MATEUS SARTORI ZANDONADI

PROGRAMANDO JOGOS DIGITAIS COM UNREAL 4: UM ESTUDO COMPARATIVO ENTRE A PROGRAMAÇÃO POR BLUEPRINTS E C++

JI-PARANÁ

2019

MATEUS SARTORI ZANDONADI

PROGRAMANDO JOGOS DIGITAIS COM UNREAL 4: UM ESTUDO COMPARATIVO ENTRE A PROGRAMAÇÃO POR BLUEPRINTS E C++

Monografia apresentada à Banca Examinadora do Centro Universitário São Lucas, como requisito de aprovação para obtenção do Título de Bacharel em Sistemas de informação

Orientador: Prof. Maigon Pontuschka.

JI-PARANÁ

2019

SÃO LUCAS
EDUCACIONAL

ATA DE TRABALHO DE CONCLUSÃO DE CURSO

ATA Nº 03/2019 DE TRABALHO DE CONCLUSÃO DE CURSO

No vigésimo primeiro dia do mês de junho de 2019, no horário das 19h às 22h reuniram-se o(a) Orientador(a) professor(a) Prof. Me. Maigon Nacib Pontchuska e os(as) professores (as) Prof. Esp. Diego da Fonseca e Prof. Esp. José Rodolfo Milazzotto Olivas para comporem Banca Examinadora de Trabalho de Conclusão de Curso, sob a presidência do(a) primeiro(a), para analisarem a apresentação do trabalho "Programando Jogos Digitais com Unreal 4: Um Estudo Comparativo entre a Programação por Blueprints e C++". Após arguições e apreciação sobre o trabalho exposto foi atribuída à menção como nota do Trabalho de Conclusão de Curso do(a) acadêmico(a): Mateus Sartori Zandonadi.

Obs: Trabalho de Conclusão de Curso (X) aprovado ou ( )reprovado com nota total de 9,7 (nove pontos e sete décimos) pontos, sendo atribuídos o valor 9,6 (nove pontos e seis décimos) ao trabalho escrito e 9,9 (nove pontos e nove décimos) à apresentação oral.

Mateus S.
Mateus Sartori Zandonadi

Maigon Nacib Pontchuska
Prof. Me. Maigon Nacib Pontchuska
Orientador

Diego da Fonseca
Prof. Esp. Diego da Fonseca

Prof. Esp. José Rodolfo Milazzotto Olivas

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

Quem me oferece sua gratidão como sacrifício honra-me, e eu mostrarei a salvação de Deus ao que anda nos meus caminhos.

Salmos 50:23

RESUMO

Os *game engines*, ou motores de jogos, são tecnologias concebidas para facilitar o desenvolvimento de jogos digitais. Por outro lado, um jogo deve ter sua estrutura de programação bem otimizada, a fim de melhorar o desempenho. No contexto da indústria de jogos, o desempenho é uma característica chave e a linguagem de programação usada pode influenciar diretamente nisso. O Unreal Engine 4 é uma ferramenta que fornece aos desenvolvedores capacidades para apresentar seus projetos no mercado de forma prática e simples com a programação de linguagem Blueprints ou C++. Este estudo pretende compreender a operação dessas duas formas de programação de jogos no Unreal Engine 4 e o desempenho dos games ao usar uma ou outra opção que o motor de jogo oferece. Neste trabalho, um experimento de comparação foi realizado usando um computador pessoal com os requisitos necessários para o teste com o Unreal Engine 4. O mercado já afirmava que a linguagem C++ era mais eficiente e começamos a partir dessa hipótese, mas seria importante testá-la e saber se isso é verdade e em que medida. O objetivo final do experimento foi analisar as vantagens e desvantagens de uma linguagem em relação à outra ao criar jogos com Unreal Engine 4 e comparar a eficiência de ambas. Como resultado, provamos que a linguagem C++ pode atingir um desempenho até 13 vezes mais rápido do que pela programação por Blueprints

Palavras-chave: Unreal Engine 4, C++, Blueprints, Jogos

ABSTRACT

Game Engines are technologies designed to facilitate the development of digital games. ON th other hand, a game must have its programming structure well optimized in order to improve performance. In the context of the gaming industry, performance is a key feature and the programming language used can directly influence this. The Unreal Engine 4 is a tool that provides developers with capabilities to present their projects in the market in a practical and simple way with Blueprints or C++ language programming. This study intends to understand the operation of these two forms of game programming in Unreal Engine 4 and the performance of the games when using one or the other option the game engine provides. In this work, a comparison experiment was performed using a personal computer that complied with the necessary requirements for testing with Unreal Engine 4. The market has already stated that the C++ language was more efficient and we started from this hypothesis, but it would be important to test it and to know if this is true and to what extent. The final goal of the experiment was to analyze the advantages and disadvantages of one language with regard to the other when creating games with Unreal Engine 4 and comparing the efficiency of both. As a result, we proved that the C++ language can achieve a performance up to 13 times faster than by Blueprints programming.

Keywords: Unreal Engine 4, C++, Blueprints, Games

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS

2D	Duas dimensões
3D	Três dimensões
FPS	First-Person-Shoot
UE	Unreal Engine
MMO	Massively Multiplayer Online
MMORPG	Massively Multiplayer Online Role-Playing Game
RPG	Role-Playing Game
GDD	Game Design Document
VR	Virtual Reality

1. INTRODUÇÃO

Recentemente ouvimos sobre a evolução de tecnologias e com isso games também tem um grande impacto nessa evolução com os motores gráficos. Quando se fala em games, falamos de algo que pode se revolucionário até mesmo como algo educacional (HUIZINGA, 2001). Com o surgimento dos games diversos conceitos foram apresentados como por exemplo um game pode ter sua funcionalidade para formas de ensino e culturas (PEREIRA, 2013 *apud* CALLOIS, 1990*)*.

A partir dos motores de jogos é possível criar jogos digitais para diversos conceitos tanto quanto para educacionais como para entretenimento. Neste caso, o aspecto de um jogo tradicional pode ser apresentado por uma máquina virtual criando regras lógicas de programação com as ferramentas existentes nos dias atuais.

No capítulo 4 apresentaremos os conceitos de jogos tradicionais e como partimos para os jogos digitais e também os gêneros existentes, neste capitulo veremos ainda as formas conceituais de como desenvolver um projeto utilizando conceitos famosos como o Game Design Document. No capítulo 5 falaremos sobre uma das ferramentas e suas características que torna possível a criação de um jogo digital conhecido como Motor de Jogos e também os softwares de programação para criar a lógica do jogo, isto é, regras para o conceito que cada jogo possui. No capítulo 6 apresentaremos o método que será realizado para a criação de um protótipo sobre um conceito de testes de comparação de linguagem e verificar como resultado a viabilidade das linguagens e o impacto de performance e desempenho dos jogos. No capítulo 7 será definido a prototipação do projeto e o conceito para a criação do jogo, contendo informações sobre como o jogo será em questão de gênero, notabilidade, personagens e outras características. No capítulo 8 estará destacado o cronograma para colocar em prática toda a ideia conceitual do protótipo que será desenvolvido com base os estudos que será adquirido. Por fim no capitulo 9 finalizaremos com as considerações finais.

2. PROBLEMATIZAÇÃO

Os jogos eletrônicos sempre foram conhecidos como uma maneira de entretenimento e diversão para quem joga. Ao longo dos últimos anos, com a evolução da tecnologia, surgiram muitos motores de jogos ou Engines para o desenvolvimento de games. Entretanto, para o desenvolvimento de um jogo não basta apenas a ideia e a criatividade do desenvolvedor mas,além da lógica de programação, também é necessário que um dispositivo tenha uma boa performance durante o jogo e as formas de programação podem influenciar neste aspecto.
Neste trabalho, pretendemos estudar o motor de jogos Unreal 4 e as duas formas possíveis de programação de jogos que proporciona: por meio dos chamados “Blueprints”, que são uma forma gráfica de programar, bem como o uso de programação C++ e a performance de cada uma delas. Nosso trabalho procura responder quais são as vantagens e desvantagens de utilizar Blueprints ou C++ na programação de jogos digitais utilizando Unreal 4.

2.1. HIPÓTESE

Algumas alegações de vantagens de utilização das Blueprints em relação à programação C++ no Unreal 4 se referem ao fato de que, com as Blueprints, o usuário não precisaria aprender programar em linguagens de programação, mas bastaria entender de lógica de programação, pois as *Blueprints* funcionam por meios de ligação em uma interface baseada em nós para criar elementos de jogabilidade com a mesma lógica de programação tradicional. A desvantagem das Blueprints estaria no tempo de processamento e execução o que acarretaria em um desempenho não muito agradável ao jogador.

Por outro lado, programar um jogo em C++ no Unreal 4 exige um certo nível de conhecimento, mas, a vantagem estaria em um processamento e execução bem mais rápidos do que na programação em Blueprints. Queremos testar se estas alegações são verdadeiras.

2.2. DELIMITAÇÃO

O presente estudo procurará fazer uma comparação do desenvolvimento de jogos utilizando a programação baseada em Blueprints e a C++ no Unreal 4. Um mesmo jogo será criado com as duas técnicas e tanto o processo de criação como a performance final deste jogo serão mensurados e avaliados. Não se trata propriamente da criação de um game para entretenimento, mas um game de teste apenas.

2.3. OBJETIVO GERAL

Elaborar uma apresentação sobre a utilização do Unreal 4 usando Blueprints e C++ como programação no desenvolvimento de jogos, e comprovar as vantagens e as desvantagens da utilização de cada uma das técnicas do ponto de vista do processo de produção e da performance final.

2.4. OBJETIVOS ESPECÍFICOS

- Levantar informações sobre o desenvolvimento de jogos utilizando o Unreal 4.
- Fazer uma comparação do processo de produção com Blueprints e C++.
- Fazer uma comparação do desempenho do jogo desenvolvido com Blueprints e C++.
- Identificar as vantagens e desvantagens da utilização de Blueprints e C++ no Unreal 4

3. JUSTIFICATIVA

Hoje existem vários meios para se desenvolver jogos com o auxílio de vários motores de jogos, ou *Engines*.

Segundo Dias (2017), os motores de jogos são programas executados em computadores com o objetivo de criar um cenário com uma variedade de elementos renderizados em gráficos de duas dimensões (2D) ou três dimensões (3D), simulando algo equiparado ao mundo real com um conjunto de eventos dinâmicos como: sons, inteligência para o computador, animações e a física por exemplo.

Os motores de jogos estão cada vez mais evoluídos com a capacidade de criar uma visualização muito realista e com um bom desempenho para os mais diversos dispositivos, desde PCs a consoles como o Playstation IV, o X-Box One e mesmo até para dispositivos móveis. Quando se trata do Unreal Engine, desenvolvida pela Epic Games e com desenvolvimento multiplataforma, estamos falando de facilidade em programação de jogos, além de renderização de gráficos poderosa. O Unreal 4 é um motor de jogos que está despontando no mercado de produção de games de tipo AAA, e vem ganhando espaço em relação ao seu principal concorrente, o Unity. Muito já se fala e se estuda sobre o Unity, mas ainda são poucos os estudos com Unreal. Por isso, optamos por fazer este estudo com ele.

4. REFERENCIAL TEÓRICO

Faremos, a seguir uma breve revisão acerca do conceito de jogos em geral. Também abordaremos a origem e evolução dos jogos digitais.

4.1. CONCEITO GERAL DE JOGO

Segundo Huizinga (2001), jogos são atividades sem restrições capazes de nos tirar da vida que estamos acostumados como, trabalhos, deveres, a vida diária em si, nos levando a um envolvimento de alegria ou preocupação seguindo um certo padrão de restrição. Sendo assim é uma ação livre que possui regras, com a expectativa de criar grupos, sendo fora da vida diária (HUIZINGA, 2001, p. 18).

Assim jogos em geral podem ser executados em diversos modos tendo a sua serventia até mesmo para a aprendizagem desenvolvendo sua capacidade em forma de brincadeiras, não só isso, mas, trabalhar em culturas (PEREIRA, 2013 *apud* CALLOIS, 1990).

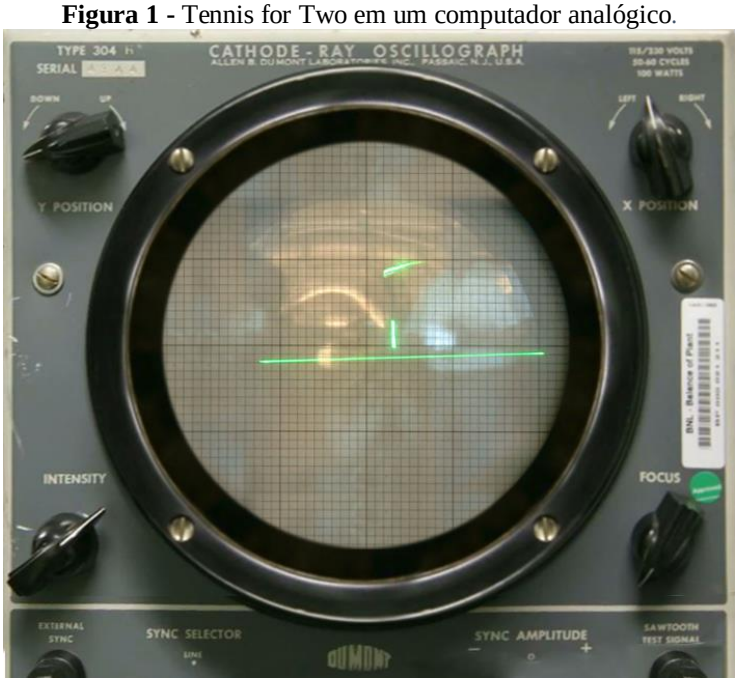
Segundo Crawford (1984, p. 16) nos jogos as crianças podem aprender de forma divertida e até mesmo em um mundo de fantasias. Sendo assim, os jogos podem de fato ser um instrumento para ser utilizado na didática.

4.2. COMO SURGIRAM OS JOGOS DIGITAIS?

O primeiro jogo digital surgiu em 1958 e se chamava Tennis for Two. O jogo representava uma simulação de um jogo de tênis. Era produzido em um computador mainframe em um display redondo de um osciloscópio. Gerava um gráfico de duas dimensões que mostrava a trajetória de uma bola de tênis e que tinha duas barras, uma em cada lado, para rebater esta bola. Tennis for Two não foi um game comercial. Era apenas uma representação desenvolvida no laboratório “Brookhaven National Laboratory” para mostrar a potencialidade do computador mainframe da instituição em um dia de visitas da comunidade. (PACHECO, 2013).

O jogo contava com um controle que possuía dois botões: um que girava e o outro para ser pressionado, a função desses dois botões era de fato para a dinâmica proposta pelo game, sendo possível então bater a bola e também ajeitar o ângulo para rebater.

Figura 1 - Tennis for Two em um computador analógico.



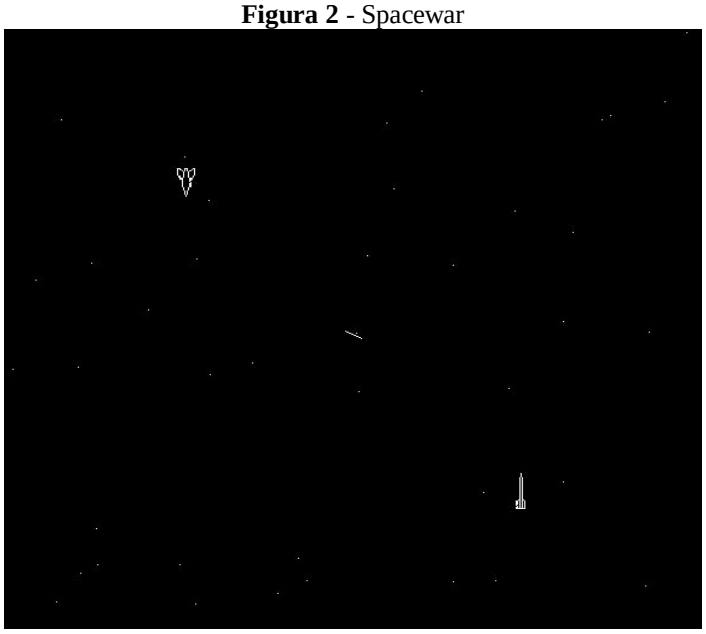
Fonte: CIRO, 2014

Segundo Bellis (2017), na década de 1960, um programador chamado Steve Russell do Massachusetts Institute of Technology, deu origem a um jogo popular chamado Spacewar (Guerra Espacial), era um jovem que buscou inspirações nas obras escritas do autor Doc. Smith.

Russell era o cabeça por de trás da equipe do jogo que viria a ser intensamente conhecido. O jogo demorou muitas horas para ser finalizado, era produzido em um computador conhecido como PDP-11, O computador também era conhecido por ser um computador com interação na época.

Além disso, de acordo com Bellis (2017) Spacewar nunca lucrou com os seus feitos. Passou a ser produzido com a ideia de gerar lucros somente quando Russell mudou para a universidade de Stanford e passou seus conhecimentos a um jovem chamado Nolan Bushnell, que deu início à Atari Computers, e fez o primeiro jogo árcade que era movido por dinheiro real. Assim como Tennis for Two, Spacewar também pode ser considerado uma das raízes para os jogos digitais atuais pois, de fato, esses dois jogos contribuíram para o que veríamos a chamar de jogos digitais.

Figura 2 - Spacewar



Fonte: BARTON e LOGUIDICE, 2009

4.3. O QUE SÃO JOGOS DIGITAIS

Os jogos em uma plataforma digital, ou seja, fora de jogos que envolvam peças, no caso os tabuleiros e entre outros jogos que não utilizam as máquinas, acolhe esse nome por serem executados em equipamentos tecnológicos como tablets, computadores, celulares atualmente.

Ainda seguindo o raciocínio (HUIZINGA, 2001) os jogos sempre foram um meio de entretenimento, e dependendo do modelo de jogo que o jogador está lidando, ele é capaz de desenvolver habilidades observar culturas e também absorver tudo capaz de ser proporcionado baseado nas regras e conteúdo.

> Poderíamos considerá-lo uma atividade livre, conscientemente tomada como "não-séria" e exterior à vida habitual, mas ao mesmo tempo capaz de absorver o jogador de maneira intensa e total. É uma atividade desligada de todo e qualquer interesse material, com a qual não se pode obter qualquer lucro, praticada dentro de limites espaciais e temporais próprios, segundo uma certa ordem e certas regras. (HUIZINGA, 2001).

No caso desses jogos eletrônicos o mesmo conceito se aplica, a única diferença é que esses jogos são elaborados em um mundo de fantasia que utilizam os computadores como requisito.

4.4. IMPORTÂNCIA DOS JOGOS DIGITAIS

Com base nos conceitos que já vimos e também nas palavras de Huizinga (2001), nós podemos aprender bastante com jogos pois, se trabalhada de maneira correta e a ideia por de trás de um jogo digital for realmente boa, ela pode nos ensinar muito, de natureza igual a uma maneira inesperada.

Segundo Magnani (2017) os jogos eletrônicos atingem amplo escalão de jovens e também podem auxiliar desenvolver uma série de capacitações de uma pessoa.

Para que um jogo seja usado como uma utilidade de ensino é preciso uma tarefa bastante complexa em associação a isso, pois quem desenvolve um jogo precisa ter uma base ampla de conhecimento, isso é o que motiva o jogador a ter simpatia com o que está processando-se.

> Se um jogo tem péssimos princípios de aprendizagem em seu design, então não será aprendido nem jogado e não vende bem. Seus designers buscarão trabalho em outro lugar. No final, então, videogames representam um processo, graças ao que Marx chamou de "Criatividade do capitalismo", que leva a melhores e melhores projetos para uma boa aprendizagem e, de fato, boa aprendizagem de coisas difíceis e desafiadoras. (GEE, 2003)

Criar um bom jogo não é tão simples quanto pode parecer. É primordial que haja um entendimento sobre o conteúdo a ser trabalhado. Se o conteúdo for bem trabalhado, o jogo fará com que uma pessoa aprenda de maneira divertida. Um jogo digital pode despertar a imaginação pois, pode representar um ambiente com desafios e colocar o jogador para resolve-los, melhorando sua capacidade de aprender; Alguns jogos podem até mesmo te forçar a aprender outros idiomas, para entender o que é apresentado (DIAS, FURLANETI, *et al.*, 2014).

Um exemplo disso é o jogo God of War, produzido pela Santa Monica Studio. Este jogo explora muito a mitologia grega e, pelo fato do protagonista ser um espartano, também é possível adquirir conhecimento através do jogo sobre como eram os treinamentos dos soldados de Esparta. Por meio do jogo é possível despertar uma pessoa para o conhecimento de história sobre a mitologia grega, conhecer os deuses do Olimpo e suas características, por exemplo.

Isso é uma das coisas que God of War pode trabalhar, mas, não literalmente como a mitologia grega é em si, pois o jogo possui sua própria história e narrativa. Esse modo de contar uma história através de um jogo, pode ser feito de maneira divertida fazendo com que o jogador fique, de fato, interessado na mensagem que um game passa.

