CENTRO UNIVERSITÁRIO DO VALE DO IGUAÇU
CURSO SUPERIOR DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

NICOLAS CIOCZEK JAKYMIU

DESENVOLVIMENTO DE JOGOS NA ENGINE UNITY

UNIÃO DA VITÓRIA – PR.
2021

NICOLAS CIOCZEK JAKYMIU

ESTUDO SOBRE O DESENVOLVIMENTO DE JOGOS NA ENGINE UNITY

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação, Área das Ciências Exatas do Centro Universitário do Vale do Iguaçu – Uniguaçu, como requisito à obtenção de grau de Bacharel em Sistemas de Informação. Professora Orientadora: Andrea Tomko.

UNIÃO DA VITÓRIA – PR

2021

TERMO DE APROVAÇÃO

DESENVOLVIMENTO DE JOGOS NA ENGINE UNITY

NICOLAS CIOCZEK JAKYMIU

Trabalho de conclusão de curso apresentado ao Curso de Sistemas de Informação do Centro Universitário do Vale do Iguaçu, como requisito para obtenção do Grau de Bacharel em Sistemas de Informação, considerado aprovado pela banca examinadora e avaliado como nota: _____ em sua defesa pública.

_____

Orientadora: Profa. Andréa Tomko
Centro Universitário do Vale do Iguaçu

_____

Prof:
Centro Universitário do Vale do Iguaçu

_____

Prof:
Centro Universitário do Vale do Iguaçu

União da Vitória – PR
23 de março de 2021

AGRADECIMENTOS

Agradeço a minha família que esteve comigo me auxiliando durante toda a jornada acadêmica, oferecendo o melhor de si para que eu concluísse a graduação na área que eu amo.

Agradeço aos professores que me instruíram com profissionalismo e sempre buscaram sanar as minhas dúvidas sobre os mais diversos assuntos. Sem a ajuda de vocês, não seria possível concluir essa caminhada de aprendizado e formação profissional.

Gostaria de agradecer também aos meus colegas de turma, por terem proporcionado momentos únicos e que sempre estarão em minha mente. Sem amigos tudo se torna mais difícil, então se estou aqui agora, em parte se deve ao apoio dos que estavam ao meu redor.

Por fim gostaria de agradecer especialmente a professora Andrea, por ter me ajudado na construção deste projeto, e ao professor André que sempre esteve disponível para resolver questões relacionadas ao curso.

EPÍGRAFE

Não há nada a lamentar sobre a morte, assim como não há nada a lamentar sobre o crescimento de uma flor. O que é terrível não é a morte, mas as vidas que as pessoas levam ou não levam até a sua morte.

(Charles Bukowski)

RESUMO

JAKYMIU, Nicolas Cioczek. **Desenvolvimento de jogos na Engine Unity**. 2021. 90 f. TCC(Graduação) - Sistemas de Informação, Centro Universitário Vale do Iguaçu Uniguaçu, União da Vitória, 2021.

O mercado de jogos eletrônicos é um dos ramos mais promissores atualmente, apresentando resultados positivos mesmo mediante uma pandemia mundial. Devido ao enorme fluxo de projetos e acesso a informações, é necessário uma parametrização e utilização de ferramentas que auxiliem os profissionais desta área, para que seja obtido o máximo de eficiência e produtividade. Baseando-se nisso que este projeto apresenta o Unity como uma ferramenta especializada no desenvolvimento de jogos, apresentando todo o material necessário para criar projetos excepcionais e adequados ao mercado de videogames atual. Ao utilizar uma engine para desenvolvimento, o profissional recebe um ambiente especializado onde é possível realizar diversos tipos de operações de maneira simples e dinâmica. Além de optar pelo desenvolvimento no ambiente correto, o profissional deve optar por boas práticas, que auxiliem no trabalho em equipe e otimização do projeto. Como uma representação da utilização da ferramenta Unity, propõe-se a elaboração de um jogo em 3D, que apresenta algumas das mecânicas mais populares dos jogos eletrônicos.

Palavras-chave: unity, desenvolvimento de jogos, level-design, engine.

ABSTRACT

JAKYMIU, Nicolas Cioczek. Game development at Engine Unity. 2021. 90 f. TCC (Graduation) - Information Systems, University Center Vale do Iguaçu Uniguaçu, União da Vitória, 2021.

The electronic games market is one of the most promising sectors today, showing positive results even in the face of a worldwide pandemic. Due to the enormous flow of projects and access to information, it is necessary to parameterize and use tools that help professionals in this area, so that maximum efficiency and productivity can be obtained. Based on that, this project presents Unity as a specialized tool in game development, presenting all the necessary material to create exceptional projects suitable for the current video game market. When using an engine for development, the professional receives a specialized environment where it is possible to perform different types of operations in a simple and dynamic way. In addition to opting for development in the correct environment, the professional must opt for good practices that help in teamwork and project optimization. As a representation of the use of the Unity tool, it is proposed the elaboration of a 3D game, which presents some of the most popular mechanics of electronic games.

Keywords: unity, game development, level-design, Engine.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

NPC Non-playable character, Personagem não jogável.
HTML Hypertext Markup Language, Linguagem de marcação de hipertexto
CSS Cascading Style Sheets, Folhas de estilos em camadas
SQL Structured Query Language, Linguagem de consulta estruturada
UI User interface, Interface do usuário

Sumário

1. INTRODUÇÃO

Pode-se afirmar que a Indústria dos Jogos nasceu graças às inovações advindas da Revolução Industrial, que ocorreu por meados do século XVIII, e acarretou inúmeras mudanças em relação à área tecnológica. Na sociedade contemporânea, grande parte dos indivíduos já tiveram contato com jogos eletrônicos, isso em parte, é reflexo da exposição tecnológica presente desde o nascimento.

Pesquisas indicam que a Indústria dos Jogos movimentou mais de US$120 Bilhões, no ano de 2019. Os dados levantados pela empresa de análise de mercado SuperData, enfatizam a alta rentabilidade do ramo de jogos eletrônicos. A pesquisa contempla a compra de jogos em dispositivos móveis, consoles e computadores.

A profissionalização do ramo de desenvolvimento de jogos acarretou inúmeras ferramentas e ambientes voltados para a elaboração de jogos. Um exemplo são as Game Engines, que são ambientes específicos para o desenvolvimento, com uma série de componentes e elementos pré-estabelecidos que visam atingir aspectos positivos em relação à produtividade.

Portanto, o objetivo deste trabalho é abordar o desenvolvimento de jogos em uma Game Engine específica (Unity), apresentando todas as etapas de um projeto, conceitos e técnicas.

1.1 JUSTIFICATIVA

Os jogos eletrônicos representam uma forma de lazer extremamente popular nos dias de hoje. O que era inicialmente um entretenimento infantil, e tornou-se acessível para todas as idades, devido a grande quantidade de gêneros e plataformas.

A indústria dos jogos apresenta resultados extremamente positivos, onde é possível vislumbrar a alta rentabilidade desta área do comércio. É possível confirmar a popularidade dos jogos a partir de grandes redes sociais voltadas para usuários que transmitem suas partidas em tempo real, como a Twitch e o Youtube.

O desenvolvimento de jogos trata-se de um ramo comum de mercado, onde é necessário ter uma equipe, planejamento, orçamento e cronogramas. Com o passar dos anos houve aprimoramentos nesta área e tecnologias voltadas para o desenvolvimento de jogos para diversas plataformas.

Ao utilizar as ferramentas corretas, é possível obter o máximo em qualidade e eficiência. O desenvolvimento de jogos também contempla boas práticas de

programação, que quando utilizadas garantem que um objetivo seja atingido com o mínimo de esforço.

1.2 OBJETIVOS

1.2.1 **Objetivo geral**

Apresentar as etapas que constituem o desenvolvimento de jogos, e introduzir boas práticas de programação voltadas a eficiência de código e produtividade, a fim de gerar um ambiente organizado, com suas próprias peculiaridades e fatores que influenciam positivamente no desenvolvimento.

1.2.2 **Objetivos específicos**

- Enfatizar métodos e práticas de programação voltados para a área do desenvolvimento de jogos.
- Desenvolver um jogo orientado à objetos utilizando a linguagem C#.
- Apresentar a estruturação de um mapa com inimigos inteligentes.
- Demonstrar as funcionalidades de uma Game Engine.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 O HOMEM E A TECNOLOGIA

A tecnologia está presente na maior parte da sociedade moderna, seja na saúde, na educação ou na economia. As inovações que auxiliam no dia a dia da sociedade não existiam algumas décadas atrás, isso é porque a tecnologia é uma área de recorrentes mudanças, e isso se mantém desde os primórdios da humanidade.

Daniel R. (2009, p2) afirma que a evolução do homem ocorreu juntamente com o ambiente em que estava inserido, e com as ferramentas que ele utilizava. Nossos ancestrais foram capazes de mudar sua anatomia, a ponto de andar eretos e modificar ferramentas de acordo com suas necessidades.

O domínio do fogo agregou muito para os seres humanos na pré-história, por se tratar de uma ferramenta para assustar predadores, assar alimentos e manter-se longe do frio, segundo Daniel R. (2009, p3). A busca por alimentos era realizada com suas ferramentas com pontas afinadas, sempre em busca da presa mais fácil ou de restos deixados por animais selvagens.

Entretanto, o ato de gerar o fogo conscientemente é uma das habilidades desenvolvidas originalmente pela espécie Homo Erectus, que utilizavam carvão para realizar a fabricação do fogo, segundo Ian McNeil (1996, p7). Algumas tribos chegaram a utilizar o fogo, porém sua principal preocupação era manter a chama acesa, sem ter a noção do que fazer caso apagasse. O autor ainda afirma que o fogo serviu de base para a mineração e fundição, construção de meios de transporte e confecção de peças específicas, como o vidro.

> The change from nomadic Hunter to settled agricultural villager did not happen overnight, even over centuries. It must have taken several Thousand years. It started some time about 10,000 BC, When a great event took place-the end of the last Ice Age when the melting ice flooded the land and brought to life a host of plants that had lain dormant in seeds (Ian McNeil, p.11).

A citação acima apresenta um fenômeno, que segundo o autor, modificou os meios de sobrevivência do ser humano após a Era Glacial. Em decorrência das mudanças do ambiente surgiu a possibilidade para que a agricultura fosse desenvolvida.

Portanto, desde a antiguidade o ser humano adapta-se de acordo com suas necessidades, onde o fator que tem maior peso é a sobrevivência. O manuseio do

fogo, de minérios e de armas, só prova que as civilizações surgiram da capacidade do homem de aprimorar suas técnicas e processos.

Ian McNeil (1996, p. 5) afirma que um dos principais modelos históricos que utilizamos nos dias de hoje, foi amplamente desacreditado durante sua própria época. O arqueólogo Jurgensen Thomsen propôs um modelo das três idades pré-históricas, idade da pedra, bronze e do ferro, baseando-se nos registros de materiais utilizados em ferramentas relatadas na obra de Vedel Simonsen.

2.1.1 Civilizações

Avançando um pouco na história, chega-se às civilizações já existentes, que de acordo com Daniel R. (2009, p. 17), eram compostas por muitos indivíduos que contribuíram com tributos ao estado ou serviços. Traziam uma carga cultural totalmente diferente das vistas anteriormente, compostas por religiões, literaturas, sistemas de contagem de tempo, ciências exatas e humanas.

A vida em civilização possuía hierarquias sobre os cidadãos e a liderança. Existiam os encarregados para cuidar dos animais e da área de agricultura, liderar celebrações voltadas para a religião, organizar e treinar os militares e os responsáveis pela política, segundo Daniel R. (2009, p. 17). As classes sociais variavam entre servos, guerreiros, comerciantes e artífices. Onde cada um era responsável por uma área social, com deveres e direitos diferentes.

Figura 1 – Extração e utilização do minério de ferro.

Fonte: Conhecimento Científico (2020).

Daniel R. (2009, p. 36) afirma que o local onde o ferro foi fundido pela primeira vez, foi na Anatólia. Ao realizar tal prática, encontraram muitos problemas em decorrência dos materiais utilizados, os fornos eram simples demais e os poços de fundição não conseguiam atingir as temperaturas necessárias. O resultado obtido não foi perfeito, o ferro possuía imperfeições e não possuía durabilidade, mas o uso de ferro era incentivado devido a abundância do minério.

Levando-se em consideração esses aspectos é possível afirmar que a ramificação de novas áreas com o decorrer dos tempos, garantiram que novas técnicas minuciosas fossem desenvolvidas, como a fundição e sistemas hidráulicos para grandes civilizações.

De acordo com Ian McNeil (1996, p. 229) as três maneiras de obtenção de força de forma natural eram através da água, do vento e de animais. O uso da água se fez presente em mecanismos como moinhos de grãos, que eram movidos através da força da água corrente. Com certa semelhança ao modelo que utiliza a água, a força do vento era utilizada em moinhos, de forma que seu movimento poderia moer os grãos contidos no interior da instalação. Já a participação dos animais refere-se à utilização do animal em uma posição específica em referência ao eixo, e seu movimento constitui a geração de força.

2.1.2 Revolução Industrial

Segundo Philip Parker (2017, p. 264), a revolução industrial foi um fenômeno que aconteceu na Europa, no século XVIII, responsável por introduzir inovações tecnológicas e sociais. A sociedade foi introduzida em uma transição urbana, marcada pelos novos equipamentos voltados para as indústrias.

Os impactos que a revolução industrial trouxe perduram até os dias de hoje, segundo Daniel R. (2009, p. 91). Nossas interações com a indústria ocorrem nos mais diversos ambientes, desde a obtenção de alimentos ou produtos, até no que buscamos para entretenimento.

A grande quantidade de recursos naturais impulsionou a elaboração de novos projetos tecnológicos. Porém, foi na década de 1770 que uma grande inovação surgiu, a máquina a vapor, desenvolvida por James Watts. A importância dessa invenção é baseada nos usos industriais que são possibilitados pela geração de energia.

De acordo com Daniel R. (2009, p. 91) a industrialização segue quatro princípios fundamentais, que são:

1) Divisão de trabalho: reduzir um processo complexo em sub-rotinas simples que demandem menos esforços e tempo.

2) Utilização de máquinas: utilização do maquinário para substituir a mão de obra humana em áreas como a produção, controle de envio e registros.

3) Efetividade: Corresponde a uma observação dos dois primeiros princípios, focada na obtenção da lucratividade com o menor índice de gastos.

4) Fósseis: É a área da geração de energia através de fontes não-renováveis.

Um dos pontos negativos da revolução industrial foi a situação sob a qual os trabalhadores foram submetidos. A indústria têxtil trabalhava em grande escala com um exorbitante número de funcionários, e a situação era de exploração, cargas horárias excessivas juntamente com salários miseráveis, afirma Philip Parker (2017, p. 264).

Figura 2 – Cena do filme Tempos Modernos.

Fonte: Poder 360 (2020).

A imagem acima foi retirada do filme “Tempos Modernos” (1936), que mostra a situação de trabalho em uma indústria dos Estados Unidos na década de 1930. A abordagem do filme é cômica, mas é possível analisar o certo tom de crítica pelo abuso e total descaso ao funcionário.

De acordo com Philip Parker (2017, p. 265) as práticas de produção conquistadas pela revolução, se disseminaram rapidamente entre as grandes potências industriais, facilitadas pela introdução de ferrovias, que auxiliaram no transporte de cargas e mão de obra. O autor ainda afirma que alguns anos adiante Henry Bessemer foi o criador de um processo responsável por fomentar a criação de

linhas ferroviárias, seu método para tratar o ferro o tornava de maior qualidade e maior versatilidade.

Philip Parker (2017, p. 268) afirma que a revolução industrial representou um grande impulsionamento tecnológico, onde as inovações que surgiram variaram de inovação em meios de transporte (carro, aviões, trens), na comunicação (rádio, telefone) e na vida doméstica. Entretanto, as evoluções não se limitaram à tecnologia, visto que foi nesta época que Charles Darwin desenvolveu a Teoria da evolução.

Muitas pessoas acreditavam que a industrialização surgiria pela China, devido à alta capacidade tecnológica da época. Porém, a Grã-Bretanha possuía algo que a China não tinha, uma fonte de energia barata. Enquanto a China estabeleceu um ritmo estagnado de produção, a Grã-Bretanha possuía um amplo reservatório de carvão, cobre e ferro, segundo Daniel R. (2009, p. 91).

2.2 EVOLUÇÃO DOS COMPUTADORES

De acordo com Gerard O' Reagan (2016, p2), o computador é uma máquina eletrônica que pode ser programada, suas funções variam do trato de informações ao armazenamento e backup. O funcionamento de um sistema age de acordo com as instruções que ele recebe durante a sua fase de desenvolvimento. O autor ainda afirma que a constituição física de um computador pode ser dividida em duas partes: hardware e software. O hardware corresponde a parte física, os componentes interligados que se comunicam através de barramentos de conexão. Já o software é o conjunto de operações que geralmente é descrito em forma de processos para realizar alguma tarefa.

> The original meaning of the word computer referred to someone who carried out calculations rather than an actual machine. The early digital computers built in the 1940s and 1950s were enormous machines consisting of thousands of vacuum tubes. They typically filled a large room, but their computational power was a fraction of the personal computers used today. (Gerard O' Reagan, p.2).

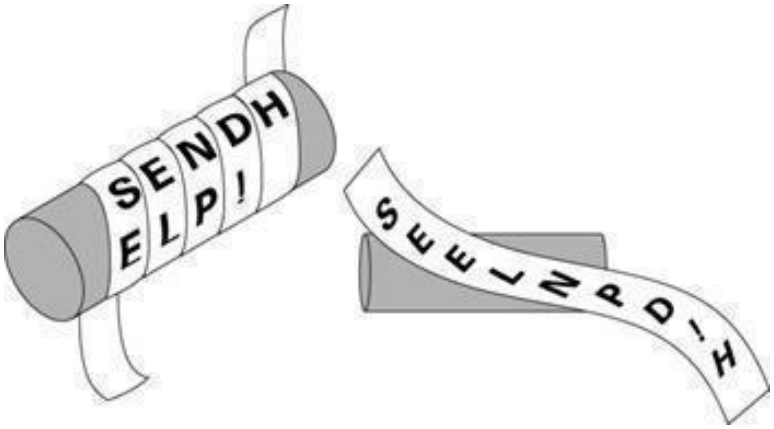
A citação acima representa a descrição do autor sobre a nomenclatura "computador", que foi utilizada pela capacidade da máquina de realizar cálculos. O autor ainda afirma que os primeiros computadores necessitavam de um amplo local de armazenamento para os equipamentos, e possuíam o poder de processamento muito inferior ao que temos em dispositivos móveis atualmente.

