UNIVERSIDADE
BEIRA INTERIOR

Post-Processing para videojogos:
A importância do post-Processing no desenvolvimento de videojogos
Versão final

Sandro Daniel Gomes Gonçalves

Dissertação para obtenção do Grau de Mestre em
Design e Desenvolvimento de jogos digitais
(2º ciclo de estudos)

Orientador: Prof. Doutor Flávio Henrique de Almeida

janeiro de 2022

Resumo

Os videojogos são uma média bem estabelecida presentemente, e a indústria em torno destes é também das mais lucrativas mundialmente. Parte deste sucesso provém da evolução visual dos mesmos e da persecução de um dia ser possível criar jogos cujo realismo visual (e não só) seja indistinguível, atravessando assim a linha entre o real e o virtual, mas por enquanto, esse objetivo ainda é distante; a distância atualmente já é bem pequena, e há exemplos em que se nada for dito e o contexto não existir, um indivíduo pode até ser enganado em acreditar que um jogo não o seja. Esta evolução vem de imensos avanços tecnológicos, assim como a criação de várias ferramentas e programas, uma destas ferramentas é o *post-Processing,* utilizado também noutros média como a fotografia, som e cinema.

Esta dissertação e o projeto que a acompanha visam principalmente conseguir entender como utilizar *post-Processing* durante o desenvolvimento de um jogo ou outro produto, na Unreal Engine, nomeadamente na versão 4.26, mas também a analisarem-se as potencialidades do *post-Processing* no desenvolvimento de jogos, o que se pode atingir com a ferramenta, os obstáculos que podem dificultar a sua aprendizagem, as bases fundamentais para conseguir dominar o *post-Processing* e uma análise da sua importância.

O projeto serve essencialmente como uma tela onde se podem realizar experimentações com a ferramenta, percebendo assim como a teoria funciona, mas também ganhando o conhecimento prático, com o intuito de criar um demo jogável cuja fidelidade visual seja o cúmulo de toda a aprendizagem do mestrado de Design e Desenvolvimento de Jogos Digitais, complementado pela aprendizagem do *post-Processing*.

Em suma, procede-se à análise do quão importante é atualmente o *post-Processing* durante a fase de desenvolvimento de um videojogo, importância esta associada aos resultados obtidos com o uso do mesmo para fins narrativos e mecânicos para além dos visuais ou estéticos.

Palavras-Chave

Videojogos, post-Processing, Unreal, gráficos, Unity

Abstract

Video games are a well-established medium nowadays, and the industry around them is also one of the most profitable worldwide, part of this success comes from their visual evolution and the pursuit of one day being possible to create games whose visual realism is indistinguishable, thus crossing the line between the real and the virtual, but while that goal is still distant, the distance today is already very small, and there are examples where if nothing is said and the context does not exist, an individual can even be fooled into believing that a game is not. This evolution comes then from immense technological advances, as well as the creation of various tools and programs, one of these tools is post-Processing, also used in other media such as photography, sound and cinema.

This dissertation and the project that accompanies it aim to understand how to use post-Processing during the development of a game or other product, on the Unreal Engine, namely version 4.26, but also to analyse the potential of post-Processing in game development, what can be achieved with the tool, the obstacles that can hinder its learning as well as the fundamental bases to master post-Processing, and an analysis of its importance.

The project serves essentially as a canvas where one can experiment with the tool, thus understanding the theory of how it works, but also gaining practical knowledge of its use, with the aim of creating a playable demo whose visual fidelity is the culmination of all the learning of the master's degree in Digital Game Design and Development, complemented by the learning of post-Processing.

Concluding with an analysis of how important post-Processing is during the development phase of a video game today. This Importance is associated with the results obtained with its use for narrative and mechanical purposes besides the visual or aesthetic.

Keywords

Videogames, post-Processing, Unreal, graphics, Unity

Índice

Lista de Figuras

Introdução

Os videojogos são uma categoria de média em constante evolução e expansão. Sendo inclusive a indústria de entretenimento mais lucrativa no momento, chegando a ter o dobro dos lucros da indústria televisiva e cinematográfica juntas, mas nas décadas de 80 e 90, esta categoria de média era bastante jovem e amplamente incompreendida, especialmente no que tocava a videojogos com um tom mais gráfico ou sério como, por exemplo, DOOM (id Software, 1993), Mortal Kombat (Midway Games, 1992), Curster's Revenge (Mystique, 1982) entre outros. Atualmente, as controvérsias, ataques judiciais e polémicas em torno dos videojogos é escassa, e está mais focada noutros componentes, como, por exemplo, companhias adicionarem métodos de apostas nos seus jogos para terem mais lucro.

Assim sendo, esta categoria de média começa a entrar numa idade de estabilização onde outras categorias de média já se encontravam. Veja-se que só a fotografia, por exemplo, existe há aproximadamente mais de duzentos anos, e o cinema/filmes há mais de cento e trinta; no entanto, os videojogos só existem há oitenta anos.

Assim sendo, os videojogos têm sofrido alterações e evoluções, e atualmente a indústria dos videojogos cria experiências interativas e simulações em vez de puramente entretenimento passivo. É desta forma que se tem vindo a criar vários ramos no que toca aos jogos, desde jogos comuns a videojogos que servem para contar uma história que outras categorias de média não conseguiriam (ou teriam dificuldade), para ajudar a tratar certas doenças, com finalidade educativa, como simuladores para treinar utilizadores para tarefas reais (por exemplo, para treinar futuros tripulantes de um submarino, ou condutores a tirarem a carta de condução), etc.

Contudo, a diversificação dos videojogos existentes não é a única evolução que os videojogos sofreram ao longo dos anos, pois a complexidade destes em todos os seus aspetos também sofreu alterações. Por exemplo, Space Invaders (Taito, 1978) (**Figura 1**), que tentava proporcionar a ideia de uma batalha espacial contra seres espaciais, possuía sons, jogabilidade e gráficos bastante simples. Atualmente, jogos como Star Wars: Squadrons (Eletronic Arts, 2020) (**Figura 2**) proporcionam uma jogabilidade percebida como real, onde o jogador pilota uma nave pelo espaço. No entanto, este encontra-se visualmente na cabine da mesma, interpretando o papel de piloto, e tanto no som e grafismo, assim como na jogabilidade, o jogo entende-se como sendo bastante realista.

Figura 1 - Captura de ecrã do jogo *arcade* Space Invaders (Taito, 1978).

Figura 2 - Captura de ecrã do jogo Star Wars: Squadrons (Eletronic Arts, 2020).

