Desenvolvimento de um *Workflow* para a Criação de Cenários Foto Realistas para Jogos.

Victor Cesário Coletto[1*] Guilherme Chagas Kurtz[1] Ricardo Frohlich da Silva[1]

Universidade Franciscana[1]

RESUMO

Nos jogos, o estilo foto realista, que é a semelhança de uma obra de arte com uma representação fotográfica, é almejado a décadas. Na indústria cinematográfica, o fotorrealismo é presente via gráficos pré-renderizados (CGI), mas nos jogos, por serem em gráficos em tempo real, demandam mais poder computacional. Apenas recentemente, com o avanço da tecnologia, pode-se dizer que o fotorrealismo em jogos ficou possível, mas apenas em cenários. Assim, este trabalho teve como objetivo geral a elaboração de um *workflow* ou *pipeline* de criação para a criação de cenários foto realistas para jogos, tendo como base processos utilizados por artistas da área. Para isso, foi executada uma pesquisa sobre fotorrealismo em jogos, avanços tecnológicos para atingi-lo, diferentes *workflows* de artistas 3D, otimizações de arte para jogos e a definição de um estudo de caso aplicando o *workflow* desenvolvido. A metodologia apresenta a criação do workflow, com uma abordagem por camadas, abrangendo desde sua concepção até finalização em dez passos. E por fim, foi elaborado um estudo de caso, validando o *workflow* desenvolvido e criando uma *cinematic*.

Palavras-chave: jogos, *workflow*, cenários, fotorrealismo.

1 INTRODUÇÃO

O fotorrealismo, que é a semelhança de uma obra de arte com uma representação fotográfica [1] começou a ser almejado pelos artistas e desenvolvedores na década de 1990, quando gráficos 3D viraram o padrão da indústria dos jogos [2]. E com os grandes saltos computacionais a cada geração de *hardwares*, ajudou a tornar o sonho cada vez mais real, o sonho de atingir o fotorrealismo em tempo real.

Na indústria cinematográfica, o fotorrealismo já é presente via CGI, *computer-generated imagery*, mas são os gráficos pre-renderizados, onde o computador pode levar horas, dias ou até semanas renderizando uma cena complexa. Já nos jogos os gráficos são exibidos em tempo real em sua maior parte, demandando mais poder computacional do *hardware*. Nos últimos anos, os jogos conseguiram atingir o realismo, mas não o fotorrealismo [3]. Mas recentemente o fotorrealismo em cenários ficou possível [4] por meio de *engines* mais poderosas, *hardwares* mais potentes, *scans* de objetos reais em 3D, texturas em alta resolução, *ray tracing* em tempo real e mais.

Com a facilidade da disponibilidade de programas profissionais ao público, o interesse de se especializar na área 3D cresceu muito com o passar dos anos [5]. E um dos problemas é encontrar conteúdo que exemplifique o processo completo de criação, desde o rascunho até a finalização, em artigos científicos ou em fontes mais renomadas na indústria. No caso de cenários nos jogos, muitos artistas publicam suas artes já finalizadas, ou sem uma explicação do processo, ou havendo somente a explicação de uma pequena parte. Com isso em mente, a ideia de unir informações de artistas da indústria de jogos sobre diferentes partes dos processos de criação num *workflow* completo surgiu.

Este trabalho possui como objetivo geral a criação de um *workflow* para a criação de cenários para jogos, com base em processos utilizados por artistas da área. Os objetivos específicos são: pesquisar sobre o fotorrealismo na indústria dos jogos e novas tecnologias para atingi-lo; pesquisar sobre diferentes *workflows* de criação utilizados por artistas 3D; pesquisar técnicas de otimização de arte para jogos; elaboração de um *workflow* para a criação de cenários de acordo com as pesquisas realizadas; criar um estudo de caso aplicando o *workflow* desenvolvido; criar uma *cinematic* mostrando detalhes do cenário finalizado; validar o uso do workflow criado.

2 REFERENCIAL TEÓRICO

Neste referencial teórico, será apresentada uma pesquisa sobre fotorrealismo nos jogos, tanto em personagens como em cenários, apresentando também algumas das dificuldades encontradas para atingir tal estilo. Após, serão explicadas ferramentas, termos e técnicas utilizadas para o desenvolvimento de um cenário realista num jogo.

2.1 Fotorrealismo

Realismo é uma arte no qual o objetivo é representar o mundo real sinceramente e objetivamente, baseado em uma observação atenta dos detalhes comuns e vida contemporânea [1]. Já o fotorrealismo, segundo James Gurney [1], é a semelhança de uma obra de arte com uma representação fotográfica. Os dois são similares, mas o fotorrealismo é um subgênero que leva muito mais a sério a representação da realidade.

2.2 Fotorrealismo em Jogos

Nos jogos, o realismo começou a ser almejado na década de 1990, quando gráficos 3D viraram o padrão da indústria. Nas décadas anteriores, devido às limitações gráficas, ele não foi tão desejado, mas ainda fascinava os jogadores se um dia isso seria possível num jogo de videogame. Com o avanço dos *hardwares* em cada geração de consoles e computadores, o realismo é ainda sem dúvidas o visual mais desejado em jogos modernos. Com isso, as pessoas começaram a dizer que o quanto mais foto realista as imagens são, melhores elas são [6]. Mesmo sendo almejado, nem sempre é bom ter gráficos realistas em todos os gêneros de jogos. “Por exemplo, gráficos realistas funcionam melhor apenas com gêneros específicos de jogos, incluindo esses, mas não apenas esses: aventura em terceira pessoa, tiro em primeira pessoa, simuladores, jogos de corrida, *survival horror* e jogos de esportes” [2].

É difícil atingir o fotorrealismo de verdade com o hardware atual. Tim Sweeney, fundador da Epic Games, disse que 40 *teraflops* são necessários para atingir cenários foto realistas dinâmicos [3]. *Teraflops*, é uma unidade de medida usada para medir o

*e-mail: vcoletto@outlook.com

desempenho computacional da unidade de pontos flutuantes (*floats*) de um processador [7]. O "*tera*" antes do nome quer dizer que o processador calcula um trilhão de operações por segundo, concluindo, 40 *teraflops* são 40 trilhões de cálculos por segundo. Uma placa de vídeo nova da Nvidia, RTX 3090 consegue executar perto dos 36 *teraflops* de poder computacional. Percebe-se o avanço computacional quando comparado ao Playstation 4, que quando foi lançado em 2014 possuía apenas 1,4 *teraflops* e hoje o Playstation 5 possui 10 *teraflops*.

Pode-se dizer que em um jogo realista os gráficos podem ser divididos em 2D e 3D, os 2D sendo mais utilizados nas áreas das interfaces, em que o jogador irá interagir, e efeitos de partículas; já o 3D pode ser subdividido em personagens e cenários.

2.3 Fotorrealismo em Personagens

Hoje pode se dizer que é atingível o fotorrealismo em personagens, com filmes usando dublês digitais, como na Figura 1 no filme Blade Runner 2049, e isso mudou o modo com que nós pensamos sobre atores, atuação, entretenimento e jogos de computadores [8]. Mas filmes com CGI, são renderizados em computadores superpotentes, onde pode levar semanas até completar a renderização do filme todo, como foi no caso do *Toy Story 4* (2019) da Disney Pixar, em que apenas um frame demorava entre 60 e 160 horas devido a tamanha complexidade presente no projeto [9]. Um jogo é renderizado em tempo real, em sua maior parte, pode conter pequenas *cinematics* (*cutscenes*) que possuem gráficos pré-renderizados, mas são mais utilizados para contar a história do jogo. Para ser realista, o jogo deve possuir gráficos realistas em tempo real, onde o jogador interage com o mundo a sua volta, assim tendo uma experiência mais imersiva.

Figura 1: CGI aplicada para criar humanos realistas em filmes. Fonte: Blade Runner 2049.

Em personagens é difícil atingir o fotorrealismo em tempo real, pois há muitos pequenos detalhes que juntos criam um ser humano, desde como a luz interage com a pele até micro animações dos músculos da face, que requerem muito processamento computacional para serem executados em tempo real. Por isso que os desenvolvedores de jogos não buscam chegar no fotorrealismo em personagens. Quando é vista uma representação humana por humanos, ela é categorizada em duas seções: representações artísticas de humanos e humanos reais [10].

Por exemplo, o personagem Mario da Nintendo, é uma representação artística de um humano, e mesmo sendo um gráfico estilizado ainda é possível se identificar com ele, pois ele está nessa categoria onde existe muito a Suspensão da Descrença, em que temporariamente permite-se acreditar em algo que não é verdadeiro, especialmente para apreciar uma obra de ficção [11]. Mas quando a representação de um personagem busca chegar o mais perto do real, fica cada vez mais difícil suspender essa descrença, o subconsciente passa a identificá-los como humanos de verdade, onde há mais padrões rígidos a serem seguidos. E quando um desses padrões não é atingido, o personagem adentra o *Uncanny Valley*, que é um termo cunhado pelo roboticista Masahiro Mori para descrever a sensação estranha que um espectador pode experienciar quando encontra robôs quase humanos [12], também utilizado nos casos dos humanos digitais. O espectador possui uma memória extraordinária quando o assunto é como as pessoas são e se movem.

A Figura 2 apresenta um gráfico sugerindo que um robô precisaria parecer e agir quase perfeitamente como um humano, se não ele entrará no vale. Ultrapassar o limite do *Uncany Valley* em jogo é algo impossível devido à complexidade de um ser humano digital, por isso os artistas preferem fazer algo mais estilizado, tentando não entrar nessa zona de rejeição.

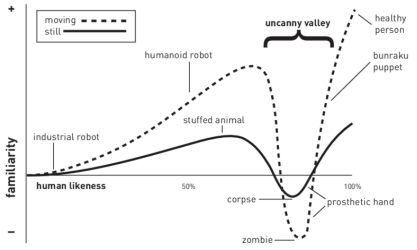


Figura 2: Gráfico representando o *Uncanny Valley*. Fonte: Masahiro Mori.

2.4 Fotorrealismo em Cenários

Para a criação de cenários realistas, há quatro fatores a serem analisados: número de polígonos, textura, iluminação e animação. [13]. Começando com os gráficos em si, o número de polígonos a serem renderizados na tela ao mesmo tempo foi drasticamente aumentando conforme a evolução dos *hardwares* foiacontecendo. Com placas de vídeos mais potentes, mais memória disponível e um maior processamento, foi possível quebrar as limitações existentes para criar jogos realistas. Com mais poder, é possível colocar um detalhamento maior em áreas onde antestinham que ser ignoradas por não ter o custo computacional necessário. Recentemente, um padrão para a criação de cenários foto realistas está sendo usado, são os *Scans 3D* ou fotogrametria de *assets*.

Para criar um mundo hiper-realista, são necessários *assets* hiper-realistas, e nada consegue superar os *assets* do mundo real para esse fim [14]. Para colocar objetos, localizações e coisas do mundo real num jogo, é utilizada a fotogrametria, que é uma técnica que envolve tirar várias fotos de objetos ou locais, com o objetivo de ter uma visão de cada um dos ângulos dos sujeitos. Depois disso, as fotos são colocadas num software de renderização 3D, que constrói texturas 3D realistas dos modelos fotografados [2]. Os resultados dessa técnica, representados na Figura 3, são surpreendentes, e ela pode ser tanto utilizada em jogos como em *renders* e CGI utilizados em filmes e séries.

Figura 3: Tronco de árvore criado em 3D por meio da técnica de fotogrametria. Fonte: Book of the Dead Demo.

Desde uma pedra, ou árvore, até casas e prédios, a fotogrametria consegue criar uma réplica digital exata, pronta para a implementação, seja numa *engine* ou qualquer outro *software* 3D. É possível criar cada objeto manualmente num *software* de modelagem 3D, mas com objetos naturais mais complexos isso se torna mais complicado, e com o uso dessa técnica, é possível alcançar resultados mais convincentes.

Não é apenas com objetos realistas que se criam cenários, a iluminação é de extrema importância para atingir o realismo, e há duas técnicas que estão sendo utilizadas hoje em dia: a rasterização e o *ray tracing*. A rasterização é a mais antiga, é um algoritmo que renderiza uma cena 3D numa tela 2D [15]. A iluminação e sombras são simuladas e calculadas separadamente a partir de fontes específicas de luz. Já o *ray tracing* calcula cada raio de luz e ilumina a cena inteira, ou seja, cada raio de luz reflete de um objeto ao outro, criando uma cena complexa e realista [16]. *Ray tracing* já era utilizado em *renders* em *softwares* 3D, mas começou a virar uma realidade nos jogos recentemente com o avanço da tecnologia, pois as placas de vídeo atuais conseguem calcular cada raio de luz em tempo real a vários *frames* por segundo. As duas técnicas ainda são utilizadas, mas para alcançar o maior realismo possível, o *ray tracing* é a melhor opção.

2.5 Ferramentas de Desenvolvimento

Para a criação de cenários ou qualquer outro item em 3D, existem vários *softwares* e técnicas que podem ser utilizadas durante o processo para atingir os objetivos. Nessa sessão serão citados programas, técnicas e termos utilizados para auxiliar na compreensão dos processos de criação explicados neste trabalho.

2.5.1 *Softwares* de Modelagem 3D

Softwares de modelagem 3D são programas que representam tridimensionalmente um objeto na tela, onde o usuário consegue modificar e assim modelar o objeto ao seu desejo [17]. Os mais utilizados são o 3ds Max [18], Blender [19] e Maya [20]. Estes conseguem simular tecidos, mas para a criação de roupas e simulações, pode ser utilizado o Marvelous Designer [21], que permite o usuário a costurar digitalmente tecidos e simulá-los em objetos 3D.

2.5.2 Criação de Texturas

Para a criação de texturas existem bibliotecas foto realistas, com centenas de modelos para escolher e importar, como a Quixel Megascans [22]. Mas para fazer uma textura ser diferente e se adequar a projetos específicos, existem programas para editar texturas, similares a um editor de fotos como Adobe Photoshop [23], tais como o Quixel Mixer [24] e Substance Painter [25]. Neles, o artista consegue modificar vários parâmetros e misturar texturas para criar o visual desejado, já visualizando o resultado final na peça em 3D.

Durante o processo de texturização, para construir uma textura em 3D, existem várias camadas chamadas de mapas, que juntas formam o material (Figura 4). Cada mapa é uma imagem 2D que representa o material e são aplicadas no objeto em 3D utilizando o mapeamento UV (*UV Maps*). Existem 7 delas: a *Diffuse*, *Albedo*, *Ambient Occusion* (AO), *Normal* (ou Bump), *Displacement*, *Reflection* (ou Specular) e *Gloss* (ou *Roughtness*).