Na sociedade atual, a obtenção de um computador se tornou praticamente obrigatória, seja para trabalho ou simplesmente por lazer. Gerard O' Reagan (2016, p1) afirma que a utilização dos computadores modernos com a internet, mudaram a maneira de comunicação global. O crescimento das redes sociais é notável, e há diversas formas para se comunicar com amigos, familiares ou desconhecidos no grande ambiente da internet.

Segundo Simson L. Garfinkel (2018, p. 21), o Ábaco foi a primeira ferramenta desenvolvida para realizar cálculos. Onde era possível utilizar os operadores matemáticos de adição, subtração e divisão para chegar nos valores desejados. O autor afirma que o surgimento do Ábaco em parte se deve aos fatores sociais em que os sumérios estavam inseridos. As cidades da Mesopotâmia possuíam economias estabelecidas e um grande fluxo comercial, a partir daí que surge a necessidade de um instrumento capaz realizar cálculos matemáticos.

Sistemas eram desenvolvidos com base no surgimento de operações específicas. Os romanos desenvolveram um método para criptografar mensagens em um ambiente de guerra, caso a mensagem caísse nas mãos inimigas, não teria validade alguma, afirma Simson L. Garfinkel (2018, p24). O mecanismo era composto por dois bastões e uma faixa de pergaminho, a escrita ocorria ao inserir a faixa através do bastão, e ao tirá-la a mensagem estaria criptografada até que fosse novamente inserida novamente no objeto plano.

Figura 3 - Sistema romano de criptografia.



Fonte: L' Arbre de Noa (2013).

Novos tipos de computadores emergiram com o passar dos anos. Foram encontrados indícios na Grécia, sobre um computador analógico capaz de calcular

eventos astronômicos. O equipamento possuía diversas engrenagens e representava ser um modelo complexo, afirma Simson L. Garfinkel (2018, p. 27).

2.2.1 Modelo binário

Um sistema numérico popularmente conhecido pelos amantes de tecnologia é o sistema binário. Gottfried Wilhelm Leibzin foi um filósofo e matemático, responsável por estudar e introduzir um modelo sistêmico que utilizava apenas dois dígitos, ‘0’ e ‘1’, segundo Gerard O’ Reagan (2016, p 40). Apesar da utilização de apenas dois dígitos, era possível realizar uma infinidade de cálculos utilizando as operações matemáticas como a adição, subtração, multiplicação e divisão, afirma o autor.

Segundo Gerard O’ Reagan (2016, p 40), Leibzin já visualizava as vantagens que o sistema binário iria oferecer aos sistemas digitais. Segundo o matemático, a representação dos números seria simplificada pela utilização de módulos, um módulo com o número “1” representaria ligado, enquanto com “0” seria desligado. Com a utilização dos relés e lógica booleana, foi possível implementar circuitos cada vez mais complexos.

Figura 4 – Conversão binária, decimal e hexadecimal.

Binário	Hexadecimal	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Fonte: FILIPEFLOP (2018)

O idioma pelo qual um computador se comunica com seus componentes é o sistema binário, afirma Simson L. Garfinkel (2018, p. 39). Popularmente conhecidos como bits, realizam a representação de valores que em combinação retornam um valor específico.

A figura (4) acima representa a conversão dos valores binários para o sistema de numeração hexadecimal. De modo que o sistema binário utiliza apenas “0” e “1”, o sistema hexadecimal utiliza desde “0” até “F”.

2.2.2 Máquina diferencial

Simson L. Garfinkel (2018, p. 49) afirma que uma inovação extremamente significativa surgiu pelas mãos de Charles Babbage, a máquina diferencial. Este computador foi projetado com o objetivo de resolver funções polinomiais, e posteriormente abrindo caminho para a geração de logaritmos.

O projeto da máquina diferencial não foi totalmente realizado por Babbage, mas foi ele quem realizou um projeto minucioso utilizando os conceitos originados por Helfrich Von Müller. O projeto de Charles Babbage era de extrema complexidade para a época, e suas aplicações em protótipos foram funcionais, afirma Simson L. Garfinkel (2018, p. 49).

Charles Babbage e George Boole são considerados os pioneiros na área de computação, afirma Gerard O’ Reagan (2016, p 41). Babbage contribuiu em diversas áreas das ciências exatas, além de conquistas relacionadas a filosofia e desenvolvimento ferroviário.

> Babbage was interested in accurate mathematical tables as these are essential for navigation and scientific work. However, there was a high error rate in the existing tables due to human error introduced during calculation. He became interested in the problem of finding a mechanical method to perform the calculations, as this would eliminate errors introduced by human calculation. Babbage wished to develop a more advanced machine than Pascal’s Pascaline or Leibniz’s step reckoner, which were limited to the basic arithmetic operations. He wished to develop a machine that could compute polynomial functions. Gerard O’ Reagan (2016, p 41)

A citação acima apresenta os motivos pelos quais Charles Babbage desenvolveu a Máquina Diferencial, seu principal objetivo era corrigir os erros através da geração de dados precisos, obtidos através de cálculos complexos. Até o momento as máquinas baseavam-se apenas em cálculos utilizando as operações básicas.

Portanto, para que um computador dos dias de hoje tivesse a capacidade de realizar milhares de cálculos por segundo, foram necessários anos de trabalho árduo e dedicação de gênios que oferecem seus serviços para a área de computação.

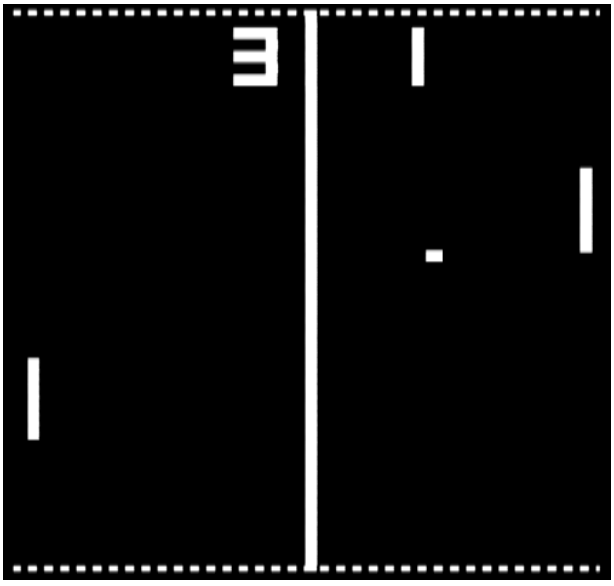
2.3 A EVOLUÇÃO DOS JOGOS

2.3.1 Pong

Os jogos atuais atingiram um nível inimaginável algumas décadas atrás, o nível de detalhamento e ambientação criam imagens que confundem o usuário entre um videogame e um cenário real. Além da parte visual, a programação dos jogos também avançou muito, com técnicas complexas sobre detalhes técnicos, como iluminação, câmeras, movimentos, e outros dados pré-estabelecidos.

Dustin Hansen (2016, p. 12) apresenta um dos jogos de maior reconhecimento na história do entretenimento, o Pong. Apesar de não ser o jogo mais bonito da época, ou com as melhores mecânicas, este jogo é reconhecido como o avô dos jogos eletrônicos por alguns fatores como a simplicidade, era possível ter contato com o jogo sem maiores instruções, era simplesmente colocar a ficha e jogar. Outro fator determinante era o ambiente, inicialmente instalado em fliperamas, o jogo Pong era acessível por alguns trocados de moeda em estabelecimentos públicos, onde era possível reunir os amigos para divertir-se com o equipamento.

Figura 5 - Pong.



Fonte: Gamasutra (2009).

O período em que o surgimento de jogos começou a ganhar evidência, foi marcado com a disputa pelo lançamento dos vídeos games. Um dos dilemas da

época, era que para criar um jogo, era necessário equipamentos de hardwares avançados, que possuíam custos absurdos, segundo Dustin Hansen (2016, p.2).

Segundo Dustin Hansen (2016, p.2), Nolan Bushnell foi um dos pioneiros no desenvolvimento de jogos, seu primeiro contato com este mundo foi em sua universidade, onde pode jogar o famoso jogo SpaceWar, um jogo de guerra espacial que colocava o jogador em um espaço de tiros e colisões.

A ideia de Bushnell caiu nas graças do povo, logo, o engenheiro teve que contratar uma equipe e treiná-la para expandir sua capacidade de criar os fliperamas com o jogo de tênis eletrônico. Essa foi a origem do Atari, juntamente com a popularização dos jogos arcade, afirma Dustin Hansen (2016, p.2).

Steven L. Kent (2001, p.80) afirma que os primeiros jogos desenvolvidos e registrados foram desenvolvidos pelos engenheiros da empresa Sanders Associates, cujos principais nomes eram Ralph Baer e Bill Rusch. Em seguida dos avanços decorrentes do desenvolvimento de jogos, a equipe de Ralph e Bill Rusch levaram a equipe a desenvolver um protótipo de jogo multijogador e multiprograma, conhecido como “Brown Box”.

A Atari foi uma das empresas pioneiras no desenvolvimento de jogos, e a maneira pela qual organizou seus colaboradores ainda é modelo para algumas empresas dos Estados Unidos, afirma Dustin Hansen (2016, p.18). O foco da empresa foi criar um ambiente que agradasse os funcionários para que sua produtividade fosse atingida através da satisfação.

Dustin Hansen (2016, p.18) afirma que a Era de Ouro dos Jogos eletrônicos foi essa etapa em que os escritórios de jogos se tornaram centros criativos, onde os funcionários poderiam ter vários campos para lazer enquanto trabalhavam no desenvolvimento de um novo projeto.

> The game was built to impress. The refrigerator-sized arcade machine was covered top to bottom in artwork to set the mood, and it featured two white buttons, one to move left AND one to move right! Next to the twin direction buttons, you’d find a superslick firing button the color of danger itself, red. But what you couldn’t see was the massive speaker hidden inside the beast, which pumped out a sound track some say inspired Jaws when it hit the big screen five years later. Dustin Hansen (2016, p.21)

Na citação acima, o autor descreve a sensação do primeiro contato com as máquinas de fliperama. A estrutura física é retratada como uma máquina grande com botões direcionais para realizar as ações de movimento e ação do personagem ou objeto.

2.3.2 Space Invaders

De acordo com Dustin Hansen (2016, p.18), os jogos de apenas um jogador emergiram com o lançamento do Space Invaders. De naves espaciais a raios lasers, o lançamento do fliperama garantiu a alegria durante o verão do ano de 1978. O jogador contava com 3 vidas por ficha utilizada, e essas três unidades poderiam se estender por horas, dependendo do nível de habilidade do jogador.

O funcionamento do jogo era baseado em uma invasão alienígena, como o próprio nome sugere, onde os seres de outro mundo realizavam formações de batalha para destruir as torres de defesas humanas, caso as torres fossem destruídas ou os inimigos atingissem a parte inferior da tela, significava derrota para o jogador, afirma Steven L. Kent (2001, p.164).

A contagem dos pontos neste jogo ocorre após uma intensa batalha em que o jogador tem a sua disposição torres de lasers. Porém, os inimigos não são indefesos, também possuem poder armamentista, que pode ser defendido por instalações temporárias presentes na base do jogador. Ao destruir uma horda completa de alienígenas, será creditado uma quantia específica de pontos ao jogador, garante Steven L. Kent (2001, p.164).

Figura 6 – Fliperama Invaders.

Fonte: Bojoga (2018).

Entretanto, a popularidade do jogo espacial não foi tão repentina quanto esperava-se, no Japão foram cerca de três meses para que o jogo desses indícios de aproveitamento do público, e em questão de um ano, a utilização do fliperama nos Estados Unidos era uma febre sem precedentes, afirma Steven L. Kent (2001, p.164).

O autor ainda comenta uma das medidas que o governo do Japão teve de tomar em decorrência da popularização dos Fliperamas. A moeda de 100 Ienes estava sendo amplamente utilizada, e por medidas governamentais, se fez necessário o aumento da produção, devido à saturação.

A popularização do Space Invaders foi possível devida à produção em alta escala, cerca de quatrocentos mil fliperamas foram construídos, e lucro bruto de aproximadamente 3,8 bilhões de dólares, que até mesmo nos dias de hoje, esse valor representa uma quantia exorbitante, afirma Dustin Hansen (2016, p. 22).

Mesmo após o sucesso do jogo, a empresa fabricante dos fliperamas Taito, observou que os temas mais pautados nos Estados Unidos eram os que simulavam a vida real, como esportes, direção e guerra. Já com Space Invaders, o tema principal era algo voltado para a ficção e ciência, de acordo com Steven L. Kent (2001, p.164)

Segundo Dustin Hansen (2016, p.22), a mente por trás do jogo Space Invaders é Tomohiro Nishikado, ele realizou grande parte do trabalho sozinho, desde as artes e o design, até a composição de hardware necessárias para rodar o jogo.

Tomohiro Nishikado é o responsável por uma lista de inovações na área de desenvolvimento de jogos, segundo Dustin Hansen (2016, p.23):

1) Introdução do sistema de pontuação classificado pelos maiores números/tempo de sobrevivência.
2) Salvamento de dados: os valores mantinham-se registrados na máquina e poderiam ser consultados a qualquer momento.
3) Modo sem fim: Era um modo infinito, onde os inimigos não paravam de retornar, com o aumento da dificuldade.
4) A introdução de um sistema de tiros, onde era possível esquivar-se utilizando barreiras disponíveis no ambiente.
5) A música tocando no fundo do jogo era uma combinação de notas em loop. Conforme a velocidade crescia, o ritmo da música também aumentava.

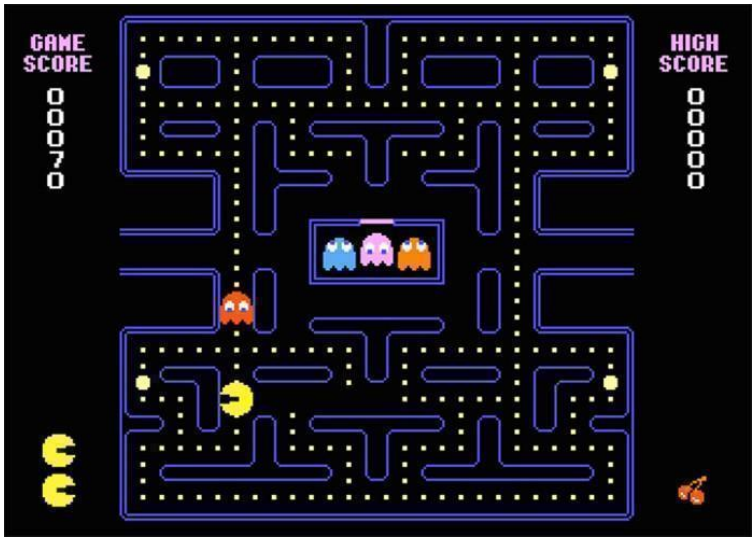
2.3.3 Pac-man

Segundo Steven L. Kent (2001, p.188), o desenvolvimento do jogo Pac-Man foi originado das ideias de Toru Iwatani, um recém-formado que ingressou na Namco. A ideia que estava presente em sua mente era desenvolver um jogo que não fosse violento, e agradasse ao sexo feminino. Iwatani abordou uma maneira de

desenvolvimento um pouco diferente do normal, ele decidiu criar um jogo em torno da palavra japonesa *taberu*, que possui o significado “comer.”

Iwatani foi encaixado em uma equipe de nove profissionais, onde o primeiro feito da equipe foi definir como seria o personagem principal. A origem do personagem ocorreu durante uma sessão de almoço, Iwatani observou uma pizza em uma caixa, onde existia a lacuna de um dos pedaços, e assim nasceu o Pac-Man, afirma Steven L. Kent (2001, p.188). Em seguida, a escolha dos personagens foi decidida com foco no público pretendido, decidiram criar fantasmas coloridos, pois segundo eles, seria agradável para o público feminino.

Figura 7 – Pac-Man.



Fonte: TechTudo (2015).

De acordo com Steven L. Kent (2001, p.187), conforme citado por Toru Iwatani, “The actual figure of Pac-Man came about as I was having pizza for lunch. I took one wedge and there it was, the figure of Pac-Man.”. Toru já participou de várias entrevistas e afirma a autenticidade de que a informação sobre a criação do boneco Pac-Man é advinda de um almoço.

Dustin Hansen (2016, p.23) descreve que a maneira como Toru descobriu como seria seu personagem, é uma das etapas do processo de criação, nem sempre uma ideia mágica irá surgir na cabeça do desenvolvedor, esta etapa requer muita criatividade e atenção ao mundo externo em busca de novos horizontes.

Steven L. Kent (2001, p.188) apresenta uma curiosidade sobre mudanças referentes ao nome do jogo. Originalmente como Puck-Man, cujo nome tinha

referência pela forma de disco do personagem, transformou-se em Pac-Man em decorrência da análise que os criadores realizaram a respeito de uma possibilidade de alterar algumas letras no nome e tornar o personagem de pizza em algo obsceno.

Em referência ao funcionamento do jogo, Steven L. Kent (2001, p.187) afirma que o jogo possuía uma forma simples de controle, com um joystick era possível mover o personagem em um labirinto, em busca dos pontos que estavam espalhados. O objetivo do jogo era esquivar-se dos fantasmas, enquanto limpava o mapa todo. Existiam pílulas de energia que permitiam que o Pac-Man devorasse os fantasmas, e caso o personagem estivesse fora do efeito da pílula e colidisse com um desses obstáculos móveis, ele morreria. Também era possível utilizar um sistema de pontos extras, onde frutas e outros objetos apareciam no mapa e contabilizavam caso o Pac-Man conseguisse ingeri-los.

