UNIVERSIDADE FEDERAL DO SUL E SUDESTE DO PARÁ
INSTITUTO DE GEOCIÊNCIAS E ENGENHARIAS
FACULDADE DE ENGENHARIA DA COMPUTAÇÃO
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

Projeto Final De Curso II

DESENVOLVIMENTO DE UM JOGO 3D DE AÇÃO NA UNREAL ENGINE 5

Messias Barros Silva

Marabá - PA

2025

UNIVERSIDADE FEDERAL DO SUL E SUDESTE DO PARÁ
INSTITUTO DE GEOCIÊNCIAS E ENGENHARIAS
FACULDADE DE ENGENHARIA DA COMPUTAÇÃO

Messias Barros Silva

DESENVOLVIMENTO DE UM JOGO 3D DE AÇÃO NA UNREAL ENGINE 5

Projeto Final de Curso II, apresentado à Universidade Federal do Sul e Sudeste do Pará, como critério de avaliação e requisitos necessários para obtenção do Título de Bacharel em Engenharia da Computação.

Orientador:
Prof. Dr. Manoel Ribeiro Filho.

Marabá - PA

2025

Dados Internacionais de Catalogação na Publicação (CIP)
Universidade Federal do Sul e Sudeste do Pará
Centro de Biblioteca Universitária

S586d Silva, Messias Barros
Desenvolvimento de um Jogo 3D de Ação na Unreal Engine 5 / Messias Barros Silva. – 2025.

Orientador(a): Manoel Ribeiro Filho.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal do Sul e Sudeste do Pará, Campus de Marabá, Instituto de Geociencias e Engenharias, Faculdade de Engenharia da Computacao, Curso de Bacharelado em Engenharia da Computação, Marabá, 2025.

1. Unreal Engine. 2. Nanite. 3. Ia. 4. Node. I. Filho, Manoel Ribeiro, orient. II. Título.

CDD: 22. ed.: 620.7

Gerada automaticamente pelo módulo Ficha Fácil, conforme os dados fornecidos pelos(as) autores(as).

Messias Barros Silva

DESENVOLVIMENTO DE UM JOGO 3D DE AÇÃO NA UNREAL ENGINE 5

Projeto Final de Curso II, apresentado à Universidade Federal do Sul e Sudeste do Pará, como critério de avaliação e requisitos necessários para obtenção do Título de Bacharel em Engenharia da Computação.

Marabá: 13 de Fevereiro de 2025

BANCA QUALIFICADORA:

_____

Prof. Dr Manoel Ribeiro Filho
(Orientador - UNIFESSPA - IGE - FEC)

_____

Prof. ª Dr Leslye Estefania Castro Eras
(Membro da Banca - UNIFESSPA)

_____

Prof. Dr João Victor Costa Carmona
(Membro da Banca - UNIFESSPA)

Marabá - PA

2025

AGRADECIMENTOS

Dedico este trabalho à minha avó, Lucimar, que partiu, mas que com certeza está em um lugar melhor agora. Sou muito grato a ela por todo o carinho e conselhos, e o que fica são as lembranças dos nossos momentos em família.

Quero agradecer especialmente à minha mãe, Sonia Mara, pois, sem o apoio dela, nada disso faria sentido. Sou também grato ao meu irmão, Gabriel, e ao meu padrasto, Almir, que nunca deixaram de me apoiar nesses anos de faculdade.

Gostaria de expressar minha profunda gratidão a cada um dos meus amigos: Lucas Antonio, Arthur, Lucas Leite, Thiago, Henrique e Felipe. Ao longo dos anos de faculdade, vocês foram uma das principais razões para eu continuar no curso. Foram anos repletos de aprendizado, perdas familiares, momentos difíceis, além de muitas vitórias e derrotas. A presença de vocês foi essencial em cada um desses momentos.

Agradeço também à Professora Leslye, que tornou o período final da faculdade ainda mais interessante, motivando-me a pesquisar e desenvolver novas tecnologias, ampliando meus horizontes.

RESUMO

Este trabalho aborda o projeto e a implementação de um jogo desenvolvido na Unreal Engine 5, explorando diversos processos de criação e tecnologias renomadas, como Blueprint para programação, Nanite para otimização gráfica e Quixel para o desenvolvimento de mapas detalhados. Serão apresentados os processos de idealização e criação de um jogo de ação em terceira pessoa passando por etapas como importação e criação de animações, implementação de mecânicas de jogo, programação de inteligência artificial (IA), além da criação de mapas imersivos e menus animados, demonstrando como essas ferramentas e técnicas contribuem para a criação de uma experiência envolvente e de alta qualidade.

Palavras-chave: Unreal Engine, Nanite, IA, Node.

ABSTRACT

This work addresses the design and implementation of a game developed in Unreal Engine 5, exploring various creation processes and renowned technologies such as Blueprint for programming, Nanite for graphic optimization, and Quixel for detailed map development. The ideation and creation processes of a third-person action game will be presented, covering stages such as importation and creation of animations, implementation of game mechanics, artificial intelligence (AI) programming, as well as the creation of immersive maps and animated menus, demonstrating how these tools and techniques contribute to creating an engaging and high-quality experience

Keywords: Unreal Engine, Nanite, AI, Node.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UE5	*Unreal Engine 5*
RPG	*Role-Playing Game*
NPC	*Non-Playable Character*

SUMÁRIO

1- INTRODUÇÃO

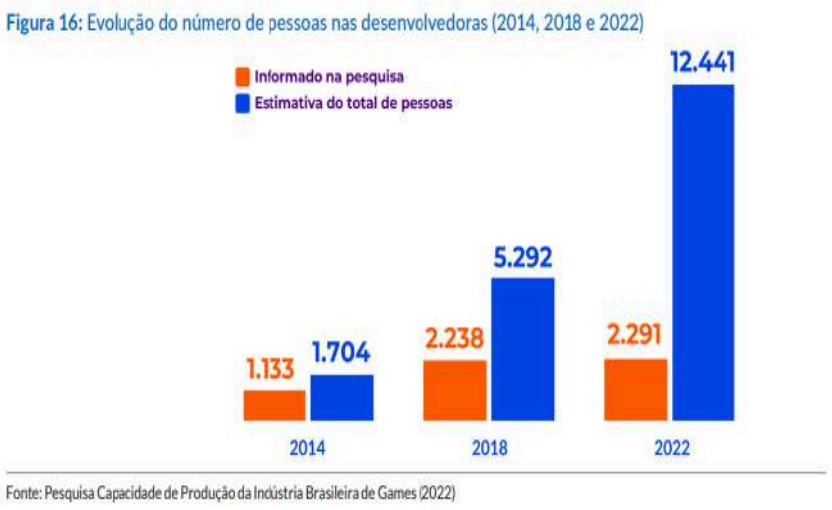
O mercado de videogames evoluiu de um simples hobby infantil para um dos maiores setores do entretenimento mundial. Atualmente, o mercado de games e um setor bilionário, com empresas gigantescas como Microsoft e Sony. Para se ter uma ideia, o mercado de games movimenta mais dinheiro do que o cinema e música (Figura 01).

Figura 01 - Comparação de Receitas

Fonte – MEUPC.NET, 2020

De acordo com Filho e Zambon (2023, online), o Brasil é apenas o décimo maior mercado de games do mundo, movimentando 13 bilhões de reais por ano. Com a expansão do setor, cresce também o interesse pelo desenvolvimento de jogos. Segundo a pesquisa feita no Brasil (Figura 02), há uma estimativa de mais de 12 mil profissionais trabalhando na área, o que mostra um grande avanço no país, já que a média de pessoas trabalhando na área em 2014 era de 8,5 trabalhadores por desenvolvedora, e hoje é de 14. (EBAC, 2024)

Figura 02 - Evolução do Mercado de Desenvolvimento de Jogos



Fonte – EBAC, 2024

Esse fenômeno de aumento de pessoas interessadas pelo desenvolvimento de jogos só foi possível graças à evolução tecnológica das engines. Jogos como DOOM (1994) e Quake (1996) popularizaram o uso de núcleos reaproveitáveis, impulsionando o surgimento das primeiras game engines modernas. Estas engines padronizaram sistemas essenciais, como renderização, física e inteligência artificial, permitindo que os desenvolvedores se concentrassem mais na criatividade do que na complexidade técnica. (Scherer, Batista, Mendes, 2020)

Ao longo da história, surgiram diversas engines que marcaram gerações e foram fundamentais para moldar o mercado de jogos atual. A Unreal Engine (1998) foi a primeira a utilizar cores de 16 bits. Já a Unity (2005) foi e continua sendo a engine mais importante para o mercado indie. (Scherer, Batista, Mendes, 2020)

Graças a todo esse desenvolvimento, hoje há várias opções de engines disponíveis. segundo Rodrigues e Cavalcante (2024, p. 2), “A escolha da engine de jogo é uma decisão crucial que impacta diretamente a eficiência do desenvolvimento, a qualidade do produto final e a experiência do usuário”. Entre as principais ferramentas no mercado, destacam-se a Unreal Engine 5 e a Unity, ambas amplamente utilizadas, mas com características distintas.

A escolha entre uma delas depende de diversos fatores, como gosto pessoal, limitação de hardware, curva de aprendizado e o tipo de conteúdo que se deseja criar. A Unreal Engine 5 possui maiores requisitos de hardware e é conhecida por sua capacidade de criar jogos 3D com gráficos ultra-realistas, graças a tecnologias como Lumen, Nanite, Quixel e MetaHuman por isso e uma engine muito mais escolhida por grandes empresas (Abhishek, Varghese, Thomas, Sajimon, Varghese, 2023).

Já a Unity é uma engine mais balanceada, permitindo o desenvolvimento de jogos 2D e 3D com excelente nível de qualidade em ambos os estilos. Além disso, ela exige menos hardware e possui uma grande comunidade de suporte, o que a torna uma escolha popular para o desenvolvimento indie. (Rodrigues e Cavalcante, 2024).

Neste TCC, será desenvolvido um jogo para demonstrar como funcionam os processos de criação dentro da UE5, passando por várias etapas de criação e utilizando algumas das tecnologias famosas da Unreal.

1.1 Justificativa

A Unreal Engine 5, lançada em abril de 2022, é uma tecnologia relativamente recente. Essa novidade resulta na escassez de trabalhos acadêmicos sobre essa nova versão, especialmente considerando o significativo avanço em relação à versão anterior, Unreal Engine 4. Portanto, torna-se imperativo explorar a Unreal Engine 5 para contribuir com o preenchimento dessa lacuna acadêmica.

Este trabalho, além de mitigar a escassez de estudos sobre a Unreal Engine 5, visa também incentivar novos estudantes interessados em desenvolvimento a considerar essa poderosa ferramenta para a criação de seus jogos ou estudos.

Outro motivo relevante para explorar a nova versão da Unreal Engine reside no fato de que grandes empresas, anteriormente usuárias de suas próprias engines, têm migrado para esta plataforma. Essa mudança facilita a inserção de novos profissionais nessas empresas, uma vez que terão acesso à mesma ferramenta utilizada por grandes estúdios. Exemplos notáveis incluem a CD Projekt Red, que está desenvolvendo The Witcher 4 na Unreal Engine 5, e a Game Science, que recentemente lançou o tão aguardado Black Myth: Wukong. Outras empresas renomadas, como 2K, Remedy, Obsidian Entertainment, Xbox Game Studios e Gunfire Games, também adotaram a Unreal Engine 5, demonstrando a força dessa engine. (Butcher, 2022)

1.2 Objetivo Geral

Desenvolver um jogo 3D de ação em terceira pessoa utilizando a Unreal Engine 5

1.3 Objetivos específicos

- Usar tecnologias da Unreal, como Nanite e Quixel;
- Desenvolver somente usando a programação visual da Unreal conhecida como Blueprint;
- Proporcionar um documento que aborde processos de criação de um jogo básico
- Explorar as configurações da UE5 de maneira didática

2 – TRABALHOS RELACIONADOS

Para a realização deste TCC, também foi realizada uma pesquisa sobre outros trabalhos acadêmicos que utilizaram a Unreal Engine para o desenvolvimento de jogos. A seguir, será feita uma breve análise de cada um desses projetos.

2.1 – Combate por turnos para videojuego RPG en Unreal Engine 5

García (2023) desenvolveu um jogo Role-Playing Game (RPG) com combate por turnos, com o principal objetivo de aprender a Unreal Engine 5 e, no processo, criar um jogo que fosse divertido e com mecânicas satisfatórias.

Um RPG por turnos opera com um sistema onde jogadores e inimigos alternam suas ações em uma ordem estabelecida. Este estilo de jogo é fundamentado na estratégia, permitindo que o jogador decida entre atacar, defender ou usar uma habilidade específica, dependendo da situação. A mecânica desse sistema pode ser visualizada na Figura 03.

Figura 03 - Interface do Jogo

Fonte - MÉNDEZ, 2023

O resultado final foi interessante, mas não utilizou tecnologias como Nanite e Quixel, que poderiam ter elevado a qualidade do projeto. Essas tecnologias serão aplicadas neste trabalho para alcançar um nível ainda mais avançado.

2.2 – Shriek: um jogo de RPG usando Unreal Engine 4 e árvores de comportamento

Rodrigues e Kaur (2021) desenvolveram um RPG utilizando a Unreal Engine 4, integrando árvores de comportamento para gerenciar a inteligência artificial. O jogo apresenta

quatro personagens jogáveis, que podem ser vistos no menu do jogo (Figura 04), e se ambienta em um cenário medieval.

O gênero RPG é fortemente centrado na evolução dos personagens, frequentemente utilizando o que chamamos de árvore de habilidades. À medida que o jogador progride, essa mecânica aumenta a sensação de que o personagem está se tornando mais poderoso.

Figura 04 –Menu do Shriek

Fonte - RODRIGUES e KAUR, 2021

No trabalho de Rodrigues e Kaur, foi utilizada uma versão anterior à Unreal Engine 5, que só seria lançada no ano seguinte. Portanto, as tecnologias introduzidas na nova versão não foram aplicadas. No entanto, o trabalho é altamente relevante para análise, pois as árvores de comportamento, mantidas na nova versão da engine, continuam sendo uma ferramenta poderosa para criar NPCs (personagens não jogáveis) com inteligência aprimorada. Esse estudo, portanto, serviu como uma valiosa oportunidade de aprendizado sobre o tema.

2.3 – Desenvolvimento de jogos com Unreal Engine 4

Hämäläinen (2020) construiu um protótipo de jogo com o objetivo de desenvolver uma mecânica de combate, focando também em animação, personagens e inteligência artificial. Embora o trabalho não tenha sido realizado na Unreal Engine 5, ele oferece valiosos aprendizados, especialmente sobre a importância de uma boa animação para garantir a fluidez do combate.

Figura 05 - 120FPS

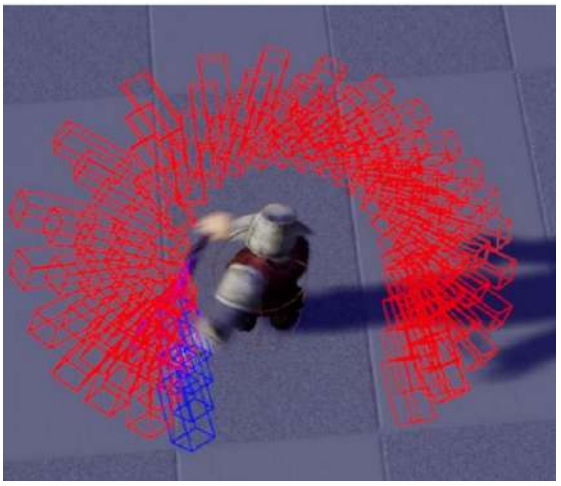
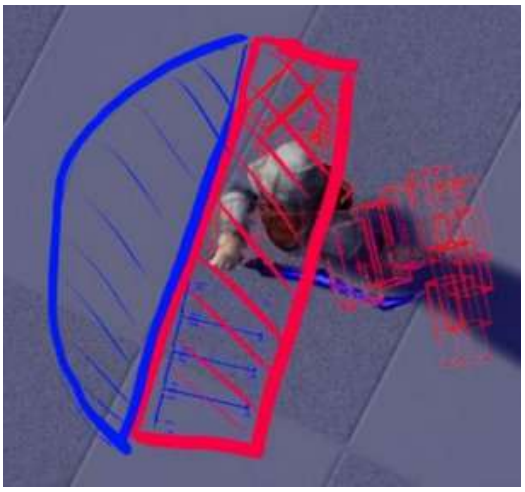


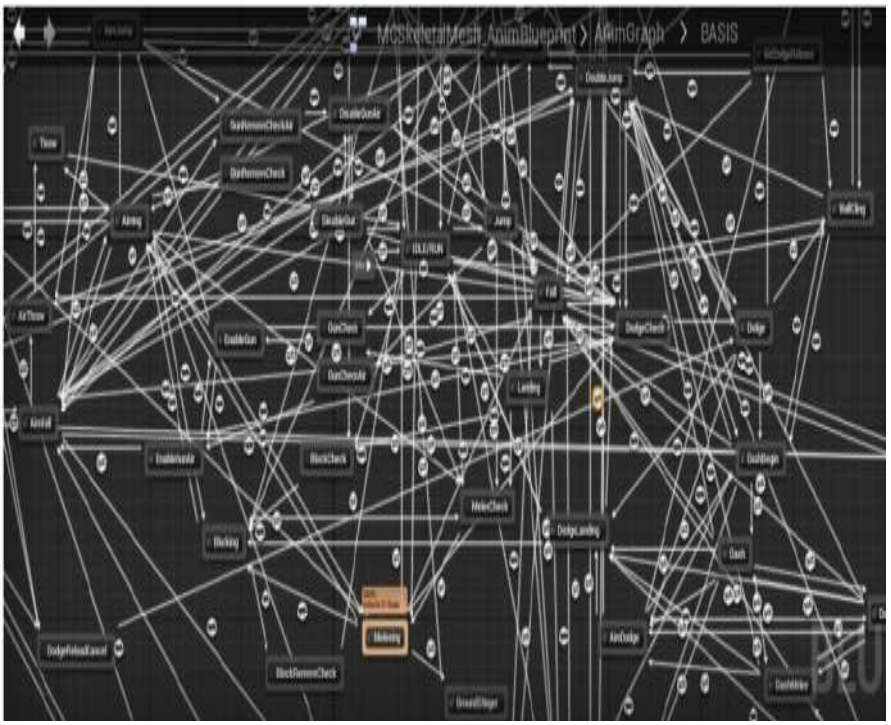
Figura 06 - 12FPS



Fonte - HÄMÄLÄINEN, 2020

As Figuras 05 e 06 ilustram a diferença entre um golpe executado a 120 fps (frames por segundo) e um a 12 fps. O golpe a 120 fps demonstra uma precisão significativamente maior, um aspecto muito importante na criação de uma boa mecânica de jogo.

Figura 07 - Código Blueprint Mal Estruturado



Fonte - HÄMÄLÄINEN, 2020

Outro aspecto crucial abordado no trabalho em análise é a organização do código. Embora a programação visual seja uma tecnologia extremamente útil e facilitadora, especialmente para prototipagem, é igualmente importante manter o código bem estruturado. Sem uma organização adequada, é fácil perder o controle do projeto, como ilustrado na Figura 07. A boa organização do código, seja em programação visual ou tradicional, é fundamental para garantir a clareza e a gestão eficaz do desenvolvimento.

