA PARAÍBA
SISTEMA INTEGRADO DE PATRIMÔNIO, ADMINISTRAÇÃO E CONTRATOS

FOLHA DE ASSINATURAS

Emitido em 13/12/2022

SOLICITAÇÃO Nº 1/2022 - CI - CCC (18.56.01)
(Nº do Documento: 1)

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 14/12/2022 20:00)
LEANDRO CARLOS DE SOUZA
COORDENADOR DE CURSO
1140339

Para verificar a autenticidade deste documento entre em https://sipac.ufpb.br/documentos/ informando seu número: **1**, ano: **2022**, documento (espécie): **SOLICITAÇÃO**, data de emissão: **13/12/2022** e o código de verificação: **d30c0a173a**

“A vida é como um jogo de vídeo game, em um jogo, você tem objetivos, e são exatamente os objetivos que te fazem TER VONTADE de jogar e não desistir até completar-los.”

Fillipe Schneider

DEDICATÓRIA

A Deus por ter me proporcionado a oportunidade da realização deste curso concedendo-me esta vitória, proporcionando-me sabedoria em todos os momentos dessa trajetória.

Aos meus pais Maria da Penha Ferreira Angelo e Manoel Angelo Filho, pois são meus melhores amigos que me ensinaram a viver, sendo responsáveis por essa vitória tão almejada.

AGRADECIMENTOS

A **DEUS**, dono de todo saber, e me proporcionou perseverança, força, sabedoria, e discernimento, colocando pessoas maravilhosas em minha vida no decorrer dessa longa caminhada.

Aos meus pais **Maria da Penha Ferreira Angelo** e **Manoel Angelo Filho**, pois me fizeram acreditar que eu sou capaz, ajudaram-me com todas as despesas no decorrer do curso e me deram suporte e apoio em todas as fases desse percurso, investindo no meu sonho, tornando-me um ser humano mais completo.

A minha esposa **Taciana das Neves da Conceição Angelo** pela paciência, dedicação, estando totalmente presente em todos os momentos da minha jornada, acreditando na minha capacidade, sendo compreensiva, carinhosa dizendo-me palavras de conforto e autoconfiança.

Ao meu orientador **Gustavo Henrique Matos Bezerra Motta** por ter aceitado participar da minha pesquisa e pela sua importante contribuição através da sua amizade, dedicação e compartilhamento de informações que foram de suma importância para realização e conclusão deste trabalho.

RESUMO

A história dos jogos eletrônicos apresenta-se inicialmente na fabricação dos jogos de videogame por William Higinbotham's que desenvolveu na tela do seu osciloscópio o primeiro jogo chamado tênis para dois. Até pouco tempo atrás desenvolver jogos não era fácil, sendo preciso ter uma equipe de especialistas para realizar o desafio, e torcer para que o jogo passasse por todas as etapas de avaliações das empresas de jogos para que o jogo fosse lançado. Atualmente existem motores gráficos prontos, como o Unity, que economizam tempo de desenvolvimento, pois contam com alguns recursos prontos como atualização de tela e sistema de colisão. Ao passo que as grandes empresas de jogos tendem a manter a forma de trabalha com os jogos, as empresas independentes podem arriscar e ousar fazer o que nenhuma grande empresa de jogos fez antes. Sendo assim, os jogos independentes podem abusar da criatividade e inovação para ganhar espaço no mercado, preenchendo lacunas deixadas pelas grandes desenvolvedoras de jogos. O objetivo deste trabalho de conclusão de curso é criar um jogo desde o inicio da criação até o final do processo do desenvolvimento do jogo, introduzindo conceitos básicos de jogos, seu histórico, conhecimento sobre Unity e a documento de design de jogo e UML que tem como finalidade registra cada informação do jogo tornando dessa forma mais simples sua implementação. É esperado que com este conhecimento o leitor possa ter conhecimento geral do processo de desenvolvimento e tenha a habilidade necessária para desenvolver seus jogos. O resultado atingido utilizando a Unity foi significativo sendo possível desenvolver um jogo. Ao entrar no jogo é possível controlar um personagem que pode explorar o mapa do jogo, luta contra inimigos e realizar teletransporte entre cenários.

Palavras-chave: GDD, Unity, Jogos, UML .

ABSTRACT

The history of electronic games is initially presented in the manufacture of video games by William Higinbotham's who developed the first game called tennis for two on the screen of his oscilloscope. Until recently, developing games was not easy, requiring a team of specialists to carry out the challenge, and hoping that the game would go through all the estimation stages of the game companies for the game to be released. Currently, there are ready-made graphics engines, such as Unity, which save development time, as they have some ready-made features such as screen updating and the collision system. Whereas big game companies tend to stick to one way of working with games, independent companies can take risks and dare to do what no big game company has done before. Therefore, independent games can abuse creativity and innovation to gain space in the market, filling gaps left by large game developers. The objective of this work to complete the course is to create a game from the beginning of the creation until the end of the game development process, presenting basic concepts of games, its history, knowledge about Unity and a document of game design and UML that has with the purpose of registering each game's information, thus making its implementation simpler. It is expected that with this knowledge the reader can have a general knowledge of the development process and have the necessary skills to develop their games. The result achieved using Unity was significant and it was possible to develop a game. Upon entering the game, it is possible to control a character who can explore the game map, fight enemies and teleport between scenarios.

Key-words: GDD, Unity, Games, UML.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

GDD	–	Game Design Document
IA	–	Inteligência artificial
MIT	–	Massachusetts Institute of Technology
MMORPG	–	Massive Multiplayers Online Role Playing Games
NPC	–	Non Player Characters
RPG	–	Role Playing Game
TCC	–	Trabalho de Conclusão de Curso
UML	–	Unified Modeling Language

SUMÁRIO

1. INTRODUÇÃO

Nos últimos anos a expansão do mercado de jogos eletrônicos proporcionou um alcance ainda maior de público. Nos dias atuais é possível notar que ainda existem pessoas que creem que jogos eletrônicos são feitos para desocupados ou crianças. No passado quando o mercado de jogos não era reconhecido talvez essa afirmação fosse verdade, mas vendo como a indústria de jogos eletrônicos esta crescendo vemos uma realidade bem diferente.

Conforme Torres (2015, p. 7): “Os jogos digitais consistem em um método que ajuda o jogador a desenvolver habilidade e conhecimentos de maneira didática, ao mesmo tempo que lúdica”. Para Lima (2011) jogar é uma realização, que trabalha o intelecto, onde se dedica ao mesmo tempo a uma atividade lúdica e também trabalha o raciocínio lógico facilitando ações que contribuem, desde cedo, no desenvolvimento do raciocinio.

No entanto desenvolver um jogo é um desafio, porque um jogo é uma aplicação mais complexa, envolvendo diversas áreas de conhecimento e formações específicas. Conforme Berto (2007) a maior dificuldade na criação de jogos é saber quando realmente está finalizado. Diversos desenvolvedores de jogos abandona o projeto ou não consegue finaliza-lo. Já que tais desenvolvedores não conhecem a próxima parte da criação do jogo e do mesmo modo não enxergar como o jogo será finalizado, podendo ter deixado partes importantes do jogo sem ser implementadas. Estas adversidades podem ocorrer em várias empresas, tanto grandes produtoras ou desenvolvedores independentes, em razão disso o documento de design de jogo é muito importante.

O desenvolvimento de jogos envolve o uso de diversas ferramentas, sendo hoje em dia, o uso de ferramentas como a Unity. A tecnologia Unity, consiste em um recurso para o desenvolvimento de jogos, tendo uma versão gratuita com algumas restrições que não interferem na criação de um game completo. Apresentando uma interface compreensível, mas ao mesmo tempo possui vários recursos importantes que facilita o desenvolvimento do jogo.

Conforme Santos e Alves (2015) o processo de desenvolvimento de jogos embora seja diferente em comparação aos softwares corporativos, pelo fato de implicar em uma quantidade de especialidades dentro das concepções do design de jogo, a Unified Modeling Language (UML) pode proporcionar uma documentação, auxiliando no desenvolvimento de jogos, possibilitando uma comunicação mais abragente entre um time multirreferencial.

1.1. Objetivos

1.1.1. Objetivo geral

O objetivo deste trabalho de conclusão de curso é criar um jogo desde o inicio da criação até o final do processo do desenvolvimento do jogo.

1.1.2. Objetivos específicos

Os objetivos específicos, definidos para que o objetivo geral deste TCC seja alcançado, foram os seguintes:

- Pesquisa sobre o processo de desenvolvimento de jogos eletrônicos;
- Modelagem da implementação do sistema em UML;
- Pesquisa sobre motores de jogos e definição da tecnologia utilizada;
- Definição dos tipos de perspectiva de câmera, controle e personagem;
- Modelagem do cenário;

1.2. Estrutura da monografia

O TCC está organizado da seguinte forma: O capítulo 2 é composto pela fundamentação teórica. O capítulo 3 trata sobre documentação do design de jogo. O capítulo 4 é a apresentação do desenvolvimento do return 7. E para concluir, O capítulo 5.

1.3. Método para desenvolvimento do game

Foi utilizada a game engine Unity, a linguagem utilizada será o C# para geração de códigos e o ambiente de desenvolvimento escolhido foi o Visual Studio.

A documentação do sistema contém um documento de design de jogo e UML que tem como finalidade registra cada informação do jogo tornando dessa forma mais simples sua implementação.

FUNDAMENTAÇÃO TEÓRICA

O desenvolvimento de jogos necessita de uma pesquisa ampla em várias áreas. Este capítulo foi feito através de pesquisas realizadas sobre desenvolvimento de jogos que abordar os aspectos conceituais e históricos dos jogos, os gêneros de jogos que existe, o conceito do documento de design de jogo para o desenvolvimento do jogo, conceitos da UML para modelagem do jogo e, por fim, apresenta-se a parte conceitual necessária da Unity para a criação do jogo. Dessa forma, a fundamentação teórica abordar toda a teoria necessária pra compreensão e desenvolvimento do jogo.

1.1. Aspectos Conceituais e históricos dos Jogos

Marcelo e Pescuite (2009) definem como jogo qualquer tipo de competição onde regras são criadas num ambiente restrito cujas regras são universais. É determinado como jogo um sistema, estabelecido por regras, onde os jogadores podem adentrar em um conflito artificial que resulta em um desfecho quantificável. (SALEN; ZIMMERMAN, 2003 apud SANTAELLA; FEITOSA, 2009).

A história dos jogos eletrônicos, no ano de 1958, apresenta-se inicialmente na fabricação dos jogos de videogame por William Higinbotham's que desenvolveu na tela do seu osciloscópio o chamado tênis para dois (Figura 1). Em 1961, três anos depois, um grupo de estudantes do MIT produziu um jogo que simulava naves que lutavam ao redor de uma estrela (Figura 2), ao qual deram o nome de Spacewar, e que só chegou a ser terminado em 1962.

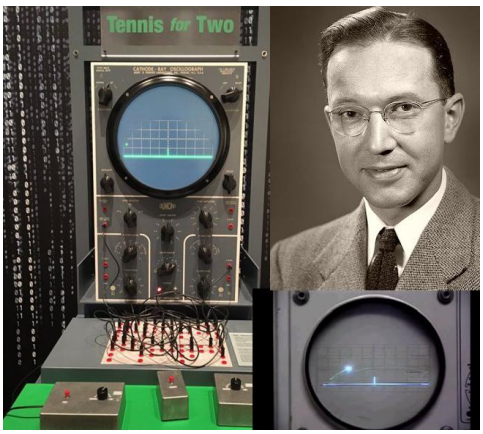

Figura 1: William Higinbotham e osciloscópio
Fonte: Adaptado de Mercury (2011), Johnson (2015) e Pisterest (2022).

Figura 2: Apresentação do jogo a esquerda e estudantes do MIT jogando a direita
Fonte: Adaptado de Gularte (2018) e Henrique (2021).

O primeiro console de jogo, surgiu em 1972, o Magnavox Odyssey baseado em um projeto de Ralph H. Baer, na década de 60. Surgiu na mesma década outro console que se tornou um dos mais conhecidos mundialmente: Atari, um videogame que utilizava a TV doméstica para projetar sua imagem. Com o surgimento de mais jogos para Atari, na década de 80, e o aumento da potência dos processadores, ocorreu o desenvolvimento de diversos jogos, que foi o principal marco para a revolução dos jogos, dando origem a alguns dos mais poderosos e realistas, até os atuais mais conhecidos jogos no mercado (MARCELO; PESCUITE, 2009).

No começo dos anos 90 começam a aparecer jogos com histórias, já que antes só eram vistos jogos – onde o que importava era qual jogador fazia mais pontos, e superava a marca dos demais. Surgem jogos clássicos como Sonic e Super Mario (Figura 3), consequentemente aparecem outros jogos como Super Metroid, Star Fox, Virtual Rancing, Star Wars, Doom e outros mais. De acordo com Ramos (2007), devido à competição tecnológica, na década de 90, começaram a aparecer os primeiros consoles de 32 e 64 bits como 3DO, Playstation e Nintendo 64, ressaltando a qualidade gráfica e desempenho.

Figura 3: Sonic da Sega a esquerda e Super Mario da Nintendo a direita
Fonte: Adaptado de Carvalho (2017).

Conforme Marcelo e Pescuite (2009) a partir de 1991 surgiram os jogos em rede que usavam o protocolo TCP/IP, possibilitando que as pessoas conseguissem jogar entre si. Em 1997, com o aumento da velocidade de conexão e a popularização da banda larga, apareceram os primeiros Massive Multiplayers Online Role Playing Games (MMORPG), é um jogo de representação de papéis online onde os jogadores desempenham o papel de um personagem do jogo em um mundo virtual que, normalmente, é medieval, mágico ou ambos podendo controlar todas as ações da personagem até a iteração com outros jogadores através das personagens deles, baseado nos videogames mais avançados e no clássico RPG de mesa.

Ainda que mundos virtuais tenham muitas aplicações além de entretenimento, sendo usados como simuladores militares e até com fins empresariais, os jogos de computador online ainda são o topo da cadeia de mercado de desenvolvimento, nessa categoria. Mundos virtuais são ambientes de socialização, comunidade, iterpretação de papéis e iteração (MASTROCOL, 2007).

1.2. Gênero

De acordo com Azevedo et al (2005) o gênero é o clima do jogo onde é obtido pelo conjunto formado por: objetivo, estrutura, elementos e iluminação. Segundo Chandler (2012) classificar os jogos em gêneros, ajuda os desenvolvedores, obtêm uma melhor compreensão da mecânica do jogo. Para Ferreira (2015) os gêneros estão fundamentados em diferentes assuntos, interface do usuário, ambiente e plataformas. Os jogos evoluiram extraordinariamente, desde os jogos baseados em textos até os jogos 3D atuais. Podemos classificar os jogos da seguinte forma conforme a tabela a baixo:

Aventura	O gênero de aventura é definido por ser capaz de percorrer todo o ambiente do jogo, coletando objetos e informações em beneficio da história idealizada.
Ação	Em geral jogos de ação englobam conflitos estratégicos, desafios de exploração e o interesse de resolver enigma simple.
Casuais	O jogo casual é caracterizado por uma jogabilidade simples e

	intuitiva.
Educacionais	Os jogos educionais são desenvolvidos exclusivamente para educação podendo desenvolver nas pessoas habilidades variadas como concentração, comunicação, memorização, raciocínio lógico entre outros.
Esportes	O primeiro jogo desse gênero foi o tênis para dois depois disso criaram vários jogos desse gênero como futebol, baseball, boxe, golfe, tênis entre outros.
Estratégia	Jogos de estrátegia incentiva o raciocinio rápido na tomada de decisões complicadas, dessa forma os jogadores são estimulados a planeja ações a frente dos competidores.
Luta	Esse gênero colocam dois personagens controlados por jogadores, ou jogador, em um combate corpo a corpo ou à distância.
Labirinto	São jogos que basicamente se passam em labirintos.
MMORPG	O MMORPG é considerado um jogo onde várias pessoas jogam simultâneamente, explorando livremente o mundo virtual podendo realizar missões e paramentar seu avatar.
Plataforma	Os jogos de plataforma surgiram nos anos 80 onde o usuário controla o herói fazendo com que salte em plataformas e enfrente desafios para alcançar um objetivo.
Puzzle	Puzzle são jogos que tem o foco em resolver problemas podendo testar várias habilidades como raciocinio lógico, métodos estratégicos e reconhecimento de padrões.
RPG	O gênero de RPG definir jogos que focar na imersão na história, onde os usuários, assumindo o papel de personagens jogáveis em um mundo imaginário.

