CARLOS EDUARDO OLIVEIRA SANTOS
JOSIEL BRAUN RODRIGUES

BARBARIAN: Desenvolvimento de um jogo digital com Unreal Engine 4

Ji-Paraná
2020

CARLOS EDUARDO OLIVEIRA SANTOS
JOSIEL BRAUN RODRIGUES

BARBARIAN: Desenvolvimento de um jogo digital com Unreal Engine 4

Monografia apresentada à Banca Examinadora do Centro Universitário São Lucas, como requisito de aprovação para obtenção do Título de Bacharel em Sistemas de informação.
.....
Orientador: Maigon Nacib Pontuschka

Ji-Paraná
2020

Dados Internacionais de Catalogação na Publicação - CIP

S237b Santos, Carlos Eduardo Oliveira.

Barbarian: desenvolvimento de um jogo digital com Unreal Engine 4. / Carlos Eduardo Oliveira Santos; Josiel Braun Rodrigues. – Ji-Paraná, 2020.
57 p., il.

Monografia (Sistema de Informação) – Centro Universitário São Lucas, Ji-Paraná, 2020.

Orientador: Prof. Me. Maigon Nacib Pontuschka

1. Jogo eletrônico - desenvolvimento. 2. Unreal Engine 4. 3. Game Design Document (GDD). 4. Software. I. Rodrigues, Josiel Braun. II. Pontuschka, Maigon Nacib. III. Título.

CDU 004.4

Ficha Catalográfica Elaborada pelo Bibliotecário Giordani Nunes da Silva CRB 11/1125

ATA DE TRABALHO DE CONCLUSÃO DE CURSO

ATA Nº 09/2020 DE TRABALHO DE CONCLUSÃO DE CURSO

No décimo primeiro dia do mês de Dezembro de 2020, no horário das 18h às reuniram-se o(a) Orientador(a) professor(a) Prof. Me. Maigon Nacib Pontuchska e os(as) professores(as) Prof. Me. Ana Flavia Moreira Camargo e Prof. Me. Thyago Bohrer Borges para comporem Banca Examinadora de Trabalho de Conclusão de Curso, sob a presidência do(a) primeiro(a), para analisarem a apresentação do trabalho "**Barbarian: Desenvolvimento de um Jogo Digital com UnReal Engine 4**". Após arguições e apreciação sobre o trabalho exposto foi atribuída à menção como nota do Trabalho de Conclusão de Curso dos(a) acadêmicos(a): **CARLOS EDUARDO OLIVEIRA SANTOS e JOSIEL BRAUN RODRIGUES**.

Obs: Trabalho de Conclusão de Curso (X) aprovado ou ( )reprovado com nota total de 9,1 (nove virgula um) pontos, sendo atribuídos o valor 8,9 (oito virgula nove) pontos ao trabalho escrito e 9,3 (nove virgula três) pontos à apresentação oral.

Carlos Eduardo O. Santos — Josiel Braun Rodrigues

CARLOS EDUARDO OLIVEIRA SANTOS e JOSIEL BRAUN RODRIGUES

Prof. Me. Thyago Bohrer Borges

Prof. Me. Maigon Nacib Pontchuska
Orientador

Ana Flávia Moreira Camargo

Prof. Me. Ana Flavia Moreira Camargo

Prof. Me. Thyago Bohrer Borges
Coord. Sistemas de Informação

São Lucas Educacional Ji-Paraná
Av. Eng. Manfredo Barata Almeida da Fonseca, 542
Jd. Aurélio Bernardi | Ji-Paraná | RO | CEP 76907-438

RESUMO

Nas últimas décadas, a indústria de jogos digitais tem influenciado as pessoas a buscar conhecimento nessa área, visto que a cada ano que se passa um número maior de jogos é lançado, tanto por grandes empresas como também por desenvolvedores independentes. Para um desenvolvedor iniciante nesse mercado, pode ser um pouco confuso, saber por onde começar e como diferenciar o desenvolvimento de um jogo para o de um software comum. Nesse contexto, esse trabalho traz de forma dinâmica e intuitiva, o processo de desenvolvimento de um jogo para a plataforma Windows, por meio da Unreal Engine 4. Foi abordado todos os processos de criação, desde a elaboração do *Game Design Document* (GDD), a escolha dos personagens, ambientação do cenário, inteligência artificial e fase de testes. Para auxiliar e agilizar o processo de desenvolvimento foi utilizada a metodologia ágil por meio do framework Scrum o que permitiu que o projeto fosse concluído dentro do prazo estabelecido.

Palavras-Chave: Jogo. Game Design Document. Unreal Engine. Processo de Desenvolvimento.

ABSTRACT

In the last few decades, the digital games industry has influenced people to seek knowledge in this area, since with each passing year a greater number of games is released, both by large companies as well as by independent developers. For a novice developer in this market, it can be a little confusing, knowing where to start and how to differentiate the development of a game from that of common software. In this context, this work brings in a dynamic and intuitive way, the process of developing a game for the Windows platform, through Unreal Engine 4. All creation processes were covered, since the elaboration of the Game Design Document (GDD), the choice of characters, ambiance of the scenario, artificial intelligence and testing phase. To assist and streamline the development process, the agile methodology was used through the Scrum framework, which allowed the project to be completed within the established deadline.

Keywords: Game. Game Design Document. Unreal Engine. Development Process.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Um jogo tem como principal característica e função a diversão entre uma ou mais pessoas, assim como traduzido do termo de origem "*Jocus*" do latim divertimento. O ato de jogar pode ser considerado um fenômeno cultural, pois é praticado desde os primórdios através das mais variadas espécies (HUIZINGA, 1980).

O jogo é fato mais antigo que a cultura, pois esta, mesmo em suas definições menos rigorosas, pressupõe sempre a sociedade humana; mas, os animais não esperaram que os homens os iniciassem na atividade lúdica. Os animais brincam tal como os homens. Bastará que observemos os cachorrinhos para constatar que, em suas alegres evoluções, encontram-se presentes todos os elementos essenciais do jogo humano. Convidam-se uns aos outros para brincar mediante um certo ritual de atitudes e gestos. Desde já encontramos aqui um aspecto muito importante: mesmo em suas formas mais simples, ao nível animal, o jogo é mais do que um fenômeno fisiológico ou um reflexo psicológico (HUIZINGA, 1980).

Não podemos falar de jogos digitais sem citar os tão amados videogames. Segundo Eucidio (2014) o termo videogame historicamente esteve limitado aos jogos de console e as máquinas de fliperama, máquinas robustas que ao terem fichas introduzidas nos levavam a um mundo de diversão com jogos que empolgavam através de cliques e manuseios em seus botões.

Já as tecnologias digitais são baseadas na microinformática, o que engloba jogos para computadores, consoles, fliperamas, smartphones, tablets e qualquer outro equipamento que venha a existir (EUCIDIO, 2014).

1.1 PROBLEMATIZAÇÃO

Para a elaboração de um jogo que funcione corretamente é necessário além da criatividade do desenvolvedor, uma precisa execução dos processos de desenvolvimento. A performance durante o jogo é um ponto muito importante, pois com a falta da mesma um jogo deixa de ser divertido e se torna frustrante, uma experiência negativa.

Ao longo desse projeto vamos abordar os principais conceitos que envolvem o desenvolvimento de um jogo, além de elucidar a forma como uma Game Engine trabalha e aplicar esse conhecimento na criação de um jogo digital para desktop.

1.2 OBJETIVO GERAL

O objetivo deste projeto é gerar conhecimento sobre a utilização da Unreal Engine e sua lógica de programação no desenvolvimento de jogo, analisando o produto final de acordo com as expectativas previstas, junto com a utilização de cada uma das técnicas do processo de produção.

1.2.1 Objetivos Específicos

Como objetivos específicos para elaboração deste projeto, temos:

- Obter referências bibliográficas para elaboração de jogos digitais;
- Criação de um *Game Design Document* (GDD)
- Implementação de um protótipo do jogo;
- Testes e avaliação;

2 REFERENCIAL TEÓRICO

2.1 ORIGEM DOS JOGOS

Os primeiros jogos são datados da década de 50 onde seus principais objetivos eram o ambiente militar ou militarização como podemos dizer. As principais bases de dados existentes nomeiam o físico William Higinbotham o desenvolvedor do primeiro protótipo de jogo digital enquanto participava do projeto Manhattan, que nasceu de um esforço para desenvolver a bomba atômica nos Estados Unidos na década de 1940 e culminou com a detonação das bombas em Hiroshima e Nagasaki, no Japão, em 1945.

Eucidio (2014) explica que, o jogo se chamava *TennisProgramming*, ou *Tennis for Two*, exposto em uma tela de 15 polegadas e projetado para ser processado em um computador analógico.

2.2 GÊNEROS DE JOGOS

Os jogos são classificados de acordo com os gêneros. Segundo Mallmann (2012 apud BATES, 2004) esses gêneros são classificados segundo a ideia e o tema proposto pelo game, isto é, a ambientação, os componentes e a finalidade do jogo.

Existe uma gama bem ampla quando se trata de jogos eletrônicos e seus gêneros e plataformas. Desde estilos focados em uma específica função como a simulação de um ambiente para objetivos mais profissionais indiferentes a diversão, e outros já focados na diversão individual ou de um grupo. Villas Bôas (2005, p.28-30), possui uma análise interessante a respeito de gênero de jogos e suas dinâmicas, o Quadro 1 a seguir mostra as considerações do autor.

Quadro 1 – Os Gêneros e Características dos Games.

GÊNERO	DESCRIÇÃO	MECÂNICA	QUALIDADES	EXEMPLOS
Aventura	Jogos baseados em histórias, geralmente voltados em solucionar enigmas para seguir com seu curso.	Nem sempre são em tempo real (híbrido c/ ação), usam mais o cérebro e menos a destreza e os reflexos.	Um mundo grande e complexo para se explorar, com Pcs interessantes e uma boa história.	Monkey Island
Ação	Jogos em tempo real, nos quais o jogador deve responder com velocidade ao que está ocorrendo na tela.	Dominado pelos FPS (First Person Shooter), menos intelectual do que os jogos de puzzles, estratégia, adventure e outros.	Explosões de adrenalina e ação que exigem rápidas escolhas e bons reflexos.	Half-Life
RPG	Geralmente o jogador dirige um grupo de personagens em alguma missão, em diversas tramas e cenários.	Grande universo do jogo com história não-linear. Sistemas de evolução dos poderes e forças dos personagens.	Microgerenciamento dos personagens: escolha de equipamentos e armas; sistema de magias e complexo.	Última
Simuladores	Jogos que simulam condições do mundo real, principalmente operação de máquinas complexas,	Quanto mais "sério" o simulador, mais próximo à realidade espera-se que seja. Jogos estilo "arcade sims" são menos realistas, com	O realismo e a precisão dos controles das máquinas e de sua operação são os principais fatores.	Flight Simulator 2002

	como aviões e carros.	controles mais simplificados.		
Esportes	Jogos que representam os esportes "reais" coletivos ou individuais.	Podem tanto simular o esporte pelo lado do atleta praticando o esporte, ou pelo lado do técnico gerenciando sua equipe.	Espera-se a completa reprodução das regras e das principais peculiaridades de cada esporte.	Fifa 2002
Luta	Jogos para dois jogadores onde cada um controla um personagem que usa uma combinação de movimentos e manobras para ataque e defesa contra o oponente.	Jogos geralmente de perspectiva lateral e de curta duração, com ampla variedade de personagens e manobras	Têm um conjunto básico de ataques, defesas e contra ataques de rápida aprendizagem, e um grupo de manobras e combinações mais complexas que exigem mais prática	Tekken 3
Casuais	Adaptações dos jogos tradicionais como xadrez, gamão e paciência. Inclui também jogos dos shows de Tv.	Jogos de Interface simples, geralmente com uma baixa curva de aprendizagem.	Jogadores esperam que as regras sejam exatamente as mesmas dos jogos em suas versões "reais".	Chess / Show do Milhão
“God Games”	Também chamados de "software toys", jogos que não possuem um real objetivo além do passatempo.	Geralmente não há critérios de vitória e de derrota, ou de erros e acertos.	Com um brinquedo simples, espera-se apenas e simplesmente que seja "divertido".	The Sims
Educacionais	Jogos cujo objetivo é ensinar enquanto se diverte jogando.	Geralmente voltados para o público infantil, utilizam uma estrutura semelhante à de desenhos animados.	O conteúdo deve ser muito bem elaborado em conjunto com especialistas, para que realmente atinja o objetivo de ensinar brincando.	Coelho Sabido
Puzzle	Jogos puramente voltados para o desafio intelectual na	Esse gênero é de jogos de enigmas e problemas	Problemas de lógica, de matemática ou mesmo enigmas	The Incredible Machine

	solução de problemas.	propriamente ditos, sem nenhum contexto de história, cenário ou outro objetivo além da solução dos problemas.	filosóficos são apreciados.	
Online/ Massive Multiplayer	Jogos que podem ser qualquer gênero anterior, com a diferença de ser jogado na internet.	Comunidades inteiras estão surgindo em torno desses jogos, que são desenvolvidos com o objetivo de favorecer o surgimento dessas comunidades.	Um gênero novo e com diversas características de jogabilidade ainda em fase de transformação.	Everquest

Fonte: Villas Bôas (2005, p.28-30) – adaptado pelos autores.