Figura 3 - God of War 3

Fonte: BAHNER, 2015

4.5. GÊNEROS DE JOGOS

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

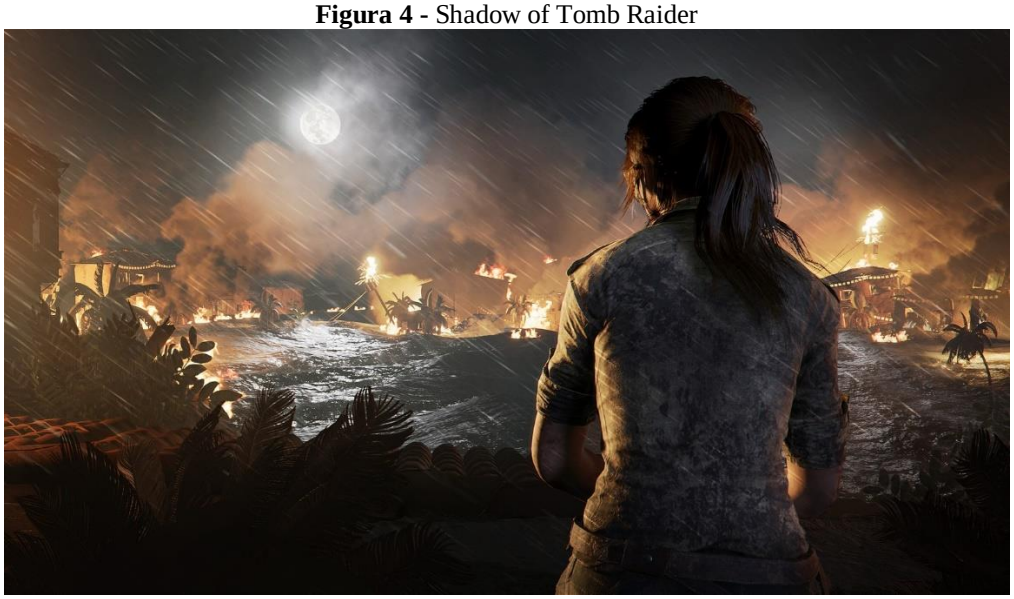
4.5.1. AVENTURA

O gênero de aventura consiste em uma história de aventura que contém vários níveis de jogo. O desafio tem a tendência de aumentar cada vez mais, surgindo novos quebra-cabeças e colocando o jogador para pensar para prosseguir com o nível (CARDOSO, 2017).

Ainda de acordo com Cardoso (2017) existem vários estilos de jogos que contribuem para o gênero de aventura, como jogos que levam o jogador a explorar o mapa do jogo para encontrar itens ou fazer missões secundárias, que fazem com que o jogador percorra todo o mapa do jogo.

Segundo Werneck (2018) um jogo que podemos citar é o jogo Shadow of Tomb Raider. A protagonista é Lara Croft, que, durante sua jornada, precisa fazer várias ações com o propósito de acabar com o apocalipse Maia. A protagonista percorre o caminho desde o México até o Peru. Durante essa jornada. Passa por muitos perigos e tem que resolver uma série de enigmas. Deste modo, o jogo se encaixa no gênero aventura.

Figura 4 - Shadow of Tomb Raider



Fonte: WERNECK, 2018

4.5.1. AÇÃO

Os jogos de ação são muito semelhantes aos de aventura, mas, de acordo com Oxford (2018) o seu diferencial é em fazer com que o jogador pense de maneira mais rápida, testando a uma reação do jogador diante de uma situação. Esse tipo de gênero é bastante associado com os jogos como GTA V, Call of Duty, Max Payne, entre outros. Em geral, destacam-se muito os jogos de tiros, como Battlefield, Counter-Strike Global Offensive, Rainbow Six Siege, The Division.

O site da UBISOFT (2018) sobre jogos desenvolvidos nessa categoria aponta Rainbow Six Siege, como um representante desta categoria. Trata-se de um jogo multiplayer composto por cinco membros em duas equipes que coloca uma equipe contra a outra, exigindo aos jogadores que joguem de maneira cooperativa e que tenham que criar uma estratégia contra o time adversário para conseguir atingir seu objetivo.

O jogo tem uma variedade de personagens e cada uma com a sua função específica. Os jogadores devem escolher o personagem com quem mais se identificam ou de acordo com a função estratégia que cada um tem no jogo. Esse tipo de jogo nos mostra como cada pessoa pode reagir dependendo do seu próprio jeito de jogar. Umas podem optar por não seguir os seus comandos e outras podem seguir os seus comandos dando interatividade entre os jogadores e preparando o jogador até mesmo para lidar com pessoas, pois, cada um age conforme o seu propósito.

Figura 5 – Tom Clancy's Rainbow Six Siege

Fonte: GAMERHUB, 2018

4.5.2. CORRIDA

Esse gênero dá muita atenção ao aspecto da colisão para calcular a maneira como o veículo irá reagir em uma variedade de situações que simulam a realidade. De acordo com Tiago (2017) os climas e as variáveis no ambiente influenciam na estrutura do carro trazendo mais dinâmica. Estas variáveis podem ser chuva, pista escorregadia, curvas, etc.

Os jogos de corrida, além de simular a direção de um veículo, em sua maioria focam na competição, proporcionando diversos aspectos na jogabilidade que intensificam o aspecto competitivo como, caminhos de atalho, veículos com dispositivos especiais e com características de dirigibilidade deferentes, entre outros.

Podemos citar nessa categoria Need for Speed Most Wanted (2005). Segundo o site da Techtudo (2010), o jogo se passa inteiramente com o protagonista dentro de um carro. A finalidade do jogo é fazer com que o protagonista fuja da polícia para ser considerado um dos maiores no que eles chamam de "Blacklist", a lista negra dos procurados. Para conseguir isso, o jogador deve ganhar inúmeras corridas e subir na hierarquia. O jogo também possui um sistema de mundo aberto em que o jogador pode optar por realizar missões secundarias para conseguir melhorias com o decorrer da campanha, assim chegando ao estágio final em condições de competir com o oponente que está em primeiro lugar da Blacklist.

Figura 6 - Need for Speed Most Wanted (2005)

Fonte: HUNSBERGER, 2017

4.5.3. ESPORTE

Esses tipos de jogos se baseiam em esportes relacionado aos que existem no mundo real, como o vôlei, basquete, ping-pong entre outros (TOSCHI, 2012). Nesse gênero, o jogador pode jogar um jogo de esporte como se fosse uma simulação na vida real. Alguns destes jogos contam com estratégias, outros contam com habilidades do jogador.

O jogador pode até mesmo gerenciar o seu time como se fosse um treinador em casos de jogos como a franquia da FIFA, esses se encaixam na categoria de gerentes (COSTA, 2014).

Segundo Toschi (2012) esses jogos também podem conter misturas de outros elementos como o RPG. Neste caso, o jogo pode apresentar componentes como habilidades, atributos, equipamentos para melhoria, exigindo também a cooperação de outros jogadores para atingir o objetivo. Nesta categoria podemos colocar o jogo de basquete NBA 2K18, lançado oficialmente há pouco tempo. De acordo com Seibel (2017) trata-se de um jogo esportivo baseado no basquete desenvolvido pela empresa 2K Sports. Neste jogo, o jogador controla o seu personagem para disputar partidas de basquete em equipe podendo alternar entre o modo offline e online. Os prêmios são calculados de acordo com o seu desempenho no jogo.

Figura 7 - NBA 2K18

Fonte: JOHNSTON, 2017

4.5.4. RPG

Segundo Toschi (2012), jogos de RPG têm por finalidade colocar o jogador na pele de um protagonista e contar a história por meio deste personagem, como se fosse uma pessoa lendo uma história em um livro. Esse gênero pode conter vários modos de batalha. Dependendo do tipo do jogo, podem conter batalhas em turno, em tempo real (conhecido como Action RPG), batalhas laterais etc. Além disso, esse gênero costuma colocar vários lugares para se explorar a fim de colocar o jogador para conseguir novas habilidades e aprimorar equipamentos.

Esse gênero, apesar de conter um modo campanha onde se passa a história, pode conter também um modo online. Nos modos online os jogadores lutam todos os dias para melhorar no ranking de posições dos diversos jogadores.

Figura 8 - Chrono Trigger (RPG)

Fonte: BATISTA, 2018

4.5.5. PLATAFORMA

Esse gênero se baseia em níveis. Nos jogos de plataforma, o jogador deve controlar o personagem até o final do mapa para conseguir passar de nível. São caracterizados por terem a jogabilidade no estilo 2D ou 3D (CARDOSO, 2017).

Esses jogos são conhecidos como o estilo *Side-Scrolling* ou Rolagem-Lateral. O jogador percorre o mapa e os desafios surgem em diversas variáveis podendo conter novos inimigos e também itens para o ajudar. Jogos como *Sonic the Hedgehog*, Nintendo. (1991). Ou *Super Mario Bros.* Nintendo. *(1985).* Da Nintendo se encaixam nessa categoria. Ambos contêm o estilo de rolagem lateral. Em todos os níveis o jogador deve percorrer até o fim do mapa para subir de nível até chegar à última fase, que é o desfecho da história onde geralmente o jogador encontra o "chefão" ou "big boss" e precisa lutar contra ele para terminar o jogo (KLAPPENBACH, 2018).

Figura 9 - Super Mario World

Fonte: MAGE, 2016

4.5.6. SIMULAÇÃO

Jogos de simulação tentam trazer o que há no mundo real para dentro de um jogo virtual, podendo também trazer um evento que já aconteceu ou um aspecto de como seria se acontecesse. Esses jogos podem estar associados com outros gêneros, mas o foco principal ainda é a simulação (MARCHELLETTA, 2016). Neste gênero podem ser destacados jogos como simuladores de voo, simuladores de vida e simulações de sobrevivência.

Como exemplo de jogo desta categoria podemos citar o *DayZ***,** Bohemia Interactive. (2018). Segundo a Bohemia Interactive (2018), trata-se de um jogo de mundo aberto online, com um estilo *survival horror* em que encontramos a simulação de um mundo pós-apocalíptico de zumbis. O jogador controla o seu personagem como se estivesse vivendo naquele mundo com outros jogadores. Toda decisão que for tomada pode mudar a história do jogo, tendo em mente que o jogo coloca o jogador para montar a sua própria história dentro deste ambiente.

Conforme Karasisnki (2012), os jogos de simulação nos permitem fazer algo que na vida real não poderíamos fazer. Os jogos de simulação permitem também não apenas criar algo do zero, mas também exercitar a nossa criatividade de modo individual, sem restrições como por exemplo, gerenciar um banco, pilotar um avião e outras categorias.

Figura 10 - DayZ

Fonte: FELIPE, 2018

4.5.7. PUZZLE

Para Newman (2018) jogos de tipo puzzle se baseiam em quebra-cabeças e como solucioná-los. Essa categoria pode ter modalidades como terminar palavras incompletas e obter soluções para continuação lógica. Alguns jogos de quebra-cabeça têm um limite de tempo para que o jogador termine o desafio. Esses jogos possuem enigmas para serem solucionados, sendo também possível a mistura desse gênero com outros gêneros de jogos. Segundo Toschi (2012) puzzles são ótimos para gêneros de RPG, Plataforma e Ação, por exemplo.

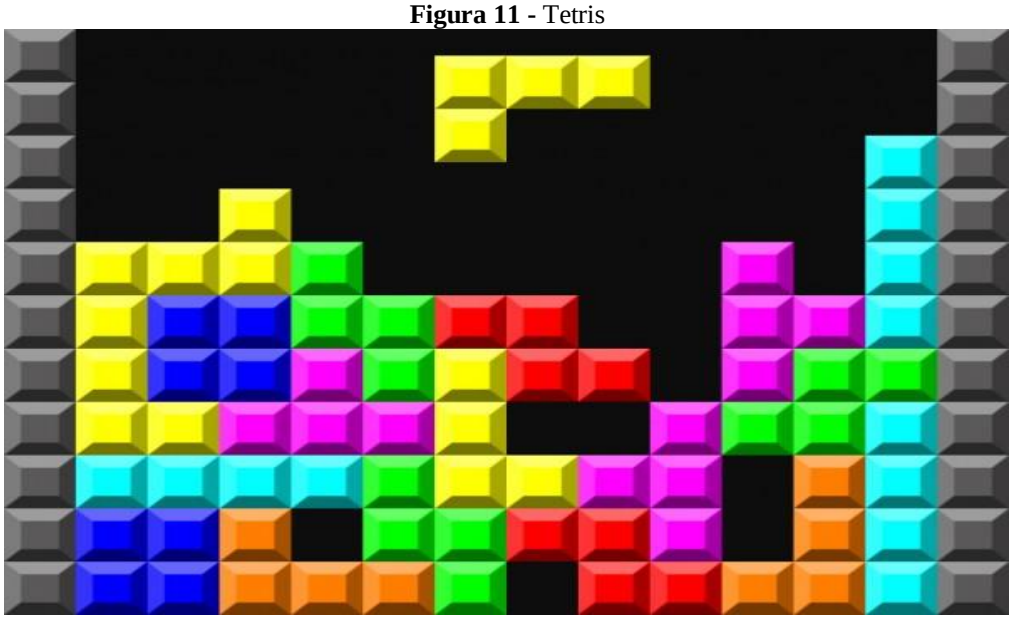
Segundo Porto (2018), Tetris como referência é um jogo de quebra cabeça, que possui blocos de formas variadas que devem ser empilhados para completar linhas horizontais. As linhas completadas desaparecem. Se a linha não é preenchida completamente, os blocos são empilhados de modo que a tela vai ficando cada vez mais cheia. Quando a tela enche de blocos com linhas não completadas o jogo acaba.

Trata-se de um jogo de perspicácia e habilidade. Porto (2018) destaca que jogos como o Tetris podem ajudar uma pessoa a desenvolver capacidade e eficiência na atividade cerebral, conduzindo-a a pensar de maneira rápida, e processar de informações cada vez melhor.

Segundo Gonçalves (2018), esse tipo de jogo coloca o jogador em diversos tipos de situação para o desenvolvimento de raciocínio lógico, tendo aplicabilidade até mesmo em sala de aula.

O Tetris foi uma das grandes inspirações para jogos Puzzle e também contribuiu para gêneros misturados que envolvem quebra-cabeças em que o objetivo é colocar o jogador para pensar e solucionar problemas por meio de raciocínio lógico matemático.

Figura 11 - Tetris



Fonte: NEMIROFF, 2014

4.5.8. ESTRATÉGIA

De acordo com Mendonça (2018), jogos de estratégia tiveram origem em jogos de tabuleiros como o WAR. A maioria desses jogos é jogada por turno, ou seja, cada jogador faz suas ações e os outros jogadores esperam por sua vez. Porém, existem outros jogos de estratégia além de jogos baseados em turnos como, por exemplo, estratégia em tempo real.

Para Dias e outros (2014) em um jogo de estratégia é necessário agir como um administrador para completar uma meta, fazendo com que o jogador resolva o problema baseado na lógica e nos recursos que possui. Sendo assim, o jogador é incentivado a utilizar suas capacidades e pensamento lógico para resolver o problema. Esse gênero de estratégia pode estar associado a outros gêneros como RPG, Ação, FPS entre outros.

Jogos de estratégia podem servir para interações educativas, contudo, a ideia de um jogo precisa ser realmente criativa e estimuladora, fazendo com que o jogador aprenda algo com o que está sendo jogado.

Figura 12 - War

Fonte: G1, 2015

4.5.9. LUTA

Segundo o site da Revista Gestão Universitária (2014), jogos de luta teve a inspiração baseado em lutadores da antiga Grécia conhecidos como gladiadores.

Para Tiago (2017) jogos de luta se encaixam em uma categoria em que dois personagens se enfrentam de um para um com golpes e poderes especiais. A jogabilidade permite ao jogador realizar *combos*, ou seja, uma série de golpes combinados para causar mais dano ao oponente e assim derrotar o adversário.

De acordo com Pinheiro (2018) para realizar esse tipo de procedimento é preciso atingir um timing perfeito, fazendo com que o jogador seja atento com o seus reflexos e agilidade para realizar golpes. O jogador treina sua agilidade e seus reflexos, assim também como desviar de golpes e magias. Exemplos são Jogos como *Dragon Ball Budokai: 2. (*GameCube, Bandai Namco). *Naruto Shippuden* (CyberConnect2, Bandai Namco), *Street Fighter (*Hiroshi Matsumoto, Takaishi Nishiyama e Yoshiki Okamoto, Capcom*), Mortal Kombat* X (Ed Boon e John Tobias, WarnerBros), entre outros.

Os jogos de gênero de luta se encontram geralmente em ambientes 2D ou 3D. O combate dura algum tempo e o jogador que cair antes do tempo ou perder toda a "vida" é derrotado.

Figura 13 - Mortal Kombat X

Fonte: MONTEIRO, 2015

4.6. METODOLOGIA DE DESENVOLVIMENTO DE JOGOS

Antes de desenvolver qualquer tipo de projeto é necessário que se tenha alguma metodologia como base para sua criação. No caso dos jogos digitais não é diferente. Existem diversas maneiras como base para a criação de jogos e como definir meios de jogabilidade, ambientação, efeito sonoro, tipo de jogo, personagem e bem como seu processo de criação.

Para Mallmann (2012) para criar um jogo temos que ter um "template" para colocar as ideias e assim construir uma base do jogo que será desenvolvido. Esse processo se passa por estágios. Para cada estágio uma equipe deve definir o que será feito em relação ao desenvolvimento. O processo de design se caracteriza por quatro atividades fundamentais, o conceito, meio artistico, as funções e outras. Um bom planejamento deve discutir uma série de outras questões práticas como: qual o software de desenvolvimento, qual software de modelagem, software de animação e outros devem ser utilizados.

Um meio para resolver esse tipo de problema é usando o método do *Brainstorming*, que tem por objetivo colocar um grupo específico em um modo de reconhecimento para solucionar um problema de um determinado cenário. "Brainstorming ou "tempestade de ideias" é uma técnica para explorar o potencial de ideias de um grupo de maneira criativa e com baixo risco de atitudes inibidoras (LIMA, 2011, p.3).

Oxland (2004) menciona um método muito utilizado para o desenvolvimento de games que se caracteriza pelo *Game Design Document*,(GDD) e que sem ele a probabilidade do projeto ser um desastre é alta. Um GDD pode ser visto como um protótipo da ideia inicial do jogo que ajuda a organizar a equipe para visar o objetivo principal de como estruturar o jogo.

Segundo Rogers (2010) a escrita do GDD pode se desenvolver por meio de três momentos que chama de “The one – sheet” (Folha única), “The Ten-Pager” (O Dez páginas), “GDD completo”.

4.6.1. FOLHA ÚNICA

O método “The one–sheet” ou “folha única” consiste na primeira descrição do jogo que se pretende criar. Escreve-se somente uma página com as definições básicas do jogo: título do jogo, gênero, sistema/console do jogo, faixa etária, classificação de software e entretenimento intencionado, resumo da história e o foco do jogo, modos de jogabilidade, pontos de vendas e quais seriam os produtos semelhantes com os quais competiria (ROGERS, 2010, p.60).

4.6.2. O DEZ PÁGINAS

O “*Ten Pager*” seria uma versão um pouco mais elaborada, com todas as informações do “One-sheet” ampliadas para mostrar mais detalhes. Ajuda que o leitor tenha uma rápida noção sobre o projeto sem se aprofundar muito no conteúdo ainda, sendo possível que o leitor seja até mesmo um financiador em potencial. Nesse caso é preciso certificar-se de que está tudo adequado, se preocupando com o visual, layout, formatação etc.

O modo “Dez Páginas” é um momento essencial para o esboço das características mais detalhadas do jogo. Segundo Rogers (2010), o esboço de dez páginas deve conter algumas caracteristicas como o título de jogo, o tipo de plataforma que o jogo irá ser disponibilizado, faixa etária, data de lançamento, além disso deve conter um resumo da história também. No mais, deve fornecer diagramas de jogabilidade, frases curtas e fortes para descrever o jogo que podem ser usadas como marketing, uso da terminologia para a especificação da intenção do jogo, imagens de *Concept Art*, exemplos descritivos e vivos, usar jogos modernos como títulos de comparação (ROGERS, 2010, p. 62).

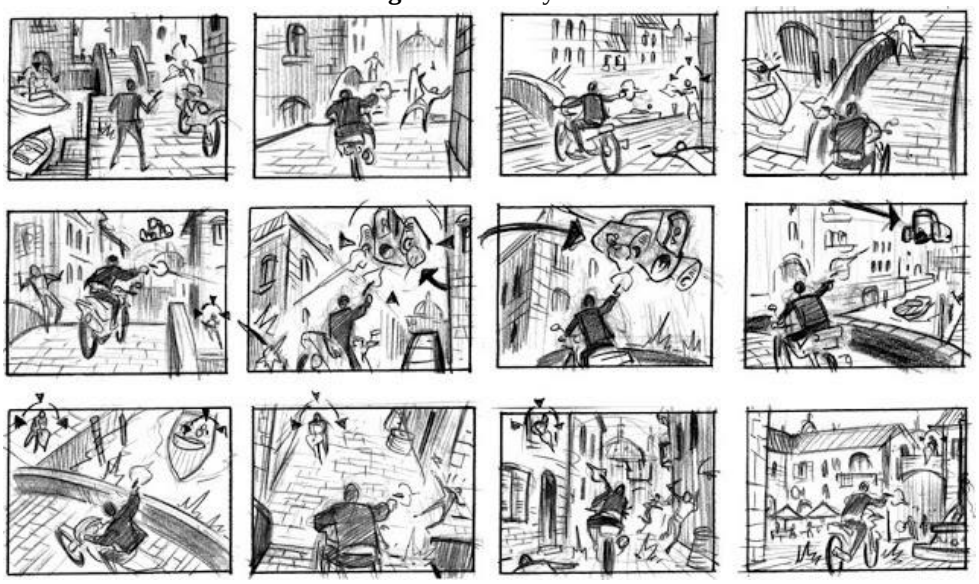
4.6.3. O GAME DESIGN DOCUMENT

De acordo com Rogers (2010), o GDD é um documento bem mais completo. Ele coloca tudo o que você fez antes nos dois modos anteriores com a intenção de fornecer informações completas e profundas a respeito do projeto. Os GDDs acabam sendo extensos porque todas informações e detalhes sobre o jogo são colocadas de forma organizada, é um documento vivo que deve ser atualizado com cada nova ideia para o jogo, tornando-se um documento a ser seguido por todos os profissionais que estão desenvolvendo o jogo. Torna-se um documento imprescindível para o levantamento de fundos e para a apresentação do projeto para investidores.