Simulação em geral	O gênero de simulação são jogos que simular a vida real, neste estilo de jogo é possivel personalizar personagens, construir cidades, gerência uma empresa, prestar treinamentos profissionais e muito mais.
Tiro	Jogos de tiro concentram-se em ações variadas utilizando armas de projeteis com o objetivo de eliminar todos os adversários e obstáculos que aparecem no mapa.

Tabela 1: Sumário do documento de design de jogo
Fonte: Elaborada pelo Autor.

O design de jogo pode vária entre vários gêneros dessa forma criando novos, podendo também melhorar os gêneros já existentes ou até mesmo fazer combinações deles. Segundo Lima (2011) a indústria de jogos é organizada e com essas variações de gênero, ainda pode ser subdividida em várias outras subcategorias, produzindo ainda mais gêneros novos de jogos.

1.3. Documento de Design de Jogo

Para desenvolver um game o processo de criação é longo, apoiando se em documentos elaborados durante todo o processo do design do jogo. Conforme Schuytema (2008), o design de game é "a uma planta baixa – podemos ter a matéria-prima e os técnicos habilidosos para construir uma casa mas, sem um plano, o trabalho não pode continuar sem uma direção real", pois contém determinadas características e fases que são necessárias para o projeto.

O documento de design de jogo (do inglês Game Design Document ou GDD) é um artefato de importância no processo de desenvolvimento de jogo. Segundo Schuytema (2008) o documento é considerado como a planta baixa do projeto, sendo o documento mais importante na criação do jogo. De acordo com esse mesmo autor o GDD tem várias formas e modelos desde um documento pequeno a Wiki. Entenda como Wiki sites colaborativos que possibilitam criar, adicionar, modificar e excluir conteúdo, detalhadas com grandes volumes tendo várias páginas criadas. Conforme Perucia et al. (2007), o GDD é parecido com um script, de filme, onde aborda todos os detalhes do game. Para Berto e Begosso (2016), o GDD é o documento que ajuda o desenvolvedor nas etapas do projeto, não tendo um modelo exato a ser utilizado, cada documento é único podendo se ajustar ao jogo que está sendo desenvolvido. A concepção

do GDD precisar ser feito antes da etapa de desenvolvimento inicia, assim facilitando encontra pontos importantes, pois determinar onde se iniciará a implementação (BERTO e BEGOSSO, 2016).

O GDD deve conter qualquer detalhe do game por escrito. Conforme Schuytema (2008) no momento em que estiver descrevendo sobre objetos, itens e personagens do jogo, e apresenta uma definição abrangente dos aspectos para que o time de desenvolvimento possa criar de fato o jogo (SCHUYTEMA, 2008). A tabela abaixo mostrar a estrutura básica do GDD utilizado por Schuytema.

I.	Visão geral essencial
	a. Resumo
	b. Aspectos fundamentais
	c. Golden nuggets
II.	Contexto do game
	a. História do game
	b. Eventos anteriores
	c. Principais jogadores
III.	Objetos essenciais do game
	a. Personagens
	b. Armas
	c. Estruturas
	d. Objetos
IV.	Conflitos e soluções
V.	Inteligência artificial
VI.	Fluxo do game
VII.	Controles

VIII.	Definições
IX.	Referências

Tabela 2: Sumário do Documento de Design de Jogo
Fonte: Schuytema (2008).

Essa tabela pode variar conforme a necessidade do desenvolvedor, podendo-se criar uma documentação diferente dessa, aproveitar todo esse modelo de documentação, acrescentar novos elementos ao documento ou apreoveitar parte do documento. No Anexo A podemos ver os detalhes do GDD.

Outra seção que poderia ser adicionada ao GDD seria sons, músicas e efeitos sonoros, tendo em mente que a tabela pode variar de acordo com a necessidade do desenvolvedor, são elementos que contribuem na interatividade do jogo com o jogador. Segundo Torres (2015) os sons, músicas e efeitos sonoros intersificam conteúdo do jogo e induzem os jogadores a explorar o mapa do jogo. Deixando o clima do jogo realístico, os sons, músicas e efeitos sonoros podem contribuir no trajeto, mostrando o quanto os personagens progredirão durante o jogo. Conforme Schuytema (2008) os efeitos sonoros atuam como um feedback, mostrando que as ações escolhidas pelo jogador tiveram efeitos e também ajudar a se localizar no cenário do jogo.

Ao Finalizar o documento, o resultado será um documento claro e conciso, onde todas as partes possam ser usadas como referência. Segundo Torres (2015) o documento também possibilita ser utilizado como um checklist para que os desenvolvedores possam consultar.

1.4. UML

Modelos de software são representações que simplificam os conceitos e objetos do mundo real, evidenciando suas fundamentais características, conforme as necessidades do projeto (SOMMERVILLE, 2011). Qualquer modelo conforme Sommerville (2011), apresenta um ponto de vista particular de um processo, desse modo proporcionar informações especificas a respeito dele. Modelos de softwares tem a possibilidade de ser visualizados através de representações gráficas (SOMMERVILLE, 2011). Uma das linguagens mais respeitada e usada para representação gráfica de modelos de software é a UML.

Embora o processo de desenvolvimento de jogos seja diferente em comparação ao software corporativo, a UML proporcionar uma documentação que permiti uma maior

comunicação entre times multirreferencial (SANTOS e ALVES, 2015). Conforme Oliveira e Torres Filho (2014), a UML é uma linguagem-padrão para a construção da estrutura do projeto. De acordo com Ramos (2006) “a estrutura de conceitos da UML é razoavelmente abrangente, consistindo num conjunto variado de notações que podem ser aplicadas em diferentes problemas e em diferentes níveis de abstração”. O principio básico da UML são a simplicidade e correlações, em relação, a unificação de elementos diferenciados presentes nos diversos métodos (RAMOS, 2006).

Os diagramas da linguagem UML conforme Ramos (2006) são:

- Diagramas de casos de uso;
- Diagramas de classes;
- Diagramas de comportamento;
- Diagramas de estados;
- Diagramas de atividades;
- Diagramas de interação (diagramas de sequência);

1.4.1. Diagrama de caso de uso

O diagrama de caso de uso tornar compreensível um conjunto de atores, de casos de uso e de suas relações (RAMOS, 2006). De acordo com Sommerville (2011) os diagramas de casos de uso apresentam um ponto de vista mais simples de uma interação. Conforme Ramos (2006) a ligação entre um caso de uso e de um ator condiz com a relação de comunicação entre os dois elementos. Os diagramas de caso de uso exibem as interações através do sistema e seu ambiente, Figura 19, (SOMMERVILLE, 2011).

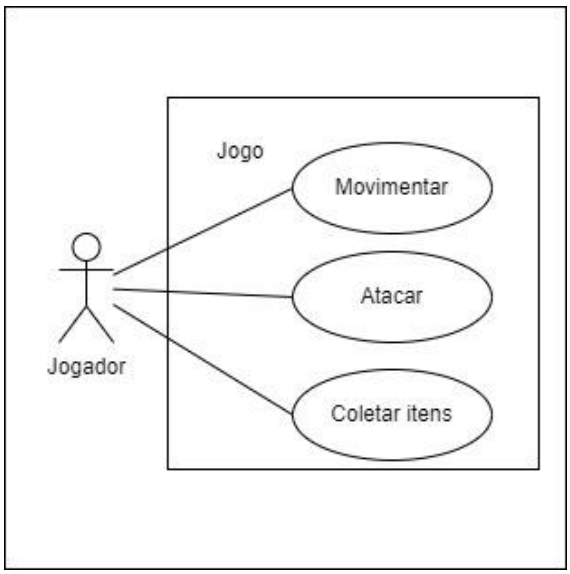

Figura 4: Diagrama de caso de uso
Fonte: Elaborada pelo Autor.

1.4.2. Diagramas de classe

Os diagramas de classe segundo Sommerville (2011) são utilizados para desenvolver um modelo de sistema, orientado a objetos, que exibir as classes de um sistema e as associações entre elas. De acordo com Ramos (2006) o diagrama de classes mostra um conjunto de classes, colaborações, interfaces e relações, geralmente, de associações, de dependência e generalização. Na UML os diagramas de classe tem a possibilidade de ser expressos em níveis distintos de detalhamento. Quando um modelo é desenvolvido, normalmente a primeira coisa é olhar para o mundo, identificando quais objetos são essenciais e representar eles através de classes (SOMMERVILLE, 2011). Conforme Sommerville (2011) a forma mais simples é escrevendo o nome da classe em uma caixa. Na Figura 20 mostrar um diagrama de classe simples com duas classes.

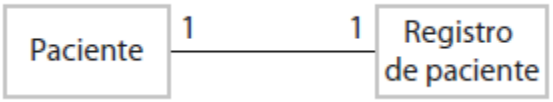

Figura 5: Diagrama de classe
Fonte: Sommerville (2011).

1.4.3. Diagramas comportamentais

Os diagramas comportamentais detalham o comportamento do sistema, em partes, ou processos de negócio referentes ao sistema. Conforme Oliveira (2016) os diagramas comportamentais são aqueles nos quais existe qualquer modificação no comportamento das classes.

1.4.3.1. Diagrama de estado

O diagrama de estado segundo Ramos (2006) possibilita modelar o funcionamento interno de um objeto determinado, subsistema ou sistema global. Os diagramas de estado conforme Sommerville (2011) indicam os estados do sistema e os eventos que resultam nas mudanças de uma estado para outro, não mostrando os fluxos de dados dentro do sistema, porém conseguem incluir dados adicionais dos processamentos feitos em cada estado. De acordo com Ramos (2006) estes diagramas demostram os prováveis estados de um objeto, suas respectivas mudanças entre estados, os eventos que provocar as mudanças no estado e as operações, de ações e atividades, realizadas dentro de um estado ou no decorrer de uma transição. Os diagramas de estado são representados por retângulos arredondados, podendo conter uma breve descrição das ações escolhidas nesse estado, tendo setas possui informações que simbolizam estímulos que impõem uma mudança de estado para outro (SOMMERVILLE, 2011). Conforme Ramos (2006) os estados são simbolizados por retângulos com bordas arredondadas com exceção dos estados inicial e final que possuem ícones específicos, e a mudanças de estado é representada por uma linha cheia dirigida. Segundo Sommerville (2011) a modelagem baseada em estados possui um problema no número de estados possíveis que aumenta muito rápido, precisando ocultar detalhes nos modelos, uma forma de resolver é utilizando a noção de um superestado que encapsula um número de estados diferentes. Vemos um exemplo de diagrama de estado de uma lâmpada, na Figura 21, onde a mudança de estado entre acessa e apagada, conforme ligar e desligar o interruptor.

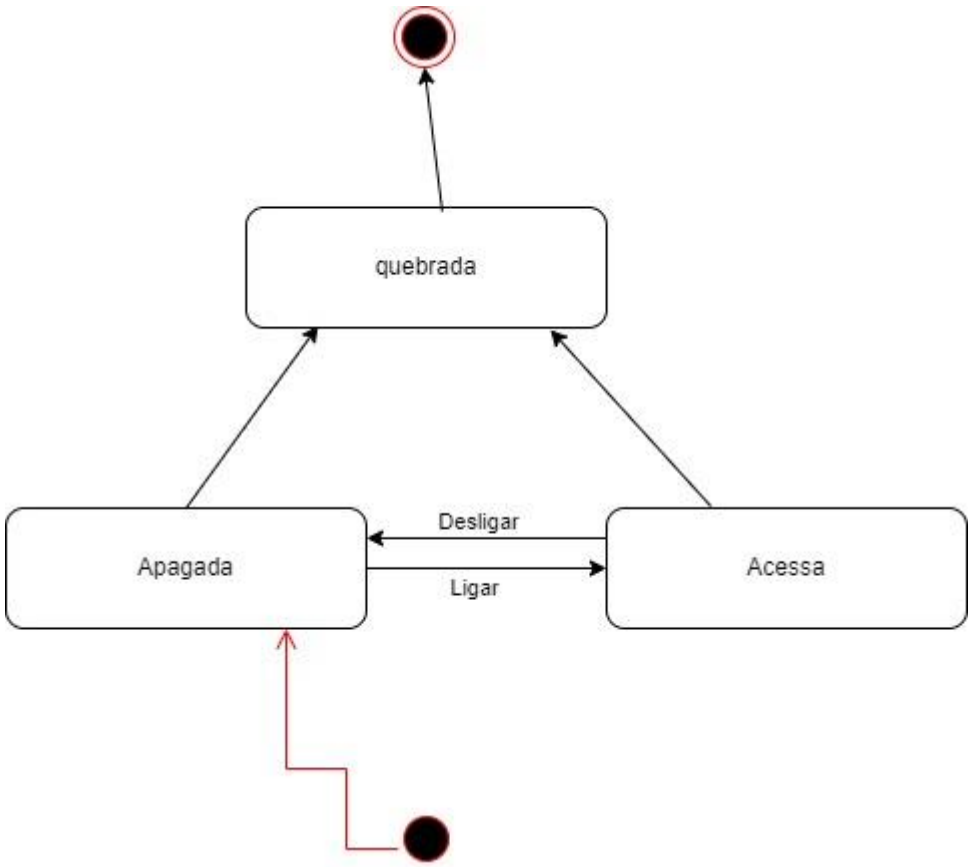

Figura 6: Diagrama de estado de uma lâmpada
Fonte: Elaborada pelo Autor.

1.4.3.2. Diagramas de atividades

Os diagramas de atividades conforme Sommerville (2011) exibir as atividades que constituem um processo de sistema e o fluxo de controle de uma atividade para outra, no começo de um processo é mostrado um círculo cheio e outro no final com um círculo cheio dentro de outro círculo, também possui retângulos com bordas arredondadas que representar as atividades. No diagrama de atividade as setas simbolizam o fluxo de trabalho da uma atividade para outra (SOMMERVILLE, 2011). Segundo Ramos (2006) o diagrama de atividade é usado para moldar o tempo de duração de um objeto ou do sistema, visto que a visualização das operações feitas pelos objetos intervenientes oferece uma visão mais simples do fluxo de controle de um processo ou de uma operação. Um exemplo de diagrama de atividade, vemos na Figura 22.

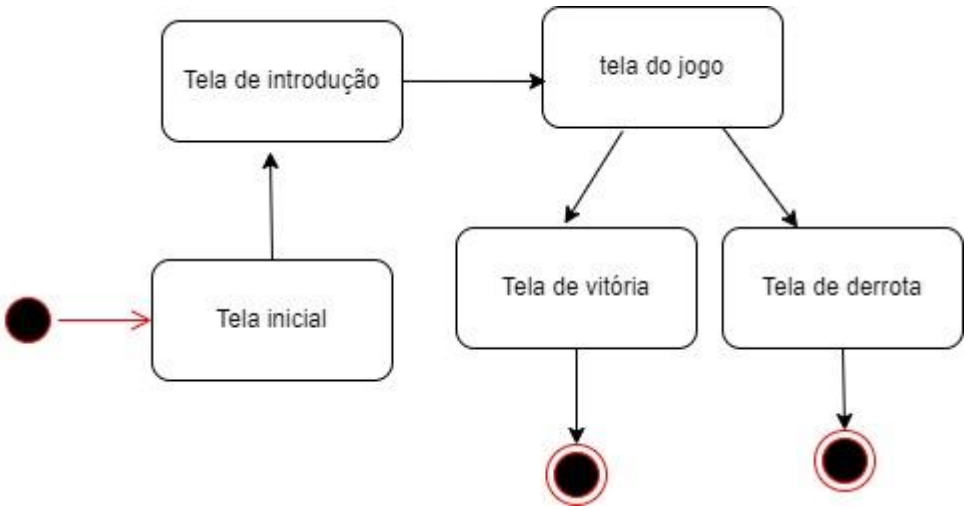

Figura 7: Diagrama de atividade
Fonte: Elaborada pelo Autor.