2.3 GAMES ENGINES

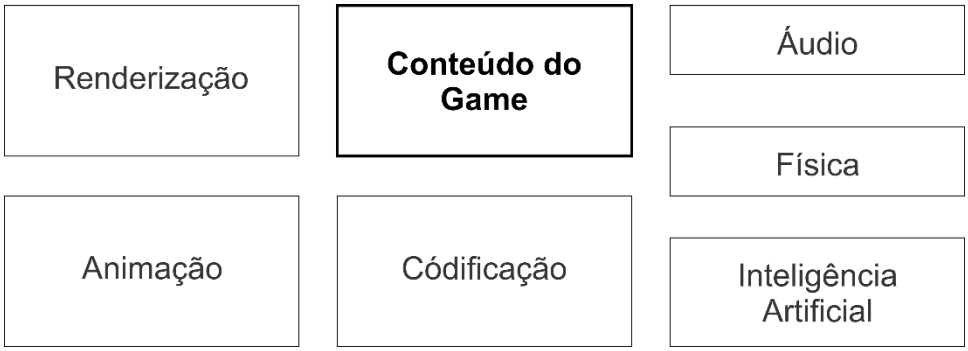
As *Game Engines* ou motores de jogos são softwares ou bibliotecas próprias para o desenvolvimento de jogos, que tem como o objetivo facilitar a dinâmica de desenvolvimento substituindo ou diminuindo a criação de códigos, assim facilitando o processo de criação.

São ferramentas muitas vezes complexas e polivalentes para a criação de jogos e conteúdo multimídia. Oferecem um ambiente para um desenvolvimento eficiente, às vezes, mesmo sem o conhecimento de programação. Os motores de jogos abrangem muitas áreas diferentes do processo de desenvolvimento de jogos, como renderização, física, áudio, animação, inteligência artificial e a criação da interface do usuário (ŠMÍD, 2017).

2.3.1 Componentes

Os motores de jogos contemporâneos se baseiam na arquitetura dos componentes. Como podemos observar na Figura 1, eles divididos em vários componentes, cada um fornecendo uma funcionalidade especial (EBERLY, 2005).

Figura 1 – Análise dos componentes das *Game Engines*.



Fonte: ŠMÍD, 2017

2.3.2 Renderização

Quando queremos renderizar objetos 3D na tela, o motor inicia a API *(Application Programming Interface)* gráfica, como *OpenGL* ou *DirectX*. Essa interface nos permite controlar a placa gráfica fazendo com que ela mostre nossos objetos. Essas APIs são de baixo nível e para o desenvolvimento simples de jogos, seria inviável acessá-los diretamente porque seria necessário mais tempo gasto em programação de baixo nível (ŠMÍD, 2017).

2.3.3 Animação

Nem todo objeto da cena é estático, eles podem ser movidos pela física ou animados. Praticamente em todos os jogos animações são necessárias. Considere a animação uma alteração predefinida em determinados parâmetros ao longo do tempo, sendo possível a animação de posição onde personagens inteiros podem ser animados com esqueletos ou os vértices formando poses e animações dinâmicas (ŠMÍD, 2017).

2.3.4 Física

A física é uma parte crucial da jogabilidade. A maioria dos jogos tem como objetivo uma simulação do mundo real onde os objetos devem agir de maneira fisicamente correta. O motor de jogo geralmente fornece uma solução para essas simulações físicas. Cada objeto possui um colisor (limite simplificado do objeto usado para a física) e algum material físico que determina a massa e as propriedades da superfície. O mecanismo calcula vetores de movimento e colisões entre os objetos. Podemos distinguir entre objetos estáticos que não se movem como o solo e corpos

rígidos que têm uma massa, um material com atrito, que reage a forças, cai com a gravidade etc. Além disso, o motor pode simular corpos moles, que mudam de forma de acordo com as forças externas. Embora a simulação física não precise ser extremamente precisa, é uma parte importante do desenvolvimento de um jogo (ŠMÍD, 2017).

2.3.5 Inteligência Artificial

Além do personagem controlado pelo usuário, a existência de inimigos para interação é crucial. Esses personagens não jogadores, precisam de uma certa quantidade de inteligência para comportar-se razoavelmente para que o jogador não fique entediado com um sistema de jogo linear.

A principal tarefa da inteligência artificial é encontrar um caminho do ponto A ao ponto B conectando o personagem a esse caminho involuntariamente. Normalmente, o motor gera uma superfície de navegação, que se baseia nos objetos estáticos da cena. Essa superfície pode ser modificada dinamicamente durante o jogo de acordo com o movimento dos personagens ou à física corporal atingida. Os personagens AI são capazes andar nessa superfície onde interagem em caminhos compostos por ela fazendo com que a inteligência artificial escolha uma animação apropriada para cada ação (ŠMÍD, 2017).

2.3.6 Codificação

O motor do jogo também deve fornecer uma maneira de descrever os componentes do jogo e seus comportamentos. Podemos escrever scripts como componentes para objetos para que os objetos podem reagir aos impulsos do jogador e interagir entre si. Na Unity, a linguagem de script é C # e javascript, já a Unreal oferece C ++ nativo ou Blueprint através de scripts visuais. É importante que os scripts possam ser adicionados e reutilizados para interação na cena de uma maneira que qualquer usuário tenha acesso (ŠMÍD, 2017).

2.3.7 Áudio

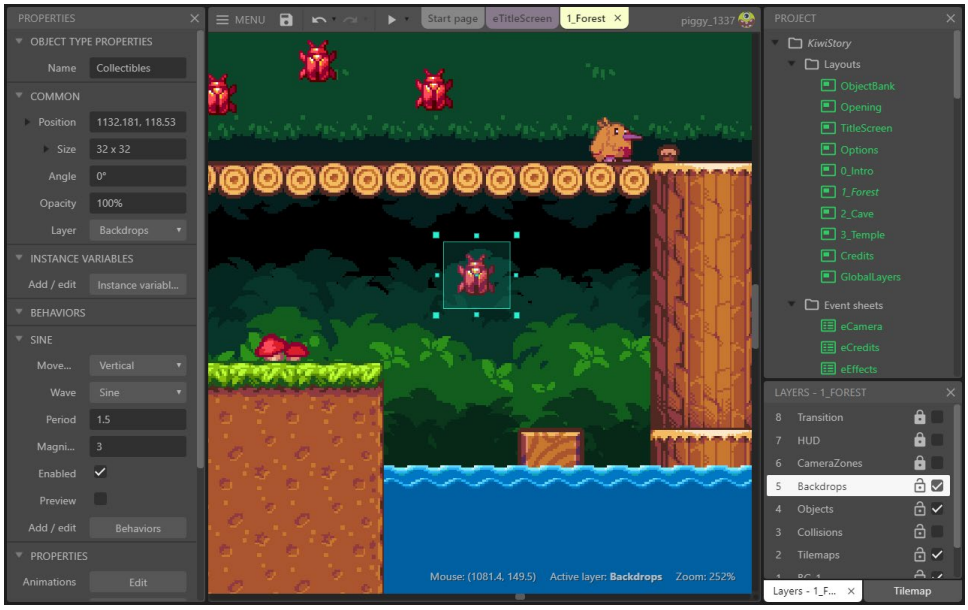
Uma parte crucial em todo jogo é imersão proporcionada elas faixas de áudio, que tem a função de induzir o usuário a atmosfera. Os motores de jogo fornecem ferramentas para tocar e parar trilhas sonoras com automação baseada em eventos realizados pelo jogador. Mecanismos de áudio avançados podem simular ecos

Tridimensionais (3D), tendo múltiplas aplicações como reproduzir um som espacial de acordo com a posição do jogador (ŠMÍD, 2017).

2.3.8 Construct

A Construct é uma das principais ferramentas quando o assunto é facilidade no desenvolvimento. Através dela é possível criar jogos usando uma estrutura intuitiva em HTML5 que posteriormente podem ser convertidos para jogos Web, Desktop e mobile. A engine faz com que o ambiente de desenvolvimento de jogos 2D seja de fácil acesso independente do conhecimento do desenvolvedor, ela vem com um conjunto completo de recursos poderosos, tem a capacidade de suportar várias plataformas, possuindo um sistema de programação visual fácil de entender (SUBAGIO,2014). A figura 2 a seguir, mostra como é sua interface na versão 3.0.

Figura 2 – Interface Construct v.3.0.



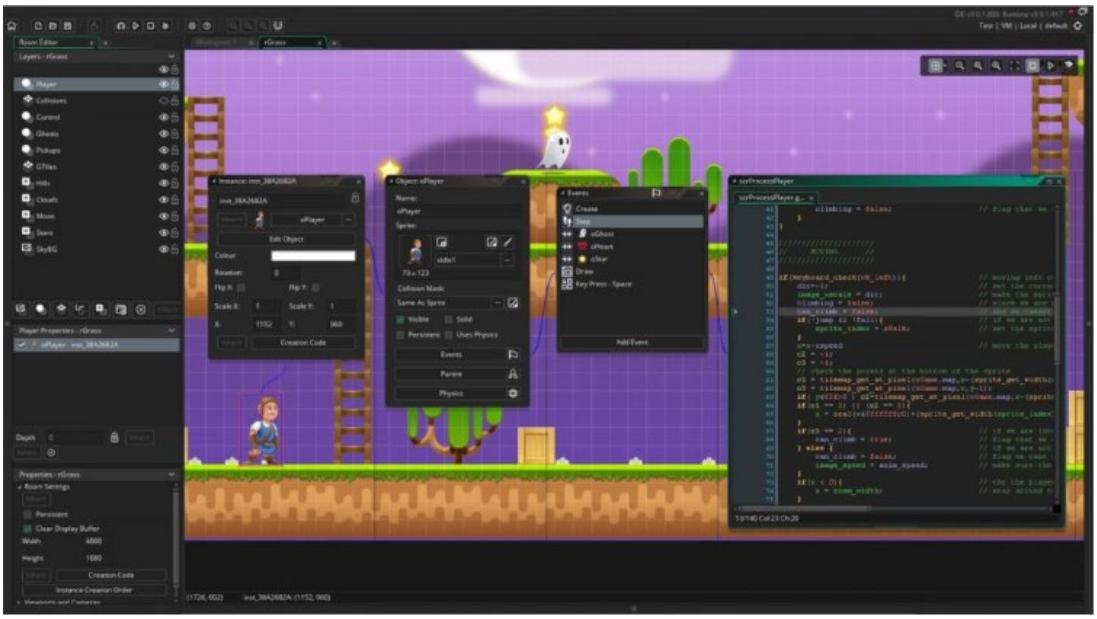
Fonte: SCIRRA, 2020.

2.3.9 Game Maker

Uma das engines mais antigas do mercado e também uma das mais simples, produzido pela YoYo Games é um motor que disponibiliza a criação de jogos de uma forma simples e intuitiva (figura 3). A Game Maker provê um ambiente simples que permite a iniciantes a criação de jogos, usando um sistema baseado em ícones de eventos e ações. A técnica de programação ‘’arrasta e solta’’ disponibiliza uma fácil

aprendizagem que te permite a criação de jogos sem o uso da programação tradicional (ROHDE,2014).

Figura 3 – Interface Game Maker v.2.2.



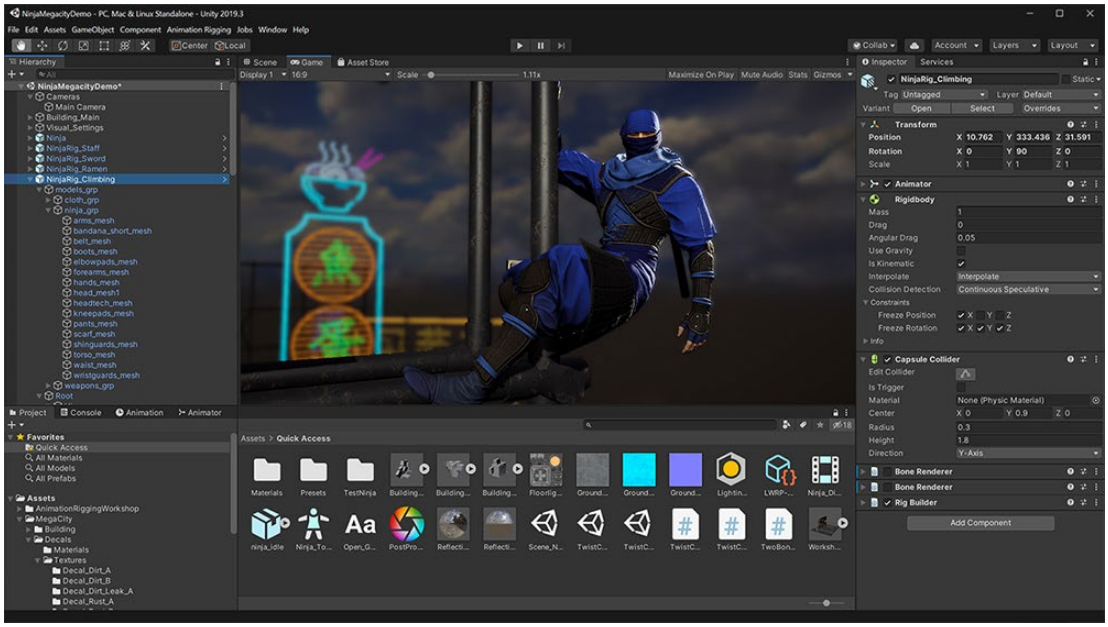
Fonte: YOYO GAMES, 2017.

2.3.10 Unity

O Unity é um poderoso mecanismo e editor de jogos integrados, permitindo a criação de objetos de forma rápida e eficiente, importar ativos externos e vinculá-los todos com o código principal. É conhecida por sua alta gama de funções e ambiente de desenvolvimento, sendo considerada por muitos a engine mais poderosa. É muito utilizada por desenvolvedores independentes, mas também por grandes empresas do ramo como Disney, Ubisoft, Disney e Electronic Arts (MENARD; WAGSTAFF, 2015).

Altamente flexível permite a criação de jogos 2D simples até jogos 3D com gráficos exuberantes. O Unity também possui um ambiente de script integrado, embutido recursos de rede e a capacidade de criação e implementação para várias plataformas (MENARD; WAGSTAFF, 2015). Na figura 4 a seguir podemos observar sua interface na versão 2019.3.

Figura 4 - Interface Unity v.2019.3.