Diffuse é a cor ou foto padrão do material, é o mesmo resultado caso o material fosse fotografado. *Albedo* é similar a *Diffuse*, mas com a vantagem de que as sombras e luzes foram removidas, e dependendo da textura, pode-se escolher entre uma ou outra. *Ambient Occusion* (AO) são as sombras do material, podendo ser usada junta com a *Diffuse* para sombras mais definidas. *Normal* ou *Bump*, que podem possuir os nomes em programas diferentes, sendo que este mapa possui 3 cores (vermelho, azul e verde) juntas numa imagem, em que cada uma representa um eixo de direção diferente (x, y e z), auxiliando o computador a entender forma. O mapa *Normal* é utilizado para acrescentar detalhes em 3D ao objeto. *Displacement* é usado para deformar a malha 3D, usada em superfícies como chão e pedras, aumentando significativamente o detalhe do objeto. *Reflection* ou *Specular* mostra estados onde pode e não pode aparecer reflexo na textura, usada apenas em alguns programas. *Gloss* ou *Roughness* controla a nitidez das reflexões, sendo que os dois termos são intercambiáveis - são o opostos um do outro [26].

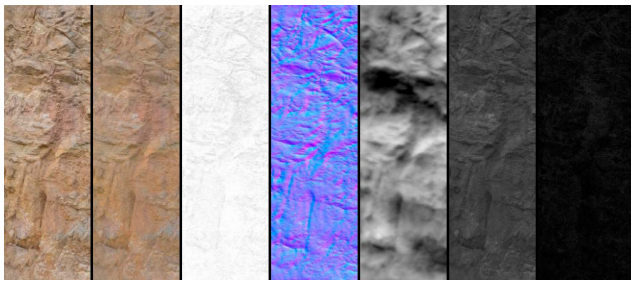


Figura 4: Todos os mapas de texturas respectivamente da esquerda para direita: *Diffuse*, *Albedo*, *Ambient Occusion* (AO), *Normal* (ou *Bump*), *Displacement*, *Reflection* (ou *Specular*) e *Gloss* (ou *Roughtness*). Fonte: Poliigon.

2.5.3 *Engines*

No mundo dos jogos, existem as *engines*, que são um produto multiuso com vários recursos importantes para o desenvolvimento de jogos, incluindo física, ferramentas de animação, inteligência artificial e ferramentas de programação [27]. As mais conhecidas são a Unreal Engine [28] e Unity [29], devido ao fato de serem gratuitas.

2.5.4 Termos

Existem alguns termos utilizados por artistas e desenvolvedores que são importantes serem definidos. *Assets* são como peças e itens de arte utilizadas no projeto. *Low-poly* e *high-poly* são termos que informam o número de polígonos que um *asset* 3D possui, sendo que quando há muitos polígonos ele é chamado de *high-poly*, e quando há poucos *low-poly*. Isso influencia no tamanho do *asset* para o projeto, pois um *high-poly* possui mais detalhes que um *low-poly*. *Props* é o termo utilizado para um *asset* que é uma peça de tamanho pequeno na cena, como por exemplo uma chave de fenda, celulares e controles.

Paintover é o processo de desenhar ou pintar por cima de uma foto, seja por um meio físico ou digital, podendo ser um esboço por cima, adicionando novas ideias, ou uma arte finalizada. *Decals* são materiais projetados numa superfície existente, geralmente são pequenos detalhes como rasgaduras ou buracos de balas, são "borrifados" no local desejado.

Baking é um termo utilizado na Computação Gráfica que pode ser usado em vários processos diferentes, como texturização, animação e luz, mas que geralmente consolida um sistema de dados em uma forma simplificada e mais permanente [30]. Em outras palavras, *Baking* é uma gravação de um resultado de um processo gráfico em outro meio, facilitando o seu uso computacional.

3 Trabalhos Relacionados

Estes dois trabalhos possuem estilos de arte diferentes, mas os dois utilizam o fotorrealismo de alguma forma. O primeiro busca comparar o fotorrealismo e o não-fotorrealismo com pesquisas e

estudos de caso com estilos de arte diferentes, buscando justificar suas descobertas. Já o segundo procura fazer uma remasterização foto realista de um mapa de um jogo clássico. O importante dos trabalhos são os meios em que o processo foi documentado e apresentado ao leitor.

3.1 *Photorealism Versus NonPhotorealism: Art Styles in Computer Games and the Default Bias.*

Na tese de 2013 do Nathan Jarvis [27], são comparados o fotorrealismo e o não-fotorrealismo em jogos, buscando entender a razão do porquê gráficos foto realistas são mais utilizados nos jogos atuais, para no fim tentar criar meios de trazer um balanço entre os dois estilos. Isso é feito por meio de pesquisa, com fontes tanto acadêmicas como jornalísticas, buscando encontrar teorias e opiniões sobre o assunto, com experimentos para validar essas teorias.

Foram encontradas duas possíveis razões. A primeira de que simplesmente o fotorrealismo é o estilo de arte mais popular entre os consumidores. Então foi feita uma pesquisa para saber as opiniões em grupos de consumidores, e o resultado foi de que nenhum dos dois estilos é mais popular que o outro, e que de fato os dois são favorecidos igualmente. A segunda possível razão é de que o fotorrealismo é o estilo de arte mais simples de se produzir. Nathan continua afirmando que o fotorrealismo tem um encaixe perfeito em renderizações 3D pelo fato de os dois serem formas de arte mais técnicas, já o estilo não-fotorrealista requer mais atenção do artista, pois é mais difícil de um computador recriar, e há muitas peças a serem movidas e modificadas para chegar num estilo mais estilizado. Esse fenômeno é chamado de *default bias*, onde os *softwares* 3D e *engines* de jogos possuem mais ferramentas e um aprendizado melhor para o estilo do fotorrealismo. Para testar isso, Nathan, fez vários experimentos tentando criar estilos não-foto realistas na *Unreal Development Kit* (UDK), e ao mesmo tempo que desenvolvia, ele documentava os meios e técnicas para buscar alcançar os objetivos dos casos especificados. Como resultados, há falhas nos modos em que algumas ferramentas foram desenvolvidas, que limitavam a criatividade ao fazer qualquer desvio do fotorrealismo.

O mais interessante na tese do Nathan são os experimentos que buscavam criar os estilos não-foto realistas e sua documentação com fotos durante o seu desenvolvimento, criando um passo a passo para a criação desses estilos. O estudo deste trabalho será de grande auxílio, devido ao modo como a documentação foi elaborada e as análises realizadas ao criar o *workflow* foto realista.

3.2 *Democratizing 3D Environment Workflows*

No artigo escrito por Adan Chaumette [5], é explicado como *assets* escaneados podem ajudar na criação de cenários foto realistas, que antes eram inimagináveis, principalmente para desenvolvedores *indie*, pois a criação de *assets* detalhados eram muito caros devido ao grande tempo consumido em sua criação e na captação de *scans* realistas. Além disso, é detalhado o processo de criação de um cenário remasterizado, que neste caso é o mapa Dust II do jogo Counter-Strike, feito pelo Wiktor Öhman.

Com ajuda da técnica de fotogrametria para se ter as exatas dimensões do mapa original e ajuda no primeiro esboço da modelagem 3D, e de texturas e materiais da biblioteca da Quixel Megascans, com apenas algumas texturas criadas separadamente, Wiktor recriou a Dust II (Figura 5) com um novo visual foto realista, dando um ar totalmente diferente ao cenário. E todo o processo foi documentado com imagens e explicações das soluções para os problemas encontrados no caminho, bem como o *workflow* com algumas ferramentas.

Figura 5: Comparação entre o cenário da Dust II (acima) e sua remasterização (abaixo). Fonte: Adnan Chaumette.

A importância desse artigo é de mostrar como foi o processo de criação, ou de recriação, de um cenário usando a fotogrametria e *assets* da Quixel Megascans, que também será utilizado neste trabalho, mas não exclusivamente. É possível ter uma ideia geral do modo com que foi desenvolvido, mas não com grandes detalhes, algo que será levado em conta na descrição do *workflow* a ser desenvolvido neste trabalho.

4 Identificação do Problema e Proposta de Solução

Um dos grandes problemas identificados na área de desenvolvimento de jogos é a falta de uma metodologia completa para a criação de cenários foto realistas para jogos em fontes como artigos e teses. A proposta deste trabalho é a criação e mesclagem de diferentes metodologias em um *workflow* unificado e mais detalhado para a criação de cenários foto realistas, desde sua concepção até finalização numa *engine* de jogos.

5 Metodologia

A proposta deste trabalho é a elaboração de um *workflow* para a criação de cenários, podendo ser utilizada em diferentes estilos, como o fotorrealismo. Deste modo, inicialmente foi feita uma pesquisa sobre fotorrealismo nos jogos, pois são importantes para a compreensão da evolução da criação de *assets* realistas. Após, foram estudados alguns trabalhos, com a demonstração de *workflows* para diferentes estilos, tanto foto realistas como mais artísticos, e será apresentado um *workflow* almejando ser o mais completo para a criação de cenários realistas em jogos, adicionando passos não mostrados pelos trabalhos relacionados. E por fimserá elaborado um estudo de caso, validando o *workflow* desenvolvido neste trabalho junto de uma *cinematic*, um vídeo mostrando detalhes do cenário elaborado.

5.1 Workflow

Um *workflow* ou fluxo de trabalho ou *pipeline* de produção, é onde ocorre o desenvolvimento do projeto desde sua idealização até sua finalização concreta [17]. O *workflow* proposto neste trabalho é dividido em dez partes, abrangendo desde a criação da ideia até a sua visualização final. A ideia geral por trás desta metodologia é a abordagem por camadas, indo desde as partes maiores do cenário como paredes e pilares, até itens menores como chaves e controles, adicionando detalhes a cada passo.

As referências para este *pipeline* de produção são baseadas em estudos realizados sobre o *workflow* do desenvolvimento de

cenários 3D de Wiktor Öhman [5], que consiste do uso de programas da Quixel, e baseado também em um painel apresentado na Gnomon - escola de artes visuais especializada na educação de computação gráfica para carreiras na indústria do entretenimento. Neste painel, artistas da indústria de jogos compartilharam dicas e técnicas utilizadas em seus trabalhos em empresas gigantes do entretenimento como Blizzard, EA DICE e Naughty Dog [31].

Este *workflow* será desenvolvido para a criação de cenários, porém, é difícil generalizar um *workflow* para todos os tipos de cenários, já que, por ser um *design*, irá envolver a solução de problemas durante suas etapas. Haverão problemas específicos em cada situação com respostas únicas. A ideia geral é apresentar uma estrutura que pode ser adaptada a diferentes tipos de cenários, e é esse o objetivo desta metodologia.

5.1.1 Primeiro Passo: História

Para poder criar uma história, precisa-se visualizar o mundo que será criado, com o intuito de gerar novas ideias para preencher o cenário e também descartar as que não funcionam com as regras desse mundo. Com isso, inicialmente deve ser criado um pequeno parágrafo contendo uma história, sem muita complexidade, mas com argumentos que irão ajudar a entender este mundo. Por exemplo:

"Num futuro distante a humanidade está sob ameaça de monstros gigantes e a única arma efetiva contra estas criaturas são robôs mechas gigantes pilotados por uma equipe de elite. Há estações de treinamento e despache de patrulhas mechas ao redor de cidades protegidas por gigantes domos de vidro. Essas estações são extremamente tecnológicas, com oficinas para a manutenção de robôs, salas de reuniões, de descanso, de treinamento e de pesquisa, onde fica a inteligência do exército."

A história é importante para poder limitar a criação, com o intuito de criar soluções dentro dessa limitação. Além disso, ao final do projeto, quando um jogador andar pelo cenário, será perceptível que tudo foi idealizado para encaixar em seu devido lugar, fazendo sentido no mundo criado.

5.1.2 Segundo Passo: *Concept Art* e Referências

De costume, na maioria das empresas criativas, há *concept artists*, artistas que criam o *concept*, ou conceito, do que será realizado, dando uma visão e o caminho a ser seguido no projeto. Um *concept* é necessário justamente para dar uma base, uma ideia, quando será modelado. É uma representação visual criada com base na história desenvolvida no último passo. Para facilitar essa criação, o uso de referências visuais similares é altamente recomendado, usando sites como Pinterest [32] e ArtStation [33] como fontes de pesquisa. Outros cenários, lugares reais, filmes, séries, desenhos, outros *concepts*, texturas, imperfeições, tipos de iluminação, paletas de cores, são algumas referências e pontos a serem pesquisados para se criar um cenário.

Após esta pesquisa, serão aplicadas referências num *moodboard*, um quadro onde há presente todas as referências num local só, de rápido acesso. Aplicativos como PureRef [34] são auxílio nas tarefas, nele é possível arrastar cada imagem e separá-las num quadro digital infinito e organizar por seções.

Finalmente, iniciam os desenhos da cena. Recomenda-se que seja em dois pontos de perspectiva, para possuir uma melhor visualização da tridimensionalidade do local. É possível criar mais de um desenho, alguns dando ideias mais gerais do cenário e outros focando em pequenas partes, adicionando mais detalhes a elas. Uma outra opção, é fazer um esboço do cenário em 3D, usando apenas formas básicas numa *engine*. Outros pontos importantes a serem levantados, são silhueta e tempo. Além dos pequenos detalhes, a silhueta é importante para dar um dinamismo maior a cena, dependendo do mundo criado. Caso seja uma sala, não será de grande importância; já uma cena de céu aberto, considere a sua silhueta. Um tempo maior deve ser gasto nesta etapa para desenvolver e possuir a ideia geral do que é exigido do cenário em si, ter um *concept* sólido é obrigatório para continuarnos próximos passos.

5.1.3 Terceiro Passo: Blockout

Na criação de um cenário, há a necessidade de otimizar os processos para economizar tempo na produção, por isso, ter *kit* modulares é uma boa ideia no início da modelagem. Independente do cenário ser orgânico, possuindo mais natureza, ou inorgânico, tendo mais construções humanas, ainda é possível criar peças e construir uma cena. Num programa de modelagem 3D, como 3ds Max, Maya e Blender, inicia-se criando as maiores peças do cenário, como paredes, tipos de chãos, tetos, escadas, janelas, pilares, que são o esqueleto do cenário. Caso seja um cenário mais orgânico, julga-se o mais necessário para criar a silhueta do cenário idealizado nas etapas anteriores, como árvores, montanhas e pedras. O importante destas peças criadas são sua conectividade entre elas. Almejando isto, deve ser definida uma distância de largura das peças, como por exemplo dois metros, usando isto como uma linguagem padrão entre elas. Caso seja necessário, na montagem, a modificação ou alteração de uma das peças, será apenas preciso a mudança desta peça específica e então encaixar outra no mesmo local.

