Redieh Dilli Scarano Rodrigues

DESENVOLVIMENTO DE LEVEL DESIGN PARA O JOGO SHARDS

Projeto de Conclusão de Curso submetido ao Programa de Graduação da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Design.

Orientador: Prof. Dr. Flávio Andaló

Florianópolis
2019

Ficha de identificação da obra elaborada pelo autor
através do Programa de Geração Automática da Biblioteca Universitária
da UFSC.

Rodrigues, Redieh Dilli Scarano
Desenvolvimento de Level Design para o jogo Shards / Redieh Dilli Scarano Rodrigues;
orientador, Flávio Andaló, 2019.
116 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro de Comunicação e Expressão, Graduação em Design,
Florianópolis, 2019.

Inclui referências.

1. Design. 2. Level Design. 3. Game Design. I. Andaló, Flávio. II. Universidade Federal de Santa Catarina. Graduação em Design. III. Título.

Redieh Dilli Scarano Rodrigues

Desenvolvimento de Level Design para o jogo Shards

Este Projeto de Conclusão de Curso (PCC) foi julgado adequado para obtenção do Título de Bacharel em Design e aprovado em sua forma final pelo Curso de Design da Universidade Federal de Santa Catarina.

Florianópolis, 22 de novembro de 2019.

Profª. Mary Vonni Meürer, Dra. Coordenadora do Curso de Design UFSC

Banca Examinadora:

Prof. Flávio Andaló, Dr. (Universidade Federal de Santa Catarina)

Prof.ª Mônica Stein, Dr.ª (Universidade Federal de Santa Catarina)

Prof. Renan De Paula Binda, Ms. (Universidade Federal de Santa Catarina)

Documento assinado digitalmente
Flavio Andalo
Data: 04/12/2019 16:59:20-0300
CPF: 712.101.649-49

Flávio Andaló
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Gostaria de agradecer primeiramente ao meu amigo Igor Freitas, coautor do projeto e parceiro de muitos outros.
À minha família que sempre apoiou minhas escolhas.
Aos professores que colaboraram para a minha formação.
Aos meus amigos do Design e da minha cidade natal por me incentivarem a sua própria maneira.
Ao meu amigo Rafael, que me ajudou a conceber este projeto.
E a todos os outros que eu possa ter esquecido de mencionar.

RESUMO

O trabalho desenvolvido neste projeto visa a elaboração de um Nível para o jogo em 3 dimensões (3D) *Shards* para a plataforma Computador (PC), baseado nos conceitos estudados *sobre game* e *level design* e a forma como essas áreas interagem durante o processo de criação de um protótipo jogável. O projeto foi composto dentro da Engine de desenvolvimento de jogos Unreal Engine 4.

Palavras-chave:Game Design. Level Design. Jogo 3D. Unreal Engine.

ABSTRACT

The work developed during this Project aims to create a level for the 3D game Shards, for the PC platform, based upon the studied concepts of game and level design and the way these fields interact during the process of creating a playable prototype. The project was made in the game development engine, Unreal Engine 4.

Keywords: Game Design. Level Design. 3D Game. Unreal Engine.

LISTA DE FIGURAS

Figura 1 – Jogo Pong para Atari....18
Figura 2 – Super Mario Bros para NES....18
Figura 3 – Mercado mundial de games por segmentos.....21
Figura 4 – Metodologia Double Diamond....22
Figura 5 – Gráfico do Flow State....26
Figura 6 – Representação gráfica dos Quatro Elementos Básicos....27
Figura 7 – Representação gráfica dos estados de vida do Mario em Super Mario World (1990)....29
Figura 8 – O Colar de Pérolas....32
Figura 9 – Jornada do Herói....33
Figura 10 – Jogo The Legend of Zelda: The Wind Waker (2002)....35
Figura 11 – Jogo Medal of Honor: Allied Assault War Chest (2002)....35
Figura 12 – Representação de Weenies em Journey (2012) e nos estúdios de Hollywood....39
Figura 13 – Weenies nos estúdios de Hollywood....39
Figura 14 – Representação de como utilizar a luz para guiar o jogador em Uncharted 4 (2016)....41
Figura 15 – Representação de gray box....42
Figura 16 – Cenário externo de Journey (2012)....43
Figura 17 – Cenário interno de Journey (2012)....43
Figura 18 – Objeto em 3D e sua UV em 2D....52
Figura 19 – Chá de Labrador....53
Figura 20 – Chá de Labrador 3D....53
Figura 21 – Chá de Labrador com baixo e alto número de polígonos....54
Figura 22 – Chá de Labrador montada....55
Figura 23 – Grupos da planta Chá de Labrador....55
Figura 24 – Tronco e galhos da árvore.....56
Figura 25 – Galhos com folhas antes e depois do bake....57
Figura 26 – Arvore finalizada....58
Figura 27 – Modelo de baixa fidelidade da pedra Monolito....59
Figura 28 – Modelo de alta fidelidade da pedra Monolito....60
Figura 29 – Painel do Script....61
Figura 30 – Pedra de alta e baixa fidelidade....62
Figura 31 – Artes conceituais do personagem....63
Figura 32 – Visão frontal do modelo do personagem....64
Figura 33 – Visão lateral do modelo do personagem....64
Figura 34 – Visão frontal da máscara do personagem....65
Figura 35 – Visão lateral da máscara do personagem....65
Figura 36 – Configurações da Blueprint do bloco....67

Figura 37 – Configuração da ação a ser tomada 68
Figura 38 – Configurações da blueprint do emissor 69
Figura 39 – Configuração precedente a função Cast Light 70
Figura 40– Configuração para que o laser não seja infinito 71
Figura 41 – Configuração da associação do material ao laser 71
Figura 42 – Configurações da blueprint do receptor 72
Figura 43 – Configuração do contato do laser com objetos 72
Figura 44 – Configuração caso o laser reconheça o espelho 73
Figura 45 – Configuração caso o laser reconheça o receptor 73
Figura 46 – Arte conceitual do nível do jogo 74
Figura 47 – Seção A do jogo construída no gray box 75
Figura 48 – Primeiro obstáculo do jogador 76
Figura 49 – Seção tutorial sobre arrastar objetos 77
Figura 50 – Representação do uso de plataformas e dos objetos que podem ser movidos 78
Figura 51 – Seção tutorial sobre a ativação de plataformas 79
Figura 52 – Imagem do ponto central do mapa de Shards 80
Figura 53 – Primeira metade da seção B do mapa de Shards 81
Figura 54 – Apresentação da mecânica dos lasers 82
Figura 55 – Apresentação dos espelhos para o jogador 83
Figura 56 – Puzzle dos espelhos completo 83
Figura 57 – Apresentação da primeira metade do caminho para seção D 84
Figura 58 – Plataformas flutuantes da seção C 85
Figura 59 – Terceiro puzzle com lasers, localizado na seção C 86
Figura 60 – Plataformas que levam à segunda metade da seção C 87
Figura 61 – Quarto Puzzle com lasers, localizado na seção C 88
Figura 62 – Quinto e último Puzzle com lasers, localizado na seção C 89
Figura 63 – Ativação do primeiro receptor 90
Figura 64 – Ativação do segundo e ultimo receptor 91
Figura 65 – Representação da câmera do jogador antes de ativar o receptor 92
Figura 66 – Representação da câmera do jogador depois de ativar o receptor 93
Figura 67 – Cenário antes das mudanças 93
Figura 68 – Cenário depois das mudanças 94
Figura 69 – Comparação entre o antes e depois da reconstrução das salas 95
Figura 70 – Disposição das salas da seção B antes da reorganização 96
Figura 71 – Disposição das salas da seção B depois da reorganização 96
Figura 72 – Disposição das salas da seção C antes da reorganização 97
Figura 73 – Disposição das salas da seção C depois da reorganização 97
Figura 74 – Plataformas da seção A antes do ajuste 98

Figura 75 – Plataformas da seção A depois do ajuste 99
Figura 76 – Landscape esculpido com base na disposição dos objetos no nível Gray box.....100
Figura 77 – Edificações reconstruídas com base nos edifícios do gray box.....101
Figura 78 – Volumes invisíveis que delimitam por onde o jogador pode se mover 102
Figura 79 – Aplicação das texturas de neve e terra.....103
Figura 80 – Disposição das árvores no cenário.....104
Figura 81 – Contraste entre a forma das pedras de basalto e pedras simuladas.....105
Figura 82 – Lua no cenário 106
Figura 83 – Cenário apenas com luzes naturais 107
Figura 84 – Cenário com luzes naturais e pontos de luz de apoio.....107
Figura 85 – Apresentação das mensagens de tutorial ao jogador.....108
Figura 86 – Cenário populado com flores e vegetação baixa.....109
Figura 87 – Pártículas de neve espalhadas pelo cenário.....110
Figura 88 – Visão frontal do personagem finalizado 111
Figura 89 – Visão posterior do personagem finalizado.....111
Figura 90 – Representação da visão do jogador durante a cutscene.....112

LISTA DE ABREVIATURAS E SIGLAS

3D – Três dimensões.
PC – *Personal Computer*
NES – Nintendo Entertainment System
PUBG – *Player Unkwown's Battlegrounds*
MMORPG – *Massive Multiplayer Online Role Playing Game*
FPS – *First Person Shooter*
2.5D – Duas dimensões e meia
UV – As coordenadas horizontal e vertical da imagem
2D – Duas Dimensões

SUMÁRIO

1 INTRODUÇÃO 17
1.1 OBJETIVOS 19
1.1.1 Objetivo Geral 19
1.1.2 Objetivos Específicos 19
1.2 JUSTIFICATIVA 20
1.3 METODOLOGIA 21
1.4 DELIMITAÇÕES 23
2 DESENVOLVIMENTO TEÓRICO 24
2.1 JOGO 24
2.1.1 O que é um jogo 24
2.2 GAME DESIGN 24
2.2.1 *Flow State* 25
2.2.2 Os quatro elementos base 26
2.2.2.1 Mecânicas 27
2.2.2.2 História 31
2.2.2.2.1 Jornada do Herói 32
2.2.2.3 Estética 34
2.2.2.4 Tecnologia 36
2.3 LEVEL DESIGN 37
2.3.1 O Level 37
2.3.2 Gray Box 41
2.3.3 Level design como mecânismo de ensino 44
3 DESENVOLVIMENTO 46
3.1 A IDEIA 46
3.2 TECNOLOGIA DESENVOLVIDA 47
3.3 HISTÓRIA DESENVOLVIDA 48
3.3.1 Resumo 49
3.4 ESTÉTICA DESENVOLVIDA 49
3.5 MECÂNICA DESENVOLVIDA 50
3.6 CRIAÇÃO DE ASSETS 51
3.6.1 Vegetação Baixa 52
3.6.2 Árvores 56
3.6.3 Pedras e Ruínas 58
3.6.3.1 Pedras Esculpidas 59
3.6.3.2 Pedras Simuladas 60
3.6.4 Personagem 62
3.6.5 Animações 66
3.7 CRIAÇÃO DAS MECÂNICAS 73
3.8 DESENVOLVIMENTO DO LEVEL DESIGN 73
3.8.1 Nível para Testes 74
3.8.2 Desenvolvimento do Gray Box 74
3.8.2.1 Primeira Iteração 75

3.8.2.2 Segunda Iteração 91
3.8.3 Montagem Final 99
4 CONCLUSÃO 113
REFERÊNCIAS 115

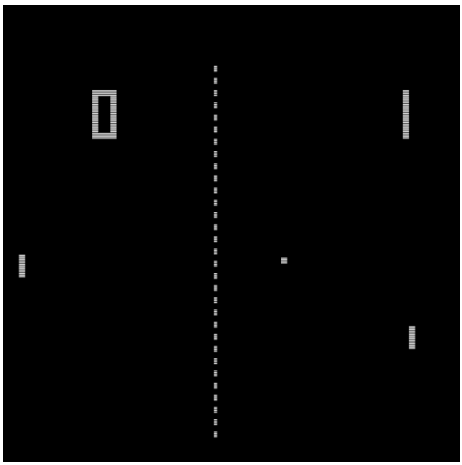
INTRODUÇÃO

O Ato de criar um jogo é algo que quase todos podemos entender. Quando criança, a maioria das pessoas em algum ponto acaba criando uma inovação no pega-a-pega, pique-esconde ou até mesmo desenvolvendo um jogo completamente novo com suas regras e sistemas. Criar jogos não é uma tarefa nova, porém os jogos digitais são relativamente recentes na história dos *games*, datando aproximadamente a 1950 com o desenvolvimento dos primeiros jogos eletrônicos, porém se popularizando na era dos fliperamas em 1970 com títulos como PONG. Durante o desenvolvimento dos primeiros jogos, a maior parte do trabalho era feito por apenas uma pessoa, o programador, que na época também desenvolvia os níveis e os gráficos a serem utilizados para representar o jogo. Com o passar dos anos, jogos começaram a se tornar mais complexos com gráficos melhores, trilhas sonoras exclusivas e narrativas próprias, influenciando o mercado a trabalhar com profissionais especializadas em diferentes setores no desenvolvimento de um jogo.

Durante a época de criação dos primeiros jogos, níveis eram tratados como uma plataforma para que o jogo acontecesse na qual o nível de complexidade da fase nem sempre era diretamente relacionada com a dificuldade do nível que você estivesse. Com a evolução dos gráficos em jogos de formas para *sprites*[1] cada vez mais complexos, artistas e *level designers* possuíam a oportunidade de ambientar o jogador cada vez mais, fazendo-os sentir que a plataforma em que estavam se movimentando fazia realmente parte do mundo, dando um peso maior para as interações entre o jogador e os níveis.

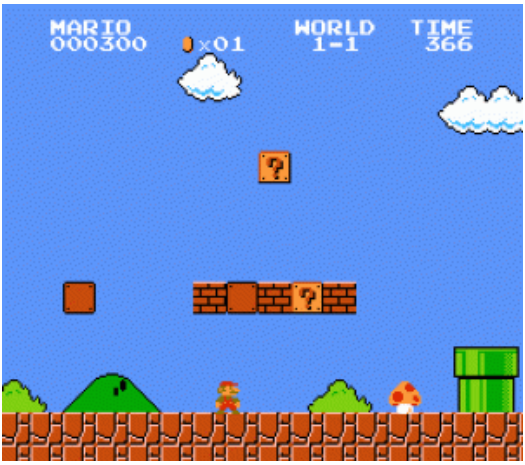
[1] Objetos bi ou tridimensionais que se movem pela tela.

Figura 1 - Jogo Pong para Atari.



Fonte: www.lifewire.com

Figura 2 - Super Marios Bros para o Nintendo Entertainment System (NES).



Fonte: www.retrogames.cz

Os jogos da época, porém, ainda estavam limitados a duas dimensões de movimento permitindo aos jogadores moverem-se nos

eixos X e Y. Com a evolução das placas gráficas e técnicas de renderização, foi possível a adição de mais um eixo de movimentação para os jogadores, o eixo Z, que trazia profundidade para o mundo dos *games*.

O advento do 3D nos jogos foi o que cimentou a posição de *level designer* dentro das empresas, um cargo especificamente dirigido às pessoas que criam o mundo físico do jogo, seus *puzzles*, desafios e plataformas. A profundidade permitiu que desenvolvedores criassem mundos virtuais com fortes semelhanças à nossa realidade, fortalecendo nossa relacionabilidade com eventos que acontecem nos mundos virtuais. Um dos primeiros jogos a fazer isso conhecido como DOOM, abordou a criação de níveis de forma hiper-realista, usando bases militares reais como referência direta para a criação de seus níveis, porém essa é uma técnica que pode não funcionar tão bem devido ao fato de que estruturas e locações da realidade não foram construídas com o intuito de serem "jogadas" sendo assim, o fator funcionalidade presente nelas, pode acabar interferindo com o fator diversão. Essa é uma das diversas abordagens que artistas utilizam até hoje para criar níveis para jogos e também será aplicada no desenvolvimento do jogo *Shards*, um *game single-player*[2] com foco em *puzzles* e elementos de plataforma. O projeto a ser desenvolvido busca aplicar os conceitos de *game* e *level design* e desenvolver um protótipo da primeira fase do *game Shards*, jogável para a plataforma PC.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Desenvolver o nível do jogo.

1.1.2 Objetivos Específicos

Desenvolver o protótipo de um jogo 3D com foco nos princípios de *game* e *level design*.

Desenvolver um conjunto de *assets*[3] para a construção do jogo e de um portfólio profissional.

Desenvolver o personagem.

Analisar os conceitos de *level design* e usá-los ao favor do usuário.

[2] Jogo jogado por apenas uma pessoa.

[3] Todo objeto ou item utilizado para o desenvolvimento do jogo.

1.2 JUSTIFICATIVA

A área de *games* é uma das que mais cresce no mundo inteiro, resultando num total de 2,5 Bilhões de jogadores mundiais (*The European Mobile Game Market*, 2016) e ocupando um espaço de 51,6% (European Retailers Association, 2018) no mercado do entretenimento, dominando a área. Os números continuam a impressionar quando o assunto se trata do Brasil, classificando-se como o segundo maior consumidor de *games* na América Latina e o 13º no mundo todo segundo a empresa especialista em análises no mercado de jogos, *Newzoo*.

Esse crescimento em boa parte, deve-se ao fato de que jogos são extremamente versáteis, desde os Óculos de realidade virtual até os *smartphones*, *games* facilmente se adaptam às novas formas de tecnologias podendo ser reaproveitados em formas de relançamentos como o famoso PUBG, que está disponível em 5 plataformas diferentes (Android, Playstation 4, Xbox One, iOS e Microsoft Windows) ou até mesmo reimaginações do mesmo produto como *Final Fantasy XV*, que possui uma versão para consoles, computadores e uma alternativa para celulares. Essa ideia é reforçada pelo crescimento do mercado de jogos em tecnologias emergentes como *smartphones* e *tablets* que somadas representam 45% do mercado mundial de *games*, se sobrepondo aos consoles de mesa que sempre dominaram o mercado.

Figura 3 *- Mercado mundial de games por segmentos.*

Fonte: www.newzoo.com (2019)

Tendo em vista o crescimento do mercado de jogos e as diversas oportunidades que ele apresenta, chegou-se à conclusão de que o momento é oportuno para se perseguir uma carreira como desenvolvedor de jogos digitais.

1.3 METODOLOGIA

A metodologia a ser utilizada no projeto, será a *double diamond*, criada em 2004 pela *Design Council*, se tornou referência de processo para *designers* e pessoas que procuram encontrar a solução através do *design thinking* para algum problema. A metodologia consiste de 4 etapas, sendo elas:

- *Discover*: É a primeira etapa da expansão do diamante e ela é definida por pesquisa. É aqui que o designer irá aglomerar um conjunto de ideias que tenham relação ao projeto a ser desenvolvido. Assim como no *brainstorming*[4], dentro desta

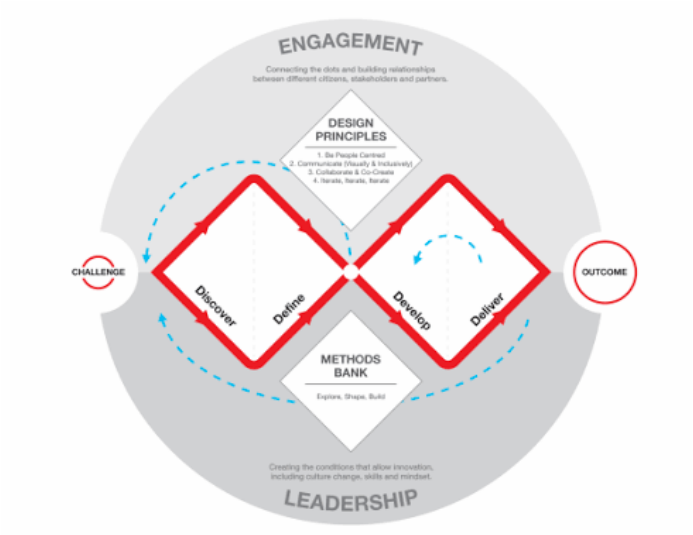
[4] É uma dinâmica em grupo utilizada para a geração de solução para problemas específicos.

etapa o importante são as alternativas levantadas, não importando o quão divergentes elas são.

- *Define*: Aqui as ideias começam a convergir. Filtra-se o resultado da pesquisa da etapa *discover* e procura-se focar em um caminho, limitando o projeto ao que se parece viável ou necessário.
- *Develop*: A segunda etapa da expansão, visa desenvolver um protótipo em cima das ideias definidas na etapa *define*. É na terceira etapa que acontecem a maior parte dos testes de novas ideias e soluções de problemas.
- *Deliver*: *Deliver* consiste do refinamento do produto desenvolvido nas anteriores etapas e a sua finalização.

A metodologia *double diamond* foi selecionada pelo fato de se alinhar a forma que jogos normalmente são produzidos, além de ser simples e podendo ser utilizada para resolver os mais diversos problemas nas diversas etapas de concepção do projeto.

Figura 4 *- Metodologia Double Diamond*



Fonte: www.designcouncil.org.uk

1.4 DELIMITAÇÕES

O projeto a ser desenvolvido visa criar o protótipo de um *game* 3D dentro da *engine* de jogos *Unreal Engine 4*. O protótipo será criado por 2 pessoas, mantendo uma divisão clara entre as tarefas e responsabilidades de cada um dos autores. Por conta disso, não serão estudados os tópicos referentes a texturização de objetos, *uv mapping*, iluminação, rendering, interfaces e som durante a criação deste projeto.

A partir dessas delimitações, serão abordados tópicos como modelagem dos objetos e personagens a serem utilizados no projeto, com foco no balanço entre a otimização e a qualidade visual.

Será também desenvolvido o nível do jogo, dentro da Unreal Engine 4 com base nos conceitos de game e level design.

A programação do jogo será desenvolvida usando os sistemas de blueprints da UE4, e terão um caráter extremamente simples, tendo em consideração o fato dos autores do protótipo não possuírem conhecimento algum sobre programação.