4.6.3.1. STORYBOARDS

O GDD pode e deve conter um ou diversos *storyboards*. O *storyboard* é uma técnica que também é utilizada em filmes, desenhos e etc. Os *storyboards* são histórias em quadrinhos com a intenção de produzir um "roteiro", ou seja, contar a história com uma sequência de imagens. (ROGERS, 2010).

Figura 14 - Storyboard



Fonte: BROWN, 2012

4.6.3.2. DIAGRAMAS

Segundo Rogers (2010), os diagramas darão o exemplo de gameplay, sendo representados por figuras ou artes conceituais, tendo como requisito uma legenda para que quem esteja lendo entenda o significado do que está sendo representado.

4.6.3.3. ANIMATICS

Para Rogers (2010), os *animatics* devem ser entendidos como um exemplo de visualização animada de gameplay. Neste caso o autor menciona alguns programas que podem ser utilizadas para trabalhar como o PowerPoint ou Flash. Ajudam na concepção da animação do jogo e em como ele deverá ficar quando pronto.

4.6.3.4. O GRÁFICO DE RITMO

O gráfico de ritmo coloca todo o conceito do seu jogo e a informação em uma só página mostrando a estrutura e o ritmo de cada fase do jogo. Algumas fases devem ter um ritmo forte e rápido, mas é importante se ter momentos de calma para recuperar o fôlego. Ter somente um ritmo de jogo o tempo todo pode tornar o jogo enfadonho. Segundo Rogers (2010), criar um gráfico de ritmo é muito importante ao examinar o progresso do gameplay e ele cita alguns elementos para descrever o ritmo de cada momento do jogo.

- Nome do nível.
- Ambientação.
- Horário que o jogo se passa.
- Elementos da narrativa da história para o nível.
- Progressos.
- Tempo de jogo estimado.
- Cor da ambientação dos cenários.
- Tipos de inimigos e chefões.
- Jogabilidade/Mecânicas.
- Dificuldade.
- Possíveis elementos de auxílio.
- Habilidades, equipamentos, armas.

- Quantidade de tesouros a serem encontrados.
- Quantidade de elementos bônus ao jogador.
- Trilha musical para cada nível/ambiente.

4.6.3.5. O WIKI DA EQUIPE

Para Rogers (2010) seria interessante a possibilidade e publicar o GDD por um meio eletrônico com o objetivo de manter a equipe sempre atualizada sobre as informações do projeto, assim os membros da equipe podem auxiliar na criação. Vale lembrar que a organização é importante em cada característica mencionada.

5. UNREAL ENGINE 4

5.1. O QUE É O UNREAL ENGINE 4

O Unreal Engine 4 (EPIC GAMES, 2018) é um motor de jogos que pode ser utilizado para a criação de jogos de alta qualidade, peças publicitárias e até mesmo filmes, e executá-los em plataformas como PC, Realidade virtual, Realidade aumentada e consoles.

> Se você sempre ouviu esse nome, mas não entendia do que se tratava, saiba que uma game engine (em português, *motor de jogo*) consiste em um programa de computador ou um conjunto de bibliotecas capazes de juntar e construir todos os elementos de um jogo em tempo real. (DIAS, 2017)

O Unreal Engine 4, oferece capacidade aos desenvolvedores de produzirem seus conteúdos mesmo que não tenham muita expertise com linguagens de programação. Proporciona uma série de ferramentas prontas que criam mundos virtuais e todos os elementos estruturais necessários para a criação de um jogo proporcionando que os game designers possam se concentrar mais no aspecto da ideia e história do jogo sem serem limitados por questões de linguagem de programação. O Unreal Engine proporciona a liberdade de criar mecanismos juntamente com uma vasta biblioteca de elementos e ideias dos produtores de jogos para enriquecer a criação de jogos em 2D ou 3D (EPIC GAMES, 2018).

Segundo a EPIC GAMES (2018), desenvolvedora do Unreal Engine 4, este motor de jogos proporciona uma série de ferramentas de produção para a produção de jogos, que veremos a seguir no tópico de características do Unreal 4

Para Dias (2017), os games engines permitem que até mesmo uma só pessoa consiga criar um jogo que, comparado aos anos 80, precisaria de uma equipe inteira de produção. O autor cita algumas plataformas que têm compatibilidade com esse motor de jogos:

- PC
- Dreamcast
- GameCube
- Wii
- Wii U
- Xbox
- Xbox 360
- Xbox One
- Playstation 2, 3 e 4.

5.2. COMO SURGIU O UNREAL ENGINE 4

Segundo a Pix Studios (2015) esse motor gráfico possuiu 3 versões anteriores sendo a primeira versão lançada em 1998 com o nome Unreal Engine 1, possuindo características de colisão, inteligência artificial e a renderização, com a chegada desta primeira versão surgiram jogos como X-COM: Enforcer (Hasbro Interactive) e Tatical Ops: Assault on Terror (Kamehan Studios, 2000) Unreal Tournament (Epic Games, Digital Extremes, 1999) entre outros.

A segunda versão, o Unreal Engine 2, foi lançada em 2002 com um jogo chamado America's Army (Exército dos Estados Unidos, 2002), sendo reanalisada a estrutura de renderização e modificada completamente, ganhando suporte para os consoles da época como Playstation 2, Game Cube e Xbox, além de um incremento de física para os veículos também.

Mais para a frente chegou a terceira geração do Unreal Engine, com o lançamento de Gears of War (EPIC GAMES, 2006) um jogo de console no Xbox 360, tendo os visuais reformados, renderizações com mais qualidade. A empresa Epic Games ainda firmou uma parceria com a NVIDIA, uma das empresas que trouxe qualidade de processamento visual ainda mais intensas além de outras características como o PhysX, para compor a física do jogo. Vários jogos usaram essas características.

Em 2009 a Epic Games divulgou um anúncio de que o Unreal Engine 3 a partir de então se tornaria em uma versão gratuita para desenvolvedores de conteúdo: o UDK - Unreal Development Kit. O UDK abriu grandes portas a inúmeros desenvolvedores pois possuía um kit para auxiliá-los no desenvolvimento, animação facial, a facilidade de construir cenários, inteligência artificial, programação e outras características. Atualmente a versão não existe mais por conta da nova versão que veremos a seguir.

Em 2002 foi apresentada a versão alpha do Unreal Engine 4 e ela foi o motivo da Epic Games deixar de lado o UE3. A nova versão, segundo o site da Pix Studios (2015), foi apresentada na Game Developers Conference, que tinha como objetivo apresentar o foco na oitava geração dos consoles Xbox One, Playstation 4, Wii U, além de outras plataformas também como PC. Além disso ainda possuindo a característica de pode desenvolver jogos para iOS, Mac OS X, HTML 5. Essa versão trouxe a possibilidade de os desenvolvedores trabalharem com mais facilidade em ações cinematográficas e até mesmo em Realidade virtual VR. Além disso, a Epic Games em 2014 abriu um mercado virtual online onde os desenvolvedores poderiam comprar recursos para os desenvolvimentos, como objetos 3D, sons, animações, códigos em Blueprint e C++, além de também dar a possibilidade de as pessoas venderem seus conteúdos. (WATERS, 2014)

Em 2015 a Epic Games disponibilizou a ferramenta gratuita para os produtores, mas com a condição de pagar uma taxa equivalente a 5% dos seus ganhos caso o projeto passasse a ser comercializado.

5.3. CARACTERISTICAS DO UNREAL ENGINE 4

Veremos a seguir as características do motor de jogos Unreal Engine 4 baseado nas informações que o site do Unreal Engine (EPIC GAMES, 2018) fornece para os desenvolvedores.

5.3.1. RENDERIZAÇÃO FOTOREAL EM TEMPO REAL

O Unreal Engine 4 apresenta melhoras na camada visual além da física do Unreal Engine, tendo alternativas de sombras avançadas e dinâmicas, iluminação para melhorar os aspectos gráficos, com poderosas capacidades de renderização em real time.

5.3.2. CÓDIGO FONTE C++ COMPLETO

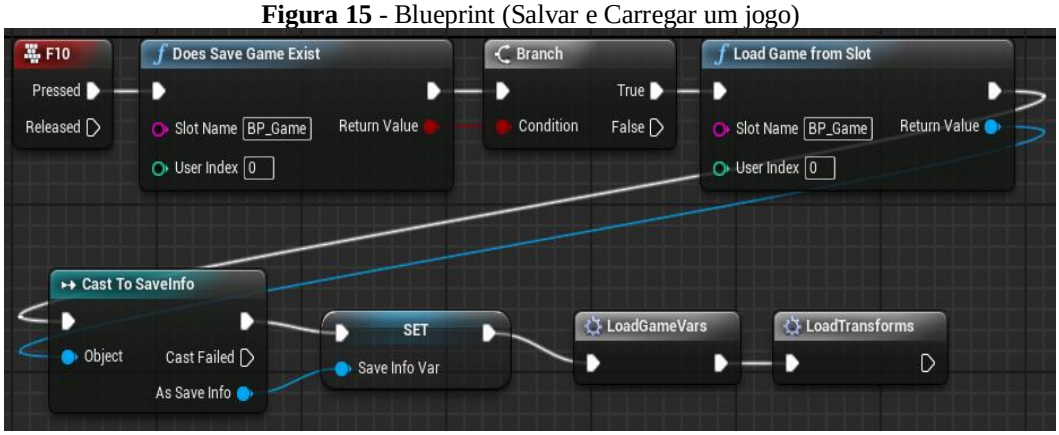
O Unreal Engine fornece a possibilidade de acessar o código C++ da ferramenta. Segundo o site do Unreal Engine (2018), essa característica é atribuída para estudos, personalização e modificação.

5.3.3. BLUEPRINTS: A PROGRAMAÇÃO SEM LINHAS DE CÓDIGO

A Epic Games (2018), afirma que é possível criar protótipos sem interagir com uma linha de código. Para isso foi desenvolvida uma implementação semelhante ao da versão anterior do Unreal Development Kit o ***Kismet***. Nessa nova versão, as Blueprints, constituem uma interface gráfica que utiliza uma linguagem visual que permite programar uma série de objetos, jogadores, controles, cálculos matemáticos, interfaces e inúmeros outros objetos. Em resumo, é como uma programação de código, porém, baseada em nós de ligação com interface visual.

Na figura 14 podemos ver a representação da programação de um Save, que grava o jogo, nível de fase e status do jogador por meio de um Blueprint.

Figura 15 - Blueprint (Salvar e Carregar um jogo)



Fonte: ROMERO, 2015

5.3.4. ESTRUTURA MULTIJOGADOR

O Unreal Engine melhorou a questão da ferramenta multijogador. A Epic Games afirma que os testes foram realizados em várias plataformas com o objetivo de deixar o aspecto multijogador com mais qualidade e eficiência, fornecendo uma estrutura de servidor e cliente.

5.3.5. SISTEMA DE PARTICULAS E VFX

O sistema de partículas consiste em trabalhar efeitos de iluminação em uma variedade de módulos, como um brilho de uma espada ao atacar, as chamas de uma fogueira, e em uma variedade efeitos em cenas de movimentos, também introduzido o aspecto de “polimento” trabalhado por meio de VFX, além de seus efeitos visuais muito utilizados em cinemas por exemplo.

5.3.6. EFEITO PÓS-PROCESSO DE QUALIDADE DE FILME

Esta ferramenta auxilia na qualidade das cenas com efeitos semelhantes aos efeitos visuais de cinema, contando com a oclusão de ambiente, reflexos de lente, *anti-aliasing* e outras. Todas essas características contribuem para deixar os jogos com um aspecto extremamente realista.

5.3.7. EDITOR DE MATERIAL

O editor de materiais funciona por meio da aplicação de camadas de texturas com efeitos programáveis dentro de uma Blueprint, podendo, por exemplo, criar um sombreamento físico, aspectos rugosos, controlando a textura dos objetos e personagens, água e, também, com a possibilidade de animar estes materiais e obter sensação de movimento por meio das Blueprints, por exemplo para ter movimento da água ou aspecto de material molhado.

5.3.8. EXTENSIVO CONJUNTO DE ANIMAÇÃO

Com esta ferramenta é possível modificar os personagens personalizando a malha esquelética e também mudando a “máquina de estados”, ou seja, uma programação de como o personagem irá reagir com determinada função de programação específica. Neste caso, a “máquina de estado” irá orientar o personagem ou objeto a agir com certa animação definida diante de determinada ação. Para isso os Blueprints irão trabalhar para dinamizar as animações criadas e programadas.

5.3.9. SEQUENCIADOR: CINEMATOGRAFIA DE ULTIMA GERAÇÃO

Essa ferramenta dá a possibilidade de os desenvolvedores criarem sequências de cenas profissionais. É possível realizar edições dentro do programa sem usar outros aplicativos para isto, podendo-se modificar o ambiente como a área de iluminação, a movimentação da câmera. O resultado é a produção de "*Cut scenes*" com qualidade comparável à de filmes de Hollywood.

5.3.10. EDITOR COMPLETO EM VR

Com essa função os desenvolvedores têm a capacidade de criar jogos com realidade virtual. Através de um controle de movimento e sensores, é possível manipular objetos virtuais como se fossem na vida real, dando a possibilidade de criar ambientes virtuais para a interação neste tipo de jogo.

5.3.11. CONSTRUÇÃO VR, AR e XR

Segundo o site do Unreal Engine Features (EPIC GAMES, 2018), a Epic Games se preocupou em fornecer uma solução com qualidade alta para criar a realidade virtual (VR) e também a realidade aumentada (AR), tendo suas características bem amplas em renderizações sem que o computador perca o processamento e a taxa de frames, ou seja, sem interferir no desempenho da máquina.

5.3.12. TERRENO E FOLHAGEM

Essa ferramenta fornece a opção de criar grandes escalas de terrenos com o sistema *Landscape*, com o qual, graças ao sistema de LOD (Nível de detalhamento) e o uso da memória, é possível criar mapas enormes e detalhados. Esse sistema funciona como uma espécie de pincel, você seleciona o objeto e configura quantos objetos você deseja colocar em determinado ponto até outro ponto. Isto pode ser feito com gramas, pedras, decorações e etc.

5.3.13. INTELIGENCIA ARTIFICIAL AVANÇADA

Essa função tem sido muito utilizada de fato em vários jogos, a capacidade desta função no Unreal Engine 4 é muito ampla em relação a consciência do ambiente ao redor do jogador, tendo uma dinâmica com movimentos inteligentes e uma estrutura atualizada em tempo real.

5.3.14. MOTOR DE AÚDIO

Uma inovação em efeitos dinâmicos com a utilização dos Processadores Digitais de Sinal (DSP), é a possibilidade de personalizar os áudios físicos. A Epic Games fez uma parceria com a Valve por meio da qual foram criados sons com qualidade para integrar o que eles mencionam de plugin Steam Audio no motor de jogos UE4 solucionando o problema de grande parte das plataformas com relação ao áudio. Essa versão foi inicializada a partir da versão UE 4.16.

5.3.15. NAVEGADOR DE CONTEÚDO

Essa função dá a capacidade de os produtores importar e exportar materiais, texturas, objetos 3D, dando a possibilidade de organização com criação de pastas, além de filtrar e pesquisar por itens guardados em pastas. Também é possível usar um sistema de clica e arrasta para mover objetos para o cenário tornando muito mais fácil a construção do ambiente. A empresa afirma que é possível realizar compartilhamento de conteúdo para facilitar a produção dos jogos.

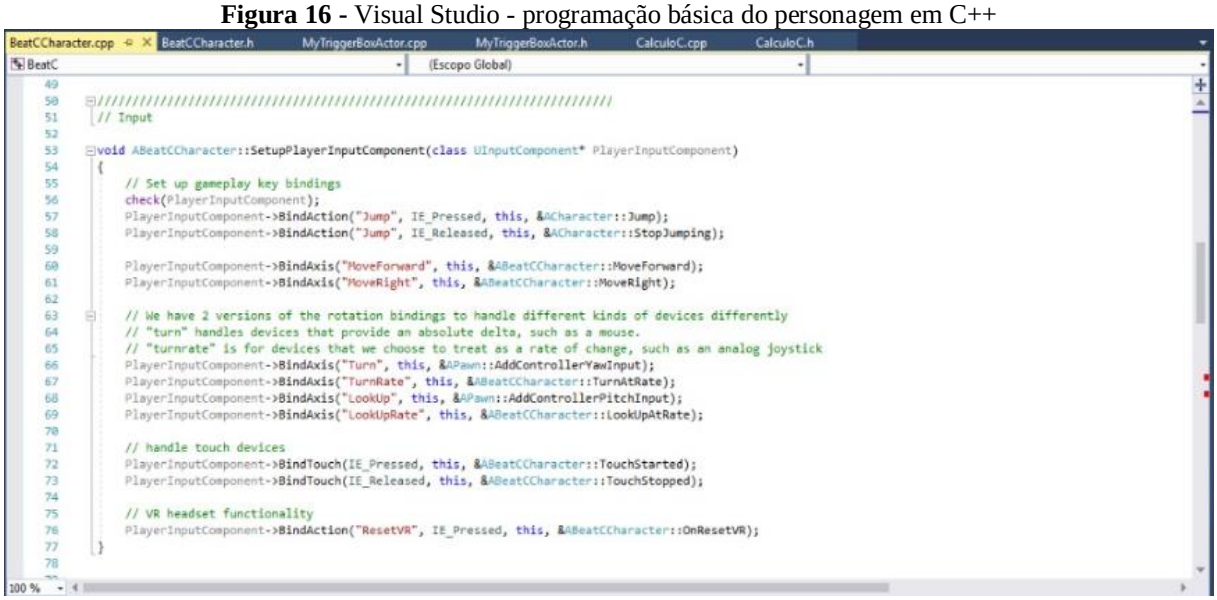
5.3.16. ECOSSISTEMA DO MARKETPLACE

O Unreal Engine possui um mercado online para agilizar e melhorar o processo de desenvolvimento dos jogos fornecendo plugins, personagens, sons, objetos e *blueprints*. Para um iniciante é possível comprar esses recursos ao invés de criá-los, que custaria mais. O Marketplace dá a possibilidade de vender seus produtos para outros desenvolvedores, fornecendo uma possibilidade de trabalho para quem produz gráficos e faz programação para jogos de comercializar itens para jogos.

5.4. O UNREAL ENGINE 4 E A PROGRAMAÇÃO C++

Segundo a Epic Games (2018), na documentação sobre o Unreal Engine, este útimo pode ser programado com C++ para criar objetos no jogo e meios de jogabilidades. Neste sentido, o C++ é voltado para o "gameplay" de modo configurado pelos próprios desenvolvedores do projeto. Para isso é necessária a utilização de IDEs (ambiente de desenvolvimento integrado) que usem C++ para a programação. A IDE mais utilizada, neste caso, é o Visual Studio, apresentando agilidade no processo de programação e compilação. Há também o Xcode, porém, este é menos utilizado. A empresa ressalta que o Unreal Engine foi implementada no Visual Studio, podendo ser adquirida até mesmo pela instalação do Visual Studio como um complemento.

Figura 16 - Visual Studio - programação básica do personagem em C++



Fonte: Próprio autor

Atualmente, o Unreal Engine mais atual trabalha com a versão mais atual da IDE do Visual Studio 2017.

5.4.1. COMO SURGIU O C++?

Segundo Alisson (2012) a linguagem C teve sua invenção em uma máquina chamada PDP-11 com sistema operacional "Unix". Essa é mesma máquina que rodaria "*Tennis for Two*" mais para frente, em meados da década de 70. Na época a linguagem tinha sido desenvolvida por Dennis Ritchie a partir de uma linguagem já existente que era chamada de linguagem B

criada por Ken Thomson. Podemos dizer, então, que a linguagem C acabou sendo a linguagem mais atualizada na época e é um avanço da linguagem B.

Essa linguagem se tornou uma das linguagens mais utilizadas para a produção de softwares. Na década de 80 surgiram ideias de integrar novas funcionalidades à linguagem, o que viria a ser conhecido como “C com classes”. A ideia pertencia a Bjarne Stroustrup e, por conta dessas ideias e a implementação destas funcionalidades, a linguagem C passou a ser conhecida como linguagem C++ e passou a suportar orientação a objetos.

Estrutura de um programa simples em C

```
/* Início do Programa, tem que declarar as Bibliotecas*/
#include<iostream.h>
int a global /* Declaração de variáveis Globais*/
/*Declaração de funções e procedimentos caso se tiver*/
int main () /*Declaração da função principal, é sempre necessário*/
{
float num1; /*variáveis locais*/
/*Comando*/
}
<iostream.h>
```

Para Casavella (2004) a linguagem C pode facilmente ter compatibilidade e um objetivo geral em aspectos de execução, comportamento, portabilidade. Por conta de várias IDEs desenvolvidas surgiu a necessidade de padronizar por conta de algumas incompatibilidades e erros que algumas apresentavam, essa padronização ocorreu em 1989 pelo instituto ANSI – Instituto Nacional Americano de Padronização que tem por objetivo a facilitação em padronizar e facilitar o trabalho dos membros.

5.4.2. O QUE SÃO IDEs E QUAIS UTILIZAM C++

Segundo Novaes (2004) IDEs são ambientes de desenvolvimento integrados. Trata-se de um software específico para agilizar e deixar mais fácil a programação para os desenvolvedores, contendo as funções que uma linguagem possui. A partir disto é possível criar outros softwares para computador, celulares e etc. Ainda segundo o autor, IDEs apresentam a vantagem de otimizar o tempo de compilação dos códigos programados, além de mostrarem erros, assim, facilitando a identificação de possíveis "bugs". Porém, por outro lado, a IDE não faz tudo. O programador, de fato, precisa ter um conhecimento mínimo de programação e noção lógica de como a IDE se comporta. Algumas IDEs que auxiliam na programação em linguagem C++ são o DevC++; o Visual Studio e o C Builder.

Pires (2017) acrescenta algumas características sobre esses ambientes integrados. Algumas IDEs têm compatibilidade em realizar projeções de interfaces gráficas visuais. Ele ressalta também que as IDEs também podem apresentar desvantagens, pois o autor acredita que pode deixar o programador, de certa forma, preguiçoso. Este recurso automatiza alguns elementos que os programadores precisam dominar e não o fazem porque confiam nas IDEs para fazer isso.