1.4.4. Diagramas de interação

Os sistemas em geral resultam em algum tipo de interação, podendo ter interação do usuário que resultam em entradas e saídas, interação entre um sistema sendo desenvolvido e outros sistemas, ou interação entre dois ou mais componentes do sistema (SOMMERVILLE, 2011). De acordo com Ramos (2006) uma interação é demonstrada como um diagrama de interação pode ser utilizado para especificar o uso de um caso de uso e também utiliza-lo para uma operação envolvendo objetos diferentes. O diagrama de interação pode ser apresentado como diagramas de sequência.

1.4.4.1. Diagramas de sequência

Os diagramas de sequência segundo Sommerville (2011) são utilizados para modelar as interações entre os objetos e os atores no sistema e as interações entre objetos, mostrando a sequência de interações que acontecem durante um caso de uso específico ou em uma instância de caso de uso. De acordo com Ramos (2006) o diagrama de sequência, Figura 23, demostrar uma interação conforme uma visão temporal. O diagrama é representado por duas dimensões, a horizontal que caracteriza o conjunto de objetos e a vertical que caracteriza o tempo (RAMOS, 2006). Segundo Ramos (2006) as setas do diagrama de sequência são ilustradas horizontalmente de modo a representar a indivisibilidade da operação sendo indispensável para enviar o estímulo.

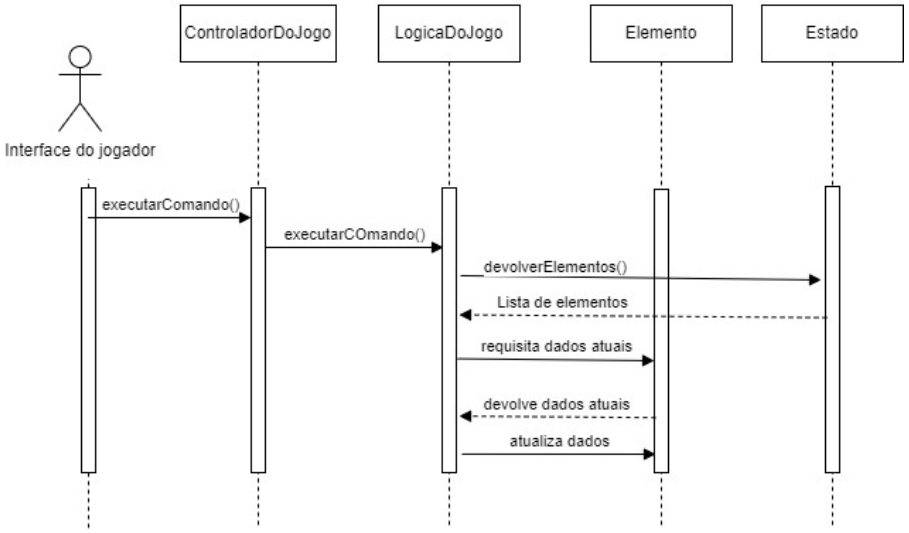

Figura 8: Diagrama de sequência
Fonte: Elaborada pelo Autor.

1.5. Unity

Para elaboração de um jogo é necessário conhecimento em diversas áreas como técnico em 3D, editor de som e vídeo, programador, design de jogo e etc. Trabalhar com todas essas áreas é extremamente difícil, pois necessita de conhecimento técnico em áreas distintas. Para ajudar a resolver esse problema a Unity possui ferramentas e técnicas que simplificam o processo de desenvolvimento de jogos, auxiliando os denvolvedores em áreas complexas (TORETTI JUNIOR, 2013). Conforme Alves (2021), a Unity é uma das engines mais usadas mundialmente pela eficiência e simplicidade no desenvolvimento de jogos, oferecendo aos desenvolvedores a oportunidade de criar jogos 2D e 3D. De acordo com Belarmino (2017) há várias ferramentas para desenvolver jogos possibilitando aos desenvolvedores que não possuem conhecimento nenhum possam criar seu próprio jogo, a Unity fornece diversos recursos disponíveis fazendo com que os desenvolvedores tenham mais atenção ao funcionamento do jogo.

A Unity é uma ferramenta que permite criar jogos utilizando um editor visual, tendo uma interface fácil e simples de utiliza, possui GameObjects que constituem objetos no jogo podendo ser posicionados na visualização da cena (do inglês Scene view), selecionando um objeto na cena na aba inspector pode ser observado as informações desse objeto, na barra de ferramentas possui recursos necessários para manipular o objeto em cena. Utilizando hierarchy exibi os GameObjects que podem ser utilizados no jogo. No project fica a biblioteca de asserts

onde fica os asserts e prefabs. Na aba game fica a demonstração da tela do jogo. Nas seções seguintes irá ser explicado em mais detalhes os pontos mencionados.

1.5.1. Game Objects

A Unity funciona de acordo com Ferreira (2015) com o gerenciamento de GameObjects que constituem objetos que se encontram dentro do projeto e se estendem a objetos sólidos, inanimados e outros. Segundo Torres (2015) a Unity funciona fundamentado em cenas dentro delas existem game objects, que são objetos colocados dentro da cena. Conforme Etto (2016), o GameObject são objetos integrados a cena do projeto e suas características podem ser modificadas por Components conhecidos como Scripts que possibilitam lançar eventos, alterar as propriedades de outros Scripts ou do GameObject no decorrer do jogo, além da interatividade com o jogador. Segundo Keulen et al. (2013), a Unity é ordenada em volta dos game objects, tangiveis ou intangiveis, podendo incluir componentes como Scripts e outros elementos ao game objects fazendo com que deixem de ser so objetos simples na cena. Um exemplo básico de game object seria o cubo mostrado na Figura 27.

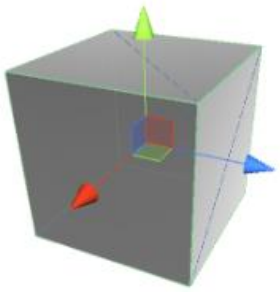

Figura 9: Um simples Cubo GameObject
Fonte: Unity (2016).

1.5.2. Interface da Unity

A interface da Unity conforme Ferreira (2015) é fácil e simples de aprender em pouco tempo de uso torna-se possivel dominar as interações por meio dos visualizadores. Segundo Toretti Junior (2013) a Unity possui uma interface do editor de projetos personalizada, Figura 28, podendo modificar a posição, tamanho ou inserir ou remover os componentes da tela.

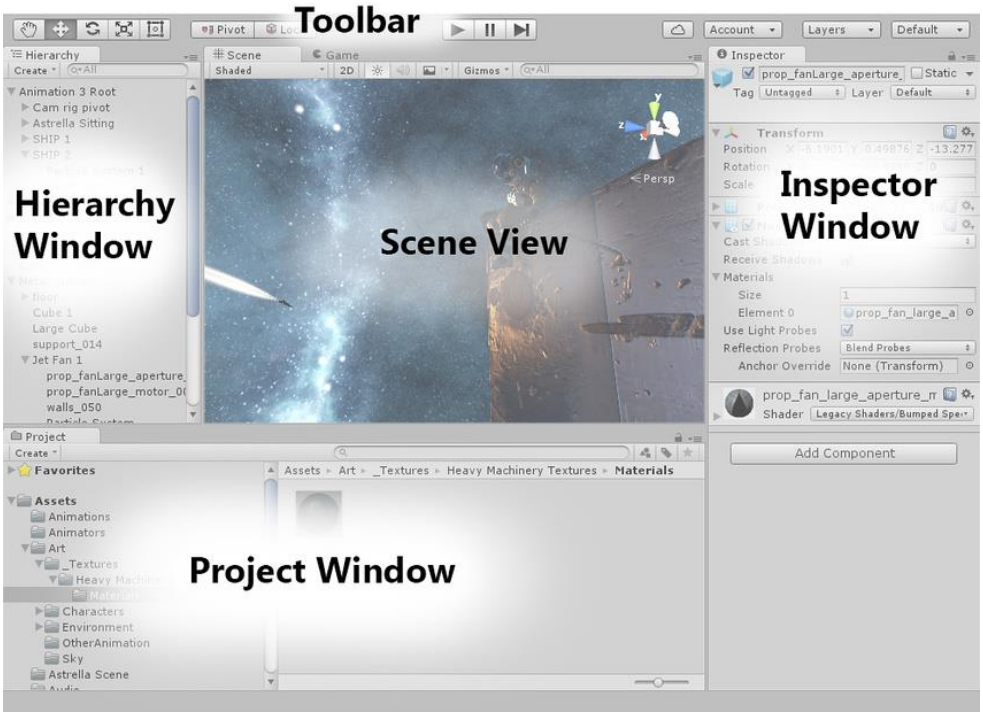

Figura 10: Interface do Unity
Fonte: Unity (2016).

A interface padrão da Unity inicia com Scene View, Toolbar, Inspector, Hierarchy, Project e Game.

1.5.2.1. Scene view

A Scene view, Figura 29, é usada para selecionar e colocar numa determinada posição os objetos do cenário como árvores, cadeiras, casas, jogador, as câmeras, inimigos, e quaisquer outros GameObjects. De acordo com Toretti Junior (2013), scene view é um elemento da interface em que projeto do jogo é desenvolvido e possibilita a visualização dos objetos em cena. Segundo Ferreira (2015), constitui a cena do jogo e onde de fato tudo é desenvolvido, é onde todos os GameObjects e outros itens são colocados. Conforme Etto (2016), deixar editar e navegar visualmente na cena tendo uma perspectiva 3D ou 2D, conforme o tipo de projeto em desenvolvimento. A Scene View dispõe de um conjunto de comandos de navegação que ajudar a se mover de modo ágil e eficaz (UNITY, 2016).

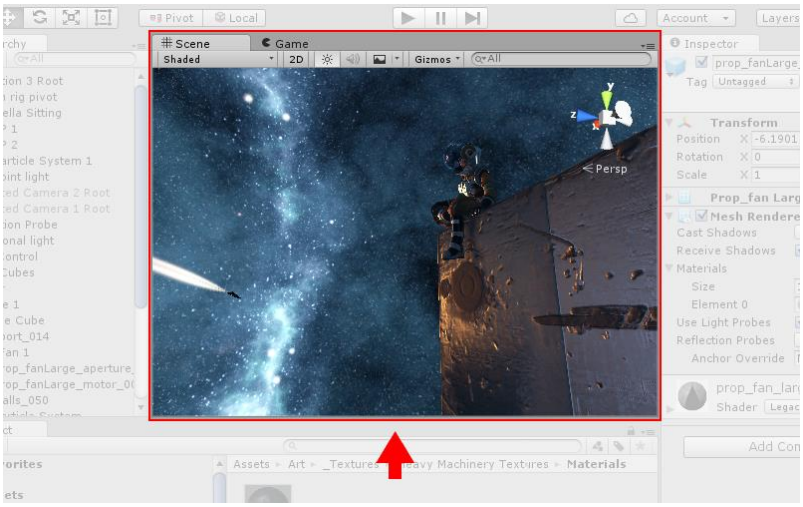

Figura 11: scene view
Fonte: Unity (2016).

A Scene view possui o scene gizmo que mostra a orientação atual da câmera e proporciona alterações de forma rápida no ângulo de visão e no modo de projeção, Figura 30.

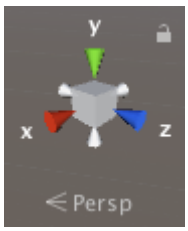

Figura 12: scene gizmo
Fonte: Unity (2016).

1.5.2.2. Inspector

O inspector segundo Toretti Junior (2013) é uma ferramenta que ao selecionar o GameObject mostra informações sobre seus atributos e componentes, Figura 31. O inspector é utilizado para visualizar e editar os objetos do game através das propriedades e configurações, quando selecionado o GameObject na Hierarchy ou Scene view, o inspector exibirá as propriedades de cada componente e materiais desse objeto dessa forma possibilitará a edição (UNITY, 2016). De acordo com Ferreira (2015), todos os GameObjects contém propriedades e detalhamento de diversos parâmentros no projeto, deixando a possibilidade de mudar determinadas informações antes da integração do elemento na cena. Conforme Keulen et al. (2013), no momento em que os componentes são inseridos em um game object, há a possibilidade de ser modificados na aba inspector quando o objeto especifico for selecionado, visto que passam a ser propriedade do objeto.

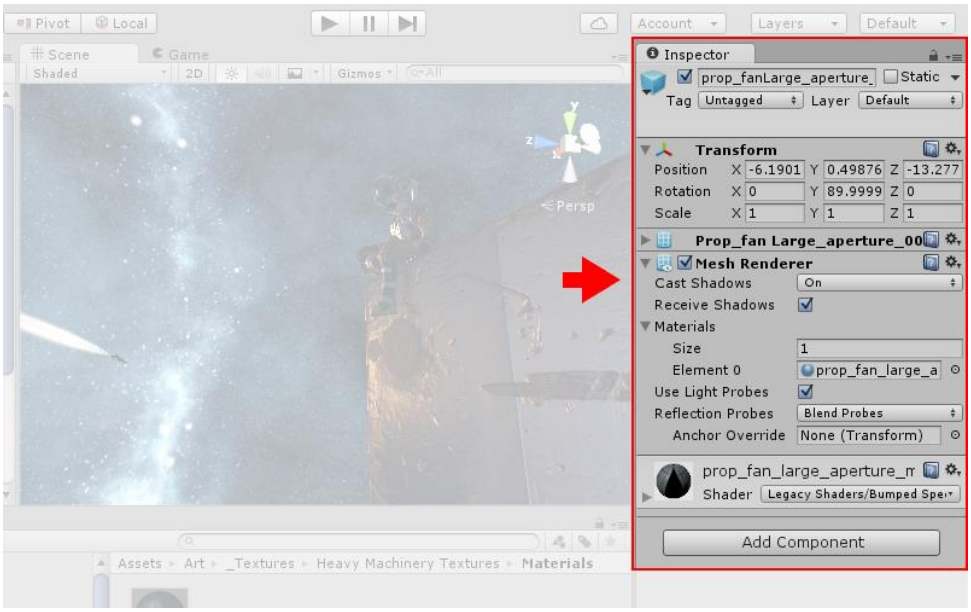

Figura 13: Inspector
Fonte: Unity (2016).

1.5.2.3. Toolbar

O toolbar ou barra de ferramentas segundo Etto (2016) permite acesso aos recursos necessários de trabalho. Conforme Ferreira (2015) a barra de ferramentas possui características que são simplicidade na aprendizagem, velocidade nas alterações e testes. Ainda conforme Ferreira (2015) “Os botões do lado esquerdo são para manipulações dos objetos, como alterar o tamanho, rotação e posição. Os botões do meio são controles de execução do jogo e para testar as suas etapas, alternando do ambiente de criação para o ambiente do jogo”.

1.5.2.4. Hierarchy

A hierarchy exibi cada GameObject em uma cena, podendo utilizar para classificar e agrupar os GameObject que são utilizados na cena. A hierarchy de acordo com Etto (2016) mostra como forma a estrutura dos objetos ligados um ao outro. Segundo Ferreira (2015) a hierarchy faz uma listagem de todos os objetos utilizados na cena, podendo incluir recursos novos.

1.5.2.5. Project

Conforme Etto (2016) project é onde se encontrar a biblioteca de Asserts que estão à disposição para uso no projeto.

1.5.2.6. Game

A aba game segundo Toretti Junior (2013) é usada quando o botão play é apertado e fazendo com que o projeto entre em execução.