2.4 – Desenvolvimento de um jogo de RPG utilizando Unreal Engine 5

Alexandrov (2024) fez uma análise sobre o gênero RPG para fazer um jogo (Figura 08) utilizando a Unreal Engine 5, explorando tecnologias avançadas como Lumen para iluminação global em tempo real e Nanite para geometria virtualizada de micropolígonos. O jogo apresenta personagens detalhados e um mundo imersivo, com gráficos de alta qualidade e jogabilidade envolvente

Figura 08 - Ambientação

Fonte - ALEXANDROV, 2024

O jogo desenvolvido por Alexandrov inclui um menu de seleção de personagens, onde os jogadores podem escolher entre diferentes heróis, cada um com habilidades únicas. A implementação de inimigos com comportamentos variados e animações detalhadas adiciona um nível de desafio e estratégia ao jogo, tornando a experiência ainda mais rica e envolvente.

O trabalho de Alexandrov, assim como outros trabalhos analisados neste capítulo, tem um foco significativo no RPG. No entanto, este TCC seguirá uma abordagem diferente, pois será desenvolvido um jogo mais geral, com ênfase maior nos processos de criação e menor no gênero.

3 – METODOLOGIA

Neste capítulo, serão abordadas as ferramentas utilizadas, as etapas de criação e os requisitos para utilizar a Unreal Engine 5, bem como os requisitos de hardware usados para o desenvolvimento do jogo proposto neste TCC.

3.1 – Ferramentas de Desenvolvimento

As ferramentas apresentadas nesta seção serão utilizadas para a criação do jogo e para obter os assets (ativos). Todas as ferramentas selecionadas para este trabalho foram escolhidas por serem úteis no desenvolvimento e, ao mesmo tempo, gratuitas.

- **Unreal Engine 5:** A Unreal Engine é um motor gráfico desenvolvido pela Epic Games, lançado inicialmente em 1998 para o jogo Unreal Tournament. Desde então, a engine passou por várias atualizações significativas. No entanto, ela ganhou ainda mais destaque com o anúncio da versão 5, lançada em 5 de abril de 2022, que introduziu diversas novas tecnologias, como Lumen e Nanite. (Unreal Engine, 2022)

- **Quixel Bridge:** Em novembro de 2019, a Epic Games anunciou a aquisição da Quixel, responsável pelo Quixel Bridge e pela biblioteca Megascans. O Megascans é um repositório onde desenvolvedores e artistas podem acessar uma vasta coleção de ativos fotogramétricos 2D e 3D. Com a aquisição, a Epic Games integrou o Megascans à Unreal Engine 4, assim como fez com o Metahuman. (Dias, 2019, Online)

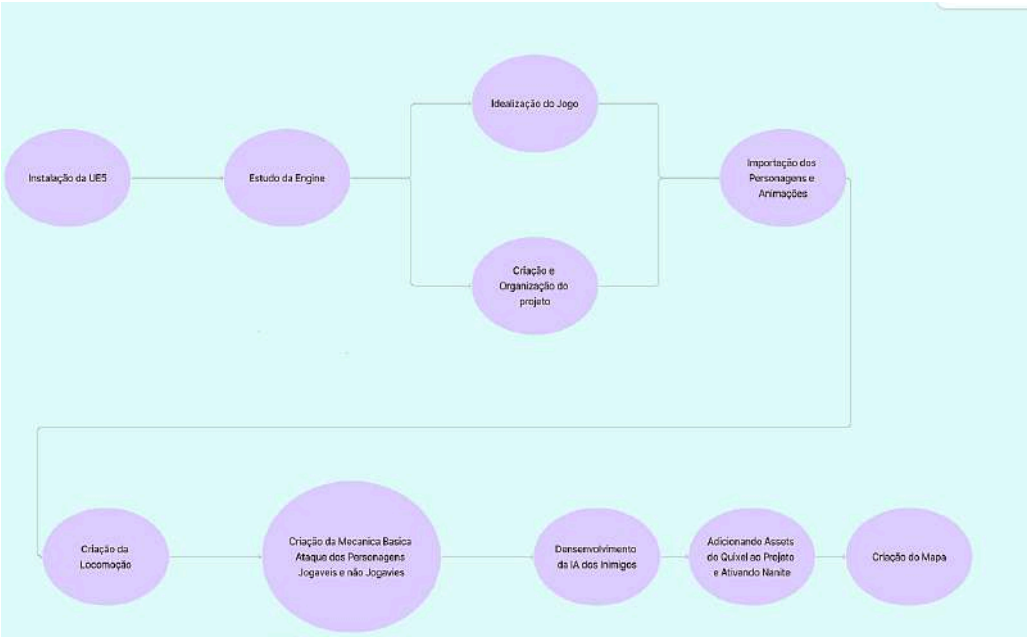
- **Mixamo:** O Mixamo é um serviço online adquirido pela Adobe em 2015. Ele oferece assets de personagens, animações e funcionalidades como rigging automático. Além disso, o Mixamo foi integrado ao Photoshop, permitindo que os usuários criem, animem e renderizem personagens diretamente no Photoshop. O serviço também é amplamente utilizado em jogos, design e experiências de realidade aumentada. (GarageFarm.Net, Online)

3.2 – Etapas de Desenvolvimento

Primeiramente, foi realizada a instalação e um estudo sobre a Unreal Engine 5 (UE5) para familiarização com a plataforma e seus processos de criação, além da linguagem Blueprint. Em seguida, foi idealizado o projeto a ser desenvolvido, definindo a quantidade de inimigos, os mapas e o gênero do jogo. Paralelamente, foi feita a criação e organização do projeto.

Após a conclusão dos downloads e importação dos personagens na engine, foram desenvolvidas a locomoção e toda a mecânica do jogo, incluindo a criação do ataque, sistema de dano e inteligência artificial (IA) dos inimigos. Finalmente, foi realizada a criação do mapa do jogo (Figura 09).

Figura 09 - Fluxo das Etapas de Desenvolvimento



Fonte - Autor

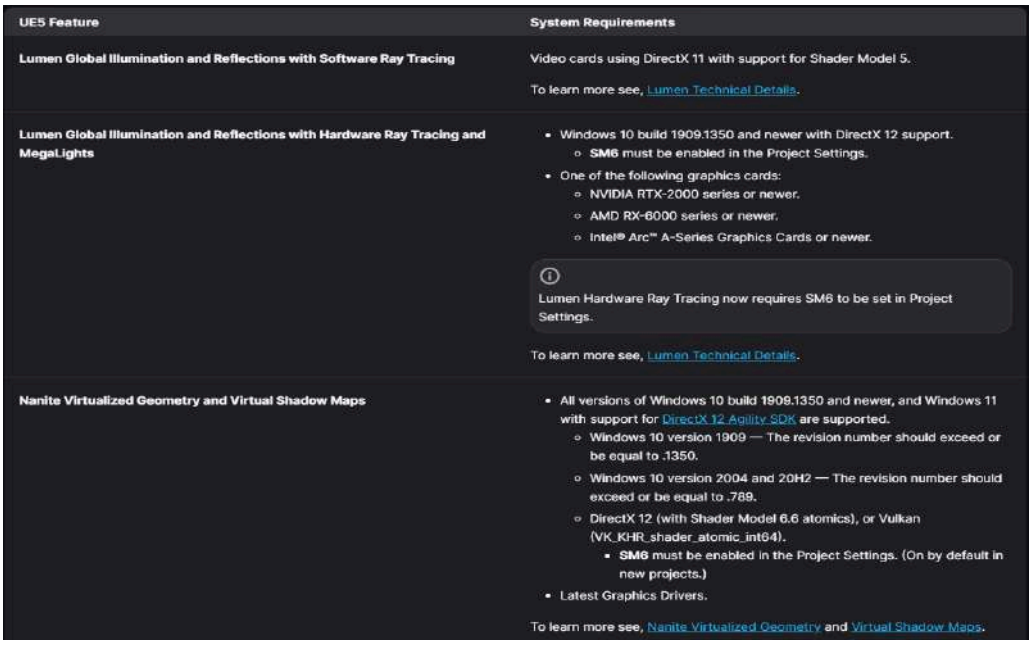
3.3 – Requisitos de Hardware

A Unreal Engine 5, por si só, já exige altos requisitos de hardware para o desenvolvimento de projetos nela (Figura 10). Além disso, algumas tecnologias específicas, como o Lumen e Nanite, possuem requisitos adicionais para serem utilizadas (Figuras 11). Neste TCC, foi utilizada a versão 5.5 da UE5.

Figura 10 - Hardware Recomendado

Fonte - Epic Games

Figura 11 - Requisitos do Lumen e Nanite



Fonte - Epic Games

Para o desenvolvimento do jogo proposto neste trabalho de conclusão de curso, foi utilizada a seguinte configuração de hardware:

- Processador: AMD Ryzen 5 5600 com 3.50 GHz
- Memória RAM: 16 GB
- Sistema Operacional: Windows 11

- Placa de Vídeo: Nvidia GeForce RTX 3060

A configuração descrita acima corresponde aos requisitos de base para jogar o jogo, uma vez que não foram realizados testes em outros computadores para determinar requisitos mínimos. Portanto, para rodar o jogo de forma satisfatória, é necessário utilizar uma configuração muito próxima ou superior à mencionada anteriormente.

4 – PROJETO E IMPLEMENTAÇÃO

Apresenta-se neste capítulo o conceito do jogo que será criado e suas várias etapas de desenvolvimento.

4.1 – Ideia do Jogo

O jogo apresenta uma premissa clara e objetiva: trata-se de um jogo de ação em que o jogador utiliza uma espada para neutralizar seus adversários. A estrutura do jogo é linear, com uma perspectiva de câmera em terceira pessoa e gráficos tridimensionais (3D). Adicionalmente, o jogo contará com uma ambientação sombria, complementada por efeitos sonoros e trilha sonora, conforme ilustrado na Figura 07.

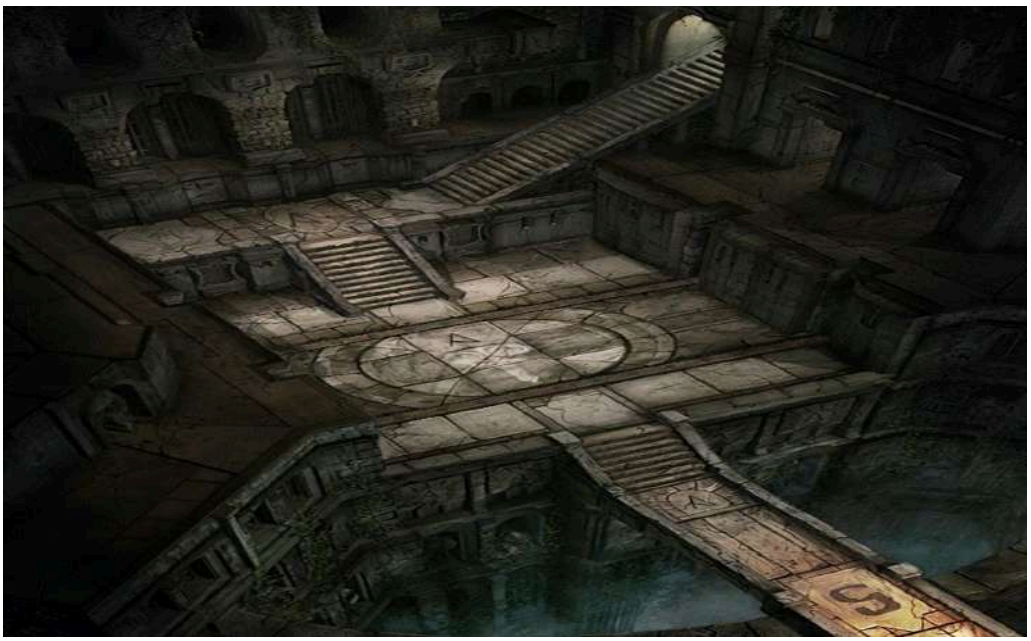
O principal objetivo do jogo é derrotar o chefe do mapa. Para alcançar esse objetivo, o jogador deve primeiro enfrentar e eliminar inimigos comuns. Existem dois tipos distintos de inimigos: o chefe, que possui três tipos diferentes de ataques, e os inimigos comuns, que possuem apenas um tipo de ataque. O chefe é caracterizado por uma maior quantidade de pontos de vida e maior dano em comparação aos inimigos comuns.

Quadro 1 - Informações Gerais do Jogo

Informações Gerais do Jogo
Nome do Jogo: Darkness
Ambientação: Sombria
Mapa: Linear
Quantidade de Fases: 1
Câmera: Terceira Pessoa
Estilo do gráfico: 3D
Tipos de inimigos: Chefe e Inimigo Comum
Gênero: Ação

Fonte - Autor

Figura 12 - Referência para o mapa do jogo



Fonte - Santana, Pinterest

No jogo, os comandos serão configurados da seguinte maneira:

- **W, A, S, D:** movimentação do personagem para frente, direita, esquerda e para trás.
- **Mouse:** movimenta a câmera.
- **Botão do meio do mouse:** trava a mira no inimigo.
- **Botão esquerdo do mouse:** realiza ataques.
- **Barra de espaço:** executa uma manobra de rolamento, ideal para esquivar dos ataques inimigos, considerando que o jogo não possui um sistema de defesa.

Quadro 2 - Comando do Jogo

Comandos do Jogo
WASD para Locomoção
Mouse Move a Camera
Botão do Meio do Mouse Trava a Mira
Botão Esquerdo do Mouse Ataca
Barra de Espaço Usa Rolamento
Esc Pausa o Jogo e Abre Configurações

Fonte - Autor

4.2 – Importação de Personagens e Animações

Modelos de personagens e animações foram baixados da plataforma Mixamo e importados para a engine de desenvolvimento do jogo. Dois desses modelos foram designados como inimigos, enquanto um terceiro modelo foi atribuído como personagem jogável. As representações visuais dos inimigos podem ser vistas nas Figuras 13 e 14, enquanto o personagem principal controlado pelo jogador está ilustrado na Figura 15.

Figura 13 - Chefe

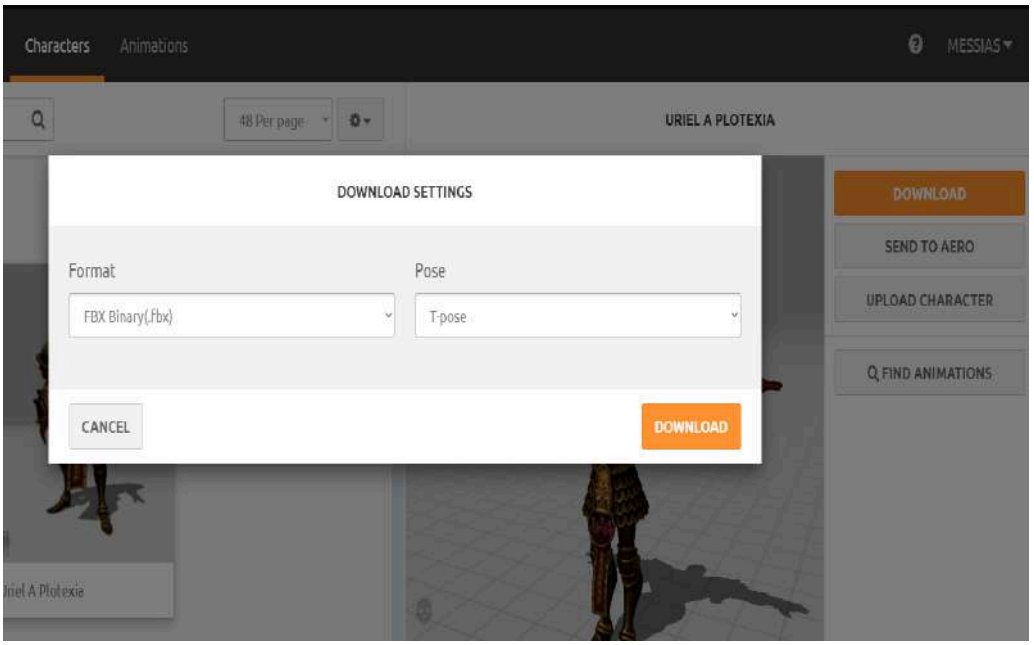
Figura 14 - Inimigo Comum

Figura 15 - Personagem Principal

Fonte - Personagens do Mixamo, fotos tiradas pelo autor

O procedimento para o download de personagens e animações do Mixamo é claro e eficiente. Primeiramente, seleciona-se o personagem desejado e efetua-se o download do modelo na posição T-Pose, no formato FBX (Figura 16).

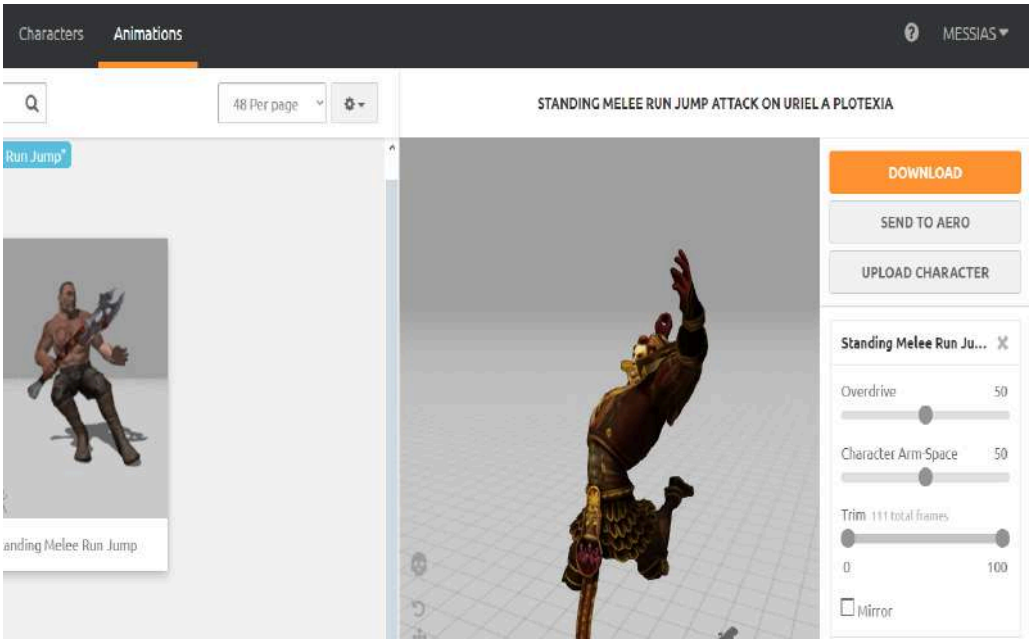
Figura 16 - Download T-pose



Fonte - Personagem do Mixamo, foto tirada pelo autor

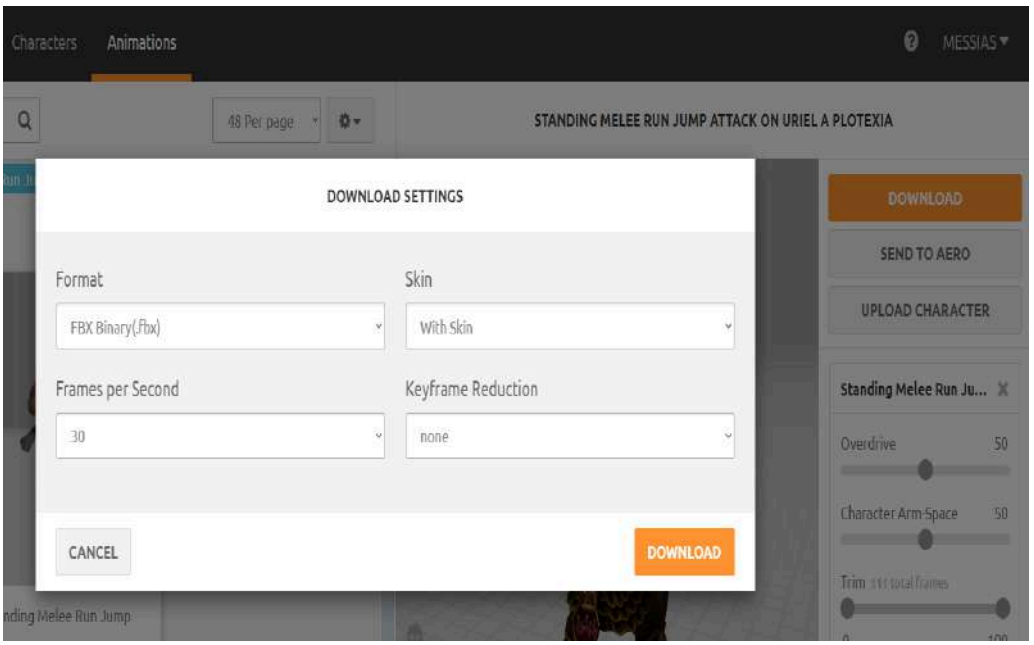
Posteriormente, acessa-se a aba de animações e selecionam-se as animações desejadas, que serão automaticamente aplicadas ao personagem previamente escolhido. Ao clicar em download, deve-se escolher a taxa de quadros por segundo (FPS) da animação e manter a opção "With Skin" marcada para baixar a animação com a skin selecionada, conforme ilustrado nas Figuras 17 e 18.