2 DESENVOLVIMENTO TEÓRICO

2.1 Jogo

2.1.1 O que é um Jogo

A maioria de nós sabe o que é um jogo de forma intrínseca, porém explicar esse conceito em poucas palavras parece ser algo complicado, de forma que no mundo do *game design* não existe uma definição padrão para determinar o que é um jogo, e ela varia entre autores. Em seu livro *Game Design: A Book of Lenses*, Schell (2014) procura criar sua própria definição a fim de entender a estrutura que define um jogo, tornando mais fácil o ato de categorizá-lo e permitindo um estudo mais a fundo sobre esse assunto. "Um jogo é uma atividade de solução de problemas da qual você aborda de forma lúdica." (Schell, 2014, p.47 tradução nossa).

A partir de sua definição e do estudo de diversos outros autores, Schell (2014) chega a conclusão dos 10 mais importantes fatores que categorizam um jogo.

1. Jogos são jogados voluntariamente.
2. Jogos possuem objetivos
3. Jogos possuem conflito.
4. Jogos possuem regras.
5. Jogos podem ser vencidos e perdidos.
6. Jogos são interativos.
7. Jogos possuem desafios.
8. Jogos podem criar seu próprio valor interno.
9. Jogos cativam jogadores.
10. Jogos são sistemas formais, fechados.

2.2 *Game Design*

Desde os primórdios dos *games*, ser um *game designer* significava tomar decisões, "*Game Design* é o ato de decidir o que um jogo deve ser" (Schell, 2014, p.39 tradução nossa). Ao tomar decisões sobre o que um jogo deve ser, o *designer* procura determinar a experiência que ele quer que o usuário tenha, proporcionando o jogador com as ferramentas para mantê-lo entretido e não determinando de que forma o jogador deve se divertir. Para que o designer consiga guiar o usuário de forma prudente, é fundamental o conhecimento e experiência

para com as diversas nuances presentes dentro do *game* design, sendo uma delas o *level design*.

2.2.1 *Flow State*

O *flow state* ou estado do fluxo é um conceito estudado pelo psicólogo Mihaly Csikszentmihalyi que pode ser determinado como: "Um sentimento de foco completo em uma atividade, com um alto de nível de satisfação e realização." (Schell, 2014, p.138, tradução nossa). O *flow state* pode ser observado quando uma pessoa sente-se completamente entretida por certa atividade, desviando sua atenção de seus arredores e focando-se inteiramente na ação que ela performa. Esse sentimento é exatamente o que um *designer* de jogos procura invocar em seus consumidores, uma experiência que possibilita um afastamento do mundo real e uma imersão na experiência virtual.

Para que esse estado seja alcançado, Schell (2014) aponta alguns dos pontos principais necessários ao criar uma atividade que induza o usuário entrar no *flow state*.

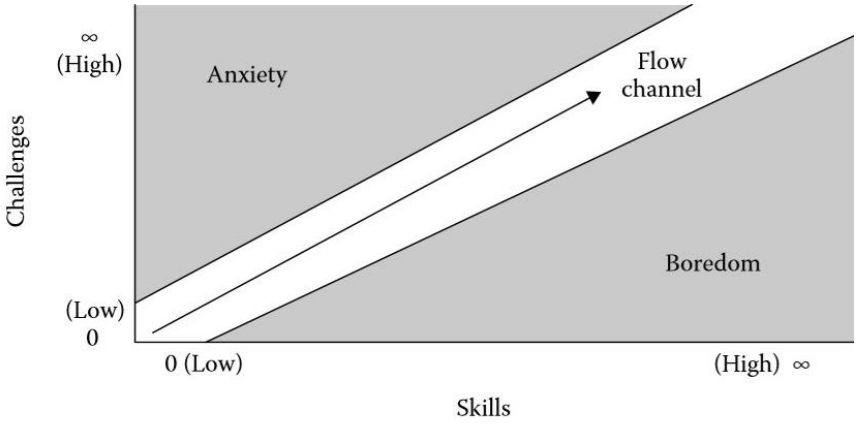
- Objetivos Claros: Jogos devem possuir objetivos claros ao serem criados, pois quando o jogador começa a duvidar se suas ações importam no contexto de progressão do jogo, ele acaba distraindo por pensamentos desnecessários.
- Sem distrações: Distrações fazem com que o jogador não preste atenção nas tarefas que importam e perca o foco. Segundo Schell (2014, p.139, tradução nossa): "Trabalho servil sem pensar faz com que a mente vague; apenas sentar e pensar pode tornar as mãos inquietas."
- *Feedback*[5] instantâneo/direto: Para cada ação tomada em um jogo, o jogador necessita uma confirmação de que ela realmente teve um efeito, caso contrário a espera nos torna ansiosos, ou focados em aspectos que subtraem da experiência do jogo.
- Constantemente desafiador: Desafiar o jogador é uma das diversas maneiras que *designers* nos mantém focados. Esses desafios porém, devem ser compostos de atividades que os jogadores sintam que possam

[5] Efeito retroativo de reação à um estímulo.

> completá-las. Se o desafio for grande demais, o jogador pode se sentir frustrado por não conseguir superá-lo e desistir. Por outro lado, se o desafio for fácil demais, ele pode se tornar entediado e buscar outra atividade para satisfazê-lo.

Manter um jogador dentro do *flow state* é uma atividade complexa, pois exige um tênue balanço entre a ansiedade e o tédio. Em jogos especialmente, é importante notar que enquanto um jogador progride, ele ganha um maior nível de experiência e familiaridade com a atividade que está exercendo, fazendo com que adversidades passadas tornem-se de certa forma triviais. Cabe ao *designer* então, providenciar um aumento gradual constante do nível de dificuldade dos desafios que ele irá enfrentar evitando que o jogador torne-se entediado, tendo cautela com o grau da dificuldade a ser imposta de modo que ela não cause o jogador a tornar-se demasiadamente ansioso.

Figura 5 – *Gráfico do Flow State*



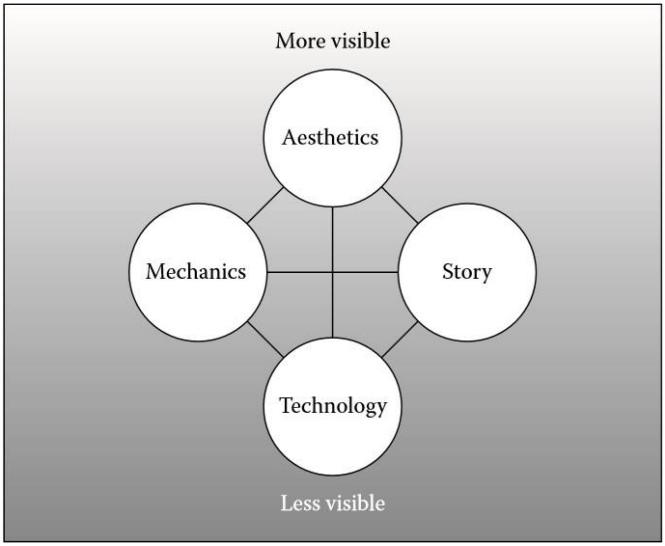
Fonte: *The Art of Game Design: A Book of Lenses* (2008).

Para que o *designer* possa manter o jogador dentro do *flow state*, é importante que ele conheça os elementos que compõem um jogo, permitindo assim que ele os manipule e tenha maior controle sobre a experiência que ele deseja que o jogador tenha.

2.2.2 Os quatro elementos Básicos:

Esses elementos não são necessariamente coisas que o jogador irá perceber ao jogar um jogo, alguns são mais obscuros do que os outros ao seu olhar, porém todos os elementos que compõem um jogo possuem um impacto análogo uns aos outros, de forma que o foco deve ser distribuído de forma uniforme entre eles pelo *game designer*, otimizando a experiência do jogador em todos os níveis.

Figura 6 *- Representação gráfica dos Quatro Elementos Básicos.*



Fonte: *The Art of Game Design: A Book of Lenses* (2008).

2.2.2.1 Mecânicas

As mecânicas representam o que um jogo realmente é. Quando não existem elementos de história, estética ou tecnologia, as mecânicas ainda estarão presentes determinando as possíveis interações permitidas no jogo. Elas variam entre mecânicas simples como um pulo à complexas com inteiros sistemas que determinam a economia de um MMORPG.

> Esses são os procedimentos e regras do seu jogo. Mecânicas descrevem o objetivo de seu jogo, como os jogadores podem e não podem tentar alcançá-lo, e o que acontece quando eles tentam. Se você comparar jogos a experiências de entretenimento mais lineares (livros, filmes, etc.), você irá notar que experiências lineares envolvem tecnologia, história e estética, elas não envolvem mecânicas, porque são as mecânicas que fazem um jogo ser um jogo. (Schell, 2014, p.51 e 52, tradução nossa)

Para um melhor entendimento e praticidade na hora de criar um jogo, Schell (2014) criou uma lista categorizando as principais e mais importantes características remetentes às mecânicas que compõem um jogo:

- Espaço: Não deve ser confundido com o espaço estético, mas sim a dimensão em que o jogo ocorre. A estética é então aplicada em cima dessa dimensão, reforçando certas características desejadas pelo *designer*.
- Tempo: Tempo pode ser usado de diversas maneiras, como mecânica central compondo um jogo como Braid, na qual o jogador deve resolver *puzzles* controlando o fluxo do tempo. O tempo pode também ser irrelevante em relação às ações do jogador em jogos baseados em turnos, como em Final Fantasy, na qual o inimigo não irá agir (o tempo estará basicamente congelado) até que o jogador determine sua ação primeiro. Outra forma de usá-lo, particularmente popular na época dos fliperamas, era como parâmetro do quão bom um jogador é, registrando o tempo que ele necessitou para completar certo objetivo.
- Objetos, atributos e estados: Objetos são os *assets* que compõem um jogo, personagens, vegetações, prédios são todos categorizados como objetos. Cada um desses objetos possui um ou mais atributos que funcionam como categorias de informação sobre um objeto, sua localização no espaço, a textura que ele apresenta, seu tamanho etc. Esses atributos são então controlados pelos estados. Usando o jogo Super Mario como exemplo, podemos identificar o personagem Mario como um objeto. O objeto Mario possui um atributo que representa sua “vida” e podemos observá-la como possuindo 3 estados

diferentes. Ela pode estar cheia, representada por Mario em sua Altura máxima, pela metade, representada por Mario com sua altura reduzida, ou então zerada, que é quando Mario morre. Todos os objetos de um jogo possuem atributos, mas nem sempre esses atributos necessitam de estados.

Figura 7 *- Representação gráfica dos estados de vida do Mario em Super Mario World (1990)*

Fonte: O Autor.

- Ações: As ações podem ser definidas como o que o jogador pode fazer dentro do jogo. Existem 2 tipos de ações, as ações básicas e ações estratégicas. Ações básicas são as ações mais simples que o jogador pode executar como andar ou pular. Ações estratégicas por outro lado são ações que o jogador toma para alcançar um objetivo. Podemos tomar como exemplo um jogo que possui um sistema de habilidades ofensivas e defensivas. O jogador pode então decidir focar todo seu investimento em habilidades defensivas, pois o que ele almeja é poder receber uma grande quantidade de dano sem morrer. Ações estratégicas podem ser consideradas como uma forma de *gameplay*[6] emergente, nas quais o jogador utiliza de mecânicas que o jogo providencia para superar certos obstáculos não exatamente da forma que o *game designer* planejou. *Gameplay* emergente é um dos grandes fatores de rejogabilidade, permitindo experiências diferenciadas a cada jogatina.
- Regras: Regras basicamente são o jogo. Um jogo não pode existir sem as regras que o compõem, pois elas ditam tudo que é e não é possível dentre os diversos sistemas de um jogo. Além de determinar as possibilidades, as regras determinam também os objetivos. Segundo Schell (2014), os objetivos do jogo precisam ser concretos, alcançáveis e recompensadores, de outra forma os jogadores irão perder o interesse ou nem mesmo tentar. Objetivos devem ser claros e instigantes.
- Habilidade: A habilidade foca na aptidão necessária que um jogador precisa ter para poder experienciar o jogo. Elas podem ser divididas em 3 categorias, habilidades físicas, mentais e sociais. Alguns jogos serão mais exigentes de certas habilidades do que outros. Futebol por exemplo requer que o jogador tenha uma habilidade física bem elevada, diferente de um jogo de xadrez, em que os movimentos físicos são mínimos porém requer um maior esforço das habilidades mentais do jogador. É importante saber quais habilidades e em que nível elas são necessárias para jogar um jogo por questões de balanceamento além de entender sobre qual foco o *designer* deseja que seu jogo tenha.
- Chance: Chance em jogos trata-se da incerteza e surpresa sobre certas ações que o jogador toma. Ela pode ser uma poderosa

[6] É a experiência do jogador enquanto interage com os sistemas que compõem um jogo

ferramenta para manter um jogador focado em seus objetivos. Um gênero popular de jogos nos dias atuais são os *Looter Shooters*, jogos em sua maioria FPS, baseados em matar uma grande quantidade de inimigos para conseguir equipamentos melhores, permitindo o jogador então matar inimigos mais fortes, conseguindo mais equipamentos e o ciclo continua. Porém, em sua grande maioria, a chance do jogador conseguir o almejado equipamento é extremamente baixa, exigindo por muitas vezes horas de dedicação por apenas um dos equipamentos. Esse é um balanço complicado para o *designer*, pois a aleatoriedade presente no jogo pode ser vista de forma tanto negativa ou positiva por seus jogadores. Se a chance for muito baixa, talvez os jogadores sintam que seja impossível adquirir o item e apenas param de tentar. Se ela for alta demais, eles podem considerar o jogo desprovido de desafio, e parem de jogar.

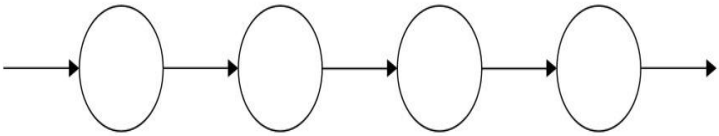
2.2.2.2 História

Com os jogos da atualidade se tornando cada vez mais cinematográficos a história tem possuído mais espaço no mundo dos jogos, desenvolvedoras como a Quantic Dream que mantém suas mecânicas relativamente contidas em comparação à outros jogos, estão ganhando notoriedade por seus produtos no qual o ponto chave de venda é a história. Diferente de filmes, no entanto, o jogador ainda possui um certo nível de controle (por muitas vezes artificiais) sobre a história. No mundo do *game design* 2 métodos predominam na indústria quando se trata de contar histórias. Eles são conhecidos como o Colar de Pérolas e A Máquina de Histórias.

> “História: Esta é a sequência de eventos que se desenvolvem em seu jogo. Ela pode ser linear e pré-determinada ou pode ser emergente e ramificada. Quando você possui uma história que quer contar através de um jogo, você deve escolher as mecânicas que irão simultaneamente fortalecer e permitir que a história aflore. Como qualquer contador de histórias você deve escolher a estética que ajude a reforçar as ideias da sua história e a tecnologia mais apropriada para a história que irá sair de seu jogo.” (Schell, 2014, p. 52, tradução nossa)

O Colar de Pérolas é assim chamado pela forma como é visualmente representado. É um meio utilizado para contar uma história não interativa, no sentido de que o jogador não irá influenciar a resolução da história, mas sim acompanhá-la, através do colar (o *gameplay*) até alcançar uma das pérolas (onde a história terá um momento de exposição). Com a evolução das tecnologias de captura de movimento e as técnicas de rendering, esse método tem sido refinado com o passar do tempo para criar incríveis experiências de *storytelling*[7], rivalizando aquelas de filmes, com jogos como Journey, The Last of Us e God of War. Esse método permite um ótimo balanço entre *gameplay* e *storytelling*, possibilitando ao jogador se envolver com o mundo ao seu redor e levantar certas questões que podem então ser saciadas através das pérolas.

Figura 8 *- O Colar de Pérolas.*



Fonte: *The Art of Game Design A Book of Lenses* (2008).

A Máquina de Histórias por outro lado é um método usado para criar jogos na qual a história é praticamente inexistente ou então jogos que disponibilizam seus jogadores com as ferramentas necessárias para que criem sua própria história. É um método especialmente popular em jogos do estilo *sandbox* como Minecraft ou The Sims, nos quais os jogadores podem ser comparados a deuses que possuem controle total sobre o mundo que os cerca.

2.2.2.2.1 Jornada do Herói

A jornada do herói também conhecida como o monomito é uma estrutura de narrativa na qual um herói parte em uma jornada, absorve o conhecimento que ela lhe proporciona e retorna para seu lar, como uma pessoa diferente de quando partiu. Joseph Campbell, autor do livro *The*

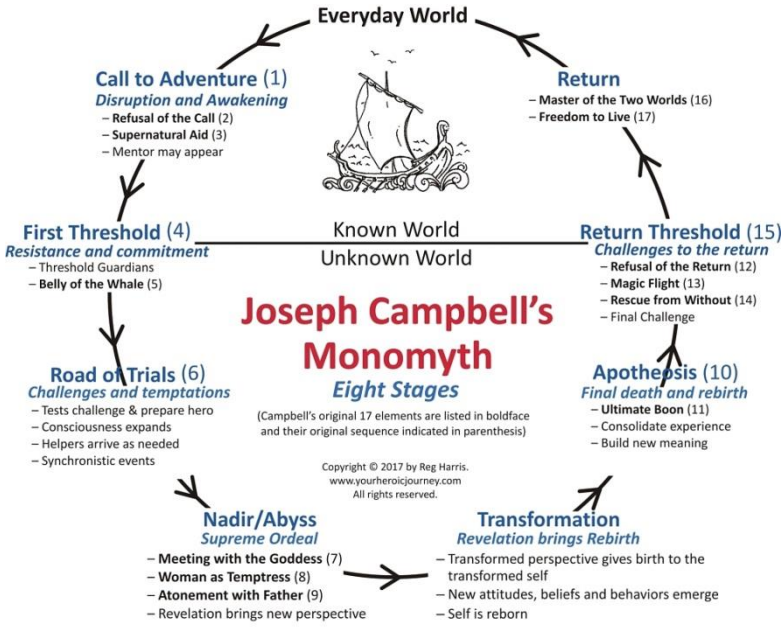
[7] O ato de contar uma história.

Hero with a Thousand Faces (1949), criou uma estrutura para que autores pudessem visualizar o monomito.

A estrutura do Monomito possui 17 etapas. Nem toda história utiliza todas as etapas ou da forma que foram propostas. Elas podem então ser divididas entre 3 pontos básicos.

- A partida: O herói parte para uma aventura, deixando o mundo que lhe é familiar para trás.
- A iniciação: O herói é confrontado com os desafios do novo mundo e aprende a superá-los.
- O retorno: O herói retorna para seu lar, com o conhecimento e a carga adquirida em sua jornada.

Figura 9 – *Jornada do Herói*



Fonte: www.*sfcenter.ku.edu*

A estrutura de histórias do monomito é particularmente popular em jogos e filmes, devido ao fato de que o jogador ou espectador experiência o mundo a partir dos olhos do protagonista da história ou de

alguém que o acompanhe. A história, portanto, não progride antes que os personagens principais tomem ação dentro de seus respectivos mundos, tornando a jornada do herói a estrutura ideal para contar essas histórias.

2.2.2.3 Estética

Quando falamos sobre a estética de um jogo, uma das primeiras coisas que pensamos sobre, são em seus gráficos. Jogos como Red Dead Redemption 2 que tomam proveito da tecnologia de ponta de suas plataformas para apresentar um visual extremamente realista para a era em que foram lançados. Porém, existe um argumento a ser feito sobre o quão bem esses gráficos irão resistir a passagem do tempo. Com a rápida evolução da tecnologia de renderização, os consumidores acabam tornando-se mais exigentes sobre o que consideram de fato gráficos realistas. É possível identificar esse mesmo elemento problemático em filmes antigos como Tron, (1982) que combinava efeitos de computação gráfica considerados de ponta com ação *live-action*[8], porém ao analisar tais obras a partir da lente de um consumidor atual, elas acabam não sendo tão impactantes. Uma das maneiras de superar esse obstáculo e tornar uma obra de certa forma atemporal é estiliza-la.

Podemos analisar uma comparação entre *The Legend of Zelda: The Wind Waker* (2002, Nintendo) e *Medal of Honor: Allied Assault War Chest* (2002, *TKO Software*), dois jogos desenvolvidos na mesma época, com diferentes objetivos em mente. Enquanto *Medal of Honor* procurava explorar a tecnologia da época para apresentar gráficos realistas de ponta, Zelda preferiu adotar um estilo nada realista, com cores vibrantes, formas cartunescas e um alto contraste. Ao observamos a comparação, é possível percebermos através de fatores como o número de polígonos e a baixa resolução das texturas, que são jogos desenvolvidos em outra época, porém, Zelda não passa a mesma sensação de ser um jogo datado por conta de seu estilo.

[8] Trabalho ou filme realizado por atores reais e não animados.

Figura 10 *- Jogo The Legend of Zelda: The Wind Waker (2002)*

Fonte: www.venturebeat.com

Figura 11 *– Jogo Medal of Honor: Allied Assault War Chest (2002)*

Fonte: www.gamefaqs.gamespot.com

Os gráficos de um jogo, porém, não compõem a estética sozinhos, eles são acompanhados pelo som. Em certas empresas como *That Game Company*, a música ou trilha sonora, pode acabar guiando

todo processo criativo ao desenvolver o jogo (*Designing Journey*, 2013), como aconteceu com o jogo Journey, segundo Jenova Chen, *game designer* do projeto.

Quando todos estes elementos, e as mecânicas conversam entre si e reforçam uns aos outros, eles acabam criando uma atmosfera que cerca o jogo. Segundo Schell (2014, p.387, tradução nossa): "A atmosfera é invisível e intocável, mas de alguma forma ela nos envolve, nos permeia e nos torna parte do mundo."