Motivo

Considerando as mudanças anteriormente referidas, as três que o jogador normalmente tem em maior consideração, são a jogabilidade (os controlos, movimento, história, narrativa, etc.), o som (desde a música de fundo aos efeitos sonoros como tiros, motores, explosões, etc.), e, provavelmente, a que costuma ressaltar mais à vista da maioria dos jogadores: o grafismo (realista ou estilizado). É conhecido que a evolução gráfica dos videojogos é uma constante, com videojogos atualmente no limite do que começa a ser considerado como real, começando a ser comum que a distinção do real e do virtual seja impossível.

É então na categoria visual dos videojogos que este projeto e dissertação se encontram, mais precisamente na parte do *post-Processing*, devido principalmente à complexidade teórica e toda a reflexão e meditação que certos elementos do *post-Processing* apresentam, de modo a usufruir corretamente desses efeitos.

Parte do interesse em estudar esta área do desenvolvimento de videojogos provém também do facto de, atualmente, nas áreas de fotografia, cinema e animação (tanto 3D como tradicional) se falar da importância de um bom tratamento de imagem e *post-Processing* da mesma, de modo a esta atingir o seu potencial. Este assunto é, no entanto, pouco falado na área de desenvolvimento de videojogos, em relação aos supramencionados.

Partindo então da complexidade na utilização devida do *post-Processing* e a falta de documentos que falem sobre o mesmo no desenvolvimento de videojogos, surge a necessidade do desenvolvimento desta dissertação e projeto.

Parte teórica (Estudo do *post-Processing*)

Servindo como primeiro capítulo da parte teórica desta dissertação, começa-se por falar da história e evolução do *post-Processing* desde os princípios da fotografia, abordando a história dos videojogos, a evolução da fotografia, cinema, computadores e consolas e ainda vários pontos de interesse na implementação de *post-Processing* nos videojogos.

No segundo capítulo, inicia-se o estudo do tema, começando-se por uma breve explicação, no ponto de vista de utilização da Unreal Engine, versão 4.26.2, do que é o *post-Processing* no desenvolvimento de videojogos e as suas componentes. A escrita é realizada mediante o que é analisado na documentação oficial da Unreal Engine, disponível *online* pela Epic Games, assim como na pesquisa noutras fontes, em simultâneo. O que provém dessa documentação e pesquisa é estudado e confirmado no projeto, para obter uma validação tanto teórica como prática, associando o conhecimento e pesquisa teóricos com os estudos, análise e validações práticas. Isto resulta assim na explicação das várias componentes, de como proceder com a utilização das mesmas e algumas orientações no que toca à análise dos problemas em questão de modo a apresentar a melhor solução de uma maneira rápida e eficaz. Fala-se ainda de alguma história de cada componente e, quando possível, uma breve explicação resumida da ciência e programação por trás de como a componente funciona.

No terceiro capítulo, confere-se o impacto que o uso da ferramenta tem, tanto visualmente, mas também nos aspetos narrativos, jogabilidade e *interface*, explorando vários exemplos de videojogos que utilizam o *post-Processing* para intensificar uma determinada ideia ou conceito, como é o exemplo da visão noturna em jogos modernos de guerra, efeito este que pode ser atingido ao remover a saturação total da cor, convertendo o que se vê numa imagem a preto e branco, e de seguida aplicar um filtro com uma tonalidade comummente verde para reproduzir este efeito de visão noturna ou de óculos infravermelhos, e é

também através desses exemplos que se demonstra como o *post-Processing* pode facilitar a produção de um videojogo no que toca à implementação de conceitos e ideias.

No quarto e último capítulo, conclui-se com a resposta à pergunta de abertura desta parte; fala-se da importância de utilizar *post-Processing* durante o desenvolvimento de um videojogo, da dificuldade de aprender e utilizar *post-Processing* devidamente e da importância que este apresenta em relação a outros pontos do desenvolvimento de um videojogo.

Parte prática (Relatório de desenvolvimento do projeto)

Devido aos pontos da dissertação, realizou-se o projeto acompanhante com o intuito de funcionar como elemento de teste dos vários efeitos constituintes do *post-Processing*. Ademais, utiliza-se a documentação oficial da Unreal Engine 4.26 (Epic Games 2020), com o método de experimentação e verificação, permitindo-se assim fornecer um conhecimento prático e apresentar um demo jogável, que servirá também para avaliar a opinião de vários utilizadores durante o seu desenvolvimento; esta opinião serve também como forma de aprovação do *post-Processing* no projeto.

Devido ao projeto ser partilhado com outra dissertação (Design de Som nos Videojogos. Criação do Projeto The Last Star. Escrita por Henrique Candeeias sobre a orientação do professor doutor Flávio Almeida) e possuir o objetivo final de criar um produto final de qualidade, o relatório relativo ao projeto possuirá elementos externos ao tópico explorado, relatando alterações que este sofrerá mediante o ponto de vista do artista visual do videojogo.

Assim sendo, o relatório possuirá elementos para além do *post-Processing*, mas na sua essência serve para relatar as decisões tomadas, assim como a reflexão por trás de cada, e, em última análise, facilitando a compreensão dos pontos escritos previamente durante o estudo do *post-Processing*.

Objetivo

Este trabalho tem como principal finalidade poder servir como documentação de apoio a uma multiplicidade de artistas e utilizadores na implementação das várias componentes pertencentes ao *post-Processing* através dos vários exemplos, comparações, explicação objetiva de cada controlador, dicas e recomendações.

Serve ainda esta dissertação o propósito de consolidação pessoal do conhecimento de *post-Processing* acumulado ao longo dos anos, enquanto artista, sendo eliminadas as questões que punham em causa a coerência do que era adquirido apenas pela prática.

Parte teórica

Qual é a importância do post-Processing no desenvolvimento de videojogos?

A questão que se coloca nesta dissertação e projeto recai no estudo de quão importante é a utilização de *post-Processing* no desenvolvimento de um videojogo atualmente. Com isto, ao longo dos capítulos que se seguem será realizada uma análise da história do *post-Processing*, definido o que é o *post-Processing* no desenvolvimento de videojogos, assim como este é constituído (efeitos e filtros), comentadas algumas finalidades do *post-Processing* no desenvolvimento de um videojogo, para além dos efeitos meramente visuais, e finalmente realizada uma rápida análise da sua importância, mediante todos estes parâmetros.

1 A evolução do post-Processing e dos videojogos.

Tendo em conta que o *post-Processing* envolve várias áreas, desde Pintura, Fotografia e Cinema assim como a arte gerada por computador (CGI) mas também áudio, fabrico de produto entre muitos outros, é normal que quando se observa a história deste, que esta seja bastante ampla, e que, abranja várias áreas. No entanto, foquemo-nos apenas nas áreas das artes visuais, ou seja, reveja-se a história do *post-Processing* nas áreas da fotografia, cinema e CGI.