Figura 17 - Animação de Ataque



Fonte - Animação do Mixamo, foto tirada pelo autor

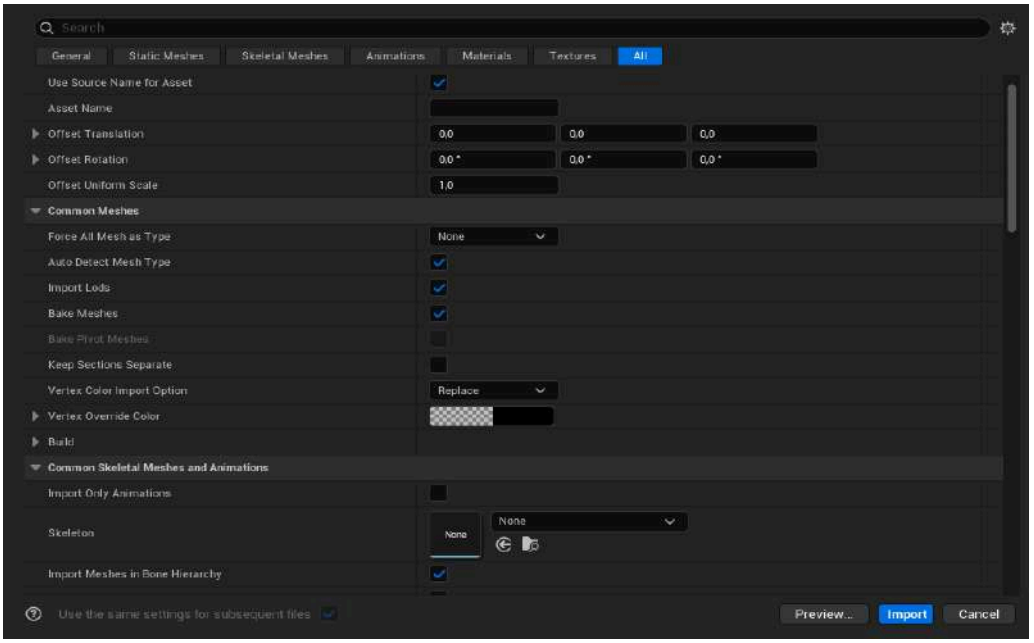
Figura 18 - Download da Animação de Ataque



Fonte - Animação do Mixamo, foto tirada pelo autor

Após o download dos personagens e animações, deve-se importar todos os elementos para a Unreal Engine 5. O processo inicia-se com a importação do modelo do personagem em T-pose (Figura 19). Quando o modelo é importado, ele é composto por cinco partes (Figura 20), sendo uma delas o esqueleto (skeleton), que é necessário para vincular as animações a esse esqueleto específico durante a importação das animações.

Figura 19 - Importação T-pose Chefe



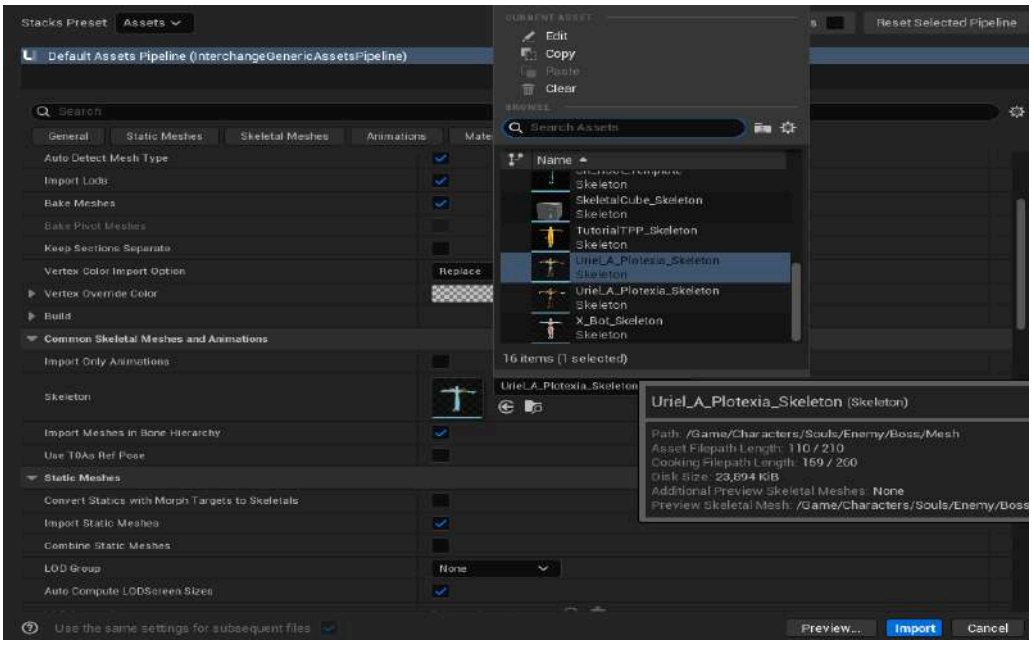
Fonte - Autor

Figura 20 - Modelo do Chefe Importado

Fonte - Autor

Em seguida, importam-se as animações, garantindo que sejam vinculadas ao esqueleto específico do personagem correspondente, conforme mostrado na Figura 21. Caso contrário, ocorrerão erros. Por exemplo, tentar importar as animações do inimigo comum para o esqueleto do chefe resultará em falhas.

Figura 21 - Importação das Animações do Chefe



Fonte - Autor

4.3 – Criação da Mecânica de Jogo

Na criação da mecânica, vamos abordar os seguintes tópicos:

- Criação da locomoção
- Sistema de ataque e de aplicação de dano
- IA dos inimigos

4.3.1 – Criação da Animação de Locomoção

Quando importamos as animações, elas vêm separadas. Portanto, é necessário uni-las para criar uma animação de locomoção, na qual o personagem precisa se mover para frente, para a direita, para a esquerda e para trás. Ao criar uma animação de locomoção, é imprescindível que a animação seja acompanhada pelo esqueleto. Para isso, é necessário ativar o root motion.

Figura 22 - Funcionamento do Root Motion Parte 1

Fonte - Documentação da Unreal

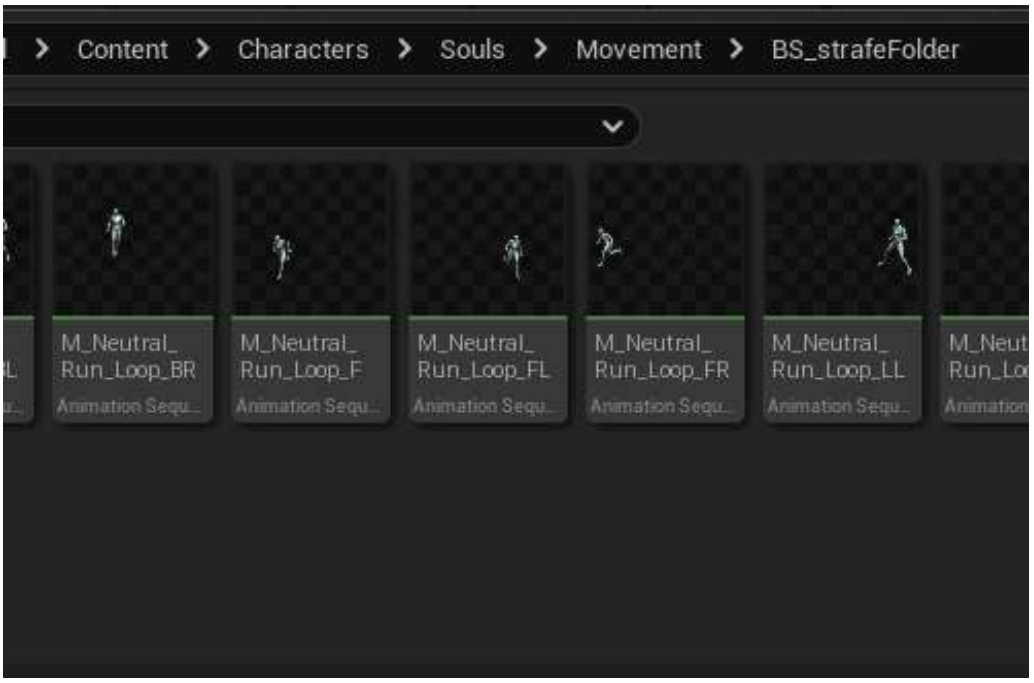
Na Figura 23, observa-se uma comparação: do lado esquerdo, o root motion está desativado, e do lado direito, ele está ativado. A cápsula que envolve os personagens e o osso raiz, quando a animação é de movimento, necessita que o root motion esteja ativado para que o osso raiz acompanhe a animação. Caso contrário, por exemplo, em uma animação de pulo sem root motion, a animação de pulo será executada, mas, ao final dela, o personagem "teleportará" para trás novamente. Isso ocorre porque o osso não acompanha a animação. Na Figura 23, vemos do lado esquerdo sem o root motion ativado e, do lado direito, com o root motion ativado. Podemos notar que, do lado esquerdo, a cápsula fica para trás.

Figura 23 - Funcionamento do Root Motion Parte 2

Fonte - Documentação da Unreal

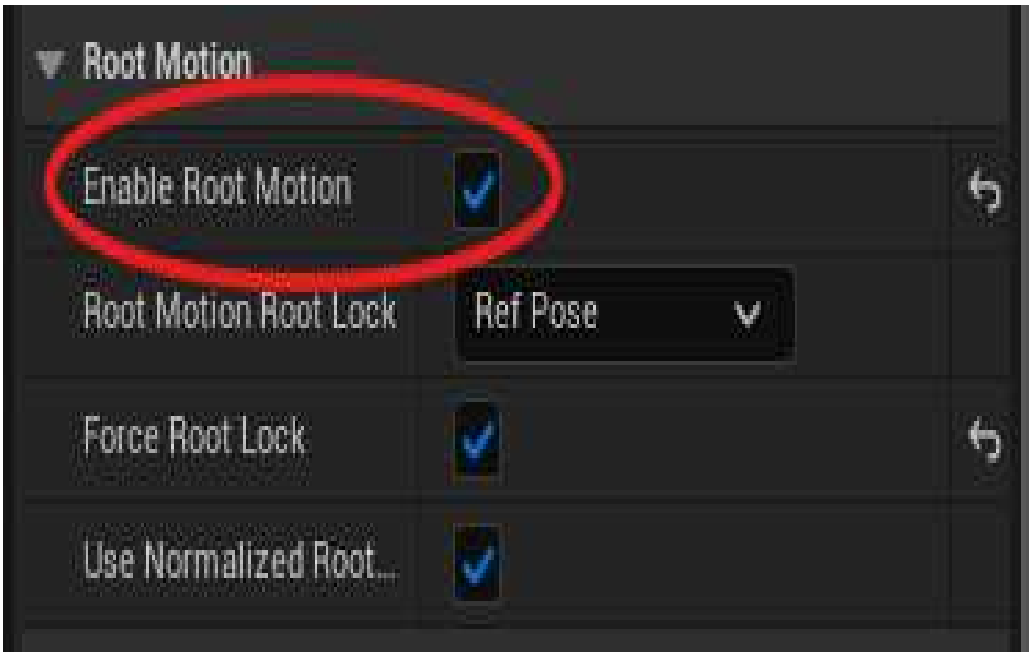
O root motion será habilitado em todas as animações mostradas na Figura 24. Para isso, é necessário clicar na animação para abrir o painel de configuração. Em seguida, ativa-se a opção root motion, conforme ilustrado na Figura 25.

Figura 24 - Animações de Locomoção



Fonte - Autor

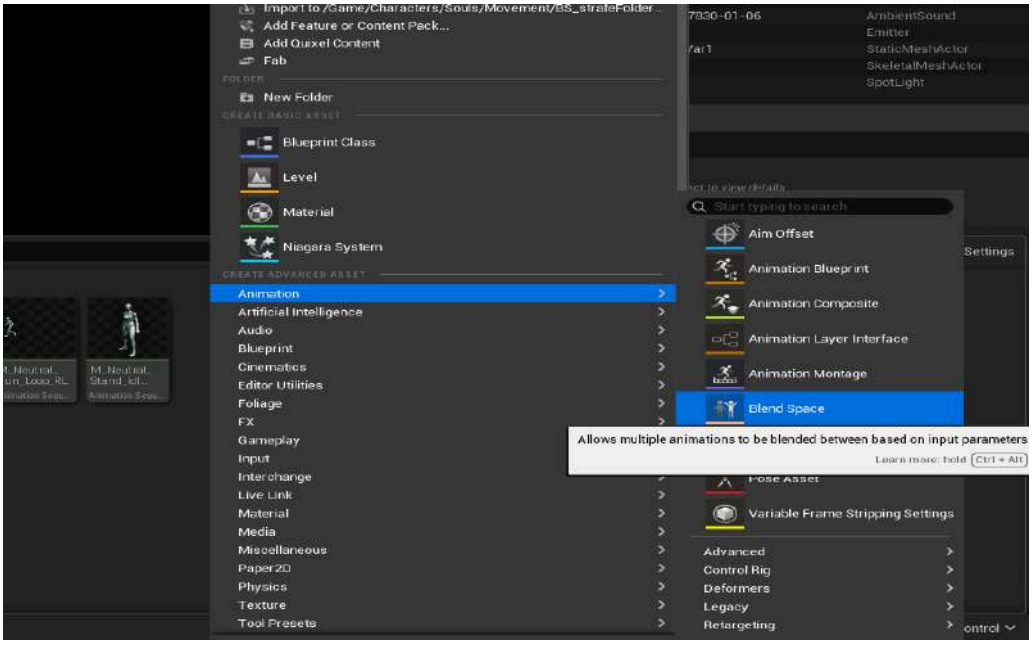
Figura 25 - Ativando o Root Motion



Fonte - Autor

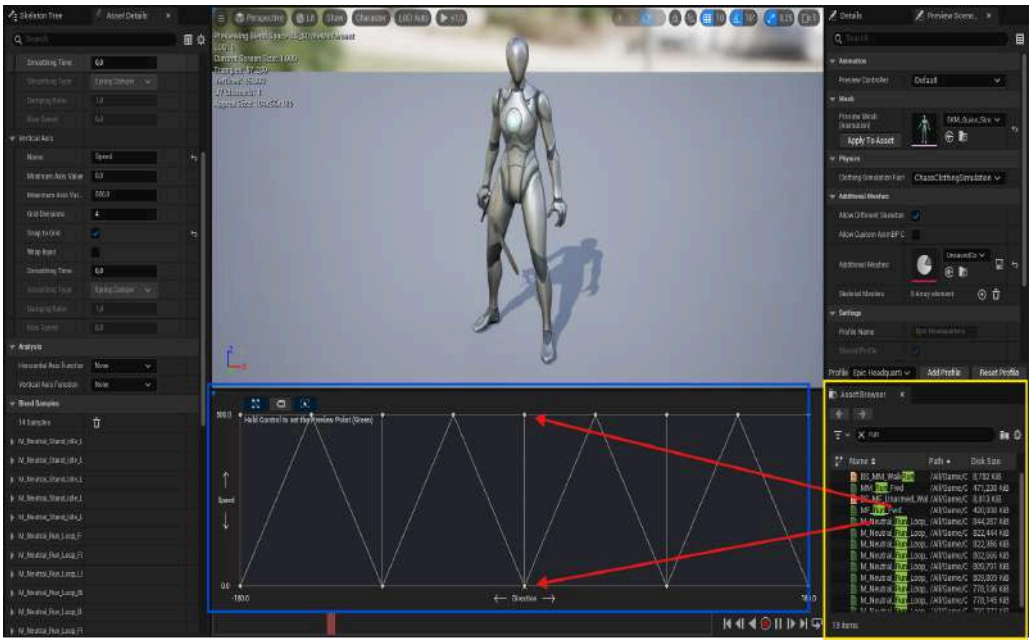
Após ativar o "root motion", é possível criar a animação de locomoção. Para isso, é necessário unir todas as animações que vieram separadas em um único conjunto, criando um "blend space", conforme ilustrado na Figura 26.

Figura 26 - Criando Blend Space



Fonte - Autor

Figura 27 - Criando Animação de Locomoção



Fonte - Autor

Na Figura 27, observa-se a interface de criação de uma animação. Na área destacada em amarelo, estão dispostas as animações que podem ser selecionadas e arrastadas para a área destacada em azul, onde a animação será efetivamente criada. Existem diversas formas de organizar as animações, variando conforme o desenvolvedor. No presente trabalho, a organização adotada foi a seguinte: na parte inferior, foi posicionada a animação de idle, correspondente à situação em que o personagem está parado. Na parte superior, foram inseridas as animações de correr para frente, para a direita, para a esquerda e para trás.

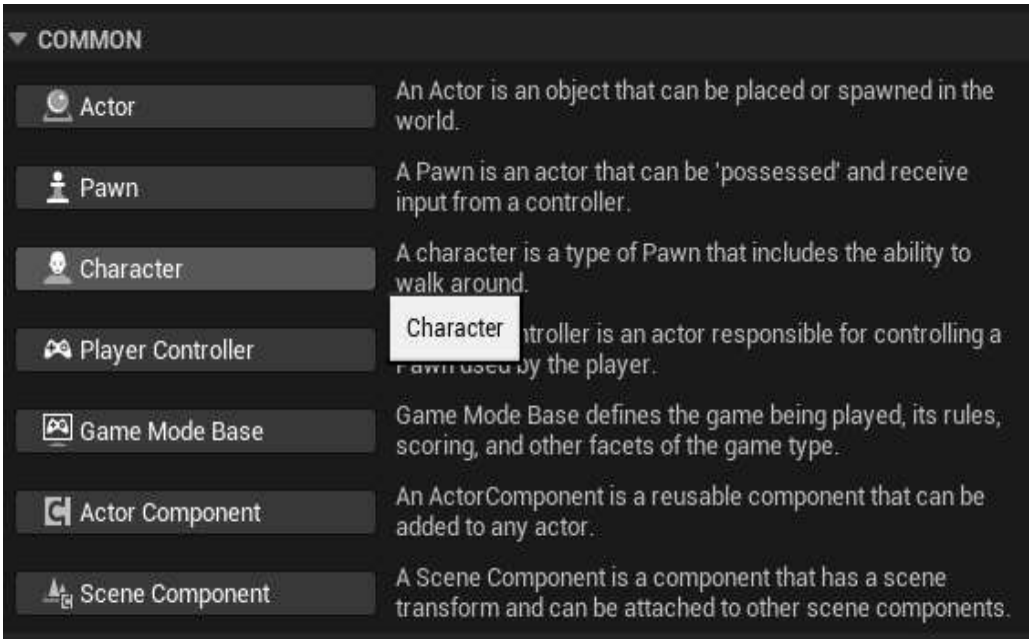
4.3.2 – Sistema de Ataque e de Aplicação de Dano

A Unreal possui uma tecnologia de programação visual denominada Blueprint. Com essa ferramenta, é possível desenvolver um jogo completo sem a necessidade de escrever uma única linha de código em C++, que é a linguagem utilizada na engine. Para implementar o sistema de ataque, utilizaremos dois tipos distintos de Blueprints.