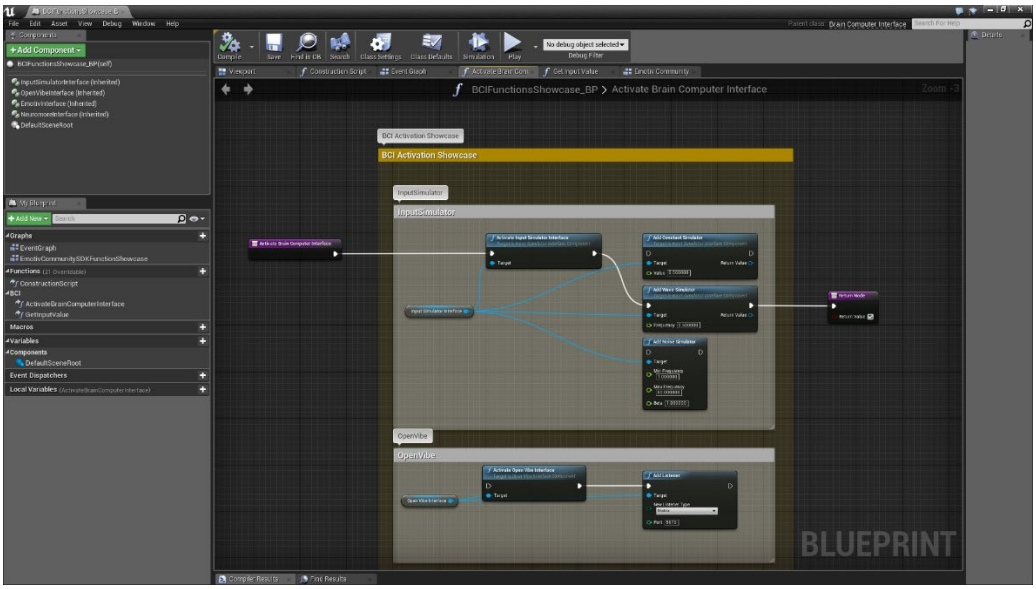
Fonte: UNITY, 2020.

2.3.11 Unreal Engine

Unreal é a engine que tem o maior poder gráfico da atualidade, sendo desenvolvida pela Epic Games. Devido ao seu poder gráfico é usada por grandes empresas do ramo em seus projetos.

O mecanismo Unreal é atualmente o mecanismo líder em visualização realista, vegetação e criação de terrenos. Além disso tudo, a ferramenta é excelente para iniciantes no mundo do desenvolvimento de jogos, pois não há necessidade de usos de código se assim desejar facilitando imensamente o desenvolvimento. Tudo isso graças ao sistema de blueprints, que é uma linguagem de scripts visuais. O motor é completamente gratuito, o que torna a Unreal uma das melhores opções para quem quer desenvolver um game 3D além dos vários assets gratuitos na biblioteca da mesma, o que facilita ainda mais o desenvolvimento quando se trata da elaboração, modelagem e animação de personagens e cenários (ŠMÍD, 2017). Na figura 5, podemos ver sua interface na versão 4.25.

Figura 5 - Interface Unreal v.4.25.



Fonte: EPIC GAMES, 2020.

2.3.11.1 Blueprints

A Unreal possui o sistema blueprint que é um grande diferencial em substituição da programação para componentes visuais.

As blueprints são gráficos feitos de blocos que quando conectados criam a ação desejada, baseada na lógica de programação, substituindo a codificação escrita que conhecemos (ŠMÍD, 2017).

Existem duas maneiras de codificar comportamentos no UE4: C ++ e blueprint. As Blueprints são um sistema de script visual muito populares na indústria de jogos digitais com o foco na prototipagem rápida. Seu design visual baseado em nós é funcionalmente mais rápido para trabalhar e mais fácil de depurar. As classes blueprint também podem ser convertidas em classes C ++ nativas caso desejado para optimização do desempenho do tempo de execução (ŠMÍD, 2017).

2.3.11.2 Como elas funcionam?

Em sua forma básica, os blueprints são adições visualmente com script para o jogo. Ao conectar nós, eventos, funções e variáveis a fios, é possível criar elementos de jogabilidade complexos. Os blueprints funcionam usando gráficos de nós para vários propósitos: construção de objetos, funções individuais e eventos gerais de

jogabilidade específicos de cada instância do blueprint, a fim de implementar o comportamento e outras funcionalidades (EPIC GAMES, 2020).

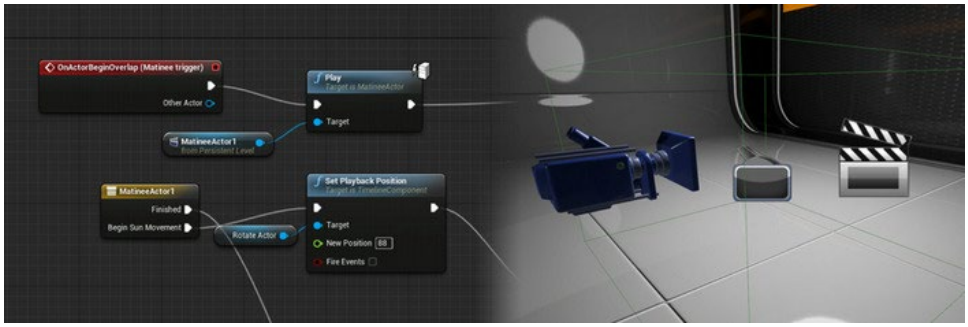
2.3.11.3 Tipos de blueprints

Os tipos de blueprints mais usadas são as *Level* blueprints, que são relacionadas aos níveis do jogo, e as blueprint classes que são suas classes (EPIC GAMES, 2020).

2.3.11.4 Level blueprint

Cada nível possui seu próprio blueprint de nível, e isso pode fazer referência e manipulação de atores dentro do nível, controlar cinemáticas e gerenciar itens como fluxo de nível, pontos de verificação e outros sistemas relacionados a níveis. O Level blueprint (Figura 6) também pode interagir com as Classes (EPIC GAMES, 2020).

Figura 6 – Level Blueprint.



Fonte: EPIC GAMES, 2020.

2.3.11.5 Classe blueprint

As classes blueprint são ideais para criar ativos interativos, como portas, interruptores, itens colecionáveis e cenários destrutíveis. Devido à natureza independente elas podem ser construídas de forma que você possa colocá-las em um nível e simplesmente automatizar sua função. Significando assim que a edição de uma blueprint que está sendo usada em um projeto atualizará todas as instâncias automaticamente ao serem alteradas (EPIC GAMES, 2020).

Na Figura 7 a seguir, pressionando o botão um evento é ativado dentro da blueprint da porta, fazendo com que ela se abra.

Figura 7 – Classe Blueprint.

Fonte: EPIC GAMES, 2020.

As blueprints podem ser adaptadas para a execução de diversas funções, tendo uma variedade de aplicações usando sua automatização dinâmica e precisa facilitando processo de criação e desenvolvimento.

3 MATERIAIS E MÉTODOS

Segundo Reis (2003, p. 5), “Processo de Software é um conjunto de atividades realizadas para construir software, levando em consideração os produtos sendo construídos, as pessoas envolvidas, e as ferramentas com as quais trabalham.”

Dentre os vários modelos de processos de software, temos as chamadas metodologias ágeis. Essa metodologia se mostra bem eficaz, quando aplicada no desenvolvimento de pequenos projetos e onde não há regulamentos externos que afetam o software (SOMMERVILLE, 2011).

3.1 METODOLOGIAS ÁGEIS

Após a criação do Manifesto para Desenvolvimento Ágil de Software em fevereiro de 2001, as metodologias ágeis se concretizaram. Insatisfeitos com os métodos tradicionais, representantes de diversas metodologias uniram seus conhecimentos para elaborar o manifesto, o qual traz os princípios que as metodologias ágeis devem ter, para assim serem classificadas (SOMMERVILLE, 2011). O Quadro 2 a seguir mostra esses princípios:

Quadro 2 – Princípios dos métodos ágeis

Princípios	Descrição
Envolvimento do cliente	Os clientes devem estar intimamente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar suas iterações.
Entrega incremental	O software é desenvolvido em incrementos com o cliente, especificando os requisitos para serem incluídos em cada um.
Pessoas, não processos	As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Membros da equipe devem desenvolver suas próprias maneiras de trabalhar, sem processos prescritivos.
Aceitar as mudanças	Deve-se ter em mente que os requisitos do sistema vão mudar. Por isso, projete o sistema de maneira a acomodar essas mudanças.
Manter a simplicidade	Focalize a simplicidade, tanto do software a ser desenvolvido quanto do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar a complexidade do sistema.

Fonte: Sommerville, 2011

Essas metodologias, tem como foco principal os indivíduos e como eles interagem entre si, no desenvolvimento do software e nas respostas rápidas a possíveis mudanças. Isto é alcançado através dos ciclos iterativos e com objetivos de curto prazo, tornando possível ajustar os requisitos até entrar na fase de desenvolvimento (CARVALHO et al., 2011).

Dentre as metodologias ágeis conhecidas podemos citar o *Dynamic Systems Development Method* (DSDM), *Adaptative Software Development* (ASD), *Extreming Programming* (XP) e o *Agile Modeling* (AM) o qual será usado para o auxílio durante o desenvolvimento deste projeto.

3.1.1 Scrum

Schwaber e Sutherland (2017), define Scrum como, um *framework* dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível.

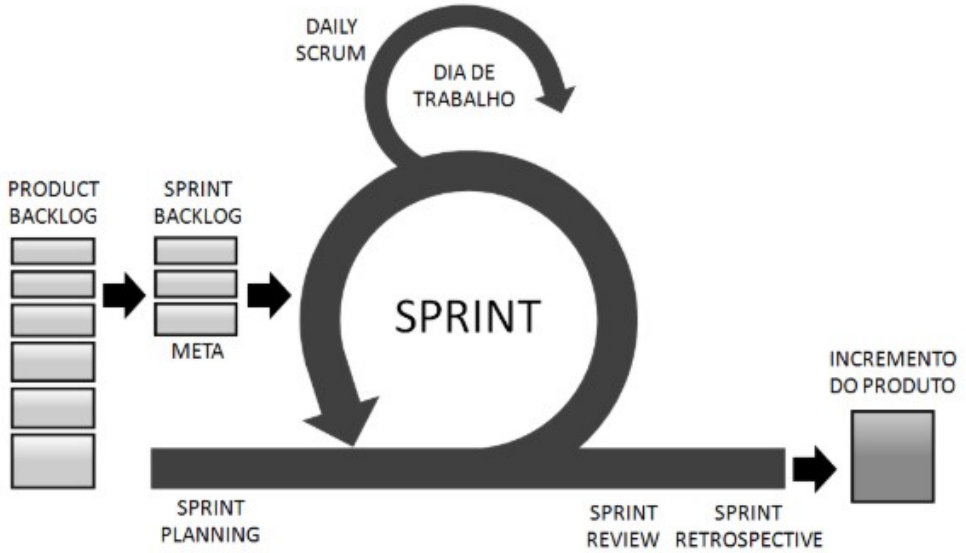
Essa metodologia divide os papéis a serem desempenhado por cada integrante da equipe em três: *Product Owner*, *Scrum Master* e *Scrum Team* (SCHWABER; SUTHERLAND, 2017).

- O *Product Owner* (Dono do produto): tem a função de representar os interesses da equipe, gerenciar o Backlog e maximizar o valor do produto final;

- O *Scrum Team* (Time Scrum): fica encarregado de desenvolver o software e ao final de cada Sprint devem entregar um incremento pronto do produto, eles se organizam internamente e distribuem as tarefas entre si;
- *Scrum Master* (Mestre Scrum): responsável pela execução de todas as regras do Scrum.

A Figura 8 abaixo apresenta uma ilustração do ciclo do Scrum:

Figura 8 – Ciclo do Scrum.



Fonte: Oliveira ,2019

Os eventos do Scrum criam uma rotina, diminuindo assim a necessidade de reuniões não definidas, todos os eventos têm um prazo de duração máxima o que permite que o tempo seja gerenciado de forma eficaz, garantindo assim que não haja perdas no processo (SCHWABER; SUTHERLAND, 2017).

A *Sprint* é o principal evento do Scrum, e geralmente dura um mês ou menos. Ao final desse período um incremento “pronto” é criado e uma nova Sprint é iniciada. Cada Sprint tem uma definição do que será desenvolvido e durante o ciclo isso é detalhado e revisado entre o time e o *product owner*, e mudanças podem ser solicitadas desde que não afete o objetivo inicial da Sprint (SCHWABER; SUTHERLAND, 2017).

O Scrum Team realiza reuniões diárias com duração de 15 minutos, onde se planeja o que será feito nas próximas 24 horas e analisa-se o progresso do dia

anterior. Durante a reunião os membros da equipe devem responder ‘’aos seguintes questionamentos (SCHWABER; SUTHERLAND, 2017):

- O que foi feito desde a última reunião?
- O que será feito até a próxima reunião?
- O que está impedindo na conclusão do trabalho?

Quando a *Sprint* termina, é realizada a reunião de revisão da Sprint com objetivo inspecionar os incrementos e adaptar o backlog se for necessário. Conta com a participação de toda a equipe além de todas as partes interessadas no projeto. A reunião deve ter a duração máxima de 4 horas, a duração varia de acordo com o tamanho da Sprint (SCHWABER; SUTHERLAND, 2017).

No Scrum os artefatos são projetados para maximizar a transparência das informações para toda a equipe, garantindo assim que todos os incrementos prontos serão entregues no prazo estipulado. Os principais artefatos são: o *backlog* do produto, o *backlog* da Sprint e o incremento do produto (SCHWABER; SUTHERLAND, 2017).

3.2 GAME DESIGN DOCUMENT (GDD)

Neste tópico será apresentado o *GDD* do jogo Barbarian, que será desenvolvido durante a construção deste projeto.