> Another big difference after the shift from Japan to America was Pac-Man's reception. In Japan, the game started off slow, but in America it was an overnight success. So successful, in fact, that it surpassed Space Invaders, the current king of the arcade hill, by earning over a billion dollars in quarters in its first year alone. By the early 1990s, the always-hungry Pac-Man had brought in over 2.5 billion dollars just in quarters. That is 125,000 pounds of quarters! Dustin Hansen (2016, p.30)

A citação acima descreve a recepção do público ao jogo Pac-Man, onde no Japão foi lentamente caindo nas graças do povo, enquanto nos Estados Unidos a euforia originou-se no seu lançamento. Os resultados foram otimistas, o faturamento chegou a mais de dois bilhões de dólares em apenas um trimestre.

Steven L. Kent (2001, p.187) afirma que a indústria dos jogos passou por diversas mudanças após o sucesso do jogo Pac-Man. A primeira foi o crescente interesse no tema de labirintos, em um ambiente onde a maior popularidade era destinada a atirar em alienígenas, guiar carros ou praticar esportes.

Dessa forma, os fliperamas estavam inseridos nas cidades, eram as principais atrações em diversos centros comerciais. Em parte, pelo rendimento que era possível se obter adentrando no ambiente de jogos, e de outro lado era necessário se ater às novas tecnologias para que seu negócio não adquirisse uma desvantagem em relação ao comércio local.

2.3.4 Donkey Kong

Dustin Hansen (2016, p.50) afirma que a Nintendo teve muita expectativa em um fliperama chamado Radar Scope, cujo tema era ação espacial. Porém, devido a baixa resposta do público sobre o videogame, os três mil equipamentos que foram enviados pela Nintendo, foram desativados para correções.

Por volta de 1980, o líder da Nintendo tomou uma decisão para mudar o rumo do Radar Scope, afirma Dustin Hansen (2016, p.50). Ao definir Shigeru Miyamoto, um designer, para assumir a nova identidade do projeto. Miyamoto realizou uma análise sobre as tendências da época e presumiu que um dos elementos que faltavam era a história. Boatos sugerem que Miyamoto se inspirou em Popeye para idealização dos personagens e relações. O que foi definido pelo designer era que o enredo se manteria em torno de um macaco, uma princesa aguardando seu resgate, e um carpinteiro que conseguia se esquivar de obstáculos.

> There are a lot of stories about where the name Donkey Kong actually came from—everything from a bad fax that made the Nintendo of America team misread Monkey Kong, thinking the M was a D, to its being named after King Kong. But in the end, Miyamoto said it was simpler than that. They wanted an English name because they knew the game would be a hit in America. The word donkey was used to imply something silly, or dumb, and in Japan, kong is a slang word used for an ape. Basically, Miyamoto and crew were naming the game Silly Ape, but they felt Donkey Kong was, well, just more fun to say. Dustin Hansen (2016, p.54)

A citação acima refere-se a discussão sobre a origem do nome Donkey Kong, onde muitos acreditaram tratar-se de um erro de digitação, na troca de “M” por “D”, enquanto outros acharam que o Kong se tratava de uma referência ao grande macaco dos cinemas, King-Kong. O autor apresenta o posicionamento de Miyamoto defendendo que a nomenclatura do jogo foi idealizada com uma simples visão, a futura popularidade. Eles acreditavam que o jogo seria muito popular na América, então utilizaram termos populares em inglês, onde Donkey significa burro e Kong é uma gíria chinesa para Macaco.

De acordo com Steven L. Kent (2001, p.203) a base do Donkey Kong já existia, os grandes arcades que rodavam o Radar Scope. Miyamoto precisava ater-se às ideias sobre o jogo e seus novos rumos. Uma equipe de Nintendo de engenharia foi destinada a converter as ideias de Miyamoto em um jogo.

> Like Eugene Jarvis, Miyamoto began by inventing an elaborate story to explain his game. The story involved a gorilla escaping from its master, a carpenter, and kidnapping his girlfriend. First the gorilla climbed to the top of a seven-story construction site. When his master followed, the ape rolled barrels at him. Players helped the carpenter leap over the barrels as he followed the gorilla. Steven L. Kent (2001, p.203)

A citação acima refere-se ao desenvolvimento da história do jogo. Miyamoto elaborou um roteiro onde uma mulher era sequestrada por um Gorila, e um carpinteiro que estava em busca de resgatá-la. O objetivo do jogo era esquivar-se de barris lançados pelo Gorila e chegar ao topo de uma estrutura, onde a namorada do carpinteiro estava sendo mantida, afirma Steven L. Kent (2001, p.203).

Figura 8 – Donkey Kong.

Fonte: TechTudo (2018).

Dustin Hansen (2016, p.52) apresenta algumas características interessantes sobre o lançamento do Donkey Kong:

1) Observando a popularidade do jogo, a Nintendo concretizou vários outros projetos de jogos no modelo de plataforma. Além de servir de inspiração para diversos outros como Mega Man e Sonic.
2) A nova mecânica: era notável que nos outros jogos o sistema de locomoção era feito de maneira linear, com a vinda do Donkey Kong a utilização de saltos se fez presente. Pode parecer uma função simples pela visão atual da indústria dos jogos, mas para a época foi um grande diferencial.
3) O primeiro jogo de sucesso de Miyamoto: Apesar de já estar envolvido em outros projetos, o que garantiu maior renome na comunidade de

desenvolvimento de jogos, com certeza foi o Donkey Kong. Servindo como uma enorme bagagem para o futuro promissor de Miyamoto.

4) A origem do Mario: Mario é um outro jogo de extrema popularidade, e seguem os rumores que este é o carpinteiro do jogo Donkey Kong.

Miyamoto possuía gostos específicos por projetos de brinquedos, além do seu gosto pela música e especificamente dos Beatles. Mesmo após ter crescido no mercado dos designs de jogos eletrônicos, quando questionado, Miyamoto ainda garante ter interesse por inserir-se na área de brinquedos infantis, afirma Steven L. Kent (2001, p.203).

2.3.5 Super Mario Bros

Segundo Dustin Hansen (2016, p.66), a indústria dos jogos estava passando por um problema grave, por volta do ano 1982. Devido a alguns escândalos envolvendo a empresa Atari, a credibilidade de muitas outras foi colocada em risco. O apoio que o público destinava anteriormente, tinha se tornado um questionamento sobre a durabilidade da mania dos jogos eletrônicos.

> In fact, in 1982 more consoles were released by more companies than in any other year in video game history. Emerson released the Arcadia 2001, Coleco shipped the ColecoVision, Milton Bradley got into the game with their Vectrex, and just to keep it interesting, Coleco put out a second console, the Gemini. 1982 was a busy year. Dustin Hansen (2016, p.66)

A citação acima representa os esforços que as empresas realizaram para lançar novos consoles. Marcas como a Emerson, Coleco e Milton Bradley ofereceram novos dispositivos que segundo o autor só fomentaram a confusão entre o público, onde a conversão dos fliperamas para as residências não ocorreram de forma orgânica.

De acordo com Dustin Hansen (2016, p.66), a Nintendo tentou manter-se à frente da situação atual do mercado. Mesmo com a baixa em vendas de dispositivos residenciais, a Nintendo seguiu firme e forte na busca por segmentos que aceitassem a comercialização dos videogames, até que finalmente em Nova York, conseguiram um lugar que conseguiu comercializar uma pequena quantidade de dispositivos.

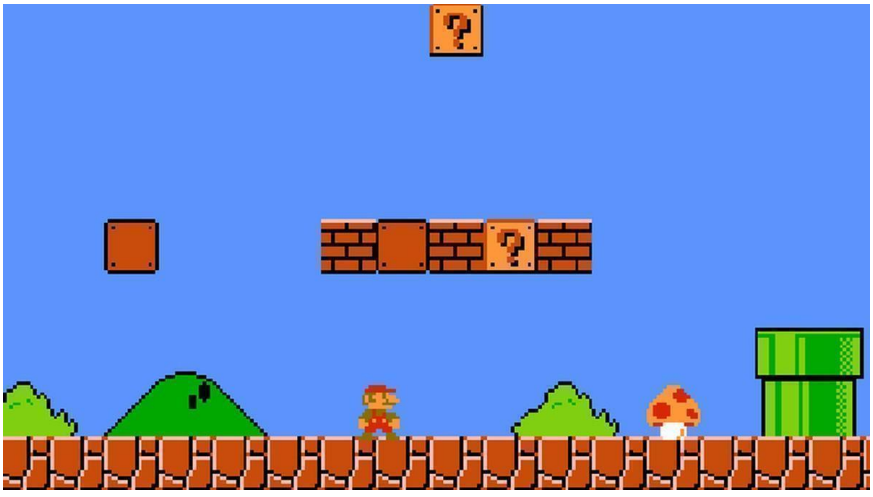
Steven L. Kent (2001, p.233) afirma que o retorno do Miyamoto ocorreu com seu trabalho relacionado ao desenvolvimento de um outro jogo, onde um gorila sequestrava uma princesa, e o carpinteiro que atravessava diversas fases diferentes antes de derrotar o vilão e salvar a princesa.

A mecânica e ambientação do jogo eram totalmente diferentes do Donkey Kong, pois não se tratava mais de uma cena fixa onde os eventos ocorriam, e sim de uma ampla paisagem linear onde o boneco poderia percorrer, enquanto a câmera o acompanhava. Segundo o autor, essa mecânica de movimentação de câmera veio a ser nomeada como “rolagem lateral”, por Arnie Katz, funcionário da empresa Electronic Games.

> The first step in Jumpman’s evolution was a name change—he became Mario in his next outing, a game called Donkey Kong Junior. Next, Miyamoto gave Mario a brother named Luigi and converted him from a carpenter into a plumber in the 1983 game Mario Bros. Up to this point, Mario always appeared in small games in which you could place the entire field of play on a single screen. That changed in 1985, with the release of Super Mario Bros. Steven L. Kent (2001, p.333)

A citação acima apresenta a evolução do personagem Jumpman, que se tornou Mario. O título Mario Bros tem relação com a introdução de Luigi, irmão do encanador saltitante, em 1985 os irmãos foram oficialmente liberados para os videogames.

Figura 9 - Super Mario Bros.



Fonte: TechTudo (2017)

De acordo com Steven L. Kent (2001, p.333), o principal objetivo do jogo era guiar o encanador Mario na busca pela princesa Peach, que havia sido sequestrada

por Bowser (o inimigo). Durante o decorrer do jogo, fases diferentes eram alcançadas e inimigos mais difíceis apareciam, os principais eram tartarugas e cogumelos denominados como Goombas.

Super Mario Bros trouxe uma elementos que vivem até nos dias de hoje, os “Easter Eggs”. Numa tradução direta, entende-se algo parecido com “Ovos de Páscoa”, o termo popularmente utilizado em inglês refere-se a elementos ocultos, que necessitam a exploração do jogador ou de alguma interação extra ao jogo. Em Super Mario Bros, havia a possibilidade de achar fases secretas, moedas, itens e poderes especiais, segundo Steven L. Kent (2001, p.333).

> By the end of the year, Nintendo engineers succeeded in creating a home version of Super Mario Bros. for the Famicom. This product defined the difference between games for the old Atari systems and games that could be played on Nintendo cartridges. Although the home version of Super Mario Bros. was not identical to the arcade game, it was an extremely close approximation. Steven L. Kent (2001, p.300).

A citação acima refere-se à adaptação do jogo Super Mario Bros para os dispositivos residenciais, através de cartuchos. O sucesso do jogo aos fliperamas foi sucedido por uma versão extremamente semelhante, vendida em cartuchos.

Miyamoto conseguiu a atenção do presidente da Nintendo com o Mario Bros, o jogo era exatamente o que Yamauchi queria, uma maneira de aumentar a venda dos videogames, especificamente falando do Nintendo Entertainment System, afirma Dustin Hansen (2016, p.68).

Um outro diferencial apontado por Dustin Hansen (2016, p.67) foi que as equipes de desenvolvimento eram normalmente compostas por engenheiros, cujo principal foco era programar um jogo funcional e divertido. Apesar da visão deles ser válida, um outro horizonte surgiu com a presença de Miyamoto, cuja formação também contemplava conhecimentos artísticos. A ambientação dos jogos na maioria dos casos era minimalista e monocromática, porém o Super Mario era o total oposto desse conceito. Com cores vivas, personagens visualmente agradáveis e paisagens naturais figura (6).

A influência do Super Mario Bros ultrapassou barreiras, entrando na lista dos jogos mais vendidos de todo o tempo. Sua música tornou-se um ícone, facilmente identificável e popular até os dias de hoje. Além de jogos 2D e 3D, a série de jogos

chegou até as televisões como desenho animado, e nos cinemas através do filme, afirma Dustin Hansen (2016, p.68).

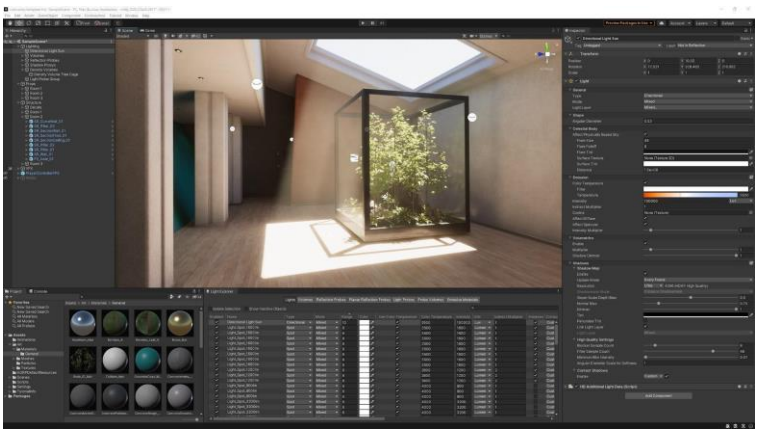
2.4 DESENVOLVIMENTO DE JOGOS

2.4.1 Game engine

Oliver Düvel; Stefan Zerbst (2004, p.4) fazem uma analogia ao termo “Engine”, que em português significa “Motor”. O motor de um carro é utilizado para impulsionar um veículo através da junção dos componentes e alguma força motora. Enquanto um motor de jogos é a parte do projeto que introduz e garante que as funcionalidades realizem determinadas tarefas. Uma série de mecanismos que correspondem a modelos 2D e 3D, onde o programador não necessita possuir o conhecimento técnico sobre a programação de baixo nível, o motor se encarrega por esta tarefa.

Jason Gregory (2009, p.11) apresenta a origem do termo “Game Engine”. Em meados da década de 1990, em decorrência a uma prática realizada em jogos de tiro, como o popular jogo Doom, inúmeros outros projetos começaram a modificar suas formas de desenvolvimento. Segundo o autor, o jogo Doom foi esquematizado com módulos de separação de componentes (áudios, gráficos e físicas).

Figura 10 – Desenvolvimento em uma engine.



Fonte: Blender (2020).

A partir disso, surgiu a possibilidade da criação de novos produtos, efetivando mudanças mínimas na ferramenta base. É semelhante a técnica de reaproveitamento de código em programação, porém uma game engine estabelece as ferramentas e permite que o usuário as modifique a seu gosto.

Algumas equipes de desenvolvimento e estúdios independentes iniciaram nas ramificações do desenvolvimento de jogos, com base em jogos já existentes e com a utilização de conjuntos de predefinições estabelecidas gratuitamente pelos desenvolvedores, afirma Jason Gregory (2009, p.11).

Oliver Düvel; Stefan Zerbst (2004, p.4) afirma que definir com exatidão o conceito de *engine* é relativamente difícil, pois cada programador possui uma definição sobre o funcionamento e os aspectos. Porém, o autor afirma ser capaz de citar algumas funcionalidades que estão presentes na maioria das situações: Controle e cálculo de dados, comunicação entre instâncias e interação com dados advindos de instâncias.

> The most complex engine is an 3D engine. The days of pumping 3D data into a graphics adapter's 3D pipeline are long gone because the graphic accelerator cards do more jobs on their own inside the 3D pipeline. Nowadays a 3D engine needs to organize things in a way suitable to speed things up. The so-called object and scene management is one of the most important parts. In a perfect situation, the user would name scene data to the engine and the engine itself would take care of organizing and structuring the data. After initialization time, the user would then need to make only some render calls and not care about state switches and similar things.The engine should catch those render calls and apply them in the most performant way. Oliver Düvel; Stefan Zerbst (2004, p.6)

No trecho acima, os autores afirmam que os motores de jogos mais complexos são os destinados ao desenvolvimento em 3D. Onde espera-se que exista um gestor de objetos nativo para processar e controlar os elementos inseridos pelo desenvolvedor.

De acordo com Jason Gregory (2009, p.11), as empresas em diversas situações optam pelo licenciamento de um mecanismo próprio de desenvolvimento de jogos, onde é possível estabelecer métodos e utilizá-los em qualquer momento do desenvolvimento. Porém, o autor enfatiza que é necessário um razoável investimento em engenharia de software para desenvolver um Motor de Jogos (Game Engine).

2.4.2 Conflito

Evan Skolnick (2014, p.12) afirma que um conceito importante relacionado ao desenvolvimento de jogos é o conflito. Para que uma história seja curiosamente atrativa para o público, é necessário estabelecer pontos de interesse. Às vezes podem estar inseridos em tramas menores, maiores ou subtramas. O autor afirma que o

conflito é algo extremamente importante na experiência de desenvolver um conto. Além de uma história atrativa a ser desenvolvida, objetivos e obstáculos contribuem para que o jogador se sinta desafiado a concluir o jogo.