> Estética: Isso é o visual, som, cheiro, sabor e toque de seu jogo. Estética é um aspecto extremamente importante de game design já que tem a relação mais direta com a experiência do jogador. Quando você possui um visual, ou tom, no qual você quer que os jogadores experenciem e se tornem imersos, você deve escolher uma tecnologia que não apenas permitirá a estética ser entendida, mas sim reforçada e amplificada. Você deve escolher mecânicas que façam com o que o jogador sinta-se em um mundo que a estética definiu, e uma história com eventos que permitam a estética emergir no momento certo e ter o maior impacto. (Schell, 2014, p.52, tradução nossa)

2.2.2.4 Tecnologia

> Tecnologia: Nós não estamos exclusivamente nos referindo à "alta tecnologia" aqui, mas à qualquer material e interação que torne seu jogo possível como papel, lápis, moedas de plástico ou lasers de alta potência. A tecnologia que você escolhe para seu jogo o permite fazer certas coisas e proíbe outras. A tecnologia é essencialmente o meio no qual a estética sucede-se, as mecânicas irão ocorrer e pelo qual a história será contada. (Schell, 2014, p.52, tradução nossa)

A tecnologia é o meio pelo qual o jogo acontece. Em Super Marios Bros, por exemplo, podemos dizer que a tecnologia é o console Super Nintendo e a televisão que o reproduz, ou então no jogo Damas, o tabuleiro e suas peças. A ideia de tecnologia pode ser então dividida em duas categorias: fundamental e decorativa.

A tecnologia fundamental refere-se à tecnologia base existente para que um tipo de experiência possa existir. O Nintendo Switch, por exemplo, não poderia existir sem que as tecnologias de rastreamento de movimento existissem.

A tecnologia decorativa por outro lado, refere-se à melhorias aplicadas em conjunto com a tecnologia base, para criar uma experiência superior. Continuando a usar as tecnologias de rastreamento de movimento, podemos usar de exemplo o Xbox 360 e o Kinect. O kinect é uma câmera, vendida separadamente do Xbox, que quando instalada, permitia o usuário usufruir de certas aplicações no console através de movimentos que eram rastreados pela câmera. Um dos problemas que usuários encontravam com esse produto no entanto, era o campo de visão limitado da câmera do Kinect que forçava os jogadores a terem que se posicionar em pontos específicos em frente à câmera para terem seus movimentos captados. Já o nintendo switch consegue realizar a mesma ação de captar o movimento do jogador através dos controles do console, sem a necessidade de uma câmera, permitindo maior liberdade ao usuário.

2.3 *Level Design*

"O propósito básico do level design é interpretar as regras do jogo e as traduzir em um construto que facilite o ato de jogar." (Kremers, 2009, p. 18, tradução nossa).

Level design pode ser interpretado como uma derivação de *game design*, devido ao fato de que para desenvolver o mundo de um jogo, o *level designer* deve conhecer as regras que compõem esse mundo, e as regras que o compõem são criadas pelo *game designer*. Sua importância, porém não é reduzida por conta de sua posição hierárquica, pois ambas as atividades dependem uma da outra. Enquanto que o *game designer* procura criar as ferramentas pelas quais um jogo pode ser desenvolvido, o *level designer* às aplica, criando o espaço no qual o jogo irá acontecer e ensinando jogadores como desfrutar do que o jogo tem a oferecer.

2.3.1 O *Level*

Segundo Rogers (2010), mapas de jogos tridimensionais podem ser divididos em duas categorias: becos e ilhas.

Os becos criam uma experiência de *gameplay* mais direta, utilizando o nível de forma que ele guie o jogador ao seu objetivo final. Esses becos podem ser expansivos como em The Last of Us, onde os jogadores seguem um trecho linear que é de certa forma mascarado por conta das diversas possibilidades de movimento em um mesmo nível. Os becos podem também ser estreitos como em Portal, onde as escolhas de movimento do jogador são limitadas. Segundo Rogers (2010, p. 219, tradução nossa), as vantagens que os becos possuem ao serem criados são:

- "É mais fácil de colocar pontos de gatilho para a movimentação das câmeras quando você sabe onde e como o jogador irá entrar e se mover por um nível."
- "Você pode ser dramático com os movimentos de sua câmera para informar o jogador ou então realçar a ação e o drama."
- "Você pode remover os controles de câmera do jogo, permitindo o jogador concentrar-se apenas no gameplay."
- "Você pode criar eventos de jogo pré-programados, já que você sabe para onde o jogador está olhando."
- "É mais fácil coreografar o combate e outros eventos de jogo como armadilhas."
- "Gargalos podem ser criados para prevenir que os jogadores retrocedam no jogo."
- "O designer pode utilizar narrativa ilusória para contar a história do nível."

As ilhas por outro lado, tomam vantagem da liberdade que elas oferecem aos jogadores, permitindo que eles determinem como irão experienciar o jogo. Esse tipo de *design* é especialmente popular, e deu luz à um gênero específico de jogos, o *sandbox*. Segundo Rogers (2010), criar ilhas pode ser mais desafiador do que becos pois a liberdade dos jogadores é uma ferramenta que pode inutilizar certos elementos que o *designer* criou para o jogador, pois eles podem simplesmente evitá-los. Para que isso não aconteça frequentemente, Rogers recomenda o uso de *weenies*.

Weenie é um conceito utilizado em parques de diversões, que é considerado como um ímã visual, utilizando um elemento que se destaque, aplicado de forma que chame a atenção dos visitantes dos parques, guiando-os de uma área à outra. Em jogos, *weenies* se comportam de uma maneira similar, guiando o jogador de forma que ele encontre uma locação de interesse como uma área secreta, o próximo

caminho a ser tomado ou o objetivo final do nível ou até mesmo do jogo. *Weenies* podem ser compostos de diversos elementos presentes no jogo como um prédio, raios de luzes, montanhas, etc.

Figura 12 *- Representação de Weenies em Journey (2012) e nos estúdios de Hollywood.*

Fonte: www.chiragraman.com

Figura 13 – *Weenies nos estúdios de Hollywood*

Fonte: www.chiragraman.com

Apesar das diferenças conceituais entre becos e ilhas, o *designer* pode escolher criar o jogo com os dois em mente, criando seções de jogo que intercalam entre eles. Como exemplo podemos utilizar Skyrim. O jogo possui seções de *gameplay* presentes no *overworld*[9] onde o *gameplay* pode ser considerado como uma ilha, onde o jogador tem a liberdade total de escolha sobre onde ele quer ir ou as ações que deve tomar. Em contraste porém, existem cavernas, onde o jogador é confinado em um espaço apertado e linear com apenas um caminho para que ele possa completar essa seção, se alinhando mais fortemente com os elementos que constituem um beco. Segundo Rogers (2010, p. 223, tradução nossa), as vantagens que as ilhas possuem são:

- "Ilhas promovem a exploração, e encorajam os designers à preencherem os "espaços" com segredos, missões adicionais e objetivos."
- "As opções de jogo são expostas defronte ao jogador como um Bufê."
- "Jogos com veículos (como corridas e combate com carros) é melhor em um espaço aberto do que em becos estreitos."

Segundo Rogers (2010), ao criar um mapa o *designer* deve guiar o jogador pelo caminho que você quer que ele tome. A utilização de *weenies* é uma das formas de guiar o jogador, mas ela pode se tornar repetitiva ou previsível caso seja utilizada de forma exagerada. Rogers (2010, p.226, tradução nossa) sugere aos designers que:

> Utilizem geometria e luz para ajudar o jogador a continuar através do caminho. Jogadores são atraídos pela luz, enquanto eles tendem a evitar ou negligenciar lugares escuros. Você pode utilizar formas como linhas diagonais para atrair o olhar do jogador em uma direção específica.

[9] A área principal de um videogame que interliga todos seus níveis e locais.

Figura 14 *- Representação de como utilizar a luz para guiar o jogador em Uncharted 4 (2016)*

Fonte: www.gamasutra.com

Backtracking[10] pode ser um importante elemento de *gameplay*. Diversos estúdios utilizam dessa técnica para poderem criar um presságio do que há por vir, além de economizarem recursos ao criarem seus jogos e aproveitar um nível ao máximo. Podemos tomar como exemplo o jogo The Legend of Zelda: A link to the Past. As masmorras do jogo em sua maioria, permitem que o jogador alcance a porta que os levem a completar essa seção, assim que eles entram na masmorra, porém, elas estão sempre trancadas. O jogador deve então completar o resto da masmorra, e retroceder para poder destrancar a porta. Isso cria um objetivo claro na mente do jogador, que tem conhecimento de seu objetivo final e uma sensação de antecipação em relação ao que há por trás daquela porta.

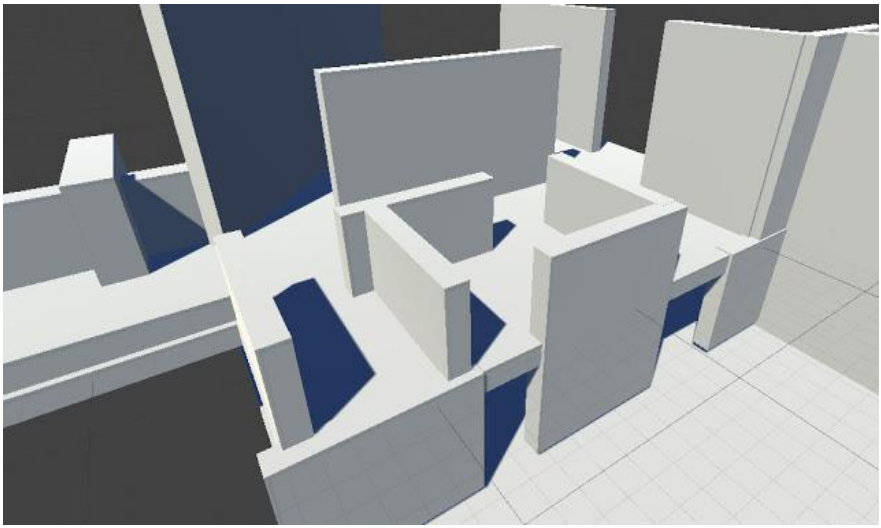
2.3.2 *Gray Box*

Gray boxing em *level design* é a prototipação de um nível. Ele consiste da criação do mapa com caixas cinzas, para que o *designer* possa focar nos elementos de *gameplay* e não ser distraído pela estética do jogo. Segundo Rogers (2010, p.233, tradução nossa): “Um nível gray box mostra a escala, tamanho, e relação entre elementos básicos

[10] Ato de retornar à um local já visitado em um jogo.

pertinentes à câmera e ao personagem. Ele é pivotal em determinar a escala,câmera e ritmo.”

Figura 15 *- Representação de gray box.*



Fonte: www.blog.radiator.debacle.us

“Divida seu jogo com grandes momentos e pequenos momentos” (Rogers, 2010, p.235, tradução nossa). Se seu jogo for composto apenas de pequenos momentos, ou momentos de calmaria, ele será entediante, da mesma forma que se apenas momentos grandes ou grandiosos estiverem presentes, o jogador poderá se cansar ou ficar ansioso. É importante que exista um balanceamento entre esses elementos para que o jogador possa entrar no *flow state*.

Essa variedade deve ser aplicada não apenas nos momentos de *gameplay* em si, mas também decorrer do nível e do jogo. O jogo pode alternar entre *gameplay* em espaços interiores como prédios, casas ou cavernas para o exterior em florestas, campos e cidades. Segundo Rogers (2010, p.235, tradução nossa): “Jogadores se sentem seguros em espaços abertos. Espaços apertados dão uma sensação de mistério e perigo”. Essas variações podem ser usadas não só alternando dado o interesse de variar, mas para exaltar certas características da história ou sensações que o designer deseja que o jogador sinta. Em Journey por

exemplo, os desenvolvedores utilizam a luz para refletir o estado de espírito do personagem. O jogo em sua maioria, acontece em mapas exteriores e bem iluminados sem a presença de inimigos, exaltando a liberdade do personagem, porém, ao entrar no subsolo, o jogador é confrontado com inimigos que ele deve evitar, os desenvolvedores então exaltam a sensação de perigo e mistério criando um mapa mais apertado e escuro.

Figura 16 *– Cenário externo de Journey (2012)*

Fonte: www.alexaraycorriea.wordpress.com

Figura 17 – *Cenário interno de Journey (2012)*

Fonte: www.mobygames.com

Em 3D, uma das variações necessárias ao criar níveis pode ser atribuída à verticalidade. Poucos lugares do mundo são planos. Ao observar os ambientes presentes em nosso planeta é possível perceber

que eles são permeados por escadas, morros, rampas etc. Ao criar verticalidade em um mundo virtual, o *designer* passa a torná-lo mais natural, além disso, segundo Rogers (2010, p.236, tradução nossa): "Quando um jogador sobe ou escala, eles sentem que estão progredindo e indo em direção à um objetivo".

Ao criar um jogo, é importante estabelecer uma linguagem visual, para que o jogador entenda onde ele pode e não pode ir. Pedras, arbustos ou árvores são recursos comuns utilizados para indicar que um jogador não pode passar por determinado lugar. Alguns jogos como os da série Grand Theft Auto, criam seus mapas em ilhas cercadas por água, dando uma sensação natural de fronteira para o jogador. As infames barreiras invisíveis funcionam também para o propósito de impedir o jogador de entrar uma certa área, porém são consideradas um recurso barato que quebra a sensação de liberdade do jogador, tornando o mundo menos real ou tangível.

2.3.3 *Level design* como um mecanismo de ensino

"Um bom *level design* ensina o jogador como jogar e aproveitar o jogo." (Kremers, 2009, p.26, tradução nossa)

O conceito de que *level design* ensina os jogadores a como jogar o jogo é de extrema importância. É através do *level design* que os jogadores são apresentados às regras que compõem um jogo. Segundo Kremers (2009, p.27, tradução nossa): "Bom *level design* não apenas ensina ao jogador as regras do jogo, mas também permite que o jogador use essas regras de forma que elas sejam recompensadoras e divertidas." É importante então ter em mente que o *level designer* não irá apenas ensinar aos jogadores como utilizar as mecânicas presentes no jogo, mas também criar situações em que elas poderão ou deverão ser usadas para poder progredir no jogo.

As formas que o *designer* pode ensinar o jogador sobre o jogo, são variadas em relação à seu uso e contexto. É improvável que sejam criadas regras sobre como ensinar o jogador, que funcionem para todos os jogos tendo em vista que dentro do mundo dos jogos, existem diversos gêneros e plataformas nada parecidas umas com as outras. Kremers, porém sugere algumas abordagens que podem ser utilizadas em diversos casos:

Ensino através do exemplo prático: é recomendado que ao introduzir um conceito ou mecânica nova ao jogador, que ele possa testá-la assim que possível.

Reforço positivo: Recompense os jogadores por completarem um desafio, pois ao ensinar os jogadores que existem consequências positivas por superar certas dificuldades, eles se tornarão ansiosos para o próximo desafio.

Prepare o jogador de forma justa: Tenha certeza de que os jogadores possuem o conhecimento e as habilidades necessárias antes de fazer com que eles necessitem desse conhecimento. Ensine sobre novas mecânicas aos seus jogadores em ambientes seguros, nos quais eles podem falhar sem grandes consequências.

3. DESENVOLVIMENTO

3.1 A Ideia

Inicialmente, a proposta do projeto *Shards* era ser um jogo *side-scroller*[11] com foco narrativo, desenvolvido na vista 2.5D com elementos de exploração, plataforma e *puzzle*. A partir dessa ideia, se deu início a primeira etapa da metodologia *double diamond*, na qual foi feita uma coleta de dados e referências sobre jogos que possuíssem uma proposta similar para que a solidificação da ideia ocorresse. Os 3 principais títulos analisados durante essa etapa foram: Trine 2 (2011), Limbo (2010) e Ori and The Blind Forest (2015).

Apesar do estilo de plataforma *side-scroller* em 2.5D apresentar fortes argumentos para o desenvolvimento de um jogo que utilize elementos de plataforma, ele acaba limitando o jogador no quesito exploração, devido ao fato do movimento estar restrito em boa parte a duas dimensões. Por conta de umas das principais ideias conceituais do jogo, que se relaciona à narrativa, na qual o jogador iria descobrir elementos da história do jogo não apenas através do progresso que ele obteve, mas também ao explorar o mundo do jogo, foi decidido reavaliar a proposta de utilizar uma câmera 2.5D e considerar a utilização do 3D como um todo, com uma câmera em 3ª pessoa para uma melhor visualização dos *puzzles*.

Foi dado início a mais uma coleta de dados com foco em jogos inteiramente 3D. Na pesquisa foram então identificadas as obras que mais se alinhavam com a intenção do projeto: Journey (2012), Rime (2017) e The Legend of Zelda: Breath of the Wild (2017).

A partir dos títulos identificados, foi feito uma análise de elementos que poderiam servir como inspiração para o desenvolvimento do jogo *Shards*.

Journey principalmente foi uma das obras das que mais influenciaram as ideias centrais do jogo. Ele parte da ideia de utilizar o jogo como plataforma para criar uma experiência emocional mais intensa do que normalmente se encontra na indústria. O jogo possui uma baixa quantidade de mecânicas, tendo destaque para a mecânica central da qual o *gameplay* é baseado em torno, que consiste em voar e planar.

Rime é um jogo de aventura e *puzzle* no qual o jogador guia o personagem pelo cenário e avança no jogo através da solução de *puzzles*

[11] Um jogo onde a câmera se posiciona de forma lateral, e os personagens em sua maioria podem se mover apenas da esquerda para direita.

relativos ao ambiente em que ele se encontra, utilizando elementos do cenário para completá-los. A forma da qual Rime constrói e apresenta os *puzzles* em conjunto com a ideia de uma mecânica central utilizada para mais de um propósito de journey, foram a base da qual as mecânicas do jogo *Shards* foram construídas sobre.

Por fim, Legend of Zelda: Breath of the Wild após uma pequena área de introdução às principais mecânicas do jogo, da total liberdade para o jogador, permitindo que ele explore os cenários à seu bel prazer, impondo pouquíssimas restrições, ocasionando até mesmo na possibilidade do jogador completar o jogo sem experienciar metade do que ele pode lhe oferecer. Uma tarefa difícil mas, não impossível. O jogo sugere que a exploração porém, traz frutos que facilitam a jornada do usuário, instigando-o a não fazer um caminho linear. Essa forma de nos introduzir ao mundo influenciou diretamente a criação do primeiro nível de *Shards*.

3.2 Tecnologia desenvolvida

Começa então a segunda etapa da metodologia *double diamond*, na qual foram definidos diversos aspectos que futuramente constituiriam o projeto. Para que o desenvolvimento de um jogo digital seja possível, é necessário uma engine de jogos, um programa que abstrai diversos aspectos da produção como a criação de um sistema de simulação de física, animações, colisões etc, além de um motor gráfico que permite a renderização de objetos em 2D ou 3D. Na atualidade, existe uma grande variedade de engines disponíveis no mercado para desenvolvedores de jogos porém 2 se destacam, e são elas: Unreal Engine 4 e Unity. Apesar de serem utilizadas por empresas profissionais, são programas gratuitos, permitindo que os usuários experimentem com ferramentas de alto nível para o desenvolvimento de jogos sem uma barreira de custo. Isso possibilitou que a engine escolhida para o jogo fosse de fato, a melhor opção de acordo com as necessidades do projeto.

As 2 engines possuem uma grande comunidade ativa, da qual uma pletora de informação pode ser extraída para a criação de jogos, facilitando o processo de descobrimento e solução de dúvidas em relação aos programas.

Tendo em vista essas observações, podemos destacar que as maiores necessidades do projeto são: facilidade para programar eventos e mecânicas por conta da falta de conhecimento de programação do autor do projeto e um sombreador que permitisse liberdade ao definir o estilo visual do projeto.

Apesar da Unity, possuir um sombreador superior ao da Unreal para os propósitos estéticos do projeto, a Unreal Engine possui um sistema de programação através de *blueprints*. Segundo a documentação de uso da Unreal Engine 4, o sistema de *blueprints* é: “ O sistema de *scripting* visual da Unreal Engine é um sistema completo de roteiro de *gameplay* baseado no conceito de usar uma interface baseada em nós, para criar elementos de jogo de dentro do editor da Unreal.” Esse sistema providencia o usuário com a habilidade de usar elementos e conceitos disponíveis para programadores de forma visual, transformando o processo de programação do jogo em uma etapa mais intuitiva, apesar de ainda necessitar que os usuários entendam o que a programação de fato faz. Além disso, o autor do projeto possui experiência prévia com a Unreal Engine 4, devido a projetos desenvolvidos anteriormente na universidade, portanto a Unreal Engine 4 foi escolhida como base para o desenvolvimento do projeto.

3.3 História desenvolvida

A história do jogo, se baseia em torno de uma premissa simples, criada em conjunto pelos autores do projeto que procura definir o mundo no qual o jogo se passa, determinando o que é e não é possível, as características que serão observadas pelo jogador e também servir de apoio para as decisões que se remetem a estética e as mecânicas.

O projeto utilizou o monomito como uma base estrutural para a criação da história e também os níveis que compõem o jogo, permitindo assim uma sincronia entre as etapas da jornada do herói e as fases pelas quais o jogador irá passar, reforçando o sentimento em destaque no momento.

A estrutura narrativa tomou como base o colar de pérolas, visando oferecer seções onde o jogador irá percorrer o mundo e levantar questões sobre as interações e elementos que ele observara (o colar) e então saciar suas dúvidas através de momentos de exposição da história (as pérolas). Os principais pontos da história serão expostos de forma que nenhum jogador poderia perdê-los, em pontos chaves dos níveis. Porém, existiram informações escondidas através dos níveis não necessárias para o entendimento da história mas que complementariam os pontos chaves expostos, tornando a experiência mais profunda através da exploração.

3.3.1 Resumo

Para uma contextualização dos objetivos visando ser alcançados no primeiro nível do jogo, foi criado um resumo dos primeiros momentos da história.

A história começa em um mundo familiar, repleto de árvores e vegetação nativa. O jogador toma controle de uma criatura misteriosa, desprovida de gênero ou etnia. Ela está vagando o mundo quando uma luz no céu chama sua atenção. A lua brilha e repentinamente fragmentos de seu brilho são lançados ao mundo. Um desses fragmentos caiu próximo a criatura, que se sente compelida a investigar. O caminho porém, é repleto de obstáculos que ela precisa superar para alcançar seu destino.