Não é necessário que o *kit* modular seja muito grande, é importante começar do básico, e dependendo do cenário, modelar as outras peças de modo a remover o visual modular simples, respeitando a linguagem antes criada para que se conectem. É importante lembrar que são peças simples, portanto, não possuem muitos detalhes.

O primeiro cenário criado nem sempre será a melhor opção final. Com as peças, há várias possibilidades de montagens diferentes. Neste momento inicial, deve-se testar a maioria das possíveis soluções para identificar alguma que se encaixe nos objetivos de cenário criados anteriormente, mesmo que não seja idêntico ao *concept art*. Lembrando que o *concept* é criado com o intuito de dar uma direção ao que será modelado. É importante também testar as modelagens fazendo uma exportação para a *engine* escolhida para o projeto.

5.1.3.1 Adicionar *Hero Assets*?

Nesta parte do processo, é necessário se perguntar se é preciso criar *hero assets*, que são peças únicas, que são o foco da cena e que possuem mais detalhes, ou seja, são mais densas poligonalmente e ocupam mais espaço na memória. Justamente por serem o foco, elas demandam mais tempo para que sejam detalhadas e refinadas em relação as outras peças do kit modular. Caso julgue-se necessário a criação destas, começar imediatamente a modelagem delas é essencial. Por serem peças mais detalhadas, consomem muito tempo e podem atrasar a produção. Tendo isso em mente, focar em poucas *hero assets*, como uma ou duas, é recomendado. Outras peças como pilares e chão precisam também de um maior detalhamento, justamente por serem mais vistas pelo jogador, quando as *hero assets* forem finalizadas, segue-se a adicionar mais detalhes as peças do kit modulares escolhidas na versão final.

5.1.4 Quarto Passo: Materiais e Texturização

Neste estágio, o objetivo é criar texturas e materiais para os objetos grandes, como paredes e pisos, e tentar minimizar o número de repetições, focando na reusabilidade de texturas modulares em combinações diferentes. Finalizar as texturas não é necessário, mas criar as *Normal Maps* e os AO (*Ambient Oclusion) Maps* para se

ter uma rápida representação e analisar o conjunto na cena sim. Estes materiais serão finalizados numa etapa futura.

A técnica de *baking* (Figura 6) é extremamente útil em criações de mapas de normais. *Baking* consiste em aplicar detalhes de uma versão *high poly* de um objeto numa outra versão *low poly* do mesmo através dos mapas de textura, assim economizando vértices e mantendo o detalhamento. Uma recomendação é utilizar a ferramenta *render to texture* no 3ds Max, para criar rapidamente *bakes* de objetos.

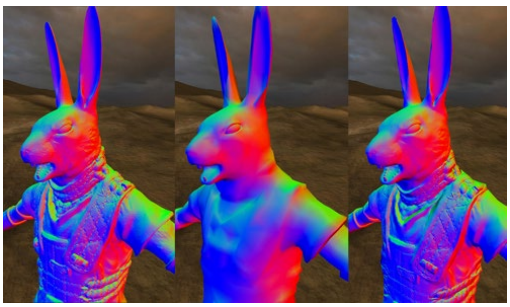


Figura 6: Processo de *baking*: a esquerda, versão *high-poly*, no centro a versão *low-poly* e a direita a versão *low-poly* com o mapa de *normal*. Fonte: Wolfire Games.

5.1.5 Quinto Passo: Cores e Iluminação

A adição de cores é umas das partes mais importantes para dar característica a um cenário. Geralmente a pesquisa sobre cores é concluída na etapa de *concept*, mas caso seja necessário, é possível pesquisar novamente referências das mesmas. O uso de *paintovers* são sempre uma boa solução para testar novas ideias, buscando responder as seguintes questões: quais cores usar? Quais *props*? Que tipo de iluminação? O ideal é elaborar *paintovers* com algum *software* utilizado para desenho digital, como o Adobe Photoshop, pintando rapidamente sobre uma captura de tela do cenário e testar suas novas ideias, sem gastar muito tempo.

Para descobrir paletas de cores, sites como Adobe Color [35] ajudam a explorar e encontrar paletas com cores publicadas pela comunidade que combinam com o projeto. Pode-se também utilizar uma imagem, seja ela uma das referências antes pesquisadas, e a ferramenta mostrará a palheta de cores presente. As cores são importantes porque realçam o cenário, mas caso haja muitas cores que não combinam, elas podem acabar atrapalhando e deixando um visual desagradável. Importante realçar que haverá um personagem neste cenário e é a responsabilidade do cenário em fazê-los brilharem. Durante o processo de criação da paleta de cores, deve-se buscar cores principais, secundárias e cores para as luzes.

Para a definição da iluminação, deve-se fazer o teste de possíveis fontes de luz em softwares de desenho digital com uma imagem do cenário e analisar seguindo as regras do mundo criado na história, com base nas referências e nos *concepts* desenhados. É importante levar em consideração o teste de diferentes tipos de ângulos de luz, analisando as luzes e as sombras geradas. Também deve-se lembrar de inserir fontes de luz direcionadas a lugares que possuem *hero assets*, já que elas são o foco da cena.

5.1.6 Sexto Passo: *Backdrop*

O *backdrop* é o fundo que aparece nas janelas e em lugares longes do cenário. Existe para situar o jogador em que mundo ele está presente. O *backdrop* pode ser tanto a *skybox*, que são imagens 2D que juntas se transformam no céu visível no jogo, como algo modelado em 3D, mas preferencialmente é conjunto dos dois. Além de criar *skybox*, é possível fazer o download de céus em sites como HDRIHaven [36] e utilizar em seus cenários. O *backdrop* possui *assets* simples que geralmente ficam longe do jogador, por isso não é necessário adicionar muitos detalhes pequenos a elas. O importante de lembrar ao criar esses *assets* simples são as suas silhuetas e sua cor na composição do fundo.

Em jogos, há vários modos de enganar o jogador que o mundo é totalmente detalhado em 3D. Prédios a longa distância viram imagens 2D com resoluções baixas e *assets* modeladas possuem apenas a face visível ao jogador. Também não há a necessidade de adicionar mapas de texturas com normais para adicionar detalhes, pois o objetivo é economizar tempo e priorizar o que o jogador consegue ver de perto, focando recursos nessas áreas.

5.1.7 Sétimo Passo: Props

Props são objetos pequenos que ajudam no detalhamento da cena e no mais importante, a escala. Utilizando as referências e o *concept* criados anteriormente, para definir a lista de *props* a serem modeladas. Nesta etapa deve-se criar principalmente, objetos que mostrem a escala, como telas de computador, teclados, corrimões, objetos que demonstram rapidamente a escala ao jogador. São essencialmente necessários no cenário quando não há nenhum personagem na cena, pois são objetos de fácil reconhecimento, já que são comuns no nosso cotidiano, e as pessoas vão buscar por esses objetos para entender a dimensão da cena.

Lembrando que é possível voltar um passo atrás e fazer um outro *paintover* caso esteja ficando sem ideias para a cena.

5.1.8 Oitavo Passo: Finalizando Todos os Materiais

Para a finalização dos materiais antes já criados, é importante lembrar das imperfeições antes já buscadas nas referências. É neste estágio em que serão adicionados todos os pequenos detalhes a cena. Utilizar programas como Quixel Mixer e Substance Painter são ótimos para adicionar mais detalhes as texturas, pintando diretamente ao objeto 3D.

Na Unreal Engine é possível criar *masters materials*, que funcionam como uma instância do material que, quando alterada, a mudança ocorre em todos os objetos que possuírem este material. E adicionando detalhes a este material e com o uso de máscaras, é possível criar diferentes tipos de texturas com os mesmos mapas em um mesmo *master material*. Por exemplo, numa parede que foi adicionado um material de gotas d'água por cima com uma camada com máscara, é possível pintar apenas onde as gotas irão aparecer no material, e é possível repetir isso colocando as gotas em diferentes posições pelo cenário.

Bibliotecas de texturas como o Quixel Megascans são ótimos em trazer referências, e com o uso do Quixel Mixer é possível misturar texturas e criar algo único e exportar diretamente para a *engine* ou programa 3D desejado. O uso de *decals* também são recomendados para adicionar imperfeições aos materiais.

5.1.9 Nono Passo: Finalização

Na finalização, serão adicionados detalhamentos extra, coisas que irão preencher a cena, completando o seu visual. Podem ser coisas simples como luzes piscando, folhas se mexendo, correntes de água, partículas de vapor, raios de sol, névoa. Dependendo do cenário, julga-se o que irá complementar o visual.

Finalizar a iluminação é uma das partes mais importantes, pois é necessário compreender como o sistema de luz funciona no cenário e replicar os melhores meios pela cena. Utilizando os *paintovers* criados no quinto passo, para criar estes pontos de luz ou até mesmo pode ser adicionando mais pontos de luz, chamando atenção em partes do cenário, analisando suas intensidades. É possível também adicionar iluminações falsas para preencher mais a cena, como por exemplo o sol, que geralmente é a fonte de luz principal do cenário, porém, há partes onde está pouco iluminado, e aumentar a

intensidade do sol irá atrapalhar o balanço de cores, gerando um desequilíbrio de branco. Com isso, adicionar pequenas luzes para dar uma iluminação maior em pequenos locais complementam o visual. Deve-se ter uma atenção redobrada para não acabar com a credibilidade da iluminação, pois estas luzes extras não são para serem percebidas pelo observador.

5.1.10 Décimo Passo: Otimização na *Engine*

Neste passo serão apresentadas algumas técnicas que podem ser utilizadas na *engine* para otimizar a experiência do jogador. A otimização deve estar em mente durante todo o desenvolvimento já que com cenas mais complexas, facilmente o número de polígonos sobe e a demanda gráfica sobe proporcionalmente. A otimização entra em ação buscando entregar a melhor experiência mesmo para os usuários que não possuem a melhor placa gráfica.

Baking 3D é sem dúvidas a técnica mais importante e mais utilizada nos jogos, tanto em cenários como em personagens. Tendo a possibilidade de manter os detalhes de um objeto *high-poly* e serem utilizados em outro *low-poly*, isso torna-se uma grande vantagem e economiza recursos gráficos.

LOD, ou *level of detail* (Figura 7), é uma ferramenta usada na *engine* que, dependendo da distância entre o jogador e o objeto, a *engine* irá renderizar menos polígonos. Se o jogador estiver perto, serão renderizados todos os polígonos do objeto, e conforme o jogador vai se distanciando, a *engine* irá renderizar menos polígonos, mas mantendo a silhueta do objeto e assim usando menos recursos e focando apenas nas áreas onde o jogador consegue ver os detalhes.

Figura 7: Diferentes níveis de LOD numa abóbora em 3D. Fonte: Quixel.

Em objetos pequenos como plantas e gramas, desligar o *real time shadows*, a aplicação de sombras em tempo real do objeto, irá economizar recursos gráficos. O jogador raramente percebe a mudança, justamente por serem objeto pequenos. Dependendo da cena, é possível testar em outros objetos e verificar o ganho de performance e analisar se vale a pena ou não.

A visibilidade do jogador é algo que pode ser otimizado, pois nem sempre é necessário renderizar todos os objetos da cena, mas sim apenas os quais o jogador consegue ver. Deve-se desligar a renderização dependendo da posição do jogador e se caso ele está visualizando o objeto diretamente ou o objeto está atrás de outros.

Para a iluminação global é feito com o uso de *light maps*, que são luzes e sombras pré-calculadas em mapas que são utilizados nas cenas. O único problema é que isso impossibilitará a mudança dinâmica de luz [37].

No futuro teremos que fazer cada vez menos a otimização dos jogos com a chegada de novas tecnologias, como a Nanite e a Lumen da Epic Games. Com a Nanite é possível usar *assets* com milhões de polígonos e não ter uma grande perda em desempenho, sem a necessidade de LOD e *normal maps*. Já com a Lumen é possível ter luzes globais dinâmicas com suas sombras e reflexões calculadas e exibidas em tempo real, sem o uso de *light maps* e com a possibilidade de movê-las. Porém, essas tecnologias chegarão apenas em 2022 com o lançamento da Unreal Engine 5 [4].

6 Estudo de Caso e Validação

Para a validação do *workflow* proposto neste trabalho, será desenvolvido um estudo de caso utilizando o mesmo, com a criação de um cenário foto realista num motor de jogos. O motor escolhido foi a Unreal Engine 4, para a montagem final da cena junto com suas texturas, iluminações e efeitos adicionais. A Unreal foi escolhida por apresentar mais ferramentas de fácil uso para artistas de cenários comparada com a Unity, outro motor de jogos.

A seguir, será apresentado o passo-a-passo, com a resolução dos problemas encontrados durante seu processo da criação do estudo de caso. Como ele consiste em um cenário avulso, não fazendo parte de um jogo completo, o início do desenvolvimento criativo do *workflow* foi alterado, iniciará pelas referencias e depois a história. O estudo de caso será montado na Unreal Engine, mas o seu uso final será numa *cinematic*.

6.1 Referências

A primeira parte foi a busca de referências, para dar inspiração ao trabalho, já que o mesmo ainda não tinha sido definido completamente. No apêndice A, é apresentado o *moodboard* para o cenário feito no PureRef [34], com referências de desertos e cânions encontrados no ArtStation [33] e Pexels [39].

Após isto, a busca por um *hero asset*, um personagem principal da cena, foi iniciada. Buscou-se algo para dar contraste com o cenário, para torná-lo interessante. A solução encontrada foi em um monólito, pois é uma peça de arte que foi colocada no deserto e em outros lugares do mundo em 2018 e chamou muita atenção da mídia [40]. Foi necessário pesquisar as suas dimensões, fotos com mais detalhes de texturas e imperfeições, já que este *asset* será o *hero asset* e terá que ser modelado. No Apêndice A contém o *moodboard* completo com as referências para o cenário e para o monolito.

6.2 História

A história desenvolvida para este cenário não exigiu muita complexidade nem muitos elementos de construção de mundo, pois se refere a um ambiente real que existe no mundo. Desta forma, a história desenvolvida foi:

"Um deserto em Nevada nos EUA, um lugar seco em que já se passou por anos com chuvas e a erosão das pedras se desgastando com o tempo, que onde um dia apareceu um objeto não identificado de 3 metros feito inteiramente de metal."