1.5.2.7. Asserts

Asserts é todos os ativos para o desenvolvimento de um jogo. Um asserts é qualquer elemento como arquivos de áudio, scripts, modelos 3D e etc. De acordo com Ferreira (2015) os asserts são elementos prontos a serem usados tais como modelos 3D, kits de efeitos de sons entre outros. Para inserir asserts no projeto selecionar o aqruivo e soltar dentro do project view ou utilizar o menu da Unity selecionado assets depois em import new asset, desse modo, o asset ficara disponivel para ser usado no jogo.

1.5.2.8. Prefabs

Os prefabs são um conjunto de game objects e components podendo ser usados nas cenas do jogo. Um único prefabs pode gerar vários objetos idênticos, caso for necessário alguma mudança em qualquer objeto, é só fazer a alteração no prefab e todos herdarão essa mudança. Conforme Ferreira (2015) os prefabs são necessarios quando precisa ter vários objetos repetidos, quando ocorrer alguma mudança em algum desses objetos, estende-se essa alteração para os outros objetos também.

O capitulo 2 apresentou todo o conteúdo teórico estudado para fundamentação desse projeto. O tópico 2.1.1 mostrar a história dos jogos eletrônicos do surgimento do primeiro jogo, dos consoles e dos jogos em rede que usavam o protocolo TCP/IP. O tópico 2.2 apresentou os gêneros dos jogos para obtêm uma melhor compreensão da mecânica desses jogos. O tópico 2.3 apresentou e explicou os elementos do GDD onde as partes desse documento possam ser usadas conforme a necessidade do jogo, de forma clara e concisa, e também possibilita ser

usado como checklist para consultar. O tópico 2.4 dar uma visão geral da UML sobre requisitos funcionais e não funicionais, casos de uso e diagramas, existem muitos detalhes que devem ser aprendidos para utilizar com mais eficiência. O tópico 2.5 explicou o que é a Unity, game objects e interface da Unity que possui elementos que servem para facilitar o desenvolvimento do jogo.

ESPECIFICAÇÃO DO PROJETO

Este capítulo apresentar o GDD do jogo, Return Seven, elaborado durante o desenvolvimento do TCC, além dos diagramas referentes ao jogo criado. O modelo dessa documentação segue a linha de raciocínio do Schuytema (2008) e a utilização da UML como material de apoio para melhor entendimento do desenvolvedor.

2.1. Visão Geral Essencial

Este tópico apresenta uma visão do jogo de uma forma breve, de modo que, qualquer pessoa possa se familiarizar com o conceito do jogo entendendo sua ideia e jogabilidade.

2.1.1. Resumo

Sete **Chaves**. Sete **Possibilidades**. Sete **Caminhos**.

Em um futuro distante, onde a humanidade voltou a um estado pré-civilização, a Terra se tornou um planeta desolado, cheio de sequelas das guerras nucleares do passado. Um dia nossa heroína, enquanto explorava na floresta descobre um mistério que pode ser a **chave** do futuro da humanidade. Return Seven colocará o jogador no controle da personagem que irá lutar contra as forças do mal para salvar a humanidade.

2.1.2. Aspectos fundamentais

O jogo tem o estilo parecido de Horizon Zero Dawn que é um RPG de ação pós-apocalíptico. O jogador poderá controlar o personagem enquanto ele explorar o mapa e também necessitará exibir suas habilidades no controle do personagem para salvar o mundo do vilão.

Ao longo do jogo o personagem irá enfrenta os desafios propostos, derrotar os inimigos e obter a vitória sobre o vilão do jogo, além da possibilidade de fazer quests, encontra powerups e obter itens de inventário.

2.1.3. Golden nuggets

O jogo não possui um diferencial quanto a sua criação ou um método mais elaborado de desenvolvimento, mas seu diferencial está na sua história e trama que alterar de acordo com a forma que o jogador avança no game. Return Seven se passa em um mundo pós-apocalíptico onde o jogador começará com a heroína numa pequena vila, privada de toda e qualquer habilidade, mas cheia de atitude e coragem.

2.2. Contexto do Game

O “Contexto do Game” descreve o mundo do jogo aonde o jogador irá se aventura, em resumo a história do jogo.

2.2.1. História do Game

O jogo se passa em um mundo futurista pós-apocalíptico, onde o uso espadas e arco e flechas se fez necessário novamente.

Até que um dia, Nana, nossa heroína, enquanto explorava a floresta, encontra-se com uma luz piscando, escondida no meio de destroços e vai investigar, e encontra uma máquina estranha, mostrando uma mensagem enigmática: (7...777...7...7...7...777777). Curiosa, pega o dispositivo e continua andando pela floresta e repara que o bip vai ficando mais rápido e fonte enquanto segue em uma certa direção.

Até que acha uma porta de algo parecido com uma dungeon cheio de folhagem e símbolos esquisitos em cima e vê um slot/buraco que é exatamente do tamanho da máquina que está andando intuitivamente aproxima o dispositivo, e vê a porta se abrir. Dentro da dungeon, encontra uma sala com várias máquinas funcionando e mostrando telas vermelhas com uma advertência.

WARNING: ALL SERVERS OFFLINE, CANNOT CONTINUE... 777777777777777 7 77 7777.

Apertando um botão aleatório na sala, de repente um mapa aparece.

A ideia é que existem sete chaves perdidas no mundo que devem ser encontradas para que as máquinas possam ser reiniciadas.

2.2.2. Eventos anteriores

O mundo foi destruído pelas guerras nucleares causadas por um homem ambicioso que de propósito causou o apocalipse para facilitar sua ascensão ao poder.

A Nana é um personagem forte e independente. Perdeu seus parentes quando era criança, e sobreviveu sozinha na floresta até ser encontrada por grupo de pessoas e levada para uma pequena vila, por vive um tempo na floresta Nana adquiriu habilidades de sobrevivência, como caça.

Krauber, o vilão, um homem ambicioso causou o apocalipse para adquirir poder, ficou gravemente ferido após luta com os pais de Nana causando a mortes deles. Sofreu várias cirurgias e melhoramento físico para ficar mais forte.

2.2.3. Principais jogadores

Nana: é a protagonista do jogo, e sua intenção é salvar o mundo. Ela começa praticamente sem nenhuma habilidade, ao longo do jogo, pode obter itens que darão suporte na sua aventura.

Krauber: vilão principal é um dos personagens principais que Nana está tentando tira o poder dele para restaurar o mundo. Ele descobre o esforço de Nana em tentar impedi-lo e criar obstáculos para impedir seu avanço em encontrar as sete chaves. O combate final do jogo será entre a Nana e o chefão.

2.3. Objetos essenciais do game

Este tópico descreve os objetos que surgem no jogo, podendo influência na jogabilidade e experiência do jogo.

2.3.1. Personagens

Nana: é uma mulher forte e bonita, treinada pelos guerreiros da vila para uma vida dura e difícil, é o alter ego do jogador. A Nana ficar mais forte conforme encontrar armas e objetos diversos durante o jogo.

Guerreiros da vila: protegem a vila de ataques de monstros.

Zumbi: após término da guerra nuclear essas criaturas sugiram. Eram pessoas normais que viviam suas vidas, mas graça a ganancia do Krauber e das guerras nucleares que ele proporciono. Por causa dessas guerras houve mutações genéticas que transformaram algumas pessoas em mortas vivas. Os zumbis aparecem em diversas partes do jogo dificultando a vida da heroína.

Krauber: Odiado por todos governa o mundo com punhos de ferro. Ele aparece sempre que Nana está próxima de encontrar uma das chaves para salvar o mundo.

2.3.2. Armas

O jogador possui apenas a arma inicial, uma espada longa. Conforme avança no jogo e sobe de nível a arma acompanha esse avanço ficando mais forte.

2.3.3. Estruturas

As estruturas principais do jogo são:

- Checkpoints: locais onde o jogador aparece após a morte do herói;
- Vila: local onde se encontra a heroína no início da história;
- Sala do vilão: local onde vai ocorrer a batalha final.

2.3.4. Objetos

Chaves

Imagem: uma chave velha e antiga.

Onde pode ser encontrada: protegidas por zumbis e pelo vilão em determinadas áreas do jogo.

Comportamento: as sete chaves constituem a principal busca do jogo. Quando todas as chaves são encontradas pode ser usada para acessa toda tecnologia escondida pelo vilão para salvar a humanidade.

2.4. Conflitos e soluções

Há vários tipos de sistemas de conflito no jogo referente a este TCC, com as seguintes soluções:

- Chave: as chaves coletadas durante o jogo só será utilizadas no final para ativar as tecnologias perdidas;
- Movimentação: a personagem Nana, controlada pelo jogador poderá se movimenta livremente no mapa sendo apenas limitada pelo tamanho do mapa e das estruturas presentes no mesmo;
- Ataque básico: um ataque normal no qual o personagem, controlado pelo jogador, possui o dano baseando no seu nível, no qual possui um som especifico e animação, acertando um inimigo reduz a vida dele conforme o cálculo do dano;
- Inimigo: o inimigo esperar a aproximação da heroína para pode atacá-la;
- Vilão: O vilão aparece cada vez que uma chave é encontrada. Cada chave encontrada aumenta a força do vilão.

2.5. Inteligência artificial

Return Seven possuirá uma inteligência artificial limitada, os inimigos terão determinados comportamentos que pode variar conforme determinadas condições, entretanto os comportamentos básicos são os seguintes:

Esperar: enquanto a heroína, o jogador, está fora do alcance do inimigo, ele estará no estado em esperar. Quando a heroína se aproxima acionará, o inimigo que sair do modo de esperar para o modo de perseguição e só retornara ao estado de esperar se for derrotado ou se a heroína sair do alcance de visão dele.

Patrulhar: os inimigos possuem uma rota definida por um início e um ou mais destinos. Em um determinado tempo o inimigo deve seguir para o próximo destino, chegando deve ficar em modo de esperar até um determinado tempo passa ou quando a heroína, o jogador, se aproxima. Inimigo pode sair do modo de patrulha quando o jogador se aproxima e aciona o modo de perseguição.

Perseguir: quando o jogador chegar a uma determinada distância, é iniciada a perseguição, mas o inimigo não pode se distanciar da sua rota. Quando jogador se distância o inimigo volta para modo de patrulha.

Atacar: quando inimigo se aproximo do jogador, ele causa dano com esse ataque. O inimigo só parar de atacar quando o jogador derrota ele, se for morto ou o jogador se distância do inimigo fazendo com que ele volte para o modo de patrulha.

2.6. Fluxo do game

Tela inicial do jogo possui as opções de jogar, créditos e sair. Selecionando jogar, é iniciado o jogo no primeiro mapa onde terá que chegar numa estrutura misteriosa e tendo que enfrenta inimigos ao longo do caminho. Chegando nessa estrutura misteriosa o jogador será transportado para outro mapa onde novamente enfrentaram inimigos e terá que fazer uma escolha por qual caminho seguirá até chegar no vilão com todas as sete chaves. Na última batalha contra o mal, ao vencer o vilão e com as sete chaves conseguirá fazer com que a tecnologias perdidas voltem a funcionar ajudando dessa forma a salvar o mundo.

2.7. Controles

Os controles de Return Seven serão muito simples. São os seguintes: Controle por teclas para movimentação W para frente, D para direita, A para esquerda e S para trás. A tecla L para correr, a tecla K para atacar e J para recolher itens.

2.8. Áudio

O jogo tem música na tela inicial do jogo, música de fundo e sons básicos.

2.9. Referências

O jogo desenvolvido para este TCC teve como referência os jogos de RPGs em geral e com destaque ao jogo Horizon Zero Dawn. Outras referências foram de filmes como Wall-E, 11-11-11, Mad Max, A noite dos mortos vivos (1968), The road (2009) e A colônia.

2.10. UML

Nesse tópico será apresentado as informações, definições e diagramas relacionados ao desenvolvimento do jogo.

2.10.1. Diagramas

Nesse tópico mostra os diagramas relevantes para desenvolvimento do jogo.

2.10.1.1. Diagrama de caso de uso

Este diagrama demostrar as atividades do ator dentro do ambiente do jogo, dessa forma especifica quais as funcionalidades desse ator e o que ele é capaz de fazer dentro do jogo. Na Figura 19 vimos o diagrama de caso de uso do jogador, que demostra os comportamentos que o personagem terá no jogo como movimentar, atacar e coletar itens.

2.10.1.2. Diagrama de classes

O diagrama de classes é desenvolvido após a definição dos casos de uso, Figura 33, destacando os elementos encontrados no jogo. Na classe jogo são implementados os métodos para iniciar o jogo e controlar o game loop do jogo. A classe Héroi possui os seguintes atributos e métodos:

- O atributo qtdVida: determinar a quantidade de vida do personagem;
- O atributo qtdPontos: determinar a quantidade de pontos que personagem ganha ao derrota inimigos, é a experiência do jogo;

- O atributo qtdDano: é o dano que o personagem causa no inimigo;
- O método Andar(): faz com que o personagem se movimente no jogo;
- O método Correr(): faz mesma coisa do método Andar() mas numa velocidade um pouco maior;
- O método Atacar(): faz o personagem atacar o inimigo e caso acerte causa dano;
- O método Defender(): faz o personagem não ser acertado pelo inimigo;
- O método UtilizarItem(): faz o personagem utiliza os itens encontrados no jogo.

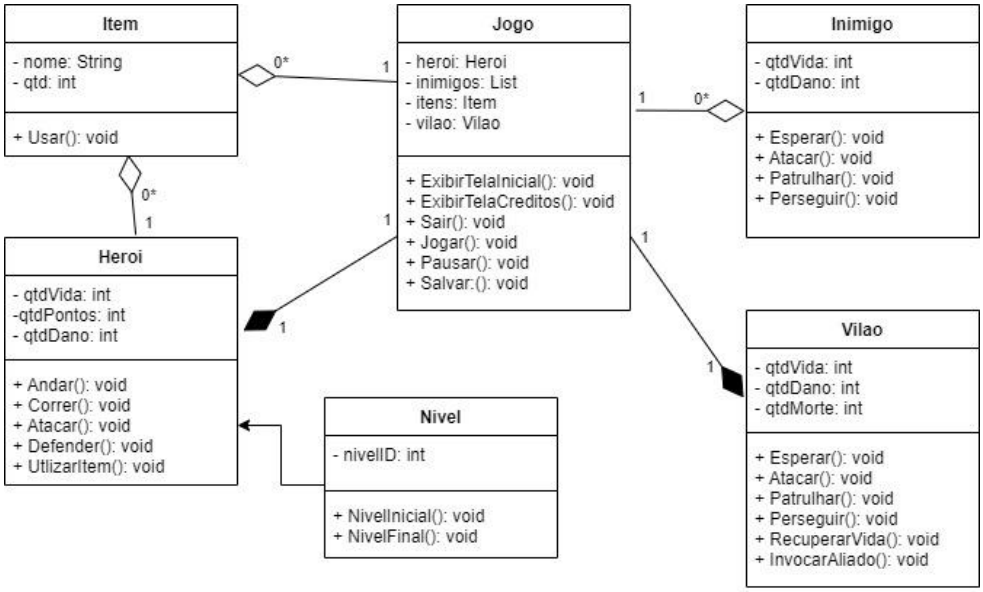

Figura 14: Diagrama de classes
Fonte: Elaborada pelo Autor

A classe item possui o método usar() que ativa a funcionalidade do item e na classe nível controlar a quantidade de pontos do personagem para passa de nível. As classes inimigos e vilão possui os mesmo atributos do herói, qtdVida e qtdDano. O classe vilão possui um atributo qtdMorte que contar a quantidade de vezes que foi derrotado no jogo e métodos RecuperarVida() e InvocarAlido() que são ativados quando atingir uma determinada condição.

2.10.1.3. Diagrama de atividade

O diagrama de atividade, Figura 34, corresponde ao funcionamento do jogo de uma forma simples.