1.1 Inicio e criação do Processamento de imagem.

Retrocedendo bastante no tempo até ainda antes de Cristo, a ideia da câmara escura (**Figura 4**) já existia, e é difícil saber quem inventou ou descobriu esse conceito, visto que este era falado por Mozi[1], Aristóteles[2] e Alhazém[3] (Hummel e Needham, 1962, p. 98). Este termo foi, entretanto, utilizado algumas vezes ao longo da história, para várias experiências, proveniente de vários autores e pontos no globo, mas é só no século XIX que aquilo que se conhece nos termos de atualmente como fotografia, dá o seu início.

[1] Mozi (Latinizado como Micius 470 - 391 AC), nome original Mo Di (墨翟), foi um filósofo chinês que fundou a escola do Moísmo durante o período das Cem Escolas de Pensamento (parte inicial do período dos Reinos Combatentes de 475–221 AC).
O antigo texto Mozi contém material atribuído a ele e a seus seguidores.

[2] Aristóteles (Estagira, 384 a.C. — Atenas, 322 a.C.) foi um filósofo grego durante o período clássico da Grécia antiga, fundador da escola peripatética e do Liceu, aluno de Platão e professor de Alexandre, o Grande.

[3] Abu Ali Haçane ibne Haitão, conhecido também pela forma latinizada Alhazém, nasceu no ano 965 em Baçorá, agora Iraque e morreu em 1040 na cidade do Cairo. Físico e matemático persa. Pioneiro da ótica, depois de Ptolomeu. Foi um dos primeiros a explicar o fenômeno dos corpos celestes no horizonte.

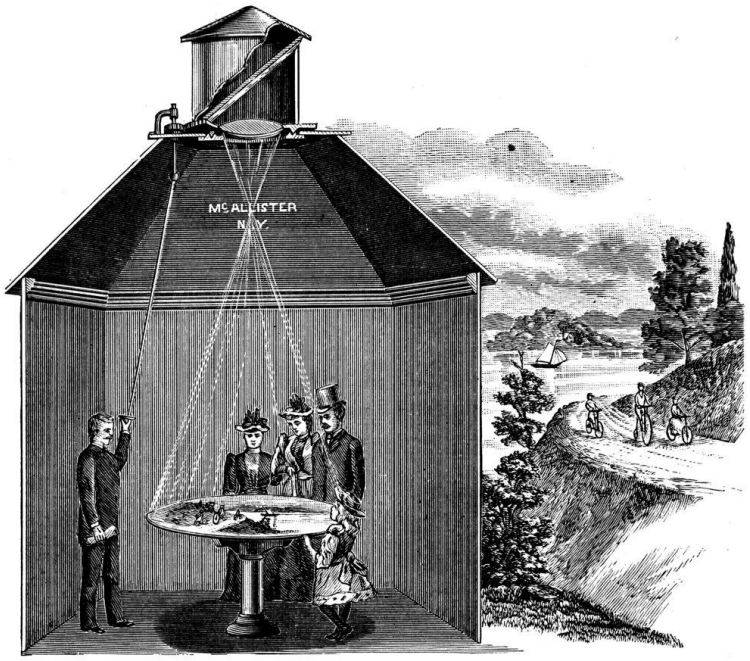

Figura 3 - (quatro) Pessoas a utilizar uma câmara escura, permanecendo invisíveis na imagem obtida (Johannes Kepler, s.d).

Os primeiros exemplos disto dão-se entre 1814 e o fim dos anos 1820, através de várias experiências de fotogravura realizadas pelo inventor francês, Joseph Nicéphore Niépce[4] uma das quais, está a obra conhecida como Point de vue du Gras (Vista da Janela em Le Gras, **Figura 4**), conhecida como a primeira fotografia no mundo, que na sua essência e como o nome indica, mostrava a vista da janela da oficina de Niépce, o processo envolveu oito horas de luz solar direta (Hirsch, 2017, pp. 11–13).

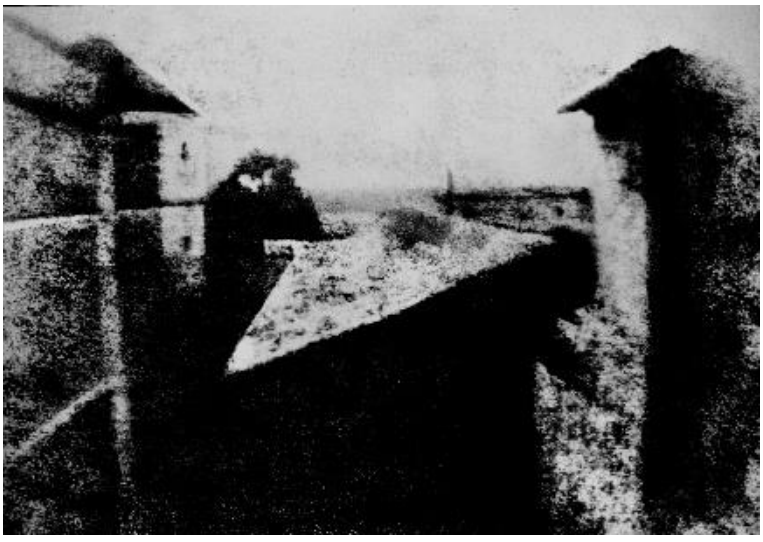

Figura 4 - Point de vue du Gras, França (Joseph Nicéphore Niépce, 1827)

[4] Joseph Nicéphore Niépce (Chalon-sur-Saône, 7 de março de 1765 — Saint-Loup-de-Varennes, 5 de julho de 1833) foi um inventor francês responsável por uma das primeiras fotografias. Mediante outras invenções suas, está também, o Pyréolophore, um dos primeiros motores de combustão interna, concebido, criado e desenvolvido com o seu irmão Claude Niépce.

Entretanto, avançando um pouco no tempo, e é nos 1840s, que se dá o início de experimentações com fotografias a cor. Em 1846 o primeiro caso de edição fotográfica é documentado por Calvert Richard Jones[5], que na altura fotografara quatro frades franciscanos no telhado de um edifício em Malta (Sears, 2016; Young, s.d.), mas, havia um problema com a foto na sua opinião, pois, nas costas dos frades, estava uma estátua (que até recentemente pensava-se ser um quinto frade (**Figura 7**) e embora não se saiba bem a razão do porquê, nem ao certo quem realizou a alteração, (previamente pensado ser devido a Jones não gostar da forma como o "quinto frade" destruía a integridade da cena), mas, a estátua foi removida da pintura. O processo utilizado foi relativamente simples, implicando apenas que, alguém, através do uso de tinta da Índia, pinta-se no negativo da foto, neste caso, a estátua (**Figura 5**), o que, depois na revelação da mesma, faria com que a estátua desaparece-se no branco do céu (**Figura 6**) (Cilia, 2020).

