UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE COMUNICAÇÃO E EXPRESSÃO
DEPARTAMENTO DE EXPRESSÃO GRÁFICA
CURSO DE ANIMAÇÃO

Renan Pereira Botelho Ramos

Render Passes na pós-produção de curta animado em 3D

Florianópolis
2021

Renan Pereira Botelho Ramos

Render Passes na pós-produção de curta animado em 3D

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

Orientador: Prof. Me. Clóvis Geyer Pereira

Florianópolis

2021

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

Ramos, Renan Pereira Botelho
Render Passes na pós-produção de curta animado em 3D / Renan Pereira Botelho Ramos ; orientador, Prof. Me. Clóvis Geyer Pereira, 2021.
37 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro de Comunicação e Expressão, Graduação em Animação, Florianópolis, 2021.

Inclui referências.

1. Animação. 2. Compositing. 3. Render Passes. 4. Pós Produção. I. Pereira, Prof. Me. Clóvis Geyer. II. Universidade Federal de Santa Catarina. Graduação em Animação. III. Título.

Renan Pereira Botelho Ramos

Render Passes na pós-produção de curta animado em 3D

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

Florianópolis, 17 de setembro de 2021.

Prof. Flávio Andaló, Dr. Coordenador do Curso de Animação UFSC

Banca Examinadora:

Prof. Clóvis Geyer Pereira, Me. (Universidade Federal de Santa Catarina)

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

Prof. Gabriel de Souza Prim, Dr. (Universidade Federal de Santa Catarina)

Documento assinado digitalmente
CLOVIS GEYER PEREIRA
Data: 30/09/2021 20:16:20-0300
CPF: 231.580.340-34
Verifique as assinaturas em https://v.ufsc.br

_____

Prof. Me. Clóvis Geyer Pereira
(Orientador)
Universidade Federal de Santa Catarina

Resumo

Este artigo aborda as etapas de produção e pós-produção de um curta-metragem animado em 3D, com ênfase no uso de *render passes* para auxiliar a composição final da imagem. O projeto tem como objetivo explorar as diferentes formas que os *render passes* podem ser utilizados para otimizar e flexibilizar a etapa de *compositing*. Seguindo uma *pipeline* de produção 3D, o autor deu origem a um curta animado completo e mostrou que o uso de *render passes* não só oferece grande controle criativo sobre a aparência final do mesmo, como também ajuda a reduzir significativamente o tempo de renderização do projeto.
Palavras-chave: *compositing*; *render passes*; pós-produção.

Abstract

This article deals with the production and post-production of an animated 3D short film, with an emphasis on the use of *render passes* to assist on the final composition of an image. The project has the intent of exploring the different ways render passes can be used to optimize the compositing stage. Following a 3D production pipeline, the author was able to both finish the short film and show that render passes were invaluable tools not only to achieve creative control over the project's final aesthetics, but also to significantly reduce its render time.
Keywords: *compositing*; *render passes*; post-production.

1. Introdução

Uma das etapas finais encontrada comumente na produção de diversos filmes e séries é o *compositing*. *Compositing* em sua essência consiste em juntar diferentes imagens e elementos para criar uma figura nova (HECKMANN, 2020). O processo pode ser tão simples quanto colocar um mapa meteorológico atrás de um repórter que está fazendo a previsão do tempo ou tão complexo quanto juntar centenas de camadas e ajustar suas propriedades individualmente dentro de um *blockbuster*. É nessa etapa normalmente que efeitos especiais como fogos e explosões são adicionados em cima de alguma filmagem *live action*.

Apesar desta definição clássica de *compositing*, a palavra assume uma conotação diferente quando introduzida a um contexto de animação 3D. Graças ao avanço tecnológico e ao fato desse tipo de imagem ser gerada artificialmente por computador e não capturada através de uma câmera analógica, *compositing* no 3D assume mais uma função de "melhorar"

a qualidade do produto final (GRIGGS, 2020) do que a de simplesmente sobrepor camadas. Para isso, utilizam-se *render passes*; imagens auxiliares criadas pelo *software* 3D que tornam a tarefa de ajustar certas propriedades específicas do *render* possível mesmo após ele ter sido finalizado.

Além de possibilitar uma maior flexibilidade na hora de compor a imagem final, utilizar *render passes* também traz o benefício de ser muito mais rápido do que tentar configurar todos os parâmetros diretamente dentro do *software* 3D (ISMAIL, 2019). Isso causa otimizações significativas no processo de diversos estúdios. De acordo com Bret St. Clair, supervisor de *lookdev*[1] do filme *Homem Aranha no Aranhaverso* (2018), se não fosse pelo uso de *render passes* na pós-produção a equipe não teria sido capaz de alterar efeitos e *frames* do filme na velocidade que fizeram (FOUNDRY, 2019), o que ocasionaria em atrasos significativos em toda a linha de produção.

Isso garante uma importância ainda maior para o *compositing*, uma vez que qualquer alternativa que otimize o tempo de renderização é vital para o sucesso de uma animação 3D; afinal essa é uma das etapas mais demoradas da produção, podendo levar meses dependendo do projeto (PUTRAMA, DARMAWIGUNA e SANTYADIPUTRA, 2017, p.60). Em um filme como *Toy Story 3* (2010), por exemplo, o tempo necessário para se renderizar uma única imagem pode variar entre 7 e 39 horas (LEHRER, 2010).

Essa otimização tem grande relevância também para produções menores, independentes e estudantis. Essas pessoas e estúdios pequenos dificilmente conseguiriam ter acesso aos mesmos recursos que uma empresa como a *PIXAR*, que foi capaz de construir fazendas de renderização gigantescas com centenas de servidores ligados 24 horas por dia (LEHRER, 2010) trabalhando à sua disposição para terminar um filme. Ao invés disso, são comuns situações como a do *Departamento de Engenharia Informática* da universidade indonésia *Undiksha*, onde estudantes acabavam trabalhando múltiplos semestres em um projeto que idealmente deveria ser concluído em apenas um, por não conseguirem terminar o *render* no prazo estipulado (PUTRAMA, DARMAWIGUNA e SANTYADIPUTRA, 2017, p.60).

É dentro desse contexto que o autor deste artigo elaborou, com o intuito de explorar diferentes aplicações de *render passes* na etapa de *compositing*, um projeto animado em 3D como parte do trabalho de conclusão do curso de Animação na Universidade Federal de Santa Catarina. Com este projeto o autor mostra que os *render passes* não apenas diminuíram de forma significativa o tempo de produção da animação, como também garantiram bastante

[1] Etapa da produção 3D onde se testa e estabelece o estilo artístico da obra audiovisual (TRAJE, 2019).

controle criativo sobre a estética final do curta. O processo e os resultados mencionados serão descritos nos tópicos a seguir.

2. Desenvolvimento

2.1 Conceitos iniciais

No projeto descrito neste artigo, a etapa de produção 3D foi realizada inteiramente através do software *open-source*[2] *Blender*. *Blender* é uma ferramenta gratuita capaz de dar conta de toda a *pipeline* de criação 3D (BLENDER), desde a modelagem até a renderização

Renderização é o nome que se dá ao processo de converter a informação presente em uma cena 3D em arquivos de imagem ou de vídeo (PUTRAMA, DARMAWIGUNA e SANTYADIPUTRA, 2017, p.60). Para fazer tal conversão, é necessário utilizar um *motor de renderização*[3].

Cada motor de renderização funciona de forma levemente diferente. O *Cycles*, motor padrão do *Blender* e que foi utilizado para o desenvolvimento do projeto, utiliza uma técnica chamada *path tracing* para gerar suas imagens, permitindo ao software reproduzir as complexidades da iluminação refletida na cena (GLAWION, 2021). *Path tracing* é o processo onde um algoritmo simula as propriedades físicas da luz por meio de raios que são atirados através de uma câmera virtual.

Uma forma simples de visualizar o que isso significa é usando a natureza como referência. A luz, ao ser emitida por uma fonte luminosa primária como o sol (BUGLIA, 2016), viaja em linha reta até ir de encontro a um objeto. Após a colisão, tal objeto muda a trajetória do raio de luz e a absorve parcialmente. O raio segue em sua nova trajetória em linha reta até entrar em colisão com algum outro objeto, e o processo se repete. Segue abaixo (figura 01) uma representação da trajetória de diversos raios de luz em uma floresta:

---

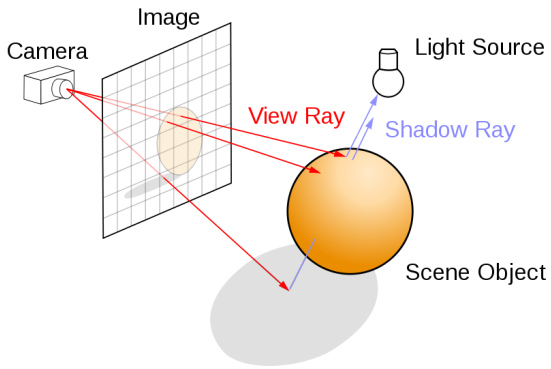
[2] Software com código aberto, onde qualquer pessoa com conhecimento de programação pode acessar e modificar (CHARLEAUX, 2021).
[3] Do inglês, "*render engine*".

Figura 01 - Raios de luz refletindo na natureza

Fonte: *Disney (2016)*

Pode-se dizer que o *path tracing* segue exatamente o mesmo princípio, mas de trás para frente. Ao invés de recriar feixes de luz saindo de uma fonte luminosa, o motor de renderização atira diversos raios através de uma câmera virtual dentro do software. Cada um desses raios segue em linha reta até atingir algum objeto e então muda a sua trajetória até alcançar um ponto de luz (figura 02).