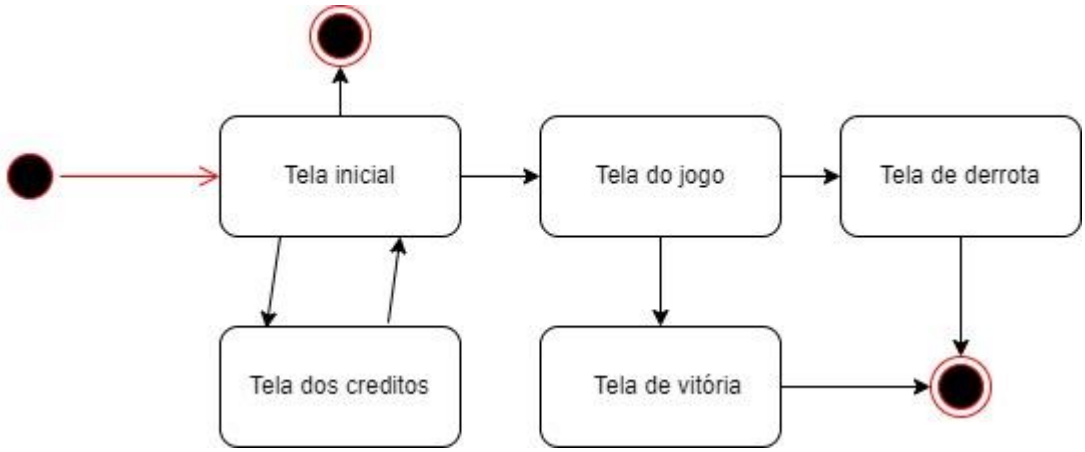

Figura 15: Diagrama de atividade do funcionamento do jogo
Fonte: Elaborada pelo Autor

O diagrama mostra o funcionamento do jogo começando pela tela do jogo com três opções sair, ir para tela de créditos ou inicia o jogo. Após determinadas partidas podem ocorrer duas possibilidades para terminar o jogo, ganhar ou perder. Outro diagrama de atividade gira em torno da vida do personagem, Figura 35, enquanto a vida for maior que zero continuar no jogo permitindo todas as ações que estão no diagrama de caso de uso do jogador.

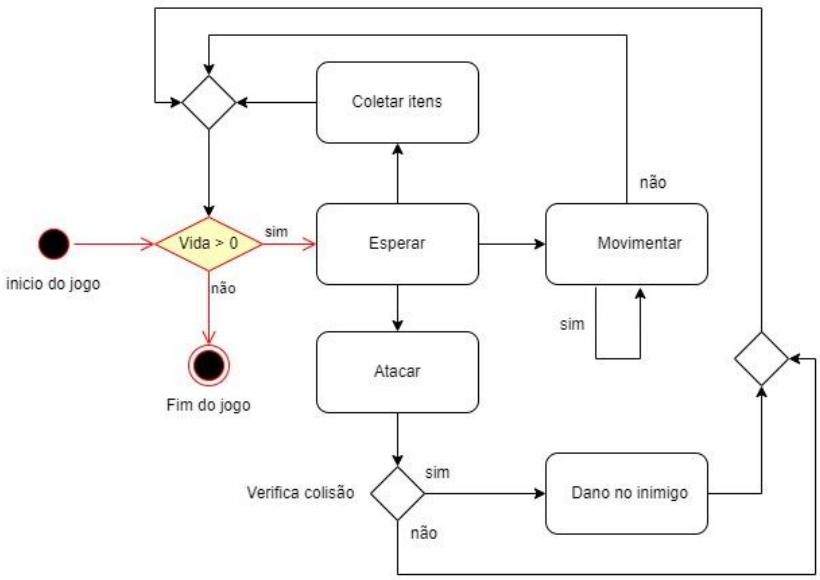

Figura 16: Diagrama de atividade jogador
Fonte: Elaborada pelo Autor

O diagrama de atividade, Figura 36, referente ao funcionamento dos inimigos presentes no jogo. Esse diagrama refere-se ao comportamento do inimigo, enquanto a sua vida é maior que zero, é definido de três maneiras possíveis de iterações, em esperar onde inimigo fica parado esperando o jogador se aproxima, a outra realiza movimentação pre determinada caso ocorra a aproximação do jogador, e por último a perseguição dentro do campo de visão do inimigo, sendo finalizado somente quando o jogador se afastar ou causa dano suficiente para que a vida do inimigo chegue a zero.

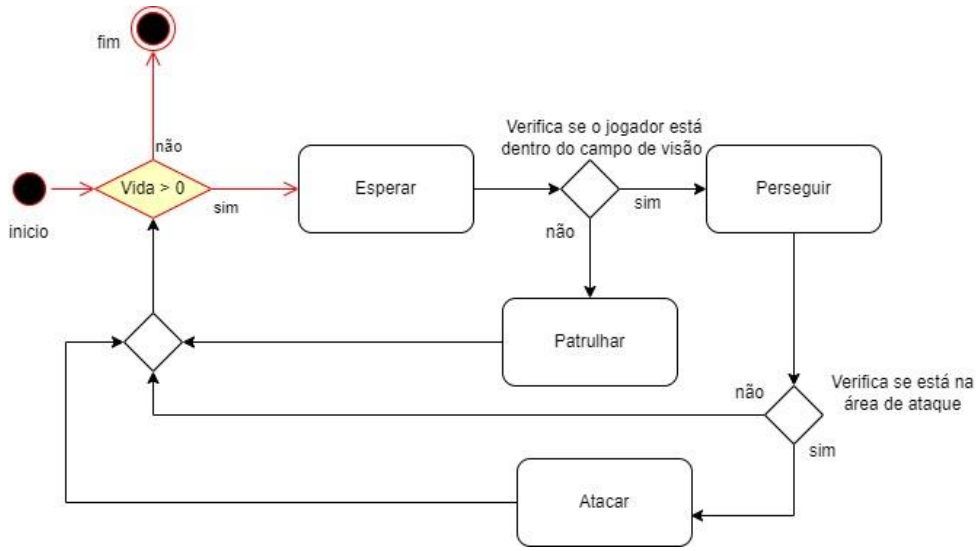

Figura 17: Diagrama de atividade inimigo
Fonte: Elaborada pelo Autor

O diagrama de atividade do vilão do jogo, Figura 37. Deve-se ter cuidado quando se tratar do chefão, pois a batalha contra ele deve ser desafiadora e possível de sucesso para o jogador. O comportamento do vilão é bem parecido em relação ao comportamento do inimigo com algumas diferenças. A primeira diferença é quando o vilão ressuscitar pelo menos uma vez dependendo da quantidade de chaves que foi recolhida pelo herói. A segunda diferença é quando atingir o limite das sete chaves o vilão ressuscitar e invocar um aliado para ajudar na batalha contra o herói.

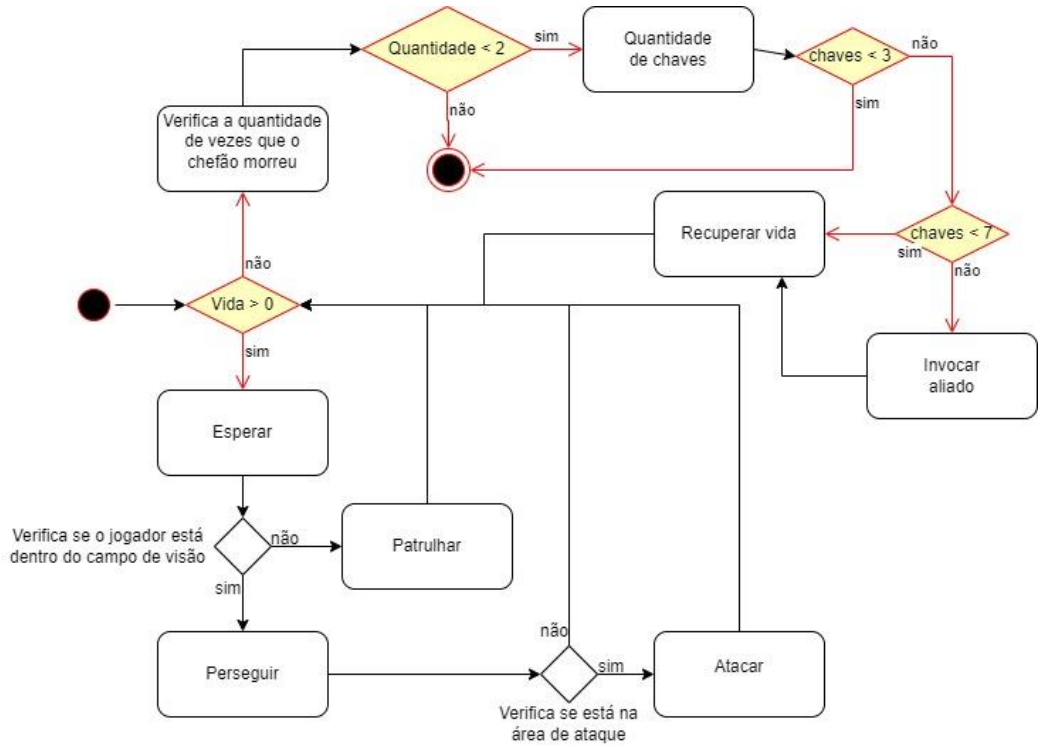

Figura 18: Diagrama de atividade do vilão
Fonte: Elaborada pelo Autor

2.10.1.4. Diagrama de sequência

Este diagrama demostrar as interações entre o ator e o sistema do jogo. O ator realiza o acesso ao jogo através da interface do jogo. Na interface o ator pode realiza três possíveis ações jogar(), créditos() e sair(). Selecionando a opção créditos() o ator vai para seção onde informa os créditos do jogo. Selecionando sair() finalizar o jogo. O ator selecionando jogar() é direcionado para a execução do jogo. No jogo em execução o ator possui as opções Movimentar Personagem(), Pegar item(), Atacar(), Pausar(), Salvar() e sair(). Desse modo é especificado quais as funcionalidades que esse ator é capaz de fazer dentro do jogo, Figura 38.

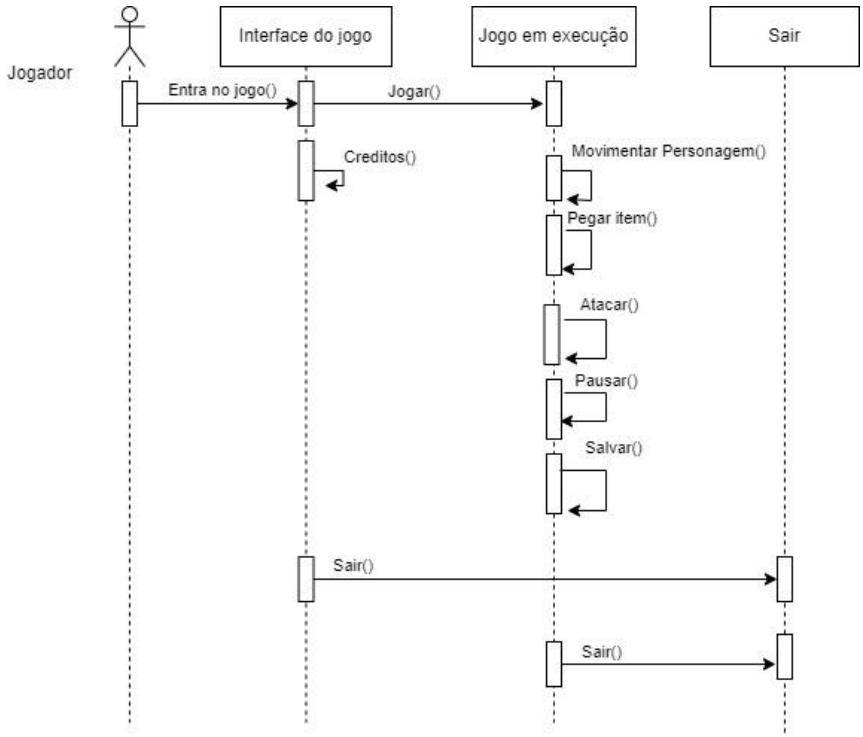

Figura 19: Diagrama de sequência
Fonte: Elaborada pelo Autor

O capítulo 3 apresentou o desenvolvimento do GDD do jogo após a aplicação da metodologia explicada no capítulo 2. O desenvolvimento foi feito seguindo a ordem proposta pelo GDD, iniciando-se pela história do jogo logo após descreve outros itens como personagens, cenário, armas, itens e outros recursos necessários para criar o jogo. A UML é introduzida como parte do GDD, ajudando a ter uma visão melhor do desenvolvimento apresentando os requisitos funcionais e não funcionais do jogo. Esses requisitos foram necessários para possibilitar desenvolver os diagramas.

DESENVOLVIMENTO DO RETURN 7

Neste capitulo do TCC demonstrar a maneira de criar um jogo eletrônico no qual foi apresentado nos capítulos anteriores os processos e elementos para elaboração de um jogo. Neste tópico foi descrito o desenvolvimento do jogo eletrônico, o processo de criação, a lógica por trás dos elementos do jogo e a tecnologia aplicada no desenvolvimento.

3.1. Desenvolvimento das telas e interações

O desenvolvimento iniciou-se pela tela do menu principal, nesta etapa o usuário pode escolher, se deseja jogar, créditos ou sair da aplicação. Há um efeito visual no menu, conforme passa o mouse por cima das palavras expandem e volta ao normal quando tira o mouse de cima.

Assim que o jogo é iniciado a tela que pode ser observada na Figura 39, é apresentada ao jogador com uma música de fundo.

Figura 20: Tela inicial do jogo
Fonte: Elaborada pelo Autor

Na Figura 40, mostra a palavra Jogar expandida quando passa mouse por cima.

Figura 21: A palavra Jogar expandida
Fonte: Elaborada pelo Autor

Para desenvolver essa tela inicia-se criando um scene então se escolhe uma pasta logo após clica no botão direito do mouse selecionando a opção create, logo depois scene conforme a Figura 41.

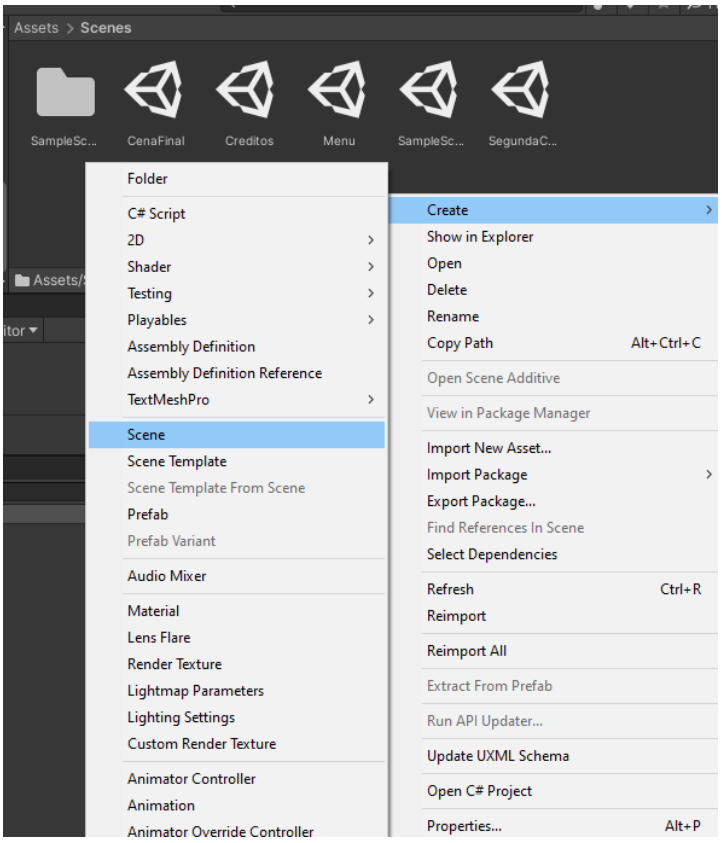

Figura 22: Criando uma cena para jogo
Fonte: Elaborada pelo Autor

Com a cena criada colocar a Scene view no modo 2D e criar uma UI, interface, clicando com o botão direito na hierarchy seleciona UI em seguida image, Figura 42. Na hierarchy após criar a interface surgir canvas e dentro dele um objeto image podendo ser renomeado para background. Selecionando background na aba inspector na parte de image tem uma opção chamada source image que pode se colocar uma imagem para o plano de fundo, Figura 43.