3.4 Estética Desenvolvida

Devido ao fato de um dos objetivos do projeto ser a construção de um portfólio profissional para os autores do projeto, foi decidido que os *assets* utilizados para o desenvolvimento do jogo seriam em sua maioria feitos em conjunto pelos mesmos, de forma que fosse possível refinar o conhecimento e habilidades de acordo com a área em que no futuro os autores pretendem almejar como profissão.

Para que isso pudesse acontecer, foram unidos elementos da história e da estética de modo que permitissem a criação de peças em sua maioria de caráter orgânico como árvores e plantas, das quais os autores possuem pouca experiência sobre.

Foi definido então o ambiente no qual o jogo se passaria, baseado na configuração geográfica do extremo norte do globo tendo como referência biomas como a tundra e taiga. O projeto porém não foi limitado à permanecer o mais realista possível, buscando reproduzir tais biomas até os mínimos detalhes. Foram tomadas liberdades artísticas, consideradas benéficas para a estética que o jogo visa obter.

Essas decisões foram tomadas a partir da história, visando reforçar sentimentos apresentados durante as etapas da jornada do herói. Além disso, por conta dos elementos fantasiosos presentes na história do jogo, foi decidido um afastamento dos gráficos realistas, optando por uma abordagem mais estilizada de forma que tornasse mais fácil para o jogador aceitar ideias que não condizem com a realidade como máquinas de luz ou plataformas flutuantes.

3.5 Mecânica Desenvolvida

Para a criação das mecânicas do jogo, o objetivo é que fossem simples, porém versáteis, de forma que possibilitassem basear o *gameplay* inteiramente em torno delas. Por conta do gênero do jogo ser plataforma e *puzzle*, eram necessárias mecânicas que permitissem uma colaboração entre os diferentes tipos de *gameplay* apresentados nos respectivos gêneros. Segundo Rogers (2010 p.9, tradução nossa), jogos em plataforma podem ser definidos como: "Um jogo plataforma, frequentemente apresenta um personagem mascote pulando (ou se balançando e pendurando) através de desafiadores ambientes de plataforma" e jogos baseados em puzzle por outro lado: " Jogos de puzzle são baseados em lógica e na resolução de padrões. Eles podem ser lentos, metódicos ou usarem coordenação entre olhos e mãos" (Rogers, 2010, p.11, tradução nossa). Para que isso fosse possível, foram utilizadas as mecânicas de movimento disponibilizadas pela Unreal Engine 4 e posteriormente implementadas 2 mecânicas centrais no jogo: arrastar blocos e um sistema de lasers e espelhos.

Durante a criação de um novo projeto dentro da *engine* de jogos Unreal, é possível escolher um dos diversos *templates*[12] que o programa disponibiliza ao usuário. Esses *templates* providenciam uma base na qual o usuário pode acrescentar ou retirar elementos necessários para seu projeto. Para a criação do jogo Shards, foi utilizado o *template* de terceira pessoa, que disponibiliza um personagem programado com animações e movimentos de navegação básicos como a habilidade de andar e correr em qualquer direção além de poder executar uma ação de pulo.

A mecânica de arrastar blocos foi implementada de forma que pudesse servir como uma ponte entre os gêneros de plataforma e puzzle. Ela é utilizada de forma híbrida, para a resolução de puzzles de forma que seja necessário que o jogador empurre blocos em locais específicos do cenário para que certos elementos sejam ativos, como portas e plataformas ou para a superação de obstáculos, quando o pulo do personagem não for o suficiente para que ele alcance um local, ele poderá então arrastar um bloco para então pular em cima dele e usá-lo de apoio para alcançar seu objetivo.

O sistema de lasers e espelhos foi introduzido para oferecer ao jogador uma maior variedade de puzzles, tornando o *gameplay* menos repetitivo. O sistema possui 3 elementos que o compõem, um emissor,

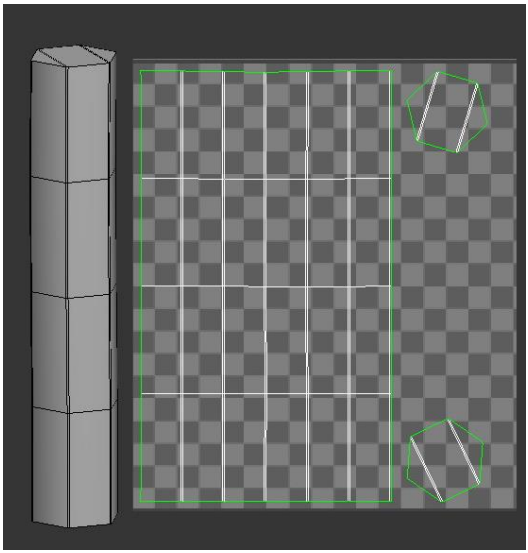
[12] É um modelo com uma estrutura pré-pronta.

um espelho e o receptor. O emissor é o ponto a partir do qual uma faixa de luz (laser) é emitida. É então o objetivo do jogador, manipular o emissor de forma que ele permita a luz que ele emite alcançar o receptor, que irá então completar o puzzle. Durante a criação dos puzzles, serão colocados obstáculos, de forma que o jogador não possa simplesmente apontar a luz do emissor ao receptor. Para superar esse problema, o jogador deve então utilizar os espelhos, elementos que conseguem refletir e redirecionar o laser para outra direção, permitindo uma maior variedade de puzzles que podem ser desenvolvidos.

3.6 Criação de *Assets*

Com o princípio da criação dos *assets*, foi dado início a 3ª etapa da metodologia *double diamond*. O foco na criação dos *assets* utilizados para construir o jogo era manter uma alta qualidade visual, porém, com um baixo custo de processamento. Para que isso fosse possível, foi amplamente utilizada uma técnica conhecida como *Texture Baking*. Uma das formas que o computador pode interpretar informações que compõem um objeto em 3D, é a partir de uv's. As uv's são planificações feitas a partir de um modelo em 3D que são representações diretas em 2D de um modelo tri-dimensional. Podemos então armazenar informações como cor, luz e sombra nas uv's de um objeto para que ela possa ser reproduzida em 3D. É possível também simular esses processos em tempo real, mas o custo de performance que eles necessitam em relação a objetos com a informação já armazenada é muito maior.

Figura 18 *– Objeto em 3D e sua UV em 2D*



Fonte: O Autor

Texture Baking consiste em transmitir as informações de um objeto em 3D para uma imagem (textura) em 2D. Dessa forma é possível criarmos 2 objetos que possuam uma forma parecida porém o objeto 1 com uma grande quantidade de polígonos (informação), e o objeto 2 com uma baixa quantidade. É então transferida a informação 3D que compõe o objeto 1 para uma imagem, convertida então em informação 2D. Essa informação pode então ser aplicada no objeto 2, para que a partir dessa textura, ele possa reproduzir um resultado final semelhante ao objeto 1 com um custo menor de processamento.

3.6.1 Vegetação Baixa

A vegetação baixa utilizada nos cenários do jogo composta por flores, musgos e grama foi desenvolvida dentro do 3ds Max, programa de modelagem tridimensional desenvolvido pela Autodesk. O processo utilizado para o desenvolvimento das plantas foi o mesmo que o exemplificado no item 3.2.

Primeiramente houve uma pesquisa sobre a flora que habita as regiões do ártico do planeta. Foram então reunidos exemplos baseados em suas cores, formas e tamanhos para que o jogo possuísse uma grande variedade na composição de sua flora, e não se tornasse repetitivo de forma que pudesse distrair o jogador. A fim de exemplificar o processo utilizado ao desenvolver as plantas, será utilizado a flor conhecida como Chá de Labrador.

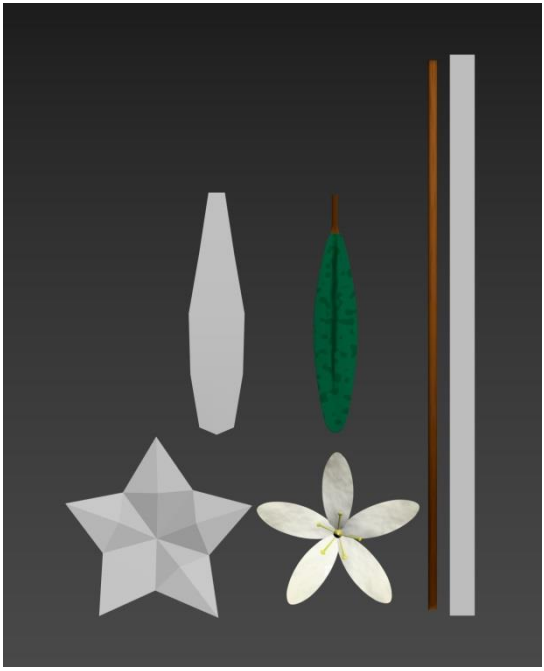
A partir da pesquisa realizada, foram reunidas referências para que fosse possível reproduzir a flor em um ambiente tridimensional de forma fiel. Com base nas referências foi então criado o modelo de alta fidelidade, com um grande número de polígonos.

Figura 19 e 20 - *Chá de Labrador e Chá de Labrador 3D respectivamente.*

Fonte: www.lgbotanicals.com e O Autor respectivamente.

Foram então criadas as texturas para a flor, de forma que possibilitasse o processo do *baking*. Com as texturas aplicadas nos objetos, foram criados modelos de baixa fidelidade, para os quais a informação seria transferida.

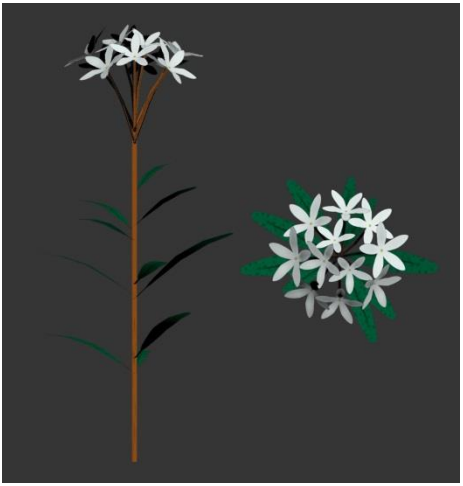
Figura 21 *– Chá de Labrador com baixo e alto número de polígonos*



Fonte: O Autor

Após o processo de *baking*, foram então “montados” os objetos de baixa fidelidade, conforme as referências. Depois de montados, foram criados grupos, nos quais uma grande quantidade de flores eram reunidas, de forma que tornasse o processo de distribuí-las pelo cenário do jogo, uma tarefa mais simples.

Figura 22 *– Chá de Labrador montada.*



Fonte: O Autor

Figura 23 *– Grupos da planta Chá de Labrador.*

Fonte: O Autor

3.6.2 Árvores

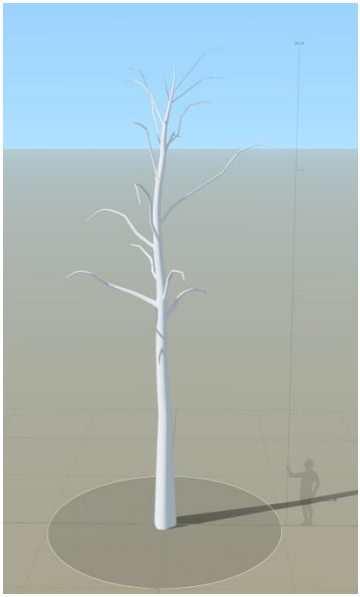
As árvores criadas para o jogo, foram desenvolvidas nos programas SpeedTree, 3ds Max e SpeedTree é um *software* de modelagem, programação e animação de vegetação, desenvolvido pela Interactive Data Visualization.

O processo para a criação das árvores foi semelhante ao utilizado no item 3.2.1, na qual foi feita uma pesquisa sobre o bioma da taiga e as espécies de árvores que habitam essa região para depois serem determinadas as referências a serem utilizadas.

Com base na pesquisa, foi iniciado o processo de modelagem do tronco da árvore dentro do programa SpeedTree. O programa tem por base nós, cada um deles com uma pletora de configurações para que o usuário possua uma grande liberdade para customizar seus modelos.

Foram então adicionados os nodes de “tronco” e “grandes galhos”, criando uma base para a qual as folhas seriam futuramente aplicadas.

Figura 24 – *Tronco e galhos da árvore.*



Fonte: O Autor

Para a criação das folhas foi utilizado o programa 3ds Max. Assim como as plantas, foram criados modelos de alta fidelidade, que foram então texturizados. Após a texturização, modelos de baixa fidelidade foram criados e as folhas então passaram pelo processo de *bake*, criando folhas individuais para serem utilizadas nas árvores.

O programa speedtree oferece opções de aplicar folhas individuais nas árvores porém, tendo em vista que o jogo utilizará dezenas de árvores simultaneamente, optou-se por não utilizar essa opção por conta do custo de processamento que é necessário para renderizar milhares de folhas em cena. Alternativamente, dentro do programa SpeedTree, criaram-se aglomerações de folhas em pequenos galhos, que serviram como um modelo de alta fidelidade para a criação de um plano que passaria pelo processo de *baking*, reduzindo o número de polígonos utilizados de 10,459 para 12.

Figura 25 – *Galhos com folhas antes e depois do bake*

Fonte: O Autor

Foram então aplicadas as folhas ao tronco, criando alternativas para evitar uma grande repetição de árvores no projeto.

Figura 26 *– Arvore finalizada.*

Fonte: O Autor

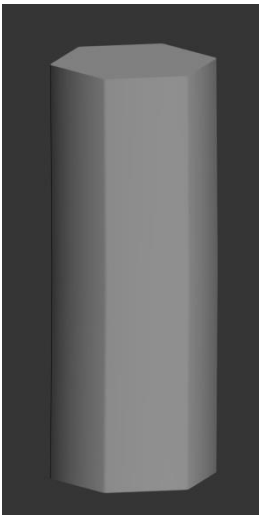
3.6.3 Pedras e Ruínas

As pedras desenvolvidas para o jogo, foram criadas nos programas 3ds Max e Zbrush. Para facilitar o entendimento do processo de criação das pedras, elas serão divididas em 2 grupos: pedras esculpidas e pedras simuladas.

3.6.3.1 Pedras esculpidas

As pedras esculpidas foram inicialmente desenvolvidas no programa 3ds Max, onde foram criados modelos de baixa fidelidade que futuramente seriam utilizados para o *baking*. Para fins de exemplo utilizaremos a pedra Monolito_1 para demonstrar o processo.

Figura 27 – *Modelo de baixa fidelidade da pedra Monolito*



Fonte: O Autor

Após a modelagem do modelo de baixa fidelidade, ele foi então exportado para o *software* Zbrush, um programa de escultura digital 3D. Zbrush é utilizado para criar modelos com uma alta quantidade de polígonos, através de um processo que se assemelha à escultura tradicional com argila. No Zbrush então, o modelo foi subdivido, aumentando a quantidade de polígonos que ele possui, possibilitando que fossem esculpidos os detalhes que compõem as rochas.

Figura 28 – *Modelo de alta fidelidade da pedra Monolito*

Fonte: O Autor

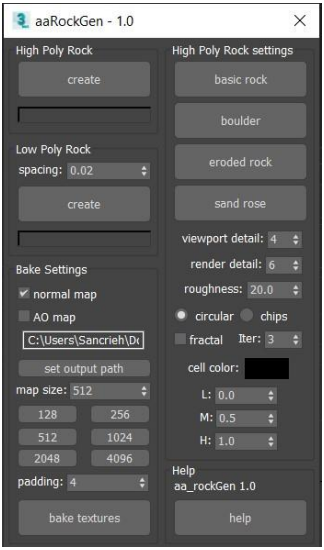
Terminando o detalhamento, o modelo é então exportado de volta para o 3ds Max onde em conjunto do modelo de baixa fidelidade é realizado o *baking*, finalizando o processo.

3.6.3.2 Pedras simuladas

As pedras simuladas foram criadas inteiramente dentro do programa 3ds Max, a partir de um *script*, disponibilizado no site scriptspot.com pelo usuário Alessandro Ardolino, criador do *script*.

O script disponibilizado permite a criação de pedras orgânicas a partir de um modelo base e uma série de configurações oferecidas através de um painel gerado pelo script. A rocha de alta fidelidade pode então ser gerada e configurada de acordo com as necessidades do usuário.

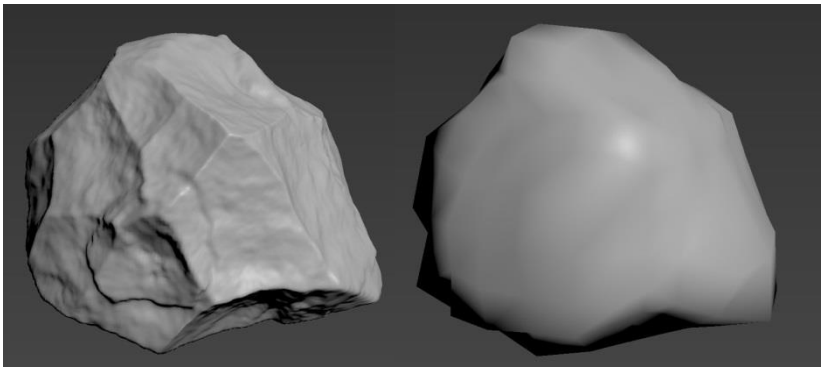
Figura 29 – *Painel do Script*



Fonte: O Autor

Após a configuração ser realizada, o usuário pode gerar uma pedra de baixa fidelidade, com base na pedra já gerada, automatizando o processo de modelagem.

Figura 30 – *Pedra de alta e baixa fidelidade*



Fonte: O Autor

Com a pedra de baixa fidelidade gerada, é possível então realizar o processo de *baking* disponível dentro do script, automatizando mais um dos processos necessários para a criação do objeto.

3.6.4 Personagem

O desenvolvimento do personagem teve uma grande inspiração em Journey (2012), de forma que o objetivo era criar um personagem humanóide, que não apresentasse características de gênero ou etnia, evitando que estereótipos fossem associados ao personagem antes do jogador interagir com o mesmo. Para reforçar essas ideias, optou-se por manter o rosto do personagem escondido.

A partir dessa proposta, foram desenvolvidas artes conceituais para uma melhor visualização das ideias que surgiram no decorrer do projeto, permitindo que fosse possível chegar a duas opções finais.

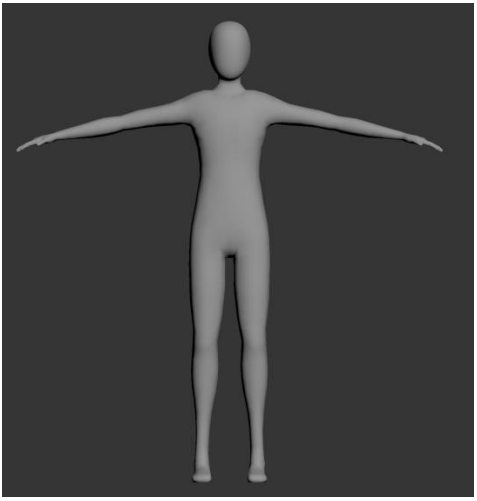
Figura 31 *– Artes conceituais do personagem*

Fonte: Coautor do projeto

Após analisarmos o nível de complexidade da construção dos dois personagens, e quem representaria melhor as ideias que a estética procurava transmitir, foi decidido pela utilização do conceito B como base para o desenvolvimento do personagem.

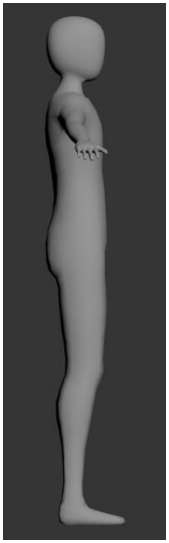
Para a modelagem do personagem, foi utilizado o programa 3ds Max. Inicialmente, foi realizada a modelagem do corpo do personagem, de acordo com as artes conceituais desenvolvidas. Pelo fato do rosto do personagem não aparecer no decorrer do jogo, foi optado por não modelá-lo, reduzindo o custo de performance do jogo.

Figura 32 *– Visão frontal do modelo do personagem.*



Fonte: O Autor

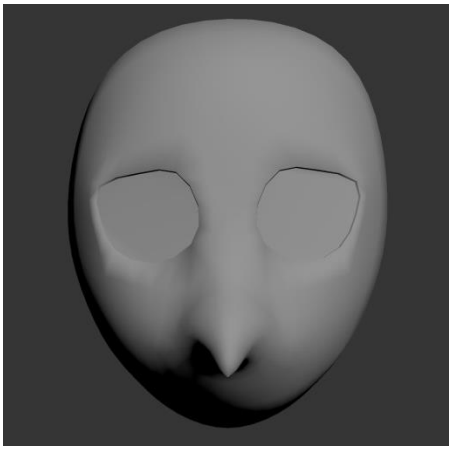
Figura 33 *- Visão lateral do modelo do personagem.*



Fonte: O Autor

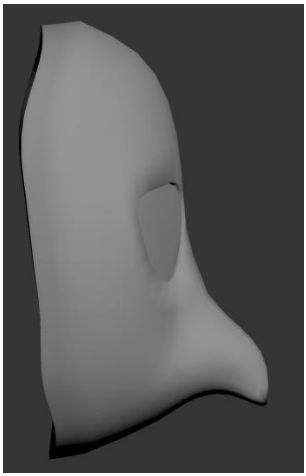
Foi então desenvolvida dentro do 3ds Max, a máscara que o personagem utilizaria para manter seu rosto escondido.

Figura 34 *– Visão frontal da máscara do personagem*



Fonte: O Autor

Figura 35 *– Visão lateral da máscara do personagem*



Fonte: O Autor

Os modelos foram então texturizados e reunidos, compondo a versão final do personagem.

3.6.5 Animações

Para que seja possível animar um personagem dentro de um ambiente em 3D, é necessário que o modelo do personagem passe por processos conhecidos como *rigging* e *skinning*. *Rigging* consiste em criar um esqueleto para o modelo, determinando a hierarquia dos ossos, seus movimentos e controles para facilitar o processo de animação. *Skinning* por outro lado, aplica o esqueleto ao modelo, de forma que ele informa aos vértices dos polígonos a posição que eles devem tomar de acordo com o movimento do osso. Esse processo pode ser realizado de forma manual em programas como 3ds Max porém, para o desenvolvimento deste projeto foi utilizado o site mixamo.com, um site no qual é possível realizar o *upload* de modelos em 3D e com um pequeno número de configurações, automaticamente realizar o processo de *rigging* e *skinning*.