Alguns jogos modernos e suas tramas principais:

1) Life is Strange (Square Enix, 2015): Max volta a sua cidade natal, e começa a presenciar eventos sobrenaturais ao reencontrar sua amiga. Max precisa entender o que está acontecendo com ela, enquanto auxilia sua amiga Chloe na busca por uma pessoa desaparecida.
2) The Witcher 3 - Wild Hunt (CD Projekt RED, 2015): Geralt de Rívia é um bruxo, que encontra sua parceira após um longo período, ambos recebem a missão de localizar Ciri, que foi aprendiz de Geralt muitos anos atrás. Porém, forças sobrenaturais dificultam a busca por Ciri, forçando Geralt a se aventurar por diversos reinos.
3) Sekiro: Shadows Die Twice (FromSoftware, 2019): Trata-se da história de um ninja que praticamente encontra a morte. Ao ser resgatado, o guerreiro é incumbido da missão de proteger o remanescente de uma antiga linhagem nobre, na luta contra poderosos inimigos.
4) Doom (Bethesda Softworks, id Software, Activision +, 2016): A extração de recursos naturais do inferno acarretou uma revolta demoníaca, um fuzileiro chamado DoomSlayer é responsável pela missão de viajar até marte e eliminar as colônias de demônios.
5) Assassin's Creed Odyssey (Ubisoft, 2018): Kassandra ou Alexios, (personagens selecionáveis), são espartanos que foram considerados amaldiçoados e jogados à deriva por conta de uma profecia. Após crescidos, eles retornam e descobrem conspirações referentes a sua família. Cabe ao personagem principal encontrar as possíveis ameaças e entender a atual situação da Grécia.

Evan Skolnick (2014, p.18) afirma que o ato de contar uma história normalmente segue um modelo, o conhecido como “Estrutura de Três Atos”. Este modelo está presente em inúmeras obras, desde teatro até cinema. O autor afirma que para alguns, imaginar que várias obras utilizam do mesmo molde pode até ser entristecedor, porém cada história difere em muitos aspectos e torna-se uma obra única.

> The core concept of the Three-Act Structure was first (and best) expressed way back in 335 B.C. by the Greek philosopher Aristotle in his Poetics: “A whole is what has a beginning and middle and end.” Evan Skolnick (2014, p19).

Na citação acima, o autor afirma que o conceito de Três Atos foi originado na Grécia, por Aristóteles. O filósofo afirmava que uma história necessariamente precisaria ter um começo, meio e fim.

Figura 11 – Modelo dos Três Atos.

Fonte: O autor (2021).

Evan Skolnick (2014, p.19) conceitua os três modelos como:

1) Começo: É a introdução do universo onde a história se passará, assim como a apresentação do personagem e de seus principais conflitos. Em algumas situações, é necessário inicializar o personagem em um momento base, e depois inseri-lo em um novo “arco” de mudanças. Um elemento importante nesta área, segundo o autor, é a existência de um evento que sirva de estopim para que o personagem principal entre em um conflito.
2) Meio: É o ponto representado pelo decorrer da história. Inserir problemas e obstáculos pelo qual o protagonista terá que enfrentar para atingir determinado objetivo. O meio é respectivamente o ato que possui maior duração, por isso o autor afirma que é necessário do criador manter o foco, evitando assim perder-se na linha do tempo do jogo.
3) Fim: É o ponto final – temporariamente ou definitivo – de uma história que foi desenvolvida, repleta de ação, drama e outros elementos artísticos que podem ser aplicados. O encerramento pode ser feliz ou triste, onde a expectativa do público pode ser atingida, ou superada. Ao definir um final positivo espera-se que o personagem atinja um destino concretamente definido, e ao estabelecer

um final triste, o autor precisa criar uma relação de quebra de expectativa, e posteriormente trabalhar em sentimentos como a tristeza.

2.4.3 Gêneros de jogos

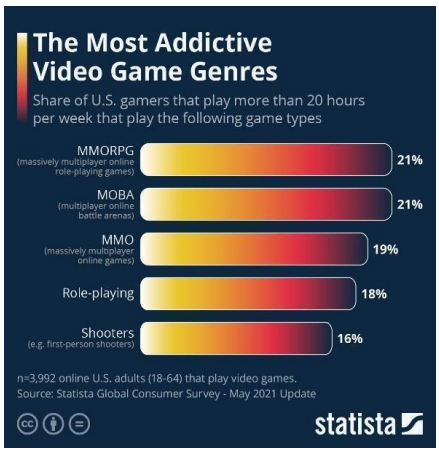
Roger Scott (2010, p.9) afirma que os jogos passaram por várias mudanças ao decorrer dos anos, uma delas foi o surgimento de gêneros e subgêneros que são os responsáveis por mostrar qual será o formato do jogo.

Evan Skolnick (2014, p.148) apresenta suas experiências como um profissional voltado para a área de desenvolvimento de jogos. Ele afirma que muitas pessoas se questionam sobre como ocorre a idealização de um jogo, se há um roteiro ou algo do tipo. O autor afirma que na maioria das vezes não há um roteiro, e sim uma decisão sobre o gênero acompanhado dos elementos característicos da obra.

> A third-person co-op exploration and puzzle game that emphasizes emotion and simple beauty, and contains no dialogue (Journey). Evan Skolnick (2014, p.148)

A citação acima representa uma definição inicial de projeto, onde os fatores definidos são: 1) O jogo é de exploração, 2) O jogo conta com um sistema multijogadores, 3) O jogo preza por passar sensações ao usuário, 4) O jogo não possui diálogos. Portanto, é possível definir o rumo que seu projeto irá seguir antes mesmo de redigir um manuscrito.

Figura 12 – Popularidade de gêneros nos jogos.



Fonte: Statista (2021).

A definição da história em diversos casos é uma parte que é aprofundada nas etapas posteriores do projeto, onde as priorizadas inicialmente são: público-alvo, orçamento, prazos, design e ferramentas de desenvolvimento, afirma Evan Skolnick (2014, p.148). Os jogos precisam estar moldados num escopo geral antes mesmo da elaboração dos personagens, diálogos, inimigos, puzzles e outros elementos.

Segundo Roger Scott (2010, p.9) os jogos passaram a ter inúmeras subtramas e gêneros ao decorrer do tempo, entre elas estão: Ação, Tiroteio, Aventura, Construção, Simuladores de Vida Real, Enigmas, Esportes, Estratégia e Condução de veículos.

A figura (7) refere-se a uma pesquisa realizada nos Estados Unidos, onde 3,992 adultos de 18-64 foram questionados sobre quais os gêneros de jogos eles preferiam. As pessoas em questão, apresentaram dados de utilização de jogos por mais ou menos 20 horas semanais.

> This list of genres and subgenres attempts to scratch the surface. Adult games, serious games, advert games, and vehicular combat are other classifications that fit within several of the genres above. As games combine several genres and subgenres, new ones are constantly being created. For example, the Grand Theft Auto series now combines action - adventure, third person shooter, driving, life simulation, and action - arcade genres into one. Roger Scott (2010, p.11)

A citação acima apresenta o posicionamento do autor a respeito do vínculo entre gêneros. Scott cita o exemplo do jogo Grand Theft Auto, que foi desenvolvido pela Rockstar, cujo gênero é uma mistura entre direção, simulação de vida real, ação e aventura.

> It ’ s one thing to have a good idea; it ’ s another thing to have a marketable one. During the course of my career, I have been told many times (usually by my colleagues in the marketing department) that my idea is a “ designer ’ s idea ” , which means that they think my idea one that I would love to play but it isn ’ t marketable to the general gaming public. Personally, I am torn with this assessment. On the one hand, I can understand their desire to make a game that will sell. If your game sells, that means you can make more games. Evan Skolnick (Scott Rogers, p.26)

Na citação acima o autor afirma que durante sua carreira, ele teve diversas ideias sobre jogos, e seus colegas classificaram isso como “ideias de designer”. Era necessário se obter uma visão voltada aos elementos de mercado, como o marketing.

O autor conceitua que divide opiniões sobre o conceito de desenvolver um jogo com uma visão comercial, e desenvolver um jogo pelo gosto individual.

2.4.4 Equipes de desenvolvimento

As equipes de desenvolvimento de jogos começaram a expandir conforme o aumento na complexidade dos projetos. Inicialmente era possível que apenas uma pessoa realizasse as etapas de planejamento, produção e testes. Porém, com o crescimento do mercado dos videogames, foi necessário ajustar o tempo de desenvolvimento com produtividade, o que pode ser realizado com maior maestria em equipe, afirma Scott Rogers (2010, p.12).

Evan Skolnick (2014, p.130) afirma que a composição da equipe posta a desenvolver o jogo importa tanto quanto a premissa. A qualidade final é diretamente afetada pelas decisões tomadas pela parte de liderança do projeto. Portanto, é de suma importância que todas as partes estejam em sintonia durante este ciclo de produção.

Scott Rogers (2010, p.12) apresenta os cargos normalmente existentes no ramo de desenvolvimento de jogos:

1) Programador: É a pessoa responsável por converter as ideias em código, através de uma linguagem de programação. O programador é responsável por configurar toda a física do jogo, sistema de interação do personagem, câmera e o mundo de ambientação.
2) Artista: É quem cria as artes para a utilização no jogo. Varia de criação de personagem, ambientes ou inimigos. O artista como o programador possui diversas áreas que podem ser abordadas: criação de artes 2D/3D, modelagem, criação de partículas e eventos especiais e animação.
3) Designer: Esta área possui o objetivo de definir o funcionamento do jogo, interação do personagem com sistemas de itens, saúde e ambiente. O designer é responsável por modelar um ambiente, com seus respectivos elementos.
4) Produtor: É a pessoa que define as equipes de trabalho, controla os prazos de entrega e desenvolvimento, controla a parte contábil do projeto e age na junção entre as equipes. Além de gerenciar a produção, na maioria dos

casos é este cargo o responsável por pela introdução e atualização do público referente ao projeto.

5) Testadores: É o nível que deve relatar detalhes sobre o jogo. O testador avalia diversos aspectos de funcionamento do jogo, desde os aspectos técnicos até os fatores relacionados à história. Entretanto, não se trata de uma tarefa simples, é necessário trabalho repetitivo e analítico, em busca de problemas, popularmente chamados de “bugs”.
6) Compositor: Os jogos foram de simples sons mecanizados à músicas masterizadas e mixadas, trilhas sonoras marcantes que por simples gatilhos são lembradas. Este é o trabalho do compositor, escrever uma trilha sonora que combine com o jogo em questão. O trabalho de elaboração de músicas para jogos é uma área diferente de outros trabalhos sonoros, por causa de elementos muito específicos, como o tempo e alternância entre sons.
7) Designer de som: É o responsável por criar os efeitos sonoros do jogo, diferente do compositor que elabora a música. Os sons de um jogo definem sua personalidade. Em jogos mais modernos, a utilização do som é uma ferramenta tanto de noção, que é possível localizar inimigos, tesouros ou ouvir sons da natureza. Um designer deve seguir sugestões da equipe sobre determinados sons, e controlar seus impactos entre as mais diversas situações: positividade, tristeza, surpresa ou suspense.
8) Escritor: O escritor no projeto de games é inserido em situações específicas do projeto, normalmente não tem uma participação em todo o projeto. Suas tarefas variam da criação da interação entre os personagens, como diálogo e manuscritos, assim como guias e tutoriais para os usuários.

Evan Skolnick (2014, p.131) elucida a falta que um profissional especializado faz em um projeto. O autor apresenta situações em que os jogos são jogados aos críticos de forma crua e que provavelmente serão detonados por uma tonelada de feedbacks negativos. Os motivos que levam ao baixo retorno são vários, mas geralmente é o apego a elementos presentes em outras obras, mantendo uma história muito superficial, e outra situação é a falta de um profissional que possua uma boa base sobre determinada área.

> So how did we get to a point where the writer role for video game development is so often overlooked? To understand, we need to go back into the early history of digital game development, when dev teams were very small. Back then, it was not unheard of for a single person to handle the entire job: story, design, art, animation, audio, programming, QA—soup to nuts. Over time, however, advances in hardware specs made increasingly ambitious titles feasible. Projects increased in scope, teams got bigger, and specializations slowly began to emerge. Evan Skolnick (2014, p.132)

Na citação acima, o autor afirma que a falta de profissionais especializados inseridos em um projeto vem do passado histórico, onde poucos profissionais realizavam diversas tarefas. O aumento do porte dos projetos exigiu que as áreas fossem desmembradas de forma que um profissional possa se dedicar apenas à complexidade de suas próprias questões de trabalho.

Scott Rogers (2010, p.12) comenta as mudanças fundamentais no desenvolvimento dos jogos e as funções no desenvolvimento. Anos atrás era possível que um especialista trabalhasse em tudo, mas por critérios de otimização, ocorreram as seguintes mudanças:

> Finally, when game content became too involved to do alone, a dedicated design position was created. While team members on current teams can still wear many hats, specialization is becoming increasingly necessary as games become bigger, more complex, and take longer to make. Scott Rogers (2010, p.12)

A área de produção de jogos também abrange a área dos editores, segundo Scott Rogers (2010, p.20). O autor conceitua as tarefas dos principais encarregados desta área:

1) Gerente de Produto: Área semelhante à do produtor, ambos são responsáveis por gerenciar o trabalho juntamente com a equipe, respeitando um cronograma combinado. Avaliam as questões judiciais relacionadas ao projeto, e realizam os acertos com outras áreas/empresas envolvidas.
2) Gerente de Criação: Esta área pode ser muito volátil, onde o profissional pode atuar em diversas áreas relacionadas a criação. Esse gerente possui o objetivo de trazer uma visão de fora a respeito do desenvolvimento do jogo. O autor afirma que teve experiências em testes de jogos, com o objetivo de analisar se o jogo se mantém agradável ao público.

3) Diretor de Arte: O diretor de arte possui certa relação com a função de Gerente de Criação, suas responsabilidades baseiam-se na identidade visual do jogo, e o rumo artístico que a equipe deve seguir. Este funcionário também é normalmente responsável por definir a arte física que acompanha o jogo, e divulgação.
4) Diretor Técnico: O diretor técnico normalmente é especialista em conceitos técnicos de programação. É o profissional que orienta as equipes na utilização de ferramentas específicas que objetivem a produção e eficiência.

Scott Rogers (2010, p.22) afirma que além dos cargos citados anteriormente, existem pessoas envolvidas em áreas essenciais para a comercialização e publicação de um jogo. É necessário ter a relação com um estúdio, advogados responsáveis por garantir futuras ações legais, profissionais voltados ao marketing, especialistas em relações públicas e diversos outros cargos.

2.4.5 Desenvolvimento de Personagens

Evan Skolnick (2014, p.167) define que a construção de um personagem é extremamente importante, e abrange aspectos fundamentais para a narrativa. Entre as diversas perspectivas sobre a elaboração de um personagem estão: aparência, contexto narrativo, animações e efeitos áudio visuais até a implementação dentro do jogo.

O autor afirma também que a qual plano o personagem será aplicado é outro fator determinante durante sua construção. Caso seja o personagem principal, é necessário desenvolvê-lo ao decorrer da história, criar uma teia de relações e contextualizações. Porém, se for um personagem não-jogável, ou popularmente conhecido como NPC, o processo de construção é totalmente diferente, pois um NPC aparecerá somente em situações específicas e por pouco tempo, já o personagem principal carrega a história do começo ao fim.

> As a potential first link in this chain, the designer must very clearly define why this character exists and, at a core level, what Design needs from it. For example, Character A needs to be able to detect, move toward, and attack the player with either melee or ranged attacks; Character B has no combat requirements but needs to be able to talk to the player to provide a quest. Evan Skolnick (2014, p.168)

Na citação acima, o autor afirma que é primordial que seja definida a origem do personagem, o motivo pelo qual ele será guiado ao decorrer da história, suas motivações e habilidades. Por exemplo, um personagem “A” terá habilidades de batalhar com armamento, enquanto o personagem “B” só poderá enfrentar inimigos lutando corpo-a-corpo.

Figura 13 – Modelagem 3D.

Fonte: Blender (2019).

Evan Skolnick (2014, p.172) afirma que para dar vida a um personagem, criar um documento com suas características contribui de diversas formas. Este documento é baseado em diversas perguntas sobre o posicionamento do personagem na narrativa, e sobre sua individualidade. O autor afirma que os autores de ficção tendem a criar descrições extremamente detalhadas, o que facilita o desenvolvimento do personagem.

Steve Rabin (2011, p.145) afirma que os NPC’s são fundamentais na experiência do jogo. Os personagens que são programados para agirem por si mesmo, de acordo com gatilhos situacionais, o personagem não jogável (NPC), se torna um objeto realista que age por si mesmo em um determinado ambiente.

> Cada vez que o jogador retorna, a urgência do pedido pode aumentar, ou o NPC pode começar a apresentar irritação ou acusar a personagem de ter uma queda por ele. Alterar as conversas e mantê-las adequadas não é fácil, no entanto, pode envolver inteligência artificial de alto nível [Krawczyk06]. Steve Rabin (2011, p.145)

Na citação acima, o autor apresenta uma técnica para aumentar a imersão do jogador com a história. Ao estabelecer uma localidade, e personagens ocupantes dela, em um momento ou outro o jogador irá se encontrar com algum NPC, e assim

iniciar um diálogo. A técnica citada por Steve Rabin consiste na alternância de diálogos com base em quantas vezes o jogador visita a vila.

Obras que abordam a vida real tendem a possuir mais complexidade de personagens por abordarem interações sociais. Os jogos, por outro lado, apresentam ações que não representam a profundidade de questões existentes na vida real. Porém, isso não ocorre em 100% dos jogos, existem diversos casos em que existe um ótimo balanço entre as relações e conflitos mentais envolventes.

> There is something magical about the character that a player controls in a game, so magical that we give that character a special name: the avatar. The word is derived from a Sanskrit word that refers to a god magically taking physical form on earth. And the name is well chosen for a game character, since a similarly magical transformation takes place when players use their avatar to enter the world of the game. Jesse Schell (2015, p.394)

Na citação acima, o autor apresenta a origem da palavra “Avatar” que na sua derivação do sânscrito possui um significado místico. Um Deus que assume uma forma física para entrar em contato com a terra, metaforicamente trata-se da mesma lógica, ao dizer que uma pessoa adentra em outro mundo através de um objeto ou personagem.