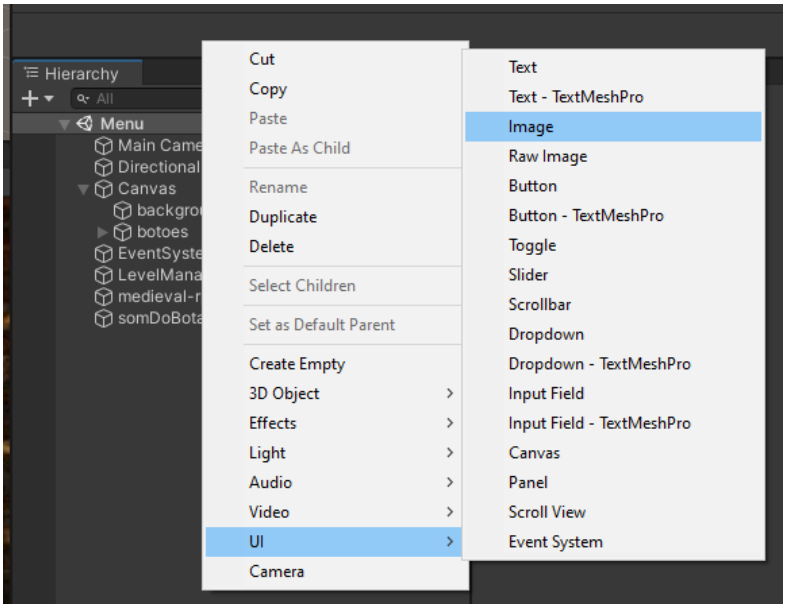

Figura 23: Criando uma interface
Fonte: Elaborada pelo Autor

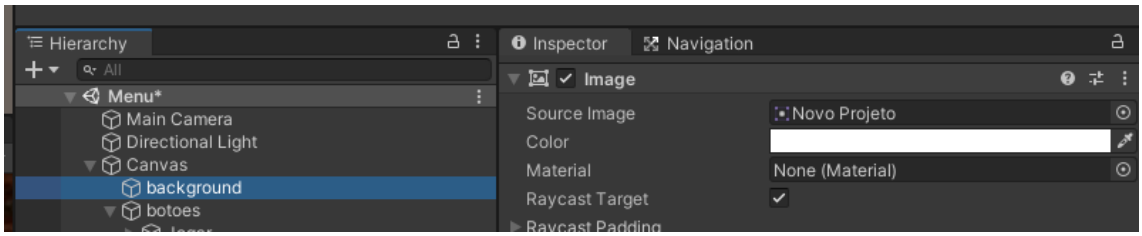

Figura 24: Source image
Fonte: Elaborada pelo Autor

Para criar os botões clicar com botão direito do mouse sobre canvas seleciona UI logo após Button TextMeshPro, Figura 44. Selecionando o botão na aba inspector na parte de image podemos colocar uma imagem do botão personalizado no formato png em source image, Figura 45. Após criar os botões é feito um script para mudança de cena e sair do jogo.

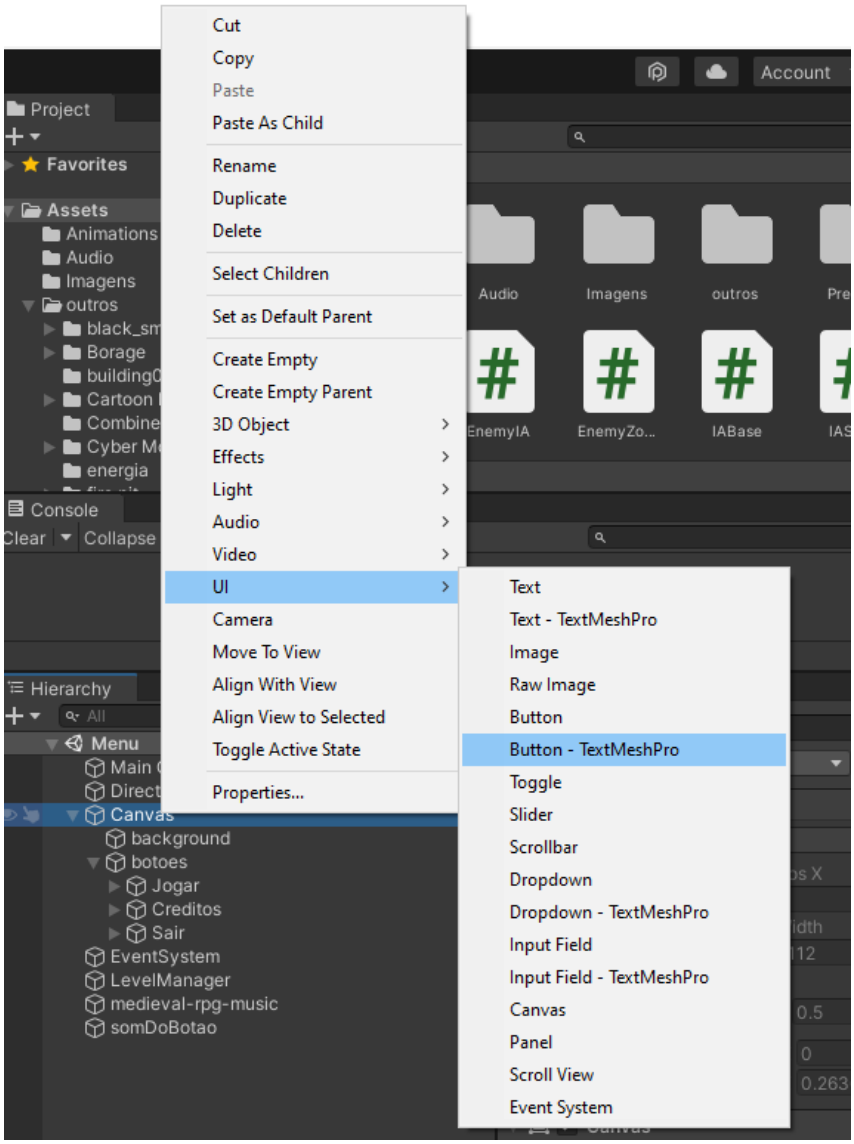

Figura 25: Criando um botão
Fonte: Elaborada pelo Autor

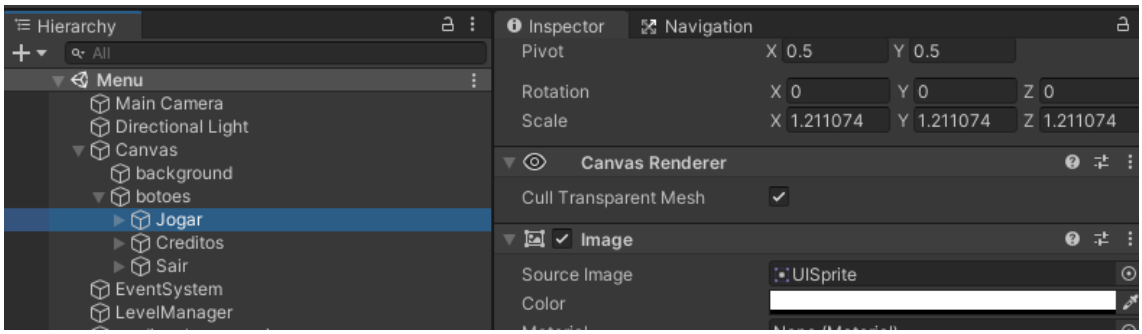

Figura 26: Colocar uma imagem no botão em source image
Fonte: Elaborada pelo Autor

A Figura 46 apresenta um script que faz a mudança de cena e finalizar o jogo no menu principal. Na linha 4, importa o SceneManagement para gerenciamento de cena. A classe LevelManager possui dois métodos o LoadScene e o sair. O método LoadScene carrega a cena pelo nome ou caminho da cena que será carregado. Dentro do método LoadScene possui uma variável pública chamada nomeDaCena, que foi criada na linha 8, que possibilita recebe o parâmetro name que será utilizado para outro método. Na linha 16 é criado um IEnumerator para uma corrotina para que possa aguardar um determinado tempo para abrir a cena, sendo necessário chamar essa corrotina no LoadScene linha 13. A execução da corrotina pode ser pausada em qualquer ponto utilizando a instrução yield quando utilizada a corrotina pausa a execução e retorna automaticamente na próxima cena. O método sair, linha 23, utiliza o comando Application.Quit faz com que a aplicação pare a execução do jogo.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.SceneManagement;
5
6   public class LevelManager : MonoBehaviour
7   {
8       public string nomeDaCena;
9
10      public void LoadScene(string name)
11      {
12          nomeDaCena = name;
13          StartCoroutine("Abrir");
14      }
15
16      private IEnumerator Abrir()
17      {
18          yield return new WaitForSeconds(0.5f);
19
20          SceneManager.LoadScene(nomeDaCena);
21      }
22
23      public void Sair()
24      {
25          Application.Quit();
26      }
27  }
28
```

Figura 27: Script da mudança de cena e finalizar o jogo
Fonte: Elaborada pelo Autor

Tendo os botões e script feitos, selecionar o botão Jogar na aba inspector em Button precisa configurar para o botão funcionar ao clicar nele. Em Runtime Only precisa colocar o script feito para os três botões, nós botões Jogar e Créditos precisar configurar o LevelManager

para LoadScene() e abaixo colocar qual cena será colocada no lugar da cena atual quando clicar no botão, Figura 47, e para o botão sair coloca o LevelManager para opção sair() terminando assim a configurar a tela de menu do jogo.

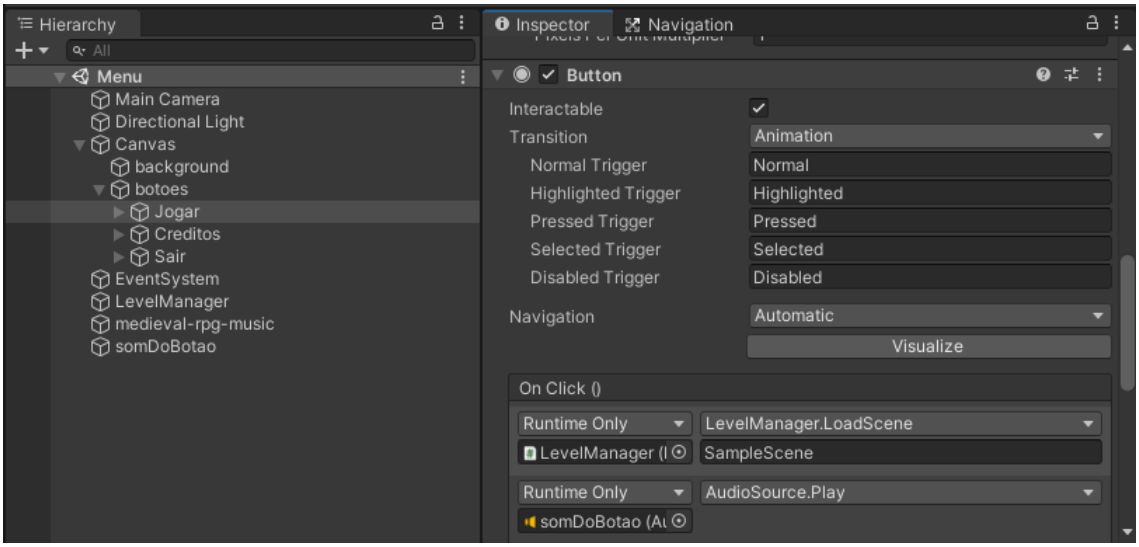

Figura 28: Configuração de tela do jogo
Fonte: Elaborada pelo Autor

3.2. Mudança de cenário

Nós jogos de vídeo game o jogador enfrenta desafios, soluciona problemas e chega ao final do cenário para ser teletransportado para o próximo cenário e começa tudo novamente. Para criar esse teletransporte são necessários dois scripts um para salvar a posição do personagem e o outro para fazer o carregamento da cena.

No primeiro script SalvarPosicao, Figura 48, possui três métodos Awake(), Start() e SalvarLocalizacao(). O método Awake() salva a cena atual na variável nomeCenaAtual. O método SalvarLocalizacao() salva o endereço de memória das posições x, y e z do personagem. No método Start() possui um if que verifica se os endereços de memória existem, ou seja se existe algum dado na variável, caso exista retornará para o personagem essa posição que já existe e já está salva.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;


public class SalvarPosicao : MonoBehaviour
{

    string nomeCenaAtual;
    float posX, posY, posZ;

    void Awake()
    {
        nomeCenaAtual = SceneManager.GetActiveScene().name;
    }

    void Start()
    {

        if (PlayerPrefs.HasKey(nomeCenaAtual + "X") &&
            PlayerPrefs.HasKey(nomeCenaAtual + "Y") &&
            PlayerPrefs.HasKey(nomeCenaAtual + "Z"))
        {
            transform.position = new Vector3(PlayerPrefs.GetFloat(nomeCenaAtual + "X"),
                                             PlayerPrefs.GetFloat(nomeCenaAtual + "Y"),
                                             PlayerPrefs.GetFloat(nomeCenaAtual + "Z"));
        }
    }

    public void SalvarLocalizacao()
    {
        PlayerPrefs.SetFloat(nomeCenaAtual + "X", transform.position.x);
        PlayerPrefs.SetFloat(nomeCenaAtual + "Y", transform.position.y);
        PlayerPrefs.SetFloat(nomeCenaAtual + "Z", transform.position.z);
    }
}

```

Figura 29: Script SalvarPosicao
Fonte: Elaborada pelo Autor

Segundo script CarregarCena, Figura 49, é responsável por carregar a cena do jogo. No método Update terá uma condição if que verifica se a variável podeInteragir é verdadeira e se o jogador clicou na tecla E do teclado para ativar a mudança de cena. Outro método é OnGUI() verificar se o jogador entrou no colisor do objeto, caso tenha entrado o jogador é informado com uma mensagem de texto para pressionar a tecla E.