Figura 02 - Representação básica de *path tracing*



Fonte: *Wikimedia*[4]

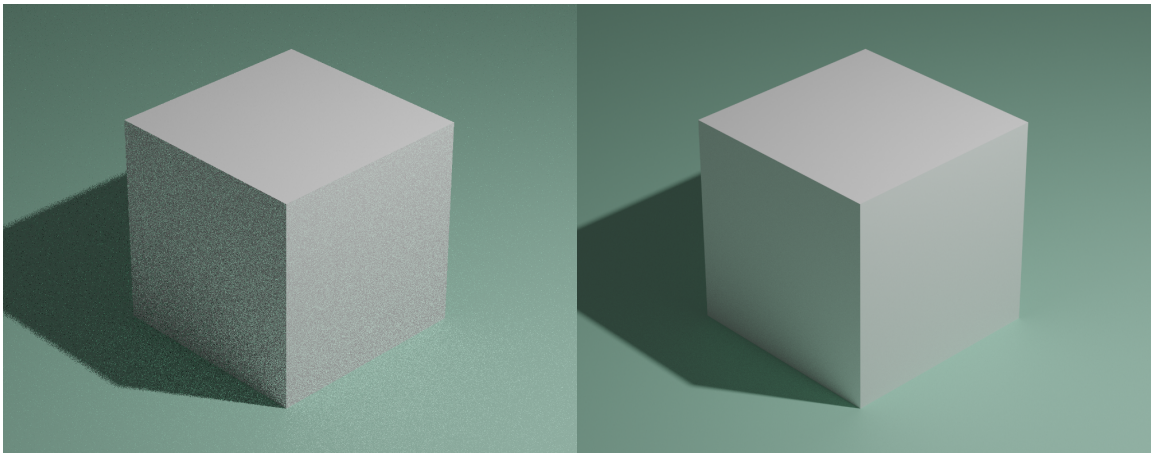
Esses raios atirados pela câmera são responsáveis por formar a imagem final: o *render*. No *Cycles*, a quantidade de raios disparados é controlada através de uma propriedade chamada *sampling*. Quanto maior o número de *samples*, melhor a qualidade final da imagem (OOST, 2021), mas maior o tempo necessário para se completar o *render*.

Na imagem abaixo (figura 03), temos a mesma cena renderizada duas vezes com diferentes níveis de *sampling* em cada. O *render* da esquerda foi feito com apenas um *sample*,

[4] Disponível em: <https://commons.wikimedia.org/wiki/File:Ray_trace_diagram.svg>.
Acesso em: 04/09/2021.

e o processo de renderização levou pouco menos de um segundo. O *render* da direita foi feito com 100 *samples* e demorou seis segundos para ficar pronto. Apesar da velocidade, é possível notar que a figura com menos *samples* apresenta muito mais ruído que a segunda.

Figura 03 - Cubos renderizados com diferentes níveis de *sampling*



Fonte: Autor

Existem diferentes tipos de raios que a câmera virtual do *Blender* pode disparar pela cena e, dependendo de como esses raios se comportam, eles registram certas informações em camadas diferentes (SELIN, 2021). Essas camadas são posteriormente combinadas para formar a imagem final. Contudo, é possível ter acesso a cada uma dessas camadas individualmente através de *render passes*.

Render passes salvam informações extras do *render* em pequenos contêineres. Esses dados não têm o intuito de serem vistos isoladamente (SELIN, 2021), mas sim o de carregar valores técnicos que auxiliam o pós processamento de imagens. Abaixo (figura 04) segue um *frame* do curta do autor com diferentes *passes* sendo mostrados.

Figura 04 - Diversos *render passes* colocados lado a lado

Fonte: Autor

Um ponto importante a ser mencionado é que, apesar de se estar usando o *Blender* como base para este artigo, as técnicas e definições de *render passes* podem ser aplicadas em diversos *softwares*. Cada motor de renderização tem um padrão próprio e funciona de forma diferente, mas os princípios gerais são os mesmos. Alguns motores muito comuns na indústria como o *Arnold* e o *Renderman* (LEE, 2019), por exemplo, sequer utilizam o termo *render passes*. Ao invés disso, esses motores chamam os *passes* de *Arbitrary Output Variables*, ou *AOVs* (RENDERMAN). Apesar da diferença na nomenclatura, *render passes* e *AOVs* carregam basicamente o mesmo significado (SHANASA, 2016).

2.2 Metodologia e *pipeline*

Desenvolver uma obra audiovisual, independente do escopo que ela possa ter, é um trabalho árduo e extenuante. Para otimizar a produção, é muito comum que estúdios de animação 3D dividam em pequenas etapas ordenadas as tarefas necessárias para o produto final ficar pronto (NAGHDI e ADIB, 2021). Isso faz com que ao invés de se ter uma tarefa única chamada "fazer filme", por exemplo, se tenham as pequenas tarefas "construir cenário", seguida por "filmar cena", "editar cena" e assim sucessivamente. A este processo dá-se o nome *pipeline*.

A *pipeline* proposta por Arash Naghdi e Payam Adib para animação 3D foi usada como metodologia para a criação do curta-metragem animado descrito neste artigo. Eles dividem o processo em três estágios principais: *pré-produção*, *produção* e *pós-produção* (2021).

Pré-produção se refere à etapa de pesquisa e planejamento do projeto. Aqui são desenvolvidas as ideias e o roteiro do curta.

A **produção** é onde grande parte do trabalho 3D acontece. É aqui onde são feitas as etapas de modelagem, texturização, *rigging*[5], animação, iluminação e *render*.

A **pós-produção** é o último estágio do projeto, onde artistas utilizam uma variedade de ferramentas para customizar a aparência final do curta. É aqui onde ocorrem as etapas de *compositing* e correção de cor.

[5] Etapa onde se cria um esqueleto invisível que define como as diferentes partes de um personagem 3D se movimentam durante a animação (GRIGGS, 2019).

Algumas alterações foram feitas na pipeline de Naghdi e Adib para melhor corresponder às necessidades do autor. Por exemplo, a etapa de *VFX* - onde efeitos especiais 3D seriam criados durante a produção - foi eliminada uma vez que não havia a necessidade disso para o curta. Além disso, por preferência pessoal, a etapa de texturização foi adiada e feita após a etapa de *rigging*.

2.3 Visão geral do curta

Para a criação do projeto, o autor trabalhou em conjunto com outra estudante da Universidade Federal de Santa Catarina e juntos deram origem ao curta-metragem animado *Pintura Digital*. Durante a etapa de pré-produção, a equipe de dois membros se auto impôs o limite de dois meses e meio - entre junho e agosto de 2021 - para a realização do projeto.

Devido às restrições de tempo e de pessoal, a equipe precisou bolar um conceito simples que fosse possível de ser executado pelos dois integrantes. Algumas estratégias de otimização foram adotadas para garantir que o curta ficasse pronto dentro do prazo, como por exemplo a animação 3D ter sido feita em um *frame rate* de 12 *frames* por segundo ao invés dos 24 fps normalmente encontrados na maioria dos filmes modernos (MORRISON, 2020). Além de facilitar o processo de animação, isso também cortou o tempo de renderização pela metade. Outra estratégia adotada foi a de fazer a história do curta se focar em um único personagem: um robô pintor com braços flutuantes destacados do corpo (figura 05). O design simples do personagem facilitou bastante os processos de modelagem e *rigging*.

Figura 05 - Protagonista do curta-metragem *Pintura Digital*

Fonte: Autor

Além do 3D, o curta também contou com uma etapa separada de produção 2D, onde o rosto do personagem e alguns efeitos especiais foram animados *frame* a *frame* pela outra integrante da equipe. Apesar de 2D não ser o foco deste artigo, algumas dessas animações foram usadas em conjunto com *render passes* durante o *compositing* e serão mencionadas quando necessário nos próximos tópicos.

2.4 Renderização aplicada ao projeto

O curta *Pintura Digital* contou com um total de 1256 *frames* divididos em doze cenas para serem renderizados na etapa final da produção 3D. Nas configurações do *Cycles*, foram utilizados 2100 *samples* e uma resolução de 2176 x 1224 para cada imagem gerada. Em termos de hardware, o autor teve à disposição um único computador pessoal para a renderização do curta, com um processador *Ryzen 7 3700X* e uma placa de vídeo *GeForce RTX 3060 Ti*.

O tempo de *render* para cada *frame* variou de acordo com a complexidade da cena. No geral, cenas nas quais a parede ocupou a maior parte do enquadramento (figura 06) foram consideradas mais simples e levaram em média de 4 a 6 minutos para serem renderizadas. Por sua vez, cenas onde vários objetos podiam ser vistos de perto ao mesmo tempo pela câmera foram consideradas complexas e levaram de 8 a 11 minutos para serem renderizadas.

Figura 06 - Exemplo de uma cena mais simples (à esquerda) e uma mais complexa (à direita) em termos de tempo de renderização

Fonte: Autor

Além da imagem final, foram renderizados também diferentes *render passes* nesta etapa da produção. O *Cycles* possui uma variedade de *passes* disponíveis[6], divididos em

[6] A lista completa está disponível em: <https://docs.blender.org/manual/en/latest/render/layers/passes.html>. Acesso em: 20/09/2021.

quatro grandes grupos principais (BLENDER). Por questões de escopo, apenas os *render passes* que o autor considerou relevantes para o desenvolvimento específico do projeto serão abordados neste artigo. Estes *passes* são: *denoising data*, *cryptomatte*, *Z*, *normal*, *UV*, *diffuse direct*, *diffuse indirect*, *diffuse color*, *glossy direct*, *glossy indirect*, *glossy color* e *emission*.