Figura 5 - Negativo alterado da obra, Capuchin Friars, Valetta, Malta (Calvert Richard Jones, 1846).

Figura 6 - Capuchin Friars, Valetta, Malta (Calvert Richard Jones, 1846).

Figura 7 - Rough Print n. º8, Capuchin Friars, Valetta, Malta (Calvert Richard Jones, 1846).

[5] Calvert Richard Jones (4 de dezembro de 1804 – 7 de novembro de 1877), foi um matemático e pintor galês, mais conhecido pelas suas paisagens marinhas.

É nos 1830s que o cinema começa os seus primeiros passos através de conceitos como o estroboscópio[6] (**Figura 8**) de Simon von Stampfer[7] na Áustria, o fenacistoscópio[8] (**Figura 9**) de Joseph Plateau[9] na Bélgica e o zootrópio[10] (**Figura 10**) de William Horner[11] na Grã-Bretanha, e em 1845 Francis Ronalds[12] inventa a primeira câmara capaz de captar registos contínuos de várias indicações de instrumentos meteorológicos e geomagnéticos ao decorrer do tempo.

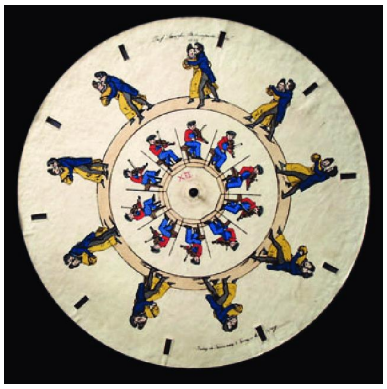

Figura 8 - Casal Valsando e um Violinista, disco estroboscópico, Observatório Kremsmünster, Áustria (Simon von Stampfer, 1832).

[6] O estroboscópio consiste num dispositivo ótico que permite estudar e registar o movimento contínuo ou periódico de elevada velocidade de um corpo, com o objetivo de o fazer parecer estacionário. Obtém-se um conjunto de imagens discretas, mas que são representativas do percurso que o corpo descreve. Esse efeito é conseguido através da alternância entre a iluminação com uma luz intensa e o bloqueamento dessa luz com um diafragma — lâmpada estroboscópica.

[7] Simon Ritter von Stampfer (26 de outubro de 1792 — Viena, 10 de novembro de 1864) foi um matemático austríaco, agrimensor e inventor.

[8] Fenacistoscópio, é um dispositivo inventado por Joseph Plateau para demonstrar a sua teoria da persistência na retina em 1829 e consiste em vários desenhos do mesmo objeto, em posições ligeiramente diferentes, distribuídos por uma placa circular lisa, que quando a placa gira em frente a um espelho, cria a ilusão de uma imagem em movimento.

[9] Joseph Antoine Ferdinand Plateau (Bruxelas, 14 de outubro de 1801 — Gante, 15 de setembro de 1883) foi um físico belga.

[10] O zootrópio, é uma máquina criada em 1834 por William George Horner, composta por um tambor circular com pequenas janelas recortadas, através das quais o espetador olha para desenhos dispostos em tiras. Ao girar, o tambor cria uma ilusão de movimento.

[11] William George Horner (Bristol, 1786 — Bath, 1837) foi um matemático britânico, proficiente em clássicos da matemática, foi um mestre-escola, diretor e auxiliar / continuo (schoolkeeper), que escreveu extensivamente sobre equações funcionais, teoria dos números e teoria da aproximação, mas também sobre ótica.

[12] Francis Ronalds (21 de fevereiro de 1788 – 8 de agosto de 1873) foi um cientista e inventor inglês, e possivelmente o primeiro engenheiro eletrotécnico, nomeado cavaleiro por criar o primeiro telégrafo elétrico funcional a uma distância substancial.

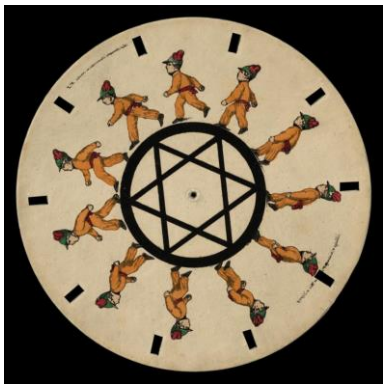

Figura 9 - Stroboscopische Scheibe n.º X (Joseph Plateau, 1833)

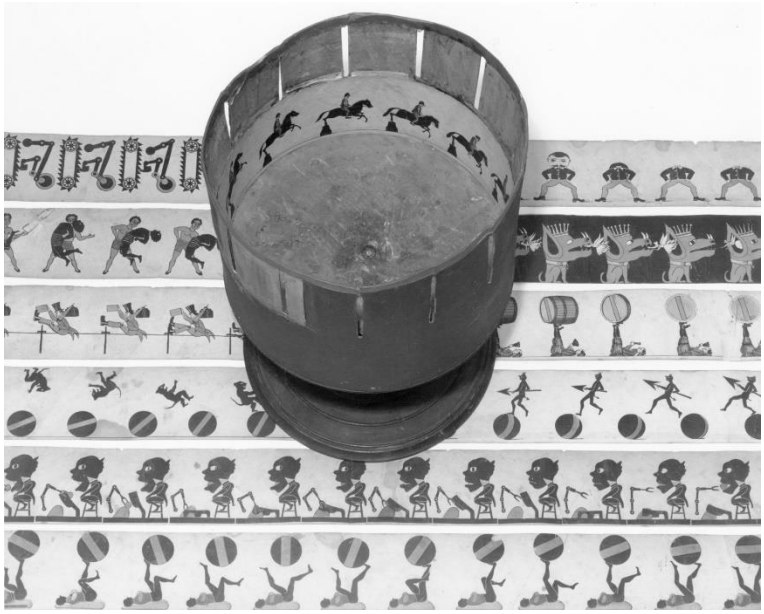

Figura 10 - Zootrópio com seis tiras de animação zootrópica (William Horner, 1834)

Em 1858, Henry Peach Robinson[13], publica Fading Away (**Figura 11**), uma fotomontagem realizada através da combinação de cinco negativos diferentes, registando assim no tempo uma das primeiras fotomontagens (The Metropolitan Museum of Art, s.d.).

[13] Henry Peach Robinson (Ludlow, Shropshire, 9 de julho de 1830 – 21 de fevereiro de 1901, Royal Tunbridge Wells, Kent) foi um fotógrafo pictórico inglês maioritariamente conhecido pelas suas fotografias pioneiras devido a combinar negativos ou impressões para formar uma única imagem criando assim um dos primeiros exemplos de fotomontagem. Também era um participante vigoroso em debates contemporâneos na imprensa fotográfica e associações sobre a legitimidade da 'fotografia artística' e, em particular, a combinação de imagens separadas numa só.