5.4.3. O QUE É O VISUAL STUDIO?

Segundo o site da Impacta (2018), o Visual Studio é uma IDE para a agilização de desenvolvimento do código com características como autocompletar na construção da programação, sendo também apresentados possíveis erros de código na hora da compilação para que o desenvolvedor corrija de forma antecipada. Em alguns casos até sugere soluções para ajudar o desenvolvedor a solucionar o problema mais rápido.

Com o Visual Studio é possível criar softwares tanto para sistema operacional Windows como para mobile. Ainda segundo o site, é possível que o banco de dados tenha conexão com a IDE. Nesse caso as instruções do banco não precisam ser reescritas. O Visual Studio dá suporte para as linguagens C, C++, C#, Visual Basic, J# e J++.

5.5. UNREAL ENGINE 4 E A PROGRAMAÇÃO EM BLUEPRINT

Segundo Valcasara (2015, p. 2), os *Blueprints* têm semelhança com o Kismet do Unreal Engine 3. A diferença é que eles são uma evolução do que era usado na versão anterior para a

programação de jogos, adicionando vários outros conceitos inovadores. O autor afirma que, com essa evolução, é possível agora criar com mais facilidade regras para os jogos, variáveis, customização do personagem, dinamizar a câmera, entre outras novas características. De acordo com o autor, os Blueprints não são apenas um Kismet comparado à versão anterior, mas agora é composto vários tipos de Blueprints incluindo, *Level Blueprints*, Classe de *Blueprint*, *Blueprint interface*.

De acordo com a Epic Games (2018) em sua documentação sobre o Unreal Engine 4, os *Blueprints* têm um conceito de programação utilizando a lógica como qualquer outra programação. A diferença é que se trata de uma interface com "nós" de variáveis para criar a jogabilidade e regras que são definidas pelo desenvolvedor, utilizando-se até mesmo a orientação a objetos.

5.5.1. COMO AS BLUEPRINTS FUNCIONAM?

Na documentação do Unreal Engine sobre o funcionamento das *Blueprints* (EPIC GAMES, 2018) vemos que, com elas, podemos criar o conteúdo do game como uma programação em qualquer outra linguagem mas, de um jeito mais simples apenas por ligação de "nós" com o auxílio de uma interface simples e intuitiva. É possível, a partir disto, criar objetos, regras, funções, *huds(heads up displays* – ou elementos gráficos da interface de jogo) e todos os componentes de um jogo sendo, portanto, uma ferramenta que permite a produção de jogos sem utilizar uma única linha de código de programação.

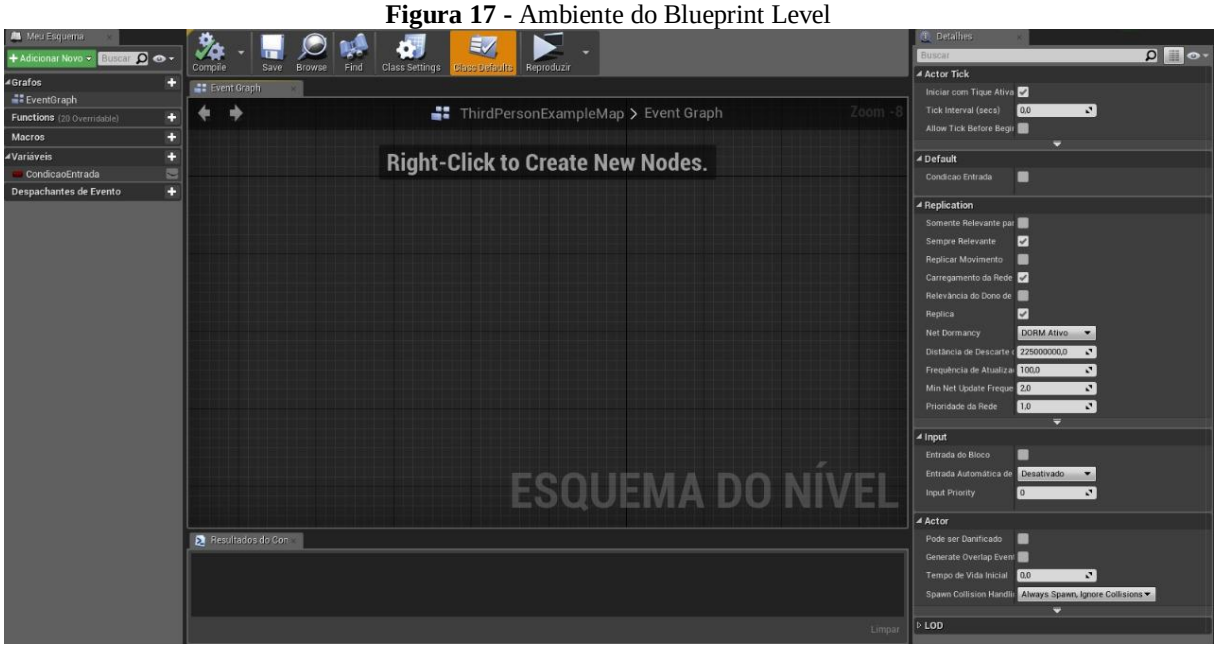
Por meio da Blueprint é possível criar variáveis para condições, somas de constantes e programação de objetos, personagens e animações e outros aspectos que compõem a jogabilidade do jogo em geral (SEWELL, 2015, p. 1).

5.5.2. LEVEL BLUEPRINT

De acordo com Valasara (2015, p. 3), o *Level Blueprint* está relacionado a programação do ambiente do jogo, sendo que o *Level Blueprint* só pode ser editado apenas na ambientação inicial da fase. Caso haja uma fase "dois" aquela mesma programação terá que ser reconstruída do zero para compor a fase atual. Sendo assim, a cada nível toda a programação será diferente para a dinamização do jogo.

Essa característica específica do “Level Blueprint” nos possibilita mexer com eventos, eventos cinematográficos, sequencias de animações de objetos ou *meshes* (Objetos 3D, ou Personagens do cenário) e outras operações de ambientação.

Figura 17 - Ambiente do Blueprint Level



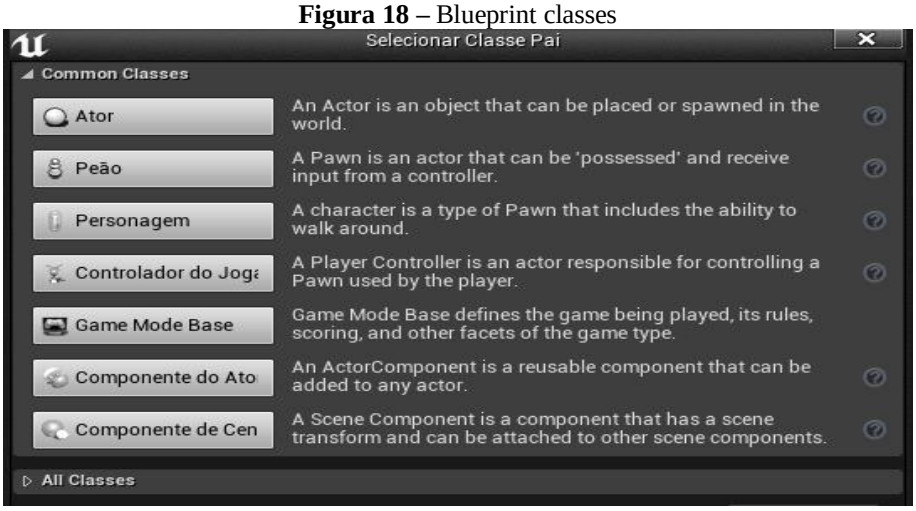
Fonte: Próprio Autor

5.5.3. BLUEPRINT CLASSES

Os *Blueprint classes* permitem a criação de objetos como um ator, peão, personagem, controle do jogador e o modo de jogo. Assim, a *Blueprint* cria as classes básicas a partir das quais serão criadas as instâncias de objetos do jogo, cada um com a sua função (VALCASARA, 2015, p. 4).

De acordo com Valcasara, existem várias classes para a criação e funções diferentes ao criar essas *Blueprint classes*. Pode-se criar atores que podem ser colocados no ambiente como objetos. Veremos, a seguir algumas, funções das classes existentes nos Blueprints do Unreal Engine 4.

Figura 18 – Blueprint classes



Fonte: Próprio Autor

Atores: A classe atores gera objetos que serão colocados no ambiente para compor o nível da fase e do cenário.

Peões: Peões que são colocados no ambiente como objetos que podem ser "possuídos", neste caso, um jogador pode "possuir" um carro por exemplo e mais uma vez a programação Blueprint será definida pelo desenvolvedor. Este peão "carro" de exemplo, necessita também de uma entrada de controle para controlar o peão possuído.

Personagem: A classe personagem nada mais é do que a classe jogadora que, por meio da Blueprint, inclui as regras que podem envolver capacidades de andar, nadar, pular, atirar e etc.

Controle de jogador: Esta classe cria um ator responsável pelo controle de um peão que será controlado pelo jogador.

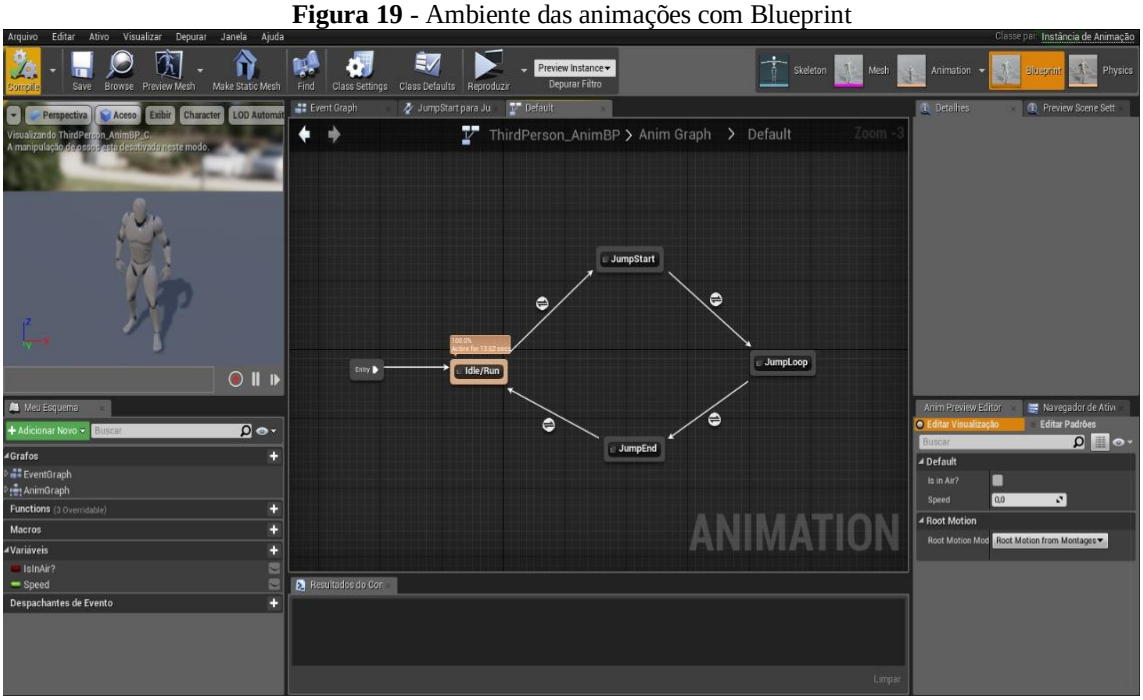
Modo de jogo: Por fim existe a classe modo de jogo que define as regras e todas as características do jogo.

Essas são as principais classes de Blueprint tanto usado em programação com a própria Blueprint, como também com o C++ para construção do jogo.

5.5.4. ANIMAÇÕES COM BLUEPRINT

O componente para animações do Blueprint é composto por Eventos de Gráficos e Gráficos de Animações. Por meio da Blueprint são definidas as regras de como será o comportamento das animações diante de alguma ação praticada, controlando os ossos da malha esquelética (personagem 3D) formulando uma pose para cada quadro de animação (SATHEESH, 2016, p. 137).

Figura 19 - Ambiente das animações com Blueprint

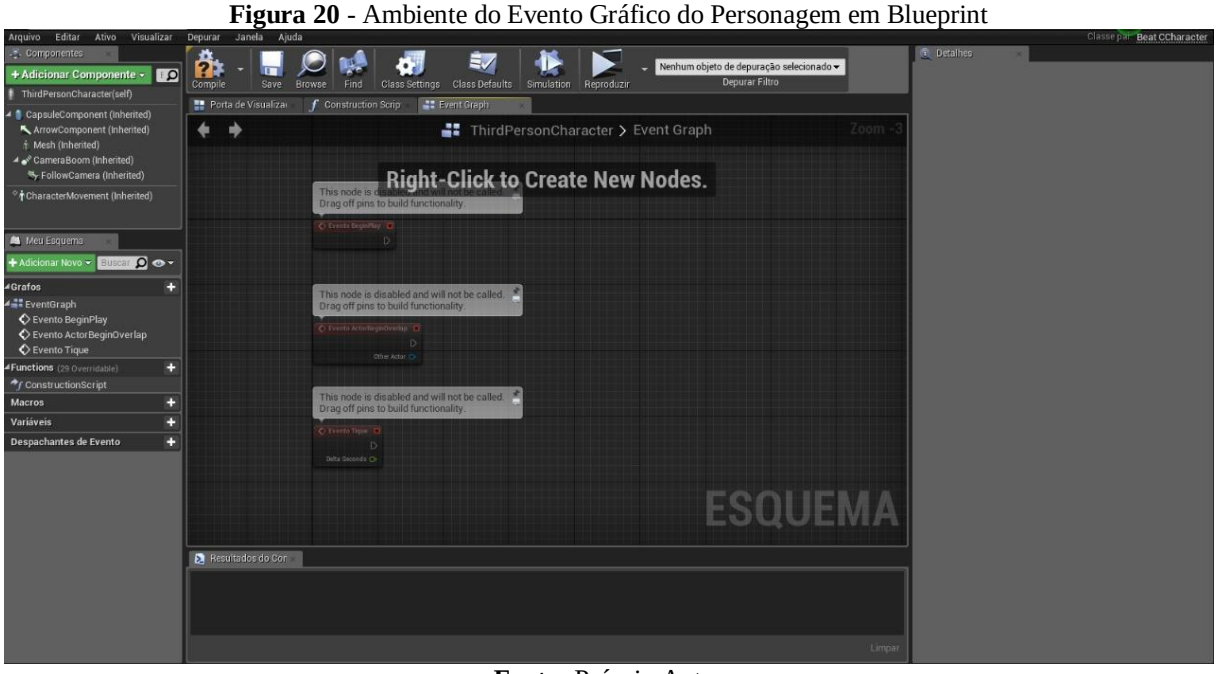


Fonte: Próprio Autor

5.5.5. GRÁFICO DE EVENTOS

De acordo com Satheesh (2016), o gráfico de eventos (Event Graph) de animação irá dirigir os eventos que ativam as animações/programações por meio de um encadeamento de "nós" que é orientado por meio das regras ou valores definidos dando "vida" ao objeto.

Figura 20 - Ambiente do Evento Gráfico do Personagem em Blueprint



Fonte: Próprio Autor

5.6. COMPARAÇÃO ENTRE A CRIAÇÃO DE JOGOS COM BLUEPRINT E COM LINGUAGEM C++ EM UNREAL 4

A respeito da programação em C++ o próprio Unreal Engine foi criado nesta linguagem. É possível programar jogos no Unreal Engine diretamente em C++. Trata-se de uma linguagem poderosa, com uma velocidade consideravel e possui um bom desempenho além de ter recursos voltados para a orientação a objetos (SHERIF, 2015, p. 4).

No caso do Blueprint, não é necessário ter tanto conhecimento de programação, mas, assim como em C++ é necessário entender a lógica de programação, pois ambos funcionam do mesmo modo com base nos "nós" (*nodes*) de interface, fazendo ligações de maneira mais simples e fácil.

Apesar de serem formas diferentes de programar, as duas maneiras funcionam de modo semelhante. Porém, quando se deseja maior velocidade e desempenho em cálculos matemáticos e processamento, a programação dos jogos com linguagem em C++ no Unreal 4 se destaca por apresentar desempenho e capacidade superior do que a programação de jogo por Blueprint.

5.6.1. VANTAGENS E DESVANTAGENS ENTRE BLUEPRINT E C++

5.6.1.1. DESVANTAGENS

De acordo com a documentação do Unreal Engine 4 (EPIC GAMES, 2018), programar jogos por meio de Blueprints é recomendado para equipes que não possuem conhecimento em linguagens de programação. Isso pode acarretar em problemas de desempenho do jogo no caso deste ser produzido apenas com Blueprints. A empresa explica que esse problema é devido ao modo como a Blueprint é executada, sendo processada por uma máquina virtual que chama funções da linguagem C++, ou seja, toda a programação produzida em Blueprint na hora da execução de um jogo é traduzida para linguagem C++, e neste processo de tradução há um custo de desempenho.

5.6.1.2. VANTAGENS

A vantagem em comum destas duas linguagens é que é possível fazer conexão com funções escritas em C++ e fazer uma conexão com Blueprint, ou seja, duas linguagens trabalhando juntas para diminuir este problema.

6. MÉTODOS

Para entendermos o processo de comparação entre estas duas formas de programar jogos no Unreal Engine 4, criamos um ambiente virtual com dois atores, um sendo C++ e o outro Blueprint. Assim, verificamos se, de fato, a performance de jogos programados com C++ de fato é melhor do que a performance de jogos com Blueprints. Ao final da construção dos jogos, aferimos a velocidade de resposta de cada um. O objetivo foi comparar estas duas formas de programar jogos e identificar o tempo de resposta que uma tem em relação à outra.

6.1. FERRAMENTAS PARA A REALIZAÇÃO DO PROTÓTIPO

Foi necessário um computador com os requisitos necessários para a utilização dos softwares respeitando os requisitos mínimos informados no site dos fabricantes. Neste caso, utilizamos o Unreal Engine 4, Mixamo e Visual Studio.

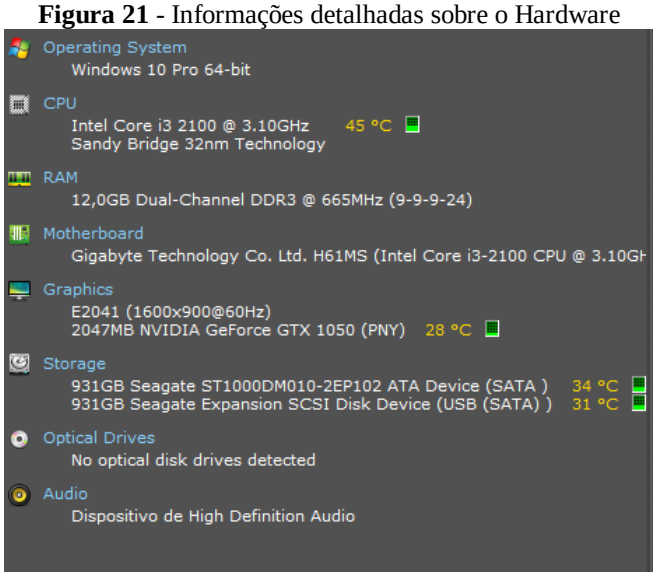
6.2. HARDWARE PARA A CRIAÇÃO DO AMBIENTE VIRTUAL

Para realizar a comparação houve a necessidade de equipamentos com requisitos necessários para a criação do ambiente onde faremos as comparações. Para isto, utilizamos uma GPU (Graphic Processing Unit – unidade de processamento gráfico) também chamada de placa de vídeo. Esse hardware especifico possui uma unidade de processamento muito alto comparado aos CPUs.

Segundo Valle (2015), GPUs têm um nível de processamento muito rápido quando se refere à taxa de quadros, conhecida também como FPS (Frames-Per-Second) além de possuir memória da placa integrada que agilizará o processamento do jogo. Com isso, estivemos retirando a possibilidade de interferência de desempenho.

Para mais detalhes sobre o Hardware que usamos, utilizamos o software Speccy (2018) para identificar as informações dos componentes, assim, analisamos a recomendação mínima para a utilização do motor de jogos Unreal Engine 4.

Figura 21 - Informações detalhadas sobre o Hardware



Fonte: Speccy

A GPU que conduziu a realização dos testes foi uma NVDIA GTX 1050 possuindo um clock para processamento de 1354MHz e um processamento de sombreamento do ambiente com 3504MHz. Essa GPU possui também um armazenamento de até 2GB. O Hardware também contou com uma memória de até 12GB para o trabalho em múltiplas tarefas para que não houvesse problemas de desempenho durante o trabalho do protótipo.

Nota-se que com essas informações, os requisitos do hardware atingem a recomendação da utilização do software Unreal Engine 4.

6.2.1. MICROSOFT VISUAL CODE

O Visual Studio foi utilizado para a programação da função que realizamos para os testes do protótipo. A IDE esteve encarregada de realizar funções de looping em C++ que podem ser chamadas pela Blueprint. A estrutura foi planejada com a mesma estrutura lógica que realizamos também na programação em Blueprint e também por ela foi realizada a estrutura de movimentação e câmera do personagem de testes.

6.2.2. ADOBE PHOTOSHOP

Segundo a empresa ADOBE (2018), o Photoshop é um software capaz de criar imagens e design gráficos de acordo com a sua imaginação, sendo assim com ele podemos realizar pinturas, edições e criar ilustrações de imagens.

Usamos então o Photoshop para a projeção conceitual do primeiro nível do protótipo que será apresentado, assim, tendo como base um “rascunho” que será produzido no Unreal mais à frente.

7. PROTOTIPAÇÃO DO PROJETO

7.1. DEFINIÇÕES PARA O PROTÓTIPO

Para realizar os testes iremos definir como será o projeto baseado no estilo folha-única, este estilo foi o mais adequado pelo motivo de que iremos descrever apenas o que se pretende criar em um projeto pequeno, sendo assim foi definido informações como: título do jogo, gênero, sistemas, e outras características apresentadas neste estilo.