6.3 *Blockout*

Na Unreal, a primeira parte foi criar um plano 3D; adicionar uma câmera fixa e utilizar os atalhos de *bookmarks* da Unreal para retornar à posição da câmera pré-definida; utilizar e deformar cubos em 3D para definir a silhueta do cenário; modificar a posição do sol já presente na cena. A partir deste *setup*, foi criado o primeiro esboço em 3D do cenário. Foi utilizada a câmera fixa para verificar como está ficando a composição, o ajuste da posição do sol foi feito para definir as sombras e foram adicionados cubos nas partes onde seriam as paredes de pedras do cenário. Utilizando as ferramentas de composição da *viewport* para facilitar a montagem. Exemplos de *blockout* das versões iniciais no Apêndice F.

6.3.1 Definição dos *Assets*

Para este projeto, ao invés de modelar cada uma das pedras, foi utilizado a biblioteca de *assets* e texturas do Quixel Megascans, utilizando uma conta da *Epic Games*, que dá livre acesso a todos os *assets* gratuitamente. No Quixel Bridge (aplicativo para ter acesso a biblioteca do Megascans), os *assets* são separados em coleções, então todos os *assets* já funcionam juntos, já que fazem parte do mesmo pacote e foram retirados do mesmo lugar durante a

fotogrametria. A coleção usada neste estudo de caso foi a *Limestone Quarry* (Figura 8) [41]. Todos os *assets* presentes na biblioteca já foram otimizados para *engines* de jogos, com diferentes níveis de LOD e texturas em diferentes resoluções. Como o objetivo é o fotorrealismo, foram utilizadas texturas em 4K.

Figura 8: Alguns *assets* utilizados da biblioteca do *Megascans*.
Fonte: Quixel.

6.3.2 *Hero Asset* e Texturização

O *hero asset* dessa cena é o monólito, o personagem principal. Como ele não existe, a modelagem foi feita no 3Ds Max e as UV foram feitas no Autodesk Maya, que possui uma excelente ferramenta para a criação de UV ao invés do 3Ds Max. A texturização foi feita no Quixel Mixer, utilizando S*mart Materials* e texturas do Megascan*s*. *Smart Materials* são texturas que estão prontas e disponíveis no Mixer, em que é possível importá-las e modificá-las. O "*Smart*" no nome vem do poder de se adaptar a diferentes *assets* importados, mas mesmo assim, modificações são necessárias para atingir o visual desejado. Uma análise do passo-a-passo mais detalhada de como o *asset* foi modelado e texturizado está descrito no Apêndice C.

6.3.3 *Assembly*

Nesta parte, *assembly* ou montagem, será feita a troca dos cubos do *blockout* na Unreal Engine pelos *assets* finais do Quixel Megascans. Para importar os *assets*, é necessário ativar o *plug-in* do Megascans na Unreal, que é encontrado no menu de *plug-ins* na *engine*.

Todos os *assets* são importados ao projeto e em seguida à cena, para verificar se as cores de cada *asset* combinam, pois mesmo eles sendo da mesma coleção, podem ocorrer diferenças de cores entre os mesmos. Para mesclar as cores em cada material, é possível abrir e modificar as cores individualmente da textura. Para uma alteração mais precisa, pode ser utilizada a visão sem luz (*unlit*) da *viewport* da Unreal.

Após esta parte de integração, são feitos vários testes de *assets* diferentes e combinações no cenário, o que faz parte do processo. Importar, mover, duplicar, ajustar é a sequência desse processo. Uma das peças fundamentais foi posicionar *assets* dentro de outros, com pequenas partes expostas, para quebrar a repetição. É importante começar com as peças maiores e indo até as menores, ou seja: pedras grandes que mudam a silhueta do cenário; em seguida pedras menores aos redores das maiores; na sequência "panquecas" (Figura 9) de pedrinhas nas beiradas; e por fim a vegetação. Para entender como o cenário funciona, é necessário analisar as referências e perceber padrões a serem replicados no cenário, e este foi um dos padrões percebidos em cenários com muitas pedras.

Figura 9: *Asset* 3D de "panquecas" de pequenas pedras.
Fonte: Quixel.

6.3.4 Ground Painting

Para adicionar detalhes ao chão, com o *plug-in* do Megascans na Unreal Engine, é possível criar um *blend material*, que mistura 3 materiais em um único material. Com o menu de *painting* e com o material selecionado é possível pintar e ajustar a mistura entre elas no próprio *asset*, neste caso o chão, e adicionar detalhes em áreas específicas do *asset*. O pincel funciona diretamente no vértice, então é necessário um chão com uma boa quantidade de polígonos caso uma área com vários detalhes seja pintada. Uma outra *feature do blend material* é a adição de poças de água (*puddles*) e áreas molhadas, que adicionam um grau de realismo em pouco tempo de trabalho. Seguem exemplos no Apêndice E.

6.4 *Props*

Como o cenário é natural, a maioria das *props* são pedras e vegetação, que adicionam mais detalhes ao cenário. Não foi necessário modelar e texturizar *props* neste estudo de caso.

6.5 *Backdrop*

Nesta etapa, foram colocados *assets* que dão continuidade a cena em áreas que o jogador não consegue ir, que faz parte do *backdrop*. Além disso, foi adicionado um céu com nuvens em 3D que se movimentam em tempo real, que faz parte do sistema volumétrico de nuvens do Unreal, adicionado da versão 4.26. É possível controlar o tamanho das nuvens, sua quantidade, o tipo e suas velocidades, entre outros controles [42]. Junto com o sol, é possível criar diferentes tipos de céus realistas.

6.6 Finalização

Nesta parte foi feito principalmente o detalhamento de mais partes do cenário, como pequenos cantos com mais pedras. Sempre tentando não mostrar sinais de repetição e utilizando as referências como guia para criar mais detalhes em cantos no cenário final e por fim *decals* de sujeira em partes do cenário.

Para adicionar um realismo, a adição de VFX foi essencial. Dentre os VFX utilizados, cita-se partículas de poeira, névoa de areia e distorção de calor, utilizando apenas as partículas já criadas na *Starter Content* da Unreal Engine e modificando-as para este estudo de caso. Configurações como cor, tamanho, velocidade e área de *spawn* foram alteradas. Já a distorção de calor foi criada sem uma base. Outro detalhe adicionado às texturas de *assets* grandes, foram as *micro detail normal*, que se encontram nos *master materials* do Megascans, com eles é possível adicionar uma normal acima da normal do objeto e com elas adicionar mais micro

detalhes para as mesmas, tornando mais realistas. Exemplo de *micro detail normal* no Apêndice G.

6.6.1 Pós processamento

Para criar o fotorrealismo, é necessário adicionar imperfeições que acontecem em câmeras à renderização do cenário. Com um *Post Process Volume* é possível modificar a renderização com *ray tracing* e outras variáveis para melhorar sua qualidade. Criar uma câmera, ajustando sua lente, tipo de sensor, ponto focal e adicionar efeitos de câmera, como a aberração cromática, lentes sujas, *bloom, motion blur, noise* e *film grain*. Modificar as cores finais alterando a temperatura do ambiente e modificando o *color correction*. Neste caso foi aumentado a saturação dos vermelhos nos *highligts* e nas sombras também, utilizando fotografias de filmes e séries como referência, com o objetivo de deixar a cena mais quente. Exemplo do render final com o pós processamento na Figura 10.

Figura 10: *Frame* do *render* final. Fonte: elaborada pelo autor.

6.7 Otimização na *Engine*

Há vários modos de verificar se a otimização está boa. Neste estudo de caso, a grande maioria dos *assets* foram importados de uma biblioteca de *assets* já otimizados com LOD, o que ajudou a deixar a cena sem grandes problemas. Um dos maiores problemas foi a iluminação com *ray tracing* (RT), que foi utilizado como a única iluminação global (GI). A maioria dos jogos dessa geração utiliza o RT em conjunto com a iluminação global dinâmica rasterizada, o RT é utilizado em grande maioria em reflexos, *Ambient Occulsion* e sombras, mas não para iluminar a cena toda, que adicionaria um senso de volume maior a cena com um *shading* difuso, mas diminuiria bastante o desempenho da mesma [43]. Como o objetivo é alcançar o fotorrealismo, a opção de RT como GI foi usada, mesmo afetando a cena e diminuindo o FPS consideravelmente. Com o GI rasterizado, a cena é jogável com mais de 160 fps em 1080p, com o RT GI ela cai para 70 fps em 1080p e em 30 fps em 4K. É ainda possível realizar mais otimizações, como ajustar o tamanho das texturas para diferentes resoluções. O desempenho da cena não é crucial neste estudo de caso, já que seu fim é para uso de *cinematics* gravadas dentro da *engine*.

A Unreal Engine possui inúmeros filtros em sua *viewport* que auxiliam o desenvolvedor na hora de identificar problemas na cena, tais como verificar o *asset* com o número de polígonos maior, efeitos ou partículas que estão diminuindo a performance, texturas mal otimizadas, para citar alguns. Por padrão a *engine* já renderiza apenas os *assets* visíveis, então numa cena grande, quando mais dividida em pedaços os *assets* estiverem, melhor para o desempenho.

6.8 Cinematic

Para a gravação da *cinematic* foi utilizado a ferramenta *Movie Render Queue* na Unreal Engine. Primeiramente foram animadas várias panorâmicas no cenário com diferentes câmeras e ajustando o ponto focal de cada uma individualmente. Cada animação foi adicionada ao *Movie Render Queue*, onde é possível escolher o *output*, resolução, ajustar filtros e adicionar linhas de código para melhorar a qualidade do *render*. O método mais importante é o *Anti-Aliasing (AA) Subsampling*, em que é possível adicionar mais qualidade a cada frame, criando várias sub amostras do mesmo, fazendo a *engine* calcular mais vezes cada frame, melhorando consideravelmente o *render* final [44]. Exemplo da diferença do *subsampling* no Anexo A. Um frame final de cada *shot* das animações do *render* foram adicionados no Apêndice B.

7 Considerações Finais

O fotorrealismo de cenários está chegando aos jogos com os avanços de novas tecnologias como a Nanite e Lumen na Unreal Engine 5 [4], enquanto o de personagens ainda precisa passar pelo *Uncanny Valley*, de modo que a visualização do personagem não cause estranheza ao jogador. Com a disponibilidade de programas gratuitos utilizados na indústria e com o interesse das pessoas, o número de informações e técnicas está crescendo cada vez mais pela internet,com artistas da indústria começando a mostrar cada vez mais seus *workflows*, o que facilitou a pesquisa deste trabalho referente a criação do *workflow* que foi apresentado.

Quanto às técnicas de otimização, foi necessário pesquisá-las com mais profundidade, em comparação às técnicas artísticas de criação, para apresentar quais são as mais impactantes no desempenho do jogo, e é provável que devam existir tantas outras para diferentes propósitos.

O *workflow* desenvolvido se mostrou eficiente e versátil. Funcionando se for modelar cada um dos *assets* e ou se for utilizar uma biblioteca, como foi o caso deste estudo de caso, ele fica mais simplificado, sem ter a necessidade de modelar, texturizar e otimizar cada *asset* manualmente. Foram feitas várias versões do cenário até chegar na versão final, a parte inicial do *workflow* (História, Referência e *Blockout*) é facilmente iterável, podendo produzir diferentes modelos, com diferentes ajustes até chegar numa final em pouco tempo.

O uso da ferramenta do *Movie Render Queue* foi simples e extremamente poderosa, podendo modificar como a *engine* renderiza cada frame, aumentando a qualidade consideravelmente com pequenas linhas do console ajustando seus filtros.

Este *workflow* tornou-se um esqueleto multiuso, que pode ser seguido na criação de inúmeros tipos de cenários, fazendo pequenos ajustes para cada situação, mas como um todo é bastante robusto.

Referências

[1] J. Gurney, Imaginative Realism: How to Paint What Doesn't Exist, 1 ed. Missouri: Andrews McMeel, 2009.

[2] M. Keo, "Graphical Style in Video Games", Dissertação de mestrado, HAMK, Hämeenlinna, Finlândia, 2017. [Online]. Disponível em: https://core.ac.uk/download/pdf/93082889.pdf. Acessado em 20 mar. 2020.

[3] J. Thang. "Tim Sweeney Criticizes Microsoft/Oculus and Talks Project Scorpio" Gamespot. https://www.gamespot.com/articles/tim-sweeney-criticizes-microsoftoculus-and-talks-p/1100-6441734/ (acessado em 25 mar. 2020).

[4] A. Battaglia. "Inside Unreal Engine 5: how Epic delivers its generational leap" Eurogamer. https://www.eurogamer.net/articles/digitalfoundry-2020-unreal-engine-5-playstation-5-tech-demo-analysis (acessado em 17 mai. 2020).

[5] A. Chaumette. “Democratizing 3D environment workflows” Medium. https://medium.com/quixel-ab/democratizing-3d-environment-workflows-1fa0c6f462ac (acessado em 03 abr. 2020).

[6] A. Järvinen, “Gran Stylissimo: The Audiovisual Elements and Styles in Computer and Video Games”, Tampere University Press, vol. 1, jun. 2002. Disponível em: http://www.digra.org/digital-library/publications/gran-stylissimo-the-audiovisual-elements-and-styles-in-computer-and-video-games/. Acessado em: 20 mar. 2020.

[7] P. Christensson.” Teraflops Definition” Tech Terms. https://techterms.com/definition/teraflops (acessado em 25 mar. 2020).

[8] T. S. Perry, "Leaving the uncanny valley behind," em IEEE Spectrum, vol. 51, no. 6, pp. 48-53, jun. 2014, doi: 10.1109/MSPEC.2014.6821621.

[9] K. Desiderio; I. Philips. “How Pixar’s animation has evolved over 24 years, from ‘Toy Story’ to ‘Toy Story 4’” Insider. https://www.insider.com/pixars-animation-evolved-toy-story-2019-6 (acessado em 25 mar. 2020).

[10] A. K. Lal. “Can Games Ever Truly Achive Photorealistic Graphics?” Gamingbolt. https://gamingbolt.com/can-games-ever-truly-achieve-photorealistic-graphics (acessado em 25 mar. 2020).

[11] M Mueller, "What Brain Activity Can Explain Suspension of Disbelief?" em SA Mind vol 25, no.1, pp. 74-74, jan. 2014. doi:10.1038/scientificamericanmind0114-74b.

[12] G. Hodgkinson, “The seduction of realism” em SIGGRAPH ASIA 2009, Yokohama, Japão, pp. 1-4, doi: 10.1145/1666611.1666615.