O primeiro é o Blueprint Class Character, empregado para programar o personagem, seja ele jogável ou não. É nesse Blueprint que serão inseridos o modelo, os inputs, entre outros elementos. No entanto, é recomendável dividir certas funções para evitar que o código dentro do Blueprint do personagem se torne excessivamente extenso. Para isso, utilizaremos o

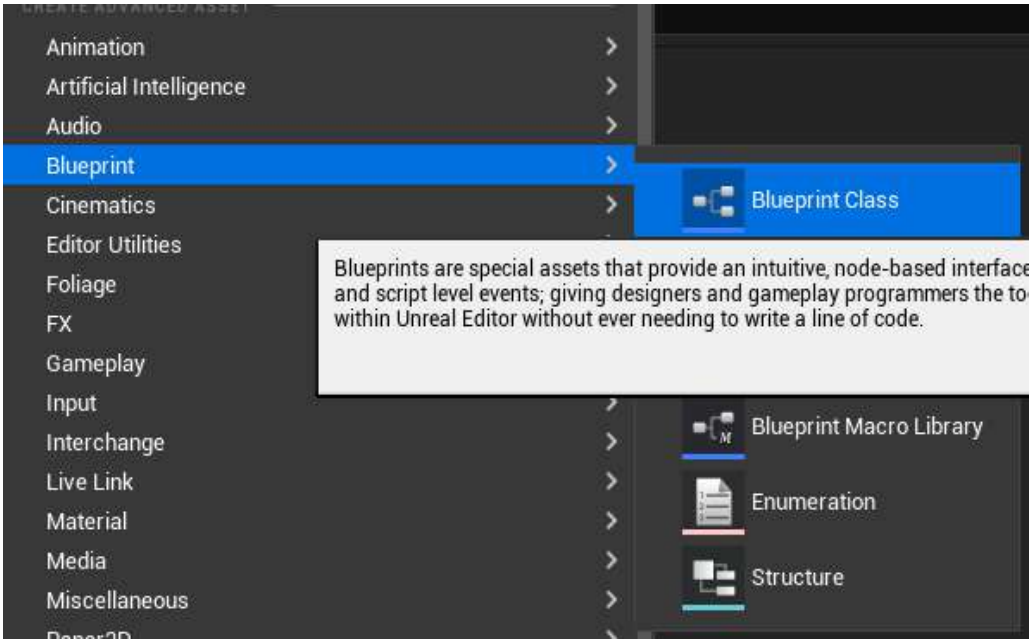
Blueprint Class, que conterá a lógica de ataque. As etapas de criação de ambos os Blueprints são ilustradas nas Figuras 28 e 29.

Figura 28 - Criando Blueprint Class Character



Fonte – Autor

Figura 29 - Criando Blueprint Class

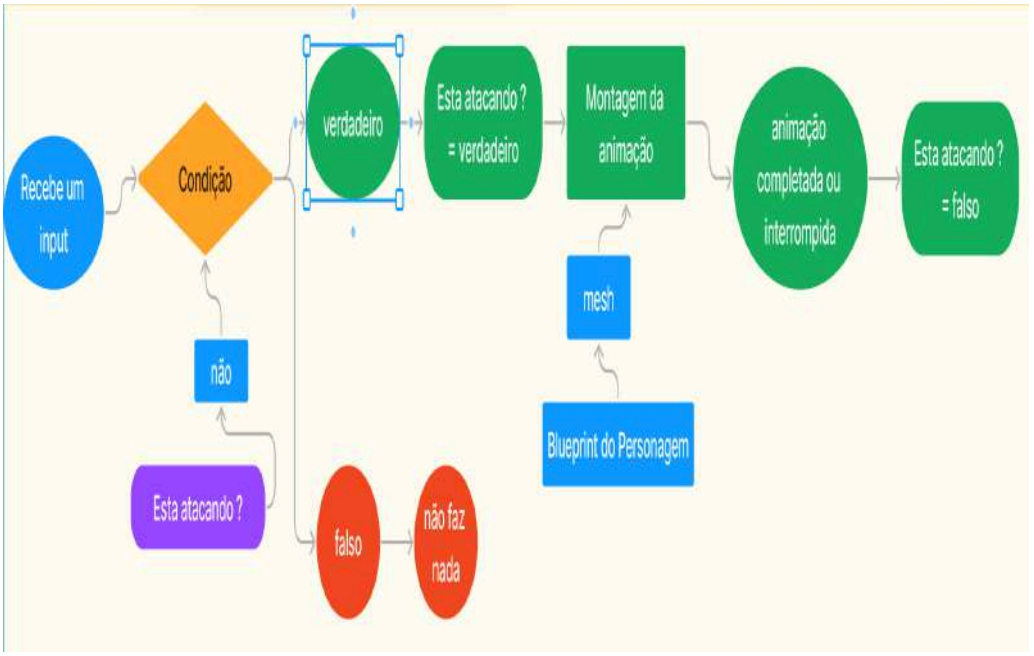


Fonte - Autor

Na Figura 30, observamos a lógica desenvolvida para realizar um ataque. Primeiro, o sistema recebe o input, que, neste caso, é o botão esquerdo do mouse, e realiza uma verificação. Caso a variável "Está atacando?" não seja verdadeira, ela é definida como verdadeira e prossegue para a montagem da animação de ataque.

Para executar a animação, é necessário movimentar a mesh (malha) do personagem. A malha inclui a geometria visual do modelo e sua textura, diferente do esqueleto, que é a estrutura interna dos ossos e controla a deformação da mesh. Contudo, no momento da programação, é utilizado o node(nó) mesh para garantir que a animação afete a parte visual do personagem. Como não estamos dentro do Blueprint do personagem, é preciso criar uma variável cujo tipo será o Blueprint do personagem criado. Dessa forma, conseguimos utilizar o node mesh para ligar à montagem da animação.

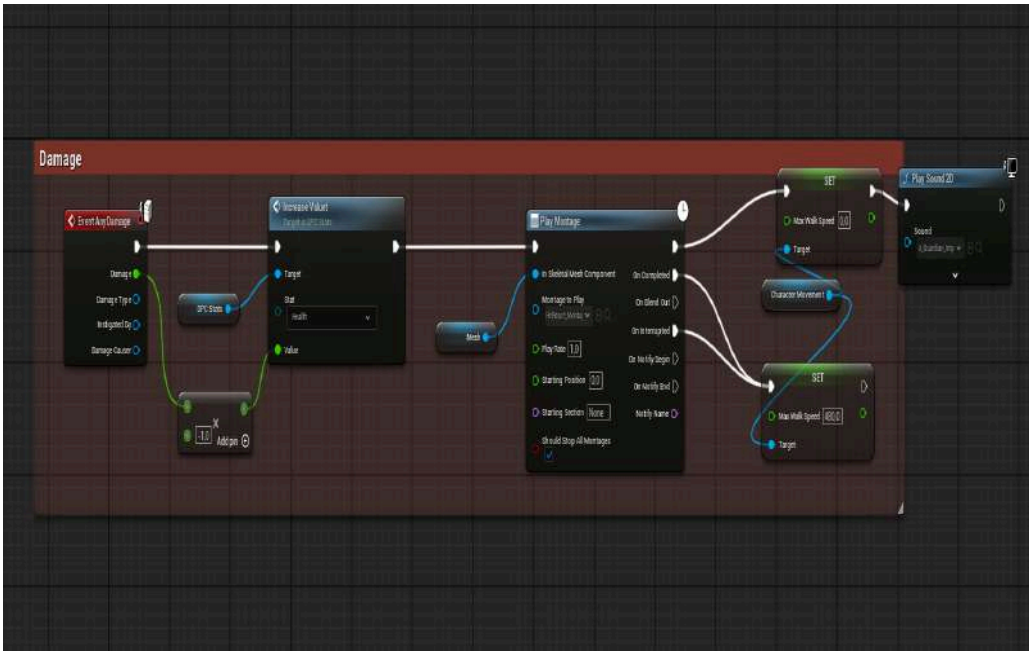
Figura 30 - Fluxograma Sistema de Ataque



Fonte - Autor

Quando a animação é completada ou interrompida por algum ataque de inimigo, a variável "Está atacando?" é configurada para falso novamente, permitindo que mais ataques sejam realizados. Isso garante que não seja possível atacar novamente durante um ataque em andamento. O código pode ser visualizado abaixo, na Figura 31.

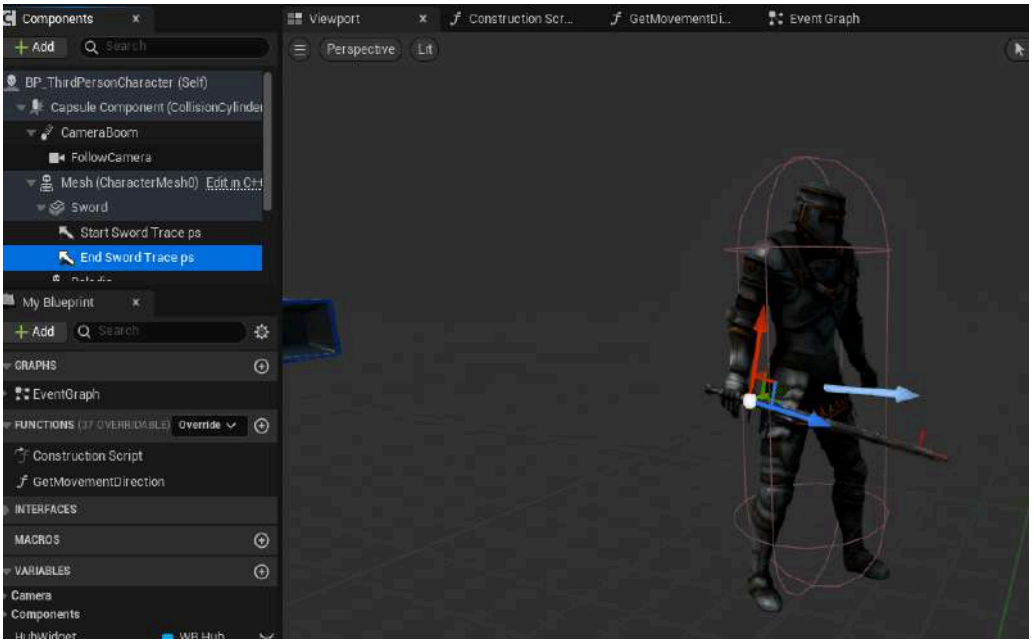
Figura 31 - Código do Sistema de Ataque Criado



Fonte - Autor

Agora é necessário identificar o ataque para que o dano possa ser aplicado. Para isso, no Blueprint do personagem jogável, foram inseridas duas arrows na espada utilizada pelo personagem. Com elas, é possível identificar a direção e a posição de um objeto. Uma arrow foi posicionada no início da lâmina e a outra no final, conforme ilustrado na Figura 32.

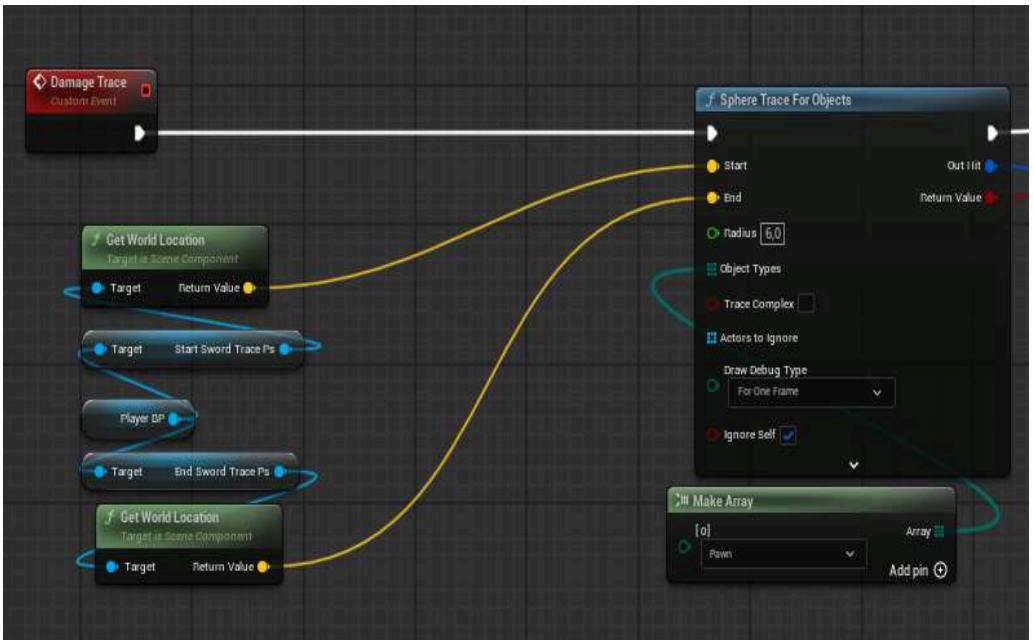
Figura 32 - Arrows



Fonte - Autor

Será utilizado um node chamado Event Tick, que, a cada frame, chama um evento denominado Damage Trace (Traço de Dano). Este evento é responsável por aplicar dano e é composto por três partes. A primeira parte consiste em obter as posições inicial e final da lâmina, que foram marcadas utilizando as arrows dentro do Blueprint do personagem. Para acessar essas informações, é necessário utilizar a variável criada anteriormente, que pertence ao Blueprint do personagem. Dessa forma, é possível referenciar as arrows e obter suas posições, as quais são conectadas ao node Sphere Trace for Objects. Este node recebe os parâmetros referentes à posição inicial e final da lâmina, assim como o tipo de objeto que se deseja atingir, que, neste caso, é um pawn, ou seja, um character. A parte 1 é ilustrada na Figura 33

Figura 33 - Código de Aplicar Part 1

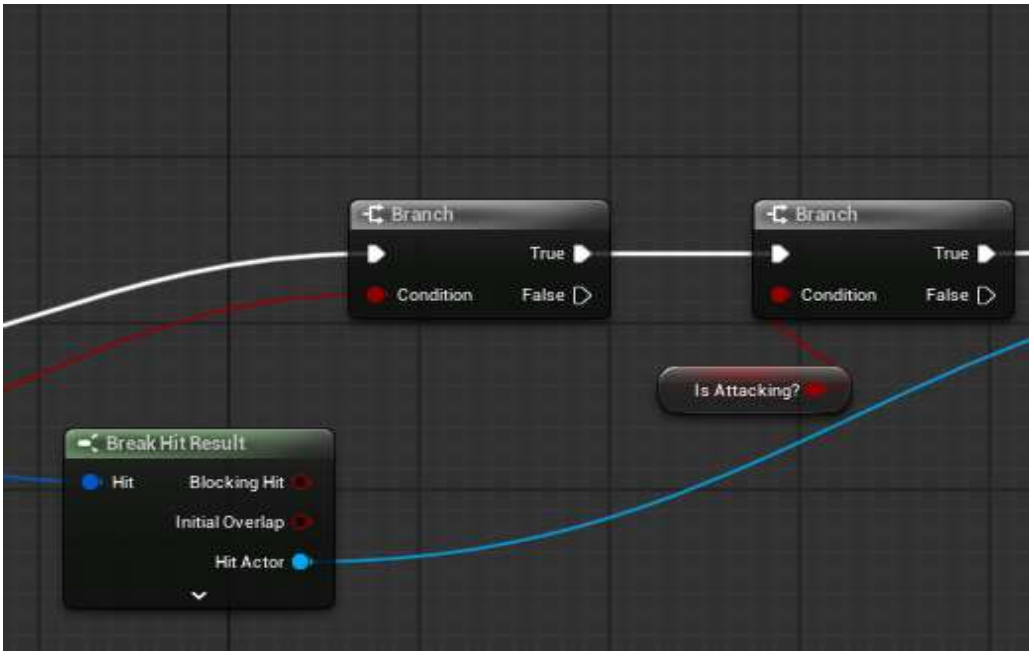


Fonte - Autor

Com isso, é gerada uma esfera em volta da espada, mas é necessário confirmar se ela está realmente atingindo um personagem. Para isso, realizamos essa verificação no primeiro Branch. Em seguida, também verificamos se a variável "Está atacando?" é verdadeira.

Além disso, o Sphere Trace está conectado ao node Break Hit Result, cuja função é identificar qual personagem está sendo atingido. Isso é importante porque o evento Apply Damage é global e, dentro de cada personagem, seja jogável ou NPC, existe um evento chamado Any Damage, que é acionado quando o Apply Damage é ativado. Dessa forma, é possível identificar qual personagem está recebendo dano. A parte 2 pode ser visualizada na Figura 34.

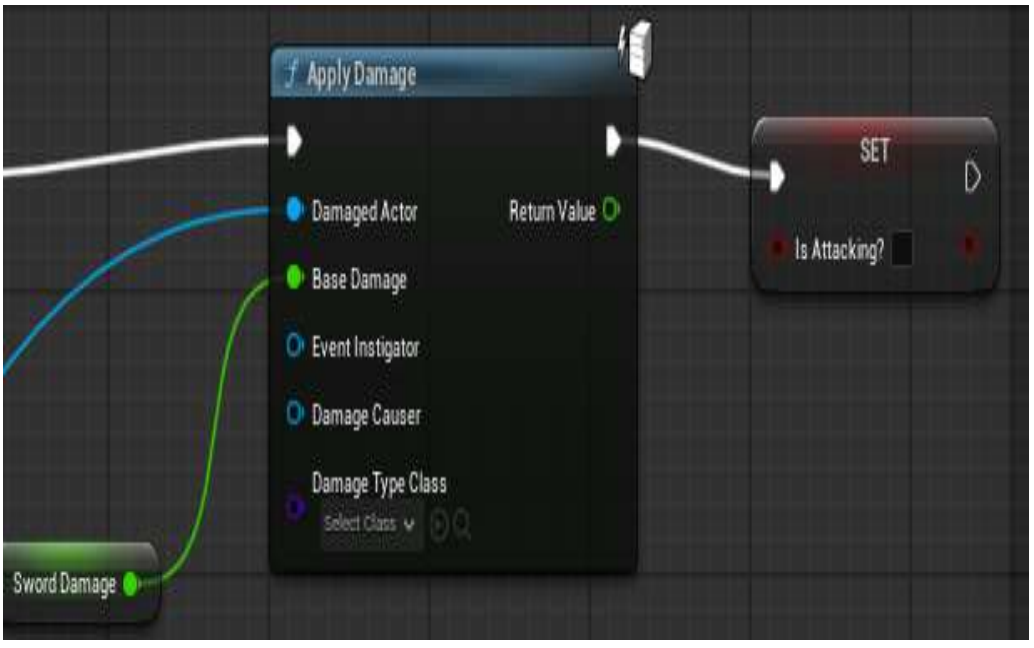
Figura 34 - Código de Aplicar Part 2



Fonte - Autor

Após serem feitas as verificações e obtida a informação sobre qual personagem foi atingido, só precisamos de mais um dado: o "sword damage" ou dano da espada, fornecido por uma variável. Com isso, o node Apply Damage possui todas as informações necessárias. Após aplicar o dano, a variável "Está atacando?" é configurada novamente como falso. Isso é ilustrado na Figura 35.