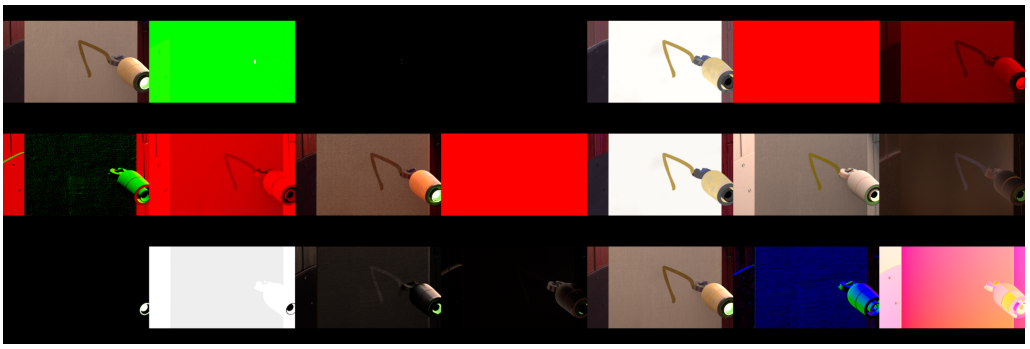
Apesar dos *render passes* citados acima serem calculados paralelamente durante a renderização, alguns deles geram informações extras que podem acabar adicionando ao processo um custo computacional a mais. Para verificar o impacto que isso pode ter na produção, o autor decidiu renderizar um mesmo *frame* de duas formas diferentes. Na primeira vez, apenas a imagem base foi renderizada, sem nenhum *render pass* adicional. O processo levou 9 minutos e 15 segundos.

Figura 07 - *Frame* renderizado em nove minutos e quinze segundos

Fonte: Autor

A cena então foi renderizada novamente, mas dessa vez todos os *render passes* que o autor utilizou durante o projeto foram renderizados juntos. Isso aumentou o tempo de *render* em 45 segundos; o processo todo levou exatos dez minutos para ficar pronto - 8,1% a mais que da primeira vez.

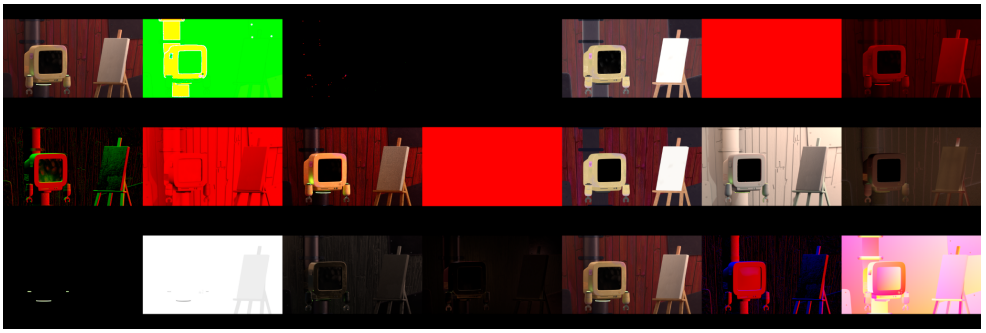
Figura 08 - Diferentes *render passes*. Todas essas imagens foram renderizadas paralelamente em dez minutos



Fonte: Autor

Este tempo adicional, todavia, diminui dependendo da complexidade da cena. O teste foi replicado ao renderizar um *frame* mais simples e a diferença foi de apenas 27 segundos - o *render* base levou 4 minutos e 39 segundos para ficar pronto, enquanto a versão com os diferentes *passes* ativados levou 5 minutos e 6 segundos. Isso representou um acréscimo de 9,7% no tempo total de renderização.

Figura 09 - Diferentes *render passes* de uma cena com menor complexidade



Fonte: Autor

Apesar do tempo adicional que eles geram, todas as cenas do curta foram renderizadas com a presença de *render passes*, uma vez que eles seriam parte fundamental e indispensável da pós-produção do projeto.

Um outro ponto importante a ser mencionado é a questão de espaço de armazenamento de arquivos. Como os *render passes* contém informações técnicas, salvá-los como uma simples imagem PNG pode resultar na perda de dados em alguns casos específicos[7]. Por conta disso, todos os render passes usados no projeto foram salvos em um

[7] Como nos *passes* de *cryptomatte* e *Z*, onde cada pixel na tela pode conter valores maiores do que 1.

formato de imagem chamado EXR, com *32-bits*[8] de cor. Isso resultou em arquivos relativamente grandes, onde cada um dos frames individuais ocupou em média 400 MB de espaço no computador.

2.5 Redução de ruído - *Denoising Data*

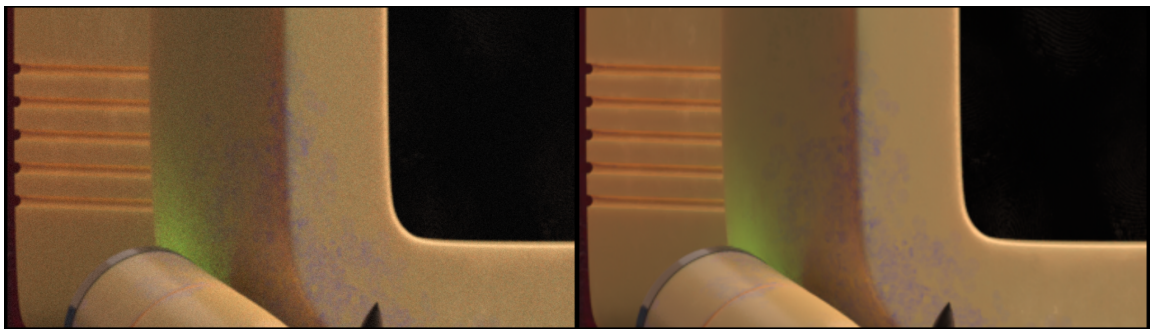
Após a etapa de renderização ter sido finalizada, iniciou-se a etapa de *denoising*. *Denoising* é o processo de reduzir a quantidade de ruído presente em um *render* preservando o máximo de detalhe possível (BLENDER). No *Blender*, essa redução de ruído é feita com a ajuda de *render passes* adicionais - os *passes* de *denoising data* (figura 10).

Figura 10 - Diferentes *passes* de *denoising data*

Fonte: Autor

Com os dados presentes em tais *passes*, o software 3D é capaz de identificar áreas com ruído e, através de um algoritmo próprio, automaticamente uniformizá-las. Abaixo (figura 11) segue um comparativo de um *frame* do curta antes e depois da redução de ruído ter sido aplicada a ele.

Figura 11 - *Frame* antes e depois do *denoising* (à esquerda e à direita respectivamente)



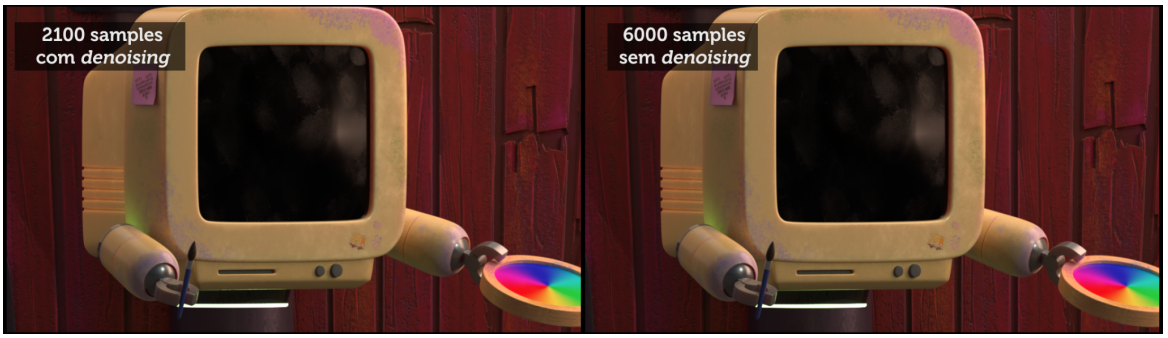
Fonte: Autor

[8] *32- bits* refere-se a um conceito chamado *profundidade de cor*. Quanto mais *bits por canal* uma imagem tiver, maior a quantidade de cor que ela será capaz de armazenar (BENZ, 2018).

Como a quantidade de ruído gerada pelo *Cycles* depende da quantidade de *samples* utilizados durante a renderização, o *denoising* salvou tempo significativo na produção do projeto. Para testar o impacto que essa técnica teve sobre o produto final, o autor renderizou um mesmo *frame* duas vezes (figura 12). Na primeira vez, a imagem foi renderizada com 2100 *samples* - mesma quantidade utilizada para a renderização final do projeto - e depois utilizou-se os *render passes* para aplicar o *denoising*. O processo inteiro de *render* levou 4 minutos e 50 segundos.

Da segunda vez, o autor tentou apenas aumentar a quantidade de *samples* na cena antes de renderizar-lá novamente, sem aplicar nenhuma outra técnica de pós-processamento para diminuição de ruído. Foram necessários 6000 samples para se obter um resultado comparável com o anterior, levando um total de 27 minutos e 13 segundos de *render*.

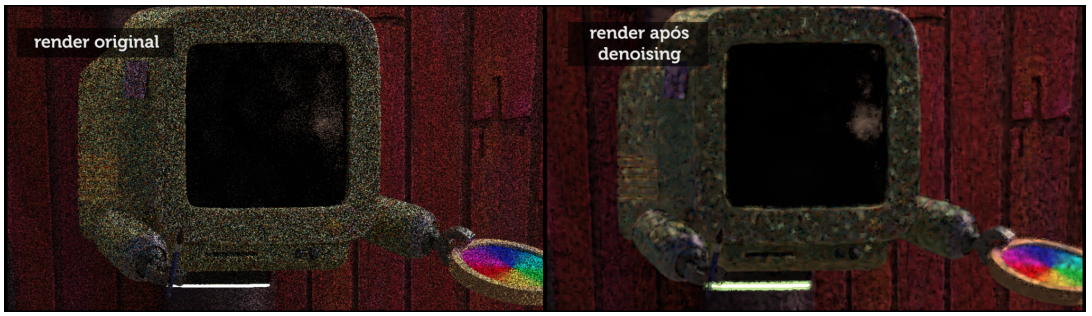
Figura 12 - Comparativo de diferentes níveis de *sampling*



Fonte: Autor

Apesar do *denoising* ser capaz de otimizar significativamente o tempo de *render*, alguns artefatos como áreas escuras e embaçadas podem acabar surgindo durante o processo. Tais artefatos se tornam bem perceptíveis principalmente quando a técnica é aplicada em imagens renderizadas com uma quantidade pequena de *samples* (figura 13). Por causa disso, é recomendável buscar um meio termo entre tempo de *render* e quantidade de ruído presente no mesmo (ISMAIL, 2019).