Figura 11 - Fading Away (Henry Peach Robinson, 1858).

1.2 Evolução até ao post-Processing nos videojogos.

Entretanto, em 1861, a primeira fotografia permanentemente a cores (visto que até agora, as fotografias a cores prévias eventualmente acabam por descolorar) é publicada por Thomas Sutton[14] (**Figura 12**) algo conseguido devido ao princípio publicado por James Clerk Maxwell[15] previamente em 1855 que consistia em realizar três fotografias a preto e branco, através de três filtros, nomeadamente um filtro vermelho, um azul e um verde (**Figura 13**), proporcionando assim ao fotógrafo os três canais necessários para recriar uma imagem colorida (Flueckiger, 2012).

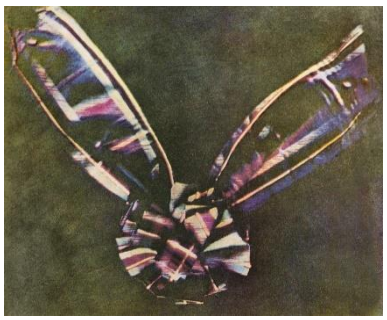

Figura 12 - Tartan Ribbon, (Thomas Sutton, 1861).

[14] Thomas Sutton (1819 — Kensington, 19 de março de 1875) foi um fotografo autor e inventor inglês.
[15] James Clerk Maxwell (13 de junho de 1831 – 5 de novembro de 1879) foi um cientista escocês no campo da física matemática onde a sua realização mais notável formulou a teoria clássica da radiação eletromagnética, reunindo pela primeira vez eletricidade, magnetismo e luz como diferentes manifestações do mesmo fenómeno.

Figura 13 - Os três filtros de cor utilizados na foto Tartan Ribbon (Thomas Sutton, 1861).

Isto permitirá também que, se as fotos fossem impressas em filmes transparentes, que estas pudessem ser projetadas através de filtros de cor semelhantes, criando o que é conhecido como o método aditivo da reprodução de cor. Já no caso de impressões em papel, teriam de recorrer a impressões em carbono das três imagens, nas suas cores complementares, utilizando neste caso o método subtrativo, algo onde Louis Ducos du Hauron[16] e Charles Cros[17] foi pioneiro no final dos anos de 1860s (Hannavy, 2007, p. 448 e 351). Foi também graças a estes princípios que Sergey Mikhaylovich Prokudin-Gorsky[18] conseguiu criar uma câmara especial que expunha as imagens filtradas pelas três cores, em partes diferentes de uma placa oblonga, mas devido a estas exposições não serem simultâneas, era comum que as suas fotos exibissem o que é conhecido como, aberração cromática (Coe, 1978) (**Figura 14** e **Figura 15**).

Figura 14 - Exemplo de uma imagem com aberração cromática (Imagem do autor).

[16] Louis Arthur Ducos du Hauron (8 de dezembro de 1837, Langon — 31 de agosto de 1920, Agen) foi um francês pioneiro na fotografia colorida, além de ser criador dos princípios fundamentais da cinematografia atual, ao lado de Henry – Desiré Du Mont.

[17] Hortensius-Emile Charles Cros (Fabrezan, 1 de outubro de 1842 — 9 de agosto de 1888) foi um poeta e inventor francês.

[18] Sergei Mikhailovitch Prokudin-Gorski (30 de agosto de 1863 - 27 de setembro de 1944) foi um fotógrafo russo que se dedicou sua carreira ao avanço da fotografia. Nasceu em Funicova Gora, no Oblast de Vladimir, Rússia, em 1863 e formou-se químico. Estudou com renomeados cientistas em São Petersburgo, Berlim e Paris, desenvolvendo as técnicas para as primeiras fotografias a cores.

Figura 15 - Correção da aberração cromática da figura anterior (Imagem do autor).

Em 1873, a descoberta da sensibilização de corantes na tinta, por Hermann Vogel[19] ajudou imenso a que a fotografia a cores se aproximasse da comercialização, visto que até agora havia problemas relacionados com a sensibilidade dos materiais fotográficos especialmente com vermelho (Hannavy, 2007, p. 1456). A 1879, Joseph Wilson Swan[20] inaugura a primeira produção de papel fotográfico especial, à base de halogéneo e prata gelatinosos, algo que se tornou o elemento principal na produção de papel fotográfico e ainda é utilizado industrialmente presentemente, e nesta altura os trabalhadores envolvidos nas revelações fotográficas já podiam ligeiramente alterar o contraste e tonalidades da imagem (Hannavy, 2007, p. 1367).

É também neste ano de 1879 que, George Eastman[21], viaja até Londres, considerada ser na altura o centro do mundo fotográfico e empresarial, de modo a obter uma patente na sua máquina de revestimento de placas. No ano seguinte, em 1880, Eastman aluga o terceiro andar de um prédio na State Street em Rochester e começa a fabricar placas secas para venda. Graças a isto, o sucesso no empreendimento da chapa seca levou a que o empresário Henry Alvah Strong[22] investisse algum dinheiro na empresa de Eastman, que em 1881 levou a uma parceria entre ambos, criando a Eastman Dry Plate Conpany (Eastman Kodak Company, s.d.). No ano 1882, Étienne-Jules Marey[23] inventa uma 'arma' cronofotográfica, capaz de capturar doze *frames* consecutivos por segundo. Em 1884 a parceria levou à criação de uma nova firma, conhecida como Eastman Dry Plate and Film Company devido à

[19] Hermann Wilhelm Vogel (26 de março de 1834 – 17 de dezembro de 1898) foi um fotoquímico e fotógrafo alemão que descobriu a sensibilização a corantes, algo que veio a ser de grande importância para a fotografia.

[20] Sir Joseph Wilson Swan (Bishopwearmouth, 31 de outubro de 1828 — Warlingham, 27 de maio de 1914) foi um físico e químico britânico, famoso pela invenção da lâmpada incandescente.

[21] George Eastman (Waterville, Nova Iorque, 12 de julho de 1854 — Rochester, Nova Iorque, 14 de março de 1932) foi um empresário estadunidense, fundador da Kodak e inventor do filme em rolo (filme fotográfico), o que ajudou a proporcionar a fotografia para o grande público.

[22] Henry Alvah Strong (30 de agosto de 1838 – 26 de julho de 1919) foi um empresário americano da fotografia e o primeiro presidente da Eastman Kodak Company.