[13] C. A. House. “Guide to Realistic Game Art: Pros/Cons and Best Practices” Concept Art House. https://www.conceptarthouse.com/news/2015/8/20/game-art-realistic-concept-art-house (acessado em 01 abr. 2020).

[14] Quixel. “Rebirth: The future of Virtual Production” Medium. https://medium.com/quixel-ab/rebirth-the-future-of-virtual-production-2825fff07ff9 (acessado em 01 abr. 2020).

[15] J. Scharr. “The Tech Challenges to Photorealistic Games” Tom’s Guide. https://www.tomsguide.com/us/photorealism-ten-years- why,review-1915.html (acessado em 03 abr. 2020).

[16] J. Hruska. “Investigating ray tracing, the next big thing in gaming graphics” Extreme Tech. https://www.extremetech.com/gaming/135788-investigating-ray-tracing-the-next-big-thing-in-gaming-graphics (acessado em 03 abr. 2020).

[17] W. Vaughan, Digital Modeling, 1 ed. Berkeley: Pearson Education, 2012

[18] Autodesk 3DS MAX. “3ds Max | Software de modelagem, animação e renderização 3D”. https://www.autodesk.com.br/products/3ds-max/ (acessado em 03 jun. 2020).

[19] Blender Foundation. “Home of the Blender Project – Free and Open 3D Creation Software”. https://www.blender.org/ (acessado em 05 jun. 2020).

[20] Autodesk Maya. “Maya |Software de animação computadorizada e modelagem”. https://www.autodesk.com.br/products/maya/ (acessado em 05 jun. 2020).

[21] CLO Virtual Fashion. “Marvelous Designer”. https://www.marvelousdesigner.com/ (acessado em 05 jun. 2020).

[22] Quixel. “Quixel Megascans”. https://quixel.com/megascans (acessado em 05 jun. 2020).

[23] Adobe. “Adobe Photoshop | O Melhor software de edição de fotos, imagens e design”. https://www.adobe.com/br/products/photoshop.html (acessado em 05 jun. 2020).

[24] Quixel. “Quixel Mixer”. https://quixel.com/mixer (acessado em 05 jun. 2020).

[25] Adobe. “Substance Painter | Substance 3D”. https://www.substance3d.com/products/substance-painter/ (acessado em 05 jun. 2020).

[26] R. Garlington. “What are the different texture maps for?” Poliigon. https://help.poliigon.com/en/articles/1712652-what-are-the-different-texture-maps-for (acessado em 18 mai. 2020).

[27] N. Jarvis, “Photorealism versus NonPhotorealism: Art styles in computer games and the default bias.”, Dissertação de mestrado, HUD, Huddersfield, Reino Unido, 2013. [Online]. Disponível em: http://eprints.hud.ac.uk/id/eprint/19756/1/Nathan_Jarvis_-_Final_Thesis_-_Jan_2014.pdf. Acessado em 03 abr. 2020.

[28] Epic Games. “Unreal Engine: The most powerful real-time 3D creation platform”. https://www.unrealengine.com/en-US/ (acessado em 05 jun. 2020).

[29] Unity Technologies. “Plataforma de desenvolvimento em tempo real do Unity | Visualizações 3D, 2D, VR e AR”. https://unity.com/pt (acessado em 05 jun. 2020).

[30] K. Trammell. “Big Idea: ‘Baking’” CG Cookie. https://cgcookie.com/articles/big-idea-baking (acessado em 18 mai. 2020).

[31] Gnomon. LA, USA. 3D Environment Art for Video Games: Artist Panel. (26, mai. 2016). Acessado em: 04 abr. 2020. [Video Online]. Disponível em: https://www.gnomon.edu/community/events/3d-environment-art-for-video-games-artist-panel

[32] Pinterest. “Pinterest”. https://br.pinterest.com/ (acessado em 05 jun. 2020).

[33] Ballistiq Digital. “ArtStation”. https://www.artstation.com/ (acessado em 05 jun. 2020).

[34] Idyllic Pixel. “PureRef”. https://www.pureref.com/ (acessado em 05 jun. 2020).

[35] Adobe. “Paleta de cores, o esquema de cores para artistas | Adobe Color”. https://color.adobe.com/pt/explore (acessado em 05 jun. 2020).

[36] Greg Zaal. “HDRI Haven”. https://hdrihaven.com/ (acessado em 05 jun 2020).

[37] C. Luksh; R. F. Tobler; R. Habel; M. Schwärzler; M. Wimmer, “Fast light-map computation with virtual polygon lights” em I3D’13, Orlando, Florida, pp. 87-94, doi: 10.1145/2448196.2448210.

[38] Savage Interactive Pty. “Procreate - Made for Artists” https://procreate.art/ (acessado em 05 jun. 2020).

[39] Pexels. “As melhores fotos gratuitas - Pexels”. https://www.pexels.com/pt-br/ (acessado em 24 mai. 2021).

[40] L. Asmelash. “Peça de metal misteriosa é achada no deserto nos EUA” CNN Brasil. https://www.cnnbrasil.com.br/internacional/2020/11/24/peca-de-metal-misteriosa-e-achada-no-deserto-nos-eua (acessado em 24 mai. 2021).

[41] Quixel. “Environment - Limestone Quarry” Quixel Megascans. https://quixel.com/megascans/collections?category=environment&category=natural&category=limestone-quarry (acessado em 24 mai. 2021).

[42] Unreal Engine. “Volumetric Clouds | Unreal Engine Documentation”. https://docs.unrealengine.com/en-US/BuildingWorlds/LightingAndShadows/VolumetricClouds/index.html (Acessado em 24 mai. 2021).

[43] Y. Xie. “Getting closer to simulating the real world: Applying ray tracing in games | Gamesindustry.biz” Gamesindustry. https://www.gamesindustry.biz/articles/2021-03-16-getting-closer-to-simulating-the-real-world-applying-ray-tracing-in-games (Acessado em 24 mai. 2021).

[44] Unreal Engine. “Movie Render Queue Setting Reference | Unreal Engine Documentation”. https://docs.unrealengine.com/en-US/AnimatingObjects/Sequencer/Workflow/RenderAndExport/HighQualityMediaExport/Reference/index.html (Acessado em 24 mai. 2021).

Apêndice A – *Moodboard* e Referências

A seguir será apresentado o *moodboard* completo, com fotos retiradas de sites como ArtStation, Pexels e da própria Quixel e montado no PureRef.