Figura 13 - *Denoising* de um *frame* renderizado com apenas um *sample*



Fonte: Autor

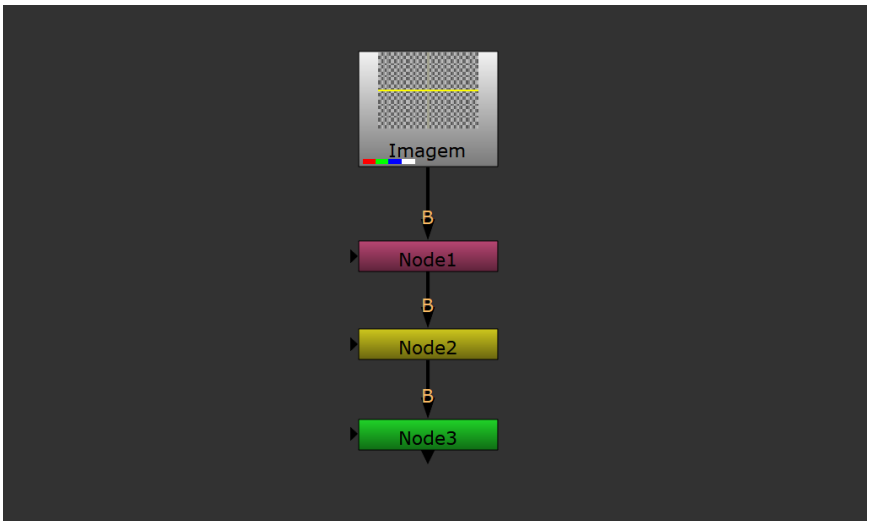
Uma característica importante dos *passes* de *denoising data* é que eles não necessariamente precisam ser aplicados apenas no *render* final. Na verdade, eles podem ser usados para reduzir o ruído de quaisquer outros *render passes* necessários (SELIN, 2021). Assim, o autor também aplicou o *denoising* nos seguintes *passes*: *diffuse direct*, *diffuse indirect*, *glossy direct* e *glossy indirect*. Uma contextualização da funcionalidade dos *passes* citados está presente na seção 2.6.2 deste artigo.

2.6 *Compositing* usando *render passes*

Após o *denoising*, iniciou-se o *compositing*. O software utilizado para esta etapa da pós-produção foi o *Nuke*. *Nuke* é uma aplicação muito utilizada no pós-processamento de filmes e séries de TV, e que proporciona bastante flexibilidade ao artista por ter uma interface baseada em *nodes* (KAUSHIK, 2020).

Nodes, em essência, são pequenos blocos capazes de realizar operações matemáticas que modificam a aparência do *node* anterior (LEAVITT, 2018). Ao conectar uma imagem ou vídeo a um determinado *node*, por exemplo, ele modifica os pixels daquela imagem de alguma maneira específica. Existem diferentes *nodes* no *Nuke* capazes de realizar diferentes efeitos. Alguns *nodes* aplicam um desfoque geral na imagem, enquanto outros são destinados a tarefas de correção de cor, aumentando características como saturação e contraste do vídeo.

Figura 14 - Estrutura de *nodes* dentro do *Nuke*



Fonte: Autor

Por se organizarem em pequenos blocos, é muito fácil reutilizar cadeias inteiras de *nodes* dentro do *Nuke*. Isso permitiu ao autor fazer o *compositing* de apenas uma cena e então replicar os resultados em todas as restantes apenas substituindo a imagem conectada no topo da cadeia.

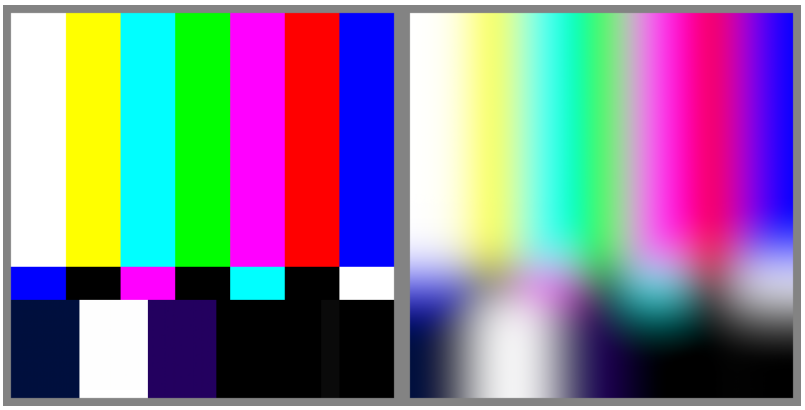
A seguir serão descritos como diferentes *render passes* foram utilizados dentro do *Nuke* para auxiliar e otimizar a pós-produção do curta animado *Pintura Digital*.

2.6.1 *Cryptomatte*

Em *compositing*, *máscara* é uma técnica utilizada para isolar certos elementos em cena. Basicamente, é o processo de se criar um mapa preto e branco que influencia na transparência de determinada imagem ou efeito. No contexto de um software como o *Nuke*, normalmente branco significa opaco e preto significa transparente (YENG, 2015).

Tais mapas podem então ser usados para isolar algum efeito em uma área especificada de um *render*. Abaixo (figura 15), duas imagens foram colocadas lado a lado. À esquerda, a imagem original pode ser vista, e à direita a mesma imagem com efeito de desfoque aplicada a ela.

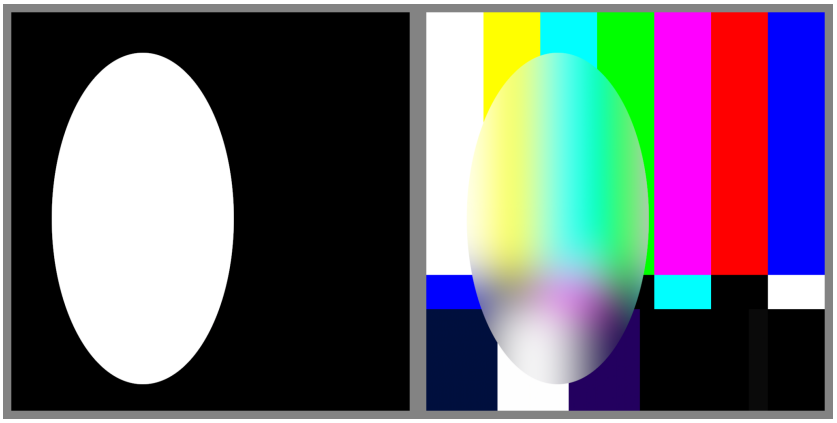
Figura 15 - Imagem sem e com desfoque (à esquerda e à direita respectivamente)



Fonte: Autor

Utilizando uma máscara em preto e branco (figura 16), é possível restringir o efeito de desfoque a apenas uma determinada região da imagem.

Figura 16 - Máscara em preto e branco sendo utilizada para controlar a região de efeito de desfoque



Fonte: Autor

Normalmente, máscaras precisam ser geradas manualmente dentro do *Nuke*. Isso pode acabar resultando em muito tempo gasto dedicado apenas a essa tarefa específica, principalmente se tal máscara precisar ser animada *frame* a *frame*. É um trabalho tão laborioso que em filmes e produções grandes é comum a presença de profissionais contratados exclusivamente para a animação de máscaras: os *roto artists* (MAHER, 2015).

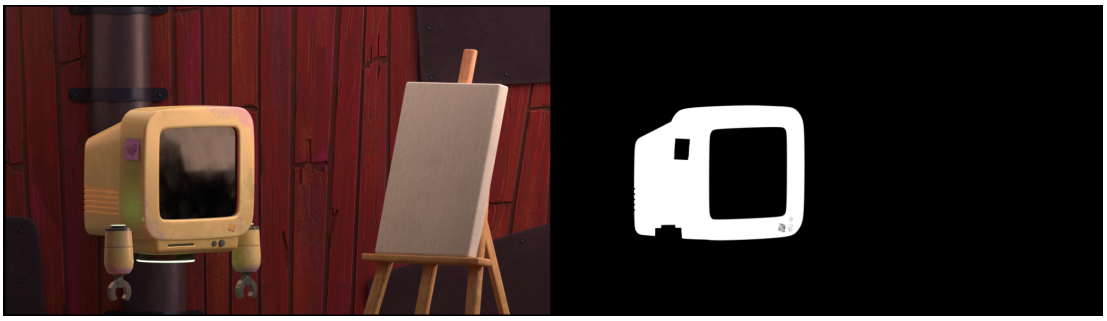
O *cryptomatte* é um *render pass* com a finalidade de automatizar esse processo. Ao renderizar este *passe*, o *Cycles* gera uma imagem com diferentes valores que podem conter desde números negativos até números tão altos que ultrapassam a escala de branco. Na prática, a imagem recebe uma aparência como essa (figura 17):

Figura 17 - *Cryptomatte*

Fonte: Autor

Apesar de não parecer muito útil quando visto dessa maneira, as informações presentes no *passe* de *cryptomatte* podem ser acessadas por um *node* no *Nuke* para gerar máscaras precisas capazes de isolar quaisquer objetos em cena - mesmo animados - durante o *compositing*. Isso permitiu ao autor simplesmente selecionar uma área desejada e o *software* automaticamente criar um mapa preto e branco com a silhueta do objeto, como na figura abaixo (figura 18).