[23] Étienne-Jules Marey (Beaune, 5 de março de 1830 — Paris, 21 de maio de 1904) foi um inventor e cronofotógrafo francês, o seu trabalho foi significativo no desenvolvimento da cardiologia, da instrumentação física, da aviação, da cinematografia e da ciência do trabalho fotográfico sendo considerado um dos pioneiros da fotografia e da história do cinema.

patente do filme em rolo com substrato de papel e cassete. Em 1888 Eastman recebe a patente para uma câmara portátil que utilizava o rolo de filme previamente mencionado, a 14 de outubro de 1888 o filme experimental Roundhay Garden Scene de Louis Le Prince[24] é filmado, tornando-se o mais antigo filme ainda existente e a 1889 a produção em massa do rolo de filme de Eastman começa, assim como a alteração do nome da empresa para Eastman Company (Eastman Kodak Company, s.d.).

A 1892 a empresa de Eastman volta a ter uma alteração, passando agora a ser conhecida como a Eastman Kodak Company (Eastman Kodak Company, s.d.), e Gabriel Lippmann[25] introduziu um processo para criar fotografias com cores naturais, baseando-se no princípio do fenómeno ótico da interferência das ondas de luz (Hannavy, 2007, p. 862), William Kennedy Dickson[26] a trabalhar sobre a direção de Thomas Edison[27], criou o *Kinethograph*, uma câmara que tirava fotográficas instantâneas em série, e a 1893 os resultados desta invenção foram mostrados ao público através de outro instrumento inventado por Dickson, o Cinetoscópio[28]. Dois anos depois, em 1895, Charles Francis Jeankins[29] e o seu projetor Fantascópio[30] tiveram uma audiência com sucesso enquanto os irmãos Lumière[31] aperfeiçoaram o cinematógrafo[32], sendo os primeiros a mostrar imagens fotográficas projetadas em movimento para uma audiência superior a uma pessoa, que pagasse. Em 1896 os primeiros cinemas foram abertos na França, Itália, Bruxelas e Londres, e Edinson mostrou o seu vitascópio melhorado, sendo este o primeiro projetor com sucesso comercial nos Estados Unidos da América (Cook, 1996, p. 1–13).

Inaugurando o século em 1902, Edward Raymond Turner[33] produziu os primeiros filmes com um processo de cor natural em vez de utilizar técnicas de colorização e o filme "A trip

---

[24] Louis Aimé Augustin Le Prince (Metz, França, 28 de agosto de 1842, desaparecido a 16 de setembro de 1890) foi um dos precursores do cinema.

[25] Jonas Ferdinand Gabriel Lippmann (16 de agosto de 1845 – 13 de julho de 1921) foi um físico franco-luxemburguês e invento, ganhou um prémio Nobel em física devido ao seu método de reproduzir cores fotograficamente baseadas no fenómeno físico da interferência.

[26] William Kennedy-Laurie Dickson (Le Minihic-sur-Rance, 3 de agosto de 1860 — Twickenham, 28 de setembro de 1935) foi um inventor escocês que criou uma câmara cinematográfica pioneira enquanto trabalhava para Thomas Edison.

[27] Thomas Alva Edison (Milan, Ohio, 11 de fevereiro de 1847 — West Orange, Nova Jérsia, 18 de outubro de 1931) foi um empresário dos Estados Unidos que patenteou e financiou o desenvolvimento de muitos dispositivos importantes de grande interesse industrial.

[28] O Cinetoscópio é um instrumento de projeção interna de filmes, inventado por William Dickson, em 1891, possuía um visor individual através do qual se podia assistir, mediante a inserção de uma moeda, à exibição de uma pequena tira de filme em ciclo, onde apareciam imagens em movimento de números cómicos, animais amestrados e bailarinas.

[29] Charles Francis Jenkins (22 de agosto de 1867 – 6 de junho de 1934) foi um americano pioneiro da primeira era do cinema e um dos inventores da televisão, embora ele utilizasse tecnologias mecânicas em vez de eletrónicas.

[30] O Fantascópio era uma máquina de projeção de filme, uma criação de Charles Francis Jenkins e Thomas J. Armat. (25 de outubro de 1866 – 30 de setembro de 1948) (inventor e mecânico americano e pioneiro do cinema, mais conhecido como o coinventor do Vitascópio (projetor de cinema) de Edison)

[31] Auguste Marie Louis Nicholas Lumière (Besançon, 19 de outubro de 1862 — Lyon, 10 de abril de 1954) e Louis Jean Lumière (Besançon, 5 de outubro de 1864 — Bandol, 6 de junho de 1948) foram os inventores do cinematógrafo, e frequentemente referidos como os pais do cinema.

[32] O cinematógrafo é geralmente considerado como um aperfeiçoamento do cinetoscópio de Thomas Edison.

[33] Edward Raymond Turner (1873 – 9 de março de 1903) foi um inventor e cineasta britânico pioneiro e produziu o primeiro filme de cinema colorido conhecido.

to the Moon", produzido, escrito e direcionado por Georges Méliès[34], inspirado no romance Da Terra à Lua (1865), de Júlio Verne[35] e a sequela, À Volta da Lua (1870), um filme que entre várias técnicas de cinema que surgiam, teve uma extensa manipulação e edição no que toca a *post-Processing*, especialmente em algumas das suas versões a cores, onde o filme fotográfico foi pintada a cores. Em 1906, foi produzido o primeiro desenho humorístico animado e no ano seguinte de 1907, os irmãos Lumière apresentaram o primeiro processo de cor com sucesso comercial, o *Autochrome* Lumière, patenteado previamente no ano de 1903, permanecendo como o processo principal de obter fotografias a cores durante este início do século XX. A 1908 surgiu a *Kinemacolor* e o curto filme A visit to the Seaside (George Albert Smith) tornou-se o primeiro filme com cores naturais a ser publicamente apresentado (ACMI, s.d.).

A 1911 os créditos começaram a aparecer no fim de filmes (Cook, 1996, p. 40), a 1915 a câmara de filmar Bell and Howell 2709 permitiu que fossem realizadas aproximações sem ter de se mover a câmara fisicamente e a 1917 a primeira versão da Technicolor foi introduzida (The New York Times, 1917, p. 9).

Durante a década de 1920 não houve melhoramentos notáveis em nenhum dos pontos, quer fotográficos, cinematográficos ou *post-Processing*, mas foi, no entanto, no ano de 1927 que saiu o filme Metropolis, produzido por Erich Pommer[36] e direcionado por Fritz Lang[37], um filme alemão de ficção científica que contou com a participação de Eugen Schüfftan[38], que criou efeitos visuais pioneiros para a altura (Loew, 2021).