Para executar esse processo, é necessário que o usuário crie uma conta Adobe, à qual o site pertence. Após a criação da conta, é possível realizar o *upload* do personagem ao banco de dados da Adobe. Com o *upload* realizado, é necessária a definição de certas características do personagem, como a localização de seu queixo, pulsos, cotovelos, joelhos e virilha, para que seja possível a realização do *rigging* e *skinning* automático.

Terminado o processo, é possível aplicar animações que a Adobe disponibiliza de forma gratuita. O banco de dados é constituído por milhares de animações que podem ser utilizadas no modelo e de forma limitada, configuradas para melhor se adequarem ao corpo do personagem.

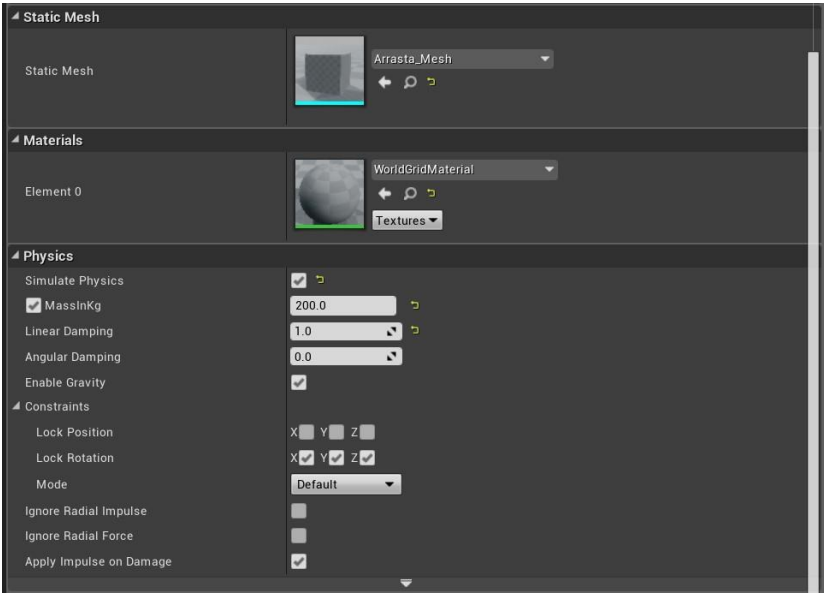
3.7 Criação das Mecânicas

A criação das mecânicas do jogo se deu em grande parte por conta da comunidade e dos diversos materiais disponíveis de forma gratuita para a Unreal Engine 4. Por conta da falta de conhecimento em programação dos autores do projeto, as mecânicas desenvolvidas no jogo foram limitadas ao que fosse realmente necessário para a concepção do jogo.

A criação das mecânicas se deu através de uma mescla do sistema de blueprints dos objetos e do nível em si.

A concepção da mecânica de arrastar blocos foi um tanto quanto simples, por conta da Unreal oferecer sistemas de simulação de física e colisão para todos os objetos disponíveis dentro do programa, além dos objetos a serem importados. Para acessar essas configurações foi criado uma *blueprint* do tipo *actor*. Dentro da *blueprint* então, foi selecionado o objeto utilizado para representá-la, seu material, e as configurações de simulação de física, onde foram determinados seu peso e a forma que ela poderia ser movimentada.

Figura 36 – *Configurações da Blueprint do bloco*

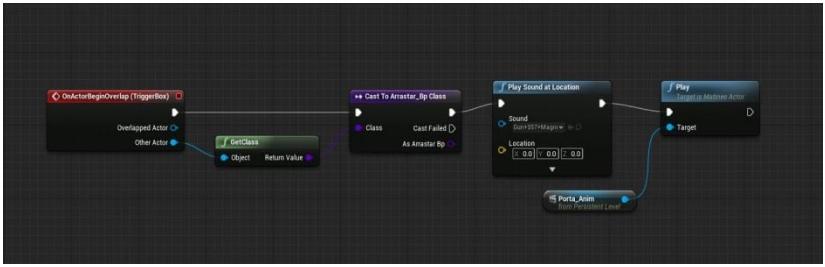


Fonte: O Autor

Tendo esses parâmetros configurados, é possível então criar a programação necessária para a criação dos *puzzles*. Para que o programa entenda que uma ação como abrir uma porta só deve ser ativada se o bloco for colocado no lugar certo, é necessária a criação de uma *trigger box*, uma marcação no cenário que detecta colisão. Dentro da *blueprint* referente ao *level*, é então criado um evento que a partir do momento

que o bloco for arrastado para o lugar correto, referente a localização da *trigger box*, uma ação será tomada, como uma a abertura de uma porta.

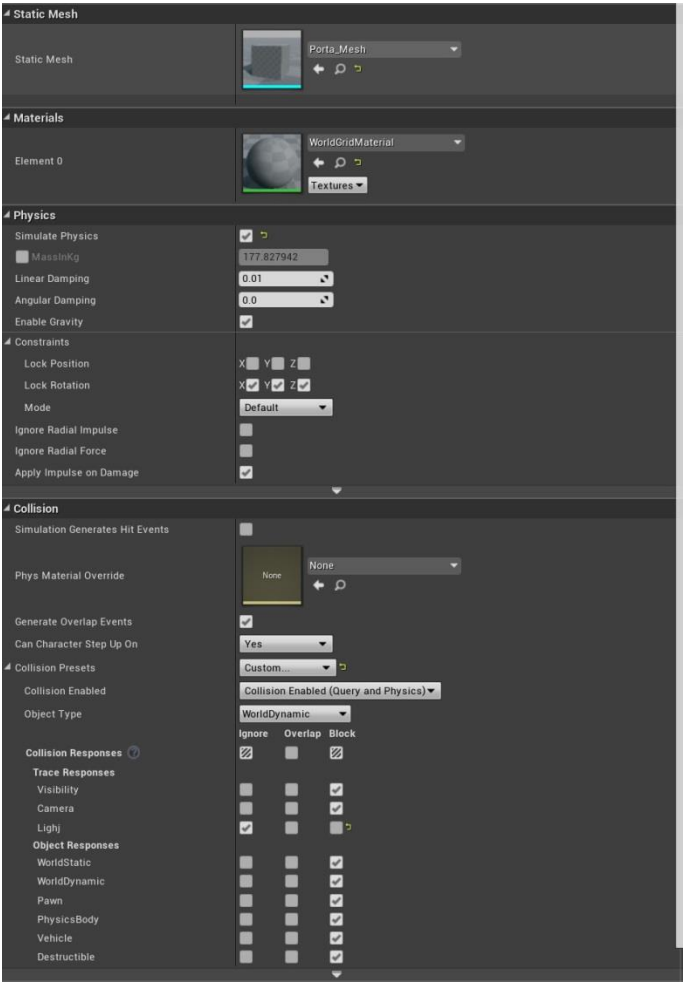
Figura 37 – *Configuração da ação a ser tomada*



Fonte: O Autor

Para a criação do sistema dos lasers, foram seguidos tutoriais disponíveis na internet, de forma que o autor do projeto não entende minuciosamente todos os detalhes incluídos em sua programação. O nível de complexidade do sistema dos lasers é muito maior que o dos blocos e necessitou da criação de diversos *blueprints* que interagem entre si, interpretando o laser. Primeiramente foi criada a *blueprint* do emissor, do tipo *actor*, composta de um objeto e um vetor, que irá indicar em que direção o laser deverá ser emitido. Além disso, foram configurados seus parâmetros físicos que determinam a forma que o jogador poderá interagir com esse objeto, seu material e uma colisão customizada, permitindo que o laser atravesse o objeto emissor.

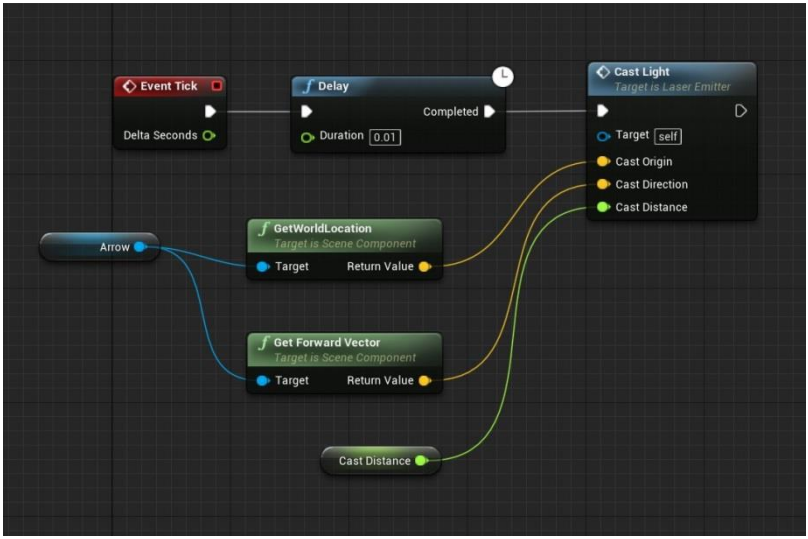
Figura 38 – *Configurações da blueprint do emissor*



Fonte: O Autor

Foi criado então um evento, que determina para onde o objeto deve emitir sua luz, além de configurar quantas vezes por segundo ela deve ser emitida e conectados na função *cast light* que é onde os parâmetros do laser são configurados.

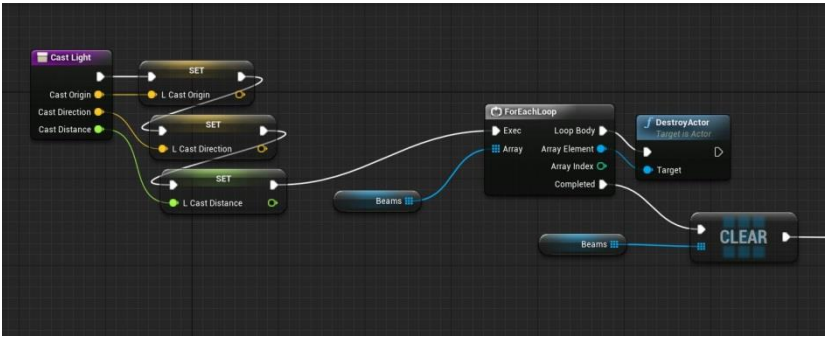
Figura 39 – *Configuração precedente a função Cast Light*



Fonte: O Autor

Dentro da função de *cast light*, é então configurada a forma que o laser será emitido, e como ele irá interagir com outros objetos presentes no cenário. Primeiramente é necessário configurá-lo de forma que o laser não seja infinitamente emitido, caso contrário o ele ficaria estático a partir do ponto em que foi emitido, e nunca sumiria, criando cópias de si mesmo.

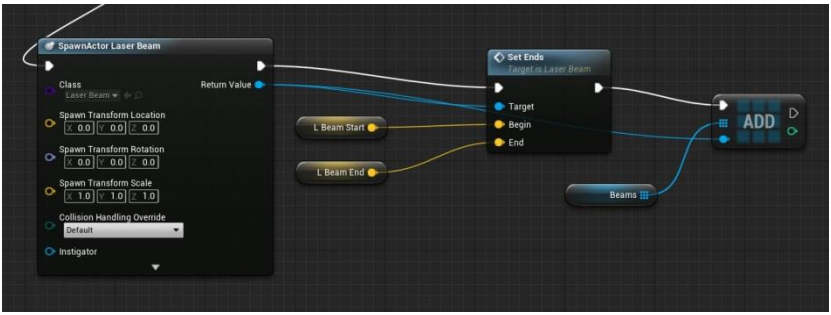
Figura 40 – *Configuração para que o laser não seja infinito*



Fonte: O Autor

Para que o laser possa ser visto pelo jogador, é necessário criar um objeto e atribuir um material, para representar o laser em si. Foi criado então um cilindro, e um material azul brilhoso para propósitos de teste. O objeto é então associado ao laser, dentro da função *cast light*.

Figura 41 – *Configuração da associação do material ao laser*



Fonte: O Autor

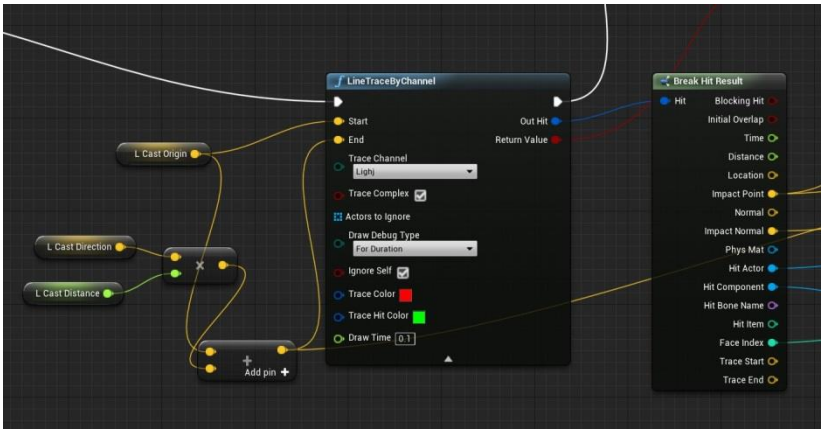
É necessário configurar então o laser de forma que ele possa interagir com o receptor e o espelho. Para o espelho, é necessário apenas designar um material que irá ser reconhecido, e então refletir o laser, já o receptor, é necessário primeiro criar uma blueprint do tipo *actor*. São então designados um objeto para representar o receptor e um material que irá agir como o receptor em si.

Figura 42 – *Configurações da blueprint do receptor*

Fonte: O Autor

Tendo um material para o espelho, e a *blueprint* do receptor, é possível continuar a configuração do laser. É criada então uma configuração que determina o que acontece quando o laser entra em contato com um objeto.

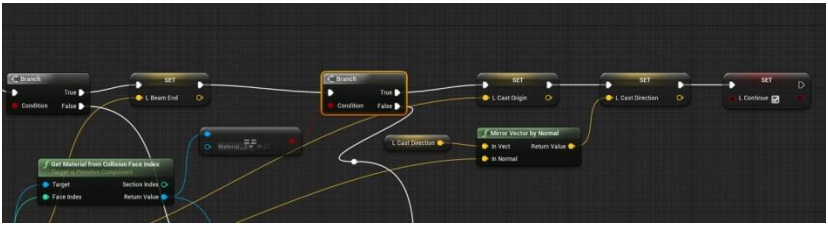
Figura 43 – *Configuração do contato do laser com objetos*



Fonte: O Autor

Entrando em contato com um objeto, o laser irá então analisar se o objeto acertado possui o material do receptor, do espelho ou de nenhum dos dois. Caso ele não reconheça nenhum dos materiais, ele irá apenas continuar a ser emitido. Caso o material seja reconhecido como o material do espelho, o laser irá então ser espelhado pelo vetor da *normal* do objeto acertado, redirecionando a direção em que é emitido.

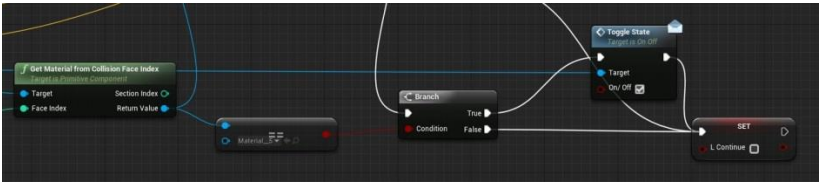
Figura 44 – *Configuração caso o laser reconheça o espelho*



Fonte: O Autor

Caso ele detecte que o objeto acertado possui o material do receptor, ele então irá ativar a blueprint do receptor em si.

Figura 45 – *Configuração caso o laser reconheça o receptor*



Fonte: O Autor

Com o laser ativando a blueprint do receptor, é possível então configurar a ação que o receptor irá executar caso ele seja ativado, como a abertura de uma porta, da mesma forma que a função de arrastar os objetos é configurada.

3.8 Desenvolvimento do Level Design

Com todos os *assets* necessários para o desenvolvimento do nível concluídos, foi dado início a fase de prototipagem, onde a partir do *gray box* foram determinados parâmetros como escala, espaço e a aplicação das mecânicas para a criação de puzzles e o nível em si.

3.8.1 Nível para testes

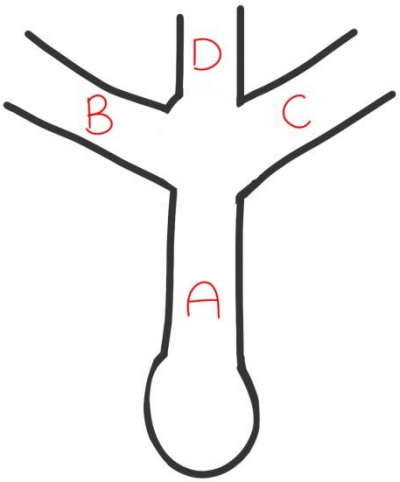
Um dos primeiros passos antes do desenvolvimento do nível de um jogo é ter certeza que as mecânicas funcionem de maneira apropriada, caso contrário podem ocorrer erros desastrosos que diminuem a experiência do jogador ou até mesmo que a impeçam.

Para isso, foi desenvolvido um nível de teste, onde foi possível em diversas situações, comprovar que as mecânicas se comportam de fato, como deveriam, providenciando assim a liberdade necessária para o desenvolvimento do level.

3.8.2 Desenvolvimento do *Gray box*

Antes do processo de gray box começar, foram determinados a partir de um mapa, a forma na qual o nível funcionaria conceitualmente. O nível foi dividido em 4 seções, que seriam completas de forma linear, introduzindo gradualmente as mecânicas do jogo, de forma que o jogador precisa estar ciente e dominá-las para poder prosseguir o nível.

Figura 46 *– Arte conceitual do nível do jogo*



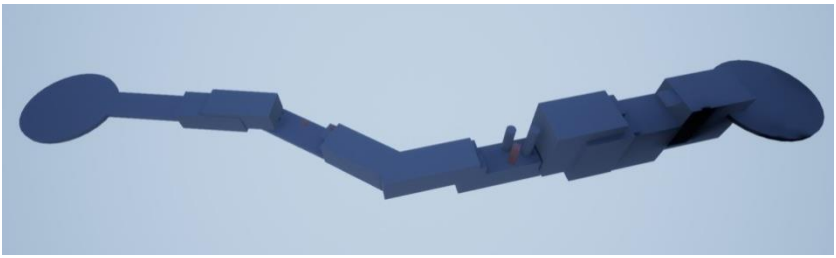
Fonte: O Autor

Para um melhor entendimento das intenções do autor durante o desenvolvimento do gray box, o nível será dividido em 4 seções: A, B, C e D facilitando assim a exposição do nível de forma coesa.

3.8.2.1 Primeira Iteração

Durante a seção A, onde o jogador inicia o jogo, são apresentadas as principais mecânicas necessárias para percorrer os níveis de Shards, o movimento e o pulo, além da possibilidade do jogador empurrar objetos.

Figura 47 *– Seção A do jogo construída no gray box*

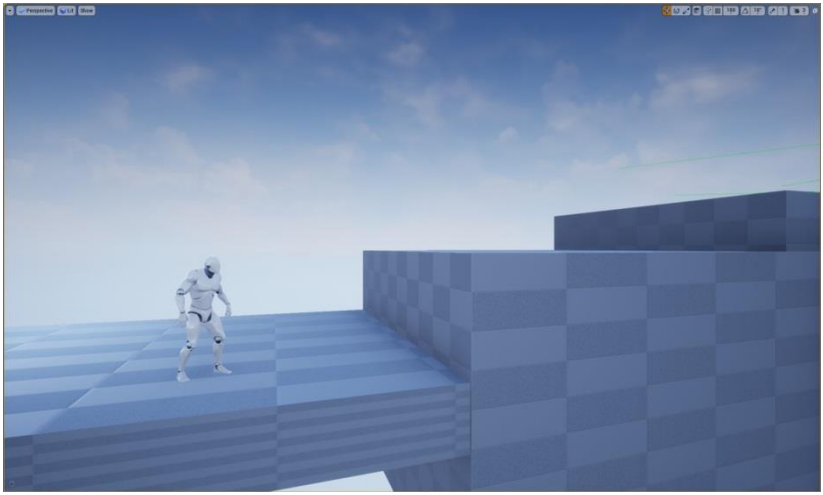


Fonte: O Autor

Durante o início da seção A, o jogador é colocado em um local com espaço aberto, dando-lhe assim a possibilidade de testar os controles de movimento e observar seus arredores. A partir da base em que ele inicia o jogo, só existe um caminho que ele possa percorrer, limitando suas escolhas de forma que ele não se sinta confuso.

Ao percorrer o caminho que lhe é apresentado, o jogador rapidamente irá se deparar com seu primeiro obstáculo, que irá ensiná-lo a pular. O pulo é uma parte essencial do jogo, então é importante que ele seja apresentado durante os primeiros momentos do jogo, e que o jogador entenda suas funcionalidades como altura e distância do pulo. São então apresentadas duas elevações no terreno, de forma que ele precise executar a ação de pulo 2 vezes, para que ela seja fixada em sua mente.