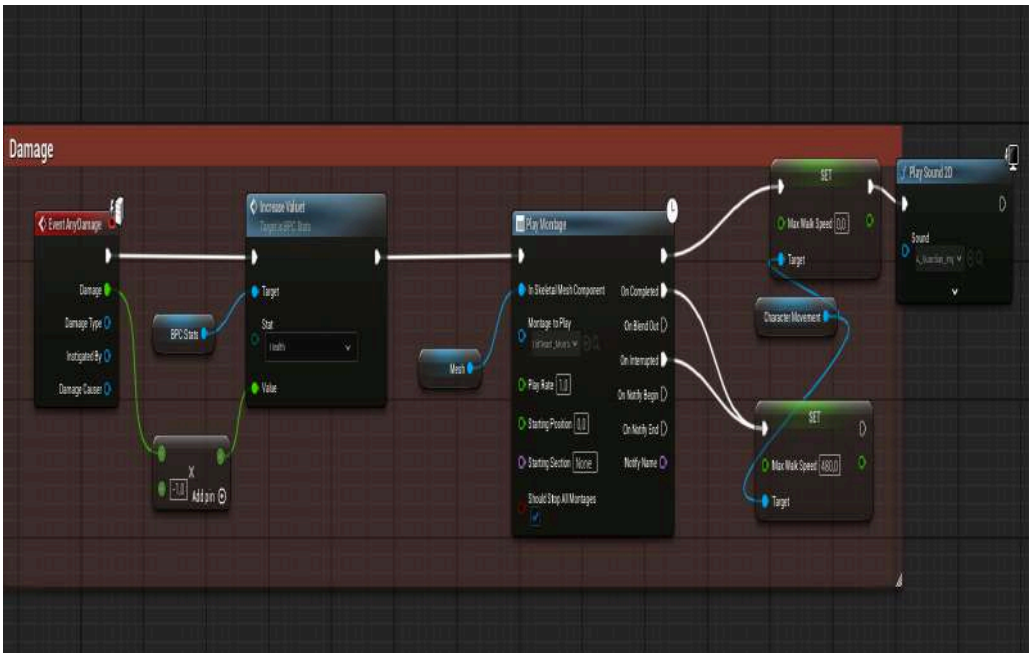
Figura 35 - Código de Aplicar Part 3



Fonte - Autor

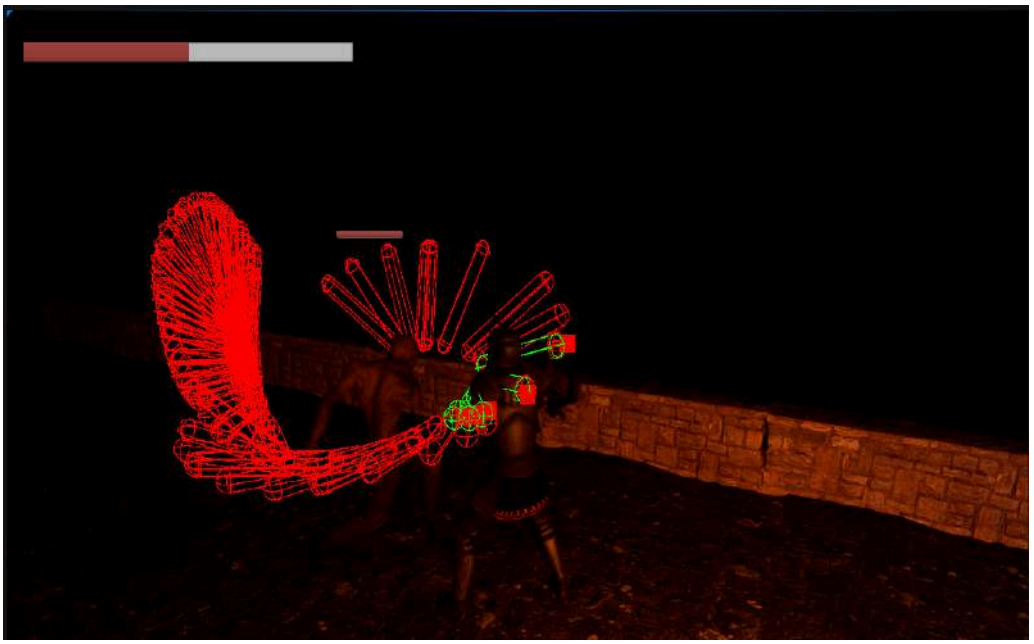
O que foi apresentado até agora serve apenas para indicar que um personagem foi atingido e que uma certa quantidade de dano foi aplicada. No entanto, para que o dano seja efetivamente aplicado, isso deve ser feito dentro do Blueprint do personagem jogável ou do inimigo, uma vez que o sistema é o mesmo, mudando apenas a quantidade de dano que o inimigo comum, o personagem jogável ou o chefe causam. O node Apply Damage se comunica com o node Event AnyDamage, que, por sua vez, chama o Blueprint Status, responsável por diminuir a vida. Em seguida, é ativada a animação de reação ao dano recebido, imobilizando o personagem que sofreu o dano. Na Figura 36, temos o evento de aplicar dano, e, na Figura 37, temos o resultado.

Figura 36 - Dano Sendo Aplicado



Fonte - Autor

Figura 37 - Ataque em Game

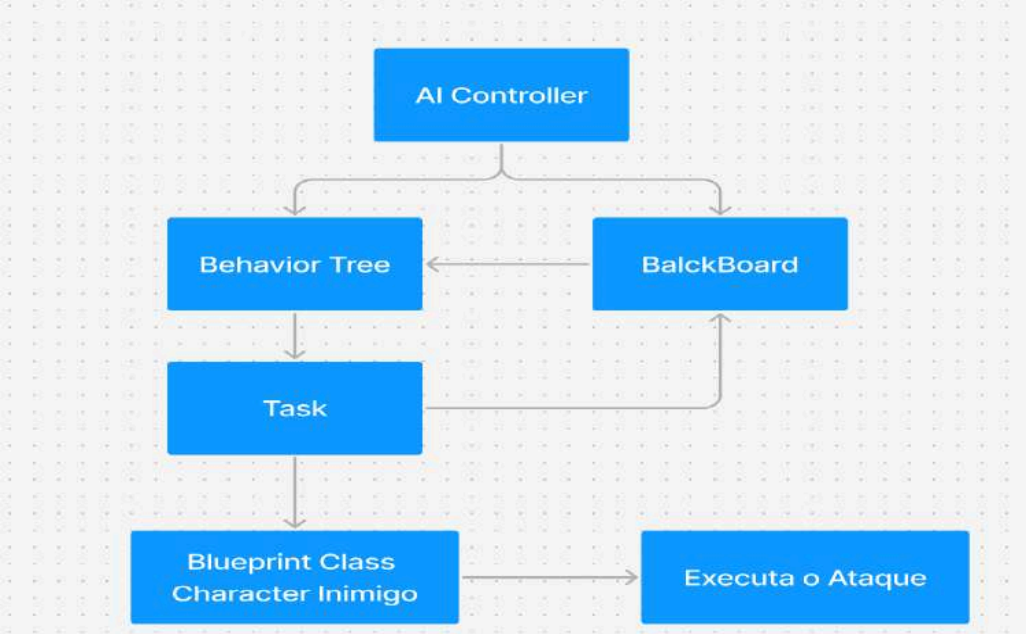


Fonte - Autor

4.3.3 – IA dos Inimigos

A IA será criada utilizando o Behavior Tree (Árvore de Comportamento), uma estrutura empregada para desenvolver a lógica de comportamento de personagens não-jogáveis (NPCs). Para criar uma IA, é necessário estruturar um AI Controller, um Behavior Tree, um Blackboard e, adicionalmente, o Blueprint Character. Na Figura 38, é apresentada a estrutura geral de funcionamento de todo o sistema, permitindo que a IA controle o NPC

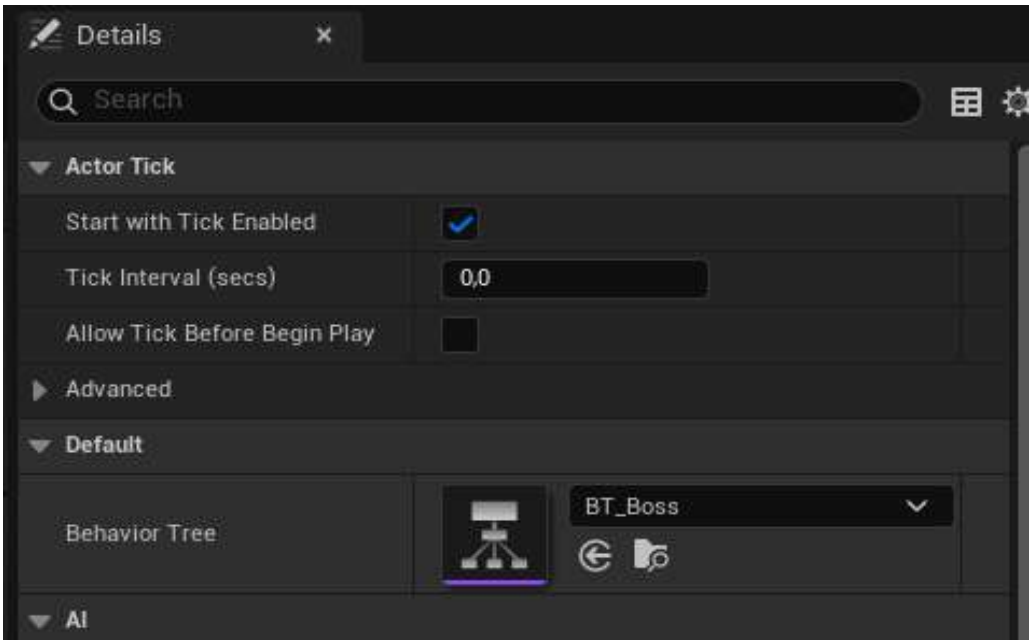
Figura 38 - Fluxo IA



Fonte – Autor

No AI Controller, será adicionado o Behavior Tree do chefe que foi criado. Dentro do Behavior Tree, será adicionado o Blackboard. Esse processo é ilustrado nas Figuras 39 e 40.

Figura 39 - Adicionando Behavior Tree a AI Controller



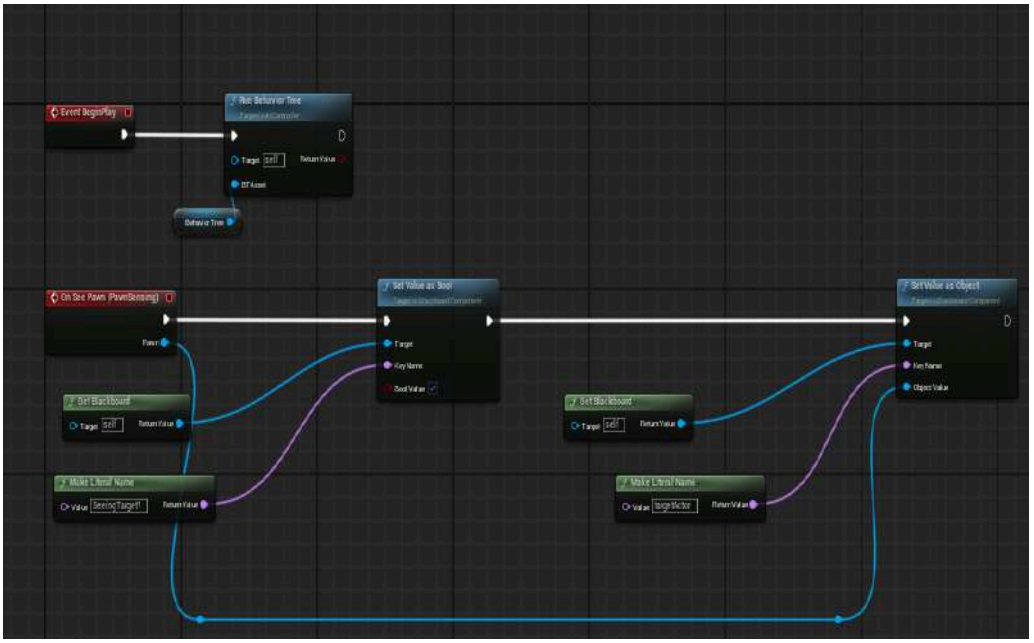
Fonte - Autor

Figura 40 - Adicionando BlackBoard a Behavior Tree

Fonte - Autor

Na Figura 41, observa-se o código aplicado no AI Controller. Basicamente, dentro deste controlador, o Behavior Tree é iniciado quando o jogo começa, e utilizamos o node denominado On See Pawn. Este node é empregado para, quando o NPC avistar o personagem, definir um valor booleano na variável "Seeing Target", indicando que o jogador está sendo visto.

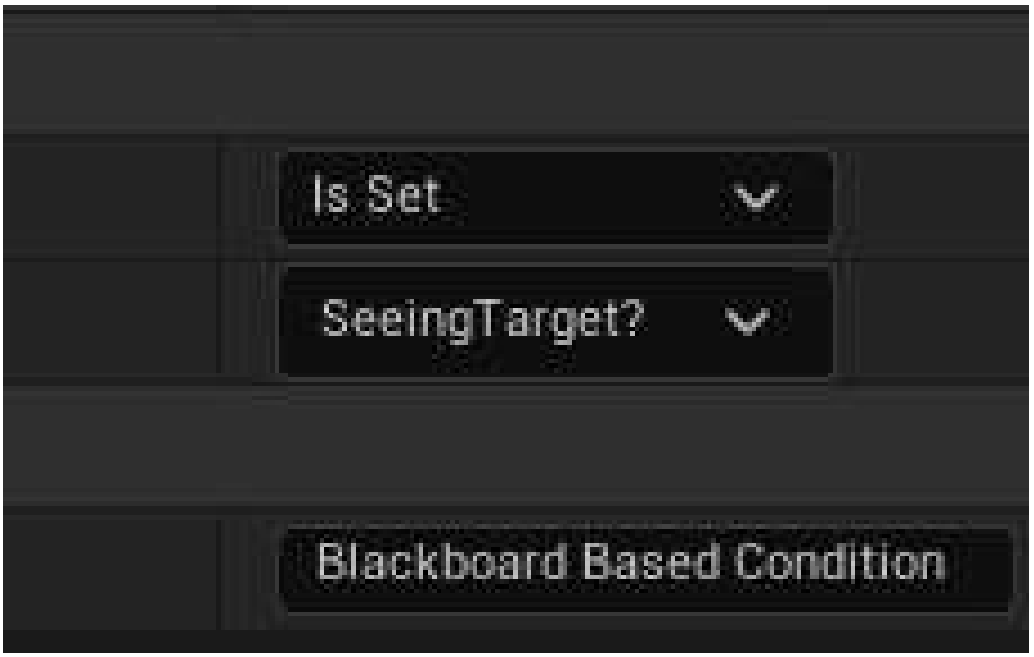
Figura 41 - Código da AI Controller



Fonte - Autor

Quem recebe a informação de que o jogador está sendo visto é o Blackboard, responsável por determinar a liberação ou não de certas execuções, conforme ilustrado na Figura 42.

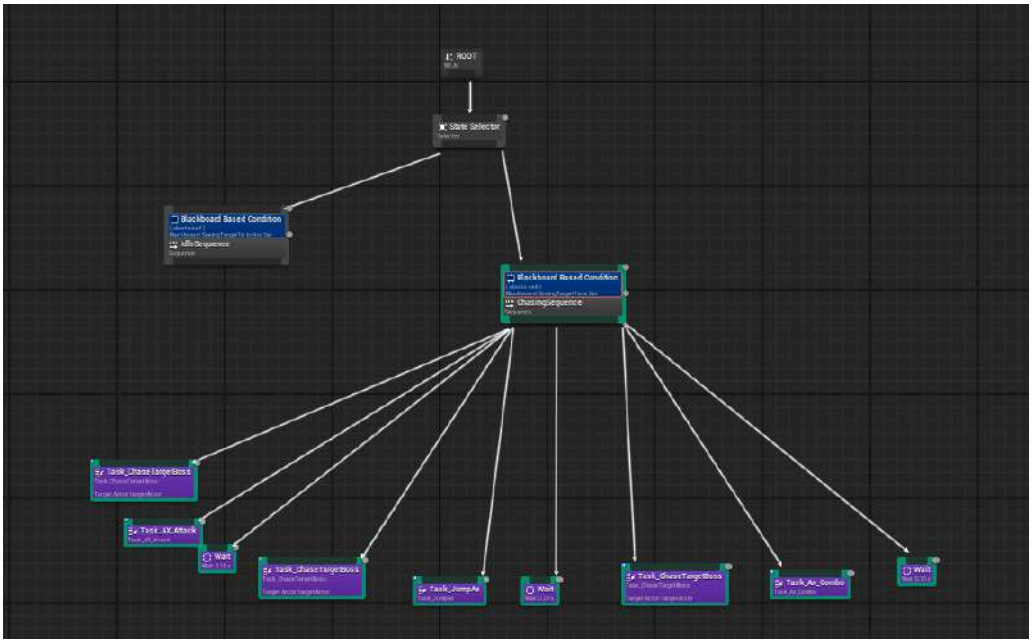
Figura 42 - BlackBoard Funcionamento



Fonte - Autor

Quando é configurado que o jogador está sendo visto, é liberada a execução das tasks (tarefas), que serão escolhidas com base na sequência que vai da esquerda para a direita, conforme ilustrado na Figura 43.

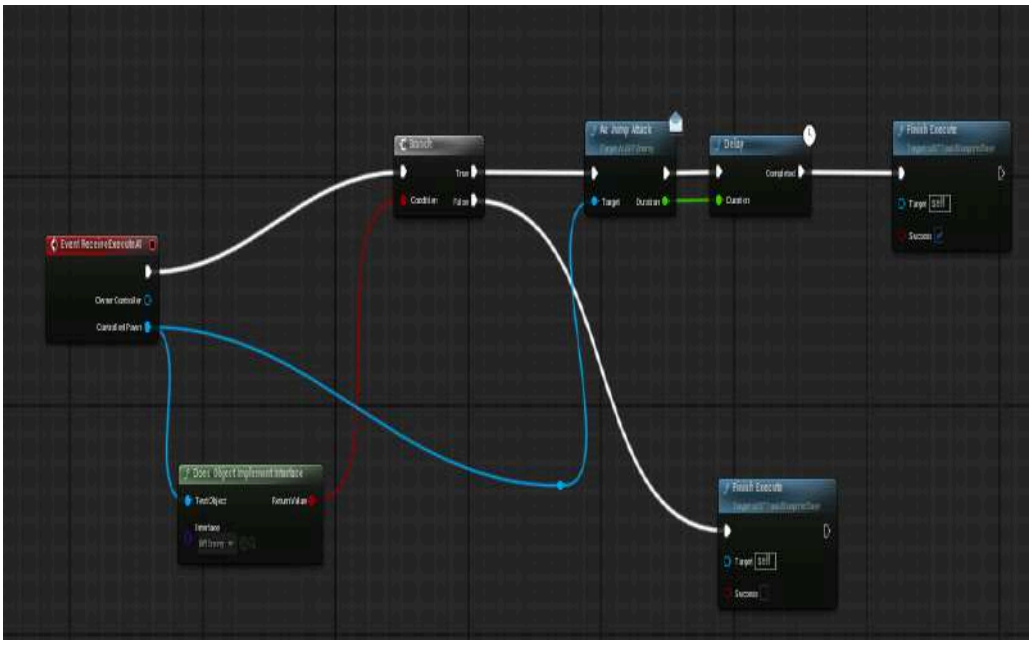
Figura 43 - Behavior Tree



Fonte - Autor

As tarefas são criadas dentro do Behavior Tree e representam as ações a serem executadas pela IA do chefe. Existem quatro tarefas: uma destinada a se aproximar do jogador após ele ser avistado, e mais três tarefas para realizar os diferentes tipos de ataques que o chefe possui. Além disso, há tarefas padrão, como a tarefa wait, que faz com que o chefe espere alguns segundos antes de executar outra tarefa.