3.2.1 História

O jogo se passa na era viking, tendo como palco uma terra desconhecida abitada por nômades de um grupo denominado *Morak’s.* Temos como principal personagem, *Brute,* um viking explorador que acabou sendo designado para o reconhecimento de tal área. Altamente treinando na arte da batalha faz uso de seu machado para derrotar seus inimigos e causar dano.

3.2.2 Cenário

O jogo será composto por apenas um cenário, com um vasto campo, árvores e uma espécie de santuário. Tendo como objetivo demonstrar terras amplas e ainda não exploradas.

3.2.3 Gameplay

O gameplay se constitui em atacar nos momentos certos, esperar as brechas dos inimigos fazendo uso da sua esquiva e ataques diretos, junto com sua movimentação. Fazer uso da análise do padrão de ataques de cada inimigo, aproveitando o momento certo para uma investida.

3.2.4 Fluxo do jogo

O principal desafio do game se constitui ao enfrentar inimigos diversos tendo como única forma de defesa a esquiva. Os inimigos irão surgir em pontos estratégicos do mapa em direção ao protagonista.

Conforme o personagem avança chegando em certa área o chefe da fase irá ficar visível.

O jogador é considerado vitorioso ao derrotar todos os inimigos. Se denomina derrota ter a sua barra de vida zerada, sem possibilidade de ressurreição tendo que iniciar todo o trajeto novamente.

3.2.5 Personagens

O jogo terá um personagem jogável, e dois tipos de inimigos, que serão descritos a seguir:

- *Brute*: Um guerreiro viking que faz uso de sua destreza de combate no intuito de derrotar seus inimigos.
- *Morak*: Raça nômade agressiva que difere ataques com as próprias mãos.
- *Narbash*: Chefe final, uma raça diferente da *Morak,* maior e mais forte, agindo como líder deles.

3.2.6 Habilidades

Cada personagem terá habilidades um ataque básico que causam a mesma quantidade de dano.

Brute:

- Ataques normais: Golpe direto com a arma equipada.
- Movimentos defensivos: Rolamento, esquiva se projetando ao solo.

Morak:

- Ataques normais: Golpe direto com as mãos.

Narbash:

- Ataques normais: Golpes diretos muito fortes com sua clava.

3.2.7 Visual

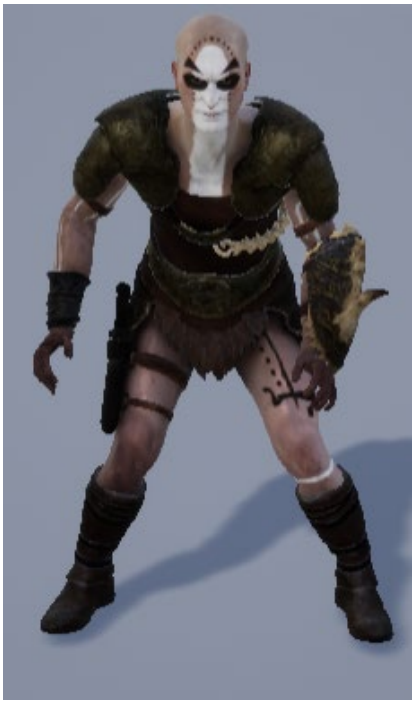
O personagem *Brute* tem um visual de guerreiro *Viking*, como podemos observar na Figura 9 a seguir.

Figura 9 – *Brute*.

Fonte: Elaborado pelos autores.

Os Inimigos *Moraks*, possuem vestimenta e aparência de um bárbaro, sua vestes foram feitas com a pele dos animais selvagens que habitavam aquela região, veja a Figura 10 a seguir.

Figura 10 – Morak.



Fonte: Elaborado pelos autores.

O Chefe final *Narbash* (Figura 11), tem a aparência de um *Troll* gigante que usa os crânios de suas vítimas como adorno, e possui duas clavas uma em cada mão.

Figura 11 – Narbash.

Fonte: Elaborado pelos autores.

3.2.8 Controles

Através dos botões de movimentação do teclado W, A, S, D se controla o posicionamento do personagem (cima, baixo, esquerda e direita); com a movimentação do mouse é possível ajustar o posicionamento da câmera; com a tecla espaço é executada a ação de esquiva, um rolamento simples; e com o clique esquerdo do mouse o ataque direto. Veja a Figura 12 a seguir.

Figura 12 – Ilustração dos controles.

Fonte: Elaborado pelos autores.

3.2.9 Visualização

O menu principal será visualizado no modo paisagem nas dimensões de 1920 pixels por 1080 pixels, tendo dois botões de interação (Figura 13).

Batalhar: que dá início ao nível.

Desistir: que fecha a aplicação imediatamente.

Figura 13 – Tela menu principal.

Fonte: Elaborado pelos autores.

A visão durante o jogo será em terceira pessoa (Figura 14), com a possibilidade de controle da câmera via mouse acompanhando o protagonista e ajustando a visualização de acordo com seu movimento.

Figura 14 – Visão da câmera.

Fonte: Elaborado pelos autores.

4 RESULTADOS E DISCUSSÃO

Ao longo deste projeto foram apresentados os processos e tecnologias que são necessários para a elaboração de um jogo.

Agora vamos colocar em prática esses processos e descrever a lógica por trás deles, buscando mostrar de maneira simplificada os passos que foram seguidos, para chegar ao resultado proposto.

4.1 ESCOLHA DOS ASSETS

A visualização do personagem e animações são um dos principais atrativos de um jogo digital, diferenciando sua jogabilidade e diversão através delas.

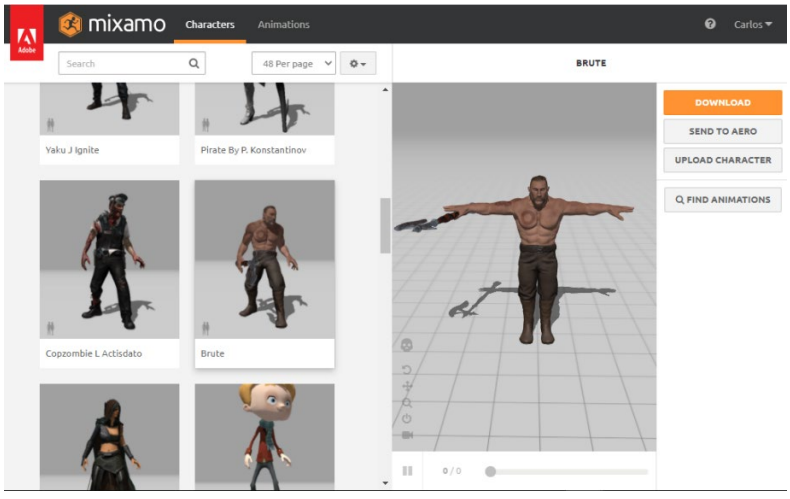
Optamos em evitar a modelagem e adquirir assets já produzidos incluindo personagens e animações através da plataforma *Mixamo* disponibilizada pela *Adobe.*

A plataforma funciona como um intermediador de assets, onde você pode até mesmo fazer upload do seu personagem modelado e automaticamente gerar um esqueleto para seu personagem possibilitando o uso das animações disponibilizadas na plataforma, assim como usar personagens e animações já criadas em seu banco de dados.

4.1.1 Personagem

O *asset* escolhido foi esse bárbaro denominado *Brute* (Figura 15), que além de relevante com a proposta do jogo já vem com a arma de combate acoplada, evitando a necessidade de inserção através de outros assets.

Figura 15 - Seleção de assets do personagem.

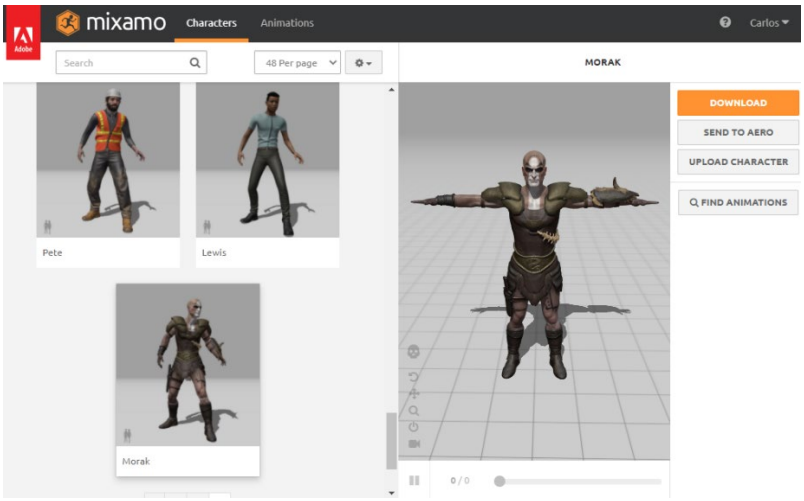


Fonte: ADOBE, 2020.

4.1.2 Inimigos

Para o inimigo, foi escolhido esse nômade chamado *Morak* (Figura 16), que irá atacar com as próprias mãos.

Figura 16 - Seleção de assets do inimigo.



Fonte: ADOBE, 2020.

Para o chefe foi selecionado um personagem já pronto do pacote *Paragon: Narbash* (Figura 17), disponível gratuitamente na loja da Epic Games.

O grande diferencial dos pacotes de personagens disponíveis pela própria desenvolvedora é que eles já vêm praticamente configurados, com suas animações, ataques, e interações, sendo necessária apenas a ação de concatenar essas informações com a *Blueprint* criada pelo desenvolvedor.

Figura 17 – Seleção de assets do chefe.

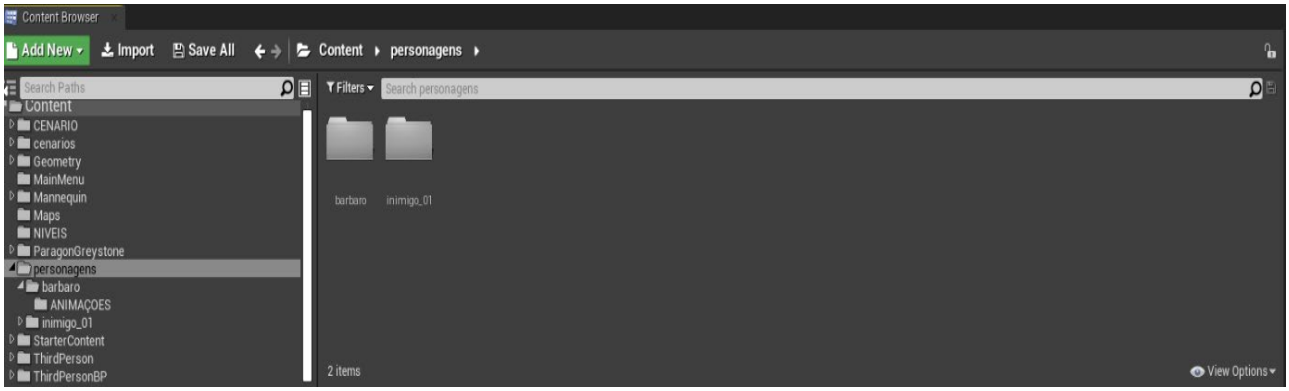
Fonte: Epic Games, 2020.

4.2 IMPORTANDO PARA A UNREAL ENGINE

4.2.1 Assets

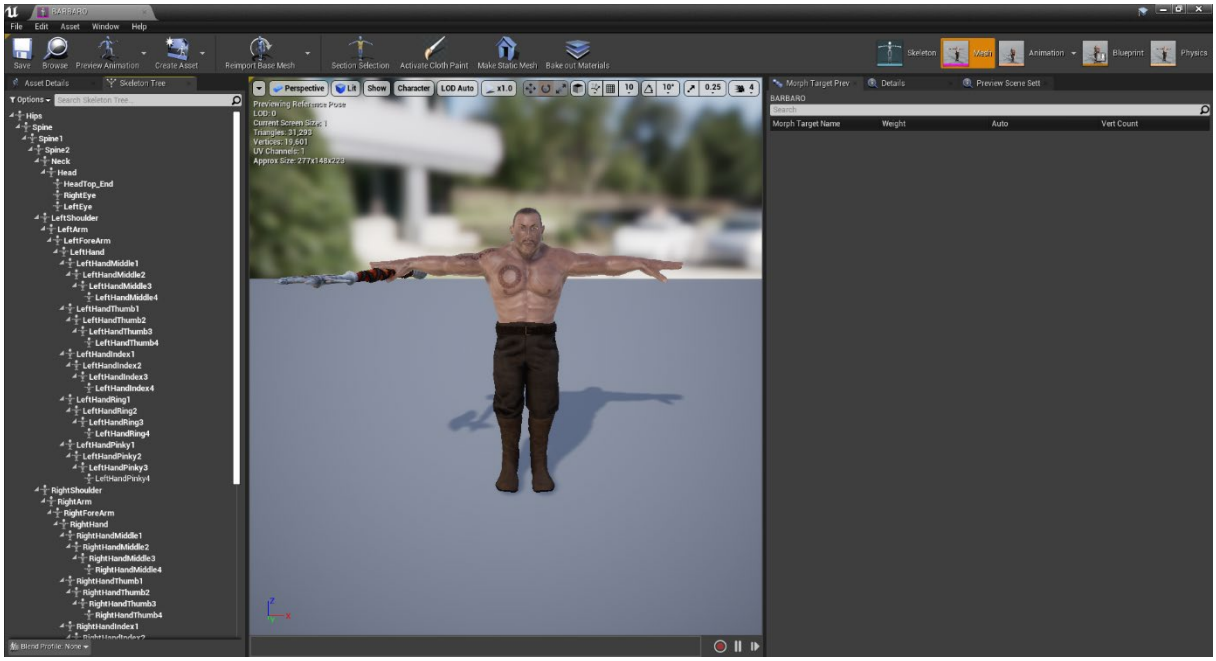
A importação é muito simples, apenas para melhor organização criamos uma pasta Personagens, e dentro dela subpastas contento as pastas respectivas a cada *asset* e animações (Figura 18). Feito isso apenas arraste o arquivo *.fbx* do *asset* desejado e solte dentro da pasta desejada. Nas Figuras 19, 20, 21, 22, 23 e 24 a seguir podemos observar que após a importação, temos acesso a todo o conteúdo como texturas, animações e assets.