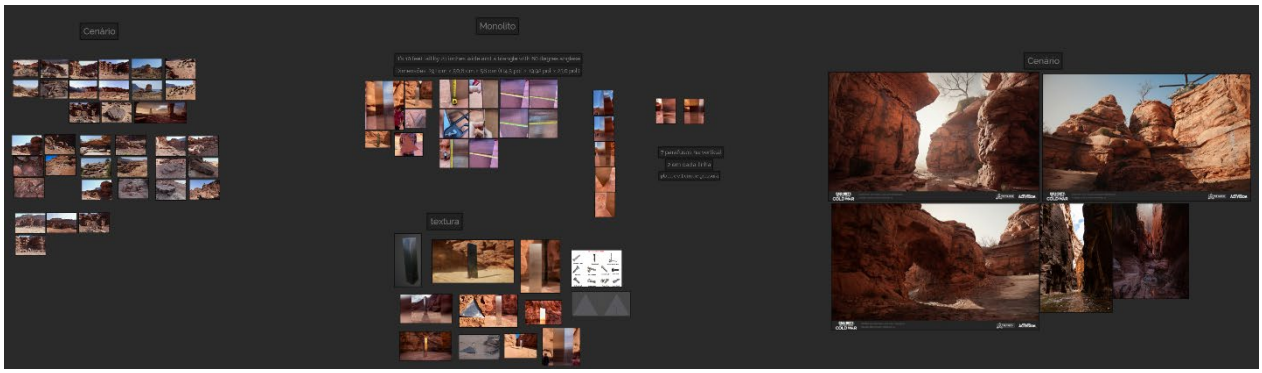


Figura: *Moodboard* completo

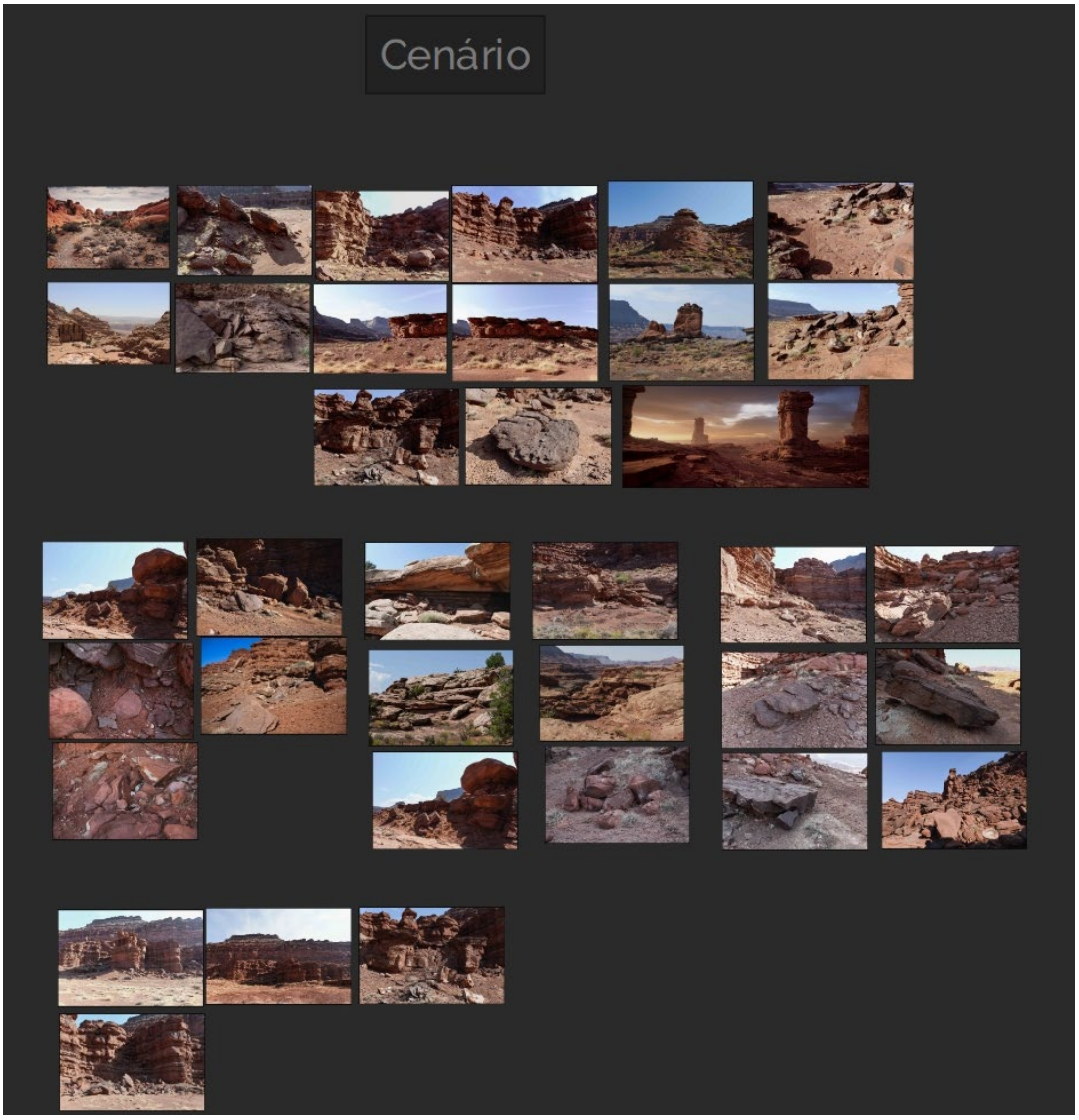


Figura: antigas referências pra a primeira versão do cenário

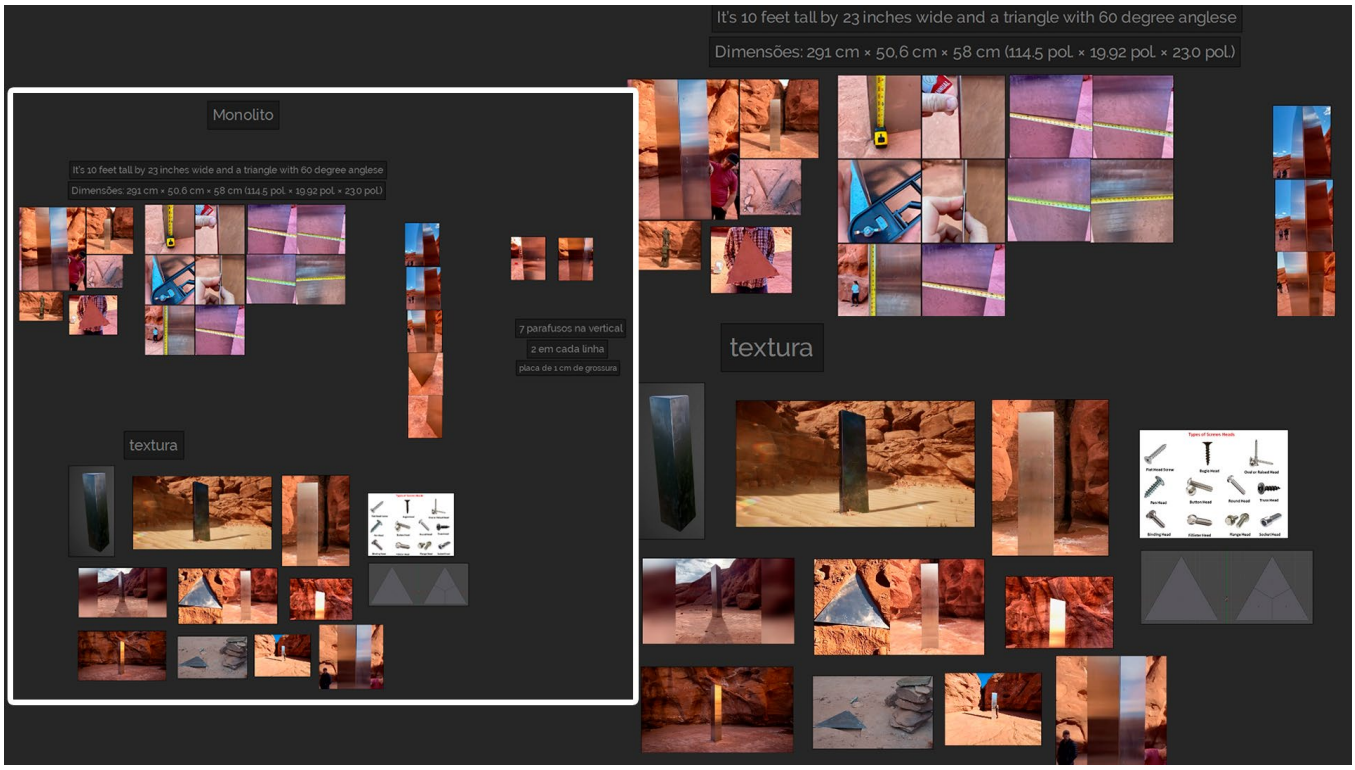


Figura: Referências para o monolito

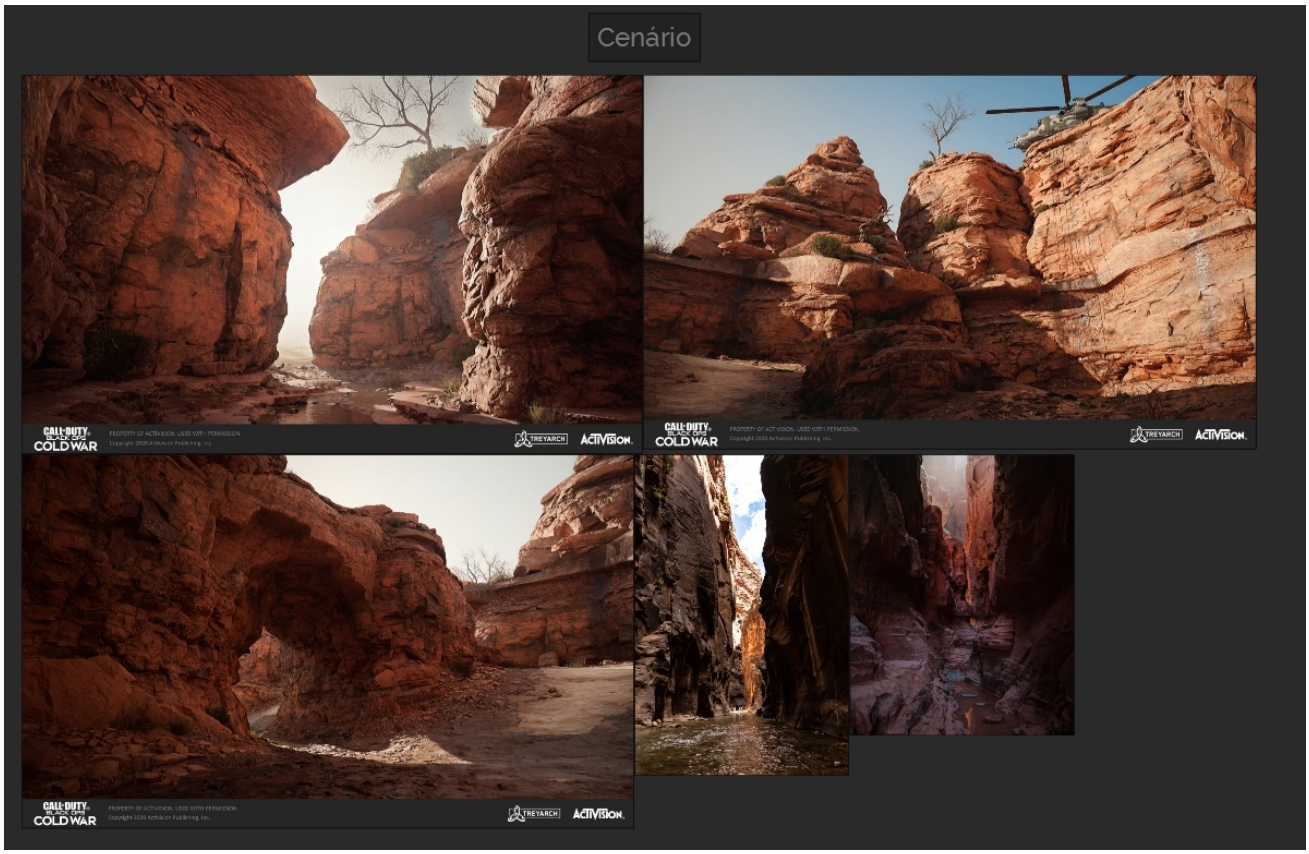


Figura: Referências para o cenário final

Apêndice B – *Renders* Finais

A seguir será apresentado mais imagens finais do cenário feitas com o *Movie Render Queue* da Unreal Engine.

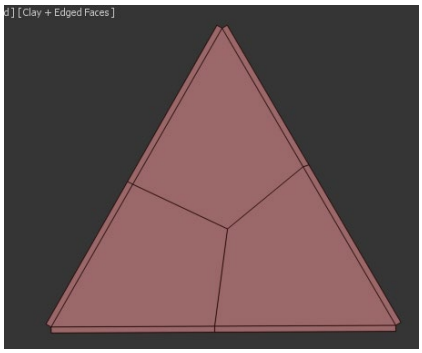
Apêndice C – Modelagem, UV e Texturização do *Hero Asset*

Neste apêndice, será mostrado mais a fundo o passo-a-passo da modelagem, UV e texturização do monólito.

Modelagem

Antes de iniciar, é criado um plano para modelar, analisando qual modo que será o mais efetivo, sem extrapolar muito em polígonos. Mesmo sendo o *hero asset* da cena, onde puder ser economizado, é aconselhável que economize, simplificando a forma com menos polígonos necessários. O monolito real possui 2.91 metros de altura, cada uma das 3 chapas de metal possui 58 centímetros e estão posicionadas em ângulos de 60 graus.

Como regra, sempre garante-se que as faces são *quads*, ou seja, que possuem quatro lados. No caso do monolito ele possui um topo que é um triângulo, ou seja, 3 lados. A solução encontrada foi cortar com a ferramenta *Cut* mais 3 linhas internas. Por fora da forma, ainda mantém-se o triângulo, mas dentro, possui 3 faces de 4 lados.

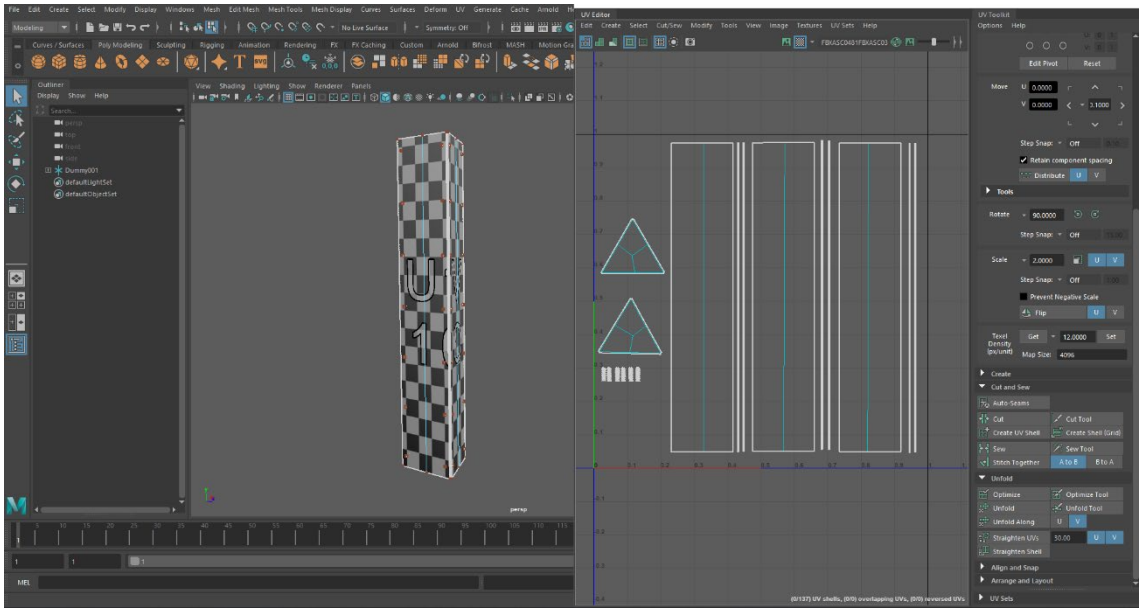


Após ajustar o topo, as chapas de metal ficaram com 5 lados, devido ao vértice que compartilham com o triângulo central. Para resolver, foi feito um *cut* contínuo pelos 3 lados do objeto, até a parte de baixo, onde possui outro triângulo. Cada forma de lados ímpares possuem atalhos e modos de serem resolvidas e virarem *quads*.

Antes de enviar para o UV, cria-se um material básico apenas com cor e definem-se as divisões de materiais. Cada material será um quadrado de UV diferente. No caso deste monolito, tudo será num UV só, então um material para o objeto todo é o suficiente, mas para *props* mais complexas, é normal ela ser separada com várias UV para mais qualidade nos detalhes.

UV

Para fazer os UV do objeto, o mesmo foi enviado para o Maya. Os programas 3ds Max e Maya possuem um *send to* para cada um deles, o que torna a importação mais rápida. Para os UV, é necessário fazer cortes no objeto com o objetivo de deixar todas as faces planas.



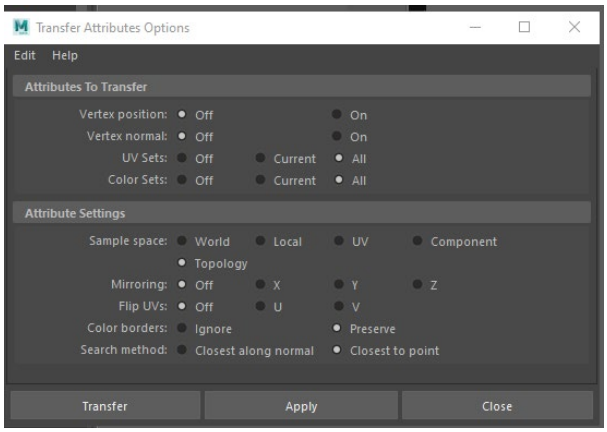
No menu do *UV Editor* do *Maya,* foi selecionado as *Edges* na *viewport*, e depois utilizada a ferramenta *Cut* no *Editor* para cortar o *asset* em áreas planas. Se for feito algum corte errado, selecionar e usar o *Sew* para costurar novamente. Uma regra imporatnte é cortar os cantos onde possuírem dobras de 90 graus, para que não seja perceptível o corte na textura.

Com o objetivo de deixar tudo plano, utiliza-se as ferramentas de *Unfold Along U* e *V*, clica-se várias vezes para abrir as faces no *UV editor*, e depois o *Unfold* para finalizar. Quando o objeto possui lados bem quadrados e retos, utiliza-se o *Straighten UVs*.

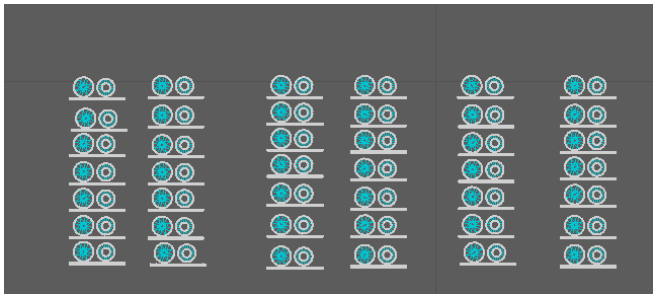
O mais importante depois de abrir tudo, é ajustar cada parte na área do UV e definir um *Texel Density*, para que todas as faces possuam o tamanho real e para que caibam na textura. Abaixo no menu, define-se o tamanho do *Map Size*, texturas 4K são 4096x4096.

Outro ponto importante é deixar a orientação das faces como são no objeto, em ordem de proximidade. Isso facilita no momento de pintar na texturização e orientar o material. Ferramentas de *snap* para *rotate* são úteis para ajustar cada face.

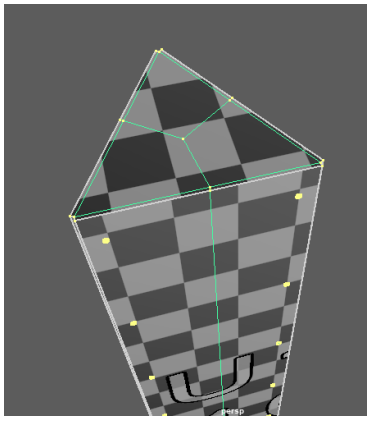
Para objetos que se repetem, a ferramenta *Transfer Atributes* é útil. O atalho dela no Maya é espaço, botão direito e depois em *Mesh* e *Transfer Aributes*. Com ela é possível copiar um UV que já foi feito e colocar em outras partes similares, como foi feito nos parafusos do *asset*. É feito o UV de um dos parafusos e depois clica-se no objeto com o UV feito e seleciona-se em seguida o objeto que queira transferir as UV do primeiro selecionado, e por fim clica-se em *Topology* no *Sample Space* e em *apply*.



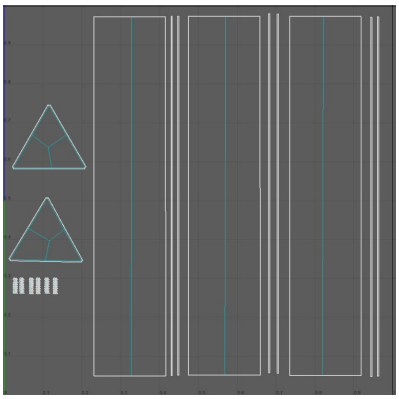
Quando dois objetos forem selecionados, as UV vão estar uma em cima da outra, então seleciona-se um objeto e move-se a UV para o lado para separá-las.



Todas as faces têm que estar visíveis, pois não podem ser sobrepostas para a texturização. No UV *Editor*, a opção de *Checker Map* é utilizada para ver uma textura quadriculada, para pré-visualizar e verificar se a mesma não possui distorções de UV nas faces.



Finalizando, organiza-se para que todas fiquem visíveis e então elas estão prontas para serem enviadas para o *3Ds Max* para serem exportadas em FBX, que é um tipo de extensão de arquivo para objetos em 3D. Com o FBX é possível importar o objeto com suas partes separadas, caso elas tenham sido feitas durante a modelagem.