Figura 18 - Máscara (à direita) gerada automaticamente com o uso de *cryptomatte*



Fonte: Autor

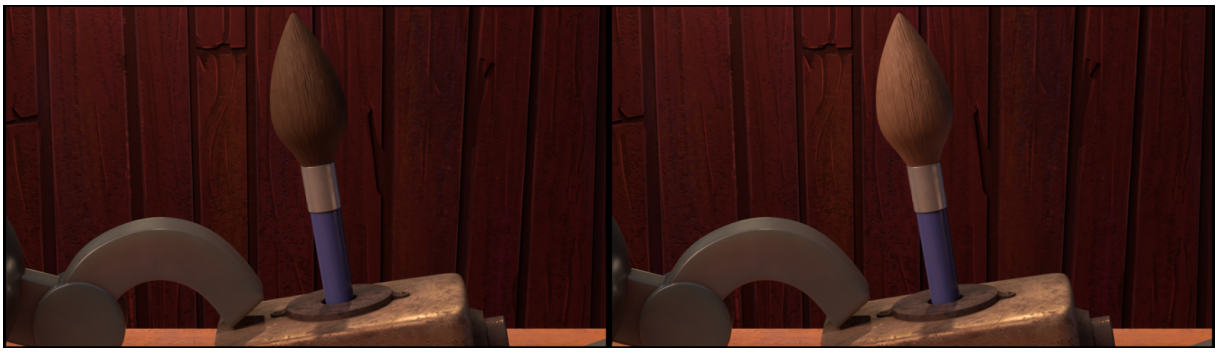
Durante a pós-produção do curta *Pintura Digital*, essa técnica foi usada para realizar de forma rápida ajustes de cor pontuais em algumas cenas. Durante uma sequência onde um adesivo com os dizeres "*FLIP CANVAS*" aparecia colado atrás de um espelho, percebeu-se que a cor do adesivo estava um pouco escura demais, dificultando a leitura do papel. Ao invés de retornar ao software 3D, mexer na iluminação e renderizar a cena inteira novamente, utilizou-se o *passe cryptomatte* para aumentar o brilho somente do adesivo.

Figura 19 - *Cryptomatte* utilizado para aumentar o brilho do adesivo em uma cena do curta *Pintura Digital*

Fonte: Autor

De forma similar, o *cryptomatte* foi utilizado em outro momento do curta para criar uma máscara da ponta de um pincel e isoladamente aumentar o seu brilho (figura 20). Isso foi importante para melhorar o contraste do objeto com o fundo e facilitar a leitura da cena.

Figura 20 - *Cryptomatte* utilizado para aumentar o brilho da ponta do pincel no curta *Pintura Digital*



Fonte: Autor

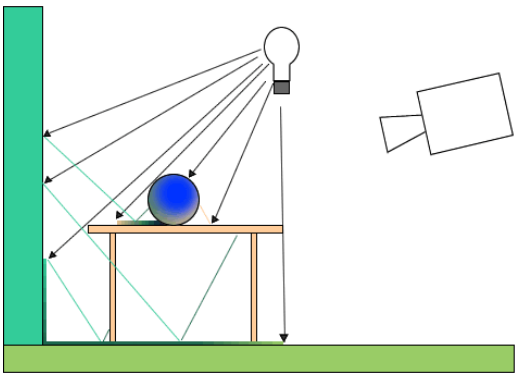
Além da correção de cor, o *cryptomatte* foi utilizado de diferentes outras formas durante a pós-produção do projeto, inclusive em conjunto com outros *render passes*. Quando presentes, os outros usos do *cryptomatte* serão descritos nos próximos tópicos.

2.6.2 *Light passes*

Light passes, ou *passes de luz*, são *render passes* automaticamente calculados durante a renderização da cena e utilizados para compor a imagem final (SELIN, 2021). Existem diferentes tipos de *passes* dependendo de como a luz se comporta em determinado contexto. Quando um raio de luz é emitido e colide pela primeira vez com algum objeto, este raio é

chamado de *luz direta* (BLENDER). Quando o raio reflete em algum obstáculo antes de atingir o objeto, ele recebe o nome de *luz indireta*.

Figura 21 - *Luz direta* sendo representada pelas setas pretas e *luz indireta* pelas linhas ricocheteadas

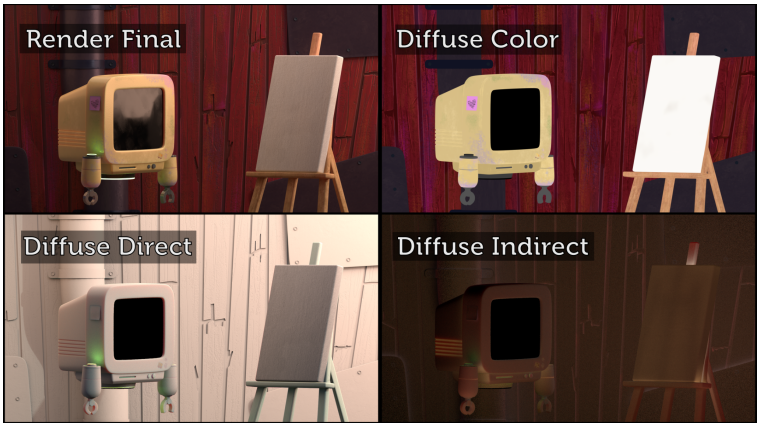


Fonte: *Rastermon*[9]

A nomenclatura que a luz recebe depende também do tipo de raio que ela é. *Luz difusa* é o nome que se dá a um raio de luz que contribui para a cor de um objeto, enquanto *luz especular* se refere a um raio de luz que contribui para os reflexos do mesmo (SELIN, 2021). Um espelho teria uma intensidade maior de *luz especular* do que um tijolo, por exemplo.

Cada um desses raios é separado em um *passe* próprio. Para os raios de luz difusa, tem-se os *passes Diffuse Direct* (referente à luz difusa direta), *Diffuse Indirect* (referente à luz difusa indireta) e *Diffuse Color* (representando a cor base dos objetos em cena sem nenhuma informação de luz atrelada). Os diferentes *passes* podem ser vistos na figura abaixo (figura 22).

Figura 22 - Os diferentes *passes* da luz difusa



Fonte: Autor

[9] Disponível em: <https://web.archive.org/web/20050204031559/https://rastermon.com/GI1.htm>. Acesso em: 01/09/2021.

De forma paralela, para os raios de luz especular tem-se os *passes Glossy Direct*, *Glossy Indirect* e *Glossy Color* (figura 23).

Figura 23 - Os diferentes *passes* da luz especular

Fonte: Autor

Existem também outros tipos de raio que podem contribuir para a iluminação da cena, como os *raios de transmissão* - capazes de atravessar objetos transparentes ou semitransparentes como vidro (SELIN, 2021) - e os *raios de volume* que contribuem para objetos volumétricos como fumaça ou fogo (BLENDER). Como não tiveram relevância para o projeto, estes dois *passes* serão desconsiderados no restante deste artigo.

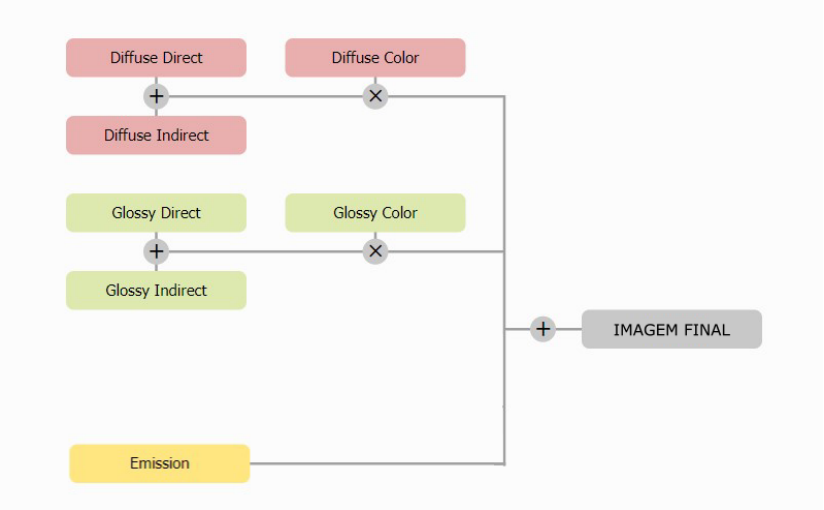
Dentro da categoria de *light passes*, existe também o *render pass* chamado de *emission*. Este é um *passe* que isola geometrias que emitem luz, como lâmpadas ou objetos brilhantes. No curta, de todos os modelos 3D, apenas aos aros nos braços e na parte debaixo do personagem foram atribuídos um material emissivo.

Figura 24 - O *passe* de *emission*

Fonte: Autor

Como cada um desses *light passes* representa uma faceta diferente do *render*, eles podem ser combinados através de certas operações matemáticas (figura 25) para formar a imagem final.

Figura 25 - *Light passes* sendo combinados para formar a imagem final



Fonte: *Blender*, com adaptações do autor

Apesar de parecer redundante realizar todo este processo só para gerar a mesma imagem que o software já cria por padrão, ter essas informações divididas em diferentes *passes* permitiu ao autor fazer modificações específicas no render durante o *compositing*.

Para fins de estilização, por exemplo, aumentou-se o contraste unicamente do *passe diffuse direct* em todas as cenas do curta. Isso ocasionou uma diferença sutil na imagem (figura 26), tornando as sombras dos objetos mais perceptíveis, mas sem afetar diretamente os reflexos.