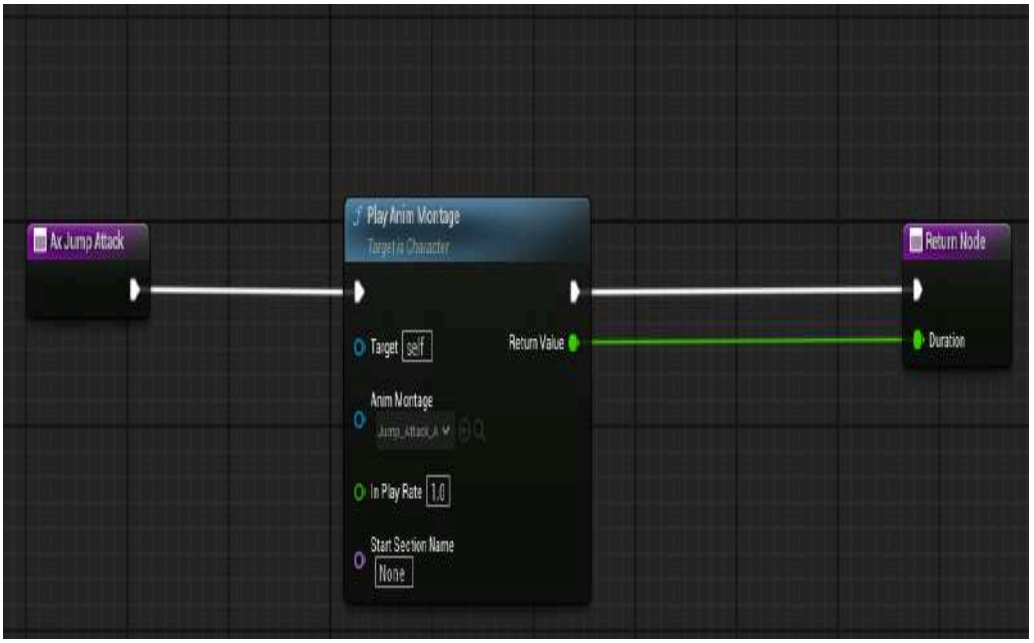
Figura 44 - Task Ataque com Pulo



Fonte - Autor

Quando uma tarefa de ataque é executada, ela envia uma mensagem pelo node Ax Jump Attack, que representa o ataque de pulo do chefe, conforme ilustrado na Figura 44. Note que este node possui um ícone de carta, indicando que ele se comunica com um node dentro do blueprint do chefe. Este node recebe a informação e executa a animação de ataque, como mostrado na Figura 45.

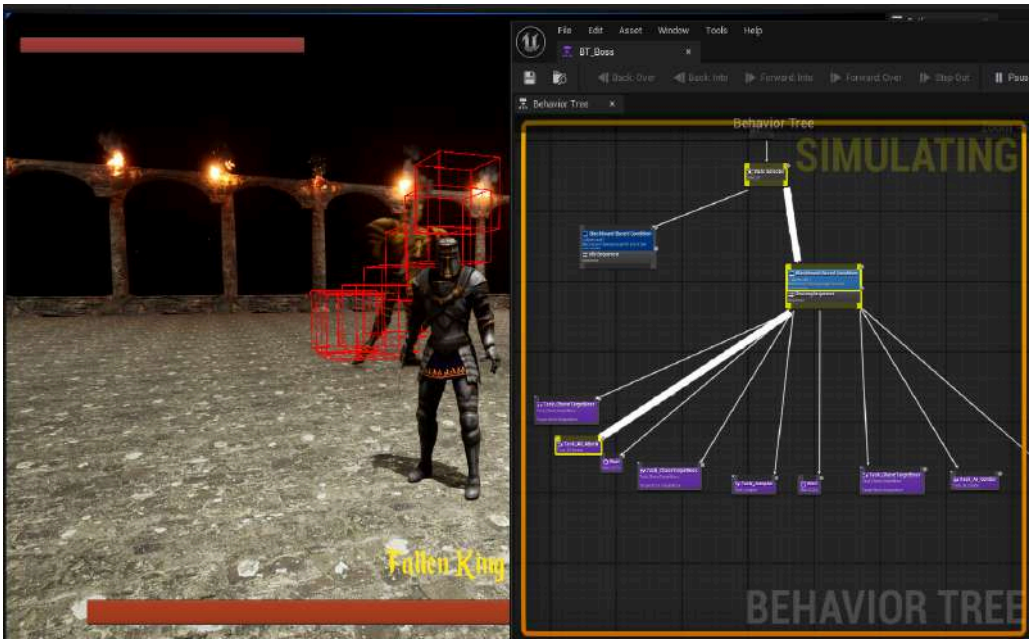
Figura 45 - Recebendo Mensagem e Executando Ataque



Fonte - Autor

Pode-se observar o funcionamento da árvore na Figura 46. A partir do momento em que o jogador entra na visão do chefe, a sequência de tarefas é ativada.

Figura 46 - Funcionamento da IA do Chefe



Fonte - Autor

4.4 – CRIAÇÃO DO MAPA

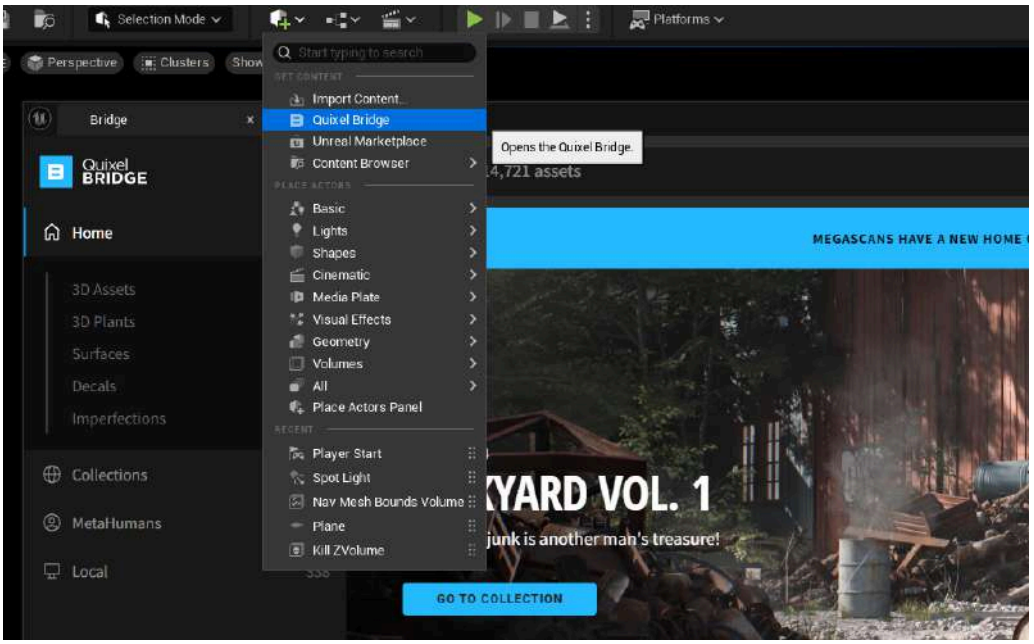
Na criação do mapa, será abordado os seguintes tópicos:

- Montagem do mapa usando assets do Quixel Bridge
- Nanite

4.4.1 – Montagem do Mapa Usando Assets do Quixel Bridge

Todos os assets, exceto os personagens, foram obtidos usando o Quixel Bridge, uma ferramenta integrada na Unreal Engine que facilita o processo de criação do mapa, como ilustrado na Figura 47

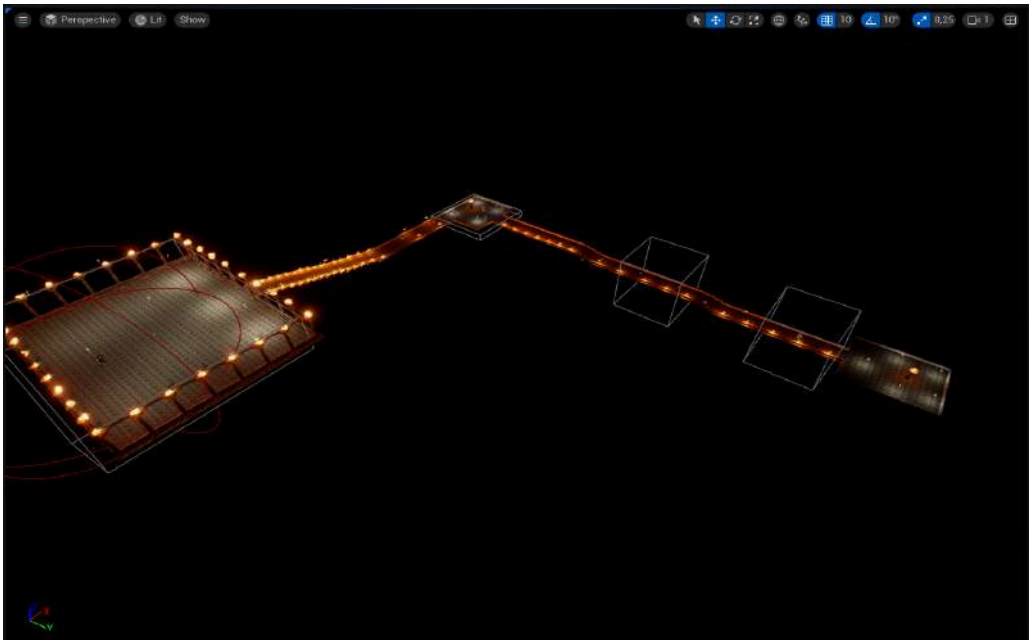
Figura 47 - Quixel Bridge



Fonte - Autor

A ideia do mapa é criar um caminho linear, onde ao redor tudo seja escuro e claustrofóbico. O jogador deve lutar em um corredor apertado com vários inimigos, como vemos na Figura 48.

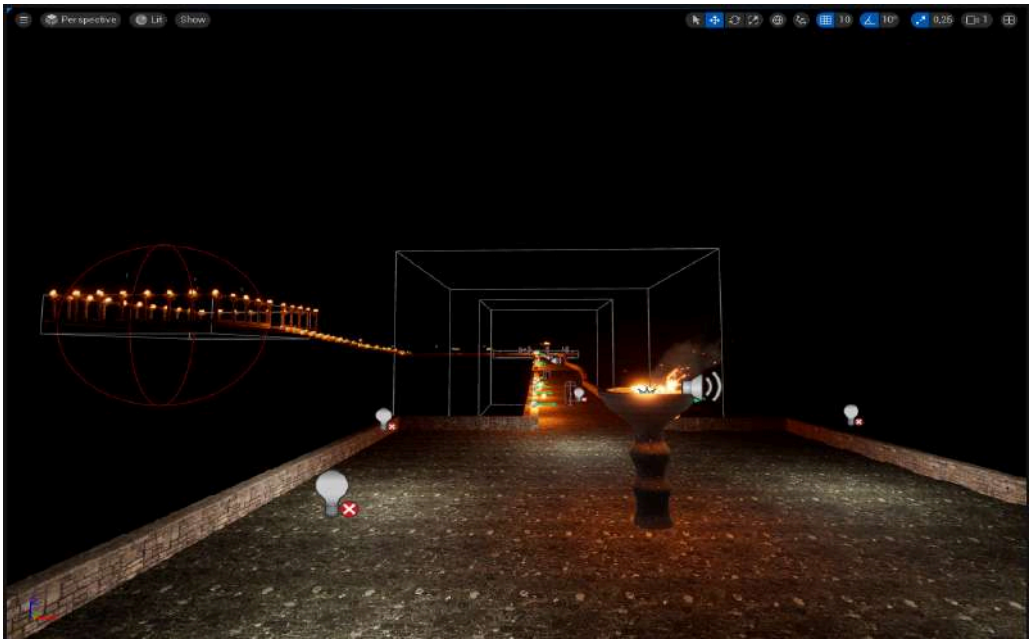
Figura 48 - Mapa Visto de Cima



Fonte - Autor

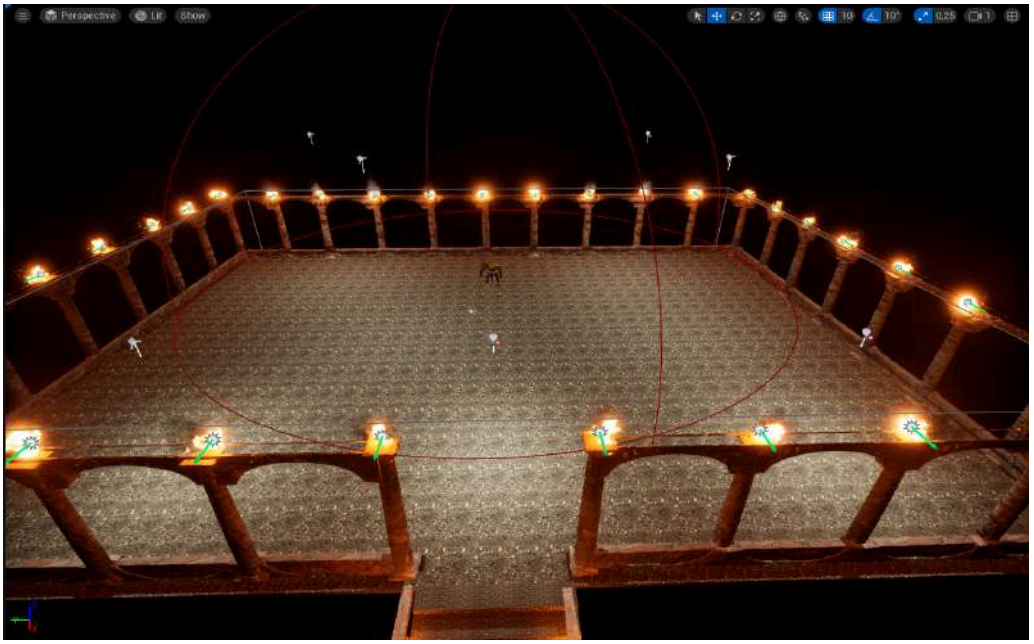
O personagem começa na parte mais baixa do mapa e deve subir até onde o chefe está para derrotá-lo e finalmente zerar o jogo. Vemos a parte inicial e final do mapa nas Figuras 49 e 50.

Figura 49 - Início do Mapa



Fonte - Autor

Figura 50 - Final do Mapa



Fonte - Autor

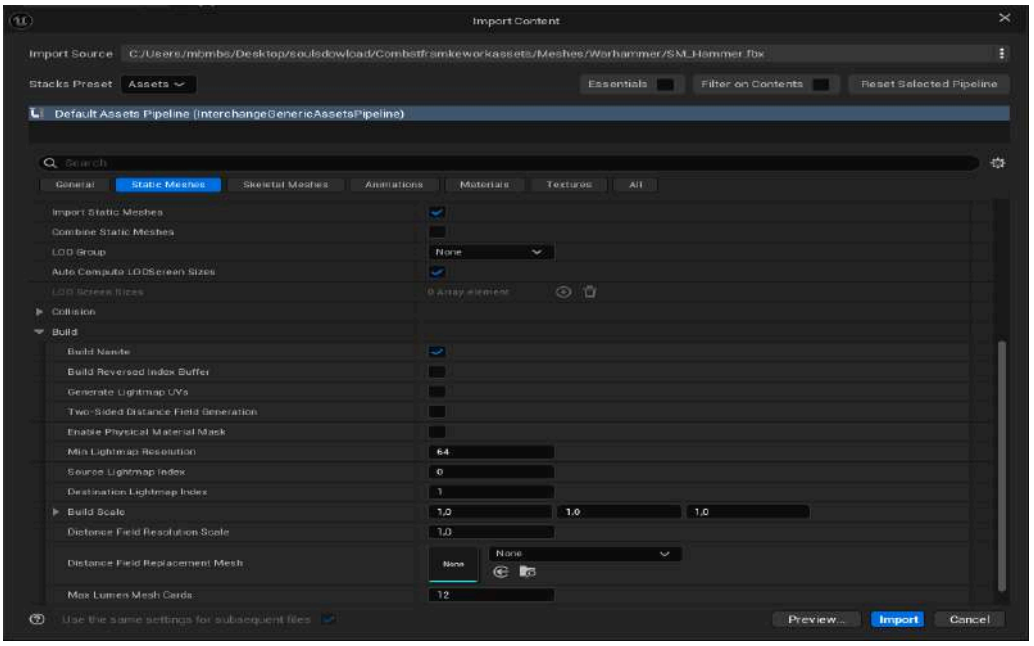
4.4.2 – Nanite

O poder que a Unreal Engine 5 oferece para a criação de jogos visualmente impressionantes se deve principalmente a uma trindade de tecnologias: Lumen, MetaHuman e Nanite. O Lumen é um sistema de iluminação global e reflexos dinâmicos, projetado para ser usado a partir da geração do PlayStation 5, Xbox Series S/X e computadores com placas de vídeo potentes, como a RTX 3060. Já o MetaHuman é um software de criação de personagens fotorealistas.

Devido a escolhas de desenvolvimento, o Lumen e o MetaHuman não foram utilizados, por isso não abordaremos esses dois temas em profundidade. No entanto, o Nanite foi utilizado, e agora veremos essa poderosa tecnologia.

Uma das tarefas mais trabalhosas no desenvolvimento de jogos é criar várias versões de um objeto, seja uma árvore, um carro, uma espada, uma casa ou qualquer outro item no jogo. Isso é necessário para implementar o sistema de diminuição de qualidade com base, por exemplo, na distância da câmera. O Nanite resolve esse problema, pois ao importar qualquer objeto para a Unreal Engine, vindo de qualquer lugar, o Nanite pode ser ativado para que o sistema de diminuição de qualidade faça esse trabalho automaticamente. A figura 51 ilustra como e marcar a opção para habilitar.