O script também possui dois métodos OnTriggerEnter() e OnTriggerExit() que são ativados quando o personagem do jogo entra no colisor do objeto que possui o script CarregarCena e ativa o OnTriggerEnter() que atribui a variável booleana podeInteragir o valor verdadeiro e quando saí do colisor do objeto ativa o OnTriggerExit() que também possui a variável podeInteragir atribuindo o valor falso.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class CarregarCena : MonoBehaviour
{
    public GameObject Jogador;
    bool podeInteragir = false;
    public string CenaACarregar;

    void Update()
    {
        if (podeInteragir == true && Input.GetKeyDown(KeyCode.E))
        {
            //Jogador.GetComponent<SalvarPosicao>().SalvarLocalizacao();
            SceneManager.LoadScene(CenaACarregar);
        }
    }

    void OnTriggerEnter()
    {
        podeInteragir = true;
    }

    void OnTriggerExit()
    {
        podeInteragir = false;
    }

    void OnGUI()
    {
        if (podeInteragir == true)
        {
            GUIStyle stylez = new GUIStyle();
            stylez.alignment = TextAnchor.MiddleCenter;
            GUI.skin.label.fontSize = 20;
            GUI.Label(new Rect(Screen.width/2 - 50, Screen.height/2 + 50, 200, 30), "Pressione 'E'");
        }
    }
}

```

Figura 30: Script CarregarCena
Fonte: Elaborada pelo Autor

Com os scripts feitos é necessário colocar o script SalvaPosicao no inspector do personagem em todas as cenas e para carregar a próxima cena, seleciona objeto que receberá o script CarregarCena, no inspector desse objeto no componente Carregar Cena, Figura 50, coloca-se o personagem e a cena que será carregada.

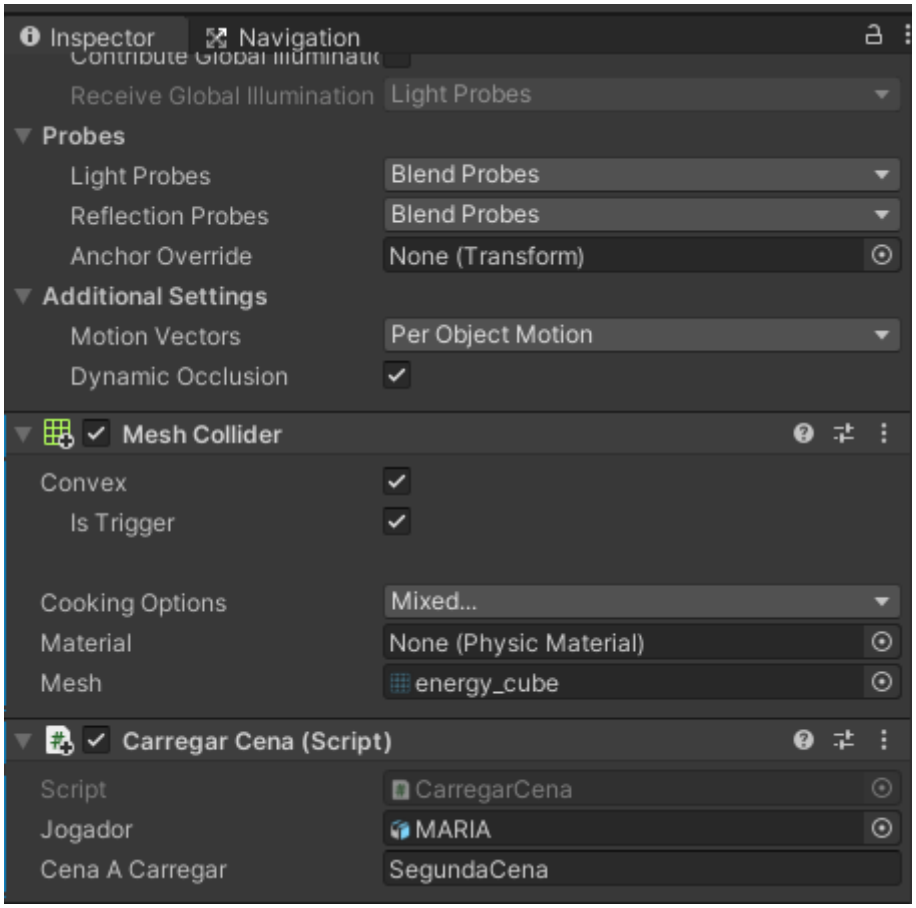

Figura 31: Componente carregar cena
Fonte: Elaborada pelo Autor

O resultado vemos na Figura 51, no lado esquerdo onde o personagem se aproxima do objeto que perde a confirmação para realizar o teletransporte e no lado direito o personagem aparecendo na posição inicial da cena.

Figura 32: Final da cena 1 a esquerda e o inicio da cena 2 a direita
Fonte: Elaborada pelo Autor

3.3. Assets e modelos importados

Para desenvolver o jogo foram importados vários assets, que são elementos do jogo como texturas, cenários, personagens, áudio ou scripts, gratuitos e também foram utilizados os que vieram por padrões na instalação do Unity. No desenvolvimento do jogo quando cria um novo projeto existe a alternativa que permiti escolher quais destes pacotes serão importados, porém podem ser acionados depois manualmente no projeto.

Foi utilizado o Assets padrão de terreno. Os Assets gratuitos foram árvores, pedras e também foram adquiridos alguns modelos de sites externos, como casa, fogueira, caixa e ponte.

3.4. Personagens

No desenvolvimento de jogos eletrônicos os personagens tanto heróis e vilões são fatores importantes na criação de jogos. A estrutura do personagem segue um modelo, no qual possui uma classe com N elementos relacionados a ele em forma de variáveis. Para o jogo optou-se por utilizar personagens prontos, no site https://www.mixamo.com/#/ que permite obter animações e também permite pegar alguns personagens 3D já animados para colocar no projeto.

O Mixamo é um repositório de Assets focado em personagens e animações 3D. Sendo necessário uma conta no site. Tendo a conta criada escolha um personagem na aba Characters, Figura 52.

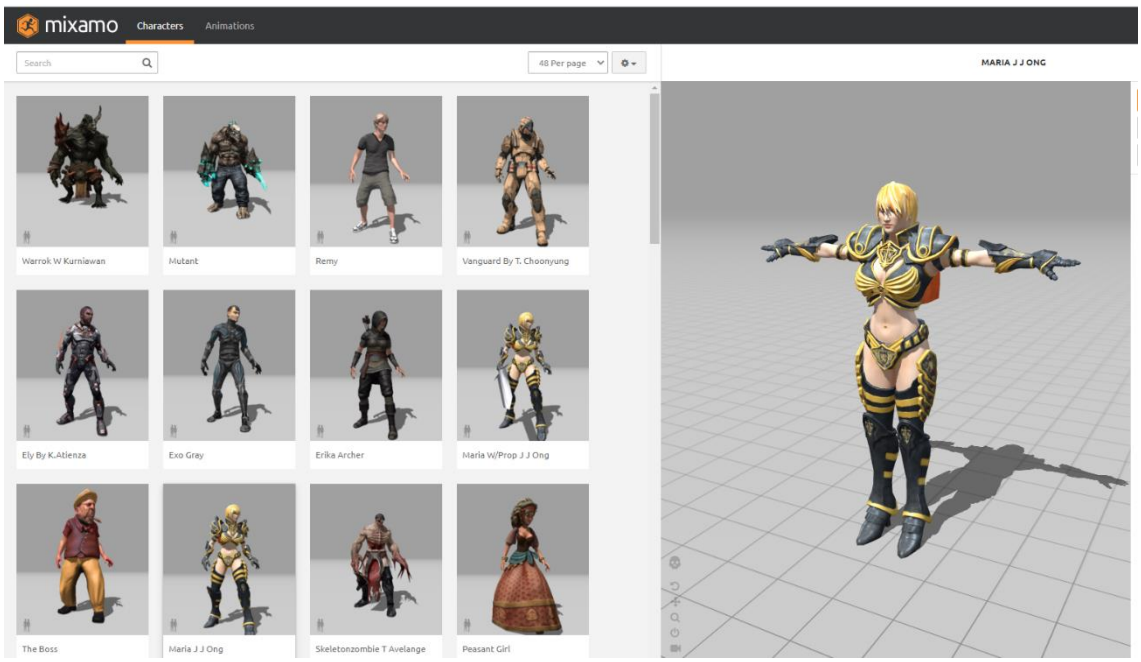

Figura 33: Selecionando personagem no Mixamo
Fonte: Elaborada pelo Autor

Para animações do personagem seleciona a aba Animations e escolha uma animação para o personagem, Figura 53.

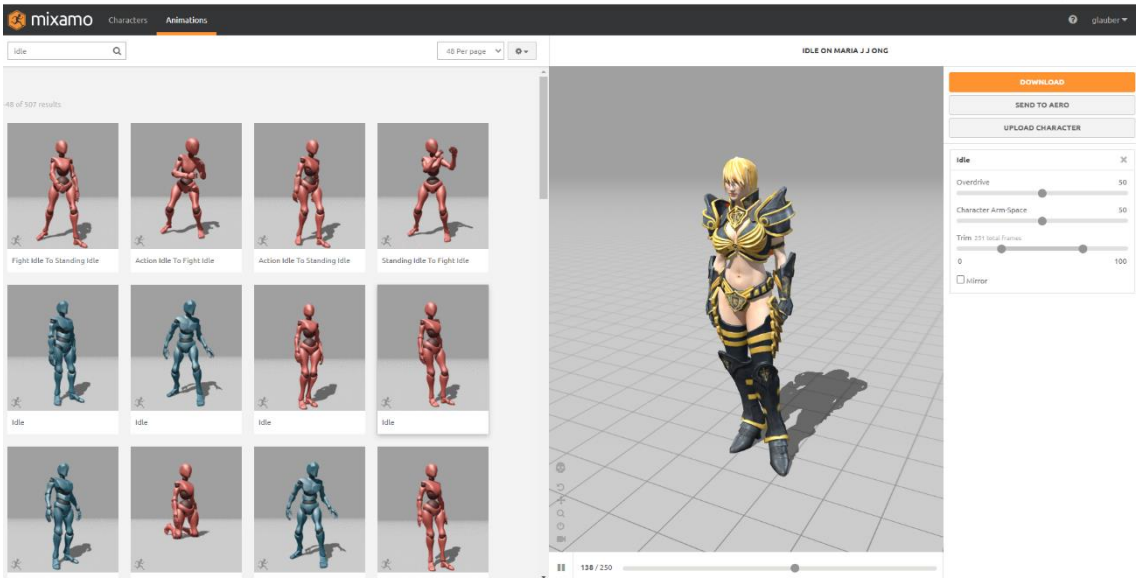

Figura 34: Selecionando animação do personagem no Mixamo
Fonte: Elaborada pelo Autor

Após selecionar a animação clique em Download e selecione FBX for Unity, Figura 54. Podendo baixar quantas animações forem necessárias para o personagem do jogo. Podendo fazer os mesmos procedimentos para os inimigos.

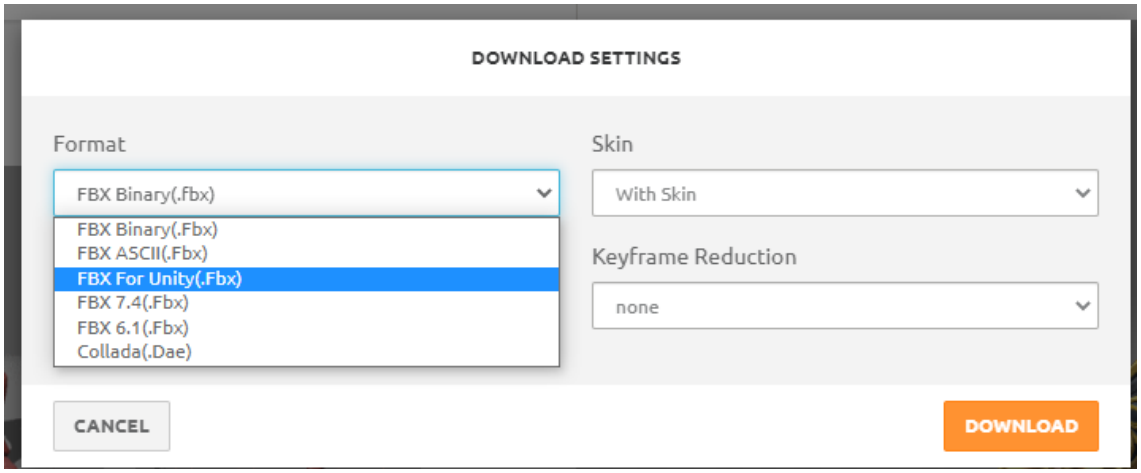

Figura 35: Download settings
Fonte: Elaborada pelo Autor

3.5. Controles do jogo

A jogabilidade dos games é realizada através da interação do jogador com o jogo e da forma como o personagem é controlado durante os desafios propostos.

Existem várias maneiras de realizar ou controlar a forma de como o personagem é controlado no jogo, mas quando se trata do computador o mouse e teclado são normalmente usados, no teclado o formato WASD é utilizado para a movimentação e N outros comandos utilizados pelos mais variados estilos.

3.6. Músicas e efeitos sonoros

Os áudios tanto de músicas e efeitos sonoros são importantes para desenvolvimento de jogos, pois ambos constituem a maior parte da emoção transmitida do jogo. As empresas de game preferem por essa razão utiliza esse recurso da forma que possa ser usado da melhor maneira possível.

Para este jogo foi criado uma pasta Audio dentro do diretório principal no visualizador Project, onde são arrastados todos os sons usados no jogo. O Unity suporta vários tipos de arquivos de sons e foram utilizados para esse jogo eletrônico os formatos mp3 e wav.

Os sons de música e efeitos sonoros foram obtidos através do site https://freesound.org/search/?q=music que é um site que possui uma base de dados de áudio colaborativa da Creative Commons que disponibiliza sons licenciados que é especificamente para propósitos de áudios que não possuem Copyright.

O capitulo 4 apresentou como desenvolver a tela do jogo, como realizar a transição de cenário no jogo, descreveu o que é assets, como conseguir personagens para uso em jogos, os comandos utilizados no jogo e como obter áudios para o game. Dessa forma o capitulo 4 ajuda a ter uma visão melhor de como desenvolver o game.

Conclusão

O desenvolvimento de jogos é uma tarefa multidisciplinar, sendo essencial possuir conhecimento em diversas áreas a fim de entender as etapas do desenvolvimento de jogos profissionais. Além de estudos teóricos, foi necessário entender o funcionamento da Unity 3D. A Unity 3D é muito complicada exigindo tempo e treinamento para entender como funciona a ferramenta.

A pesquisa realizada do Unity 3D foi relevante para desenvolvimento deste TCC, pois a facilidade de trabalhar com a ferramenta não necessita de conhecimentos específicos no campo do desenvolvimento de jogos, contanto que tenha conhecimento em programação. Tendo também uma diminuição no tempo de criação com a manipulação de todos os itens de forma simplificada e ágil.

A elaboração de um GDD utilizando também a UML foi indispensável para o desenvolvimento do jogo com essa documentação a etapa para o desenvolvimento tornou-se ágil e eficaz. Durante o desenvolvimento do jogo, essa documentação foi analisada, com o propósito de verificar se o mesmo seguia dentro do escopo descrito, tornando-se possível fazer uma analise dessa documentação e verificar se existem erros antes de iniciar o desenvolvimento, podendo corrigi-los antes de ocorrer.

Os objetivos desse trabalho têm o sentido de apresentar o processo de desenvolvimento de um jogo eletrônico, desde a criação do GDD até sua implementação. Dessa maneira, para conseguir chegar aos objetivos foi fundamental pesquisar sobre estrutura, processos e etapas que compreendem o conhecimento do campo de desenvolvimento de jogos.

Para desenvolver o jogo, a princípio foi criado o GDD baseado no gênero do jogo escolhido entre os que foram apresentados e a utilização da UML para ajudar o desenvolvedor a ter uma visão melhor na criação do jogo.

Durante o desenvolvimento a Unity 3D apresentou limitações por falta de conhecimento da ferramenta, pois houve momento que não teve como configurar determinada funcionalidade para que ocorrer se um evento de forma certa. Outro problema encontrado foi a limitação de hardware, pois não podia fazer nada muito detalhado no cenário que acabava causando lentidão na hora de rodar o jogo. Um problema na questão de visual do jogo foi os asserts gratuitos que não lembrava uma cena pós-apocalíptico e os asserts encontrados tinham tamanhos variados deixando o jogo muito grande em relação a armazenamento. Por esses motivos não foi possível concluir o jogo.

Apesar das dificuldades encontradas, o resultado atingido foi significativo. Não foi possível terminar o jogo para realização de testes com usuários para obter feedback sobre o jogo. Para trabalhos futuros, propõe-se aplicar o jogo para um grupo de usuários que verificaria a eficiência do jogo. A partir do estudo realizado com os usuários seria verificado se o jogo está funcionando corretamente, e dessa forma tentaria detectar possíveis ajustes para melhoria do jogo. Outro ponto importante é a exploração em detalhe de cada funcionalidade da Unity 3D para desenvolvimento de um jogo completo, podendo também fazer um aprofundamento da potencialidade que trás o uso da documentação para criação de um game. Para melhorar o visual do jogo poderia também trabalhar com ferramenta de modelagem como o Blender que é possível desenvolver os próprios asserts para evitar a falta de objetos e cenários com tamanho e a temática inadequada ao jogo. Para terminar o presente trabalho fornece um rico espaço para pesquisas e estudos sobre a área de desenvolvimento de jogos.

REFERÊNCIAS

ALVES, Gelderson Bezerra. **Estudo Comparativo entre Engines de Desenvolvimento de Jogos 2D.** Quixadá, 2021. Disponível em: < https://repositorio.ufc.br/bitstream/riufc/58827/1/2021_tcc_gbalves.pdf >. Acesso em: 03 Março 2022.

AZEVEDO, Eduardo et al. **Desenvolvimento de Jogos 3D e Aplicações em Realidade Virtual.** Rio de Janeiro: Elsevier, 2005.

BELARMINO, Francisco Edson dos Santos. **Utilização da Plataforma Unity3D como Processo de Aprendizagem a Programação.** Ceará, 2017. Disponível em: < https://prpi.ifce.edu.br/nl/_lib/file/doc1968-Trabalho/UTILIZA%C7%C3O%20DA%20PLATAFORMA%20UNITY3D%20COMO%20PROCESSO%20DE%20APRENDIZAGEM%20A%20PROGRAMA%C7%C3O.pdf>. Acesso em: 03 Março 2022.

BERTO, Gabriel Batistela; BEGOSSO, Luiz Carlos. **Análise e Desenvolvimento de Documentação para Jogos.** 2016. Disponível em: < https://cepein.femanet.com.br/BDigital/arqPics/1511320488P631.pdf>. Acesso em: 03 Março 2022.

CALL OF DUTY: ADVANCED WARFARE. In: Wikipedia, a enciclopédia livre. Flórida: Wikimedia Fundation, 2022. Il. Disponível em: <https://pt.wikipedia.org/wiki/Call_of_Duty:_Advanced_Warfare#/media/Ficheiro:Advanced_Warfare.jpg>. Acesso em: 16 Março 2022.

CARVALHO, Dennys. **Você Lembra de Todas as Aberturas do Sonic The Hedgehog nos Anos 90?.** 17 de outubro de 2017. Il. Disponível em: < https://www.torcedores.com/noticias/2017/10/voce-lembra-de-todas-as-aberturas-do-sonic-the-hedgehog>. Acesso em: 16 março 2022.

CARVALHO, Dennys. **Você Lembra de Todas as Aberturas de Super Mario nos Anos 80 e 90?.** 24 outubro 2017. Il. Disponível em: < https://www.torcedores.com/noticias/2017/10/voce-lembra-de-todas-as-aberturas-de-super-mario-world-nos-anos-80-90 >. Acesso em: 16 março 2022.

CHANDLER, Heather Maxwell. **Manual de Produção de Jogos Digitais.** 2ª Ed. Porto Alegre: Bookman, 2012.

ETTO, Mateus Gonçalez. **Utilização de Inteligência Artificial em Jogo RPG.** Bauru/SP, 2016. Disponível em: <https://mostra-de-tccs-bcc.github.io/TCC-BCC-Bauru-2016/etto/thesis-etto.pdf>. Acesso em: 3 Março 2022.

FERREIRA, Luis Henrique. **Desenvolvimento de Jogos Eletrônicos Utilizando a Tecnologia Unity.** Araraquara, 2015. Disponível em: <https://document.onl/documents/tcc-desenvolvimento-de-jogos-eletronicos-utilizando-a-tecnologia-unity-.html> Acesso em: 3 Março 2022.

GULARTE, Daniel. **Spacewar! e seu livre legado.** 24 Julho 2018. Il. Disponível em: < https://bojoga.com.br/retroplay/colunas/dossie-retro/spacewar-e-seu-livre-legado/>. Acesso em: 15 Março 2022.

GULARTE, Daniel. **Pac-Man (Namco, 1980).** 20 Junho 2018. Il. Disponível em: < https://bojoga.com.br/retroplay/analises-de-jogos/arcade-pinball/pac-man-namco-1980/>. Acesso em: 19 Março 2022.