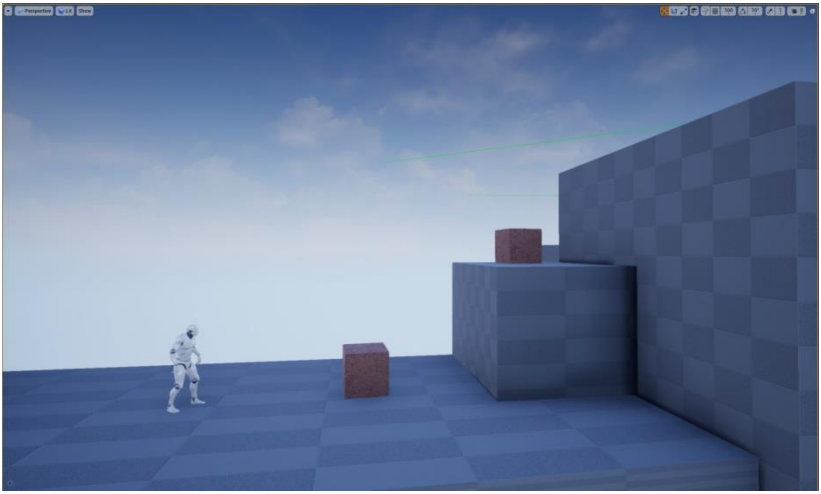
Figura 48 *– Primeiro obstáculo do jogador*



Fonte: O Autor

Após superar os primeiros obstáculos e aprender sobre a mecânica de pulo, o jogador é então confrontado com um novo obstáculo, propositalmente feito de forma que seja impossível de ser completado apenas com o pulo. Para superá-lo o jogador deve aprender sobre a mecânica de arrastar objetos. Durante os testes em *gray box*, evita-se usar texturas ou os objetos finalizados do jogo de forma que o designer possa focar inteiramente no *gameplay* e não se deixar distrair pela estética do jogo. Porém, para diferenciar os objetos que o jogador pode arrastar dos objetos fixos no cenário, foi aplicada uma textura marrom, a fim de facilitar sua identificação. Ao interagir com a caixa marrom, o jogador percebe então que ela pode ser movida, permitindo assim que ele a use como uma ferramenta para aumentar a altura que seu pulo pode alcançar. São apresentados 2 obstáculos com a mesma solução, novamente, reforçando a ideia de que certos objetos poderão ser empurrados no decorrer do jogo.

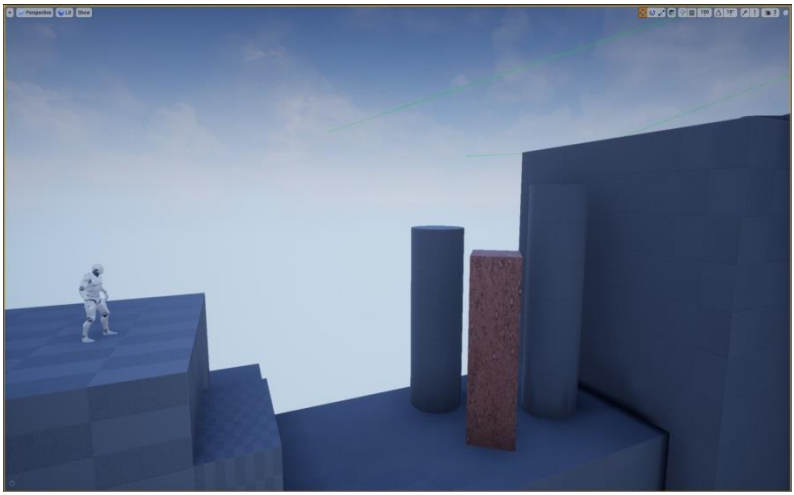
Figura 49 *– Seção tutorial sobre arrastar objetos*



Fonte: O Autor

Seu próximo obstáculo é então uma mistura das mecânicas que já lhe foram apresentadas, o pulo e os objetos que podem ser arrastados, demonstrando que o jogo irá utilizar essas mecânicas não apenas de forma isolada, mas também para complementarem umas as outras, criando assim uma experiência mais interessante.

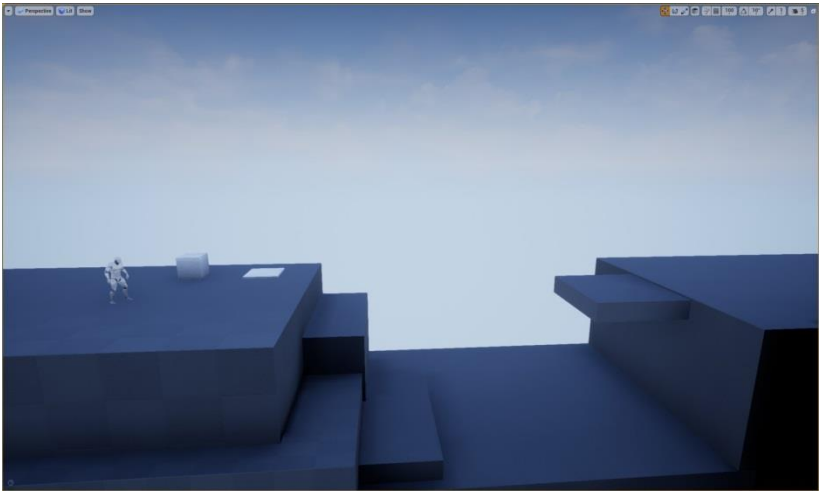
Figura 50 *– Representação do uso de plataformas e dos objetos que podem ser movidos*



Fonte: O Autor

Superando essa seção do jogo, o jogador pode notar que não há possibilidade de continuar sua travessia apenas com as mecânicas que lhe foram apresentadas, é então evidenciada uma variação da mecânica de arrastar objetos. Essa variação funciona de forma que, arrastando objetos correspondentes aos locais marcados no chão, eles irão ativar plataformas que permitem a travessia de certas seções do jogo. Para uma melhor visualização da mecânica, foram novamente utilizadas texturas para que não haja nenhum tipo de confusão aonde o jogador deve levar o objeto e colocá-lo. Em todas as instâncias que possa ocorrer uma falha por parte do jogador, no sentido de cair de uma plataforma ou errar seu pulo, foram dispostas pequenas plataformas para que ele possa voltar ao ponto inicial do desafio e não seja impedido de continuar o jogo.

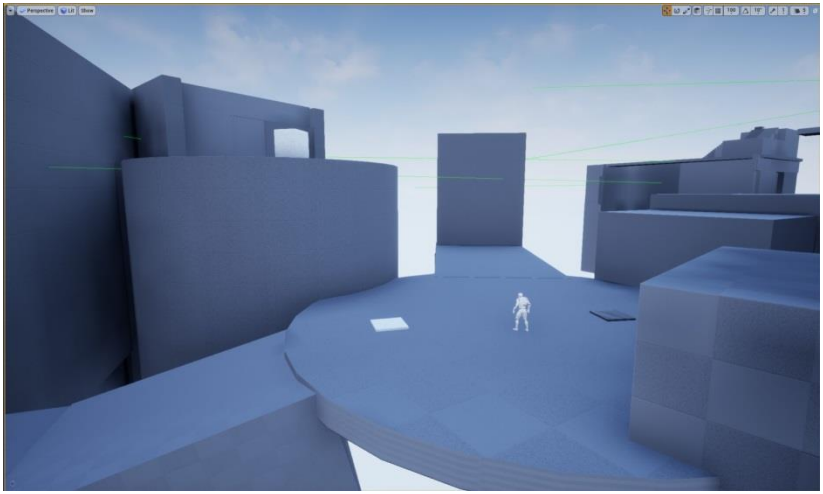
Figura 51 *– Seção tutorial sobre a ativação de plataformas*



Fonte: O Autor

A partir da superação desse desafio, o jogador tem agora o conhecimento de como ele deve movimentar-se, o pulo, e as mecânicas de arrastar objetos, que podem servir tanto como apoio para seus pulos como também como um tipo de “botão” que quando colocado no local correspondente à sua textura, irá ativar uma plataforma, permitindo que o jogador prossiga. Ele então supera a seção A do jogo, e alcança o ponto central do mapa, onde ele encontra 3 caminhos que pode vir a percorrer, a seção B, C e D. Alcançando esse ponto, o jogador pode vir a perceber, 2 marcas no chão, como a que foi utilizada para ativar a plataforma. Elas foram colocadas de forma que criem uma certa expectativa, do que elas irão fazer quando ativas. O jogador olhando ao seu redor pode então perceber uma das caixas que será utilizada para uma das marcas, sinalizando seu próximo objetivo. Durante esse ponto do jogo, apenas o caminho para a seção B está livre, impedindo que o jogador alcance prematuramente as seções C e D, então mesmo que ele não perceba a relação entre as marcas no chão e a caixa, ou até mesmo não as observe, ele apenas poderá percorrer um caminho, evitando que ele fique confuso ou preso.

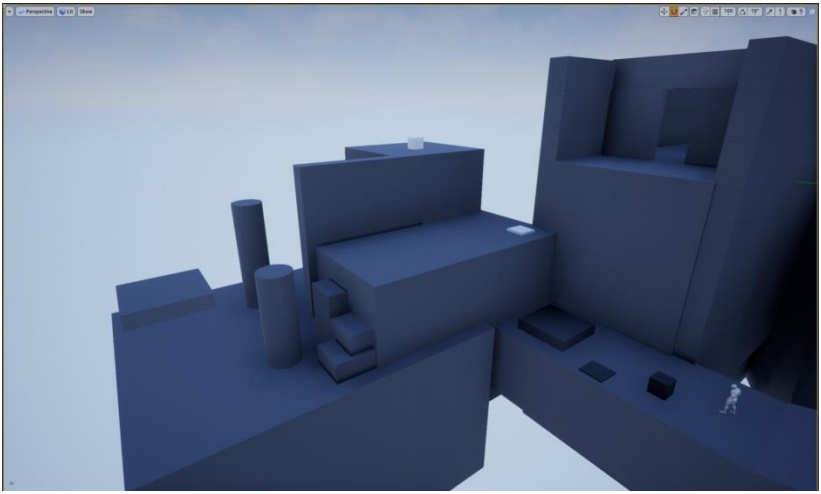
Figura 52 *– Imagem do ponto central do mapa de Shards*



Fonte: O Autor

Durante a primeira metade da seção B, o jogador deve utilizar seus conhecimentos prévios para fazer a travessia da mesma. A travessia do nível é feita de forma que não será necessário o conhecimento de algo que não lhe foi apresentado, evitando frustrações e criando uma experiência consistente, onde o jogador possa confiar no jogo.

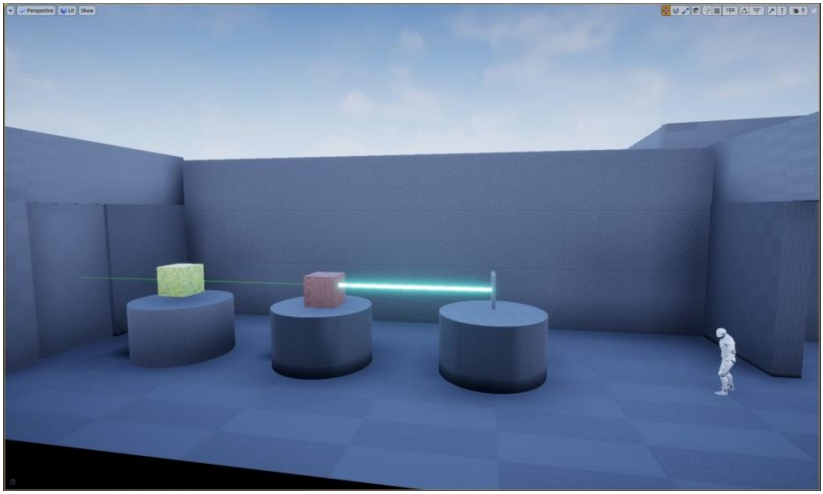
Figura 53 *– Primeira metade da seção B do mapa de Shards*



Fonte: O Autor

A partir da segunda metade da seção B, é apresentada ao jogador, a última mecânica necessária para completar o nível, a mecânica dos lasers. Primeiramente, ela é apresentada de uma forma simples, informando ao jogador que certos objetos podem bloquear o laser. Ao mover o objeto bloqueando o laser, ele então permite que o laser alcance uma caixa verde, que representa o receptor. A partir do momento em que o receptor é atingido pelo laser, uma porta se abre, indicando ao jogador seu objetivo: permitir o laser alcançar o receptor para que ele possa prosseguir.

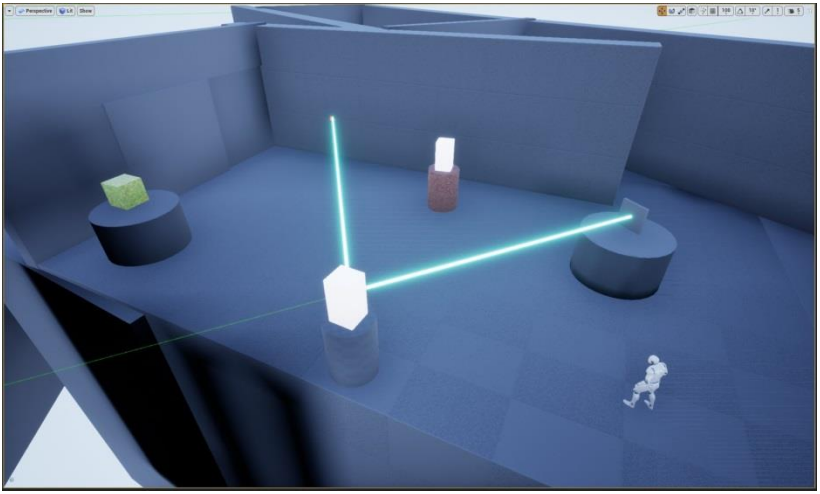
Figura 54 *– Apresentação da mecânica dos lasers*



Fonte: O Autor

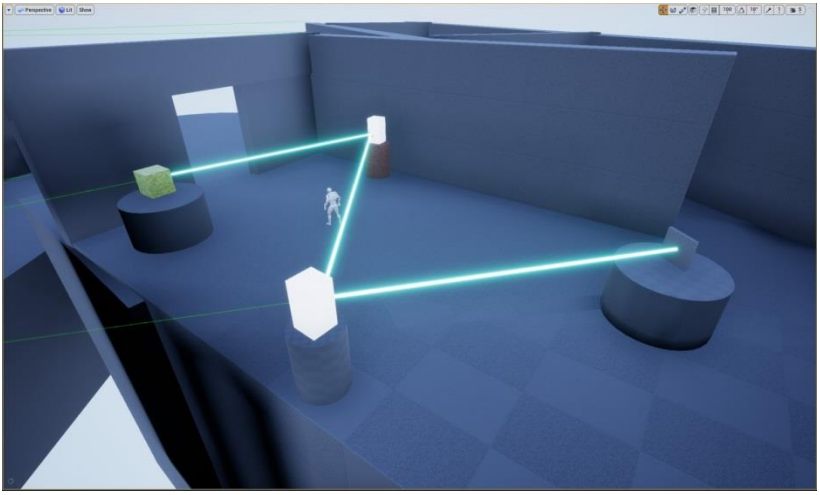
Logo em seguida, um novo *puzzle* de lasers é apresentado ao jogador, porém com um elemento a mais, os espelhos. Para que os *puzzles* possuam apenas uma resposta correta ou forma de completá-los, os espelhos foram separados em 2 categorias: os interativos e os não interativos. Os espelhos não interativos são fixos no cenário, e não podem ser movidos, limitando o jogador à resolver o puzzle com base em suas posições. Já os espelhos interativos podem ser movidos pelo cenário, porém não rotacionados, restringindo assim o *puzzle* a ser resolvido apenas da maneira pretendida. Para diferenciá-los, foi aplicada a mesma textura das caixas marrons a base do espelho interativo, sinalizando que ele pode ser interagido.

Figura 55 *– Apresentação dos espelhos para o jogador*



Fonte: O Autor

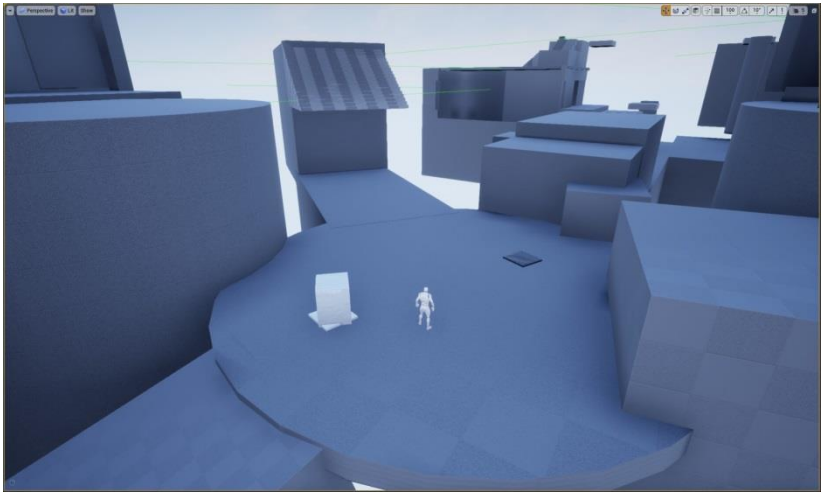
Figura 56 *– Puzzle dos espelhos completo*



Fonte: O Autor

A partir deste ponto, o jogador foi apresentado à todas as mecânicas que serão exploradas no nível. Os *puzzles* gradualmente ficam mais difíceis a fim de não perder o interesse do jogador e na tentativa de mantê-lo no *flow state*, alternando entre seções de plataformas e puzzles, criando uma variedade e personalidade para o jogo. O jogador pode então partir para a seção C, utilizando a caixa branca que podia ser observada a partir do ponto central do mapa. Caso ele a coloque no seu local de encaixe, perceberá que uma metade do caminho para a seção D será apresentada para ele, porém, ela ainda é inacessível, indicando que ele ainda necessita de mais uma peça para poder prosseguir, limitando assim sua escolha ao caminho que leva a seção C.

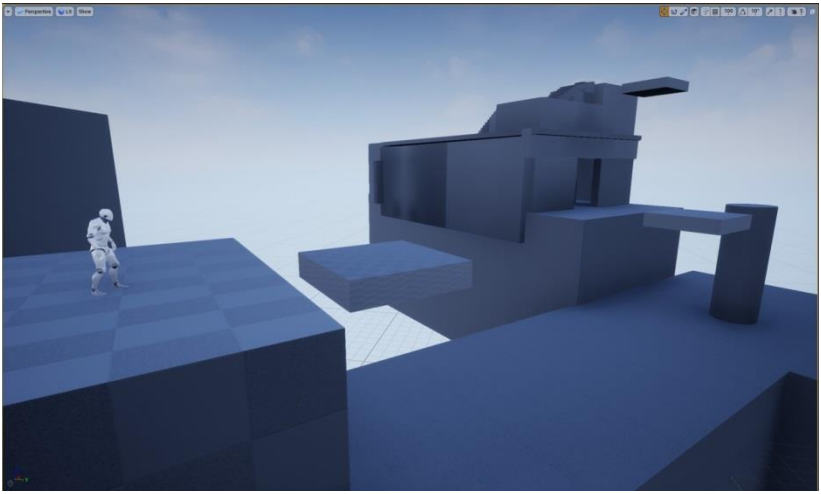
Figura 57 *– Apresentação da primeira metade do caminho para seção D*



Fonte: O Autor

O jogador passa então por uma seção de plataforma, onde agora existem 2 plataformas flutuantes, que se movem opostas uma à outra, utilizando uma mecânica já apresentada de forma que crie mais um desafio a ser superado.

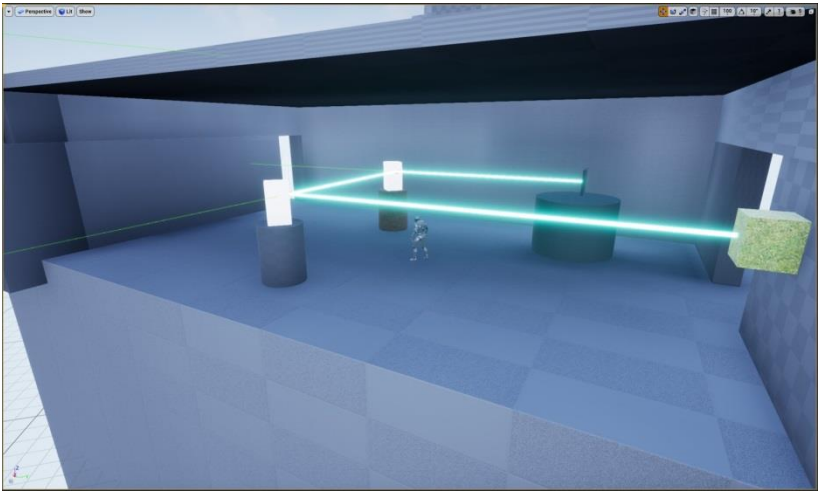
Figura 58 *– Plataformas flutuantes da seção C*



Fonte: O Autor

Ao passar com sucesso por elas, o jogador alcança mais um *puzzle* feito com os lasers, ainda mantendo a quantidade de espelhos em 2, permitindo o jogador a se acostumar com a forma que o laser interage com os espelhos.

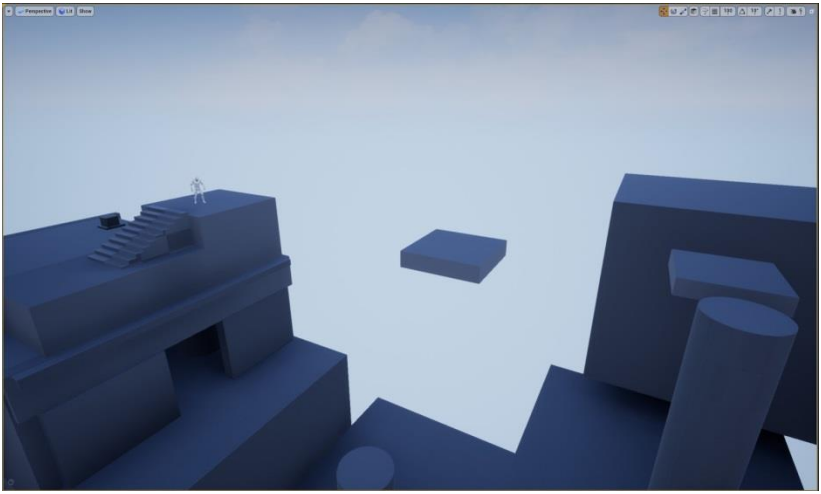
Figura 59 *– Terceiro puzzle com lasers, localizado na seção C*



Fonte: O Autor

Completando este *puzzle*, o jogador deve então ativar uma plataforma, o permitindo avançar no nível e alcançar a segunda metade da seção C.