Figura 18 - Organização de pastas no *content browser*.



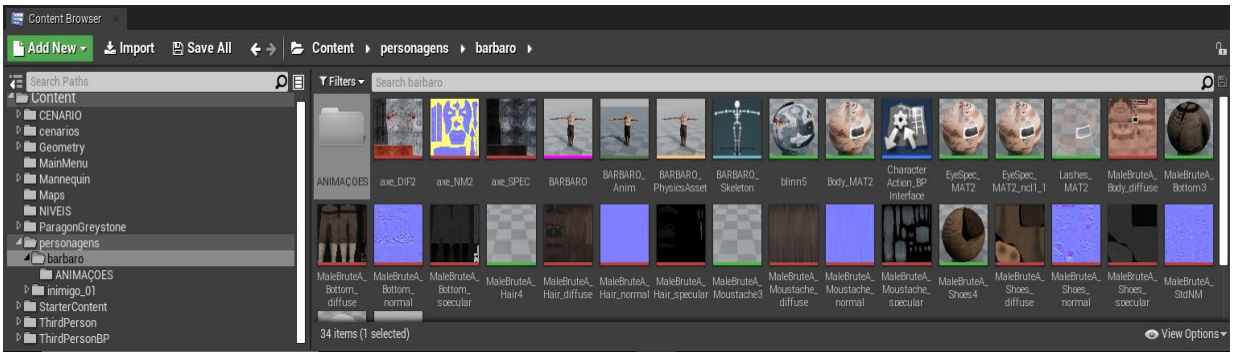
Fonte: Elaborado pelos autores.

Figura 19 - Análise do *asset Brute*.



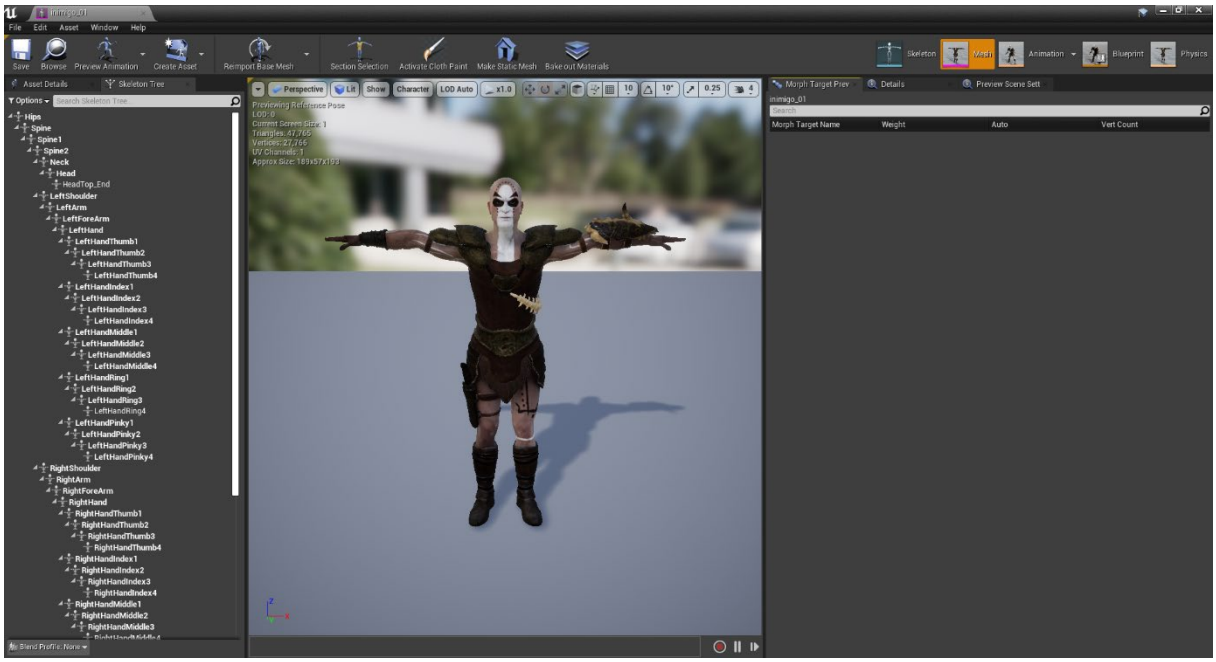
Fonte: Elaborado pelos autores.

Figura 20 – Pasta do personagem com seus materiais e *assets*.



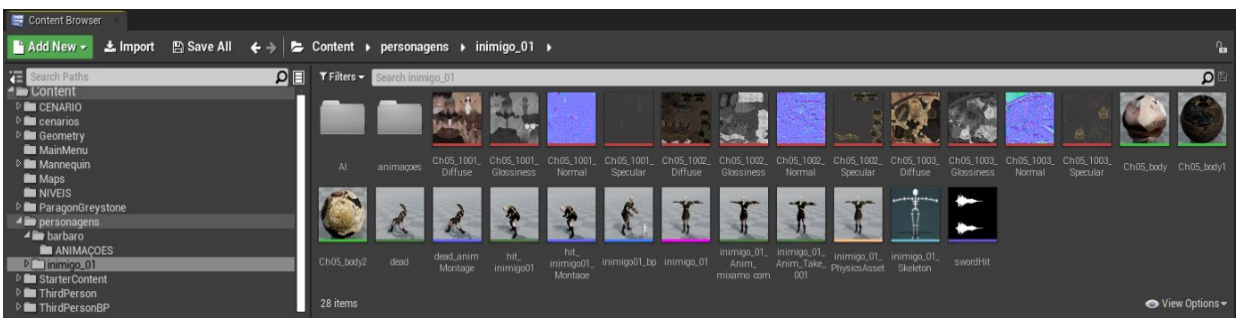
Fonte: Elaborado pelos autores.

Figura 21 - Análise do *asset Morak.*



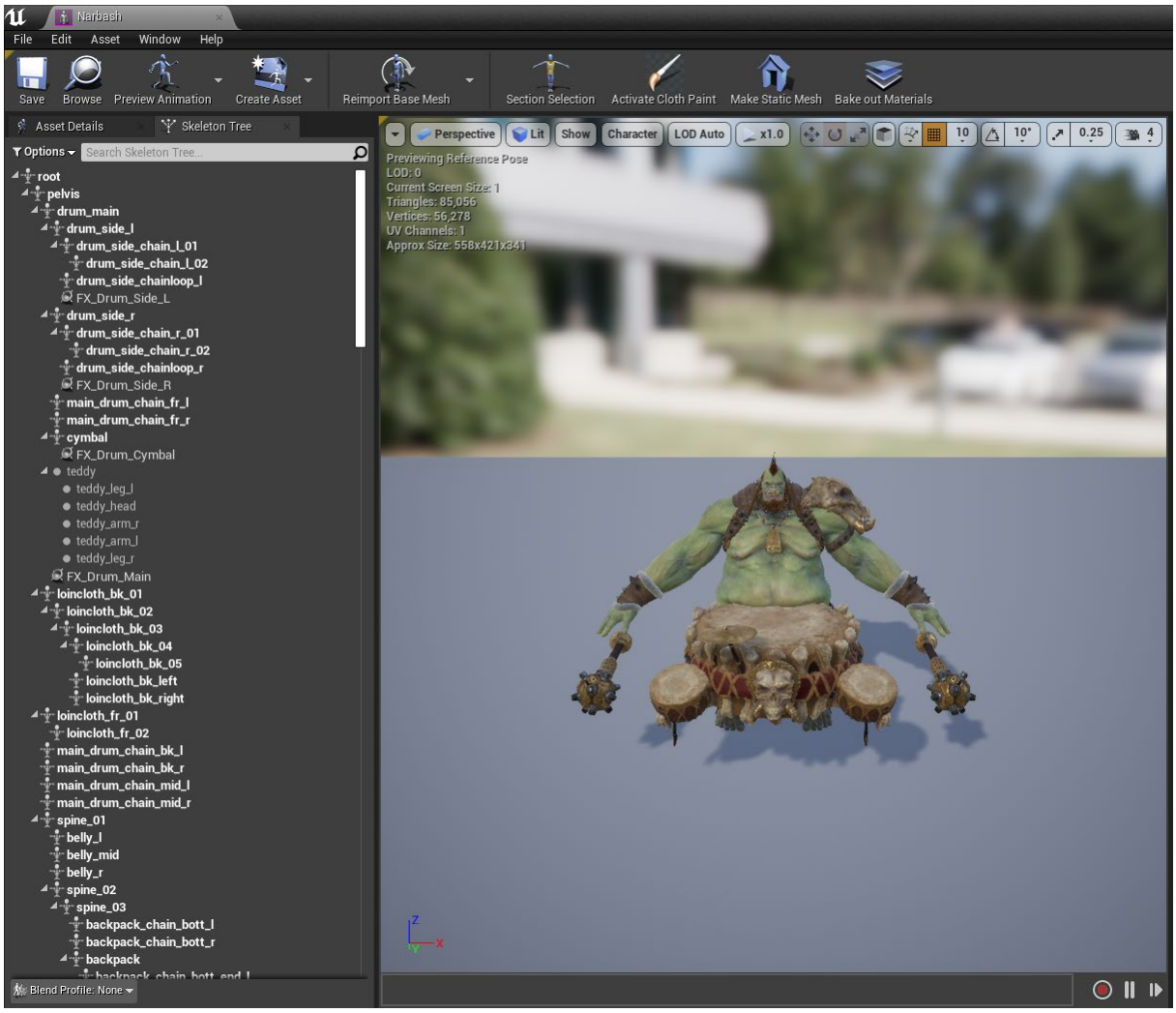
Fonte: Elaborado pelos autores.

Figura 22 – Pasta do inimigo com seus materiais e *assets*.



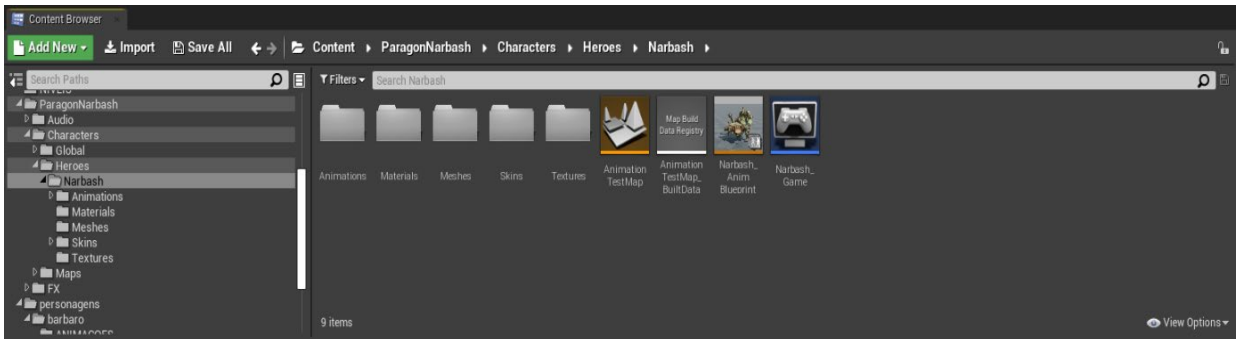
Fonte: Elaborado pelos autores.

Figura 23 - Análise do asset *Narbash.*



Fonte: Elaborado pelos autores.

Figura 24 - Pasta do chefe com seus materiais e *assets*.

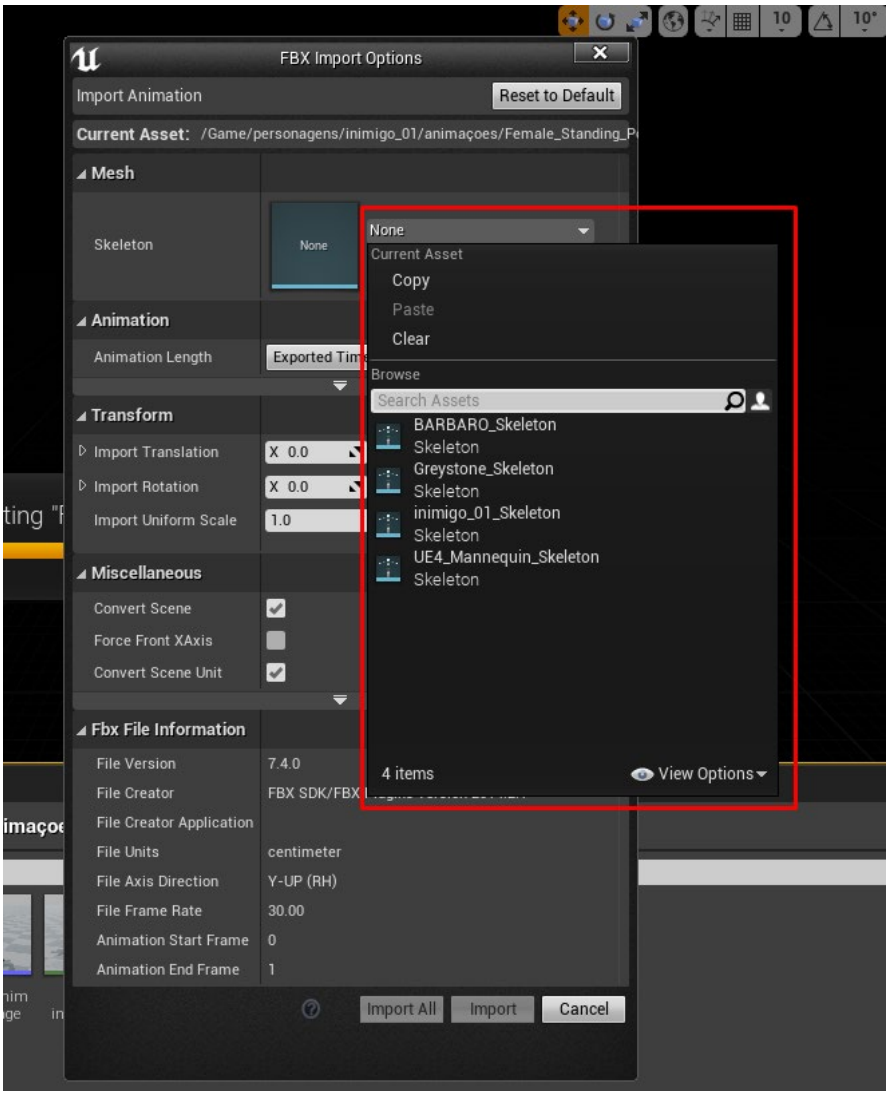


Fonte: Elaborado pelos autores.