7.2. FOLHA ÚNICA (GDD) – JOGO PROTÓTIPO – C++/BLUEPRINT

7.2.1. CONCEITO DO JOGO

O jogo Protótipo - C++/Blueprint, foi planejado com o intuito de apresentar a comparação de velocidade na linguagem C++ e Blueprint. Podemos esperar que, com isso, seja apresentado um entendimento claro sobre a utilização em pequenos ou grandes projetos com base no resultado da dinâmica de comparação.

7.2.2. MISSÃO

O Protótipo – C++/BLUEPRINT, existe com a finalidade de realizar análise comparativa de programação para jogos utilizando o Unreal Engine 4.

7.2.3. GÊNERO DO JOGO PROTÓTIPO – C++/BLUEPRINT

O jogo em si não possui um gênero, pois trata-se de um experimento comparativo, baseado na visão em terceira pessoa, no qual o jogador tem a visão por trás do personagem.

7.2.4. PÚBLICO-ALVO

O público alvo deste protótipo é destinado para pessoas que têm interesse em programações e jogos, não tendo restrições a idade, sexo ou grau de escolaridade.

7.2.5. CONTROLE DE ESQUEMA

O controle sobre o personagem é conduzido a partir de controles como Teclado/Mouse, Como se trata de um teste não haverão controles de navegabilidade, apenas um acionamento através da colisão do personagem com os atores de início do teste de performance.

7.2.6. PERSONAGEM

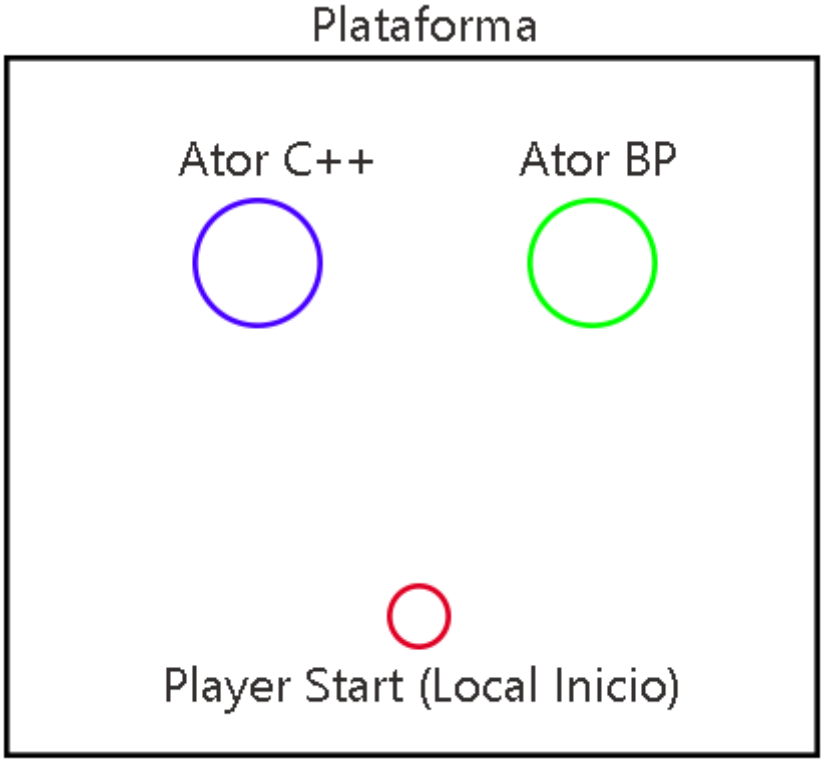
O personagem que utilizamos é um “template” padrão que a ferramenta Unreal fornece inicialmente e teve por definição somente um propósito no decorrer do jogo, colidir com um outro objeto repetidas vezes para comparar a performance da programação em C++ comparada com a programação por Blueprint. Para isto o personagem contou com a capacidade de se mover e colidir com um objeto repetidas vezes.

7.2.7. PROJEÇÃO DO AMBIENTE TESTE

Planejamos uma plataforma entre o jogador e os atores, para simplificar a projeção, através disto realizamos os testes exibindo o resultado na tela de visualização do jogador. No cenário contamos com dois atores para realizamos a interação de testes bastando que o jogador se movimentasse na plataforma até colidir com os atores.

Foi usado como referência um vídeo publicado na plataforma *Youtube* pelo canal “Alt alt” intitulado “Blueprint vs C++ Perfomance” que apresentou formas de comparação através da programação em Blueprint e C++ com a mesma estrutura de código de modos diferentes.

Figura 22 – Ambiente de teste



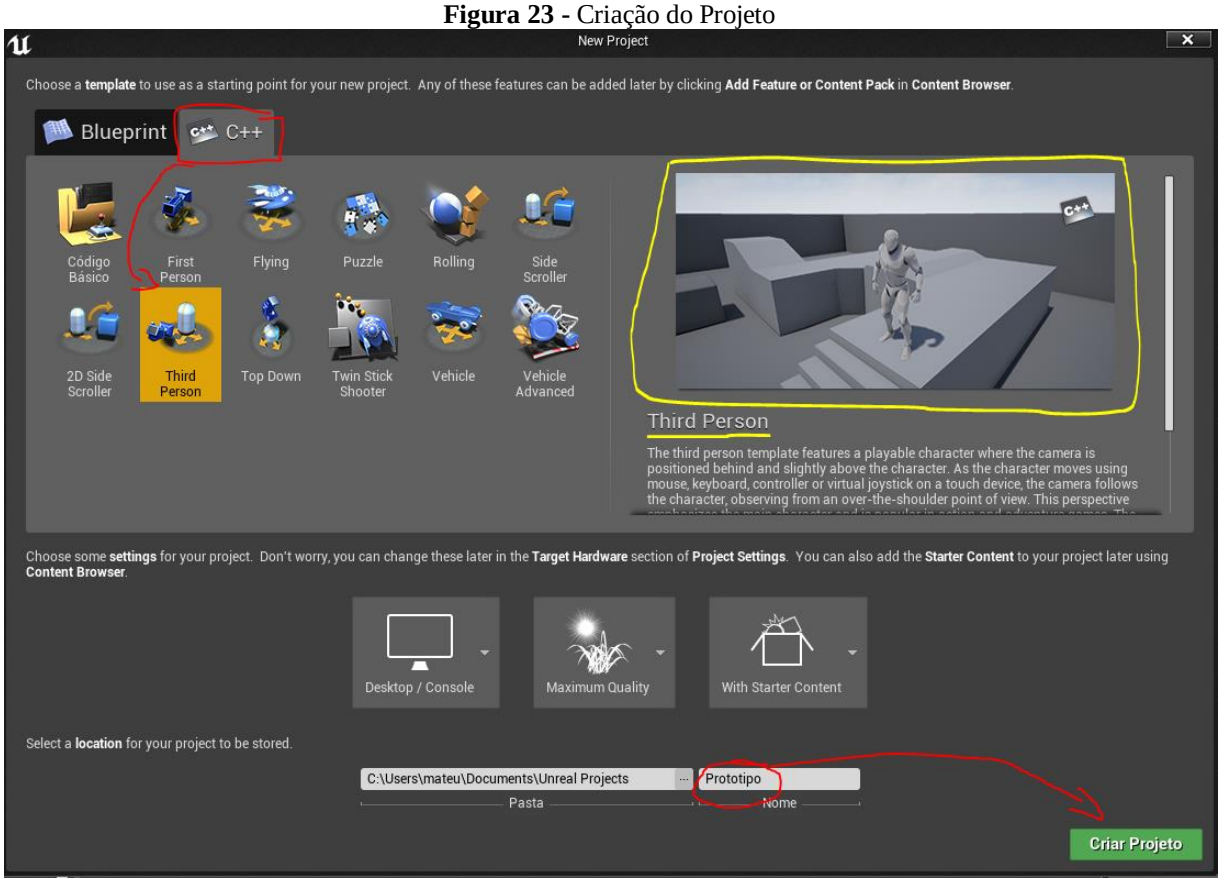
Fonte: Próprio autor

8. RESULTADOS

8.1. CRIAÇÃO DO PROJETO

Com as ferramentas instaladas na máquina, isto é, o motor de jogos Unreal Engine 4 e o ambiente de desenvolvimento Microsoft Visual Studio 2017 (Visual Studio Code), o projeto foi criado. Para isto, inicializamos o Unreal 4 e selecionaremos o modelo em C++ com as configurações voltadas a um jogo de terceira pessoa que posiciona a câmera do jogador atrás do personagem. O segundo passo foi nomear o projeto para que fosse salvo na pasta de documentos onde seria armazenado todo o conteúdo criado com o modelo de base. Depois, clicamos em “criar projeto” e, assim, o ambiente virtual foi automaticamente criado, como podemos ver na figura 24.

Figura 23 - Criação do Projeto

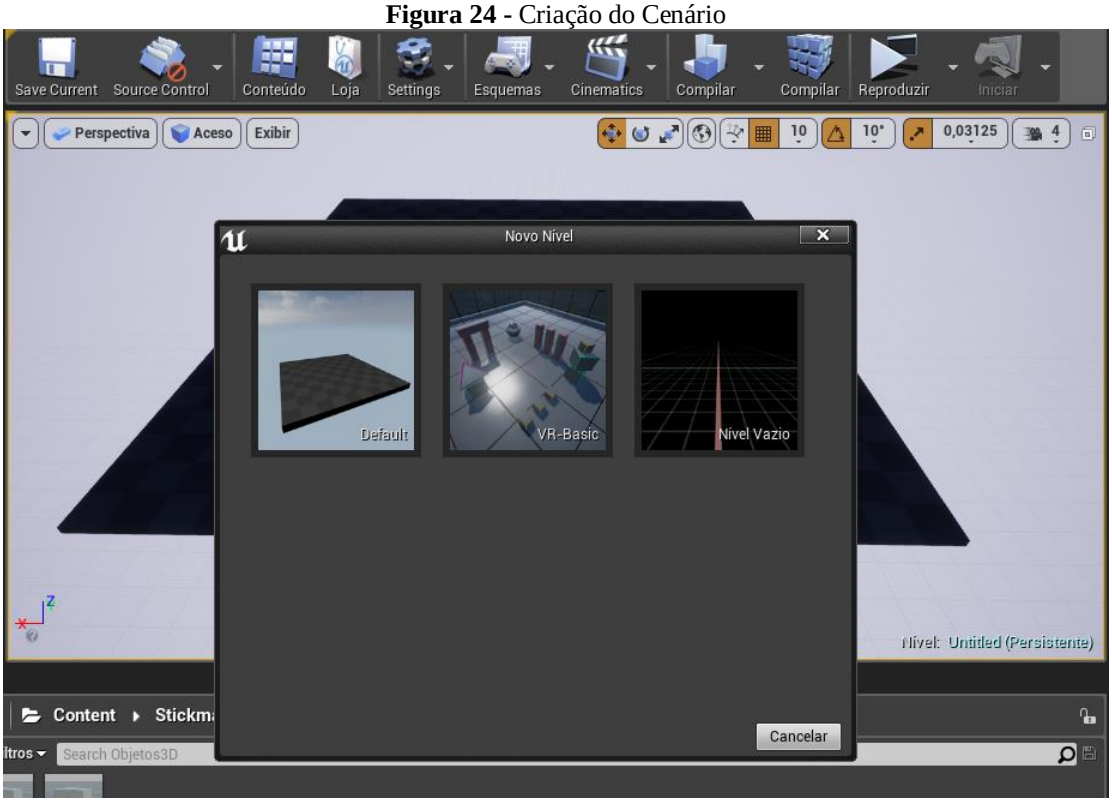


Fonte: Próprio autor

8.2. AMBIENTAÇÃO/NÍVEIS

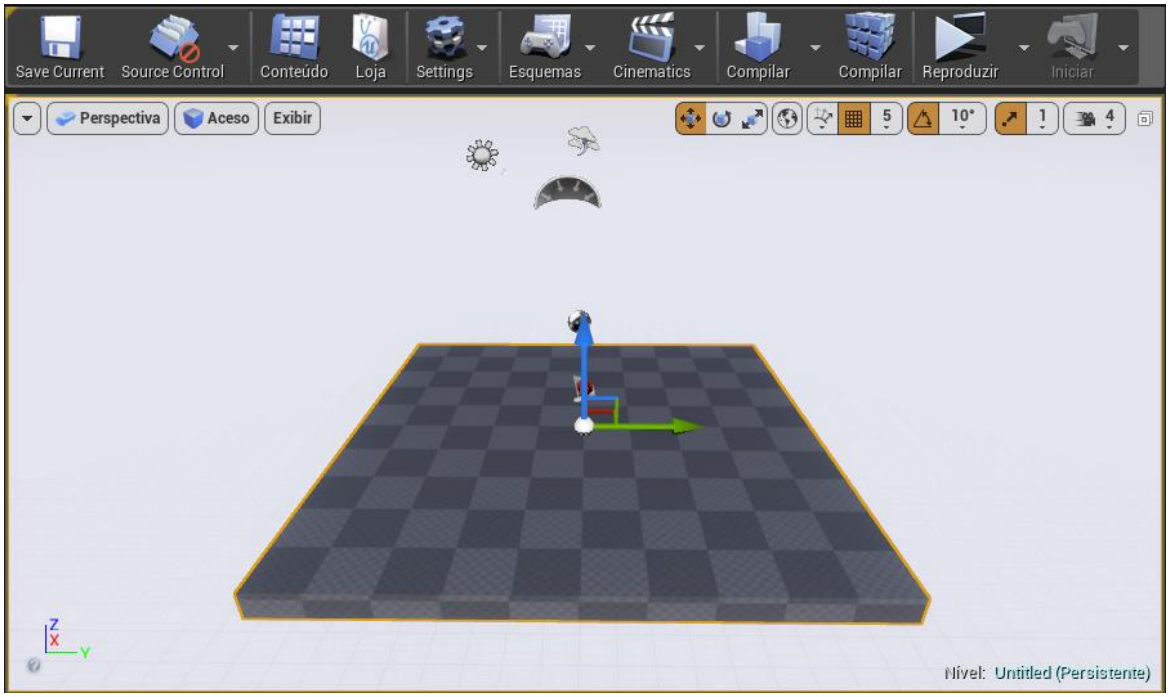
O primeiro passo foi criar um cenário. Para isto, selecionamos a opção de menu ”novo nível” para a criação de um cenário. Neste cenário teremos um elemento como a esfera do céu em volta de uma plataforma que serve como chão (ver figura 25).

Figura 24 - Criação do Cenário



Fonte: Próprio autor

Figura 25 - Ambiente do protótipo

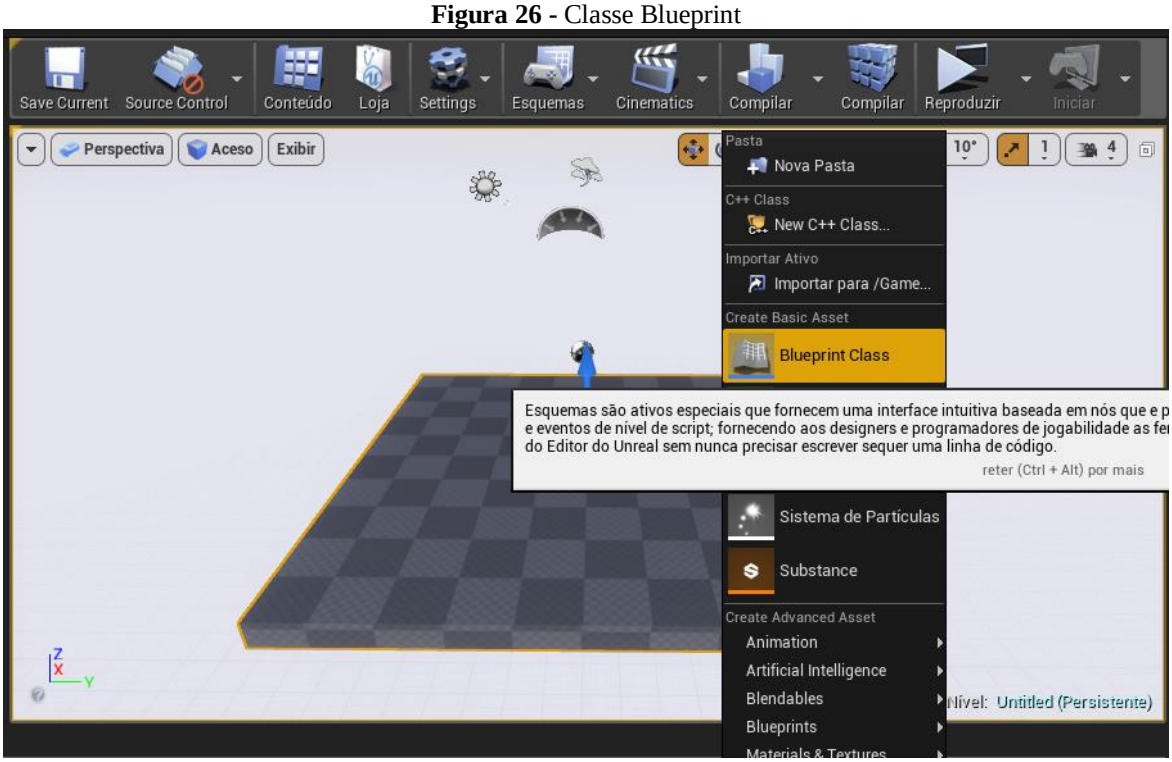


Fonte: Próprio autor

9. ATORES

Para criar um ator para o teste, clicarmos com o botão direito do mouse na área do Content Browser/Navegador de Conteúdo, abrindo uma janela com diversas opções, dentre elas a opção Blueprint Class.

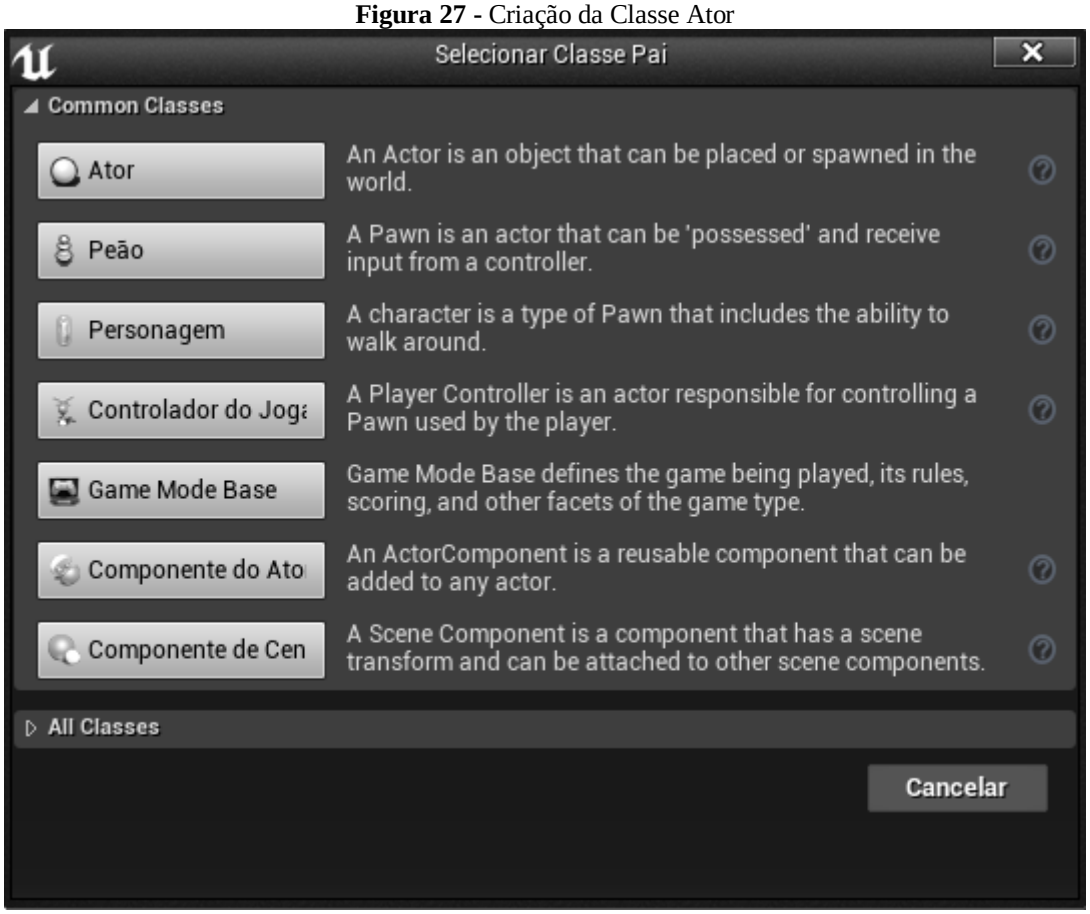
Figura 26 - Classe Blueprint



Fonte: Próprio Autor

Após selecionarmos esta opção uma nova janela foi apresentada com diversas opções de criação de classes. É nesta janela que criamos as classes Blueprints ou C++. Dependendo da opção selecionada o Engine se encarrega de modelar a base da classe de acordo com a sua definição. Escolheremos a opção de classe “Ator”, pois será um objeto que poderá ser utilizado em qualquer lugar do mundo virtual (ver figura 27).