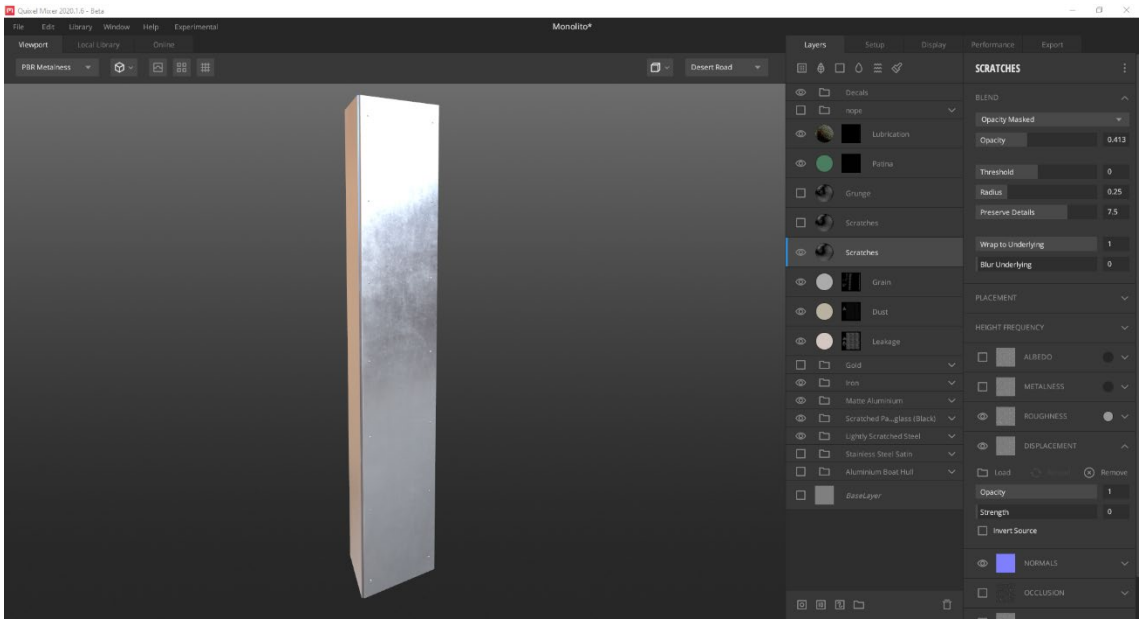


Texturização

No Quixel Mixer, em setup, é possível selecionar um *custom model* para fazer a texturização. Para isso, seleciona-se o arquivo FBX e a resolução da textura, que é 4096 px. Na *viewport* é possível trocar entre o objeto em 3D e o mapa da textura em 2D. A versão 2021, na aba dos *layers*, possui o ícone de *smart materials*, que foi utilizada para texturizar o objeto. É recomendado também utilizar as texturas do Megascans e modificar as *assets* da biblioteca no *Mixer. Após* tudo ser salvo no Quixel Bridge, todos os mapas e *assets* podem ser exportados para a *Unreal* ou qualquer outro programa.

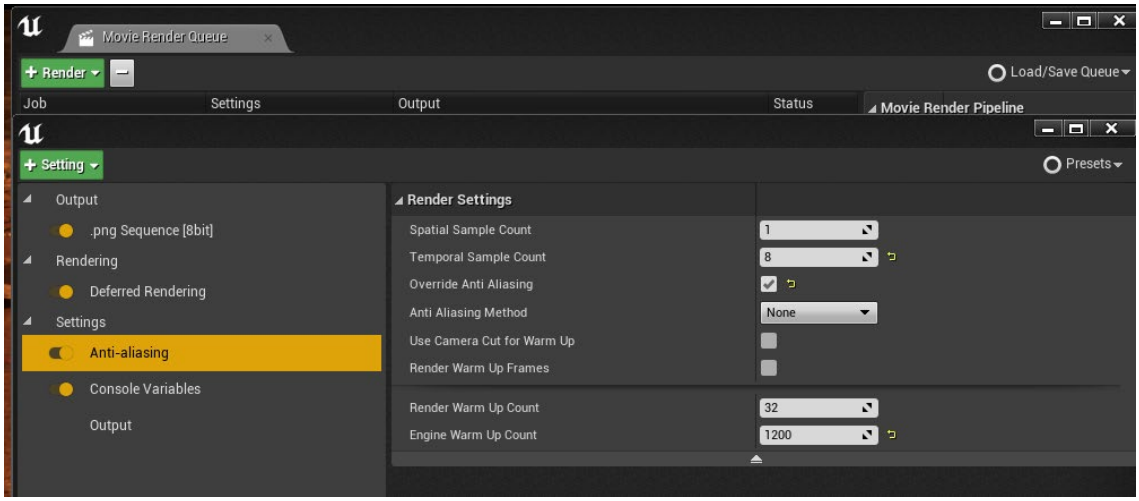
Com os *Smart Materials*, após serem importados, os mesmos viram pastas na aba de *layers.* Neste trabalho foi utilizado o *Lightly Scratchet Steel*. Os principais parâmetros que foram modificados são a textura base, alterando seu *albedo* (cor) e as imperfeições de *Grain*, *Dust* e *Leakage*, como também marcas de dedo, sujeira e marcas nas bordas do *asset*. Outros *smart materials* e imperfeições de diferentes foram usados para criar a textura final. *Decals* são possíveis de adicionar na textura, mas para uma modularidade melhor, adicioná-los na Unreal é aconselhável, pois garante a modificação em tempo real de sua posição. Um dos parâmetros principais foi a opacidade das imperfeições para ajustá-las melhor.

O programa funciona como um Photoshop, em que os *layers* fazem *stacks*, e a ordem importa para a visualização. É possível criar máscaras para que algum detalhe só apareça em lugares definidos pela máscara. Finalizando, é clicado na aba de *export* para exportar os mapas para uma pasta. Existe a opção de enviar para o *Bridge* e adicionar a sua biblioteca de materiais e fazer com a importação para o Unreal seja mais rápida, com um clique em *file* e depois *export to library*.

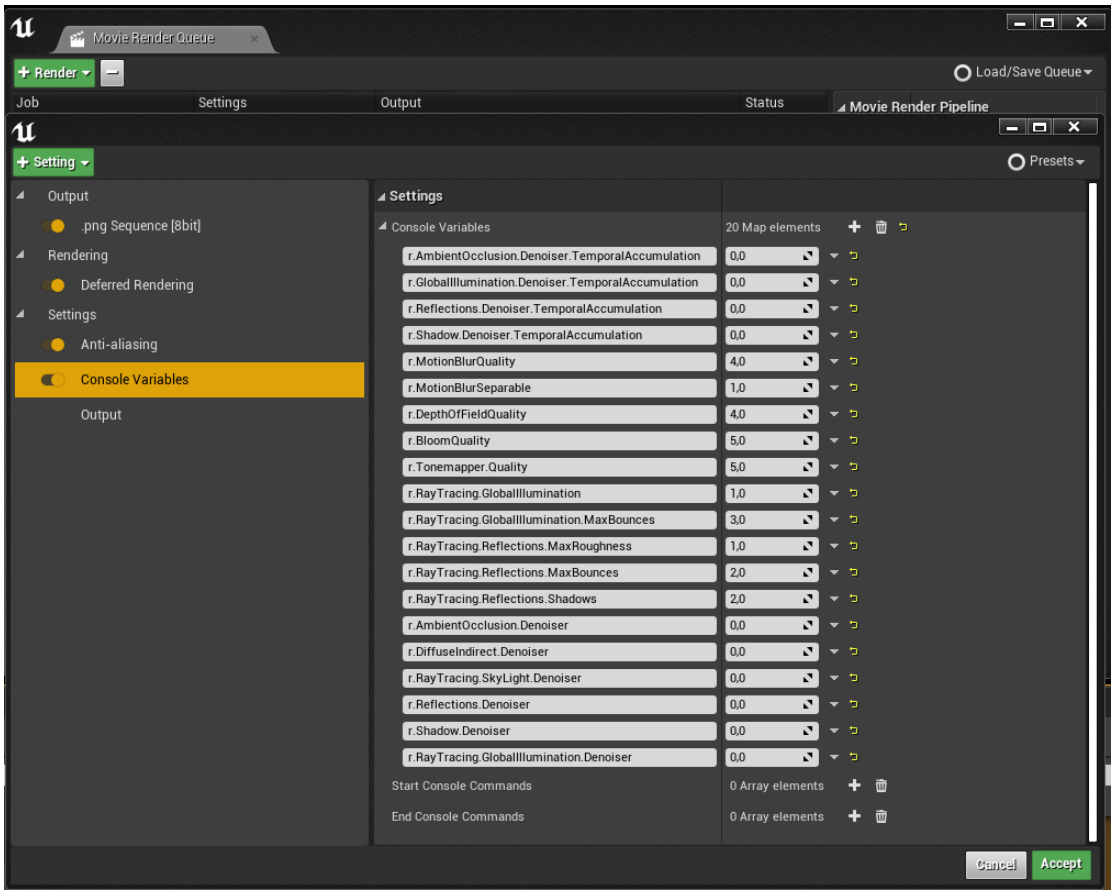


Apêndice D – Configurações do *Movie Render Queue*

Este apêndice mostra as configurações utilizadas para renderizar os frames no Movie Render Queue. Foram renderizados cada frame com uma imagem em png, depois em um programa de edição de vídeos, é possível selecionar todas as imagens e importar toda a sequência e transformá-la em um vídeo. É importante selecionar o *Override Anti Aliasing* e utilizar o *Anti Aliasing Method* em *None* para utilizar o *Subsampling*, que nesta cena foi de 8x (selecionado no *Temporal Sample Count*). O *warm up* no *Anti-aliasing* é alto para as partículas de fumaça percorrerem o cenário completo. No output, foi apenas selecionada a resolução 4K, 3840x2160 e a pasta de exportação.

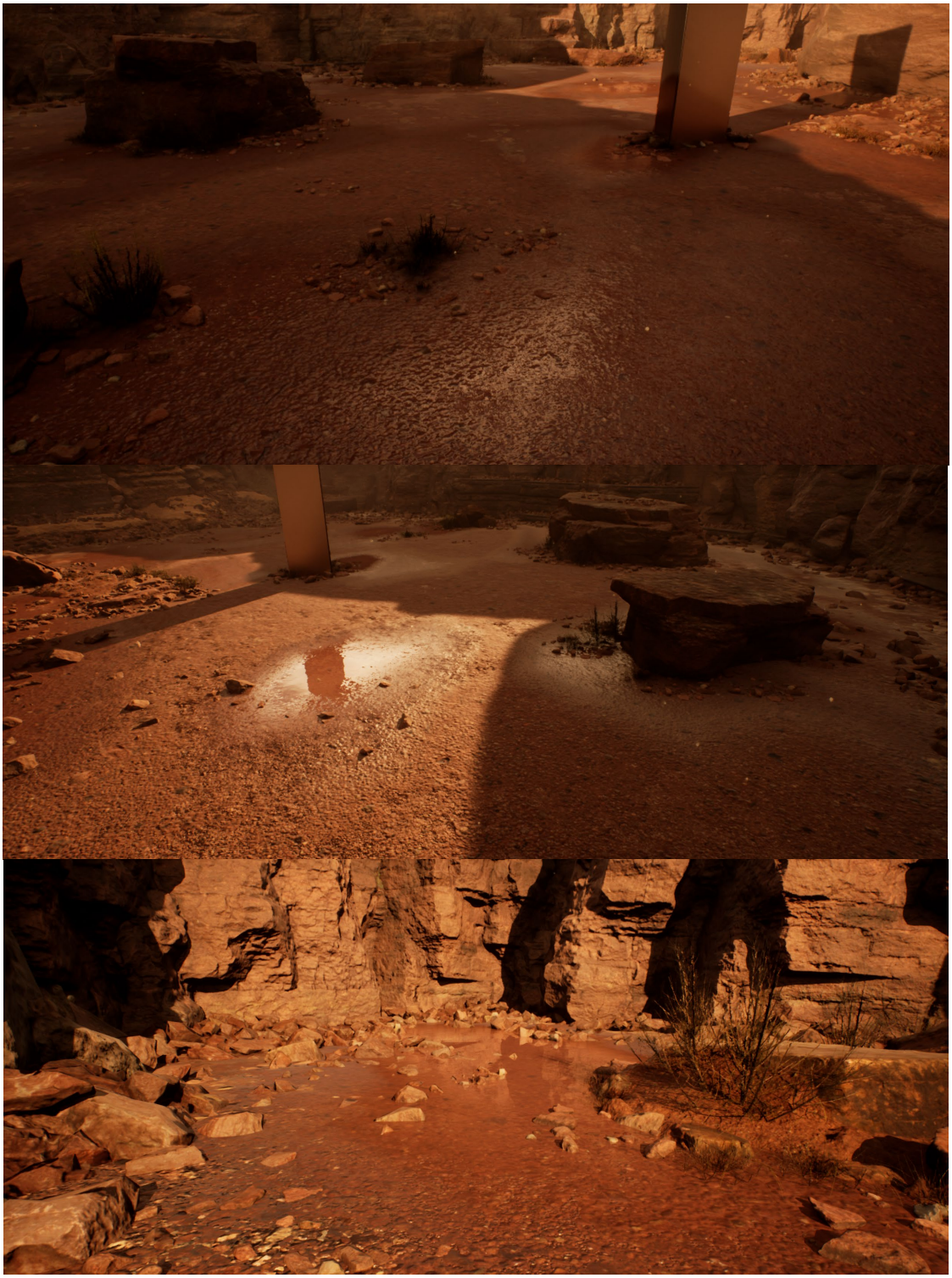


A qualidade extra vem no *Console Variables*, estas foram as linhas utilizadas, também encontradas na documentação do *Movie Render Queue* [44].