4.2.2 Animações

Para a importação das animações o processo é o mesmo, sendo necessária apenas a seleção do esqueleto desejado no qual a animação será aplicada. A figura 25 mostra como é feita a importação.

Figura 25 – Importando a animação.

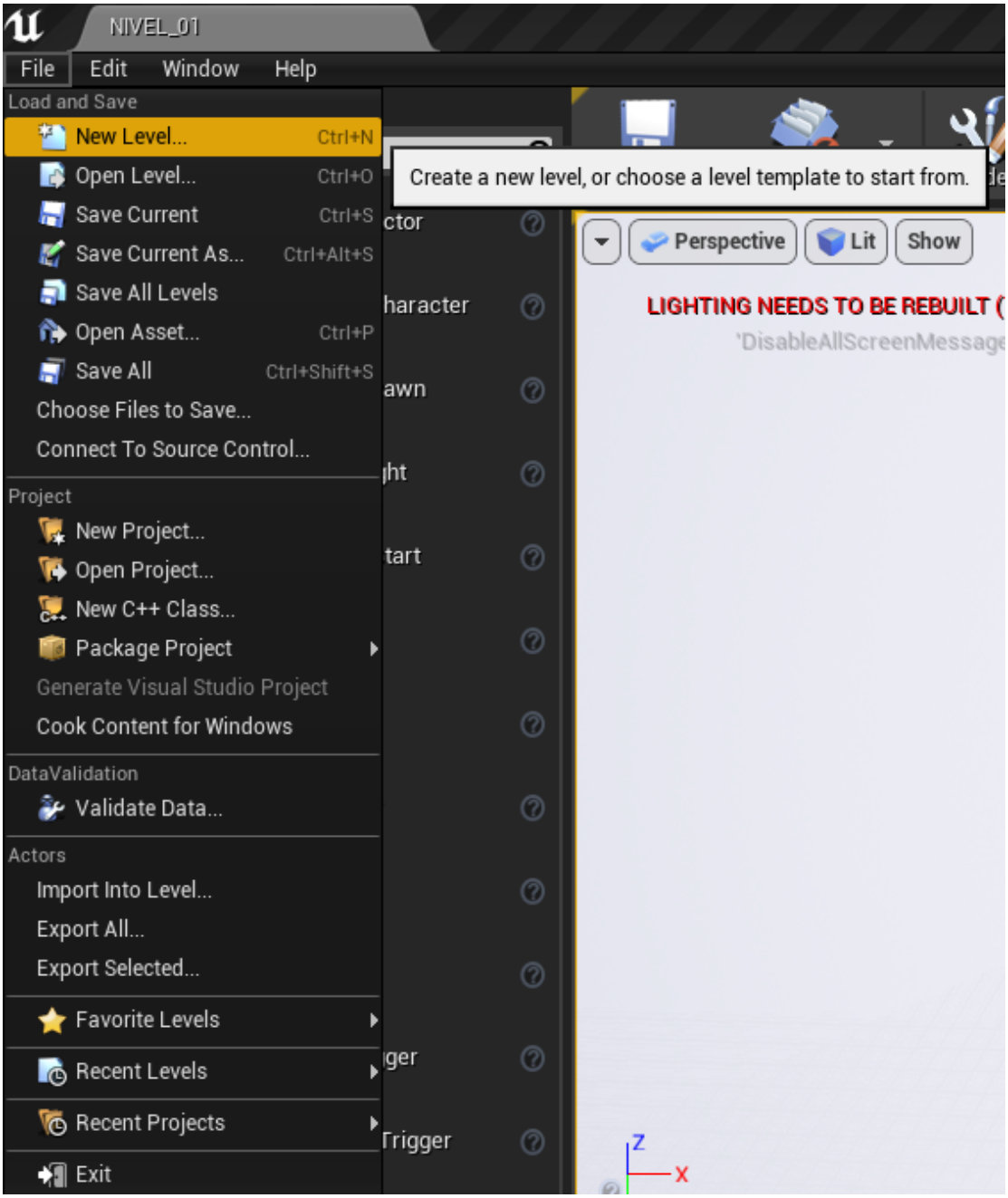


Fonte: Elaborado pelos autores.

4.3 CRIANDO O CENÁRIO

Para a elaboração do cenário foi usada uma blocagem básica sem muitos detalhes e texturas. Primeiramente foi inserido um novo nível onde ficará localizado nosso mapa principal (Figura 26).

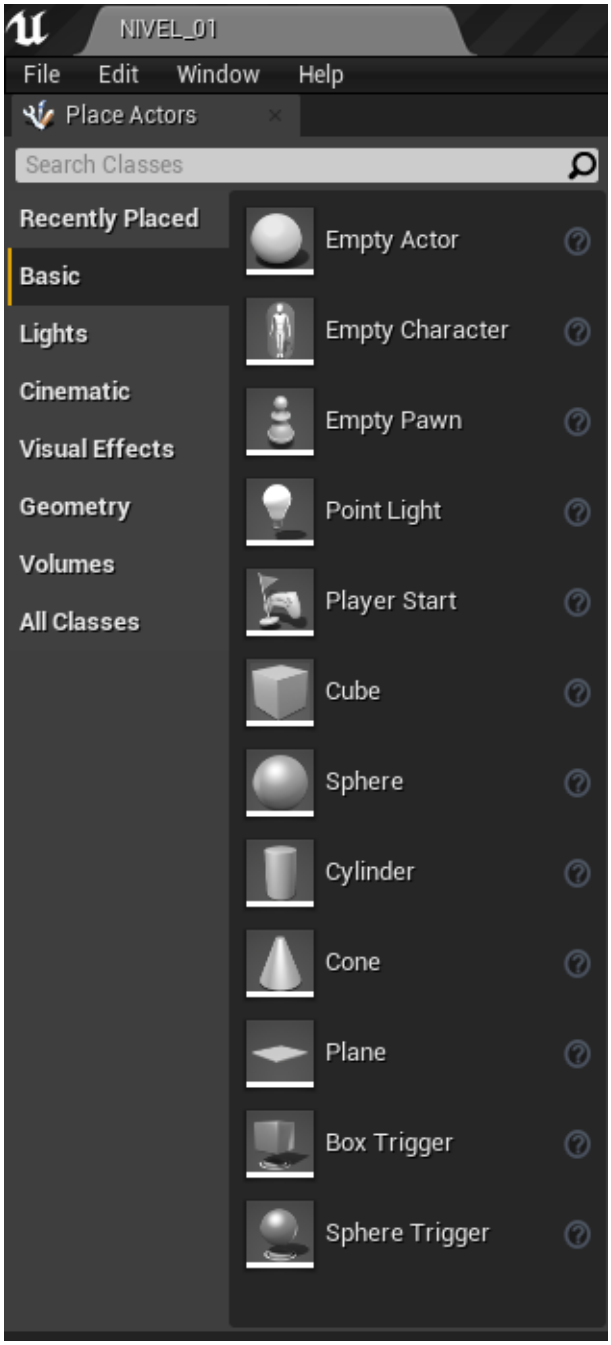
Figura 26 – Criando novo nível.



Fonte: Elaborado pelos autores.

Após a criação do nível foi inserido um *static mesh* cilíndrico (*Static Mesh Cylinder*) através do menu *Place Actors* para ser usado como chão da área do chefe (Figura 27).

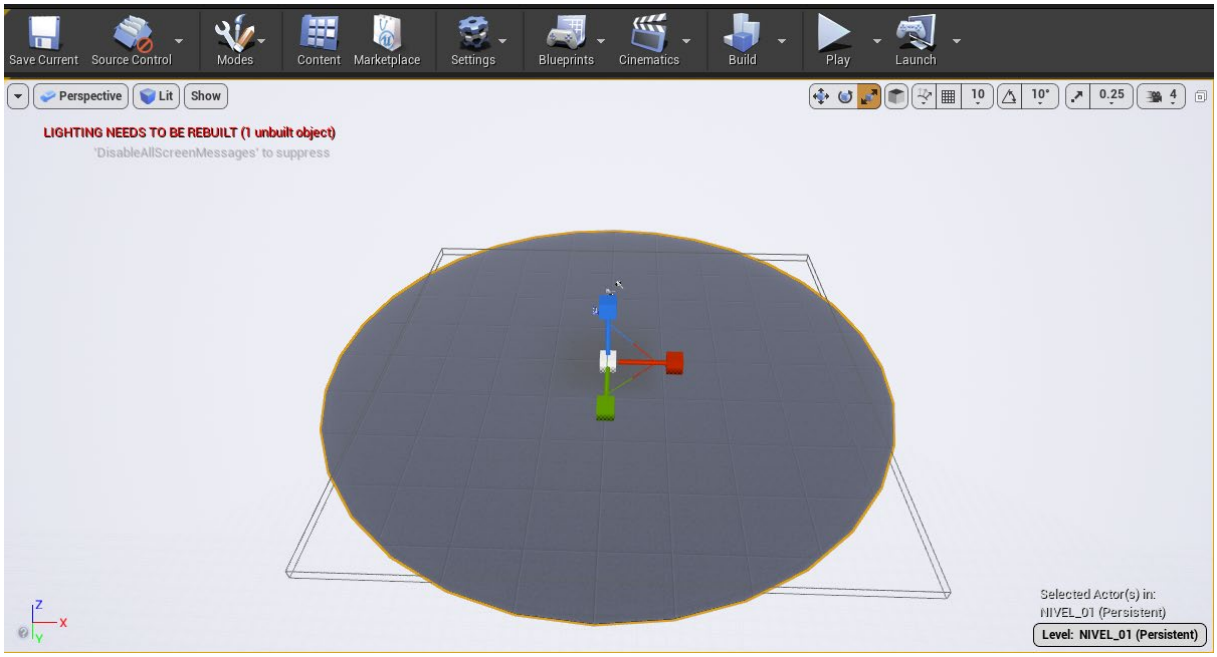
Figura 27 – Inserção *static mesh*.



Fonte: Elaborado pelos autores.

Após a seleção o *Cylinder* foi posicionado de acordo com o tamanho que será necessário para a movimentação do personagem. Como podemos observar na Figura 28.

Figura 28 – Posicionamento do *cylinder*.



Fonte: Elaborado pelos autores.

Após o posicionamento do chão foram inseridos assets como colunas, pedras e algumas texturas para melhor ambientação (Figura 29).

Figura 29 – Criando a ambientação.

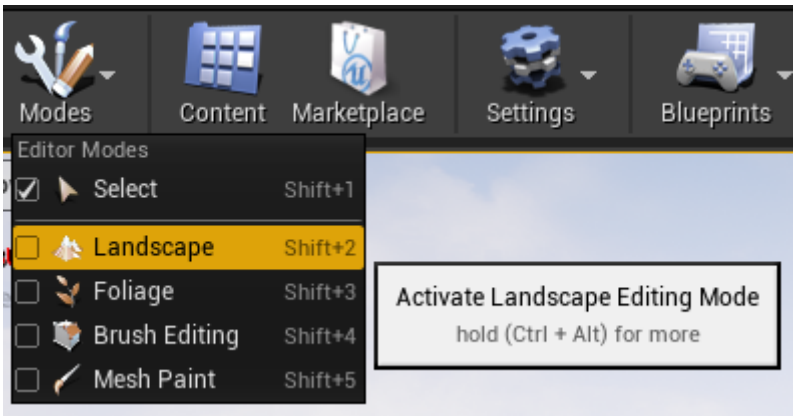
Fonte: Elaborado pelos autores.

4.3.1 Landscape

Com a área do chefe já ambientada, foi necessária a criação de um *Landscape* para melhor imersão, pois graças a esse formato de edição temos mais liberdade e facilidade na inserção de objetos para a composição.

O processo de criação de um *Landscape* é muito simples (Figura 30), sendo necessária apenas alternar o modo de trabalho para *Landscape* selecionando o tamanho de quadros desejados.

Figura 30 – Criando o *Landscape*.



Fonte: Elaborado pelos autores.

Nesse modo é possível criar inclinações no próprio mapa simulando montanhas e áreas mais íngremes como mostrado nas imagens anteriores (Figura 31).

Figura 31 – Vista aérea do *Landscape*.

Fonte: Elaborado pelos autores.

A fim de tornar o cenário mais imersivo, foi necessário a inserção de alguns elementos que ajudaram a ambientar o cenário, as Figuras 32,33 e 34 a seguir ilustram esse processo.

Figura 32 – Colunas para ambientação do *Landscape*.

Fonte: Elaborado pelos autores.

Figura 33 - Cabeça de estátua para ambientação do *Landscape*.

Fonte: Elaborado pelos autores.

Figura 34 – Rochas para ambientação do *Landscape*.

Fonte: Elaborado pelos autores.

O modo *foliage* como o próprio nome diz, serve para a inserção de plantas, árvores e quaisquer *mesh's* que deseje, agilizando muito o processo de inserção de objetos, evitando a colocação individual deles. Esse processo (Figura 35), foi utilizado para o posicionamento das árvores e folhagens.