Evan Skolnick (2014, p.173) afirma que o documento que agrupa as características de um personagem pode ser classificado em diversas categorias, entre elas estão: Básicos (Nome, Idade, Sexo, Raça, Grau de Inteligência), Pessoais: (Mudanças estruturadas, gostos, crenças, falhas) e Visuais: (Aparência física, roupas e acessórios complementares).

Jesse Schell (2015, p.345) explica que a complexidade da história dos jogos cria personagem excepcionais, com características memoráveis e inerentes. A autora elucida a diferença dos personagens de jogos para outras obras. No aspecto físico, os personagens diferem-se por não ter o ponto psíquico acentuado, pois a maioria das decisões que movem o personagem são escolhidas pelo jogador, portanto as interações do personagem são mais físicas, diferente de outras obras. Ao abordar o aspecto da realidade, pode-se notar que outras obras como livros e novelas mantém seus pés no chão, com descrições da realidade. Já os jogos, por outro lado, constituem em grande parte o apego à ficção.

2.5 LINGUAGENS E FERRAMENTAS UTILIZADAS

2.5.1 C#

Jamie Chan (2015, p.9) apresenta o C# (C SHARP), como uma linguagem de programação desenvolvida pela Microsoft, sob a liderança de Anders Hejlsberg. Trata-se de uma linguagem composta com a estrutura .NET e contempla os ambientes web, móvel, programas e console. Sua escrita assemelha-se às outras linguagens que utilizam o padrão inglês para digitação de códigos.

Terry Norton (2013, p.10) afirma que o C# possui uma vasta documentação na internet. No ambiente de desenvolvimento de jogos pode ser interessante utilizar esta linguagem pois o Motor de Jogos reconhece erros de sintaxe e auxilia na escrita do código.

Quadro 01 - Código em C#.

```
using System;
using System.Text;
using System.Threading.Tasks;
using System.Collections.Generic;
using System.Linq;

namespace PrimeiroPrograma
{
    class Programa
    {
    static void Main(string[] args)
    {
        Console.WriteLine("Olá, mundo!");
        Console.ReadLine();
    }
    }
}
```

Fonte: O autor (2021).

Por se tratar de uma linguagem de alto nível, é necessária a conversão para linguagem de máquina. Esta mudança de estado é chamada de compilação, geralmente é realizada por uma IDE que transforma uma linguagem em código de máquina, afirma Jamie Chan (2015, p.9).

2.5.2 Unity

Joseph Hocking (2015, p.30) afirma que o Unity é um Motor de Jogo que pode ser usado profissionalmente em projetos 2D e 3D. Trata-se de uma ferramenta com muitos componentes e módulos responsáveis por auxiliar no desenvolvimento. A versão básica do Unity é disponibilizada gratuitamente em seu site, enquanto a versão PRO se encaixa em casos de utilização comercial ou de renda acima de um valor delimitado nos termos. Nessa situação, existe a necessidade de adquirir uma licença profissional.

> The market for multi-platform games—especially casual games for IPhone—is extremely popular at the moment, and Unity's commitment to cross platform delivery is well proven. Originally a Mac-based authoring application that could publish to Mac and Windows, Unity unveiled its Windows version in the spring of 2009. As expected, it has opened up opportunities for PC based developers and artists. Since that time, Unity has continued to add support for iPhone, Android, iPad, and Wii and is developing support for Xbox 360 and PS3. The free Unity Web Player has surpassed 35 million installations. Sue Blackman (2015, p.5)

Na citação acima, o autor afirma que a popularidade de jogos casuais para dispositivos móveis está em alta, e o Unity oferece um serviço de qualidade para esta e outras plataformas. O Unity garante suporte para diversos dispositivos como Xbox, Playstation, Dispositivos Móveis e Computadores.

Sue Blackman (2015, p.5) afirma que o Unity acompanha uma documentação detalhada e que abrange as mais diversas questões que possam surgir durante a utilização do software. Entretanto, para iniciantes pode ocorrer a dificuldade de entendimento dos termos técnicos. Portanto, outra ferramenta de extrema ajuda é a busca em sites e fóruns sobre dicas e tutoriais.

Joseph Hocking (2015, p.34) afirma que nem tudo são rosas. A interface do Unity (Figura 9), é uma mistura entre elementos visuais e elementos voltados para a escrita de códigos. Segundo o autor, em casos de projetos mais complexos, podem ocorrer problemas relacionados ao vínculo entre os elementos e scripts. Apesar do Unity fornecer ferramentas de pesquisa, boa parte do trabalho torna-se manual, requerendo que o desenvolvedor análise elemento por elemento.

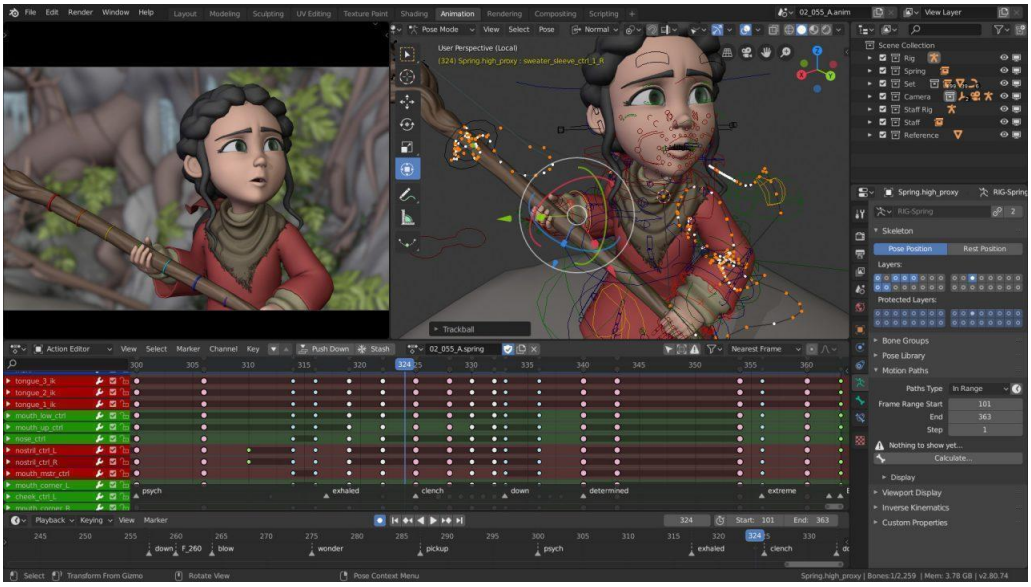
Figura 14 – Interface Unity.

Fonte: Unity (2019).

2.5.3 Blender

Jason Van Gumster (2015, p.11) afirma que o Blender surgiu originalmente como um projeto interno de uma empresa de animação holandesa, a NeoGeo. Por volta da década de 1990, a empresa liderada por Ton Roosendaal liberou versões para o download em seu site, onde era possível adquirir uma versão avançada com um custo, ou uma versão básica gratuitamente.

Figura 15 – Interface Blender.



Fonte: Blender (2019).

Com o passar dos anos, o Blender adquiriu uma grande comunidade que alavancou o lançamento da Fundação Blender, onde a ideia de estabelecer o projeto open-source surgiu. A compra dos direitos do Blender para a nova fundação ocorreu em um tempo surpreendente de 7 semanas, onde a estimativa mínima era de seis meses.

Jason Van Gumster (2015, p.10) apresenta a ferramenta Blender como um software Open-Source, voltado para a área de modelagem e animação artística. Existe a possibilidade de criar os mais diversos tipos de artes neste programa, desde simples imagens estáticas à complexas animações 3D.

> Blender has many features which allow you to create and animate models and characters in Scenes combined with stunning visual effects. Scenes may be rendered into photo realistic images and animations into video clips. Video may be edited and compiled into movies. Interactive Video Games may also be created. John M. Blain (2018, p.)

Na citação acima, o autor evidencia que o Blender oferece elementos para modelagem e animação de elementos realistas. As artes podem ser destinadas às mais diversas áreas como mídias sociais (Facebook, Instagram, Youtube), televisão e videogames.

3. METODOLOGIA

3.1 TIPO DE PESQUISA

Trata-se de uma pesquisa bibliográfica e exploratória, onde busca-se evidenciar como uma engine auxilia no desenvolvimento de jogos eletrônicos, e boas práticas empregadas em um projeto. Com base na pesquisa realizada no referencial teórico, propõe-se o desenvolvimento de um projeto baseado nos temas discutidos, utilizando a engine Unity e a linguagem C#.

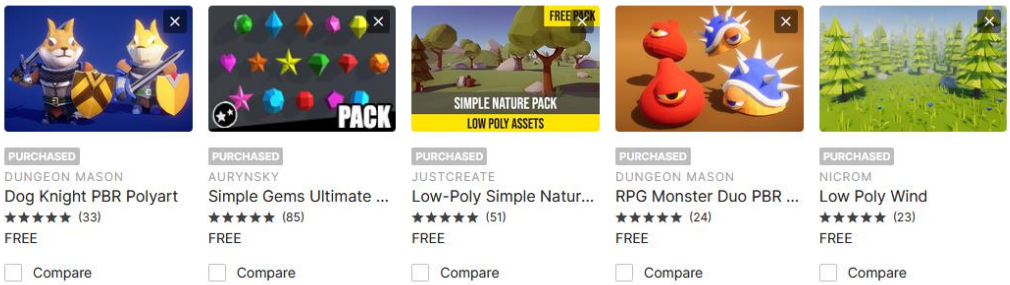
4 DESENVOLVIMENTO

4.1 PACOTES UTILIZADOS

Os conteúdos utilizados em um jogo podem ser criados de forma autoral, ou utilizando pacotes e componentes de terceiros. A Asset Store é a loja oficial do Unity, nela é possível obter conteúdo para os mais diversos projetos e ambientes. Todos os

materiais e personagens utilizados neste projeto foram obtidos de forma gratuita, de acordo com a imagem abaixo.

Figura 16 – Assets utilizados no projeto.

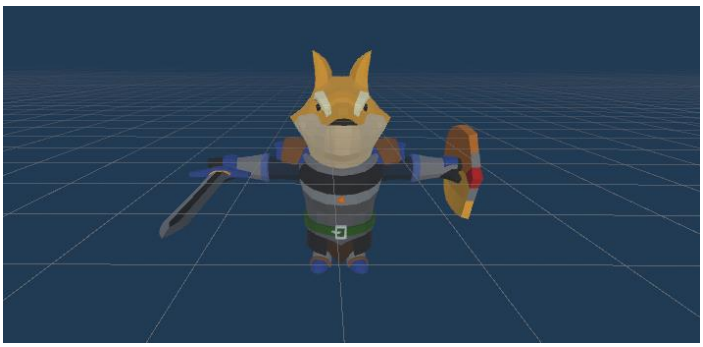


Fonte: O autor (2021).

Cada objeto inserido em uma cena responde de forma independente, e carrega uma série de componentes responsáveis por criar efeitos visuais ou relações físicas do objeto com o mapa. O *mesh render* é um componente responsável por renderizar um objeto e torná-lo visível em uma cena, e o *mesh colider* é quem cria uma área de colisão entre o objeto e outros itens da cena. Os dois exemplos citados anteriormente estão presentes na maioria dos objetos utilizados durante o desenvolvimento de um jogo, salvo exceções onde é necessário criar objetos que funcionam como gatilhos.

4.1.1 Personagem Principal

Figura 17 – Aparência do personagem.



Fonte: O autor (2021)

O personagem principal é originado do pacote *Dog Knight PBR***,** onde é possível escolher entre duas opções do mesmo boneco, porém com texturas

diferentes. Além dos pacotes de textura e da estrutura física do personagem, é possível utilizar as seguintes animações: *Attack01*, *Attack02, Defend, Die, Die Recover, Dizzy, Get Hit, Idle, Run e Walk.*

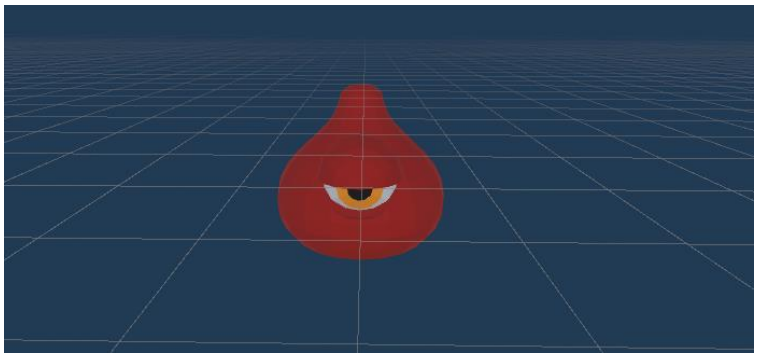
Cada animação realiza uma ação diferente, e movimenta partes específicas do corpo do personagem. É possível realizar a junção entre animações, como realizar a animação de movimento e ataque ao mesmo tempo, este tema será aprofundado mais para frente do projeto, na parte de máscara de animações.

4.1.2 Inimigos

Além do personagem controlado pelo jogador principal, monstros foram adicionados ao projeto, tais inimigos estão disponíveis no pacote **RPG Monster Duo**. Realizando funções pelo mapa como patrulhar, seguir e atacar outros personagens, os monstros são capazes seguir um inimigo e matá-lo, em uma sequência de ataques.

O pacote utilizado neste projeto apresenta dois modelos de monstros, cada um com animações individuais e exoesqueleto diferente. É possível criar modelos personalizados para seus projetos, criar e modificar diversos tipos de NPC'S, a partir de um mesmo modelo.

Figura 18 – Aparência do inimigo



Fonte: O autor (2021)

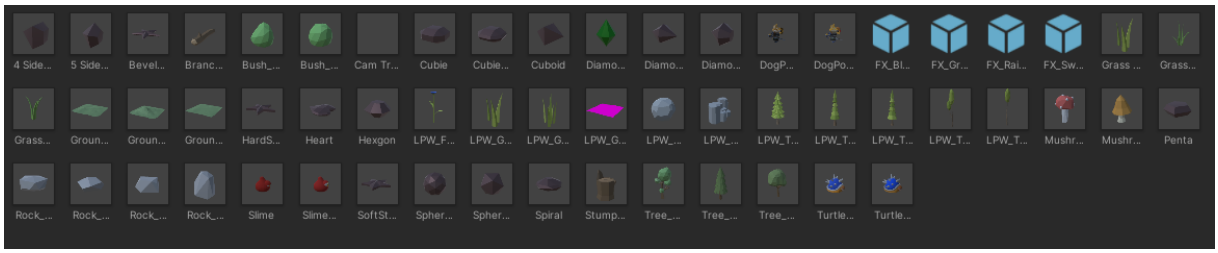
Os inimigos deste projeto funcionam de acordo com uma simples inteligência artificial randômica, onde as ações ocorrem após um sorteio baseado em uma porcentagem, ou em gatilhos de acerto ou de visão.

4.1.3 Objetos e cenários

A composição física do ambiente de um jogo é caracterizada por diversos elementos independentes, posicionados de forma objetiva, que no fim desempenham o papel de representar um bioma, uma cidade ou um local específico.

Os componentes utilizados neste projeto foram *Low Poly Simple Nature* e *Low Poly Wind***,** onde é possível obter materiais referentes a vegetações, rochas e outros tipos de itens fundamentais para constituir um cenário dinâmico. Vale ressaltar que todos os modelos de componentes que estão contidos nesses pacotes citados anteriormente, apresentam animações e texturas únicas que são feitas sob medida para as características de seu componente pai.

Figura 19 – Componentes do cenário.



Fonte: O autor (2021)

É possível analisar na figura (24), que existem 4 componentes na primeira linha que estão em forma de cubo. Estes itens referem-se a partículas utilizadas para acrescentar efeitos visuais às ações, como partículas de gotas de água para criar a animação de chuva e partículas de grama, para simular o corte.

4.2 MECÂNICAS

Durante o desenvolvimento de um jogo, cabe a equipe criar as mecânicas responsáveis por dar vida às ações de um personagem ou ambiente. Desde o básico como o movimento do personagem, até questões mais complexas, oriundas de interações do personagem com uma história ou ambiente.

É praticamente impossível comparar as mecânicas de jogos de gêneros diferentes, pois todo o foco do desenvolvimento se atém a fatores específicos voltados para o público-alvo. Um jogo de FPS não apresenta as mesmas características de um

jogo MOBA, e muito menos um jogo de corrida apresenta as mesmas especificidades de um jogo de basquete.

4.2.1 Movimentação

Com o Unity é possível desenvolver jogos para as mais diversas plataformas, e pensando nisto que vale ressaltar a variedade de modos para a entrada de dados disponível. Seja com teclado, touchscreen ou controle de videogame, é possível configurar os comandos certos para qualquer projeto.

Este projeto utiliza um sistema de teclas muito comum na maioria dos jogos, onde para se movimentar os comandos são: W (Frente), A (Esquerda), S (Trás) e D (Direita). Além da movimentação, o personagem também é capaz de realizar a ação de ataque, que é utilizada no botão esquerdo do mouse (M1).

Quadro 2 – Função de movimento.

```
public float velocidadeMovimento = 3f;

void MoverPersonagem()
    {
        direcao = new Vector3(horizontal, 0f, vertical).normalized;
        if (direcao.magnitude > 0.1f)
        {
            float anguloMov = Mathf.Atan2(direcao.x, direcao.z) *
Mathf.Rad2Deg;
            transform.rotation = Quaternion.Euler(0, anguloMov, 0);
            estadoMov = true;
        }
        else
        {
            estadoMov = false;
        }
        controller.Move(direcao * velocidadeMovimento * Time.deltaTime);
    }
```

Fonte: O autor (2021).

O movimento do personagem trata-se de transformar a interação do usuário nas teclas de comando em um chamado para alterar o eixo X e Y e Z. Além da direção, é necessário definir uma velocidade em que o objeto específico irá se movimentar, esta é a função da variável *velocidadeMovimento*, na tabela abaixo.