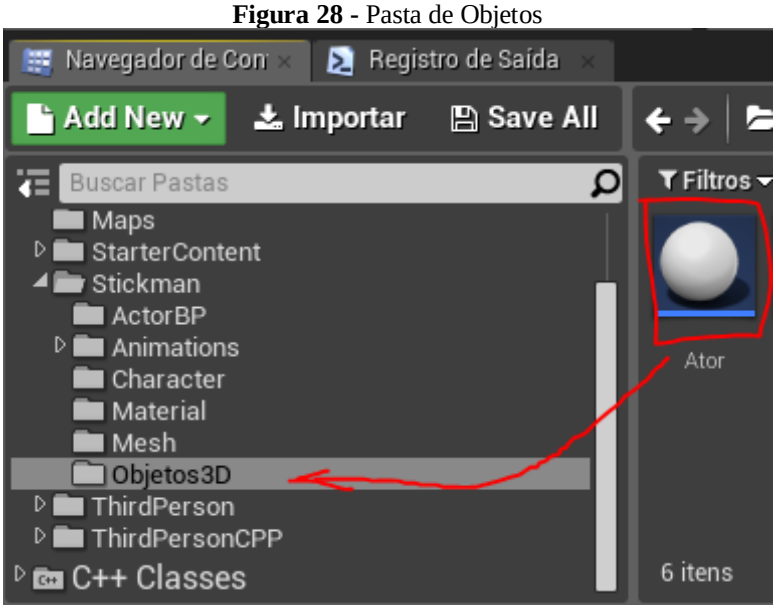
Figura 27 - Criação da Classe Ator



Fonte: Próprio Autor

Assim que criamos nossa classe, salvamos o objeto em uma pasta pré-definida, para salvar os objetos de execução de teste.

Figura 28 - Pasta de Objetos



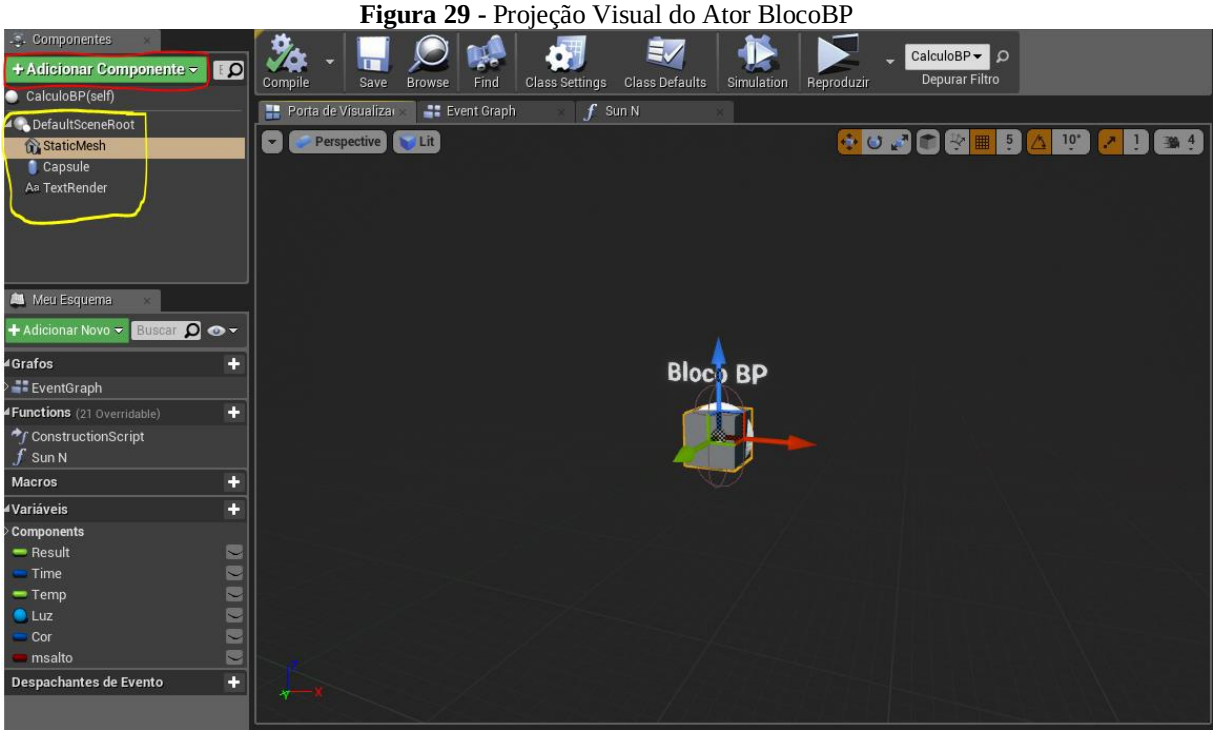
Fonte: Próprio autor

9.1. CRIAÇÃO BASE DOS BLOCOS

Criamos o primeiro ator e o nomeamos como “Bloco BP”. Depois, copiamos este ator e renomeamos a cópia como “Bloco C++”. O nome de “BlocoBP” se referente à funcionalidade e programação da linguagem em Blueprint e o outro para a programação em C++. Neste caso era importante criar dois atores com aparência idêntica, porém, cada um com a sua forma de programar para que a comparação pudesse ser realizada.

Para fazer as configurações clicamos duas vezes sobre o Ator/Objeto criado e armazenado na sua pasta personalizada, tendo o acesso pelo Navegador de Conteúdo. Ao abrir a projeção do Objeto, adicionamos três componentes ao Ator definidos como Malha Estática (“1M_Cube”), cápsula de colisão (Capsule) e um TextRender colocado acima do bloco com as descrições “Bloco BP” e “Bloco C++”.

Figura 29 - Projeção Visual do Ator BlocoBP



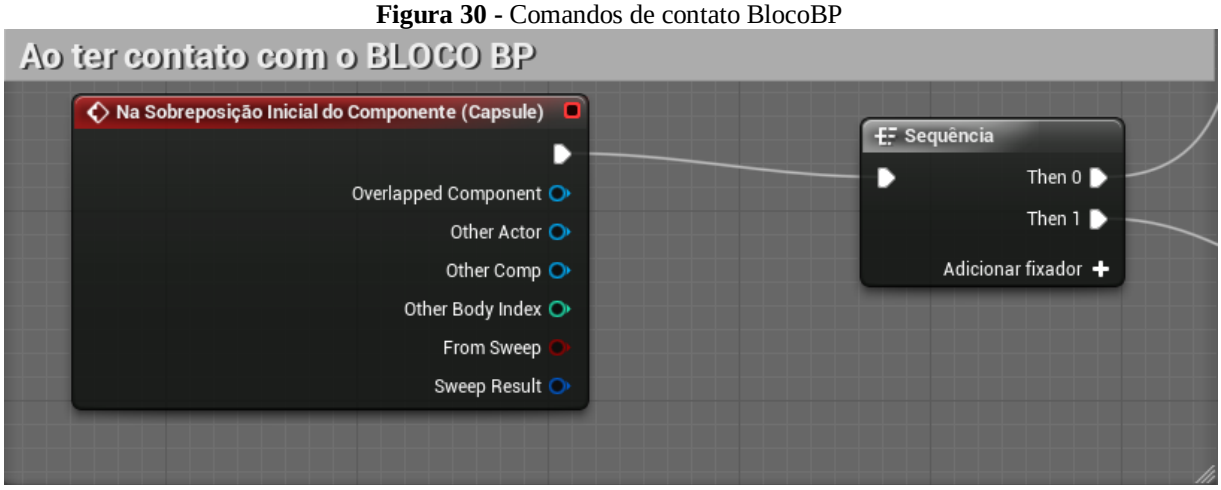
Fonte: Próprio Autor

O segundo passo foi a programação no Ator/Objeto com programação em Blueprint. Para isso selecionamos a aba de navegação ao lado de “Porta de Visualização” identificada como “Event Graph”.

Este é o local onde é desenvolvida a base de programação e cálculos para obtermos resultados em milissegundos sobre a base de velocidade dirigida pela Blueprint.

O primeiro passo foi iniciar a programação com a detecção de colisão entre o personagem e o bloco, para isso adicionamos um comando chamado “Add On Component Begin Overlap” referente ao componente de colisão (Capsule), em seguida mais à frente dinamizamos o processo e estrutura por meio de luzes dinâmicas, por isso adicionamos o comando identificado como “Sequence” e ligamos o “Nó” de saída do primeiro comando ao “Nó” de entrada do comando “Sequence”.

Figura 30 - Comandos de contato BlocoBP

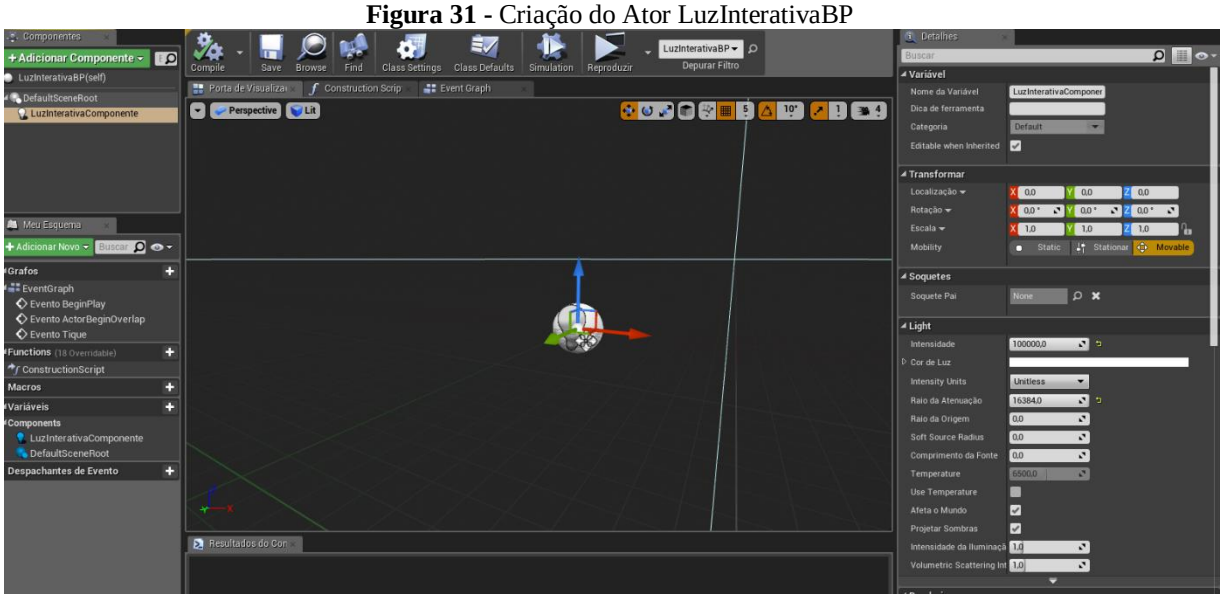


Fonte: Próprio Autor

Para criarmos a dinâmica de alterar as cores das luzes que adicionaremos ao cenário, criamos um Ator chamado “LuzInterativaBP”, o mesmo processo usado para criar o nosso BlocoBP. Em seguida, na aba de “Components”, criamos um componente identificado como “Componente de Luz Pontiuniforme” e demos o nome de “LuzInterativaComponente”.

Logo após, na barra de configurações de detalhes ao lado direito, atribuimos uma intensidade equivalente ao valor de 10000.00 com uma cor de luz emitida por um feixe branco.

Figura 31 - Criação do Ator LuzInterativaBP

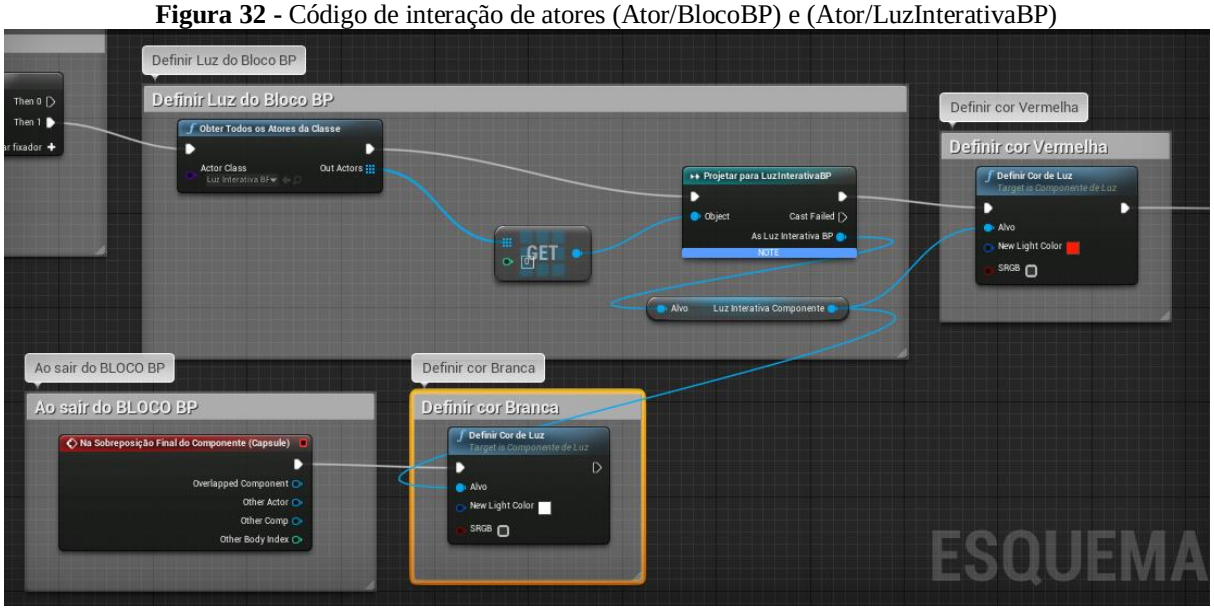


Fonte: Próprio Autor

Após concluir estas etapas, voltamos a programar o código do ator BlocoBP, seguindo de onde paramos, chamamos, então, o comando “Obter todos os atores da classe” e tínhamos como parâmetro o Ator que criamos “LuzInterativaBP”. Isso fez com que todos os objetos identificados por “LuzInterativaBP” fossem um valor a ser definido futuramente com um comando específico, não importa a quantidade que fosse adicionada ao cenário. Para isso também foi necessário adicionar um “Array” para fazer a contagem destes objetos no cenário, por isso, chamamos o comando Get (a copy) e adicionamos à saída de atores do comando “Obter todos os atores da classe”. Em seguida, precisamos informar a qual objeto seriam atribuídos todos estes comandos. Para isto, chamamos o comando “Projetar para LuzInterativaBP”, assim o código faria com que o objeto BlocoBP tivesse uma interação com o objeto “LuzInterativaBP”. Em resumo, essa interação dinâmica faz com que a luz seja alterada cada vez que o personagem entre em contato com o Ator/BlocoBP. O último passo para isto foi adicionarmos o comando “Definir cor de luz” e então informar a qual objeto do Ator/LuzInterativaBP seria atribuído, que neste caso seria o componente “Luz Interativa Componente”. Definimos a sua coloração com um tom vermelho no experimento do Ator/BlocoBP.

Para terminarmos esta etapa interação de luzes dinâmicas, adicionamos o comando “Add On Component End Overlapp” para que quando o personagem não tenha mais contato com o bloco volte a ter uma luz branca, então adicionamos o mesmo comando “Definir cor de Luz” e deixamos a sua coloração de luz branca novamente.

Figura 32 - Código de interação de atores (Ator/BlocoBP) e (Ator/LuzInterativaBP)



Fonte: Próprio Autor

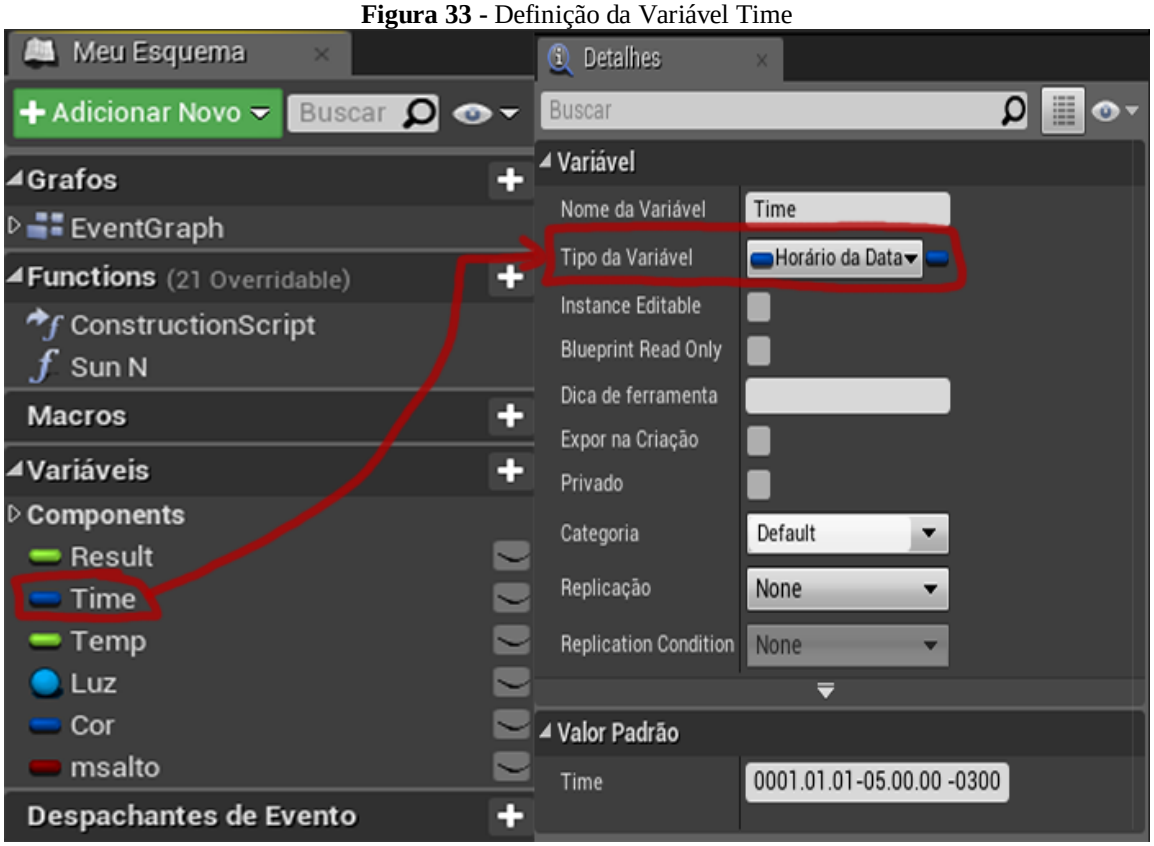
Feito isso, concluímos a etapa de interação de luzes ao colidir com o bloco. A seguir discorreremos sobre como criar a base de cálculo para obtermos o resultado do teste comparativo.

9.2. ESTRUTURA DE CÁLCULO

A estrutura de cálculo executada em ambos atores Blueprint e C++ consiste em um Loop que será definido manualmente, ou seja, a mesma ação de execução será realizada por um número “x” de vezes, fazendo com que cada Looping calcule o tempo em milissegundos e nos traga o resultado visualmente por meio da interface do jogo.

Para isso criamos uma variável no mesmo ambiente de programação dos blocos pelo menu “Variáveis” e chamamos de “Time” prosseguimos no menu “Detalhes” para definir o tipo da variável que criamos e definimos o tipo da variável como “Horário da Data”

Figura 33 - Definição da Variável Time

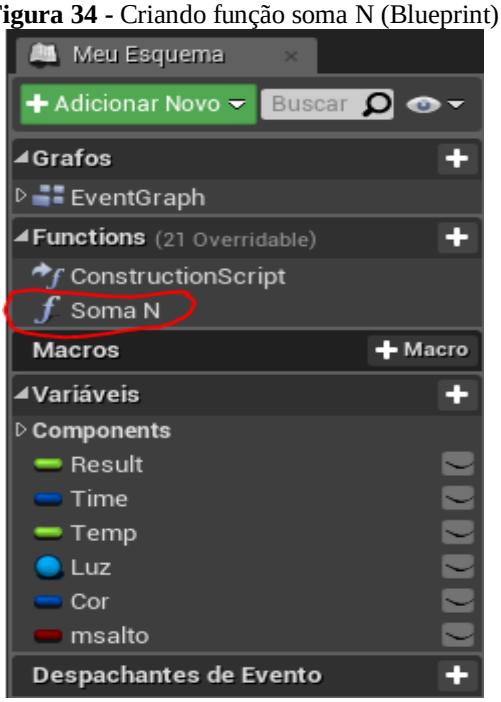


Fonte: Próprio Autor

9.3. PROGRAMAÇÃO EM BLUEPRINT

Em seguida criamos a função que define o resultado da soma e o tempo decorrido por meio do loop da estrutura da programação passando por meio deste método que chamaremos de “Soma N” que tem por objetivo verificar uma quantidade “X” de vezes pré-definida.

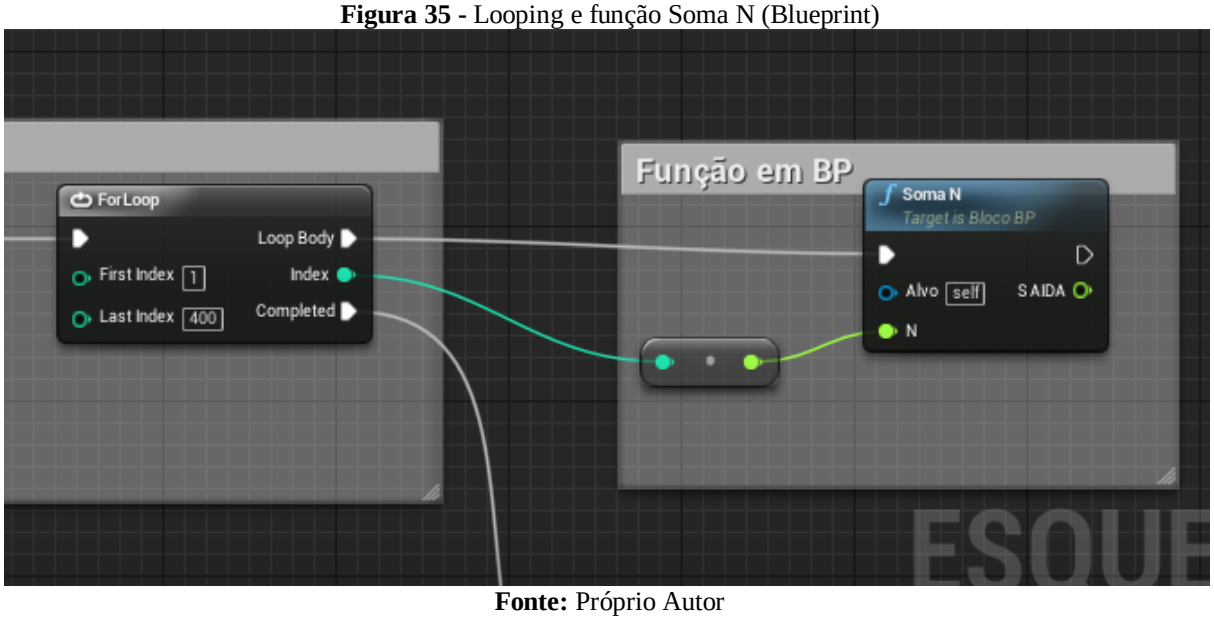
Figura 34 - Criando função soma N (Blueprint)



Fonte: Próprio Autor

Depois de criarmos a função, estabelecemos a sua estrutura de cálculo de modo que retornasse um valor assim que o looping fosse encerrado.