Figura 26 - *Frame* base (à esquerda) e *frame* onde o *passe diffuse direct* teve o contraste aumentado (à direita)

Fonte: Autor

A intensidade do *passe glossy direct* também foi alterada durante o *compositing*. A influência que tal *passe* tinha sobre as cenas foi reduzida com o objetivo de tornar mais sutis os reflexos no cenário. O *render pass cryptomatte* foi utilizado aqui para controlar exatamente sobre quais áreas essa redução aconteceria. Assim, a tela do robô - área mais problemática, onde os reflexos chamavam mais atenção - foi mais afetada do que o restante do cenário.

Figura 27 - *Frame* base (à esquerda) e *frame* onde a intensidade do *passe glossy direct* foi diminuída (à direita)

Fonte: Autor

Se tais *render passes* não tivessem sido renderizados, para reduzir o brilho da tela teria sido preciso retornar ao software 3D, mudar a iluminação e as propriedades do material e renderizar a cena inteira novamente. O tempo gasto com isso seria inviável para um projeto deste escopo.

O *passe* de *emission*, por sua vez, foi modificado durante o compositing para tingir os aros do robô de verde e aumentar a intensidade da luz emitida por eles. Isso resultou em um brilho mais perceptível e realista no modelo do personagem (figura 28).

Figura 28 - *Passe* de *emission* sendo usado para aumentar o brilho dos aros luminosos do robô

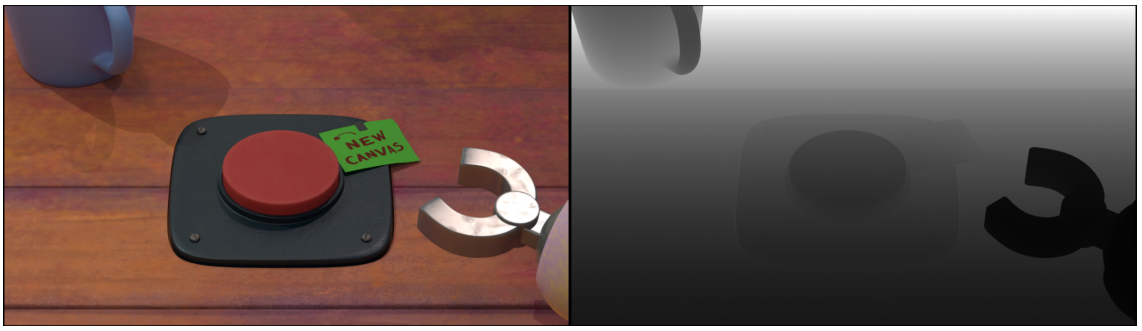
Fonte: Autor

2.6.3 *Z*

O *render pass* chamado unicamente de *Z* pode ser entendido como um *passe* de *profundidade*. Quando renderizado, este *passe* simplesmente mostra a distância, em metros, dos *pixels* gravados à câmera (SELIN, 2021). Como esse valor normalmente é maior do que um, é comum que o *render pass*, a princípio, aparente ser simplesmente uma imagem completamente branca.

Para melhor visualizar o que o *passe* representa, é necessário *normalizá-lo*; ou seja, pegar os valores presentes na imagem e mapeá-los novamente para dentro de uma escala de 0 a 1 (SELIN, 2021). Assim, obtém-se uma nova imagem em preto e branco que condiz visualmente com a profundidade da cena.

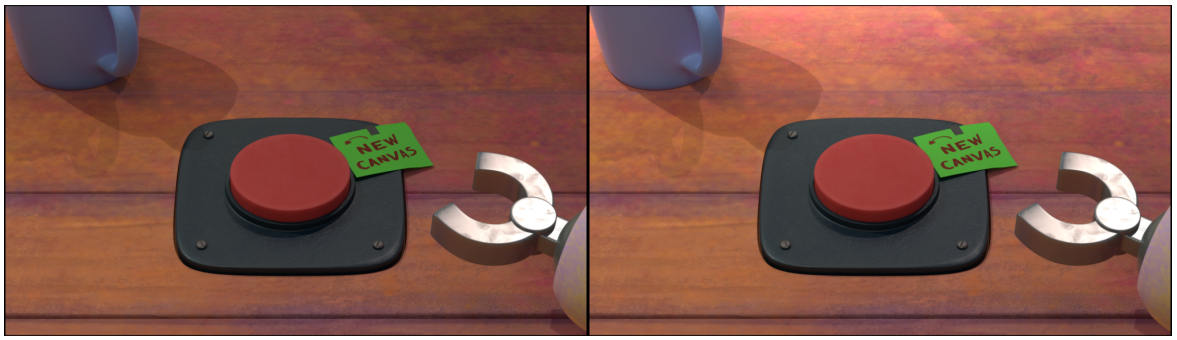
Figura 29 - Mapa de profundidade da cena (à direita) criado ao normalizar o *passe Z*



Fonte: Autor

Como o resultado deste processo é uma imagem em preto e branco, ela pode ser usada como uma máscara para efeitos de desfoque e correção de cor. Isso permite, por exemplo, iluminar ou escurecer a cena baseada na distância dos objetos à câmera, o que cria a ilusão de uma luz ter sido adicionada tridimensionalmente ao cenário.

Figura 30 - Correção de cor feita com auxílio do *passe Z*



Fonte: Autor

Contudo, não é completamente necessário normalizar o *passe Z* para utilizá-lo no *Nuke*. O software é capaz de ler as informações de profundidade presentes nele e utilizá-las para simular um efeito de *depth of field*[10]. No curta *Pintura Digital*, tal efeito foi aplicado em momentos onde queria-se dar um destaque maior a alguma área específica da imagem.

Em uma cena onde a atenção do espectador deveria estar voltada a uma gaveta (figura 31), utilizou-se o efeito de *depth of field* para criar um desfoque no chão e, consequentemente, guiar o olhar da audiência para o local desejado.

Figura 31 - Imagem sem e com efeito de *depth of field* aplicado (à esquerda e à direita, respectivamente)

Fonte: Autor

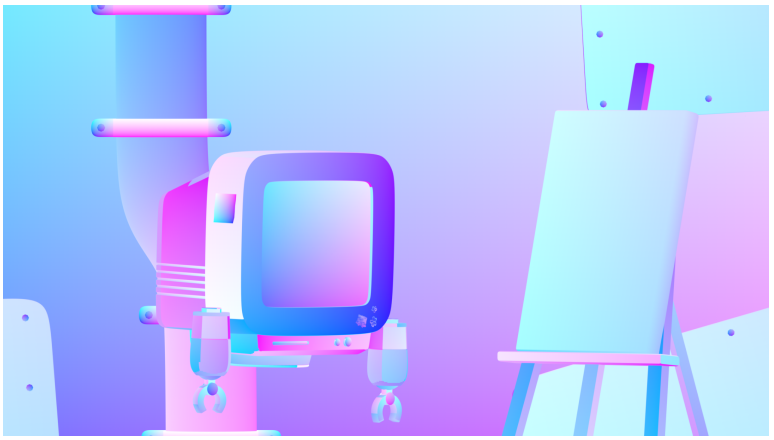
Um ponto importante a ser levantado sobre o efeito de *depth of field* é que é possível replicá-lo diretamente dentro do software 3D, sem a necessidade de nenhum *render pass* adicional. Contudo, quando feito dessa forma no *Cycles*, o desfoque se torna parte permanente da imagem e não pode ser removido durante a pós-produção. Isso diminui a flexibilidade do artista e pode causar atrasos significativos na *pipeline*; caso o resultado do desfoque não seja o esperado, seria necessário renderizar a cena inteira novamente para se obter uma nova versão.

2.6.4 *UV*

O *passe* de *UV* gera uma imagem que contém as coordenadas de onde no objeto 3D uma determinada textura foi mapeada (BLENDER). Tais coordenadas são registradas nos canais de cor da imagem gerada (figura 32) e podem ser acessadas pelo *Nuke* com a finalidade de texturizar novamente qualquer objeto em cena.

[10] Técnica fotográfica que consiste em manter alguns objetos em foco enquanto o restante da cena sofre um desfoque dependendo de sua distância à câmera (MAIO, 2019).

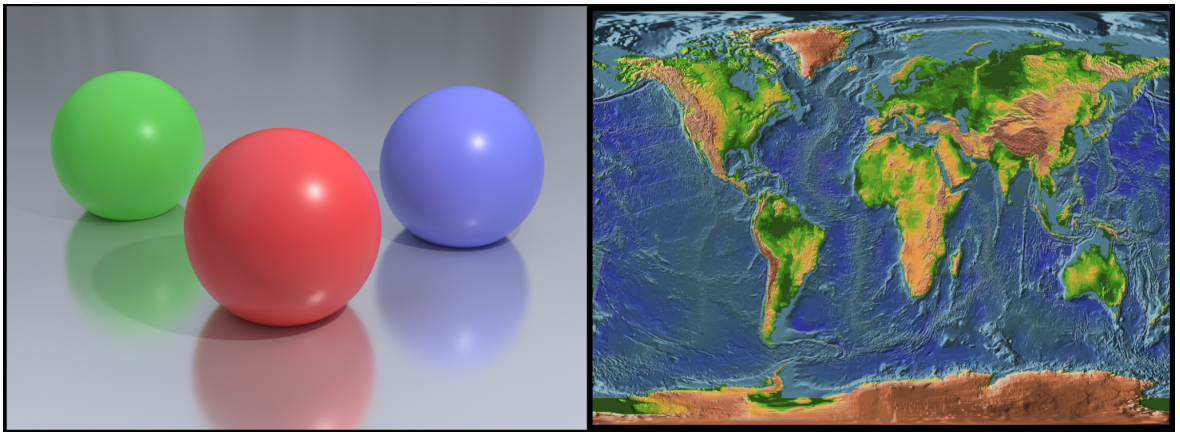
Figura 32 - *Passe* de *UV*



Fonte: Autor

A forma que essa texturização acontece é simples. Primeiro, pega-se um *render* e uma textura, como na figura abaixo (figura 33).