O Unity como uma ferramenta voltada para o desenvolvimento de jogos, auxilia muito com funções que poupam tempo e agregam funcionalidades, como por exemplo

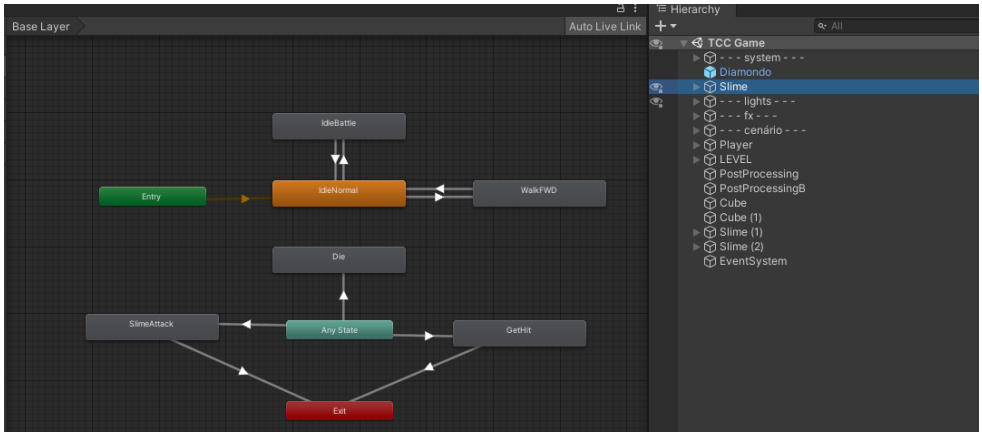
a função *Move(),* presente na tabela acima, que realiza a configuração necessária para mover o personagem desde que receba os parâmetros corretos.

Além de configurar o movimento do personagem, é necessário observar que a rotação é um fator que deve responder às interações do objeto. Portanto, de maneira simplória, um cálculo de tangente é necessário para calcular um ângulo em radianos, e posteriormente sendo convertido para graus. Este é o papel da função *Atan2(),* presente na tabela acima**,** realizar a interação entre grau e ângulo para determinar o valor de rotação do personagem.

4.2.2 Animações

As animações de um jogo representam um grande papel na parte visual do projeto, tornam as interações dinâmicas e agradáveis de serem vistas pelo usuário. Portanto, no desenvolvimento de um jogo, é necessário atentar-se na programação das animações e configuração dos personagens.

Figura 20 – Composição do Animator.



Fonte: O autor (2021).

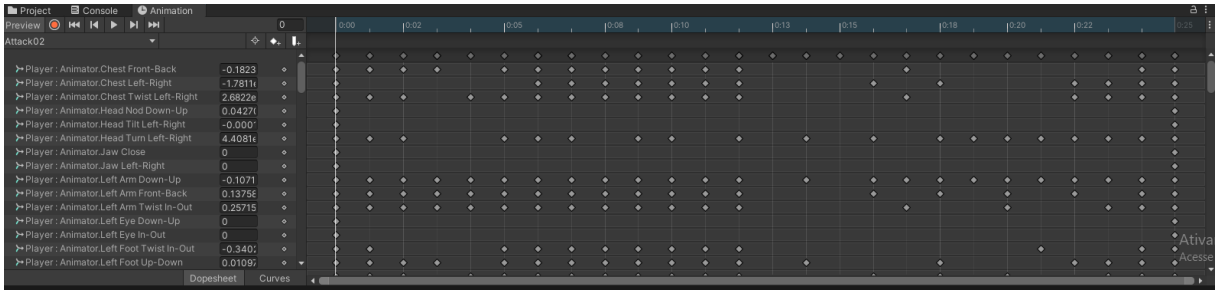
O Unity oferece um ambiente intuitivo para trabalhar com animações e efetuar a relação entre gatilhos e estados. O formato de fluxograma, garante fácil entendimento até mesmo para desenvolvedores que não tenham tanta experiência com o ramo de desenvolvimento de jogos eletrônicos.

Os retângulos que estão em cores diferentes referem-se aos estados básicos de um objeto e definem em quais momentos as animações poderão ser iniciadas. Por exemplo, uma animação que esteja ligada ao *AnyState***,** pode acontecer em qualquer

momento independente de qual seja seu estado definido, posteriormente veremos que um personagem entrará em diversos estados dependendo de suas interações.

Como citado anteriormente, os pacotes de objetos utilizados neste projeto são inteiramente gratuitos e disponíveis na Asset Store. Estes pacotes em questão, carregam algumas animações que foram utilizadas durante o projeto, como o movimento de ataque, de dano, de morte e de movimento.

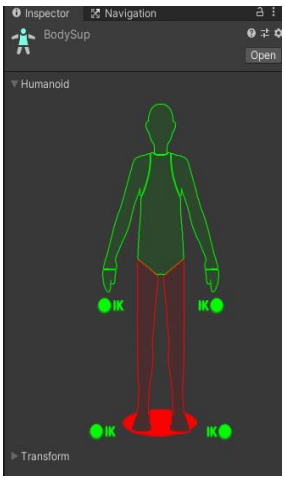
Figura 21 – Linha do tempo das animações.



Fonte: O autor (2021).

A figura (10), apresenta o esquema de linha do tempo de uma animação, onde cada ponto branco representa uma alteração na estrutura óssea do personagem. Há uma enorme variedade de configurações que se adequam às necessidades do projeto, é possível inserir animações de um personagem em outro, ou delimitar qual membro do corpo será ignorado quando a animação ocorrer.

Figura 22 – Máscara de animações.



Fonte: O autor (2021).

O esquema de linha do tempo é popularmente conhecido em diversos outros programas como editores de imagens e vídeos, o que torna sua utilização popular e garante a maior facilidade na busca por materiais referentes à esta área de desenvolvimento.

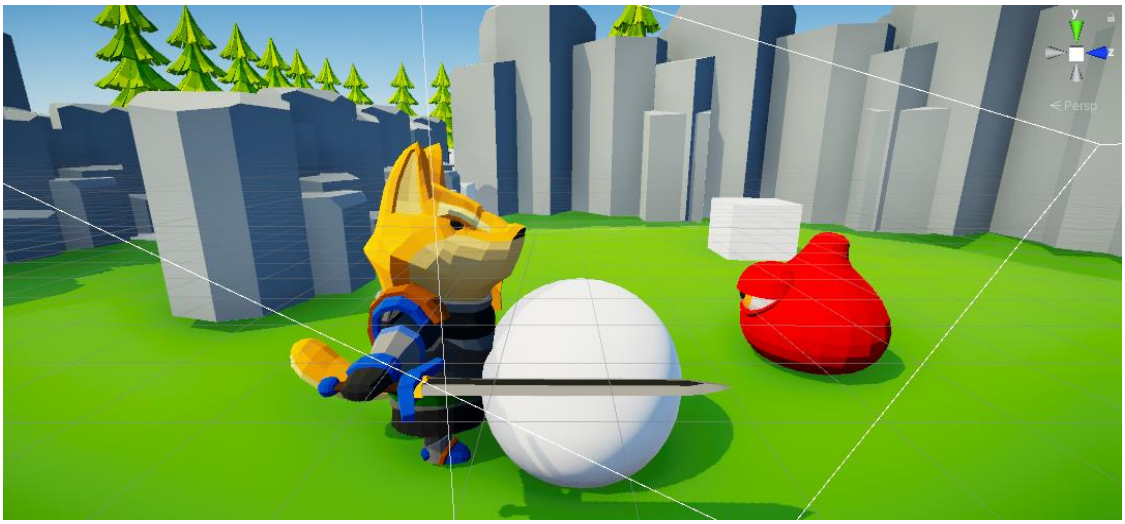
A figura (18) apresenta uma estrutura corpórea, onde é possível marcar e desmarcar membros, durante uma animação. Este componente é chamado de máscara de animação, e possui a utilidade de restringir movimentos apenas a determinadas partes do corpo. No projeto em questão, a animação de ataque precisaria estar sincronizada com a animação de movimento, portanto, o ataque foi restringido apenas para a parte superior do corpo.

4.2.3 Ataque

O ataque é a principal ação em diversos gêneros de jogos. Para criar essa mecânica tão famosa em um jogo, é necessário estar atento para alguns aspectos e artimanhas de programação e desenvolvimento. Este processo é uma junção entre animações, formas geométricas e gatilhos que representam a trajetória do ataque.

Para configurar o ataque do personagem, é necessário definir uma quantia de dano (*amountDmg*), uma área de ataque (*hitRange*) e quais objetos poderão ser atingidos (*hitMask*). Após definir os componentes acima como variáveis, é preciso garantir que as animações estejam definidas para todos os estados (*AnyState*).

Figura 23 – Mecânica da área de ataque.



Fonte: O autor (2021).

Para definir uma área de ataque, foi utilizado uma forma de esfera (figura 10). Utilizar uma forma como gatilho é uma maneira interessante de estabelecer uma relação do personagem com o ambiente, no jogo em questão, se um inimigo estiver no raio de alcance da esfera enquanto o jogador desfere o golpe, será atingido com o dano definido previamente.

Quadro 3 – Função Ataque().

```
void Ataque()
    {
      estadoAtaque = true;
      anim.SetTrigger("Ataque");
      fxAtaque.Emit(1);
      infoAtaque = Physics.OverlapSphere(hitBox.position,
      hitRange,hitMask);

      foreach(Collider c in hitInfo)
      {
         c.gameObject.SendMessage("danoHit", amountDmg,
         SendMessageOptions.DontRequireReceiver);
         }
     }
```

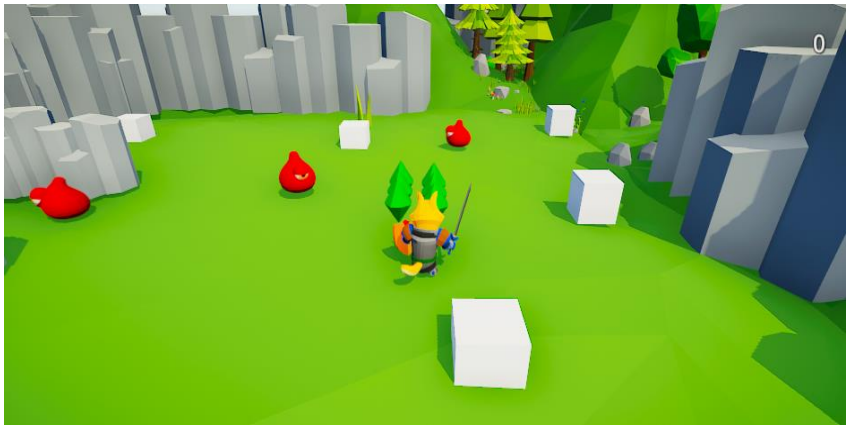
Fonte: O autor (2021).

Vale ressaltar que a mecânica de ataque precisa ser configurada minuciosamente para corrigir futuros bugs. Durante o desenvolvimento do projeto, foi possível notar que uma vez que o personagem realizava um ataque, todos os objetos próximos a ele recebiam o ataque. Uma maneira utilizada para resolver este problema foi criar um grupo específico para os objetos, e assim impedir que objetos como o chão, vegetação e paredes fossem atingidos durante uma animação.

4.2.4 Coleta de itens

Uma das mecânicas mais populares nos jogos atuais é a coleta de itens. Ao derrotar um chefão ou após explorar uma área nova, ocorre a coleta de um objeto raro ou um traje de batalha. Neste projeto foi inserido um sistema baseado nisso, ao derrotar um inimigo é possível coletar um diamante, e esse diamante é acrescentado na tela para a ciência do jogador.

Figura 24 – Itens coletáveis.



Fonte: O autor (2021).

A mecânica da coleta de itens foi aplicada utilizando os inimigos que realizam a patrulha pelo mapa, uma vez que esses inimigos são derrotados, existe uma probabilidade definida previamente de algum diamante nascer no local onde o corpo do slime se encontrava. É possível criar várias recompensas e raridades para o drop de itens de um jogo, basta instanciar diversos tipos de elementos e definir a probabilidade de cada um aparecer após a derrota de um NPC, ou a descoberta de algum lugar secreto do mapa.

Quadro 4 – Coleta de itens.

```
public void valorGem(int amount)
    {
     gemas += amount;
     txtGem.text = gemas.ToString();
    }

 private void OnTriggerExit(Collider other)
    {
        switch (other.gameObject.tag)
        {
            case "Coletar":
                _GameManager.valorGem(1);
                Destroy(other.gameObject);
                break;
        }
    }
```

Fonte: O autor (2021)

Após a coleta de itens, foi definido que o valor dos cristais seria exibido no topo superior direito da tela. Para passar os valores de variáveis para o HUD, foi necessário

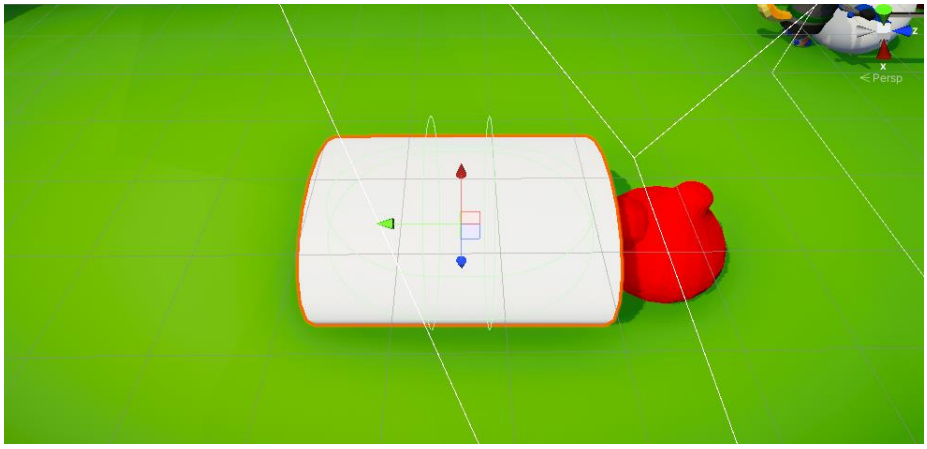
utilizar o complemento *Canvas*, que é responsável pela representação gráfica de elementos no Unity.

As mecânicas de coleta constituem o princípio para a elaboração de sistemas mais completos de negociação, compra e venda. É possível definir inúmeros esquemas baseados na coleta de itens, cabe apenas ao desenvolvedor permitir que a criatividade inunde o projeto.

4.2.5 Visão do inimigo

Para que as mecânicas de perseguição do inimigo funcionassem, foi necessário delimitar uma área para representar o raio de visão inimigo. Utilizando um modelo 3D de cilindro, foi possível posicionar um elemento à frente do personagem que iria passar os parâmetros necessários para identificar alvos que estivessem perto o suficiente.

Figura 25 – Mecânica de visão inimiga.



Fonte: O autor (2021)

Vale ressaltar que o objeto 3D presente na figura acima, é visto apenas pelos desenvolvedores, pois os parâmetros de renderização desativam a representação visual logo após o início do jogo. Muitos elementos são utilizados representativamente pelo desenvolvedor para ter uma noção de como os objetos estarão dentro da cena final do jogo.

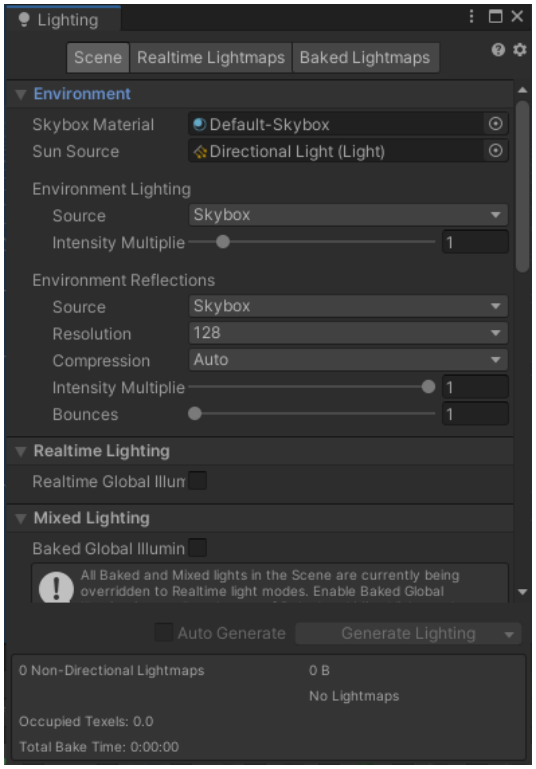
4.3 VISUALIZAÇÃO

Um dos fatores predeterminantes da qualidade de um jogo é a sua configuração de cena, principalmente voltada para as câmeras e iluminação. Esta área é fundamental para criar a imersão do jogador com o ambiente pelo qual ele está inserido, e permitir uma melhor interação do personagem com objetos e superfícies.

4.3.1 Iluminação

Para configurar a iluminação no Unity, é necessário utilizar uma aba específica que permite a alteração de diversos valores referentes a especificidades técnicas de ambiente. É possível alterar componentes como o material de iluminação, e valores referentes a intensidade, direção da luz, origem, iluminação global em tempo real, e outros valores.

Figura 26 – Iluminação de cena.



Fonte: O autor (2021)

Este projeto utilizou um sistema de luz padrão, onde nenhum valor da aba Lightning foi alterado, apenas a opção de gerar luz foi escolhida. A iluminação de uma

cena pode basear-se nas horas do dia, de acordo com a posição da luz. O componente *DirectionalLight***,** é responsável por indicar de qual direção a luz estará aplicada, podendo ser alterada de acordo com os gostos do programador.

4.3.2 CÂMERA DINÂMICA

Uma das características que compõem as mecânicas de um jogo é o estilo de visão. As variações de câmera ocorrem de acordo com o gênero, tipo de gameplay e se trata-se do formato 2D ou 3D. Existem jogos que utilizam várias formas de visão, alternando de acordo com ambientes, personagens e eventos.

Um exemplo de jogo onde é possível alternar entre os tipos de câmeras, é o *Grand Theft Auto*, jogo desenvolvido pela *Rockstar Games*, que apresenta um mundo aberto que pode ser explorado. No jogo em questão, é possível alternar na distância da câmera em relação ao jogador, ou optar pelo estilo de jogo em primeira pessoa, que apresenta um modelo em que o jogador utiliza a visão do próprio personagem.

Figura 27 – Modelo de câmera dinâmica.

Fonte: O autor (2021).