---

[34] Georges Méliès, cujo nome de batismo é Marie Georges Jean Méliès (Paris, 8 de dezembro de 1861 — Paris, 21 de janeiro de 1938) foi um ilusionista e cineasta francês famoso por liderar muitos desenvolvimentos técnicos e narrativos no alvorecer do cinema. Foi um inovador prolífico no uso de efeitos especiais, popularizou técnicas como o *stop-motion* e foi um dos primeiros cineastas a usar exposições múltiplas, a câmara rápida, as dissoluções de imagem e o filme colorido. Ele também foi um pioneiro no uso de *storyboards*. Graças à sua capacidade de manipular e transformar a realidade através da cinematografia, Méliès é lembrado como um "mágico de cinema".

[35] Jules Gabriel Verne (8 de fevereiro de 1828 – 24 de março de 1905) foi um romancista, poeta e dramaturgo francês cuja colaboração com o editor Pierre-Jules Hetzel levou à criação de Voyages extraordinaires, uma série de romances de aventura *best-sellers*, incluindo Viagem ao Centro da Terra (1864), Vinte Mil Léguas Submarinas (1870) e A Volta ao Mundo em Oitenta Dias (1872).

[36] Erich Pommer (20 de julho de 1889 – 8 de maio de 1966) foi um produtor e executivo de cinema alemão e talvez a pessoa mais poderosa nas Indústrias Cinematográficas alemãs e europeias na década de 1920 e no início da década de 1930.

[37] Friedrich Christian Anton "Fritz" Lang (5 de dezembro de 1890 – 2 de agosto de 1976) foi um cineasta austríaco-alemão-americano, guionista e ocasionalmente produtor e ator de cinema, apelidado de "Mestre das Trevas" pelo British Film Institute.

[38] Eugen Schüfftan (21 de julho de 1893, em Breslau, Silésia, Alemanha, agora Wroclaw, Polônia — 6 de setembro de 1977, na cidade de Nova York) foi um cineasta alemão, inventor do processo Schüfftan, uma técnica de efeitos especiais que empregava espelhos para inserir atores em cenários miniatura e vencedor do Óscar de Melhor Cinematografia a Preto e Branco em 1962 por seu trabalho no filme The Hustler.

Em 1935 a Kodak introduziu o Kodachrome, em 1936 foi introduzida a Agfacolor Neu pela Agfa[39]. O Feiticeiro de Oz é lançado em 1939, com a direção de Victor Fleming[40] baseado no livro O Maravilhoso Feiticeiro de Oz, de L. Frank Baum[41], também este um filme notável na evolução do cinema e do *post-Processing*, especialmente devido a sua utilização de efeitos práticos e de Technicolor *(*Vox*, 2017)*.

É na década de 1940 que os videojogos começam a dar os seus primeiros passos, começando com o Nimatron, produzido pela Westinghouse Electric era uma máquina eletromecânica inventada por Edward Edward Condon[42], diretor associativo de pesquisa na Westinghouse Electric Conpanyem Pittsburgh. A Nimatron servia para jogar o antigo jogo matemático de estratégia, Nim, e a invenção foi revelada durante a New York World's Fair em 1940 e é até agora a primeira máquina de jogos (Redheffer, 1948, p. 343). Em 1947, Thomas T. Goldsmith jr.[43] e Estle Ray Mann registam uma patente para um "dispositivo de diversão com tubo de raios catódicos (CRT)". O jogo deles, que usava um tubo de raios catódicos ligado a um visor de osciloscópio, desafiava os jogadores a disparar contra um alvo, marcando-se na historia como o primeiro verdadeiro videojogo, visto que o Nimatron era corretamente enquadrado como um jogo eletrónico e não um videojogo (“‘The Cathode Ray Tube Amusement Device,’ Probably the Oldest Interactive Electronic Game : History of Information,” s.d.).

Em 1950 foi introduzida a Eastmancolor, que se tornou o *standard* de cor para o resto do século na fotografia e cinematografia (Lev, 2006, p. 108), durante esta década de 1950 vários outros dispositivos e máquinas de jogos foram criados, na sua grande maioria com o intuito de demonstrar como a tecnologia funcionava, tais como, um artigo sobre os guias básicos de como programar um computador para jogar xadrez publicado por Claude Shannon[44]e o OXO *(*A.S.Douglas*, 1952)*, criado por Sandy Douglas[45] num computador EDSAC

---

[39] Agfa-Gevaert N.V. é uma empresa multinacional belga que desenvolve, produz e distribui produtos e sistemas digitais, assim como analógicos, na área de processamento e reprodução de imagens, mas foi durante muitas décadas uma das maiores fabricantes europeias de filmes fotográficos.

[40] Victor Lonzo Fleming (23 de fevereiro de 1889 – 6 de janeiro de 1949) foi um diretor de cinema, cinegrafista e produtor americano.

[41] Lyman Frank Baum, (Chittenango, 15 de maio de 1856 — Hollywood, 6 de maio de 1919), foi um escritor, editor, ator, guionista, produtor de cinema e teosofista norte-americano, criador de um dos mais populares livros escritos na literatura americana infantil, O Mágico de Oz. Em 1897, tornou-se membro da Sociedade Teosófica, incorporando frequentemente nos seus livros temas e símbolos desta doutrina.

[42] Edward Uhler Condon (Alamogordo, Novo México, 2 de março de 1902 — Boulder, Colorado, 26 de março de 1974) foi um físico nuclear estadunidense, pioneiro na mecânica quântica e um participante no desenvolvimento do radar e das armas nucleares durante a Segunda Guerra Mundial participando ainda no Projeto Manhattan.

[43] Thomas Toliver Goldsmith Jr. (9 de janeiro de 1910 – 5 de março de 2009) foi um pioneiro da televisão americana, o inventor conjunto do primeiro jogo de arcade a usar um tubo de raios catódicos e professor de física na Universidade Furman.

[44] Claude Elwood Shannon (30 de abril de 1916 – 24 de fevereiro de 2001) foi um matemático, engenheiro eletrotécnico e criptógrafo americano conhecido como "o pai da teoria da informação é também conhecido por fundar a teoria do projeto de circuito digital em 1937 e contribuiu para o campo da criptoanálise para defesa nacional durante a Segunda Guerra Mundial, incluindo o seu trabalho fundamental sobre quebra de código e telecomunicações seguras.

[45] Alexander Shafto "Sandy" Douglas (21 de maio de 1921 – 29 de abril de 2010) foi um professor britânico de ciência da computação, responsável pela criação do jogo de computador OXO em 1952.

de Cambridge, para jogar o jogo do galo, mas denota-se em 1958, Tennis for Two (William Higinbotham, 1958), criado por William Higinbotham[46], com o intuito adicional de proporcionar diversão e não só uma demonstração tecnológica (Donovan, 2010).