HENRIQUE, Arthur. **'Spacewar!' Original, de 1962, é Restaurado e Pode ser Jogado em um PDP-11.** 17 Maio 2021. Il. Disponível em: < https://olhardigital.com.br/2021/05/17/games-e-consoles/spacewar-original-de-1962-e-restaurado-e-pode-ser-jogado-em-um-pdp-11/>. Acesso em: 15 Março 2022.

JOHNSON, Ben. **Conserving Tennis For Two.** 24 Novembro 2015. Il. Disponível em: < https://www.gamedeveloper.com/design/conserving-tennis-for-two>. Acesso em: 15 Março 2022.

KEULEN, Fillipe Martins Van. et al. **GAME ENGINE UNITY 3D.** 2013. Disponível em: < http://re.granbery.edu.br/artigos/NDkx.pdf>. Acesso em: 03 Março 2022.

LIMA, Alessandro Peixoto. **Design de Personagens para Games Next-Gen: Volume 1.** Rio de Janeiro: Ciência Moderna, 2011.

MARCELO, Antonio; PESCUITE, Julio. **Design de jogos: Fundamentos**. Rio de Janeiro: Brasport, 2009.

MASTROCOL, Vicente Martin. **Um mundo de possibilidades**. Revista Marketing: 2007. Disponível em: < http://www.vincevader.net/games/artigo08_mmorpg.pdf>. Acesso em: 9 Março 2022.

MERCURY. **William Higinbotham... "Tennis for Two".** 25 Outubro 2011. Il. Disponível em: <http://philosophyofscienceportal.blogspot.com/2011/10/william-higinbothamtennis-for-two.html>. Acesso em: 15 Março 2022.

OLIVEIRA, Daniel. **Os Diagramas Comportamentais da UML.** 2016. Disponível em: < https://micreiros.com/os-diagramas-comportamentais-da-uml/ >. Acesso em: 21 abril 2022.

OLIVEIRA, Jhonattan Amorim; TORRES FILHO, Rogério Rivera. **Processo de Desenvolvimento de um Jogo Digital para Dispositivos Móveis.** Palhoça, 2014. Disponível em: < https://repositorio.animaeducacao.com.br/bitstream/ANIMA/11131/1/109715_Rogerio_Jhonattan.pdf>. Acesso em: 17 Abril 2022.

PERUCIA, Alexandre Souza; et al. **Desenvolvimento de Jogos Eletrônicos: Teoria e Prática.** São Paulo: Novatec, 2007.

PINTEREST. **October 1958: Physicist Invents First Video Game.** Il. Disponível em: < https://br.pinterest.com/pin/616641373952835199/>. Acesso em: 15 Março 2022.

RAMOS, Henrique Moraes. **A história dos jogos de computadores.** 2007. Disponível em: < http://www-usr.inf.ufsm.br/~hramos/elc1020/artigo_hist_jogos.pdf>. Acesso em: 9 Março 2022.

RAMOS, Ricardo Argenton. **Treinamento Prático em UML.** São Paulo: Digerati, 2006.

SANTAELLA, Lucia; FEITOSA, Mirna. **Mapa do jogo: A diversidade cultural dos games.** São Paulo: Cengage Learning, 2009.

SANTOS, Willian; ALVES, Lynn. **A Aplicação da Linguagem de Modelagem Unificada (U.M.L.): Novas Perspectivas para o Desenvolvimento de Games Educacionais.** 2015. Disponível em: < http://www.comunidadesvirtuais.pro.br/seminario-jogos/files/mod_seminary_submission/trabalho_261/trabalho.pdf >. Acesso em: 01 Abril 2022.

SCHUYTEMA, Paul. **Design de Games: uma Abordagem Prática.** São Paulo: Cengage Learning, 2008.

SOMMERVILLE, Ian. **Engenharia de Sofware.** 9º ed. São Paulo: Pearson, 2011.

TORETTI JUNIOR, Nelson. **Desenvolvimento de Jogos Digitais 2D com Unity 3D.** Americana/SP, 2013. Disponível em: < http://ric.cps.sp.gov.br/bitstream/123456789/1258/1/20132S_TORETTIJUNIORNelson_TCCPD1222.pdf >. Acesso em: 2 Março 2022.

TORRES, Ricella Delunardo. **Desenvolvendo um Jogo Para Ensinar Física com Unity 3D.** João Monlevade/MG, 2015. Disponível em:

<https://www.monografias.ufop.br/bitstream/35400000/255/1/MONOGRAFIA_DesenvolvendoJogoEnsinar.pdf>. Acesso em: 2 Março 2022.

UNITY. **Scene View Navigation.** 2016. Disponível em: < https://docs.huihoo.com/unity/5.6/Documentation/Manual/SceneViewNavigation.html >. Acesso em: 25 Abril 2022.

ANEXO A – GAME DESIGN DOCUMENT

I. Visão geral essencial

A visão geral essencial mostra uma breve visão completa, porém detalhada do game. O objetivo é fazer com que a equipe de desenvolvimento acostuma-se de forma rápida a idéia do jogo. Segundo Oliveira e Torres Filho (2014) o objetivo é apresentar e familiarizar o jogo a qualquer pessoa, uma visão breve, mas detalhada do jogo a ser criado.

a. Resumo

O resumo tem como finalidade passar uma idéia geral. Conforme Schuytema (2008) o resumo é uma sinopse de todo o conhecimento do jogo.

b. Aspectos fundamentais

Os aspectos fundamentais para Schuytema (2008) “tem a intenção de captar a essência do game, porém com foco no gameplay e no que o jogador está fazendo”. A finalidade dos aspectos fundamentais é extrair elementos fundamentais do jogo que determinará uma trama central para o divertimento e experiência do jogador (SCHUYTEMA, 2008).

c. Golden nugget

O golden nugget conforme Schuytema (2008) é basicamente uma lista de componentes do jogo que destacar-se da concorrência.

II. Contexto do game

O contexto do game de acordo com Schuytema (2008) é a descrição do mundo que envolve o jogo. Ainda de acordo com Schuytema (2008) qualquer jogo encontrar-se dentro de um contexto, que seja familiar de alguma forma para o jogador, que é indispensável para se compreender o que acontece no jogo. Segundo Oliveira e Torres Filho (2014) nessa seção descrever o mundo que rodeia o jogo, no qual o jogador detém toda a ação, história do jogo.

a. História do game

A história do game é feita por um design devendo explica e torna compreensível a história do jogo, do começo ao fim. Conforme Schuytema (2008) essa seção inclui spoilers da jogabilidade do jogo, porém é fundamental para a história de qualquer jogo, a partir da visão do jogador, um documento claro e conciso. De acordo com Torres (2015) ao compor uma história de um jogo é necessário determinar quais são as metas a serem alcançadas, sendo necessário criar um roteiro que serve como guiar no desevolvimento do jogo.

b. Eventos anteriores

Essa parte da documentação exibir acontecimentos do passado no contexto da história do jogo. Segundo Schuytema (2008) essa seção descreve o contexto da história do jogo, como um todo, dentro do universo do jogo. Para Oliveira e Torres Filho (2014) os eventos anteriores contar como o herói conseguiu seus poderes para enfrentar os desafios do jogo.

c. Principais jogadores

Nessa parte do documento apresenta e explica os personagens relevantes para o jogo, no qual os personagens são os elementos chave, do herói favorito do jogador até seu maior inimigo. Conforme Schuytema (2008) ao detalhar um personagem, podendo exibir a bibliografia dele, os poderes especiais, razões entre outras coisas que definirá o comportamento e que papel realizará na história do jogo.

III. Objetos essenciais do game

Os objetos essenciais do game são os vários objetos importantes que surgem no jogo, são aqueles que influenciam na experiência do jogo (SCHUYTEMA, 2008). De acordo com Oliveira e Torres Filho (2014) essa seção descreve os objetos que possuem um papel no jogo, afetando a jogabilidade e a experiência do jogo.

a. Personagens

Os personagens são todos aqueles, controláveis ou não, que possuem interação com o jogador no jogo e que desempenham um papel na história do jogo. Conforme Torres (2015) os jogos possuem peersonagens jogáveis, conhecidos como Player Characters, como herói, os personagens secundários ou mesmo um objeto sendo um carro ou um robô; ou não jogáveis, conhecidos como Non Player Characters ou NPC, que são os vilões do jogo ou personagens auxiliar que ajudar durante jogo. Segundo Schuytema (2008) essa parte da documentação explica quem são os personagens do jogo, dos personagens mais importantes envolvidos na história do jogo aos NPC, personagens não controlados por jogadores. Para Torres (2015) cada personagem realiza um papel no contexto do jogo, o protagonista e seus companheiros, os inimigos e chefes, todos devem ter particularidades que despertar emoções.

b. Armas

No documento segundo Schuytema (2008) definir armas como quaisquer armas ou habilidades especiais que exercem um papel fundamental no jogo, ou seja, pode ser utilizadas pelo jogador ou outra entidade do jogo.

c. Estruturas

Oliveira e Torres Filho (2014) descrevem estruturas como checkpoints que são locais onde o personagem renasce após morrer; cristais, estruturas principais da história, que proporcionam os eventos referentes à história do jogo; e Vilas locais onde encotram NPCs, dão informações sobre a história do jogo, comprar equipamentos dos personagens e vendem equipamentos. De acordo com Schuytema (2008) essa seção descreve quaisquer estruturas singulares e significativas para a jogabilidade do jogo.

d. Objetos

Nessa parte da documentação será descrito os objetos relevantes que não foram descritos ou não coincidem nas outras partes do documento. Segundo Schuytema (2008) os objetos descritos na seção são itens de inventário e powerups, item qualquer que aumenta poder, velocidade, habilidade ou outra particularidade de algum personagem, porém existir mais como

componentes para solução de um enigma.

IV. Conflitos e soluções

São caracteristicas relacionadas às iterações entre entidades do jogo. Conforme Schuytema (2008) conflitos e soluções do jogo geralmente têm muitos detalhes se for um combate precisa descreve o que acontece durante a batalha e a forma como será solucionada, ou seja, uma ação é feita pelo jogador e o que ocorre em seguida.

V. Inteligência artificial

A inteligência artificial ou IA controla e guia os NPCs para possibilitar desafios e disponibilizar ajuda ao jogador. De acordo com Torres (2015) a IA dará vidas ao jogo possibilitando desafios, proporcionado ajuda ou obstáculos, aos jogadores à medida que progridem. Conforme Oliveira e Torres Filho (2014) a IA usada em jogos, utilizar técnicas com objetivo de coordenar os inimigos do jogo e seus comportamentos. Segundo Schuytema (2008) “Nesta seção do documento, resuma qualquer comportamento que define as ações dos oponentes computadorizados e quaisquer informações (por parte da IA) que afetarão esses comportamentos”.

VI. Fluxo do game

O fluxo do game referência à localização ou, uma estrutura que mostrar os desafios do jogo conforme o jogador segue em direção ao objetivo. Para Oliveira e Torres Filho (2014) o fluxo do jogo inicia na escolha do personagem, à medida que avança nos niveis durante a diração dos três cristais a serem destruidos. De acordo com Schuytema (2008) fluxo do game aborda cada área do jogo de forma individual, seja um nível ou uma missão.

Uma regra prática para essa documentação é apenas inclui o que foi inicialmente exibido em outras partes da documentação do jogo, pois essa parte do documento determinar apenas o fluxo e a disposição dos objetos determinados dentro dos ambientes do jogo. Para Schuytema (2008) essa parte do documento fixar a localizações dos vários enigmas e aventuras do jogo, é como um guia para a equipe de desenvolvimento possa ler a explicação de um enigma e onde se localizar no jogo, podendo ter uma idéia clara e concisa do por que da implementação.

VII. Controles

Os controles segundo Schuytema (2008) contêm as ações e os controles dos jogadores. Descrevendo nessa parte do documento qualquer alteração no controle em relação a plataforma, seja qual for a abordagem ao mapeamento dos controles. Conforme Torres (2015) controles é a forma na qual o jogador interage com ambiente do jogo, tendo um mapeamento personalizado das teclas do teclado ou botões do joystick permitindo ações no jogo por meio deles, sendo bem feitos para que não cause problemas durante o jogo.

VIII. Definições

As definições são utilizadas quando estiver criando ou usando algum glossário, de termos relavantes que não estejam claros, e referências para ajudar na compreensão do diálogo encontrado no documento para o time de desenvolvimento (SCHUYTEMA, 2008).

IX. Referências

As referências incluem informações sobre qualquer conteúdo de referência que seja importante e que ajude a compreender o clima e a idéia do jogo (SCHUYTEMA, 2008). De acordo com Schuytema (2008) essa parte do documento listar filmes que inspirar ou livros que inspirem a equipe de desenvolvimento.