Este projeto baseia-se em dois modelos de câmeras, o primeiro consiste numa câmera fixa que acompanha o jogador numa visão em terceira pessoa. Já o segundo modelo de câmera apresenta-se ocasionalmente quando o jogador chega em determinadas áreas. A necessidade dessa alternância entre câmeras existe uma vez que o personagem pode entrar em cantos do cenário em que objetos se sobreponham a visão do jogador.

Para criar este tipo de funcionalidade foi necessário desenvolver duas câmeras com visões diferentes, uma segue a câmera principal na parte traseira do personagem, e a outra trata-se de uma visão superior. Também foi preciso criar um objeto para delimitar a área em que a câmera superior seria ativa, após feito isso bastou utilizar uma função para quando o personagem entrasse na área delimitada.

Quadro 5 – Câmeras dinâmicas.

```
private void OnTriggerEnter(Collider other) {
    switch (other.gameObject.tag)
    {
        case "camSensor":
        upCam2.SetActive(true);
        break;
    }
  }

 private void OnTriggerExit(Collider other) {
    switch (other.gameObject.tag)
    {
        case "camSensor":
        upCam2.SetActive(false);
        break;
    }
 }
```

Fonte: O autor (2021).

4.4 AMBIENTAÇÃO

4.4.1 Paredes invisíveis

Figura 28 – Paredes invisíveis.

Fonte: O autor (2021).

Ao desenvolver um level-design, é necessário delimitar quais áreas serão acessíveis para o personagem principal e os NPCS. Existem mecânicas que permitem

gerenciar pontos de deslocamento do personagem, uma delas é a elaboração de paredes invisíveis. Trata-se de um elemento plano, com colisão ativa, onde é possível desabilitar sua visibilidade durante a execução do jogo.

Uma mecânica muito popular dos jogos eletrônicos é a dos Easter Eggs (Ovos de Páscoa), onde objetos são escondidos em locais que exigem busca extrema por parte dos jogadores, e escondem algum bônus ou item específico. Com as paredes invisíveis é possível definir trajetórias extras, itens ocultos ou locais de batalha específicos.

4.4.2 Interação com a grama

Existem várias peculiaridades que definem o ambiente de um jogo, é possível criar um cenário estático, onde a interação do jogador com o ambiente não exerce mudanças, ou um cenário dinâmico, onde as mudanças ocorrem de acordo com as interações. Este jogo utiliza um sistema dinâmico de vegetação, onde o ataque direto do jogador carrega o efeito de corte, juntamente configurado com a liberação de partículas.

Para que o personagem interagisse com o ambiente, foi necessário realizar algumas configurações como a presente na tabela abaixo. Quando o usuário utilizasse seu ataque em uma área com a vegetação alta, seria interessante se a vegetação fosse destruída e soltasse algumas partículas para representar o corte.

Para definir qual grama seria destruída e qual não, foi necessário definir uma *layer* específica para a vegetação. Portanto, é possível atacar outros objetos e eles não sofrerão os mesmos efeitos da grama, que pode ser cortada.

Quadro 6 – Interação com a grama.

```
public ParticleSystem fxAtaque;
private bool corteGrama;

   void danoHit(int amount)
   {
       if(corteGrama == false)
       {
           corteGrama = true;
           transform.localScale = new Vector3(1f, 1f, 1f);
           fxAtaque.Emit(10);
       }}
```

Fonte: O autor (2021).

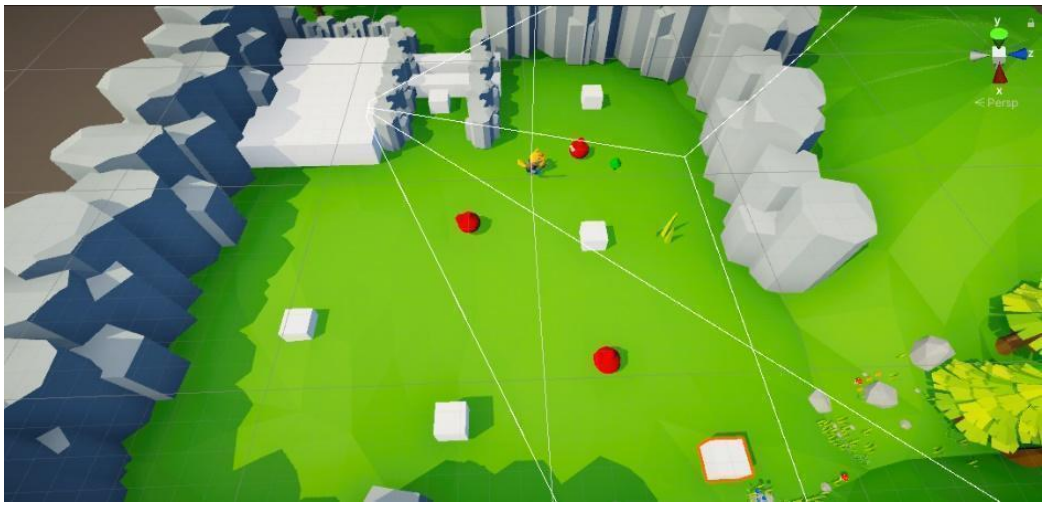
Assim que o personagem chegar na área de colisão da grama e utilizar seu ataque, o componente *Transform* do objeto grama, irá passar um novo valor, correspondente ao tamanho final da grama pós-corte. É necessário observar que exista uma variável *isCut*, seu principal papel é impedir que o player interaja com a grama diversas vezes mesmo após ela já ter sido cortada.

4.4.3 Navegação

Elaborar um cenário é uma das principais etapas do desenvolvimento de um jogo. É preciso realizar um estudo prévio sobre seus componentes e decidir qual o melhor jeito de inseri-los no mesmo ambiente de maneira funcional. O Unity oferece um amplo suporte para o level-design, e é possível encontrar na Asset Store ferramentas de terceiros, que possuem um custo, mas também auxiliam muito durante o desenvolvimento de forma eficaz e produtiva.

A figura abaixo representa os pontos de deslocamento dos inimigos, onde cada cubo representa uma posição. A escolha referente a qual caminho seguir ocorre aleatoriamente através de um sorteio entre todos os cubos (*Waypoints*). Como proposta inicial, o inimigo poderia mudar seu destino no meio da trajetória, porém o deslocamento do personagem até o ponto final criava uma proposta de patrulha que se adequa melhor ao projeto.

Figura 29 – Waypoints de deslocamento.



Fonte: O autor (2021)

Os cubos presentes na figura (29), são apenas representações utilizadas durante o desenvolvimento, ao executar o arquivo em sua forma final, o elemento chamado *mesh render***,** que é responsável por renderizar os elementos do projeto, desabilita visualmente a forma geométrica.

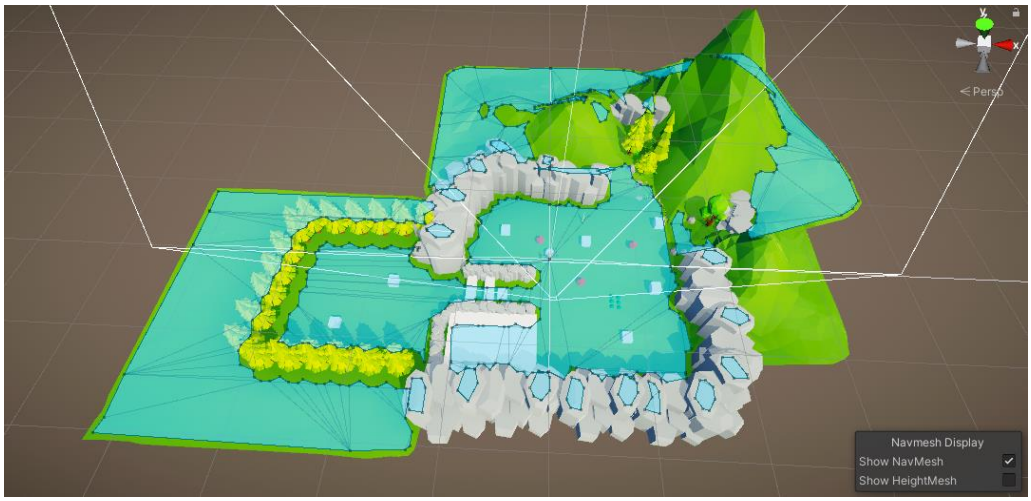
Este projeto utilizou um sistema básico de elaboração de níveis, utilizando as próprias ferramentas disponibilizadas pela Engine. Após inserir os componentes no projeto, redimensioná-los, trabalhar com as físicas e interações com outros personagens, é necessário pensar em um fator importante: a navegação.

Os componentes que definem a navegação em um jogo são essenciais, visto que sem eles, seria muito difícil definir manualmente quais áreas estariam disponíveis para o deslocamento dos personagens. Além de que é essencial para a configuração de outros elementos, que o ambiente esteja delimitado e pronto para a interação do personagem.

Apenas adicionar um objeto a cena não permite que ele realize a interação com outros personagens e objetos. Pode-se utilizar os componentes utilizados neste projeto como parâmetro, ao adicionar uma árvore, era possível atravessá-la, visto que não havia nenhuma configuração de colisão.

Ao inserir objetos na cena como pedras e vegetações, é necessário verificar em qual *layers* eles se encontram, pois durante o “cozimento da malha”, que define qual será a área de deslocamento padrão, pode acontecer problemas onde os personagens consigam subir em elevações, ou até mesmo atravessar.

Figura 30 – Área do NavMesh.



Fonte: O autor (2021)

O *NavMesh* é um componente do Unity que se baseia em inteligência artificial para definir rotas de navegação para os personagens. Observa-se na figura (10), uma área traçada em azul claro, que corresponde aos limites de movimentação das entidades do jogo.

4.4.4 Chuva e noite

Como citado anteriormente, o level-design é uma das etapas fundamentais no desenvolvimento de um jogo. É possível configurar diversos ambientes e cada um com suas características próprias. A possibilidade de incrementar diversos ambientes é uma das técnicas utilizadas para aumentar a imersão do jogador e não permitir que a experiência seja maçante e repetitiva.

Este projeto conta com alguns gatilhos de ambiente, caracterizado por locais que alteram o clima, ou o período do dia. A figura (10) representa dois locais do mapa, à esquerda um ambiente que apresenta a luminosidade referente a um dia de sol, enquanto a imagem da esquerda apresenta um ambiente noturno com um clima chuvoso.

Quadro 7 – Corotina de chuva.

```
IEnumerator RainManager(bool isRain){
     switch(isRain){
        case true:
          for(float r = rainModule.rateOverTime.constant; r <
          taxaChuva; r += rainIncrement)
              {
                 rainModule.rateOverTime = r;
                 yield return new WaitForSeconds(taxaAtrasoChuva);
              }
          rainModule.rateOverTime = taxaChuva;
          break;

        case false:
          for(float r = rainModule.rateOverTime.constant; r > 0; r -=
          rainIncrement)
                {
                    rainModule.rateOverTime = r;
                    yield return new WaitForSeconds(taxaAtrasoChuva);
                }
          rainModule.rateOverTime = 0;
          break;
                }
    }
```

Fonte: O autor (2021)

Apesar de existirem muitas configurações, o sistema de alternância de clima/horário baseia-se num simples gatilho localizado no mapa, que quando o jogador chega em contato com ele, um novo componente é ativo com as respectivas configurações de ambiente. É possível criar inúmeros gatilhos espalhados pelo mapa, e cada um portando um ***preset*** exclusivo.

Figura 31 – Mecânica de dia e noite.

Fonte: O autor (2021)

4.5 INTELIGÊNCIA ARTIFICIAL

Kai Fu-Lee (2010, p7) afirma que a inteligência artificial até pouco tempo atrás era vista como a tecnologia do futuro, responsável por permitir que robôs pudessem ter consciência e agir independentemente. Entretanto, o autor afirma que a visão social foi alterada, visto que a I.A já está inserida em nosso cotidiano em muitas plataformas presentes na Internet.

João Luís Garcia Rosa (2011, p5) afirma que a I.A é capaz de trabalhar com lógica de programação, simulações de redes neurais e cálculos complexos. Utilizando técnicas de outras áreas de estudos, como a psicologia, linguística, filosofia e ciência da computação.

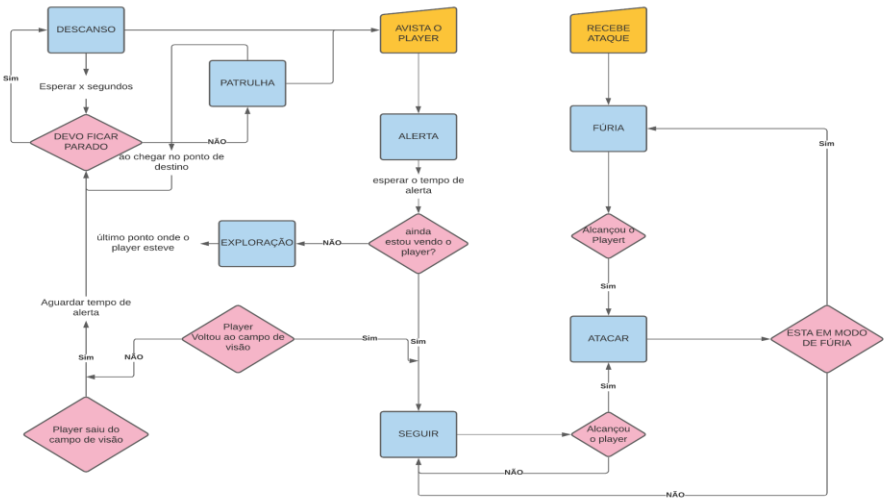
Este projeto em questão, buscou utilizar um sistema de encadeamento de ações, onde ocorre a utilização de laços lógicos para determinar qual a ordem dos acontecimentos. Trata-se de uma analogia simplória da capacidade total de uma inteligência artificial, porém foi possível estabelecer um modelo completo para as ações dos personagens.

4.5.1 Fluxograma de ações

As ações dos inimigos foram configuradas de acordo com testes de probabilidades e gatilhos de posição. Uma vez que o player entre no raio de visão do inimigo, e mantenha-se parado por 3s, o slime irá alterar seu estado e perseguirá o inimigo até que o perco de visão momentaneamente.

Diversos valores podem ser alterados diretamente pelos painéis do Unity, uma vez que uma variável tenha sido criada via código, para delimitar um parâmetro. Portanto, velocidade de movimento, vida dos inimigos, vida do jogador principal e distância de ataque, podem ser alteradas de maneira simples e intuitiva, o que agrega muito na hora de realizar testes e corrigir bugs.

Figura 32 – Fluxograma de comportamento.



Fonte: O autor (2021)

Assim como o estado de perseguição (Follow), os outros estados são baseados em interações com o ambiente e com o personagem principal. O script de máquina de estado é o responsável por efetuar as trocas de operações para que o slime responde diferente a cada estímulo.

Durante a criação dos fluxogramas de estado do personagem, foi concluído que para uma animação completa do personagem, onde é possível encontrar diferentes ações através de probabilidade, seria necessário definir alguns estados do personagem: IDLE, ALERT, FOLLOW, PATROL, FURY e DIE.

4.5.2 Corotinas

As corotinas representam uma área de código que abrange outras funções ou rotinas. Neste projeto as corotinas são utilizadas para definir as características de um estado, definir os intervalos de execução e definir os valores de componentes.

Quadro 08 – Corotinas.

```
IEnumerator ALERT()
 {
     yield return new WaitForSeconds(_GameManager.slimeAlertTime);

     if(visaoPlayer == true)
     {
         ChangeState(enemyState.FOLLOW);
     }
     else
     {
         manterEstado(10);
     }}
```

Fonte: O autor (2021).

Utilizando o código acima como referência, a corotina IDLE (ócio) possui um intervalo chamado *slimeIdleWaitTime* que é responsável pelo tempo de aguardo entre uma ação e outra do slime. Enquanto a corotina PATROL (patrulha) utiliza um componente de estado *agent.remainingDistance*, este valor refere-se a distância do inimigo do player principal.

Portanto, as corotinas auxiliam a gerenciar os estados e inserir condições para que o funcionamento seja fluído e para que boas práticas na elaboração de um código sejam prezadas, com fins de desempenho e estabilidade.

4.5.3 Estados dos inimigos

A mecânica de estados corresponde a situações pré-definidas que são ativadas através de casos situacionais ou gatilhos. Através dos estados é possível gerenciar quais cenas são exibidas durante o jogo, como diálogos, telas de carregamento, cutscenes, menu principal etc.

Quadro 9 – Gerenciamento de estados.

```
public enum estadoInimigo
  {
    IDLE, ALERT, PATROL, FOLLOW, FURY, DIE
  }
public enum estadoJogo
  {
    GAMEPLAY, DIE
  }
```

Fonte: O autor (2021)

Como citado anteriormente, o inimigo irá responder de maneira randômica em uma espécie de inteligência artificial. Para definir quais serão as ações do personagem é necessário imaginar suas possíveis ações, e assim foi definido que o inimigo pode permanecer ocioso (*IDLE*), em estado de alerta (*ALERT*), em estado de patrulha (*PATROL*), seguindo o personagem principal (*FOLLOW*), em estado de fúria (*FURY*) e morto (*DIE*).

Além dos estados do slime inimigo, a tabela acima possui dois estados: *GAMEPLAY* e *DIE*, estes correspondem aos estados do jogador e não dos inimigos, caso o jogador receba um número X de dano, também será possível que ele morra.

4.5.3.1 IDLE

Quadro 10 – Estado IDLE.

```
IEnumerator IDLE()
    {
        yield return new
        WaitForSeconds(_GameManager.slimeIdleWaitTime);
        manterEstado(50);
    }

 switch (newState)
    {
         case enemyState.IDLE:
         agent.distanciaParada = 0;
         destino = transform.position;
         agent.destination = destino;
         StartCoroutine("IDLE");
         break;
    }
```

Fonte: O autor (2021)

Corresponde ao modo de ócio do personagem. Suas ações são restringidas ao local onde ele se encontra por último. A probabilidade de um personagem manter-se

em repouso foi definida como 50%, visível na tabela abaixo no método ManterParado(50).