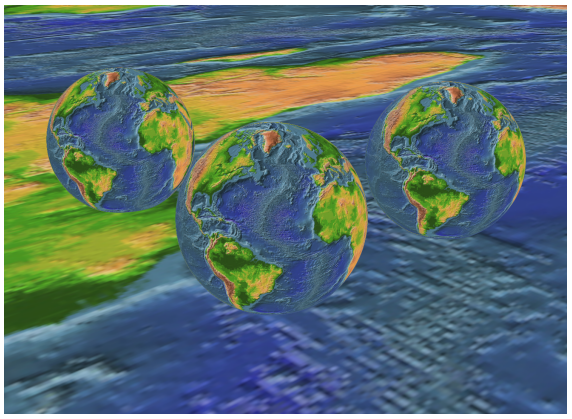
Figura 33 - *Render* 3D (à esquerda) e a textura que será aplicada (à direita)



Fonte: *Nasa* (2009), com modificações do autor

Então, o *Nuke* usa as informações presentes no *passe* de *UV* para projetar essa nova textura em todos os objetos em cena (figura 34).

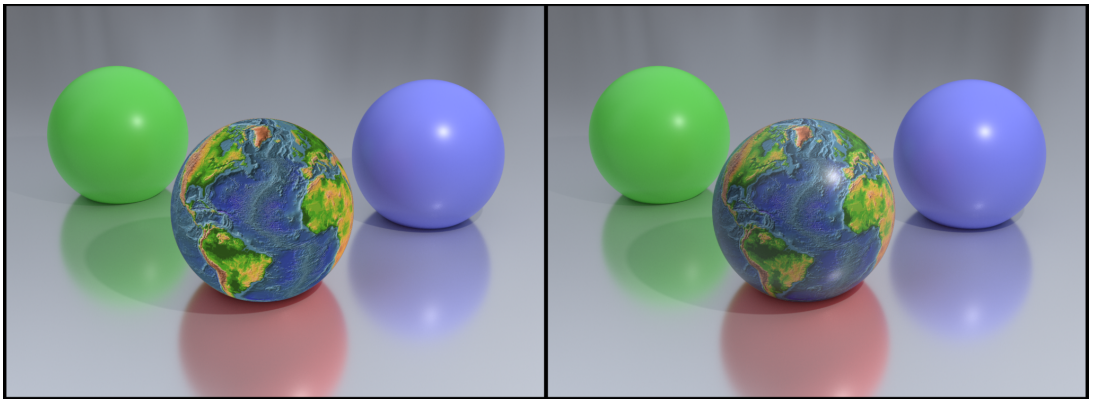
Figura 34 - Textura aplicada em todos os objetos em cena utilizando o *passe* de *UV*



Fonte: Autor

Caso se queira que essa nova textura seja aplicada em apenas um objeto, é possível utilizar o *render pass cryptomatte* para criar uma máscara que isole a área desejada. Caso se queira que os detalhes de luz e sombra no objeto sejam preservados, é possível utilizar os *light passes* difusos e especulares para reintroduzir a informação no *render* (figura 35).

Figura 35 - *Cryptomatte* utilizado para isolar o efeito da textura a um único objeto (à esquerda), e os *light passes* utilizados para reintroduzir informação de luz e sombra ao mesmo (à direita)



Fonte: Autor

Como é possível notar no exemplo acima, essa técnica de retexturização não funciona para reflexos; apesar da esfera vermelha ter sido substituída na imagem, o chão abaixo dela ainda reflete a cor original do objeto. Para que o reflexo correspondesse à nova textura, seria necessário fazer essa alteração dentro do software 3D e renderizar novamente o *frame*.

Contudo, mesmo com essa limitação, o *passe* de *UV* pode ser extremamente útil dependendo de como for implementado na pós-produção. No curta *Pintura Digital*, o *render*

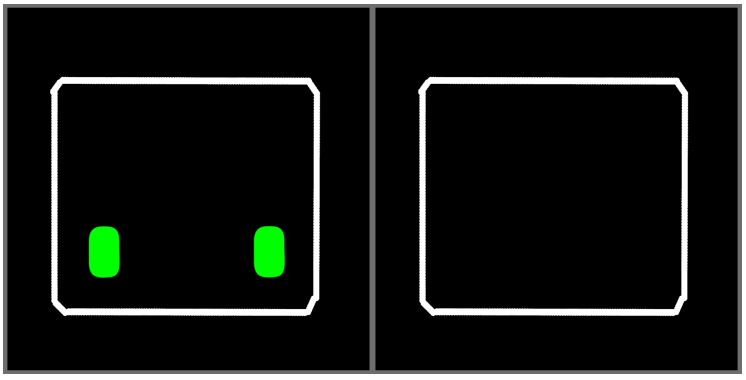
pass foi utilizado durante o *compositing* para mapear o rosto do personagem na tela (figura 36).

Figura 36 - Rosto do robô inserido durante o *compositing*

Fonte: Autor

O rosto 2D foi animado posteriormente à animação 3D, tendo como referência um guia simples (figura 37) que representava a posição dos olhos do protagonista dentro do monitor. Tal guia foi gerado usando o mapa de *UV* do personagem como base.

Figura 37 - Guia para a animação do rosto do robô



Fonte: Autor

Seguindo o modelo, diferentes expressões foram desenhadas em diferentes etapas da animação. Essas expressões foram então projetadas em todos os objetos da cena utilizando o *passe* de *UV*. Com a ajuda de uma máscara gerada pelo *passe cryptomatte*, restringiu-se o efeito para que o rosto aparecesse somente na área onde o monitor do personagem era visível. Abaixo (figura 38), é possível ver o resultado do processo.

Figura 38 - Imagem 2D do rosto do robô (à esquerda) sendo projetada no monitor (à direita)

Fonte: Autor

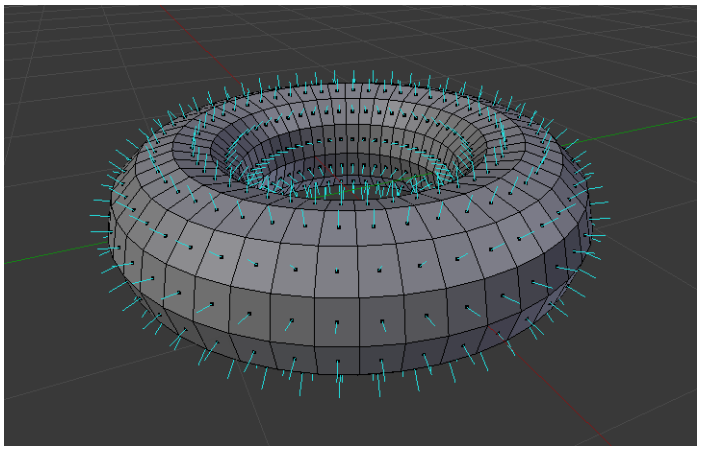
A escolha de implementar o rosto nessa etapa da produção ao invés de diretamente no software 3D foi para otimizar o método de trabalho da equipe. Isso permitiu que o rosto fosse animado *enquanto* os *frames* ainda eram renderizados; não foi necessário esperar a textura da tela ficar pronta para poder então começar o *render*.

Outra vantagem do rosto ter sido adicionado durante o *compositing* foi a flexibilidade que isso proporcionou à equipe. Qualquer alteração necessária na animação das expressões pôde ser realizada na pós-produção, sem nenhum custo de *render* adicional. Isso deu liberdade para a equipe experimentar diferentes opções e abordagens na animação do rosto, sem ter a preocupação de precisar retornar ao software 3D e gastar horas renderizando uma cena inteira novamente.

2.6.5 *Normal*

O *passe* de *Normal*, como o nome indica, é um mapa que contém as informações das *normais* de todos os objetos em cena (SELIN, 2021). Em 3D, *normal* é o nome que se dá à direção perpendicular à geometria de um objeto (BLENDER). Basicamente, é um vetor que indica a orientação na qual determinada face está apontando.

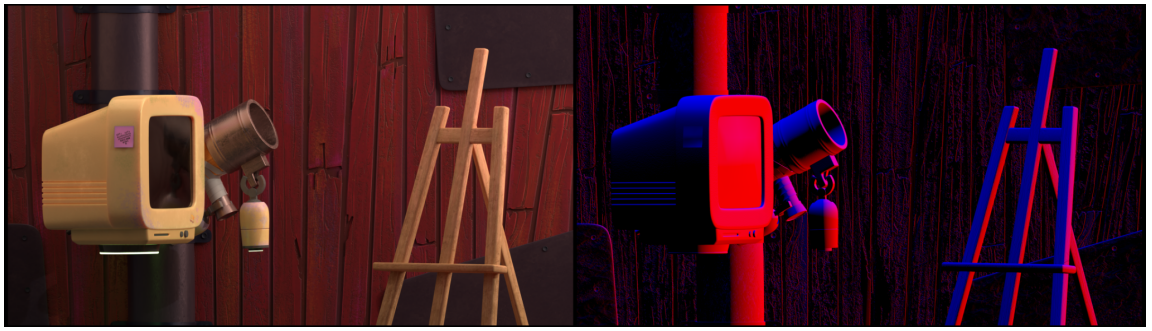
Figura 39 - Representação das normais das faces de uma geometria em 3D



Fonte: *Blender*

Por ser um vetor tridimensional, as informações de normal são registradas dentro do *render pass* em três canais diferentes: vermelho, verde e azul. Cada um desses canais representa um eixo dentro software 3D; x, y e z respectivamente. O resultado disso é um *render pass* com a seguinte aparência (figura 40):

Figura 40 - *Frame* renderizado (à esquerda) e seu respectivo *passe* de *normal* (à direita)



Fonte: Autor

Durante a pós-produção do projeto, foi possível utilizar o *passe* de *normal* no *Nuke* para extrair informações da direção dos objetos em cena e gerar uma máscara que englobasse apenas os pixels que tivessem uma mesma orientação (figura 41).