Figura 35 - Looping e função Soma N (Blueprint)



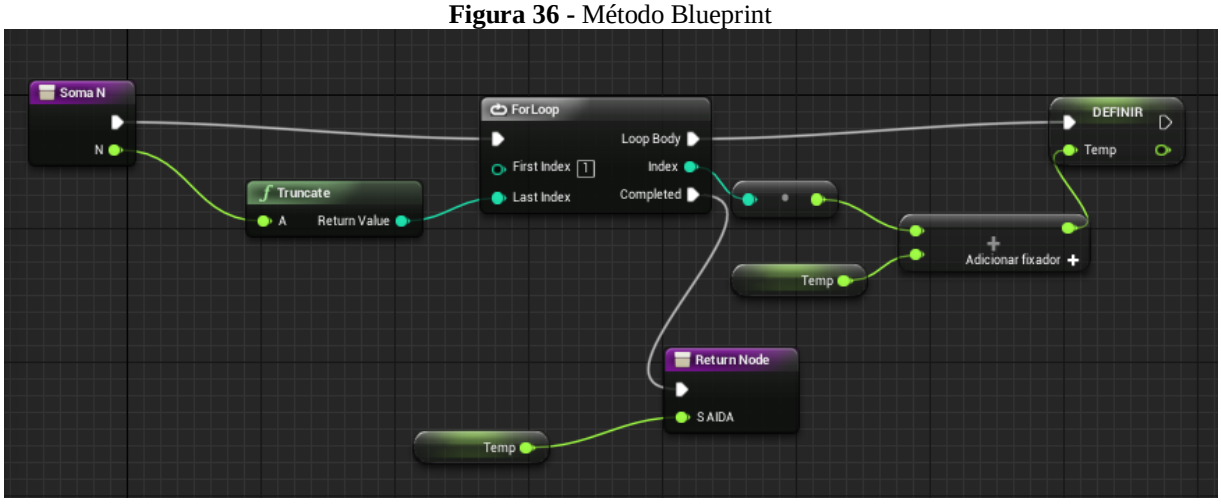
Fonte: Próprio Autor

O método "Soma N" inicia com um looping que receberá o valor pré-definido na estrutura anterior relatado na "Error! Reference source not found." no componente "ForLoop", sendo o valor até 400, inicializando em 1.

Com a função "Soma N" iniciada, temos uma variável de entrada com o nome "N" e a variável de saída chamada de "Saída", ambas de tipo *float*.

Com isso chamamos a função “Truncate”, e ligamos com a entrada da função. A função “Truncate” arredonda um valor de número como Double ou Float para um número inteiro (EPIC GAMES, 2018). Com isso prosseguimos com a leitura do looping anterior através de outro looping, desta vez essa função faz a leitura baseando-se nos primeiros valores já definidos anteriormente. Assim, conseguimos obter os resultados armazenando a soma e o tempo em milissegundos através da variável “Temp”. Deste modo completamos o looping com a saída da leitura ligada a esta variável para ser visualmente transmitida à tela do game pela estrutura que veremos adiante.

Figura 36 - Método Blueprint



Fonte: Próprio Autor

Após o primeiro looping se encerrar, também encerramos automaticamente o looping do método Blueprint como na imagem “**Error! Reference source not found.**” e “**Error! Reference source not found.**” relata. Portanto, ao se encerrar, a função “Imprimir String” é responsável por mostrar o resultado visualmente.

Para isso, é preciso chamar a função *Append* que concatena várias Strings para transformar em uma nova, neste caso, o resultado de “N” é convertido em “String” através da função “Imprimir String” e, em seguida, a função GetTotalMiliseconds é chamada, passando a variável “Time” criada anteriormente e a função “Now” que retorna o tempo atual da máquina, porém, neste caso, este valor é transformado em milissegundos.

Figura 37 - Reposta em MS (Blueprint)

Fonte: Próprio Autor

9.4. PROGRAMAÇÃO EM C++

Para realizar o teste comparativo, foi preciso criar o mesmo teste com programação na linguagem C++. Neste caso, selecionamos o ator que será programado com C++ e, clicamos sobre Content Browser com o botão direito do mouse abrindo uma janela com diversas opções, dentre elas a opção New C++ Class.

Quando criamos a classe C++ o Visual Studio inicia automaticamente. Utilizamos o ambiente de desenvolvimento Microsoft Visual Studio Code para criar a programação em código. Neste caso são criados dois arquivos com extensão cpp e h. O arquivo com extensao "h" define todas as variáveis que serão utilizadas e iniciadas, como uma espécie de construtor, e os arquivos com extensão "cpp" são os métodos.

O primeiro passo é abrir o arquivo de biblioteca com extensão ".h" e criar uma função de acesso global que possa ser acessada pelos Blueprints, para isso criamos uma estrutura do tipo "public" e usamos a função UFUNCTION e passamos o parâmetro do próprio Unreal chamado "BlueprintCallable". Em seguida definimos as variáveis que iremos utilizar, como na figura 38 abaixo.

Figura 38 - Iniciando estrutura em C++

```
public:
    //Função "BlueprintCallabe" - Habilitar a chamada de função nas Blueprints da Unreal 4

    UFUNCTION(BlueprintCallable)
    float GetTotalSum(float N);
    UFUNCTION(BlueprintCallable)
    float SumN(float h);
    // Called every frame
    virtual void Tick(float DeltaTime) override;
```

Fonte: Próprio Autor

Neste caso criamos as duas funções iniciando as variáveis GetTotalSum com o tipo "float" que passará por parâmetro a variável N do tipo *float* também, essa variável estará encarregada de armazenar a soma total. Em seguida, criamos a variável SumN do tipo *float*,

também passando por parâmetro a variável “H” do tipo *float*, que fará a soma. Basicamente com isso encerramos as inicializações das variáveis e funções que são que ficam no arquivo da biblioteca.

Em seguida foram criadas mais funções C++ por meio das quais serão obtidos o cálculo para pegar o total do tempo e o tempo total baseado no valor do loop.

Figura 39 - Métodos GetTotalSum e SumN

```
//Função de calculo para pegar o total de Tempo
float ACalculoC::GetTotalSum(float N)
{
    float result = 0;
    for (int i = 1; i <= N; i++)
    {
        result += SumN(i);
    }
    return result;
}

//Calcular o tempo total baseado no valor do Loop na blueprint view
float ACalculoC::SumN(float h)
{
    float temp = 0;
    for (int i = 1; i <= h; i++)
    {
        temp += i;
    }
    return temp;
}
```

Fonte: Próprio autor

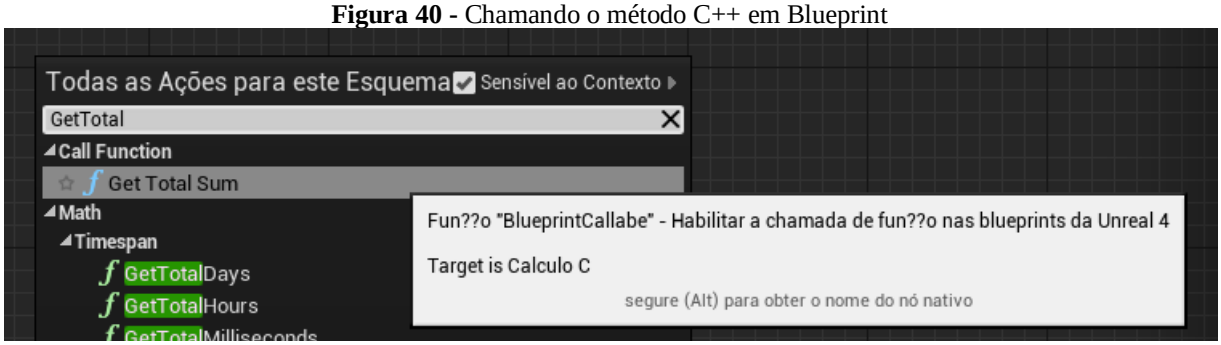
Para criamos estas funções iremos iniciar com o tipo *float* em ambos casos, passando a classe “ACalculoC” que é o nome da classe quando criamos o ator em C++, fazendo com que obtenhamos as variáveis que definimos de modo global, neste caso obtendo informações do arquivo de biblioteca onde definimos as variáveis iniciais. Sendo assim, podemos chamar as funções que iniciamos como GetTotalSum e SumN como visto na imagem.

No método GetTotalSum, iremos iniciar uma variável “result” com o valor 0 e iniciar uma estrutura “for” como um looping tendo como a variável inteira “i” para a realização do contador de referência. Para cada repetição iremos retornar a variável que iniciamos “result” em +1 somando a cada repetição e retornando o valor atualizado.

No método SumN, iremos utilizar o mesmo conceito que o método GetTotalSum, apenas mudando a variável “result” para “temp” que se refere ao tempo de looping para soma de N. Nota-se que a cada repetição de laço da estrutura SumN, o método GetTotalSum possui a varaivel “result” recebendo o valor de SumN (i) para cada repetição. Neste caso os dois métodos possuem comunicação para atualização do valor de “Result”. Com isso, terminamos nossa estrutura em C++.

Voltando ao ambiente de esquema Blueprint podemos chamar a função que criamos em C++ por meio da função que iniciamos na biblioteca a função UFUNCTION (BlueprintCallable), tornando visível para se utilizar visualmente em BlueprintView como no caso da imagem a seguir.

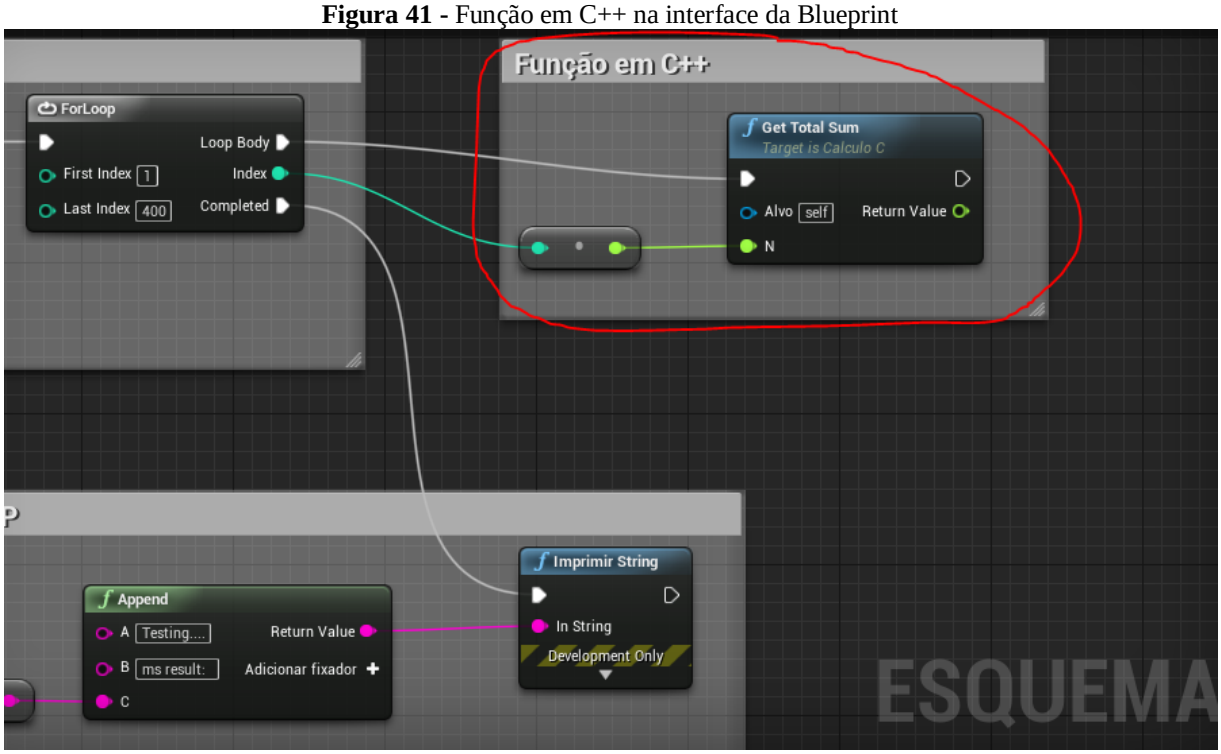
Figura 40 - Chamando o método C++ em Blueprint



Fonte: Próprio autor

Deste modo, nosso esquema em C++ e Blueprint são semelhantes em questão de estrutura lógica de programação, após chamar este componente faremos as mesmas ligações por meio de nós e então a função C++ funcionará do mesmo modo que vimos ao criar a função em Blueprint relatado no tópico 9.3 (ver figura 35).

Figura 41 - Função em C++ na interface da Blueprint



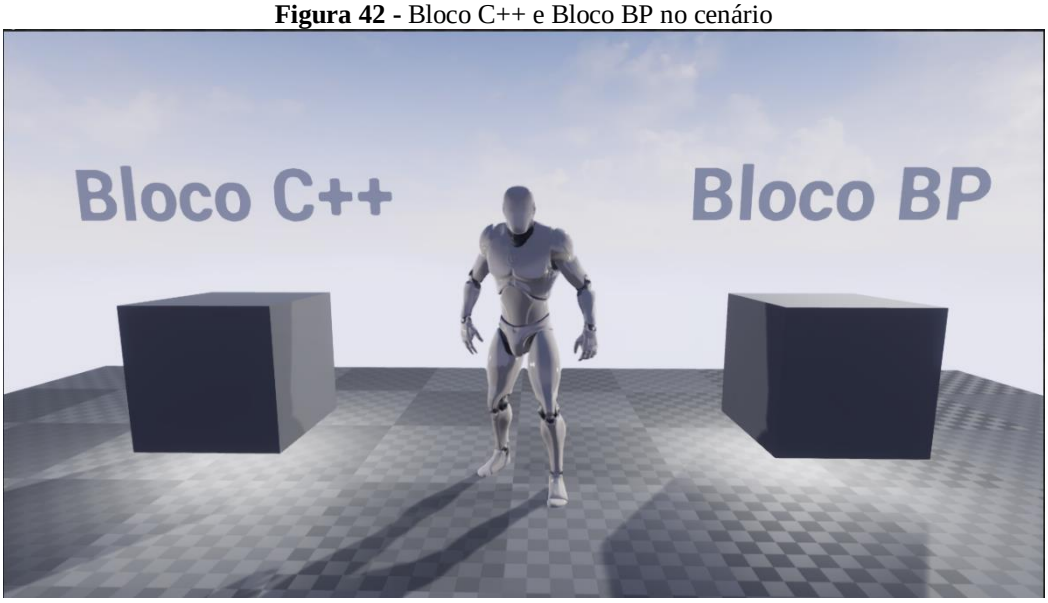
Fonte: Próprio Autor

Com isto terminamos a programação as estruturas em C++ e Blueprint fazendo com que seja possível realizar os testes por meio de uma interação ao colidir com os atores por meio da colisão do PlayerCharacter.

10. TESTE COMPARATIVO

Com as estruturas programadas, os objetos e atores definidos e a interação por colisão detectável, podemos realizar o teste comparativo entre a Blueprint e C++, podemos colocar os atores no cenário, desta forma com o personagem do jogador faríamos uma interação com os atores definidos como os blocos que colocamos como Mesh (Malha Estática).

Figura 42 - Bloco C++ e Bloco BP no cenário

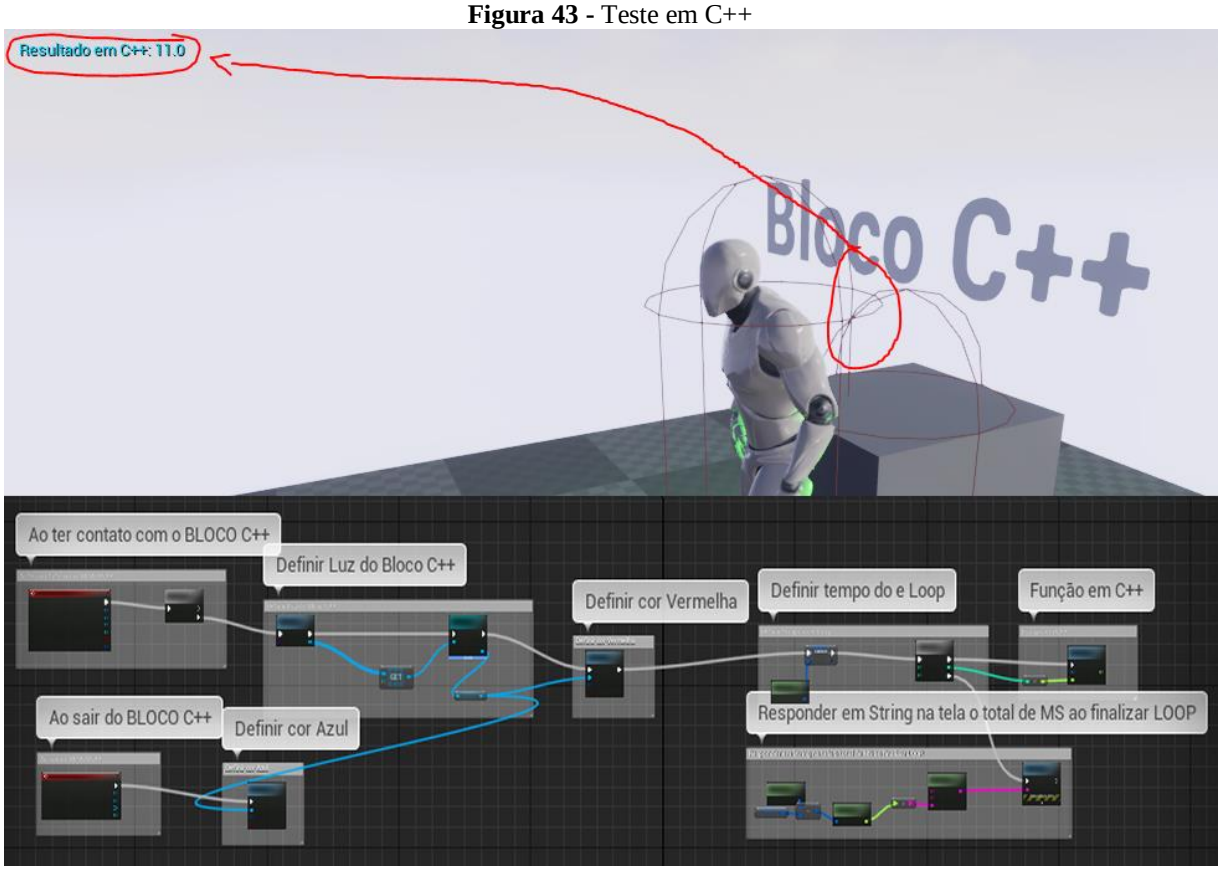


Fonte: Próprio Autor

Pelo toque de colisão, toda a programação já seria executada, imprimindo na tela o tempo em milissegundos, assim podemos retirar como informações qual a linguagem se tornou mais eficiente e poderosa em tempo de execução e desempenho, já que ambas linguagens estão estruturadas exatamente da mesma forma.

A seguir no exemplo da imagem abaixo com o bloco em C++, vemos que quando as caixas de colisões, se tocam há um gatilho de disparo que programamos na estrutura dos blocos, gerando o resultado apontado no canto superior esquerdo da tela do jogador.