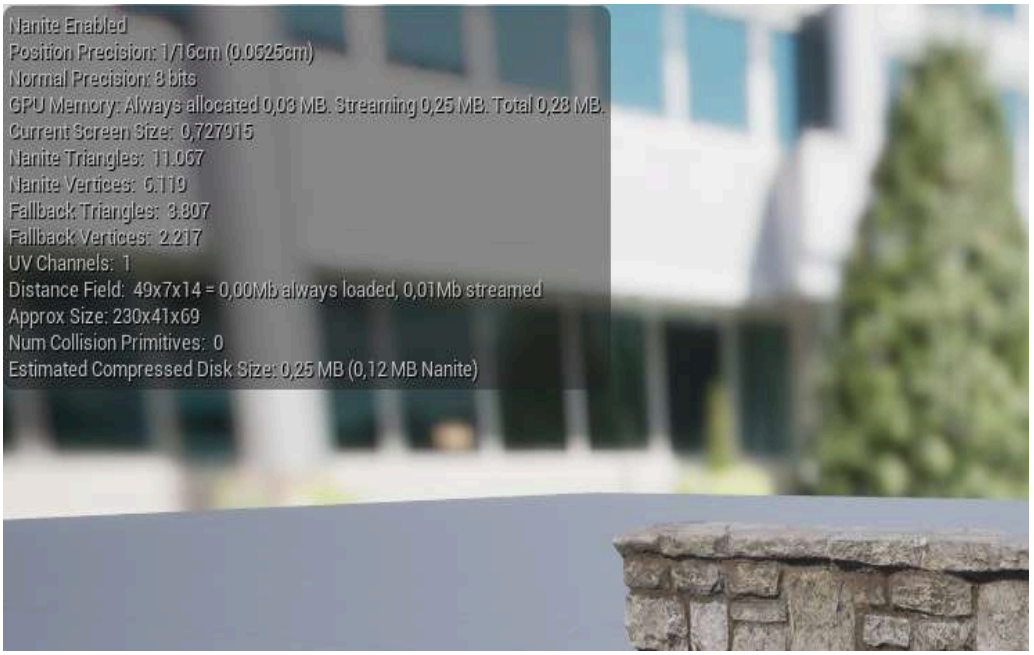
Figura 51 - Ativando Nanite



Fonte - Autor

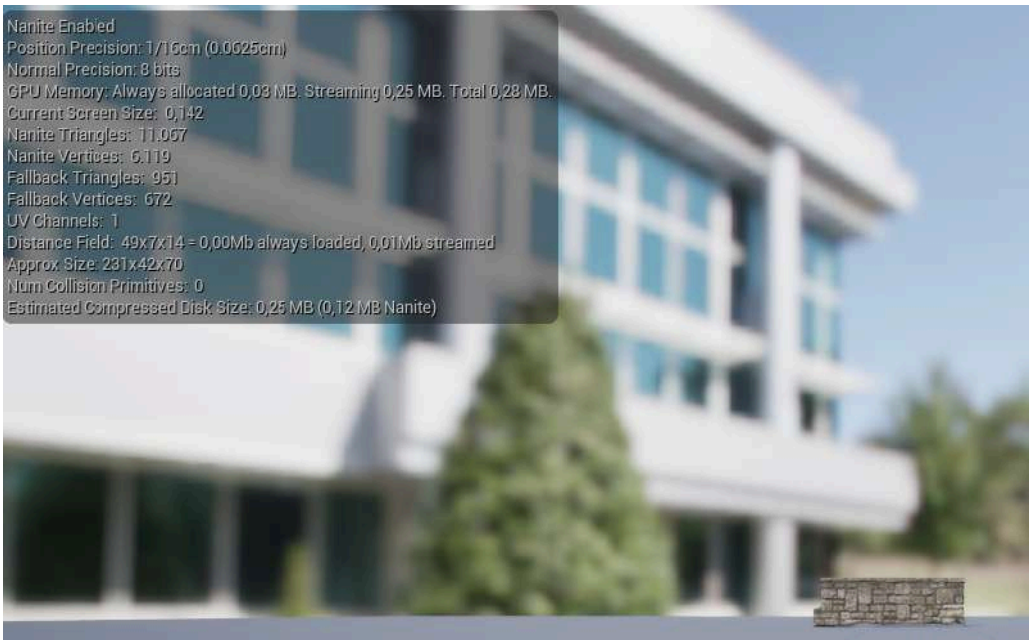
A partir do momento em que o Nanite está ativo, os LODs (níveis de detalhe) podem ser visualizados. O Nanite, por si só, faz essa troca de LODs, ou seja, ajusta a qualidade do objeto automaticamente, sem a necessidade de programação manual. Isso faz com que, caso se tenha mais versões do mesmo objeto com menos qualidade, ele vá ajustando conforme a câmera, trocando os LODs. Caso não se tenha outras versões do mesmo objeto, o Nanite fará com que os detalhes finos do objeto se tornem menos visíveis para o observador. Portanto, o sistema de renderização pode reduzir a quantidade de triângulos e vértices para otimizar o desempenho sem comprometer a qualidade visual percebida. Vemos esse processo nas figuras 52 e 53, onde, baseado na distância da câmera, a quantidade de triângulos e vértices é diminuída.

Figura 52 – Muro visto de perto



Fonte – Autor

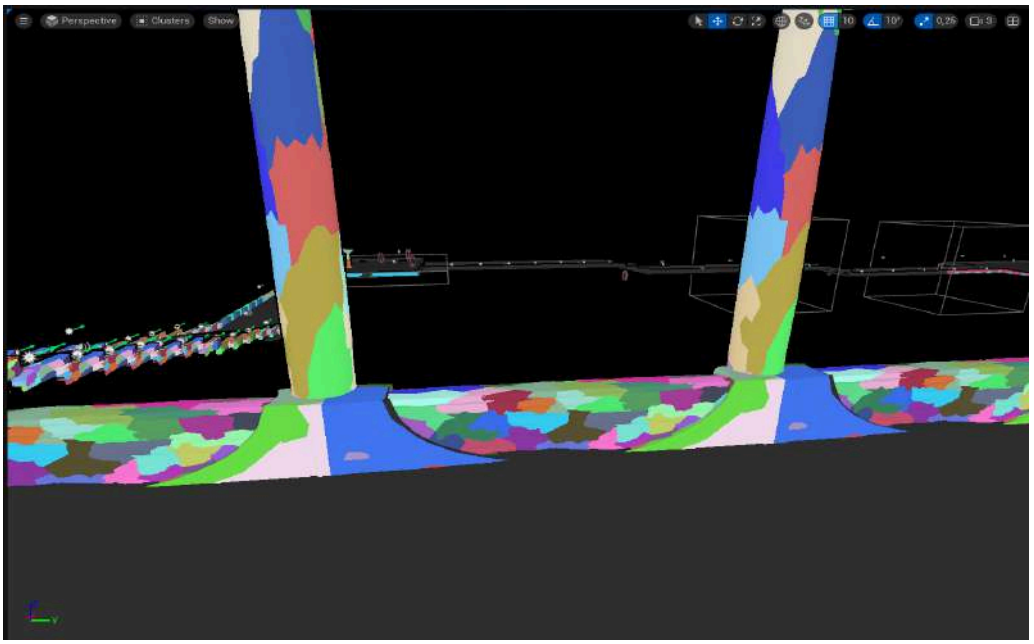
Figura 53 – Muro visto de longe



Fonte – Autor

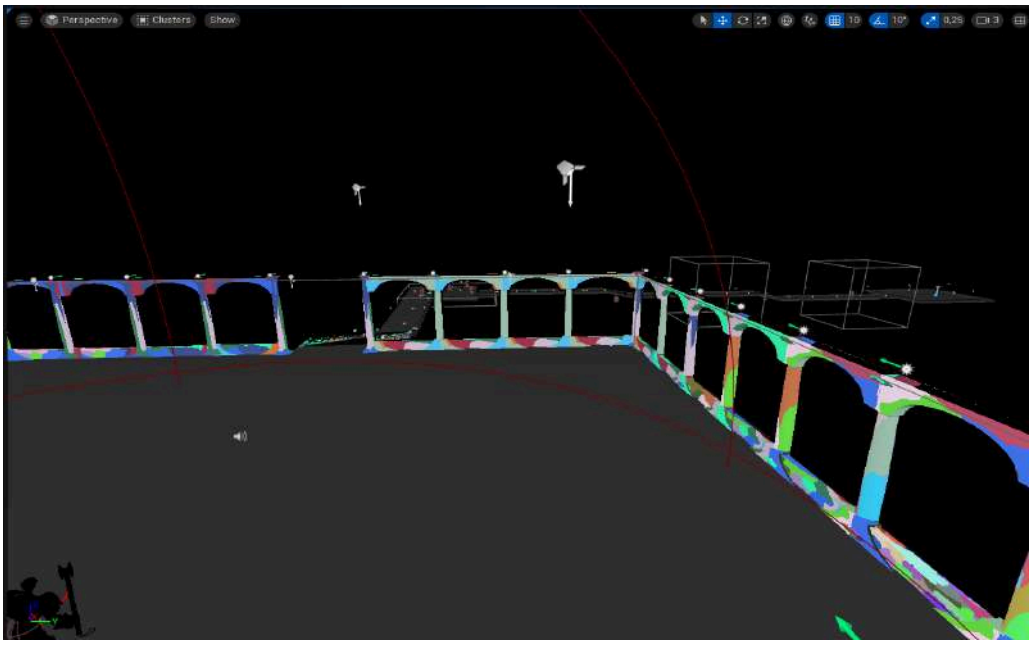
O Nanite é uma tecnologia de geometria virtualizada que sempre mostra apenas o que pode ser percebido, mantendo assim a máxima qualidade sem perda de desempenho. Nas figuras 54 e 55 , podemos ver como funciona essa troca de LODs (níveis de detalhe). Mesmo com a diminuição dos clusters, não há grandes perdas de qualidade. Isso pode ser feito mudando o tipo de visualização da cena para Nanite, permitindo ver apenas o Nanite em funcionamento.

Figura 54 - Funcionamento do Nanite Parte 1



Fonte - Auto

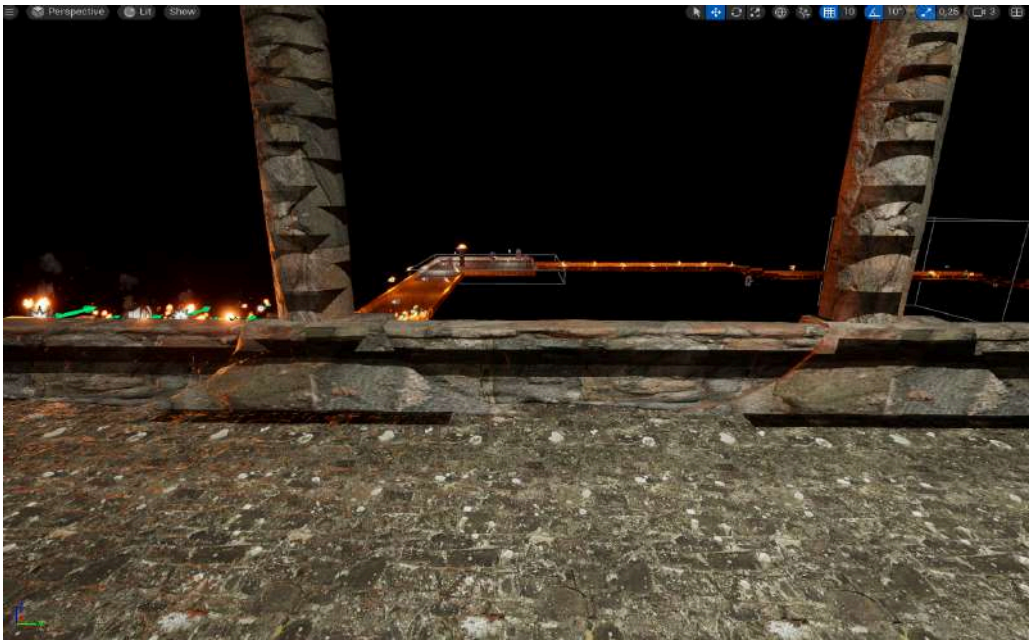
Figura 55 - Funcionamento do Nanite Parte 2



Fonte - Autor

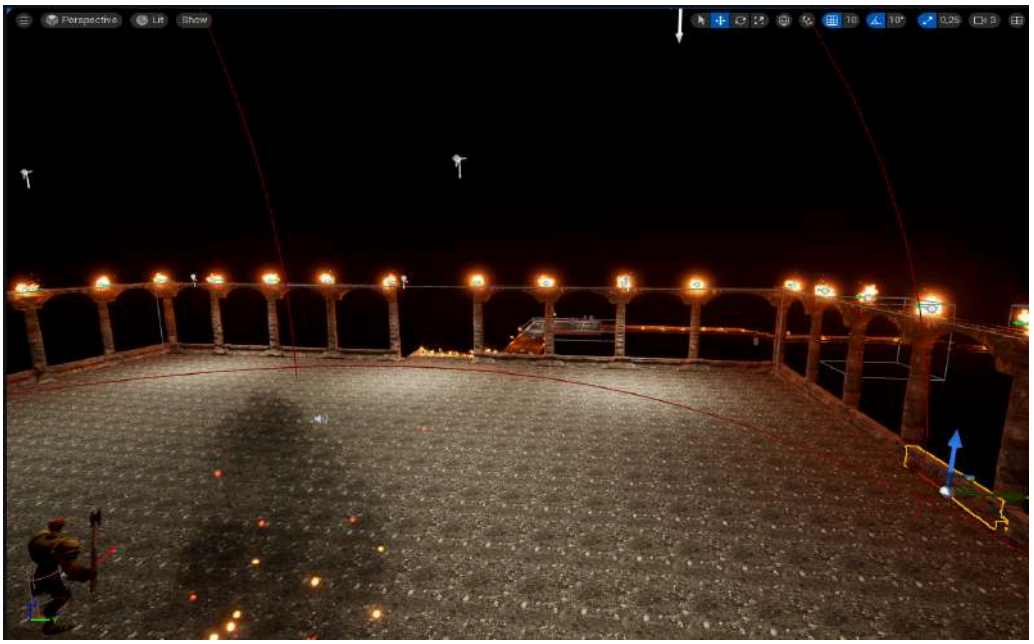
Agora, nas figuras 56 e 57, podemos ver os mesmos lugares, mas com a visualização normal. Podemos observar que, mesmo com a diminuição da quantidade de detalhes, não houve perda significativa de qualidade.

Figura 56 - Nanite em Game Parte 1



Fonte – Autor

Figura 57 - Nanite em Game Parte 2

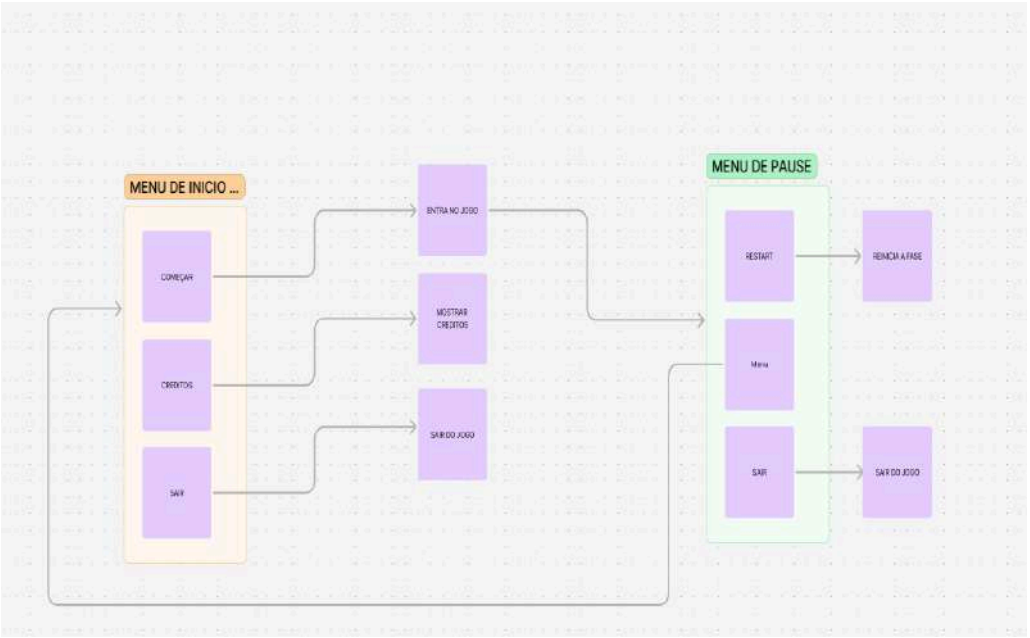


Fonte - Autor

4.5 – CRIAÇÃO DO MENU

Por fim, ficou a criação dos menus: o menu principal, que será mostrado na tela de início do jogo, e o menu de pausa, que será mostrado quando o jogador pausar o jogo. O funcionamento dos menus está sendo ilustrado pelo fluxograma da figura 58.

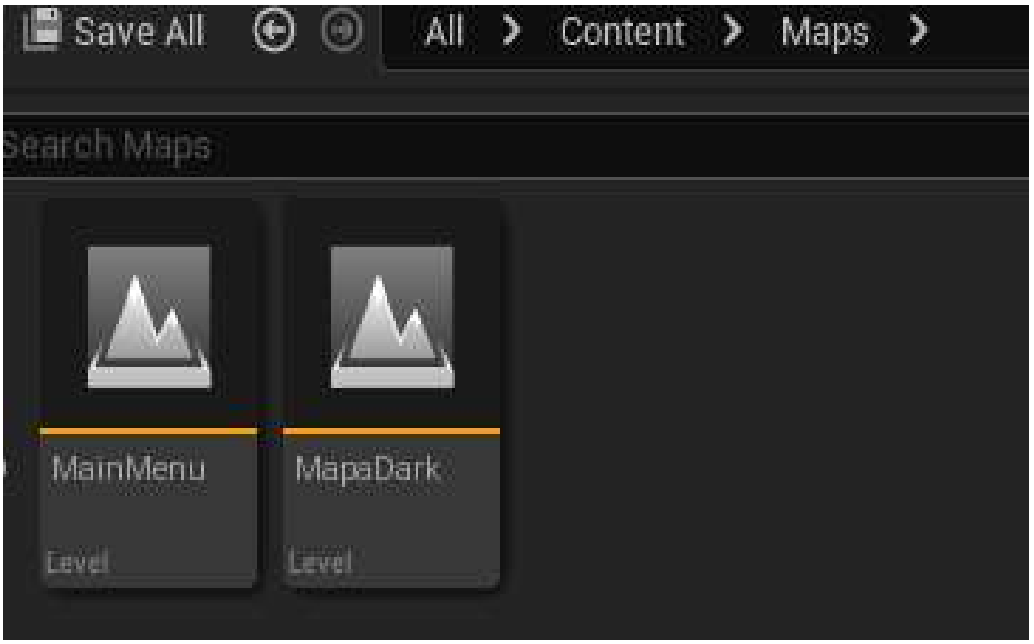
Figura 58 - Fluxo Menus



Fonte - Autor

Para tornar o menu ainda mais interessante, optou-se por animá-lo e incluir uma música em loop. A música foi obtida no site uppbeat.io, de uso gratuito.