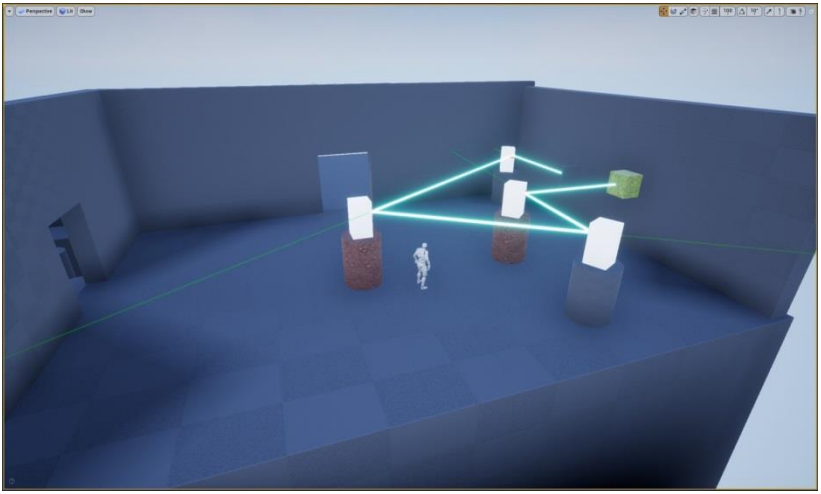
Figura 60 *– Plataformas que levam à segunda metade da seção C*



Fonte: O Autor

A partir da segunda metade da seção C são apresentados 2 *puzzles* que devem ser resolvidos para que o jogador complete o nível. O primeiro puzzle consiste de 4 espelhos, 2 interativos e 2 não interativos. Esse puzzle é propositalmente menos óbvio que os outros, tornando-o mais complicado e adicionando um maior nível de dificuldade ao jogo.

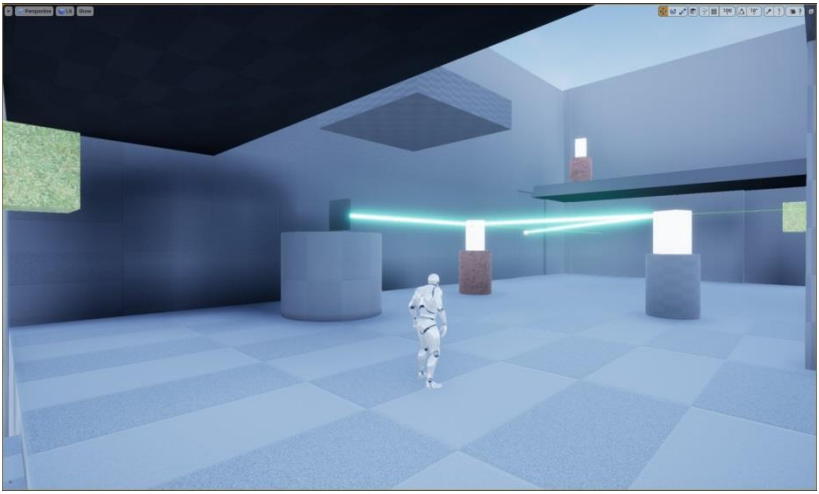
Figura 61 *– Quarto Puzzle com lasers, localizado na seção C*



Fonte: O Autor

Completando o primeiro puzzle, o jogador chega então ao seu último puzzle e desafio do nível. Ao adentrar a sala, o jogador nota que um dos espelhos está fora de seu alcance, e que não existe apenas 1 sensor na sala, mas sim 2.

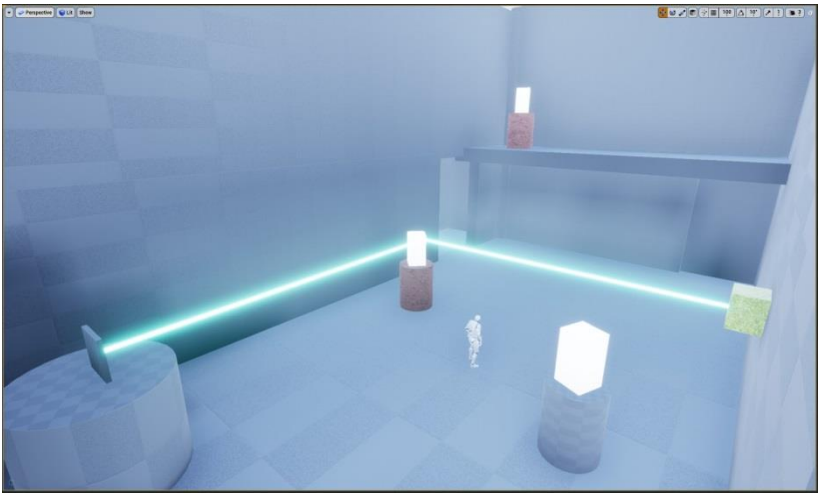
Figura 62 *– Quinto e último Puzzle com lasers, localizado na seção C*



Fonte: O Autor

O jogador precisa então completar 2 puzzles para que possa progredir no jogo. Primeiramente ele precisa abrir a porta que o permite alcançar o espelho no 2º andar. Com a forma que os espelhos são dispostos, é apenas possível que ele acerte um dos sensores, mantendo assim a linearidade do jogo e impedindo que o jogador “pule” uma etapa do jogo.

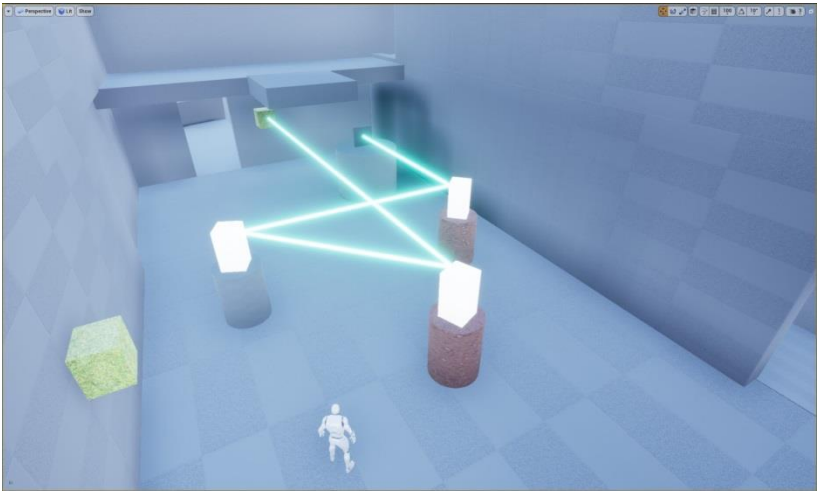
Figura 63 *– Ativação do primeiro receptor*



Fonte: O Autor

Ativando o primeiro receptor, o jogador então pode alcançar o terceiro espelho, tornando possível a resolução do *puzzle*, e providenciando assim a última peça necessária para ativar o caminho para a seção D.

Figura 64 *– Ativação do segundo e ultimo receptor*



Fonte: O Autor

Com as 2 caixas necessárias para a ativação das marcas no ponto central do mapa, o jogador pode então colocá-las de acordo com suas cores em seus respectivos lugares e ativar a passagem para a seção D, onde o nível é então finalizado.

3.8.2.2 Segunda Iteração

A partir do momento em que o nível em *gray box* estava completo, foram feitos testes informais, nos quais algumas pessoas foram convidadas para jogar o nível e identificar possíveis erros, melhorias que poderiam ser feitas e sugestões de como refinar o jogo. As principais críticas dirigidas ao jogo foram: Confusão sobre o que as caixas estavam ativando e se estavam ativando algo, erros na construção do cenário possibilitando que fosse possível pular seções do jogo e a escala de certas salas. A partir das críticas recebidas, foram realizados ajustes no *gray box*, visando uma melhor experiência de jogo.

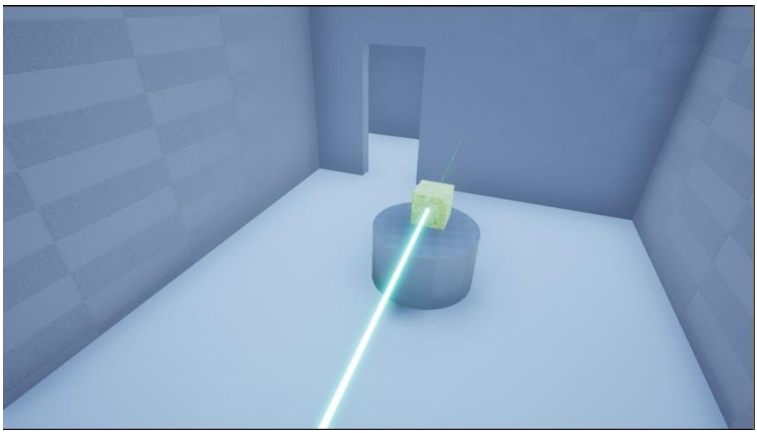
Para solucionar a confusão em relação às caixas, foram adicionados 2 elementos que ajudam o jogador a receber um *feedback* instantâneo sobre suas ações e são eles: uma *cutscene* e um som. A *cutscene* consiste em uma cena, previamente gravada, na qual o jogador

perde o controle de seus movimentos enquanto a cena toca. Ela foi utilizada de forma que quando um laser atingir um receptor ou uma caixa ativar uma plataforma, a câmera deixa de focar o personagem, e foca no objetivo que o jogador acabou de completar, dando-lhe a certeza de que sua ação realmente teve um efeito, ajudando a guiar o jogador para o próximo passo necessário para que complete o nível. Além disso, foi adicionado também um som, reforçando ainda mais o *feedback* que o jogador irá receber.

Figura 65 – *Representação da câmera do jogador antes de ativar o receptor*

Fonte: O Autor

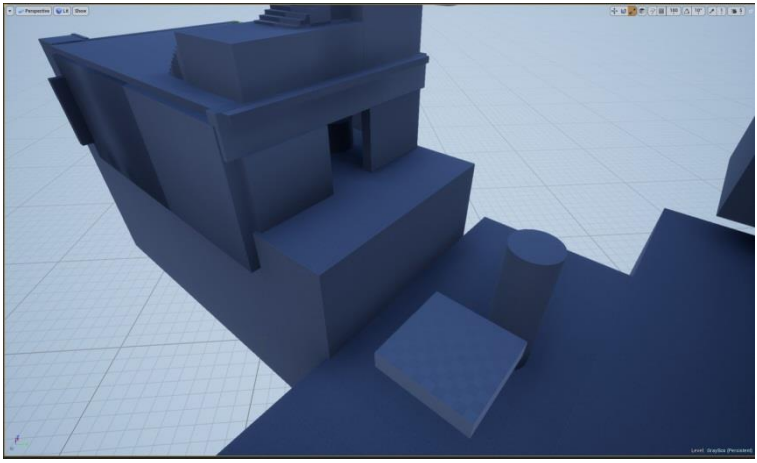
Figura 66 *- Representação da câmera do jogador depois de ativar o receptor*



Fonte: O Autor

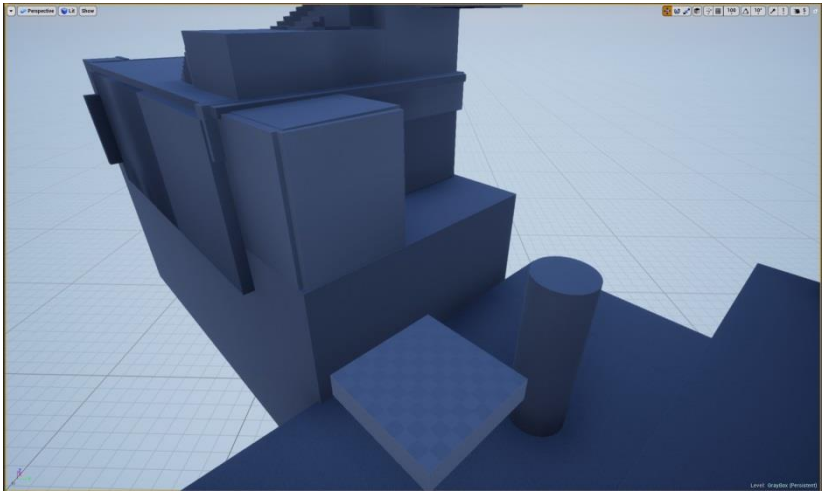
Para que os jogadores não pudessem pular seções do jogo, alguns objetos do cenário foram rearranjados, evitando assim que fosse possível burlar as etapas que o jogador deveria seguir.

Figura 67 *– Cenário antes das mudanças*



Fonte: O Autor

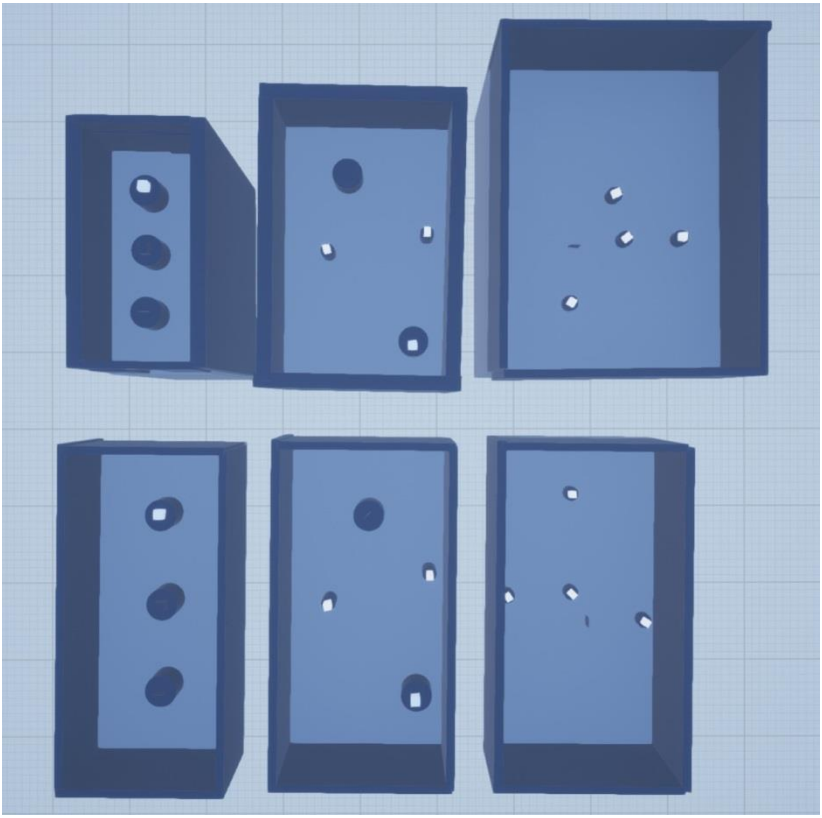
Figura 68 *– Cenário depois das mudanças*



Fonte: O Autor

Para que as salas se tornassem mais consistentes em relação a sua escala, elas foram movidas e reconstruídas lado a lado, mantendo uma uniformidade, principalmente em relação à altura.

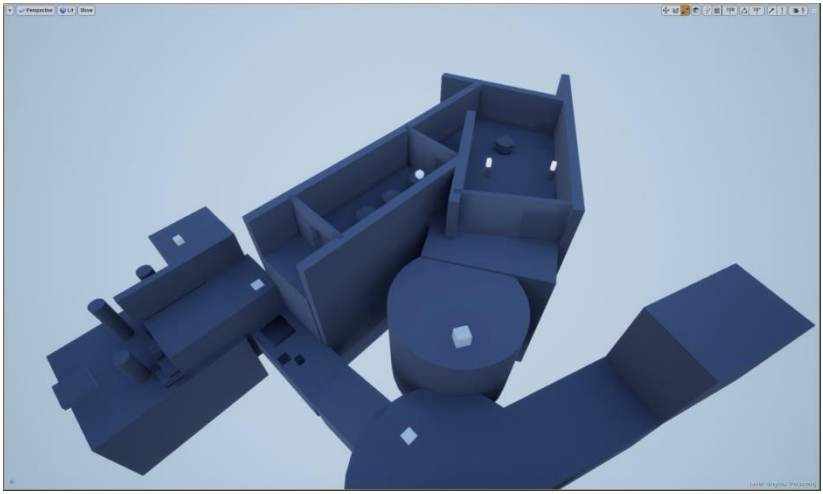
Figura 69 *– Comparação entre o antes e depois da reconstrução das salas*



Fonte: O Autor

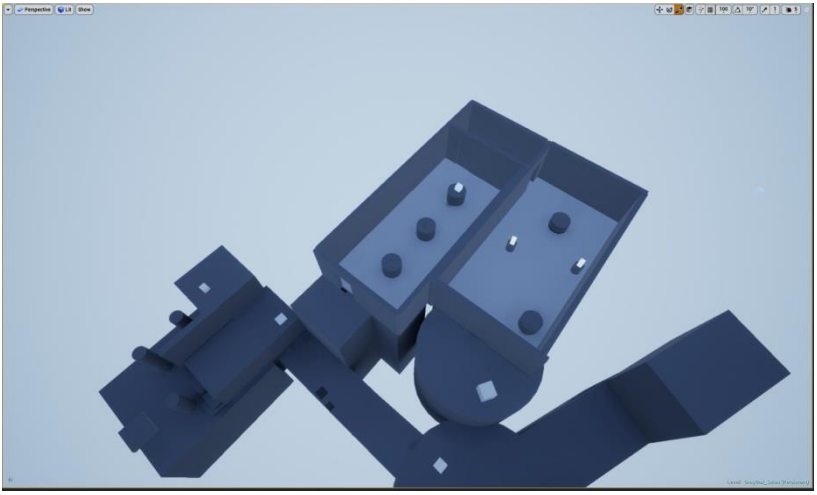
Após a reorganização das salas no cenário, é possível realizar uma comparação de como elas estavam dispostas.

Figura 70 *– Disposição das salas da seção B antes da reorganização*



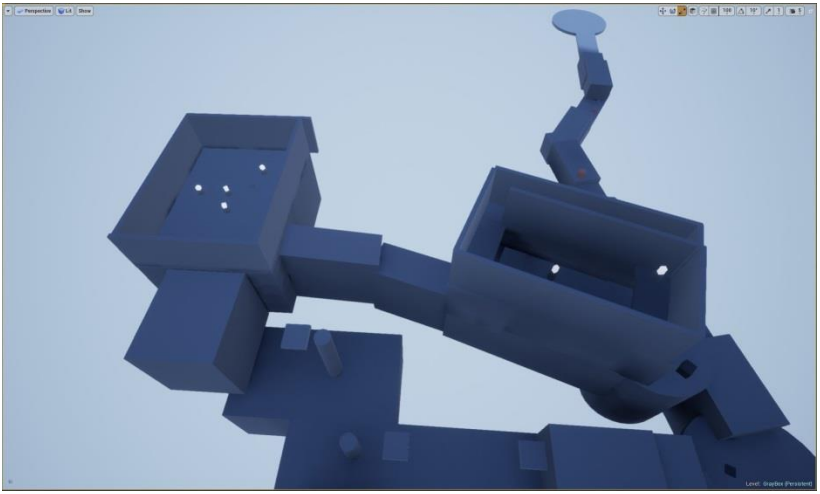
Fonte: O Autor

Figura 71 *- Disposição das salas da seção B depois da reorganização*



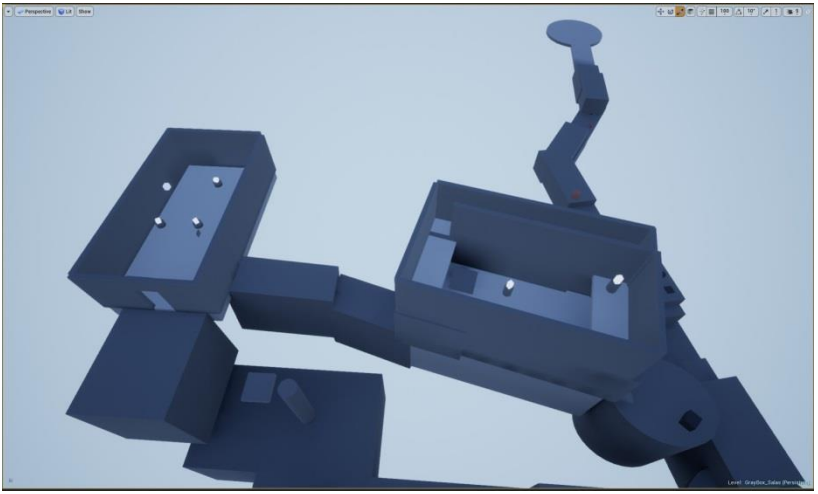
Fonte: O Autor

Figura 72 *- Disposição das salas da seção C antes da reorganização*



Fonte: O Autor

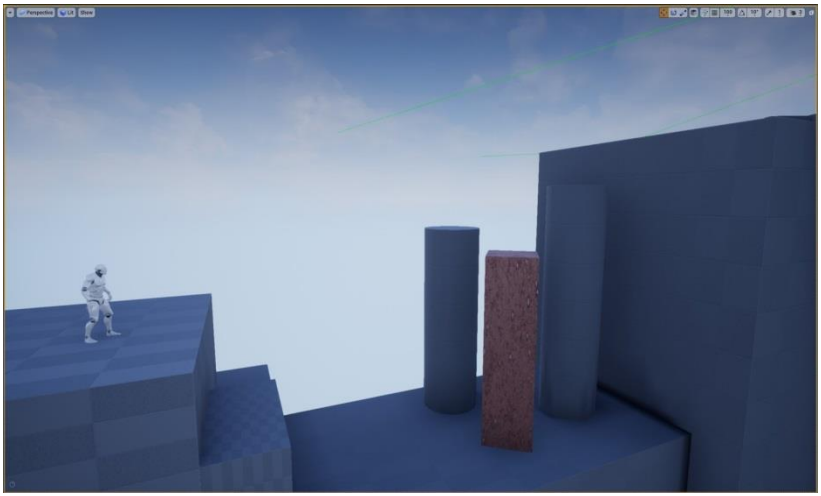
Figura 73 *- Disposição das salas da seção C depois da reorganização*



Fonte: O Autor

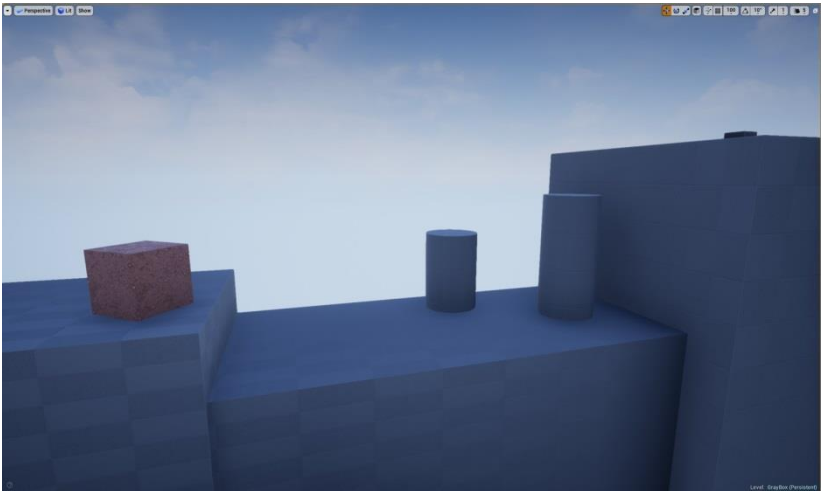
Por fim uma das partes da seção A foi refeita, pois a escala da caixa a ser empurrada estava confundindo os jogadores, e destoava do resto do cenário.