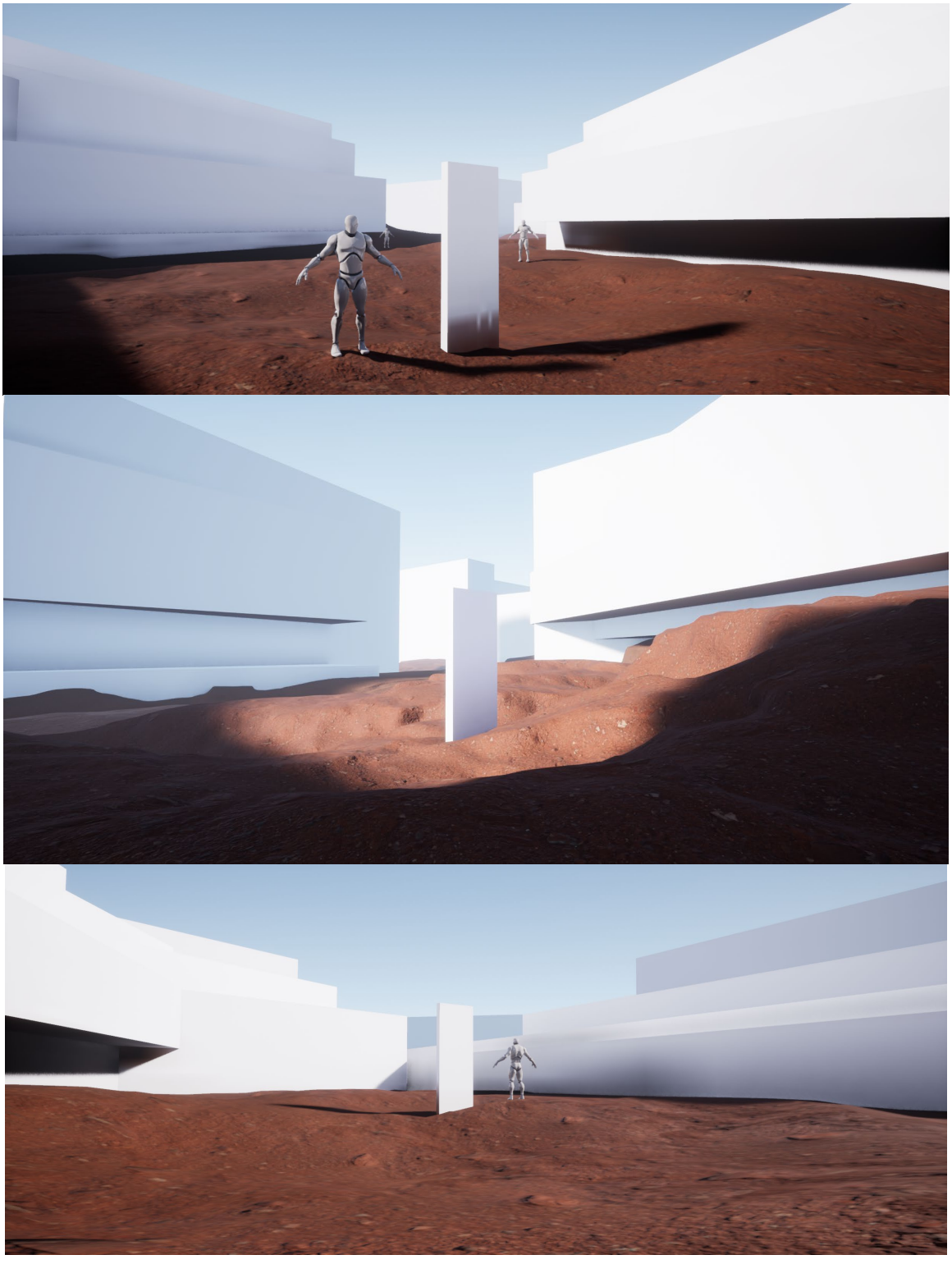
Apêndice E – *Ground Painting*

Adicionando *puddles* e partes úmidas ao cenário, se criam mais áreas com reflexos e brilhos causados pelo sol em conjunto ao *ray tracing* na cena, tornando ela realista com mais detalhes.



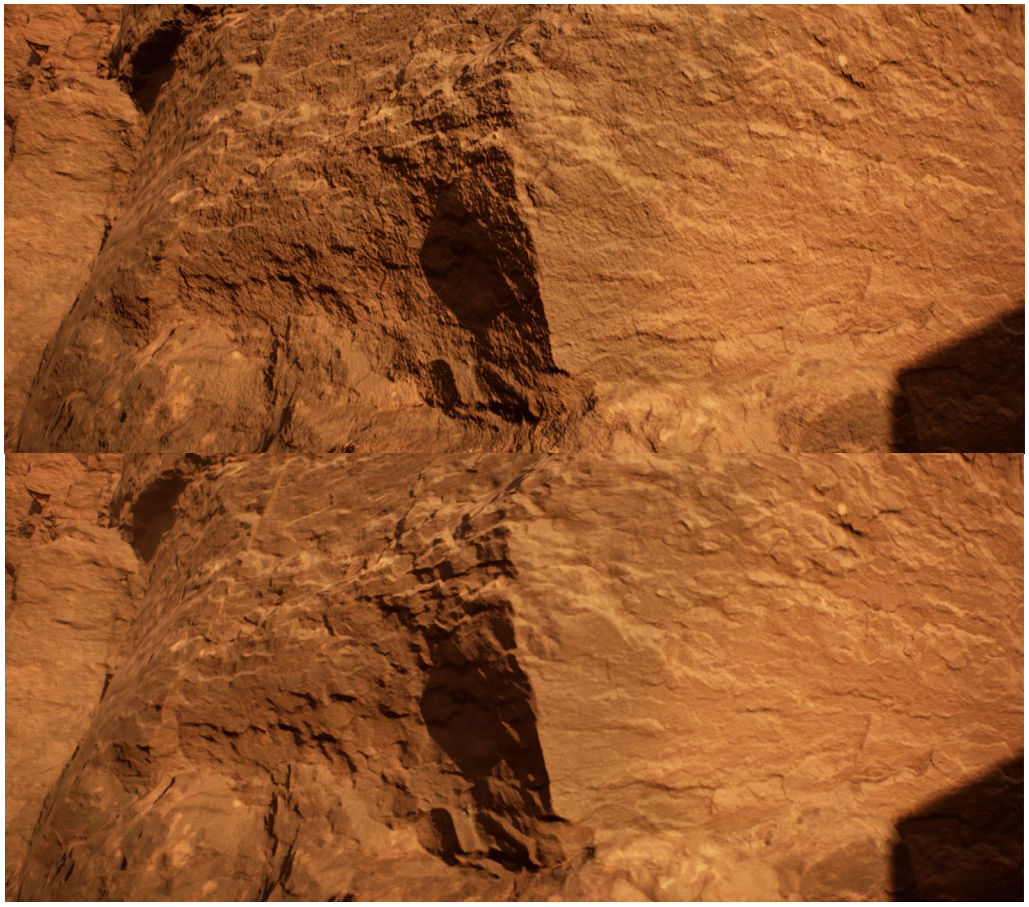
Apêndice F – *Blockout* do Cenário

Esboços em 3D iniciais da primeira versão do cenário, quando a ideia inicial era de construir uma aérea mais aberta. Uso de manequins para compreender as dimensões em relação a um humano é aconselhável nesta parte. Foi utilizado o sistema de *landscapes* da Unreal Engine para modificar o chão durante o *blockout*, sem precisar sair da *engine*, para uma rápida iteração.



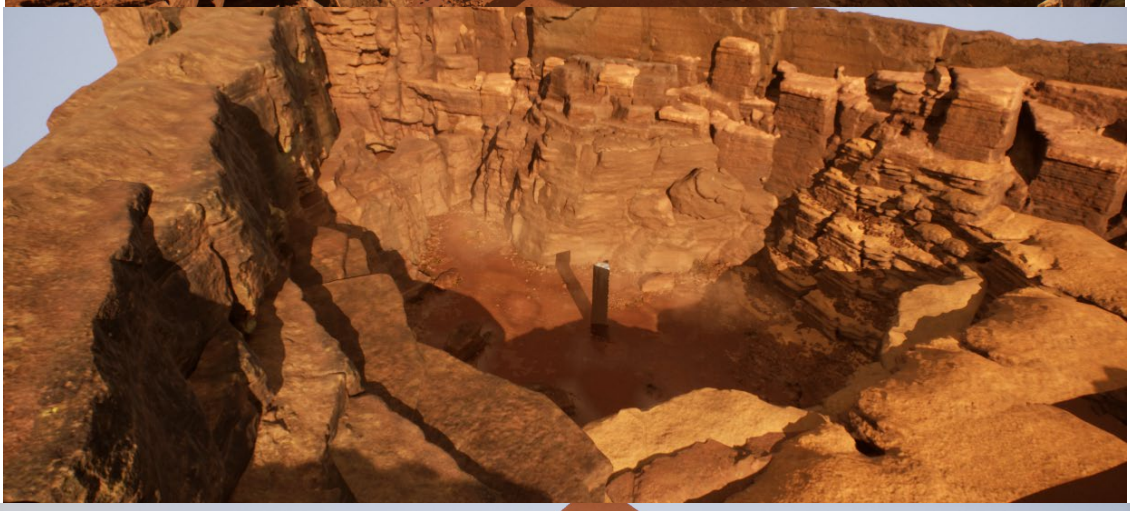
Apêndice G – *Micro Detail Normal*

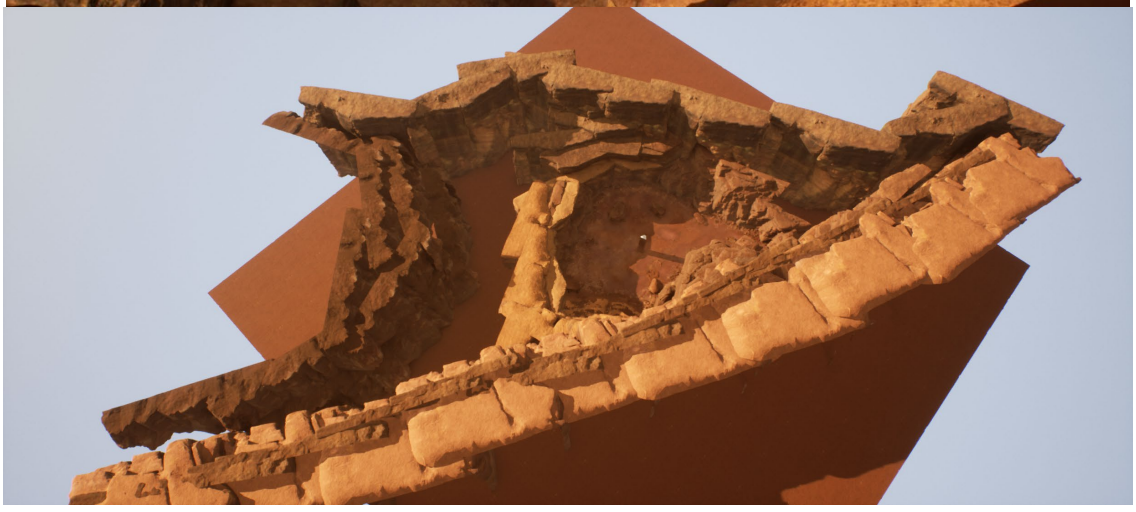
Exemplo do *micro detail normal* ligado e em seguida desligado em uma das pedras maiores no cenário final. É possível ajustar a intensidade e o *tiling* da textura utilizada.

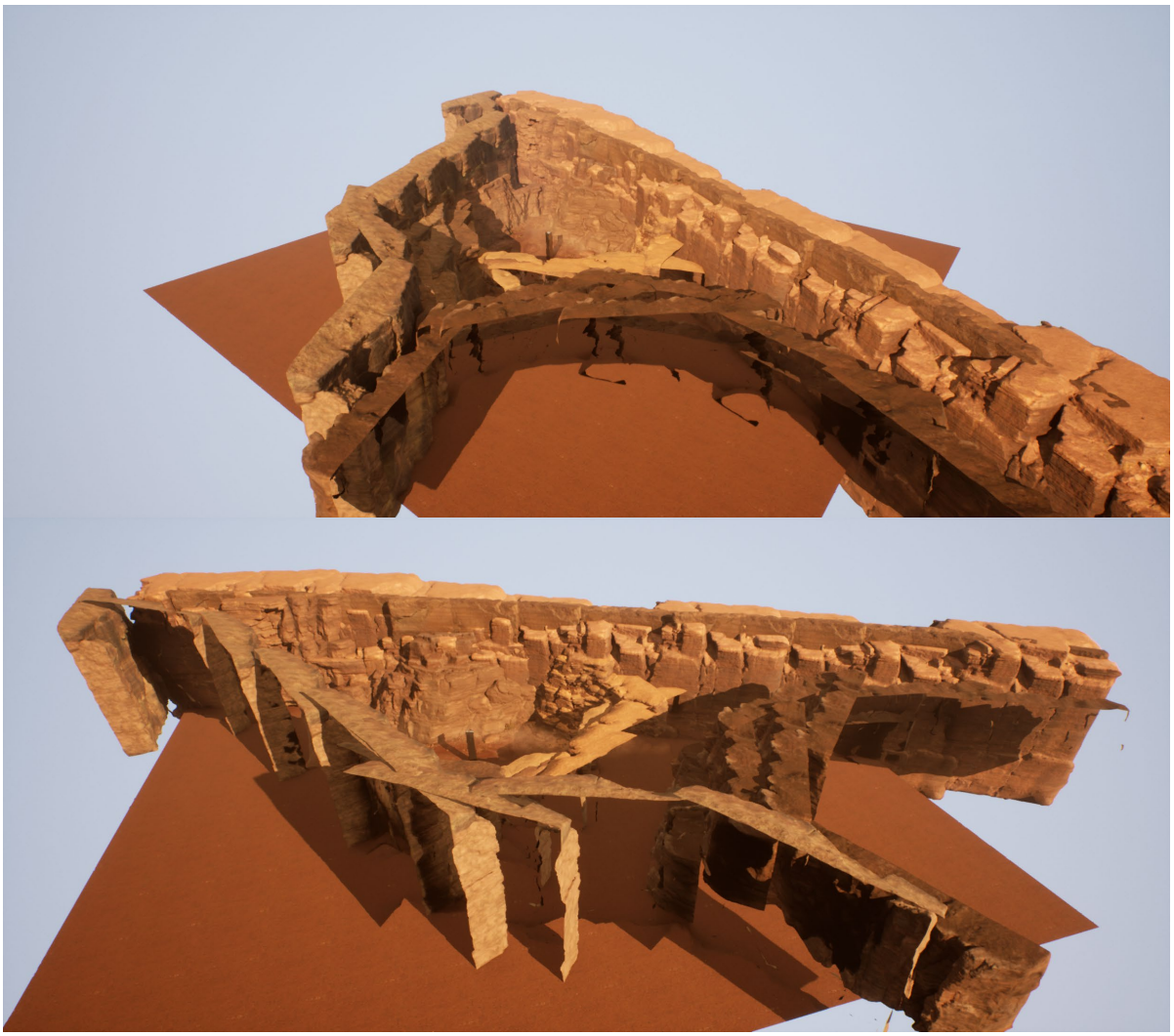


Apêndice H – *Overview* do Cenário

Screenshots de diferentes ângulos mostrando o cenário final com uma visão de fora, demonstrando melhor seu formato.







Anexo A – *Subsampling*

Exemplo do *AA Subsample* em 16x numa cena com *motion blur, retirada da documentação da Unreal Engine* feita pela *Epic Games*.[44]