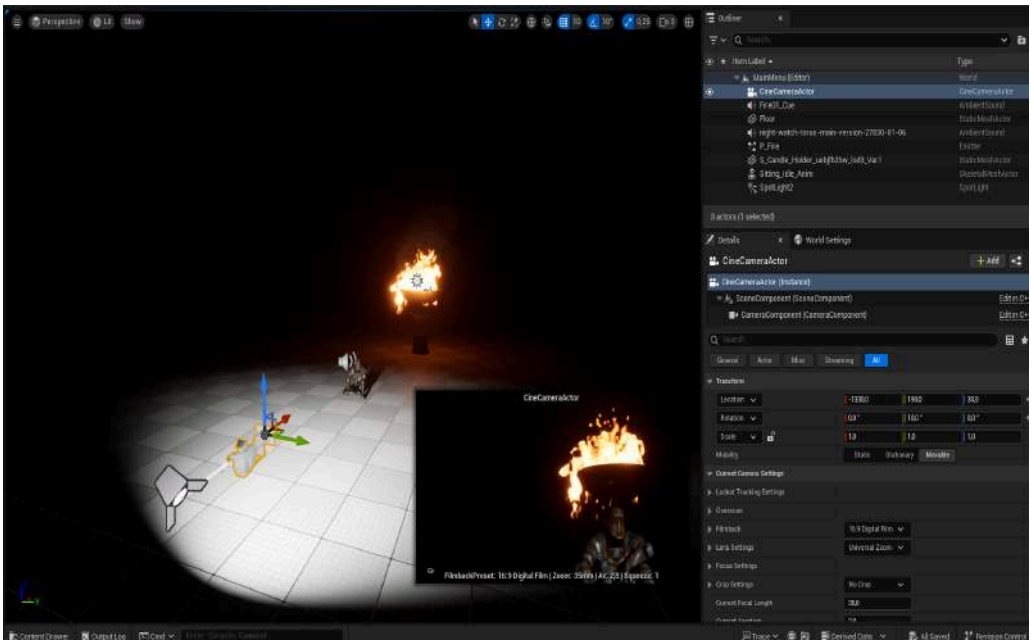
Figura 59 - Mapa do Menu



Fonte - Autor

Para criar um menu animado, primeiramente, cria-se um mapa diferente para montar a cena, como vemos na figura 59. Para montar a cena, foram usados quatro elementos: o spotlight, um CineCameraActor, uma animação de descanso sentado (que também foi baixada do Mixamo) e, por fim, um asset baixado do Quixel com uma animação de fogo, como vemos acima na figura 60.

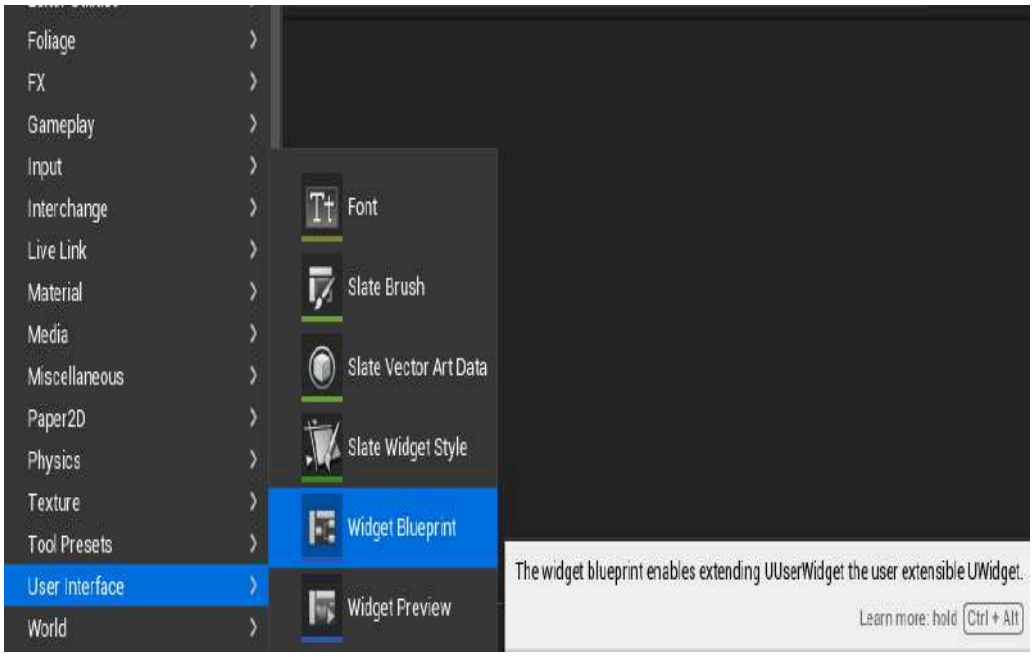
Figura 60 - Cena Criada



Fonte - Autor

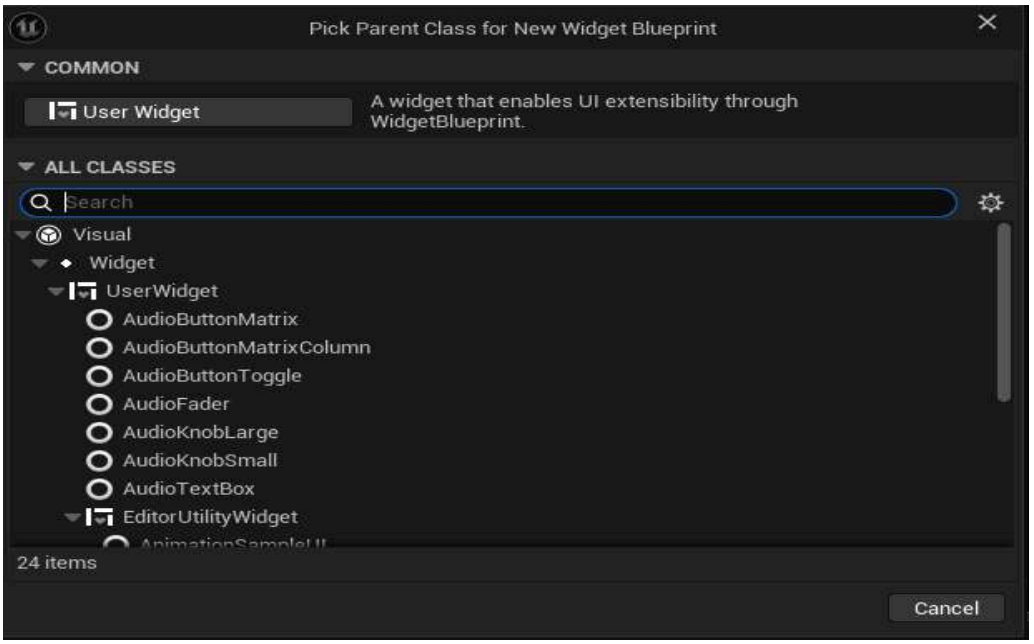
Com a cena pronta, resta apenas a criação do UI, onde terá o visual, e a programação do menu. Isso é feito indo em User Interface e criando um Widget Blueprint e, depois, escolhendo User Widget. Vemos esse processo nas figuras 61 e 62.

Figura 61 - Widget Blueprint



Fonte - Autor

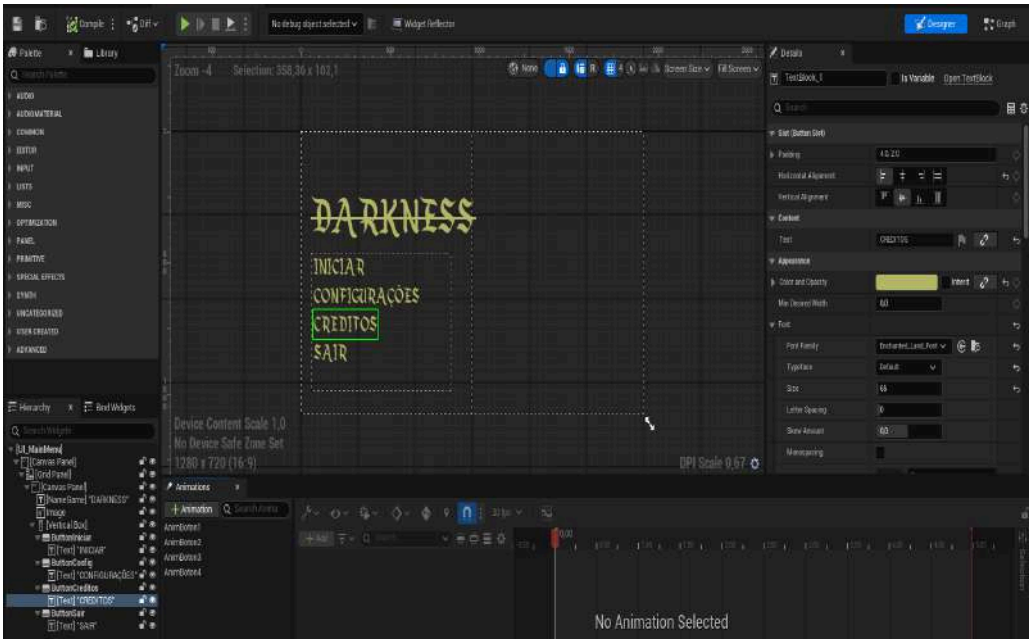
Figura 62 - User Widget



Fonte - Autor

Para criar a parte visual, primeiramente, coloca-se um canvas e, nele, serão colocados os textos e botões como ilustrado na figura 63.

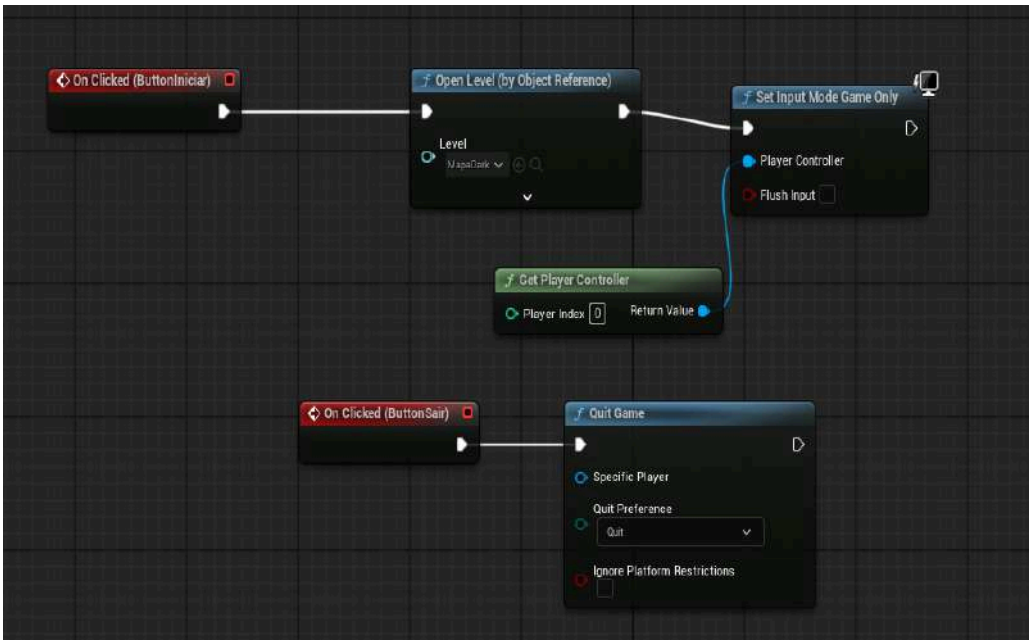
Figura 63 - UI Visual



Fonte - Autor

Na parte de programação, são chamados os botões criados na parte visual. Para iniciar o jogo, por exemplo, é usado o node Open Level (by object Reference). Esse node mostra os mapas criados, e então nele é colocado o mapa onde se passa o jogo de fato. figura 64.

Figura 64 - UI Programação



Fonte - Autor

Após criar a parte visual e programar as funcionalidades dos botões, temos os menus prontos, que podem ser vistos nas figuras 65 e 66 logo abaixo.

Figura 65 - Menu Principal

Fonte - Autor

Figura 66 - Menu Pause

Fonte - Autor

5 – CONCLUSÃO

Este trabalho possibilitou a exploração de diversos processos envolvidos na criação de um jogo utilizando a Unreal Engine 5, abrangendo diferentes etapas, como animação, programação de mecânicas, desenvolvimento de menus animados, implementação de inteligência artificial para inimigos e utilização de tecnologias avançadas do mercado, como o Nanite. O resultado foi um jogo curto, porém completo, que demonstra o potencial dessa engine.

A Unreal Engine é, de fato, uma ferramenta extremamente poderosa, com tecnologias projetadas para maximizar a qualidade visual dos jogos. No entanto, essa sofisticação tem seu preço, pois a inclusão de mais tecnologias aumenta a necessidade de otimização. Embora a Unreal Engine facilite certas etapas do desenvolvimento, como a criação de mecânicas sem a necessidade de dominar linguagens como C++ ou C# graças ao Blueprint, a otimização continua sendo um desafio considerável.

5.1 – Desafios de Desenvolvimento na Unreal Engine 5

Durante o período de desenvolvimento, diversas tecnologias foram removidas do projeto, tais como Metahuman, Gaea e Lumen, além de pacotes de texturas e árvores que seriam utilizados na ideia inicial do jogo. Esta decisão foi tomada devido à dificuldade de manter o projeto suficientemente otimizado.

A Unreal Engine, por si só, já é bastante exigente em termos de hardware, e a inclusão de várias tecnologias para tornar o jogo mais interessante e aproveitar ao máximo os recursos oferecidos pela engine pode levar ao desequilíbrio do projeto. Mesmo aplicando algumas das técnicas de otimização mais simples, ainda é necessário um conhecimento aprofundado neste quesito para viabilizar o uso de tanta tecnologia no projeto.

Os problemas enfrentados na otimização ocasionaram mudanças na ideia do jogo ao longo do desenvolvimento, resultando no jogo apresentado neste TCC.

5.2 – Trabalhos Futuros

Como trabalhos futuros, propõe-se a melhoria da inteligência artificial dos inimigos, além da implementação de Procedural Content Generation (PCG) para a criação de mapas gerados proceduralmente, tornando o jogo mais dinâmico e variado a cada partida. Também se sugere a adição de novos tipos de inimigos e a inclusão de mais obstáculos nos cenários, aumentando o desafio e a diversidade da experiência de jogo.

Além disso, recomenda-se a adição de mais personagens jogáveis, como magos e arqueiros, para diversificar as opções de combate.

Por fim, sugere-se a criação de uma cutscene de início de jogo, visando tornar a experiência ainda mais imersiva e cativar os jogadores, despertando seu interesse pelo mundo do jogo.

6 – REFERÊNCIAS

ABHISHEK, S.; VARGHESE, Ashwin; THOMAS, Ayibel; JOSEPH, Chris Sajimon; VARGHESE, Tintu. **Comparative Analysis on Unreal Engine 5 VS Unity**. International Journal of Engineering Technology and Management Sciences, v. 7, n. 4, p. 438, jul./ago. 2023. DOI: <10.46647/ijetms.2023.v07i04.058>. Acesso em: 3 jan. 2025

Blender's History. Blender. Disponível em: <https://docs.blender.org/manual/en/latest/getting_started/about/history.html>. Acesso em: 12 ago. 2024.

BUTCHER, Billy. **Epic Games divulga lista de estúdios que estão trabalhando com o Unreal Engine 5 nos seus projetos**. GameVicio, Volta Redonda, Rio de Janeiro, 05 abr. 2022. Disponível em: <https://www.gamevicio.com/noticias/2022/04/epic-games-divulga-lista-de-estudios-que-esta o-trabalhando-com-o-unreal-engine-5-nos-seus-projetos/>. Acesso em: 9 mar. 2025.

Creating Mixamo animations with Blender. GarageFarm.Net. Disponivel em:https://garagefarm.net/blog/creating-mixamo-animations-with-blender. Acesso em: 12 dez. 2024

DIAS. **Epic Games adquire Quixel e disponibiliza o Megascans gratuitamente**. Gamevicio, 2019. Disponível em: <https://www.gamevicio.com/noticias/2019/11/epic-games-adquire-quixel-e-disponibiliza-o-m egascans-gratuitamente/>. Acesso em: 12 ago. 2024.

FILHO, Márcio; ZAMBON, Pedro. **Setor de games no Brasil movimenta R$ 13 bilhões por ano, mas ainda sem uma política nacional adequada**. 2023. Disponível em: . Acesso em: 10 dez. 2024.

GARCÍA, David Méndez. **Combate por turnos para videojuego RPG en Unreal Engine 5**. Orientador: Diego Viejo Hernando. 2023. 76 p. Trabalho de Conclusão de Curso (Bacharelado em Engenharia da Computação) - Universidad de Alicante, ALICANTE, 2023. Disponível em: <https://rua.ua.es/dspace/bitstream/10045/135336/1/Diseno_y_desarrollo_de_un_videojueg o_RPG_Combate_contra__Mendez_Garcia_David.pdf>. Acesso em: 12 ago. 2024.

Games: um mercado de trabalho em expansão no Brasil. Ebac, 2024. Disponivel em : **<Games: um mercado de trabalho em expansão no Brasil>**. Acesso em: 15 fev. 2025.

HÄMÄLÄINEN, Jani. **Game Development with Unreal Engine 4**. 2020. 31 p. Trabalho de Conclusão de Curso (Bacharelado em tecnologia Empresarial) - Laurea University of Applied Sciences, 2020. Disponível <https://www.theseus.fi/bitstream/handle/10024/346676/Hamalainen_Jani.pdf?sequence=2>. Acesso em: 12 ago. 2024.

Hardware and Software Specifications for Unreal Engine. Epic Games Documentation. Disponível em: https://dev.epicgames.com/documentation/en-us/unreal-engine/hardware-and-software-specifications-for-unreal-engine?application_version=5.5. Acesso em: 15 fev. 2025.

High-fidelity digital humans made easy. Unreal Engine. Disponível em: <https://www.unrealengine.com/en-US/metahuman>. Acesso em: 15 dez. 2024.

Licensing. Unreal Engine. Disponível em: <https://www.unrealengine.com/en-US/license>. Acesso em: 12 ago. 2024.

MAKITALO, Pasi. Unity ja Unreal Engine -pelimoottorien vertailu. 2022. 29 p. Tese de Bacharelado - Universidade de Ciências Aplicadas de Tampere, Finlândia, 2022. Disponível em: <https://www.theseus.fi/bitstream/handle/10024/785308/Makitalo_Pasi.pdf?sequence=3&isAllowed=y>. Acesso em: 12 ago. 2024.

Mercado de games em 2020 cresceu 9.3%. MEUPC.Net, 2020. Disponível em:<Mercado de games em 2020 cresceu 9,3% – Blog do MEUPC.NET>. Acesso em 15 fev. 2025

MONTENEGRO, Bruna. **O que é a Unreal Engine?**. Ebac, 2023. Disponível em: <https://ebaconline.com.br/blog/o-que-e-a-unreal-engine>. Acesso em: 12 ago. 2024.

Indústria de jogos - Análise de tamanho e participação - Tendências e previsões de crescimento (2024 - 2029). Mordor Intelligence, 2024. Disponível em: <https://www.mordorintelligence.com/pt/industry-reports/global-gaming-market>. Acesso em: 12 ago. 2024.

RAYAT, Harmanjyot Kaur; em: RODRIGUES, Savio; RUKHANDE, Smita; KURICHITHANAM, Ritson Mathews. **Shriek: A Role Playing Game Using Unreal Engine 4 and Behaviour Trees**. International Conference on Nascent Technologies in Engineering. Disponível em : <https://ieeexplore.ieee.org/document/9487723> Acesso 13 ago 2024.

SILVA, Lucas Rodrigues da; RODRIGUES, Luciene Cavalcanti. **Análise de Desempenho de Diferentes Engines no Desenvolvimento de Jogos com Implementação de Wizard para Recomendação das Engines.** Faculdade de Tecnologia de São José do Rio Preto. Disponível em: <https://ric.cps.sp.gov.br/handle/123456789/23428> Acesso em: 15 dez. 2024.

SCHERER, Daniel; BATISTA, Daniele Ventura; MENDES, Aline de Cantalice. **Análise da Evolução de Engines de Jogos**. Universidade Estadual da Paraíba, Campina Grande, 2020. Disponível em: <https://sol.sbc.org.br/index.php/ctrle/article/view/11420>. Acesso em: 16 fev. 2025.