Figura 74 *– Plataformas da seção A antes do ajuste*



Fonte: O Autor

Figura 75 *- Plataformas da seção A depois do ajuste*

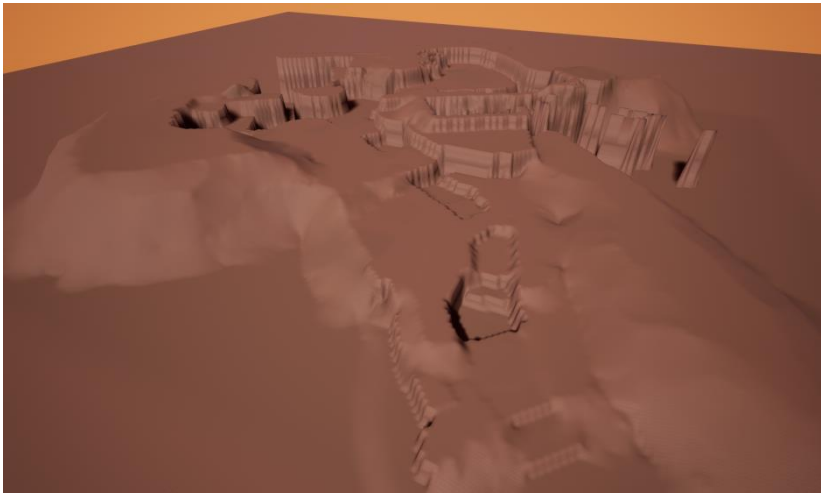


Fonte: O Autor

3.8.3 Montagem Final

Com o *gray box* finalizado, é possível dar início à 4ª e última etapa da metodologia *double diamond*, substituindo as caixas cinzas utilizadas para a concepção do gray box pelos assets desenvolvidos para o projeto, adicionar o *landscape* e a partir dele, desenvolver a versão final do nível. O *landscape* é uma ferramenta dentro da Unreal Engine 4 que permite o usuário esculpir uma malha providenciada pela *engine*, que se tornará o chão do mapa. Utilizando as funcionalidades dessa ferramenta, foi criada uma reprodução do nível *gray box*, mantendo-se fiel as escalas determinadas pelo processo de *gray box*.

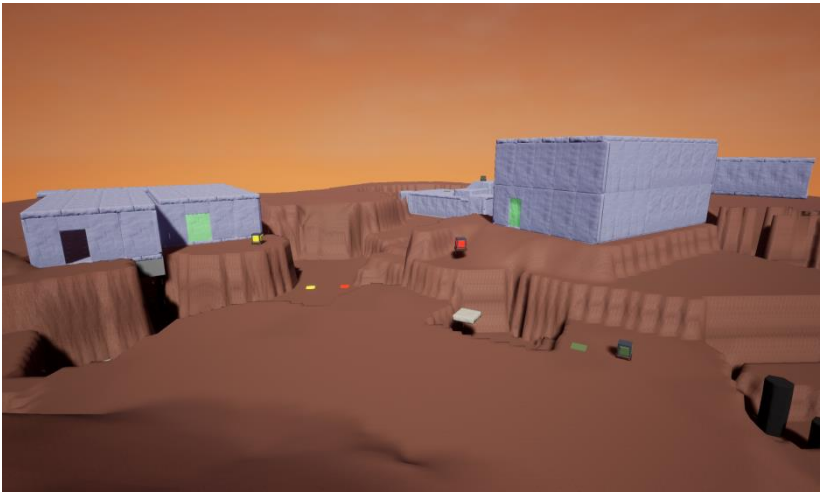
Figura 76 *– Landscape esculpido com base na disposição dos objetos no nível gray box*



Fonte: O Autor

Com o *landscape* construído, são então re-construídas as edificações, e plataformas flutuantes presentes no nível *gray box*, utilizando porém os *assets* desenvolvidos para o trabalho. Por conta do cenário ser em sua maioria orgânico, composto por vegetação e pedras, os edifícios se destacam no cenário, tornando-se *weenies*, guiando o jogador e servindo como ponto de referência para que ele se localize.

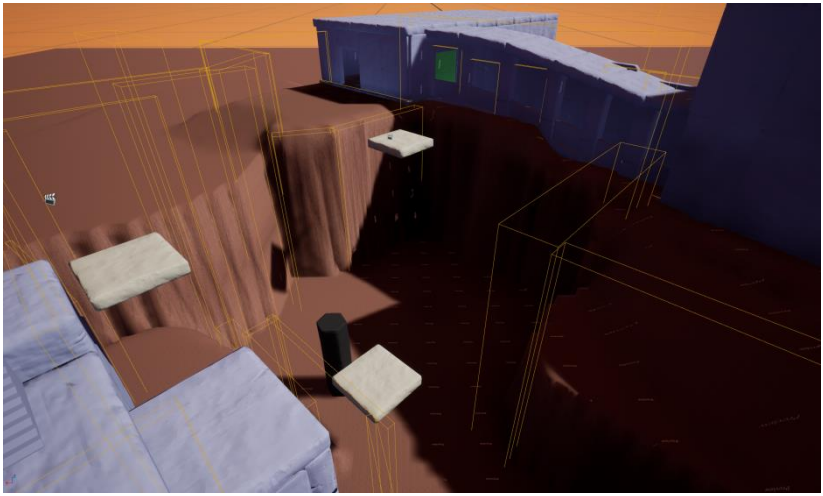
Figura 77 *– Edificações reconstruídas com base nos edifícios do gray box*



Fonte: O Autor

Por conta da maior parte do mapa do jogo possuir um caráter externo, onde não existem paredes físicas que podem ser usadas para impedir que o jogador alcance determinadas áreas, são adicionados volumes invisíveis bloqueando assim sua passagem e demarcando a área jogável do mapa. Para justificar a presença desses volumes então, são utilizadas texturas no chão, e árvores.

Figura 78 *– Volumes invisíveis que delimitam por onde o jogador pode se mover*



Fonte: O Autor

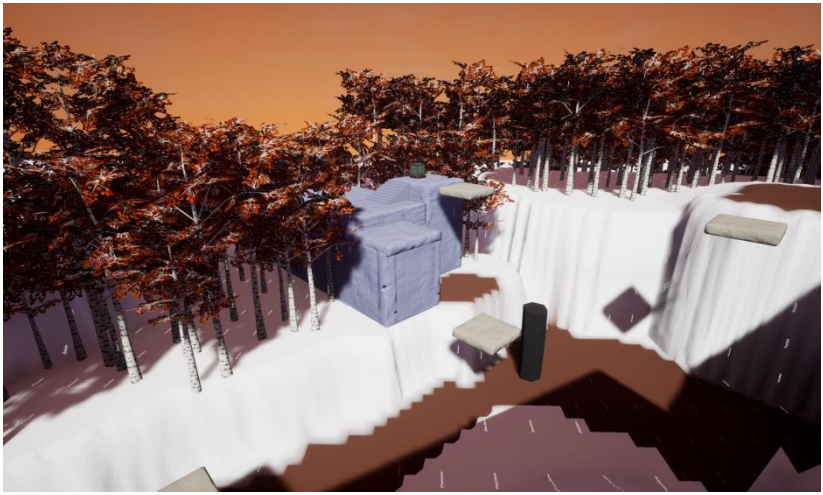
Para guiar o jogador através do nível, são então utilizadas 2 texturas diferentes para o chão do *landscape*, uma textura branca representando neve e uma textura de terra. A textura de terra é então aplicada pelo caminho a ser percorrido pelo jogador, criando um contraste com a neve branca que é utilizada apenas em áreas inacessíveis, gerando uma comunicação visual ao jogador de que ele pode ignorar áreas com neve, pois não pode acessá-las.

Figura 79 *– Aplicação das texturas de neve e terra*

Fonte: O Autor

Com o propósito de reforçar essa linguagem visual, são então adicionadas árvores onde a textura de neve foi aplicada, criando uma associação para o jogador de que ele não poderá acessar áreas que estejam diretamente bloqueadas pelas árvores. As árvores servem também como uma forma de limitar a visão do jogador, tornando o processo de descobrimento do nível algo gradual, onde as áreas do jogo não são reveladas antes que ele se aproxime delas. Além disso, elas também disfarçam o fato do jogador estar em um mapa limitado, insinuando assim que o mundo em que o jogo se passa é maior do que de fato é.

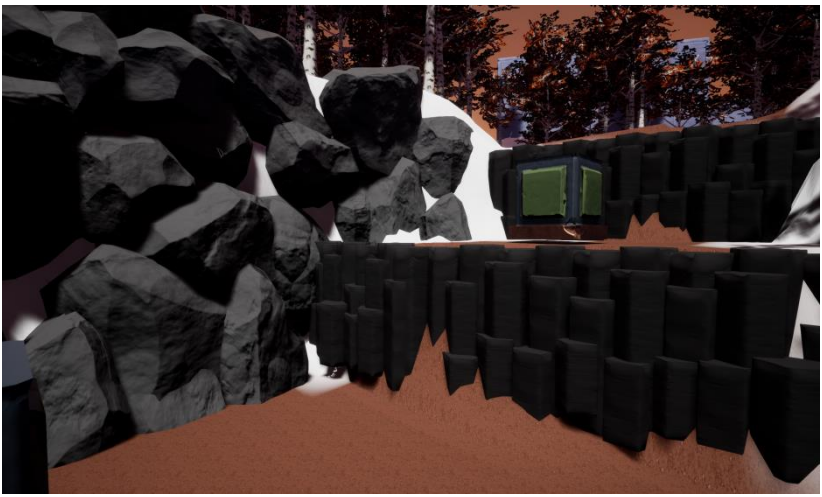
Figura 80 *– Disposição das árvores no cenário*



Fonte: O Autor

Como um dos principais elementos do jogo é o pulo, foi importante determinar uma claridade visual de onde o jogador pode pular, impedindo assim frustrações e confusões sobre sua ações. Para isso, foram adicionadas as pedras de basalto diretamente na base de elevações do cenário onde o jogador pode ou deve pular para alcançá-las. Para elevações em que o jogador não pode escalar, foram utilizadas as pedras simuladas, criando um contraste claro, que pode ser percebido nos primeiros minutos de jogo, tornando a ação do pulo mais intuitiva.

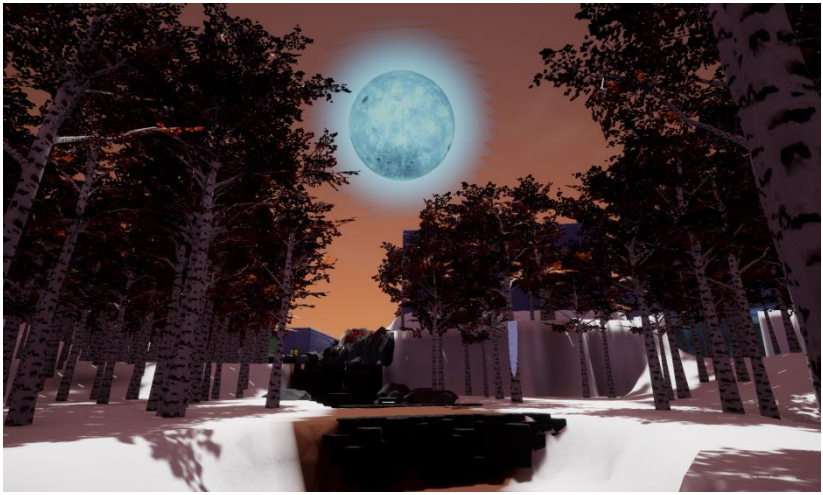
Figura 81 *– Contraste entre a forma das pedras de basalto e pedras simuladas*



Fonte: O Autor

A lua sendo um dos principais elementos da narrativa foi posicionada no cenário de forma que funcione como um *weenie* para o jogador, ajudando-o a guiar-se pelo nível, e providenciando a maior parte da iluminação ambiente do cenário.

Figura 82 *– Lua no cenário*



Fonte: O Autor

Por conta do jogo se passar em um cenário noturno, foram utilizados diversos pontos de luz adicionais no cenário, impedindo que existam áreas escuras demais para a visualização dos objetivos ou objetos a serem interagidos. Apesar de não serem completamente realistas, é importante notar que a experiência do jogador será prejudicada caso ele não consiga visualizar os objetos em seu caminho. Essas luzes funcionam também como mais um reforço visual para o jogador, determinando que os pontos mais claros do cenário podem ser alcançados, enquanto áreas que estiverem mais escuras podem ser ignoradas. Além dos pontos de luz, foram adicionados canhões de luz, utilizados para destacar os objetivos finais do jogador no nível, guiando-o para eles.

Figura 83 – *Cenário apenas com luzes naturais*

Fonte: O Autor

Figura 84 – *Cenário com luzes naturais e pontos de luz de apoio*

Fonte: O Autor

Para que o jogador entenda quais os objetivos do jogo, e as formas de alcançá-los, foram adicionadas mensagens de tutoriais que explicam as teclas de movimentação do personagem, além das mecânicas utilizadas para a construção do jogo, como as plataformas e os lasers, evitando que o jogador tenha que adivinhar como o jogo em si funciona.

Figura 85 *– Apresentação das mensagens de tutorial ao jogador*

Fonte: O Autor

Foram adicionadas as flores e vegetações baixas ao cenário, preenchendo áreas do mapa que destoavam do resto do cenário por conta de estarem vazias, sem que nenhum objeto as ocupassem.

Figura 86 *– Cenário populado com flores e vegetação baixa*

Fonte: O Autor

Para ampliar a credibilidade do cenário do jogo, no que se diz respeito à seu bioma de tundra e taiga, foram adicionadas partículas de neve, tornando a experiência mais imersiva ao jogador além de complementar a estética do jogo.

Figura 87 *– Párticulas de neve espalhadas pelo cenário*

Fonte: O Autor

Foi então adicionado o modelo do personagem desenvolvido durante o projeto e suas animações tornando a estética do jogo mais consistente.

Figura 88 *– Visão frontal do personagem finalizado*

Fonte: O Autor

Figura 89 *- Visão posterior do personagem finalizado*

Fonte: O Autor

Por fim foi criada uma *cutscene* que é reproduzida a partir do momento em que o jogo começar, introduzindo alguns dos elementos de narrativa ao jogo, na qual os acontecimentos descritos no item 3.3.1, são apresentados de forma visual, criando assim um objetivo para o jogador (alcançar o fragmento) e um elemento de antecipação em relação ao que é a luz que ele observou.

Figura 90 *- Representação da visão do jogador durante a cutscene*

Fonte: O Autor

4 CONCLUSÃO

Durante o processo de criação do projeto notou-se o quão expansiva a área de criação de jogos realmente é, de modo que o processo de constituição de uma simples demo emprega o conhecimento de diversas áreas de desenvolvimento de jogos, indicando assim que essa é uma tarefa que preferivelmente deveria ser realizada por uma equipe composta por especialistas em diversas áreas do conhecimento, permitindo assim que o projeto alcance maiores objetivos tendo em vista que cada profissional estaria focado em sua área. Pelo fato do projeto estar sendo desenvolvidas por apenas 2 pessoas, foi necessário abandonar certos aspectos da produção, proporcionando assim uma maior chance de refinamento para as áreas de especialização dos autores do projeto, e evitando que a qualidade do projeto fosse prejudicada.

Como uma das propostas do projeto era criar um portfólio profissional, foi decidido que o desenvolvimento dos assets utilizados durante a concepção do protótipo fosse feito partindo do zero. A partir disso, foi possível observar as diversas exigências necessárias para criar objetos que fossem de fato *game ready*, das quais não seriam possíveis serem atendidas sem o auxílio de uma metodologia de projeto que incentivasse a iteração, permitindo que fosse possível voltar e reavaliar decisões tomadas durante o desenvolvimento de todo projeto.

Durante a criação dos assets foi também percebida a grande exigência pelo conhecimento de diversos *softwares* para a criação dos mais variados objetos e as suas diferenças, revelando assim as diversas áreas nas quais o *designer* pode se especializar, como a criação de objetos orgânicos, personagens, *hard surfaces* ou *landscapes*. É importante notar que apesar de existir a possibilidade de uma especialização, é importante que o *designer* conheça outras técnicas e áreas do desenvolvimento de jogos, a fim de criar um ambiente de trabalho no qual é possível a comunicação entre os setores de criação do jogo, facilitando a solução de problemas e a concepção de ideias, além do entendimento dos objetivos almejados pelo projeto.

Foi possível também observar o impacto que a programação possui durante o desenvolvimento do jogo, da qual na ausência da mesma é impossível a concepção de um jogo digital. A partir disso é recomendado que o *designer* possua no mínimo um entendimento básico dos eventos, funções e variáveis para que seja possível o desenvolvimento de mecânicas que irão compor o jogo. Tendo isso em vista, vale ressaltar que a comunidade que cerca os programas de

desenvolvimentos de jogos como Unreal Engine 4 e Unity possuem um grande número de usuários dispostos a ajudarem um usuário em dúvida.

Durante o desenvolvimento do projeto e das definições de *game* e *level design*, foi possível perceber o quão conectadas essas áreas estão, de forma que durante o desenvolvimento de um jogo digital elas se tornam dependentes uma da outra. Enquanto o *game design* procura determinar todos os aspectos conceituais do jogo, desde a sua ideia inicial até a forma que o jogo será concluído, é o *level design* que de fato permitirá essa ideia ser reproduzida em um meio digital, tornando assim o papel de um game designer de certa forma menos envolvido com o projeto de forma direta, incorporando o papel similar ao de o diretor de um filme, enquanto que o *level designer* toma proveito das ideias e conceitos já estabelecidos e a partir deles, cria o meio pelo qual o usuário irá experienciar o jogo em si.

A utilização da metodologia *double diamond* por fim, manteve o projeto em um constante estado de produção, permitindo que os autores pudessem realizar os ciclos de desenvolvimento de forma independentes um do outro, aumentando o nível de produção e viabilizando um maior refinamento do protótipo final, por conta disso, foi possível o atendimento dos objetivos propostos inicialmente, de criar um portfólio profissional, e o protótipo de um jogo com base nos conceitos de *game* e *level design*.

REFERÊNCIAS

BARON, Sean. **Cognitive Flow: The Psychology of Great Game Design.** 2012. Disponível em: <https://www.gamasutra.com/view/feature/166972/cognitive_flow_the_psychology_of_.php>. Acesso em: 11 nov. 2019.

DESIGN COUNCIL. **What is the framework for innovation? Design Council's evolved Double Diamond.** 2019. Disponível em: <https://www.designcouncil.org.uk/news-opinion/what-framework-innovation-design-councils-evolved-double-diamond>. Acesso em: 11 nov. 2019.

DESIGNING Journey. San Francisco: Gdc, 2013. (58 min.), son., color. Disponível em: <https://www.youtube.com/watch?v=UGCkVHSvjzM>. Acesso em: 11 nov. 2019.

KREMERS, Rudolf. **Level Design:** Concept, Theory and Practice. Natick: A K Peters, 2009. 408 p.

MASTERCLASS. **Writing 101: What Is the Hero's Journey? 2 Hero's Journey Examples in Film.** 2019. Disponível em: <https://www.masterclass.com/articles/writing-101-what-is-the-heros-journey#want-to-become-a-better-writer>. Acesso em: 11 nov. 2019.

PUGH, Tom. **Level Design Tips and Tricks.** 2019. Disponível em: <https://www.gamasutra.com/blogs/TomPugh/20181022/329044/Level_Design_Tips_and_Tricks.php>. Acesso em: 11 nov. 2019.

ROGERS, Scott. **Level Up!:** The Guide to Great Video Game Design. Chichester: John Wiley & Sons, 2010. 514 p.

SCHELL, Jesse. **The Art of Game Design:** A Book of Lenses. 2. ed. Pittsburgh: Ak Peters, 2014. 600 p.

SHAHRANI, Sam. **Educational Feature: A History and Analysis of Level Design in 3D Computer Games - Pt. 1.** 2006. Disponível em: <https://www.gamasutra.com/view/feature/131083/educational_feature_a_history_and_.php>. Acesso em: 11 nov. 2019.

WEPC. **2019 Video Game Industry Statistics, Trends & Data.** 2019. Disponível em: <https://www.wepc.com/news/video-game-statistics/>. Acesso em: 11 nov. 2019.

WIJMAN, Tom. **The Global Games Market Will Generate $152.1 Billion in 2019 as the U.S. Overtakes China as the Biggest Market.** 2019. Disponível em: <https://newzoo.com/insights/articles/the-global-games-market-will-generate-152-1-billion-in-2019-as-the-u-s-overtakes-china-as-the-biggest-market/>. Acesso em: 11 nov. 2019.