4.5.3.2 Patrol

Este estado caracteriza-se pelo deslocamento do personagem pelos WayPoints. Definida aleatoriamente, a patrulha permite que os inimigos se desloquem pelo mapa até que ocorra o contato com o jogador principal.

Quadro 11 – Estado PATROL.

```
IEnumerator PATROL()
    {
      yield return new WaitUntil(() => agent.remainingDistance <= 0);
      manterEstado(30);
    }

switch (newState)
    {
      case enemyState.PATROL:
      agent.distanciaParada = 0;
      idPonto = Random.Range(0,_GameManager.slimeIdPonto.Length);
      destino = _GameManager.slimeIdPonto[idPonto].position;
      agent.destination = destino;
      StartCoroutine("PATROL");
      break;
    }
```

Fonte: O autor (2021)

4.5.3.3 Alert

Corresponde ao modo de alerta do personagem. A troca para este estado ocorre quando o personagem principal se mantém no raio de visão por até 3s. Uma vez que o personagem entre em estado de alerta, uma animação é ativada através de gatilho, e o personagem mantém seu olho aberto, como se estivesse procurando o inimigo.

Quadro 12 – Estado ALERT.

```
  IEnumerator ALERT()
     {
       yield return new WaitForSeconds(_GameManager.slimeAlertTime);

       if(playerVisao == true)
          {
            ChangeState(enemyState.FOLLOW);
          }
       else
          {
            manterEstado(10);
```

```
            }
        }

    switch (newState)
        {
            case enemyState.ALERT:
            agent.distanciaParada = 0;
            destino = transform.position;
            agent.destination = destino;
            estadoAlerta = true;
            StartCoroutine("ALERT");
            break;
        }
```

Fonte: O autor (2021)

4.5.3.4 Follow

É o estado de perseguição, normalmente ativo após o estado ALERT. Uma vez que o personagem entra no modo FOLLOW, ele perseguirá seu inimigo, até que ele saia do seu ângulo de visão por três segundos, após isso retorna-se ao estado de IDLE ou PATROL, através de testes de probabilidade.

Quadro 13 – Estado FOLLOW.

```
IEnumerator FOLLOW()
    {
        yield return new WaitUntil(() => !playerVisao);
        print("Mudança de estado - Inimigo perdido!");
        yield return new WaitForSeconds(_GameManager.slimeAlertTime);
        manterEstado(50);
    }

switch (newState)
    {
        case enemyState.FOLLOW:
        olharAlvo();
        destino = _GameManager.player.position;
        agent.destination = destino;
        if(agent.contDistancia <= agent.distanciaParada)
            {
              Attack();
            }
        break;
    }
```

Fonte: O autor (2021)

4.5.3.5 Fury

O modo de fúria consiste na mudança de estado do personagem, em uma situação em que o jogador principal desfere um ataque com a espada. No estado de fúria, o inimigo persegue o personagem principal até que ele seja morto, ou mate o jogador em questão.

Diversos personagens podem entrar em fúria ao mesmo tempo, e seus ataques serão contabilizados de forma separada, onde basta apenas 3 ataques para matar um inimigo, ou receber 3 hits de dano direto para morrer. O modo de fúria não é desativado mesmo que o personagem principal se mantenha fora do campo de visão inimigo, diferente do estado *FOLLOW*.

Quadro 14 – Estado FURY.

```
switch (newState)
    {
      case enemyState.FURY:
      destino = transform.position;
      agent.distanciaParada = _GameManager.slimeAtaqueDis;
      agent.destination = destino;
      break;
    }
```

Fonte: O autor (2021)

4.5.4 Estados de jogo

4.5.4.1 Gameplay

Quadro 15 – Estado GAMEPLAY.

```
void Update()
 {
     if(_GameManager.gameState != GameState.GAMEPLAY) {return;}
     Inputs();
     MoveCharacter();
     UpdateAnimator();
 }
```

Fonte: O autor (2021)

4.5.4.2 Die

Este estado é um dos mais básicos e fundamentais, pois representa o estado de morte do inimigo. Uma vez que o jogador principal ataque 3 vezes, o inimigo (Slime)

irá morrer, uma vez morto, o inimigo realiza a animação de morte e saem definitivamente do mapa.

Quadro 16 – Estado DIE.

```
    if (_GameManager.gameState == GameState.DIE && (state ==
enemyState.FOLLOW ||
    state == enemyState.FURY || state == enemyState.ALERT))
         {
              ChangeState(enemyState.IDLE);
         }


    switch(state)
        {
            case enemyState.DIE:
            destino  = transform.position;
            agent.destination = destino;
            break;
        }
```

Fonte: O autor (2021)

4.6 PÓS-PROCESSAMENTO

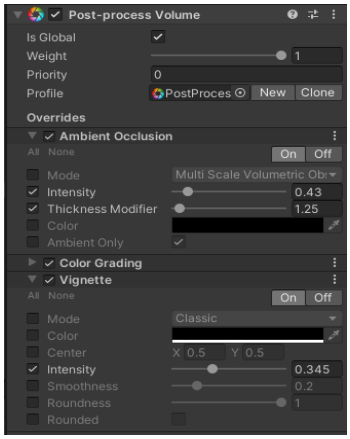
4.6.1 Efeitos Visuais

Para que o jogo ganhe vida visualmente, é necessário adicionar efeitos de pós-processamento e *anti-alising*, tais efeitos permitem o tratamento da qualidade das imagens. O Unity oferece um componente chamado *PostProcessing*, onde é possível definir diversos valores referentes a composição de imagem, como a oclusão ambiental, efeitos de vinheta, saturação e exposição.

Este projeto utilizou dois perfis de processamento, um para quando estivesse dia (*PostProcessingA*) e outro para a noite (*PostProcessingB*). Este simples gerenciamento de vários perfis garante uma variedade de ambientes e climas, é possível gerenciar diversos biomas para mapas apenas configurando os valores de iluminação e objetos físicos.

A figura (29) apresenta a configuração utilizada durante este projeto, estes valores foram salvos em um arquivo de perfil específico, sendo disponível a importação para qualquer projeto desejado.

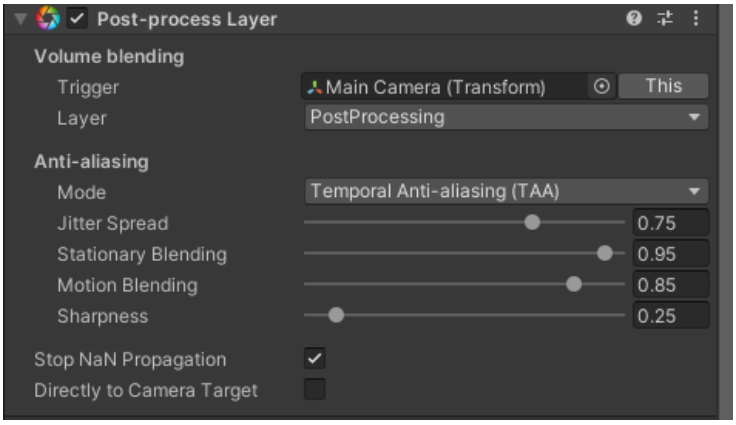
Figura 33 – Filtros visuais.



Fonte: O autor (2021).

Este projeto utilizou um sistema de pós-processamento que busca apenas exemplificar a utilização das ferramentas nativas do Unity em um projeto. Porém, é possível elaborar configurações extremamente profissionais, compatíveis com jogos famosos atualmente, entretanto, demanda muita experiência do desenvolvedor e trabalho árduo.

Figura 34 – Configurações de pós-processamento.



Fonte: O autor (2021).

RESULTADOS

Os resultados obtidos com o desenvolvimento do projeto evidenciam a necessidade de utilizar boas ferramentas durante o desenvolvimento de um jogo, a fim de trazer melhores experiências e ferramentas adequadas às mais diversas necessidades.

A criação de scripts que se referem a cada objeto foi uma das boas práticas aplicadas durante o andamento do jogo, visto que a organização é um fator determinante em projetos de grande escala. Em situações que necessitam de correção de bugs e/ou melhorias, basta apenas atentar-se aos arquivos referentes à área desejada.

Foi possível concluir que é possível desenvolver projetos baseados nos assets gratuitos, disponibilizados na Asset Store, visto que eles apresentam bons materiais e animações completas para as mais diversas situações. Cabe ao desenvolvedor a escolha se os assets gratuitos ou pagos se adequam melhor ao seu projeto, ou a criação autoral dos conteúdos.

As principais noções mecânicas foram apresentadas e exemplificadas no ambiente do jogo, tal qual a movimentação, ataque e morte. Portanto, a relação dos personagens com o ambiente foi definida de maneira dinâmica e buscou-se a correção de bugs maiores.

CONSIDERAÇÕES FINAIS

Através da utilização da Engine Unity com as parametrizações necessárias para o desenvolvimento de um jogo 3D, foi possível constatar que a área de desenvolvimento de jogos apresenta uma vasta oportunidade para desenvolvedores que se interessam pelo mercado de jogos eletrônicos. Além disso, é praticamente obrigatório buscar por um ambiente de desenvolvimento que incentive e forneça o material necessário para a criação de obras voltadas para inúmeras plataformas.

Em relação aos objetivos principais e específicos, é possível concluir que todos foram atingidos e apresentam um resultado satisfatório. Foi possível apresentar fatores básicos como a interface do sistema, até fatores mais complexos como cálculos matemáticos essenciais para a mecânica dos personagens ou do ambiente.

Com base no resultado do projeto, é possível concluir que é possível criar projetos de alto-nível, e especializar-se no mercado de desenvolvimento de jogos utilizando a plataforma Unity, voltada para o desenvolvimento multiplataforma. O desenvolvedor deve ter ciência que a formação profissional nesta área exige muito comprometimento e dedicação, além de uma boa relação entre práticas e trabalho em grupo.

REFERÊNCIAS

RABIN, Steve. **Introdução ao Desenvolvimento de Games - Volume 1: Entendendo o Universo dos jogos.** ISBN 978-8522111435. Cengage Learning. São Paulo, 2011.

ROGERS, Scott. **Level Up! The Guide to Great Video Game Design.** ISBN 978-1118877166. John Wiley & Sons, Ltd. United Kingdom, 2010.

KOSTER, Raph. **A Theory of Fun for Game Design.** ISBN 978-1449363215. O'Reilly Media; 2nd Edition. United States, 2014.

SCHELL, Jesse. **The art of Game Design: A book of Lenses, Third Edition.** ISBN 978-1138632059. CRC Press. United States, 2015.

SKOLNICK, Evan. **Video Game Storytelling: What Every Developer Needs to Know about Narrative Techniques**. ISBN 978-0385345828. WatsonGuptill. United States, 2014.

GREGORY, Jason. **Game Engine Architecture**. ISBN 978-1-4665-6006-2. CRC Press. United States, 2015.

BLACKMAN, Sue. **Beginning 3D Game Development With Unity**. ISBN 978-1430248996. Apress; 2nd Edition. United States, 2013.

PURDUM, Jack. **Beginning Object-Oriented Programming With C Sharp**. ISBN 978-1118336922. Wrox Press; 1st Edition. England, 2012.

BUONANNO, Enrico. **Functional Programming in C#**. ISBN 978-1617293955. Manning Publications. United States, 2018.

TANIMOTO, Jun. **Fundamentals of Evolutionary Game Theory and its Applications**. ISBN 978-4431549635. Springer. Japão, 2015.

STEVE, Swink. **Game Feel**. ISBN 978-0123743282. Elsevier, Inc. United States, 2009.

NYSTROM, Robert. **Game Programming Patterns**. ISBN 978-0990582908. Genever Benning; 1st Edition. United States, 2014.

HEADRICK, Daniel R. **Technology - A World History**. ISBN 978-019515648. Oxford University Press. United States, 2009.

HOCKING, Joseph. **Unity in Action**. ISBN 978-1617292323. Manning Publications Co. United States 2015.

BUNCH, Bryan. **The History of Science and Technology**. ISBN 0-618-22123-9. Scientific Publishing, Inc. United States, 2004.

CHAN, Jamie. **Learn C# in One Day and Learn it Well**. ASIN: B016Z18MLG. Learn Coding Fast; 1st Edition. United States, 2015.

HANSEN, Dustin. **Game On! Video Game History from Pong and Pac-Man to Maria, Minecraft and More**. ISBN 978-1250080950. United States, 2016.

TASCHNER, Rudolf. **Game Changers - Stories of the Revolutionary Minds behind Game Theory**. ISBN 978-1633883734. Prometheus Books. United States, 2017.

HORACHEK, David. **Creating E-Learning Games with Unity**. ISBN 978-1849693424. Packt Publishing Ltd. United Kingdom, 2014.

MECNEIL, Ian. **An Encyclopaedia of the History of Technology.** ISBN 0-415-01306-2. Routledge. United States, 1996.

PARKER, Philip**. World History: From the Ancient World to the Information Age**. ISBN 978-1465462404. Dorling Kindersley, 2017.

GARFINKEL, Simson L. **The Computer Book: From the Abacus to Artificial Intelligence, 250 Milestones in the History of Computer Science**. ISBN 978-1454926214. United States, 2018.

REGAN, Gerard O'. **Introduction to the History of Computing - A Computing History Primer**. ISBN 9783319331379. Springer. Suíça, 2016.

KENT, Steven L. **The Ultimate History of Video Games: From Pong to Pokemon and Beyond... the Story Behind the Craze that Touched Our Lives and Changed the World**. ISBN 978-0761536437. United States, 2001.

GREGORY, Jason. **Game Engine Architecture**. ISBN 978-1-4398-6526-2. A K Peters/CRC Press. New York, 2009.

ZERBST, Stefan; DÜVEL, Oliver. **3D Game Engine Programming**. ISBN 1-59200-351-6. The Premier Press and Thomson Course Technology. United States, 2004.

NORTON, Terry. **Learning C# by Developing Games with Unity 3D**. ISBN 978-1-84969-658-6. United Kingdom, 2013.

BLAIN, John M. **The Complete Guide to Blender Graphics - Computer Modeling & Animations 4nd Edition**. ISBN 978-1-138-08191-8. CRC Press. New York, 2018.

GUMSTER, Jason Van. **Blender for Dummies - A Wiley Brand**. ISBN 978-1-119-04713-1. John Wiley & Sons, Inc. New Jersey, 2015.

BORGES, Dayane. 2020. **Idade do Ferro - Contexto Histórico, Avanços Tecnológicos e Como Terminou**. Disponível em: <https://conhecimentocientifico.r7.com/idade-do-ferro/>. Acesso em: 10/06/2021.

TORRES, Demóstenes. 2020. **Tempos Modernos**. Disponível em <https://www.poder360.com.br/opiniao/justica/tempos-modernos-por-demostenes-torres/>. Acesso em: 11/06/2021.

NOA. 2013. **Códigos Secretos [1].** Disponível em: <http://larbredenoa.blogspot.com/2013/05/codes-secrets-1.html>. Acesso em: 11/06/2021.

Stephen. 2018. **Complemento de Dois: Funcionamento e Exemplos**. Disponível em: <https://www.filipeflop.com/blog/complemento-de-dois/>. Acesso em 12/06/2021.

BARTON, Matt; LOGUIDICE, Bill. **The History Of Pong: Avoid a Missing Game to Start Industry**. Disponível em: <https://www.gamasutra.com/view/feature/132293/the_history_of_pong_avoid_missing_.php>. Acesso em 14/06/2021.

GULARTE, Daniel. 2018. **Space Invaders (Taito, 1978)**. Disponível em: <https://bojoga.com.br/retroplay/analises-de-jogos/arcade-pinball/space-invaders-taito-1978/>. Acesso em: 14/06/2021..

VINHA, FELIPE. 2015. **O Primeiro Pac-Man Tinha Apenas 14kb, Revela Criador do Personagem**. Disponível em: <https://www.techtudo.com.br/noticias/noticia/2015/12/o-primeiro-pac-man-tinha-apenas-14kb-revela-criador-do-personagem.html>. Acesso em:15/06/2021.

MONTEIRO, Rafael. 2018. **Donkey Kong: Confira os 10 Melhores Jogos da Franquia**. Disponível em: <https://www.techtudo.com.br/listas/2018/05/donkey-kong-confira-os-10-melhores-jogos-da-franquia.ghtml>. Acesso em: 16/06/2021.

MONTEIRO, Rafael. 2017. **Lista Traz os Jogos de Super Mario Bros. Mais Icônicos**. Disponível em: <https://www.techtudo.com.br/listas/2017/08/lista-traz-os-jogos-de-super-mario-bros-mais-iconicos.ghtml>. Acesso em: 16/06/2021.

Unity. 2020. **Unity 2020.2 Beta**. Disponível em:<https://unity3d.com/pt/beta/2020.2b>. Acesso em: 16/06/2021.

ARMSTRONG, Martin. 2021. **The Most Addictive Video Game Genres**. Disponível em: <https://www.statista.com/chart/25117/most-addictive-video-game-genres/>. Acesso em: 17/06/2021.

Blender. 2019. **Tutorials - Modeling & Sculpting**. Disponível em: <https://www.blender.org/support/tutorials/>. Acesso em: 20/06/2021.

Unity. 2019. **A Modernized, Refined Editor UI**. Disponível em: <https://unity.com/releases/2019-3/editor-tools>. Acesso em: 20/06/2021.

Blender. 2019. **Blender, Made By You**. Disponível em: <https://www.blender.org/download/releases/2-80/>. Acesso em: 21/06/2021.

ROSA, João Luís Garcia. **Fundamentos da Inteligência Artificial. ISBN 978-85-216-0593-5. LTC - Livros Técnicos e Científicos Editora Ltda. Rio de Janeiro, 2011.**

LEE, Kai-Fu. **AI Superpowers: China, Silicon Valley and the New World Order.** ISBN 978-65-80775-05-7. Rio de Janeiro, 2019.

PEREIRA, André Luiz. **Indústria de games movimentou mais de US$ 120 bilhões em 2019.** Disponível em: <https://www.tecmundo.com.br/cultura-geek/148956-industria-games-movimentou-us-120-bilhoes-2019.html>. Acesso em: 01/11/2021