Figura 35 – Utilização do modo *foliage*.

Fonte: Elaborado pelos autores.

4.4 HUD

Mesmo que a simplicidade do projeto seja o objetivo inicial, a elaboração de um HUD (*Heads-Up Display*) é essencial, facilitando a identificação de funções necessárias dentro do jogo, melhorando o visual e a jogabilidade.

No caso atual foi elaborado um modelo básico (Figura 36), com a ajuda de um software de edição, contendo apenas o visual de uma barra de vida onde posteriormente inserimos a função de decremento da barra de vida do personagem.

Figura 36 – HUD da barra de vida.

Fonte: Elaborado pelos autores.

4.5 TELA DE FIM DE JOGO

Sua finalização ocorre quando o personagem adentra a área do chefe, sendo exibida a mensagem de fim de jogo por cerca de 10 segundos com a trilha sonora até que o jogo seja encerrado completamente (Figura 37).

Figura 37 – Tela de fim de jogo.

Fonte: Elaborado pelos autores.

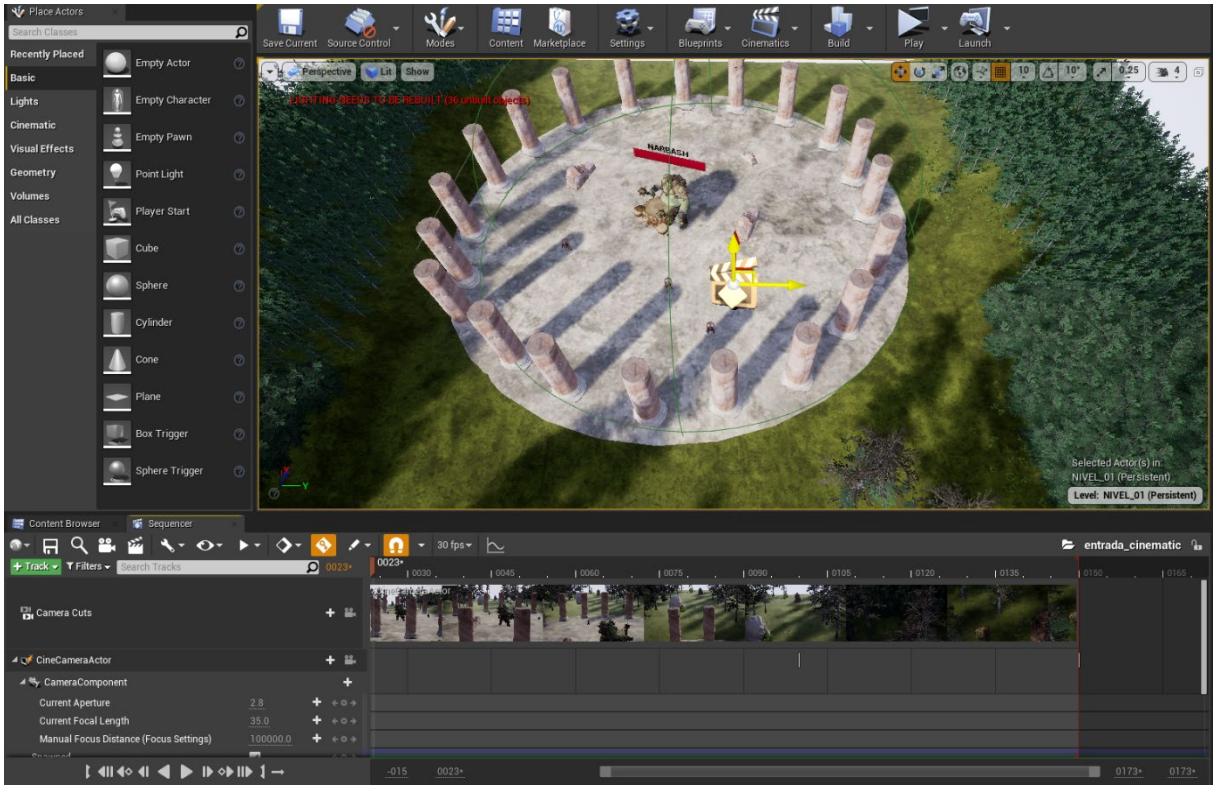
4.6 IMPLEMENTAÇÃO

4.6.1 Cinematic de Entrada

A Unreal possibilita um vasto número de transições de cenas que podem ser executadas pelo próprio desenvolvedor no manuseio da câmera, tais como movimentação, distância e efeitos de aproximação, que podem ser usadas para a elaboração de *cinematics* conhecidas também como *cutscenes,* para uma melhor impressão e visão de certos momentos pelo jogador.

Foi elaborada uma transição no início do jogo que tem início no fim do mapa, se movimentando até a posição do jogador gerando uma imersão de primeiro contato logo no início do jogo (Figura 38). Após a transição ser finalizada a movimentação do jogador é liberada.

Figura 38 – Elaboração da cutscene inicial.



Fonte: Elaborado pelos autores.

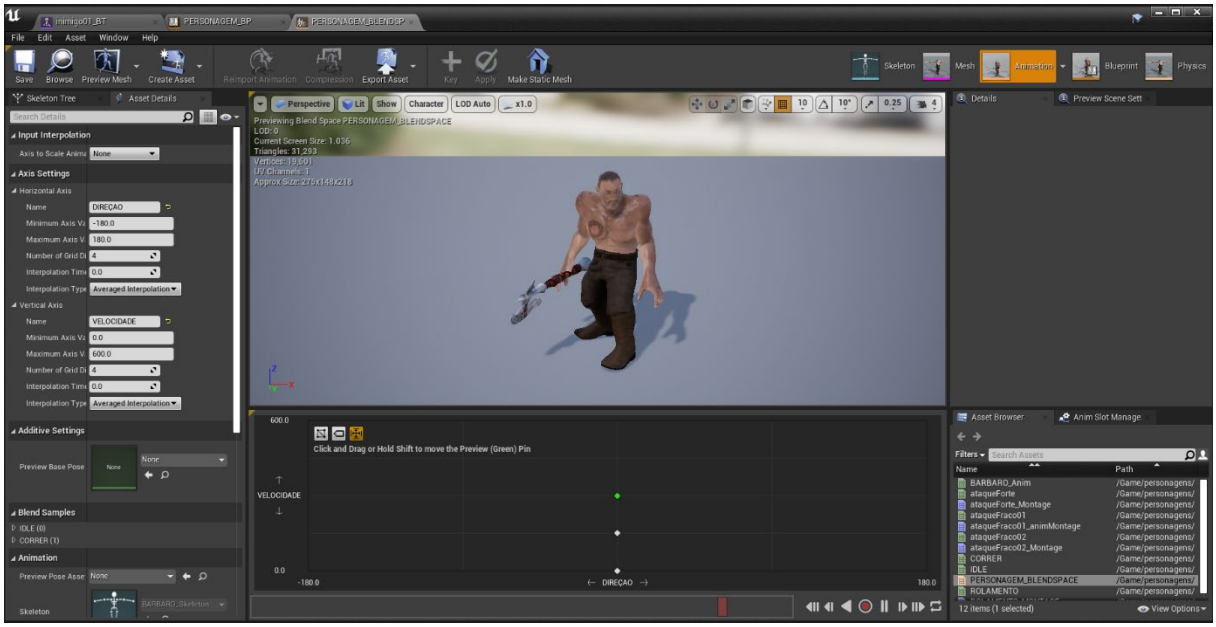
4.6.2 Animações

As animações são responsáveis pela vida e diversão produzidas em um jogo, por isso nessa etapa é muito importante a seleção e configuração correta de boas animações com o objetivo de otimizar a movimentação e criar uma profundidade na ação das mesmas.

Nesse modelo fizemos uso das *blendspaces* (Figura 39), que tem como principal função a interação entre animações de uma forma suave. Podemos ver como exemplo a interação da animação de correr, e ficar parado do nosso personagem, caso fosse feita uma transição de certa forma ‘’seca’’ a movimentação ficaria engessada passando o contrário do objetivo principal.

Fazendo uso da *blendspace* e da configuração correta de blueprints essa movimentação pode ser executada de forma suave e humana.

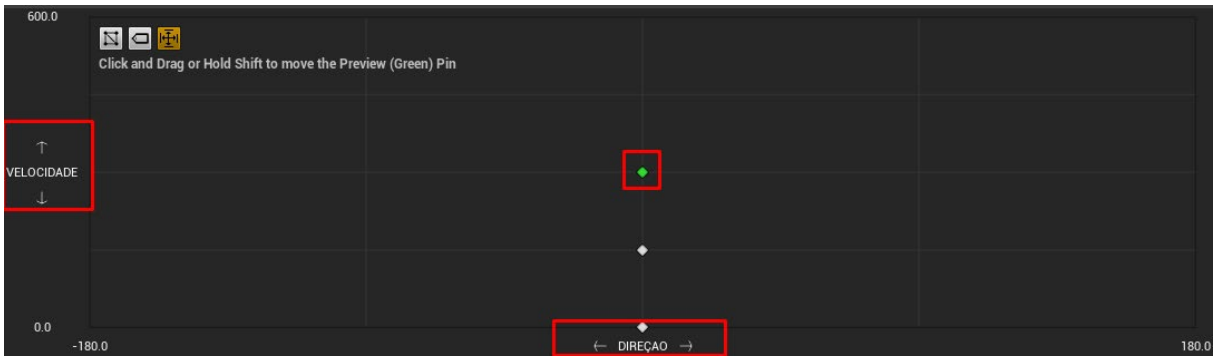
Figura 39 – Inserção da *blendspace*.



Fonte: Elaborado pelos autores.

Através desses pequenos retângulos são inseridas as animações desejadas (Figura 40), nesse caso animações de corrida, e de imóvel, fazendo com que através da movimentação e direção do personagem a animação seja alterada de parado para correndo e vice-versa.

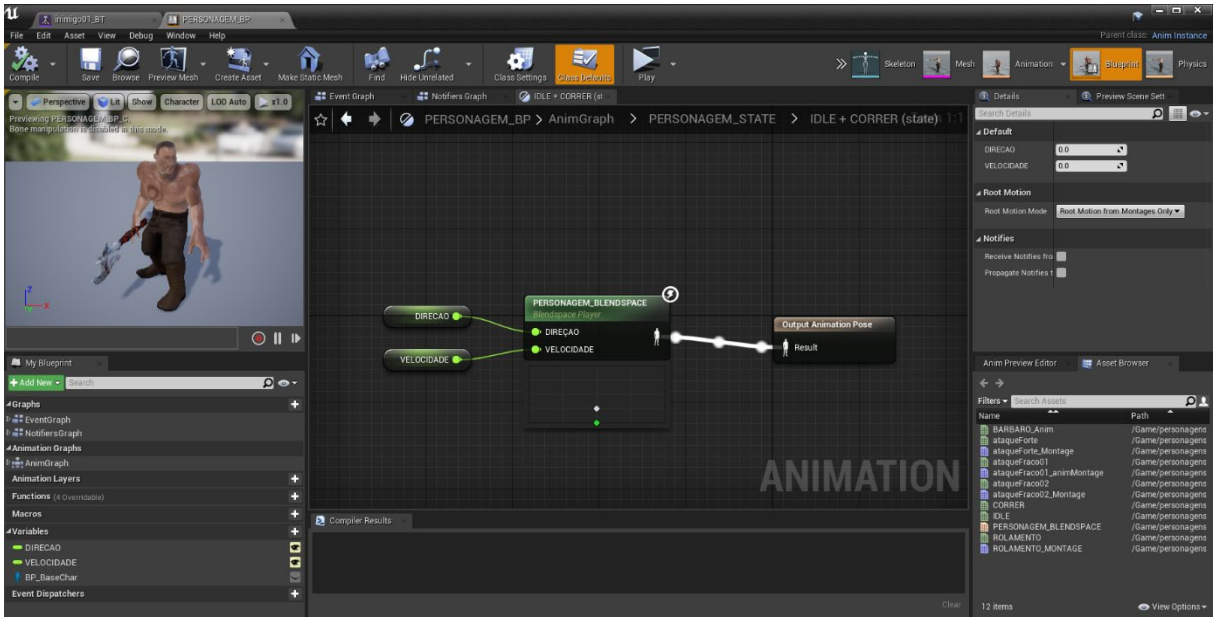
Figura 40 – Configuração da *blendspace*.



Fonte: Elaborado pelos autores.

Para a conexão dessas configurações com nosso personagem fazemos uso das *Animation Blueprint* (Figura 41), que são blueprints específicas para animações, onde importamos as *blendspaces* criadas fazendo a conexão entre eles.

Figura 41 – Inserção da *Animation Blueprint.*



Fonte: Elaborado pelos autores.

Essas funções foram executadas com todos os personagens mudando apenas alguns parâmetros referentes a suas animações como também seus posicionamentos.

4.6.3 Áudio

A imersão do jogador através de sons é uma etapa que para muitos pode passar despercebida. A Unreal oferece várias formas de aplicações sonoras através de suas funções, sendo possível a execução de uma certa faixa de áudio em uma área específica, que se dissipa conforme o jogador se distancia, e também formas básicas onde a faixa é tocada independentemente da posição do jogador, com tempos e volumes definidos.

Neste projeto foram definidas faixas de áudio para certas ocasiões durante o jogo de forma básica, com volume não interferido de acordo com o posicionamento do mesmo.

- *Menu principal*: é executada uma faixa musical com o tema *viking*.
- *Durante o jogo*: é executada uma faixa ritmada com o tema de batalha disponível apenas enquanto o jogador não se encontra na área de batalha do chefe.

- *Ataque:* após o contato do machado com os inimigos uma faixa de impacto é executada.
- *Área do chefe:* após a entrada na área do chefe a faixa de música é trocada para outra com tema mais imponente e agressivo.
- *Tela de fim de jogo:* Assim que o jogo é finalizado a faixa de áudio retorna a mesma do menu principal até que o cliente do jogo seja fechado.