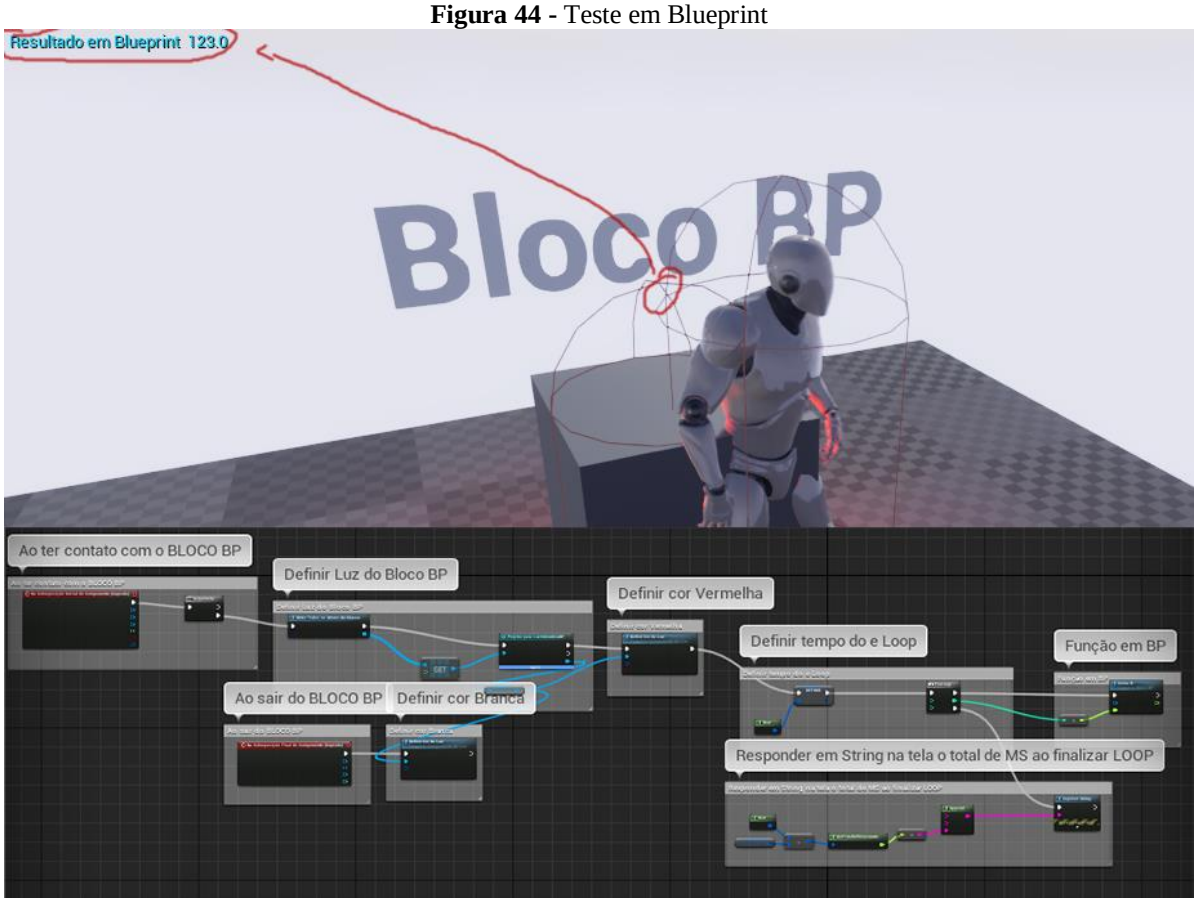
Figura 43 - Teste em C++



Fonte: Próprio Autor

O mesmo ocorre quando tocamos o bloco em Blueprint, a colisão também “chama” toda a estrutura que programamos para mostrar o resultado na tela do jogador.

Figura 44 - Teste em Blueprint



Fonte: Próprio Autor

11. RESULTADO COMPARATIVO

Por fim, podemos analisar o resultado dos dois testes e colocar em uma tabela e verificar suas variações de desempenho. Vale lembrar que, dependendo da potência da máquina, isto é, as configurações como placa de vídeo, memórias, processadores, os resultados podem ser diferentes dos que foram obtidos aqui.

Para calcular os resultados definimos inicialmente um looping de repetição que faria o mesmo processo por até 1000 vezes consecutivas, mas, para testar outros possíveis resultados e verificar o desempenho, o processo de looping foi testado com o início em 100, 300, 500, 800 e 1000 vezes consecutivas.

Definimos esses valores pois, para obtermos um resultado significativo o Looping precisa ser testado com uma variação consecutiva bem elevada, logo abaixo podemos notar em uma tabela um resultado comparativo entre as duas formas de programação.

RESULTADOS			
Looping	**Linguagem em C++**	**Linguagem em Blueprint**	**Velocidade de C++**
100 vezes	1 milissegundos	7 milissegundos	7x mais rápido
300 vezes	5 milissegundos	65~67 milissegundos	13.4x mais rápido
500 vezes	22~27 milissegundos	322~330 milissegundos	12x mais rápido
800 vezes	89~91 milissegundos	Falhou	Falha na comparação
1000 vezes	175~177 milissegundos	Falhou	Falha na comparação

12. CONCLUSÃO E CONSIDERAÇÕES FINAIS

Após os resultados comparativos na realização deste trabalho, é possível afirmar que a linguagem C++ em comparação ao Blueprint é extremamente poderosa e possui um desempenho muito alto, nota-se que em alguns casos a linguagem em Blueprint chegou a falhar finalizando o processo de execução durante o teste (crash).

Apesar de uma linguagem como a Blueprint ter sido 10 vezes mais lenta, ela possui grandes vantagens como a facilidade de programar e criar um protótipo bem mais rápido que a linguagem em C++. Neste caso, linguagem como Blueprint deve ter o seu foco em jogos direcionados a protótipos ou funcionalidades básicas devido a sua limitação de leitura durante o processo de execução.

Grande parte da lentidão que jogos que utilizam a Blueprint podem sofrer, se deve ao fato de que necessitam de cálculos matemáticos e que podem acarretar uma queda de desempenho muito grande.

Com isso notamos que, projetos iniciados como protótipos devem ter sua utilização em Blueprint por sua facilidade de programar e ter uma base visível produzido de forma rápida e em curto prazo, porém, caso o projeto se estender será necessário criar classes em C++ pois, o desempenho, como vimos no processo de comparação, será sempre superior ao Blueprint, evitando problemas de queda de desempenho, possibilitando diversas interações sem se preocupar com possíveis falhas, do contrário, o produto poderá ter um nível de aceitação negativa.

Vale lembrar que linguagem como Blueprint exige que o desenvolvedor possua apenas nível lógico de programação, enquanto C++ necessita que o desenvolvedor saiba programar em linguagens de programação além do nível lógico.

Por fim, como objetivo do trabalho foi testar a viabilidade do uso das duas linguagens. Como vimos nos testes de nossas experiências. Com base em nossas observações relatadas, podemos afirmar que as duas linguagens são viáveis dependendo do que se deseja produzir e o quão grande ele pode se tornar.

13. REFERÊNCIAS

ADOBE. Faça upload e monte personagens 3D com o Mixamo. **Adobe Brasil**, 2018. Disponivel em: <https://helpx.adobe.com/br/creative-cloud/help/mixamo-rigging-animation.html>. Acesso em: 22 out. 2018.

ALISSON. Historia do C / C++. **Devmedia**, 2012. Disponivel em: <https://www.devmedia.com.br/historia-do-c-c/24029>. Acesso em: 09 out. 2018.

ARRUDA, E. P. **Fundamentos Para o Desenvolvimento de Jogos Digitais**. Recife: Bookman, v. 1, 2014. Disponivel em: <http://srvd.grupoa.com.br/uploads/imagensExtra/legado/A/ARRUDA_Eucidio_P/Fundamento_Desenvolvimento_Jogos_Digitais/Lib/Cap_01.pdf>. Acesso em: 17 ago. 18.

BAHNER, C. God of War 3 Remastered: Alle Trophäen - Tipps und Leitfaden für 100%. **Giga**, 2015. Disponivel em: <https://www.giga.de/spiele/god-of-war-3/tipps/god-of-war-3-remastered-alle-trophaeen-tipps-und-leitfaden-fuer-100/>. Acesso em: 10 nov. 2018.

BATISTA, S. Chrono Trigger – Versão Steam recebe segundo update. **Proximonivel**, 2018. Disponivel em: <http://proximonivel.pt/chrono-trigger-versao-steam-recebe-segundo-update/>. Acesso em: 10 nov. 2018.

BELLIS, M. The History of Spacewar. **The History of Spacewar**, 2017. Disponivel em: <https://www.thoughtco.com/history-of-spacewar-1992412>. Acesso em: 16 ago. 2018.

BOHEMIA INTERACTIVE. Survive in a harsh post-apocalyptic multiplayer landscape. **DayZ | Bohemia Interactive**, 2018. Disponivel em: <https://www.bohemia.net/games/dayz>. Acesso em: 28 ago. 2018.

BROWN, S. Minor Project 11 - Option 1: Where To From Here? **sbbrown-gdd110.blogspot**, 2012. Disponivel em: <http://sbbrown-gdd110.blogspot.com/2012/11/minor-project-11-option-1-where-to-from.html>. Acesso em: 11 nov. 2018.

CARDOSO, B. O que é um Jogo de Aventura? **Seugame.com**, 2017. Disponivel em: <http://seugame.com/o-que-e-um-jogo-de-aventura/>. Acesso em: 28 ago. 2018.

CARDOSO, B. O que é um jogo de plataforma? **Seugame**, 2017. Disponivel em: <http://seugame.com/o-que-e-um-jogo-de-plataforma/>. Acesso em: 28 ago. 2018.

CASAVELLA, E. Breve historia da linguagem C. **Linguagem C**, 2004. Disponivel em: <http://linguagemc.com.br/breve-historia-da-linguagem-c/>. Acesso em: 09 out. 2018.

CONTÉMGAMES. Historia dos jogos eletrônicos - Super Mario World. **Contém Games**, 2018. Disponivel em: <http://contemgames.com.br/historia/jogos/1990-Super-Mario-World.aspx>. Acesso em: 28 ago. 2018.

COSTA, R. Quais são os gêneros de jogos de vídeo game? **Designzeroum**, 2014. Disponivel em: <https://designzeroum.com.br/quais-sao-os-generos-de-jogos-de-video-game/>. Acesso em: 28 ago. 2018.

CRAWFORD, C. **The Art of Computer Game Design**. Berkeley: McGraw-Hill/Osborne Media, 1984.

DIAS, R. Unreal Engine – Guia Completo para Iniciantes, 2017. Disponivel em: <https://producaodejogos.com/game-engine/>. Acesso em: 2 out. 2018.

DIAS, T. J. et al. Estudo sobre os Diversos Gêneros de Jogos e sua Aplicabilidade no Ensino. **Revista Gestão Universitária**, 2014. Disponivel em: <http://gestaouniversitaria.com.br/artigos/estudo-sobre-os-diversos-generos-de-jogos-e-sua-aplicabilidade-no-ensino>. Acesso em: 22 ago. 2018.

DIAS, T. J. et al. Estudo sobre os Diversos Gêneros de Jogos e sua Aplicabilidade no Ensino. **Revista Gestão Universitária**, 2014. Disponivel em: <gestaouniversitaria.com.br/artigos/estudo-sobre-os-diversos-generos-de-jogos-e-sua-aplicabilidade-no-ensino>. Acesso em: 30 ago. 2018.

EPIC GAMES. Unreal Engine Features. **Unreal Engine**, 2018. Disponivel em: <https://www.unrealengine.com/en-US/features>. Acesso em: 2 set. 2018.

FELIPE, L. DayZ ganha data de lançamento no Xbox One, e ela está bem próxima! **Combo Infinito**, 2018. Disponivel em: <https://www.comboinfinito.com.br/principal/dayz-ganha-data-de-lancamento-no-xbox-one-e-ela-esta-bem-proxima/>. Acesso em: 10 nov. 2018.

G1. Jogo de tabuleiro 'War' ganha versão digital para tablets e computadores. **G1.com**, 2015. Disponivel em: <http://g1.globo.com/tecnologia/games/noticia/2015/10/jogo-de-tabuleiro-war-ganha-versao-digital-para-tablets-e-computadores.html>. Acesso em: 10 nov. 2018.

GAMERHUB. Ubisoft lança Rainbow Six Sonic Starter Edition de Tom Clancy no PC. **Gamerhub.tv**, 2018. Disponivel em: <http://gamerhub.tv/article/3098/ubisoft-releases-tom-clancys-rainbow-six-siege-starter-edition-on-pc>. Acesso em: 10 nov. 2018.

GAMES, E. Unreal Engine Features. **Unreal Engine**, 2018. Disponivel em: <https://www.unrealengine.com/en-US/features>. Acesso em: 2 out. 2018.

GEE, J. P. **What Video Games Have to Teach Us About Learning and Literacy**. New York: Palgrave/Macmillan, 2003.

GONÇALVES, R. Origem do tetris. **Historia do mundo**, 2018. Disponivel em: <https://historiadomundo.uol.com.br/idade-contemporanea/origem-do-tetris.htm>. Acesso em: 30 ago. 2018.

HUIZINGA, J. **Homo Ludens - O jogo com elemento da cultura**. 4º. ed. São Paulo - SP: PERSPECTIVA S.A, 2001.

HUNSBERGER, D. Need for Speed: Most Wanted (2005) is overrated. **Carthrottle**, 2017. Disponivel em: <https://www.carthrottle.com/post/662jpqb/>. Acesso em: 10 nov. 2018.

IMPACTA, R. Você sabe o que é Visual Studio? **Impacta**, 2018. Disponivel em: <https://www.impacta.com.br/blog/2017/12/11/voce-sabe-o-que-e-visual-studio/>. Acesso em: 09 out. 2018.

JOHNSTON, R. NBA 2K18: Here's a List of Probable Superstars who can feature in the Game as the Cover Star. **Cldlaurentides.org**, 2017. Disponivel em: <http://cldlaurentides.org/nba-2k18-heres-list-probable-superstars-can-feature-game-cover-star/>. Acesso em: 10 nov. 2018.

KARASINSKI, V. A história dos games de simulação. **Tecmundo**, 2012. Disponivel em: <https://www.tecmundo.com.br/video-game-e-jogos/32684-a-historia-dos-games-de-simulacao.htm>. Acesso em: 28 ago. 2018.

KITAMURA, C. Parabens Visual Studio! Feliz Aniversario! **Celso Kitamura**, 09 out. 2017. Disponivel em: <https://celsokitamura.com.br/parabens-visual-studio/>. Acesso em: 09 out. 2018.

KLAPPENBACH, M. What is a Platform game? **Lifewire**, 2018. Disponivel em: <https://www.lifewire.com/what-is-a-platform-game-812371>. Acesso em: 28 ago. 2018.

LIMA, H. G. F. **Brainstorming**. Senai. [S.l.], p. 4. 2011.

MAGE, K. Uma das mais notáveis e divertidas aventuras de Mario no saudoso Super Nintendo. **Gamehall**, 2016. Disponivel em: <http://gamehall.uol.com.br/v10/super-mario-world/>. Acesso em: 10 nov. 2018.

MAGNANI, L. H. Por dentro do jogo: Videogames e Formação de Sujeitos críticos., Campinas, 2017. Disponivel em: <http://www.scielo.br/pdf/tla/v46n1/a09v46n1.pdf>. Acesso em: 19 ago. 2018.

MALLMANN, E. R. ESTUDO E DESENVOLVIMENTO DE UM JOGO UTILIZANDO, Santa Rosa, 2012. Disponivel em: <http://bibliodigital.unijui.edu.br:8080/xmlui/bitstream/handle/123456789/1368/TCC%202012%20Eduardo%20R%20Mallmann.pdf?sequence=1>. Acesso em: 22 ago. 2018.

MARCHELLETTA, C. Play Simulation Games. **Lifewire**, 2016. Disponivel em: <https://www.lifewire.com/play-simulation-games-837139>. Acesso em: 28 ago. 2018.

MATT BARTON, B. L. The History of Spacewar!: The Best Waste of Time in the History of the Universe. **GAMASUTRA**, 2009. Disponivel em: <http://www.gamasutra.com/view/feature/132438/the_history_of_spacewar_the_best_.php>. Acesso em: 10 nov. 2018.

MENDONÇA, V. G. D. Gêneros de jogos - Estratégia. **Ponto V**, 2018. Disponivel em: <www.pontov.com.br/site/game-design/67-classificacao-dos-jogos/71-generos-de-jogos-estrategia>. Acesso em: 30 ago. 2018.

MONTEIRO, R. Mortal Kombat X: conheça todos os lutadores já confirmados para o jogo. **Techtudo**, 2015. Disponivel em: <https://www.techtudo.com.br/noticias/noticia/2015/01/mortal-kombat-x-conheca-todos-os-lutadores-ja-confirmados-para-o-jogo.html>. Acesso em: 11 nov. 2018.

NEMIROFF, P. “A Very Big, Epic Sci-Fi” TETRIS Movie in the Works, Obviously [Updated]. **Collider**, 2014. Acesso em: 10 nov. 2018.

NEWMAN, L. História do jogo de quebra-cabeça - como o gênero veio a ser. **Dragon Academy Game**, 2018. Disponivel em: <https://dragonacademygame.com/history-puzzle-game-genre-came/>. Acesso em: 29 ago. 2018.

NOVAES, R. O que é e para que serve IDE'S? **Psafe**, 2004. Disponivel em: <https://www.psafe.com/blog/o-que-serve-ide/>. Acesso em: 09 out. 2018.

OXFORD, N. What's the Definition of an Action Game? **Lifewire**, 2018. Disponivel em: <https://www.lifewire.com/nintendo-action-game-1126179>. Acesso em: 28 ago. 2018.

OXLAND, K. **Game and Design**. Harlow - Inglaterra: Addison, 2004.

PACHECO, M. Tennis for Two, o primeiro game da história, completa 55 anos. **Gamehall**, 2013. Disponivel em: <http://gamehall.uol.com.br/v10/tennis-for-two-o-primeiro-game-da-historia-completa-55-anos/>. Acesso em: 15 ago. 2018.

PEREIRA, A. L. L. A Utilização do Jogo como recurso de motivação e aprendizagem, 2013. Disponivel em: <https://repositorio-aberto.up.pt/bitstream/10216/71590/2/28409.pdf>. Acesso em: 14 ago. 2018.

PINHEIRO, J. Definindo o gênero de jogos de luta: a importância e o legado de Street Fighter. **Canal Tech**, 2018. Disponivel em: <https://canaltech.com.br/games/especial-street-fighter-30-anos-114860/>. Acesso em: 17 set. 2018.

PIRES, J. IDE'S usar ou não usar? Eis a questão. **Becode**, 2017. Disponivel em: <https://becode.com.br/ides-usar-ou-nao-usar/>. Acesso em: 09 out. 2018.

PIRIFORM. **Speccy 1.31.732**. Londres: Piriform Ltd., 2017. Disponivel em: <https://speccy.soft32.com/>. Acesso em: 22 out. 2018.

PORTO, G. Tetris. **Infoescola**, 2018. Disponivel em: <https://www.infoescola.com/curiosidades/tetris/>. Acesso em: 30 ago. 2018.

ROGERS, S. **Level Up! the Guide to Great Video Game Design**. Chichester: John Wiley & Sons, Ltd, 2010.

ROMERO, M. Salvar e Carregar em Blueprints. **Romero Blueprints**, 2015. Disponivel em: <http://romeroblueprints.blogspot.com/2015/01/salvar-e-carregar-em-blueprints.html>. Acesso em: 11 nov. 2018.

SATHEESH. **Unreal Engine 4 Game Development Essentials**. Birmingham - Mumbai: Packt Publishing, 2016.

SEIBEL, G. Analise completa | NBA 2K18. **Manual dos games**, 2017. Disponivel em: <https://manualdosgames.com/analise-completa-nba-2k18/>. Acesso em: 28 ago. 2018.

SEWELL, B. **Blueprints Visual Scripting for Unreal Engine**. Birmigham - Mumbai: Packet - Publishing, 2015.

SHERIF, W. **Learning C++ by Creating Games with UE4**. Birmighan - Mumbai: Packt - Publishing, 2015.

SONIC the Hedgehog. Plataforma: Nintendo: SEGA. 1991.

STUDIOS, P. Conheça a história da Unreal Engine, o motor de jogos gratuito para desenvolvedores. **PIX STUDIOS**, 2015. Disponivel em: <http://www.pixstudios.com.br/blog/novidades-de-computacao-grafica-e-games/hist%C3%B3ria-da-unreal-engine-motor-de-jogos-gratuito-baixar-desenvolvedores-estudantes>. Acesso em: 2 out. 2018.

SUPER Mario Bros. Produção: Nintendo.: 1985.

TECHTUDO. Encare perseguições e outros modos de corrida em Need for Speed. **Techtudo**, 2010. Disponivel em: <https://www.techtudo.com.br/tudo-sobre/nfs-most-wanted-2005.html>. Acesso em: 28 ago. 2018.

TIAGO. Conheça os nove gêneros dos Jogos Digitais. **Infiniteloop**, 2017. Disponivel em: <http://www.infiniteloop.com.br/conheca-os-nove-generos-dos-jogos-digitais/>. Acesso em: 28 ago. 2018.

TOSCHI, G. Ação, RPG, Plataforma? Saiba mais sobre os gêneros dos jogos e quais games se classificam nos mais conhecidos. **Nintendo Blast**, 2012. Disponivel em: <https://www.nintendoblast.com.br/2012/04/gamedev-conheca-os-tipos-de-jogos.html>. Acesso em: 28 ago. 2018.

UBISOFT. Tomclancy's Rainbow Six Siege. **Rainbow6**, 2018. Disponivel em: <https://rainbow6.ubisoft.com/siege/pt-br/game-info/index.aspx>. Acesso em: 28 ago. 2018.

UNIVERSITÁRIA, R. G. Estudo sobre os Diversos Gêneros de Jogos e sua Aplicabilidade no Ensino. **Revista Gestão Universitária**, 2014. Disponivel em: <http://gestaouniversitaria.com.br/artigos/estudo-sobre-os-diversos-generos-de-jogos-e-sua-aplicabilidade-no-ensino>. Acesso em: 17 set. 2018.

VALCASARA, N. **Unreal Engine Game Development Blueprints**. Birmingham - Mumbai: Packet Publishing, 2015.

VALLE, C. Qual a diferença entre CPU e GPU? **Cissa Magazine**, 2015. Disponivel em: <https://www.cissamagazine.com.br/blog/gpu-vs-cpu>. Acesso em: 22 out. 2018.

WERNECK, V. Shadow of the Tomb Raider: jogamos a aventura final da origem de Lara. **Techtudo**, 2018. Disponivel em: <https://www.techtudo.com.br/noticias/2018/04/shadow-of-the-tomb-raider-jogamos-a-aventura-final-da-origem-de-lara.ghtml>. Acesso em: 28 ago. 2018.

WERNECK, V. Techtudo. **Techtudo**, 2018. Disponivel em: <https://www.techtudo.com.br/noticias/2018/04/shadow-of-the-tomb-raider-jogamos-a-aventura-final-da-origem-de-lara.ghtml>. Acesso em: 23 ago. 2018.

Blueprint vs C++ Perfomance. Alt alt. **Youtube.** 25 Abr de. 2017. 2min10s. Disponível em: < https://www.youtube.com/watch?v=V707r4bkJOY>. Acesso em: 14 out. 2018.