Figura 41 - Máscara criada usando as informações presentes no *passe* de *normal*

Fonte: Autor

Esta máscara resultante do processo foi usada então para auxiliar nos ajustes de cor em um momento do curta. Na cena em questão, uma explosão animada em 2D foi adicionada em cima do *render* 3D (figura 42).

Figura 42 - *Frame* do curta onde uma explosão 2D foi sobreposta em cima do *render* 3D

Fonte: Autor

Utilizando a máscara gerada pelo *passe* de *normal*, adicionou-se um brilho amarelado extra nas áreas da cena que estavam viradas para a explosão (figura 43). Isso resultou em uma camada a mais de realismo ao *render*, criando a ilusão de que a luz do fogo estava interagindo com os objetos 3D.

Figura 43 - *Frame* do curta após correção de cor

Fonte: Autor

3. Conclusão

Considerando a proposta deste artigo, pode-se dizer que o objetivo de se criar um curta-metragem animado em 3D foi bem sucedido. A presença de *render passes* - apesar de aumentarem o tempo de *render* - foi vital para o êxito do projeto; sem eles, não teria sido possível nem terminar o curta no prazo e nem obter o nível de qualidade desejado.

Em termos de otimização, os *passes* de *denoising data* ajudaram a reduzir o tempo de produção em mais de 20 minutos por *frame*, resultando em imagens sem ruído que de outras formas só seriam possíveis de alcançar através de uma quantidade de *samples* muito alta na renderização. Essa redução de tempo não apenas compensou aquele que foi gasto gerando os *render passes*, como também salvou horas de produção considerando todos os 1256 *frames* que precisaram ser renderizados no curta. O autor acredita que todas as produções, principalmente as independentes e estudantis por terem menos verba, se beneficiariam em adotar uma etapa de *denoising* em sua *pipeline*.

Em termos de estética, os *render passes* permitiram à equipe realizar diferentes experimentações durante a pós-produção do projeto, seja testando variações de animações de rosto para o personagem, seja modificando os *light passes* até se obter um resultado que eles considerassem visualmente agradável. Isso forneceu aos integrantes flexibilidade para manipular vários aspectos da cena sem precisar voltar ao software 3D e renderizar novamente os *frames* alterados.

Alguns dos *passes* e técnicas mencionados neste artigo podem não ser tão relevantes para todas as produções, uma vez que as estratégias retratadas aqui tiveram como foco

especificamente o desenvolvimento do curta *Pintura Digital*. Alguns projetos podem precisar de mais ou menos *render passes* do que outros, dependendo do resultado que se deseja alcançar. Por via das dúvidas, se possível, o autor recomenda estudar e renderizar o máximo de *render passes* possíveis, mesmo que não se tenha uma intenção explícita de usá-los. Afinal, é comum imprevistos ocorrerem em produções, e nunca se sabe quando uma alteração de última hora pode ser necessária.

Referências

AUTODESK. **AOVs for Image Compositing**.
Disponível em: <https://docs.arnoldrenderer.com/display/A5AFCUG/AOVs+for+Image+Compositing>.
Acesso em: 28/08/2021.

BENZ, Greg. **8, 12, 14 vs 16-Bit Depth: What Do You Really Need?!**, setembro de 2018.
Disponível em: <https://petapixel.com/2018/09/19/8-12-14-vs-16-bit-depth-what-do-you-really-need/>.
Acesso em: 28/09/2021.

BLENDER. **Blender Manual**.
Disponível em: <https://docs.blender.org/manual/en/latest/index.html>
Acesso em: 01/09/2021.

BUGLIA, Fernando. **Entenda a Diferença Entre Fontes de Luz Primária e Secundária**, dezembro de 2016.
Disponível em: <https://infoenem.com.br/entenda-diferenca-entre-fontes-de-luz-primaria-e-secundaria/>.
Acesso em: 28/08/2021.

CHARLEAUX, Lupa. **Open source: o que é e como funciona**, abril de 2021.
Disponível em: <https://www.tecmundo.com.br/software/215130-open-source-funciona.htm>.
Acesso em: 04/09/2021.

FOUNDRY. **Look development, lighting and texturing on Spider-Man: Into the Spider-Verse**, abril de 2019.
Disponível em: <https://www.foundry.com/insights/film-tv/into-spiderverse>.
Acesso em: 20/08/2021.

GLAWION, Alex. **Best Renderers (Render Engines) for Blender in 2021**, 2021.
Disponível em: <https://www.cgdirector.com/best-renderers-render-engines-for-blender/>.
Acesso em: 26/08/2021.

GRIGGS, Mike. **3D rigging: all you need to know to get started**, julho de 2019.
Disponível em: <https://www.creativebloq.com/features/3d-rigging-all-you-need-to-know-to-get-started>.
Acesso em: 04/09/2021.

GRIGGS, Mike. **Compositing in animation: Learn the basics**, novembro de 2020.
Disponível em: <https://www.creativebloq.com/how-to/compositing-basics>.
Acesso em: 25/08/2021.

HECKMANN, Chris. **What is Compositing? VFX Compositing Techniques Explained**, dezembro de 2020.
Disponível em: <https://www.studiobinder.com/blog/what-is-compositing-definition/>.
Acesso em: 24/08/2021.

ISMAIL, Ali. **Improving 3D Renders in Post Processing**, março de 2019.
Disponível em: <https://www.ebalstudios.com/blog/improving-3d-renders>.
Acesso em: 20/08/2021.

KAUSHIK, Parth. **Top 6 VFX software that dominate the industry**, fevereiro de 2020.
Disponível em:
<https://www.animationxpress.com/latest-news/top-6-vfx-effects-software-that-dominate-the-industry/>.
Acesso em: 03/09/2021.

LEAVITT, Logan. **6 Common (But Not Obvious) Nodes For Nuke Beginners**, outubro de 2018.
Disponível em: <https://www.actionvfx.com/blog/6-common-but-not-obvious-nodes-for-nuke-beginners>.
Acesso em: 03/09/2021.

LEE, Tina. **The Best Rendering Software for CG Lighting for Animation**, fevereiro de 2019.
Disponível em: <https://academyofanimatedart.com/the-best-rendering-software-for-cg-lighting-for-animation/>.
Acesso em: 28/08/2021.

LEHRER, Jonah.**1,084 days: How Toy Story 3 was made**, 2010.
Disponível em: <https://www.wired.co.uk/article/how-toy-story-3-was-made>.
Acesso em: 25/08/2021.

MAHER, Michael. **Rotoscoping: From Early Animation to Blockbuster VFX**, setembro de 2015.
Disponível em: <https://www.rocketstock.com/blog/rotoscoping-from-early-animation-to-blockbuster-vfx/>.
Acesso em: 31/08/2021.

MAIO, Alyssa. **What is Depth of Field? Examples of Shallow vs Deep Depth of Field**, julho de 2019.
Disponível em: <https://www.studiobinder.com/blog/what-is-depth-of-field-definition/>.
Acesso em: 20/09/2021.

MORRISON, Geoffrey. **Smooth movies: Are high-frame rate films a good idea?**, abril de 2020.
Disponível em:
<https://www.cnet.com/tech/home-entertainment/smooth-movies-are-high-frame-rate-films-a-good-idea/>.
Acesso em: 29/08/2021.

NAGHDI, Arash; ADIB, Payam. **3D animation pipeline: A Start-to-Finish Guide (2021 update)**, 2021.
Disponível em: <https://dreamfarmstudios.com/blog/3d-animation-pipeline/>.
Acesso em: 24/08/2021.

NASA. **Earth**, 2009.
Disponível em: <https://nasa3d.arc.nasa.gov/detail/ear0xuu2>.
Acesso em: 02/09/2021.

OOST, Richard van der. **Render Smarter in Blender, with Adaptive Samples**, 2021.
Disponível em: <https://blendergrid.com/learn/articles/render-smarter-in-blender-with-adaptive-samples>.
Acesso em: 27/08/2021.

PUTRAMA, I Made; DARMAWIGUNA, I Gede Mahendra; SANTYADIPUTRA, Gede Saindra. **An Alternative Rendering Solution in Animated Movie Making for Final Year Students: A Case Study** . Atlantis Press, 2017, p. 60-65. DOI: <http://dx.doi.org/10.2991/icirad-17.2017.12>.

RENDERMAN. **Secondary Outputs (AOVs)**.
Disponível em:
<https://renderman.pixar.com/resources/RenderMan_20/secondaryOutputs.html>.
Acesso em: 28/08/2021.

SELIN, Erik. **Render passes in Blender Cycles: Complete guide**, 2021.
Disponível em: <https://artisticrender.com/render-passes-in-blender-cycles-complete-guide/>.
Acesso em: 20/08/2021.

SHANASA, Dhyan. **Usando Render Passes para alterar reflexões em pós**, julho de 2016.
Disponível em: <https://www.layerlemonade.com/vfx/aprenda-criar-scripts-no-after-effects>.
Acesso em: 20/08/2021.

TRAJE, Ben. **What is Look Development?**, 2019.
Disponível em: <https://conceptartempire.com/look-development/>.
Acesso em: 04/09/2021.

WALT DISNEY ANIMATION STUDIOS. **Disney's Practical Guide to Path Tracing**. YouTube, 2016.
Disponível em: <https://www.youtube.com/watch?v=frLwRLS_ZR0>.
Acesso em: 26/08/2021.

YENG, Huey. **Nuke Tips – Masking in Nuke**, abril de 2015.
Disponível em: <https://taukeke.com/2015/04/masking-in-nuke/>.
Acesso em: 31/08/2021.