4.6.4 Ataques

4.6.4.1 Personagem

O sistema de ataque implementado para o personagem é um sistema simples (Figura 42), que ao detectar colisão através do machado do personagem principal com o *mesh* dos inimigos gera uma parcela de dano que ao ser ativada executa o inimigo fazendo com que sua animação de morte seja ativada (Figura 43). Após 3 segundos o mesh dele é destruído, assim sendo apagado da visualização no mapa.

Figura 42 – Inimigo golpeado.

Fonte: Elaborado pelos autores.

Figura 43 – Inimigo derrotado.

Fonte: Elaborado pelos autores.

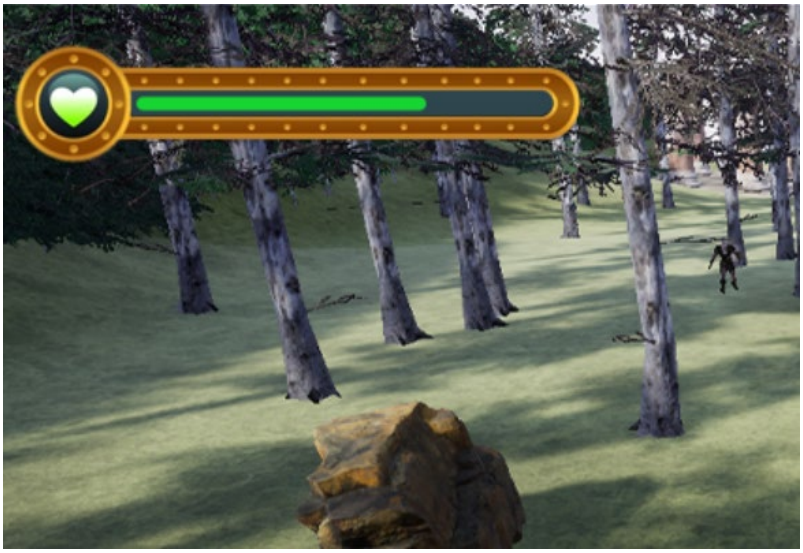
4.6.4.2 Inimigo

Após que a execução da colisão de ataque dos inimigos com o personagem é executada (Figura 44), a vida do mesmo conectada ao HUD é decrementada fazendo com que deles aguardem 3 segundos para o próximo ataque (Figura 45).

Figura 44 – Inimigos atacando.

Fonte: Elaborado pelos autores.

Figura 45 – Vida sendo decrementada no HUD.



Fonte: Elaborado pelos autores.

4.6.5 Defesa

A única ação defensiva do personagem é a habilidade de executar um rolamento que será acionada ao pressionar a tecla espaço no teclado (Figura 46). O rolamento é uma espécie de pulo que o previne de ser acertado pelos ataques inimigos. Enquanto em processo de rolamento o personagem fica imune a ataques.

Figura 46 – Personagem executando a esquiva.

Fonte: Elaborado pelos autores.

4.6.6 Arena do Chefe

Nesse protótipo assim que o personagem adentra na área do chefe a trilha sonora irá ser trocada, e após 10 segundos o jogo irá apresentar a mensagem final. Na Figura 47 a seguir, vemos como é grande a diferença de tamanho entre o *Brute* e o *Narbash*, passando a sensação de ser impossível sobreviver a um ataque.

Figura 47 – Arena do chefe.

Fonte: Elaborado pelos autores.

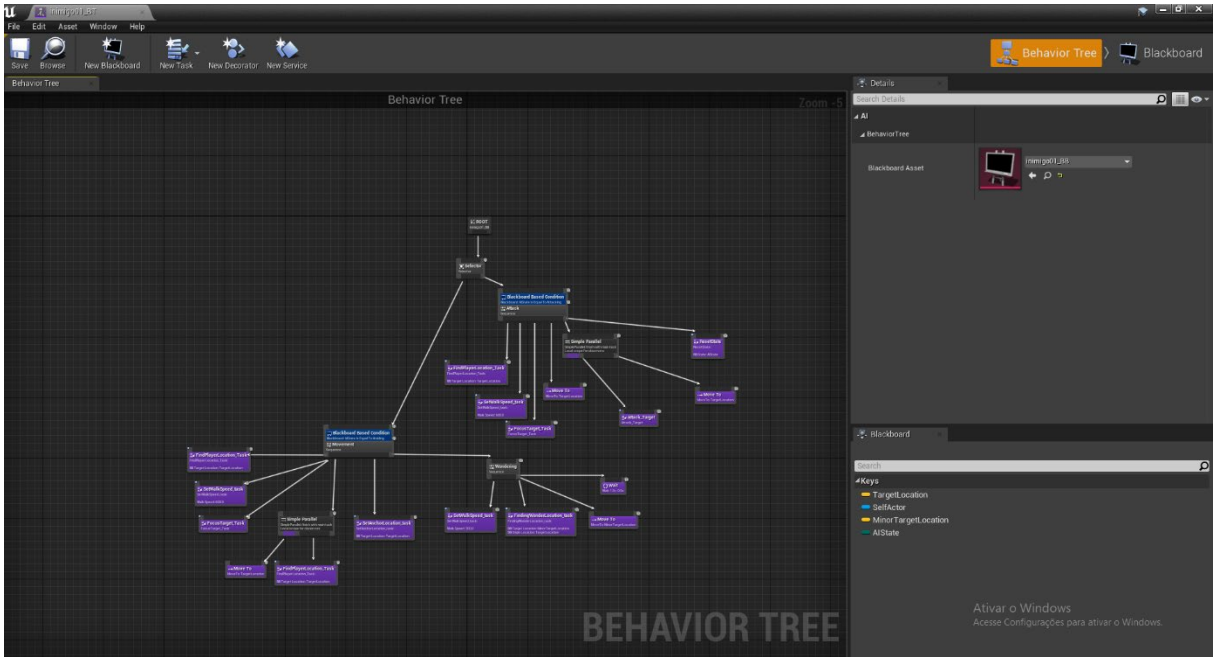
4.6.7 Inteligência Artificial

A inteligência artificial (IA) é um tema muito abordado nos jogos atuais, sendo cada vez mais complexa e surpreendente, causando sempre uma comparação do quão humano uma atividade é. Nesse projeto fizemos uso de uma IA básica e simples, fazendo com que o inimigo detecte o nosso personagem e o persiga diferindo ataques e se movimentando ao redor dele.

Para a criação e configuração de um componente de IA podemos usar as próprias blueprints de uma forma padrão, chamando funções que ao serem executadas geram atividades automáticas e intuitivas. Já no caso da configuração IA dos inimigos, foi usada uma *Behavior Tree* (Figura 48), que é como uma árvore de

comportamento que faz com que cada função seja identificada e excetuada pelo inimigo seguindo um padrão de tarefas pré-definidas.

Figura 48 – Implementação da *Behavior Tree.*



Fonte: Elaborado pelos autores.

4.6.8 Testes e Avaliação

A fase de testes é a parte mais importante de todo o processo de desenvolvimento, sendo necessária atenção e otimização no processo. Nela foram corrigidos bugs de acesso, movimentação, posicionamento e de ações em geral.

O maior problema encontrado foi a busca de informações pertinentes, já que o material disponível pela documentação da Unreal é muito específico, gerando dificuldade na aprendizagem com usuários sem entendimento aprofundando gerando uma certa frustração na hora da implementação de certas funções.

O método de estudo mais utilizado foi a constante consulta de vídeos e tutorias no *Youtube,* gerando um método de desenvolvimento e aprendizagem dinâmicos o que torna o processo em certos momentos mais eficiente, onde por mais que a mesma dificuldade da documentação surja, o desenvolvedor possui uma certa liberdade para executar funções não abordadas nas documentações e tutoriais disponíveis pela Epic Games a respeito da Unreal Engine.

5 CONSIDERAÇÕES FINAIS

Com este trabalho buscamos contextualizar o processo de desenvolvimento de um jogo digital, mais especificamente usando a Unreal Engine.

O nosso foco foi apresentar o processo de criação do jogo, desde a concepção da ideia com *Game Design Document* (GDD), a animação dos personagens e também sua implementação usando as Blueprints.

Dessa maneira, para alcançar esses objetivos, algumas etapas se fizeram necessárias, entre elas, uma pesquisa detalhada sobre a Unreal Engine e todos os processos que compreendem o desenvolvimento de um jogo.

Para uma melhor divisão do trabalho e maior agilidade durante o processo de construção do jogo, adotamos o conceito da metodologia ágil para o projeto, através do framework Scrum, o que possibilitou a entrega de cada etapa no prazo estabelecido pelo cronograma de execução.

Inicialmente foi criado o GDD, onde foi exposto a proposta do jogo, a história, cenário, gameplay, e todas as informações necessárias para dar início ao desenvolvimento.

Já com toda a documentação pronta, uma primeira versão do protótipo do jogo foi implementada, a partir dessa versão muitas alterações se fizeram necessárias, tanto na estrutura, como também na história e cenários que compõem o jogo.

Por fim, a proposta deste trabalho de mostrar o processo de desenvolvimento de um jogo digital, desde a criação do GDD, a animação dos personagens e implementação foi cumprida. Assim podemos concluir que no processo de criação de um jogo é importante que as etapas sejam respeitadas de maneira correta, pois como uma complementa a outra se apenas uma apresentar erros todo o projeto é prejudicado, assim um ambiente de trabalho organizado torna o desenvolvimento fluido e organizado.

Este trabalho trouxe de maneira dinâmica e intuitiva conhecimento sobre a implementação de jogos digitais por meio da Unreal Engine, mostrando que o desenvolvimento de jogos está se tornando mais e acessível a profissionais e entusiastas.

5.1 SUGESTÕES PARA TRABALHOS FUTUROS

No decorrer deste projeto, algumas melhorias e ajustes se mostraram necessários, mas estavam fora do contexto ou do tempo hábil para serem desenvolvidos.

Dessa forma serão apresentados aqui como sugestões para trabalhos futuros:

- Inserção de outras fases, cada uma com um chefe final.
- Implementação de uma versão para dispositivos móveis.

REFERÊNCIAS

ARRUDA, Eucidio Pimenta. **Fundamentos para o desenvolvimento de jogos digitais: eixo informação e comunicação.** Porto Alegre: Bookman, 2014.

ADOBE. **Mixamo**. 2020. Disponível em: https://www.mixamo.com/#/?page=1&type=Character. Acesso em: 18 nov. 2020.

CARVALHO, Carlos Eduardo Costa de et al. **Métodos Ágeis de Desenvolvimento de Software: um caso prático de aplicação do scrum**. 2011. Disponível em: http://www.abepro.org.br/biblioteca/enegep2011_tn_sto_135_861_18838.pdf. Acesso em: 16 jun. 2020.

EPIC GAMES. **Unreal Engine 4 Documentation**. [S. l.], 28 set. 2020. Disponível em: https://docs.unrealengine.com/en-US/index.html. Acesso em: 25 jun. 2020.

EBERLY, David H. **3D Game Engine Architecture:** Engineering Real-Time Applications with Wild Magic. San Francisco: Elsevier, 2005. Disponível em: http://read.pudn.com/downloads151/ebook/655333/3DGameEngineArchitecture.pdf. Acesso em: 17 jun. 2020.

HUIZINGA, Johan. **Homo Ludens: o jogo como elemento cultural.** 2. ed. São Paulo: Perspectiva, 1980. Tradução: João Paulo Monteiro.

MENARD, Michelle; WAGSTAFF, Bryan. **Game development with Unity.** Boston: Cengage Learning, 2015.

MALLMANN, Eduardo Rafael. **Estudo e Desenvolvimento de um Jogo Utilizando Unreal Development Kit.** 2012. 67 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade Regional do Noroeste do Estado do Rio Grande do Sul, Santa Rosa, 2012. Disponível em: https://bibliodigital.unijui.edu.br:8443/xmlui/bitstream/handle/123456789/1368/TCC%202012%20Eduardo%20R%20Mallmann.pdf?sequence=1&isAllowed=y. Acesso em: 21 jun. 2020.

OLIVEIRA, Welliton**. O que é scrum? Conceito, definições e etapas.** [S. l.], 5 fev. 2019. Disponível em: https://evolvemvp.com/o-que-e-scrum-conceito-definicoes-e-etapas/. Acesso em: 23 jun. 2020.

PEREIRA, Lúcio; DA SILVA Michel. **Android Para Desenvolvedores**. Rio de Janeiro: Brasport, 2009.

ROHDE, Michael. **GameMaker:** Studio For Dummies. Hoboken: John Wiley & Sons, 2014.

REIS, Christian Robottom. **Caracterização de um processo de software para projetos de softwares livres.** 2003. 247 f. Dissertação (Mestrado) - Curso de Ciências da Computação e Matemática Computacional, Instituto de Ciências Matemáticas e Computação, São Carlos, 2003.

SCIRRA. **Construct 3 Game Making Software**. 2020. Disponível em: https://www.construct.net/en?utm_source=scirra&utm_medium=topbar&utm_campaign=topbar&utm_content=c3link. Acesso em: 15 jun. 2020.

SUBAGIO, Aryadi. **Learning Construct 2**. Birmingham: Packt Publishing, 2014.

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

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia do Scrum:** um guia definitivo para o scrum: as regras do jogo. 2017. Disponível em: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Portuguese-Brazilian.pdf. Acesso em: 20 jun. 2020.

UNITY. **Melhorias no Unity 2019.3**. [S. l.], 13 jan. 2020. Disponível em: https://unity3d.com/pt/unity/beta/2019.3. Acesso em: 25 jun. 2020.

VILLAS BÔAS, R. Mercado de Jogos. In: AZEVEDO, Eduardo (org). **Desenvolvimento de jogos 3D e aplicações em realidade virtual.** Rio de Janeiro, Elsevier, 2005.