Na década de 1960 vê-se o surgir do primeiro videojogo de computador, Spacewar! (Steve Russell, 1962), desenvolvido por Steve Russel[47] no Instituto de Tecnologia de Massachusetts (MIT) em 1962 (Donovan, 2010). Em 1963 é introduzido filme fotográfico instantâneo a cores, através do uso de uma câmara especial, pela Polaroid[48]. Em 1967, Ralph Baer[49] cria um protótipo denominado Brown Box que, quando ligado a uma televisão, permitia os utilizadores jogarem vários jogos, dos quais o principal sendo ténis (Donovan, 2010).

É, no entanto, na década de 1970 que a indústria dos videojogos começa a ter o seu verdadeiro início, isto devido à Magnavox Odyssey, uma consola de videojogos baseada na brown box serviu de inspiração para que a Atari, fundada em 1972 lança-se a sua primeira consola cujo nome é igual ao da empresa, com o jogo Pong (Atari, 1972), isto tudo no mesmo ano em que a empresa teve a sua fundação, 1972 (Donovan, 2010). Ainda nesse ano a Apple é fundada. Em 1973 o jogo Maze(Steve Colley e Greg Thompson, 1973) introduz os jogadores ao primeiro jogo em primeira pessoa de tiros (conhecido também como *FPS* ou *First-Person Shooter*) (Donovan, 2010), em 1976, a versão do jogo Colossal Cave Adventure (William Crowther e Don Woods, 1976) criada por Don Woods[50] inspirado em Dungeons & Dragons (TSR, 1974), abre assim o caminho para futuros jogos de *role-play* (também conhecidos como *RPG* ou *role-playing games*), em 1977 sai a *Atari 2600*, uma consola caseira capaz de trocar jogos, jogos a cores, um *joystick*, e interruptores que permitiam selecionar jogos e a dificuldade neles, por fim, em 1978 o jogo Space Invaders (Taito, 1978), criado por Tomohiro Nishikado[51], inaugura o que é conhecido como a Era de Ouro das *Arcades* (Donovan, 2010).

Entrando na nova década, em 1980 o conhecido jogo Pac-Man (Namco, 1980), originalmente conhecido como Puck Man é lançado em formato de *arcade*, e nesse ano o jogo é

[46] William Alfred Higinbotham (22 de outubro de 1910 – 10 de novembro de 1994) foi um físico americano, membro da equipa que desenvolveu a primeira bomba nuclear, e um dos líderes do movimento contra a proliferação nuclear. Ocupa também um lugar na história dos videojogos com a sua criação de 1958 de Tennis for Two, um dos primeiros jogos de computador analógico interativos e um dos primeiros jogos eletrónicos a usar um mostrador gráfico.

[47] Stephen Russell (nascido em 1937), também apelidado de "Slug", é um cientista da computação americano, famoso por criar “Spacewar!”.

[48] A Polaroid era uma empresa americana conhecida pelos seus filmes fotográficos instantâneos e câmaras fotográficas. A empresa foi fundada em 1937 por Edwin H. Land, que dirigiu a empresa até 1981.

[49] Ralph Henry Baer (nascido Rudolf Heinrich Baer; 8 de março de 1922 – 6 de dezembro de 2014) foi um inventor, desenvolvedor de jogos e engenheiro alemão-americano.

[50] Don Woods (nascido em 30 de abril de 1954) é um *hacker* e programador de computador americano, conhecido pelo seu papel no desenvolvimento do jogo Colossal Cave Adventure.

[51] Tomohiro Nishikado (nascido em 31 de março de 1944 em Osaka) é um desenvolvedor de videojogos, japonês, conhecido como o criador de Space Invaders.

lançado também para a *atari 2600*, no ano seguinte, Donkey Kong (Nintendo, 1981) é lançado e no ano seguinte Ms. Pac-Man (Midwaym 1982) torna-se o jogo de *arcade* mais vendido de sempre, e a Disney lança o filme Tron (Steven Lisberger, 1982), um filme de ação e ficção-científica escrito e direcionado por Steven Lisberger[52], Tron ficou marcado na história do cinema devido a ter sido um dos primeiros filmes a usar animação de computador extensiva. A 1984 o jogo *Tetris* (Alexey Pajitnov e Vladimir Pokhilko, 1984) é criado por Alexey Pajitnov[53] e a 1985 a *Nintendo* lança nos EUA a Nintendo Entertainment System, mais conhecida como NES, previamente lançada no Japão, em 1987 o jogo Legend of Zelda (Nintendo, 1986) é disponibilizado internacionalmente, criado por Shigeru Miyamoto[54]. Nota-se ainda que a fotografia digital, que tem vindo a surgir desde 1950s, começa a ter os seus primeiros passos evolutivos notáveis. Além disso, é em 1988 que o programa ImagePro (Thomas Knoll e John Knoll, 1987) é lançado pelos irmãos Thomas[55] e John[56] Knoll, um programa que servia para aplicar efeitos em imagens num computador, sendo estas ainda a preto e branco, nesse mesmo ano, o *software* muda de nome para Barneyscan XP e de seguida para o nome que mantém até agora, após um negócio de sucesso com a Adobe, assim surgiu o Adobe Photoshop, que constatou o seu primeiro lançamento ao público em 1990 (Pagin, s.d.), um ano depois do lançamento do *Game Boy*.

1.3 O início e evolução do post-Processing nos videojogos até hoje.

Na década de 1990 começa-se novamente a observar a indústria do cinema, da fotografia e do *post-Processing*, a sofrerem algumas alterações consideráveis devido à evolução tecnológica, assim como a indústria dos videojogos, que prossegue com um bom ritmo evolutivo. Sendo assim, como dito previamente, é no ano de 1990 que o Adobe *Photoshop 1.0* é lançado exclusivamente para computadores Macintosh, o que inicia um marco no que toca à edição, Pós-Produção e *post-Processing* fotográfico atualmente (outro marco seria em 2003, quando a Adobe lançou a *Creative Suite* e como tal o *Adobe Photoshop CS (Photoshop 8.0)*, sendo esta a versão que catapultou o *software* comercialmente.) (Lindblad, 2020)

Em 1993, Dark Edge (Sega, 1993), lançado para o sistema Sega System 32, guarda um lugar na história como sendo o primeiro videojogo a utilizar reflexões como um espelho, em

---

[52] Steven M. Lisberger (nascido a 24 de abril de 1951) é um diretor de cinema, produtor e escritor americano famoso por dirigir Tron em 1982.

[53] Alexey Leonidovich Pajitnov (Moscovo, 14 de março de 1956) é um engenheiro de computação, nascido na União Soviética, hoje residente nos Estados Unidos. Tornou-se conhecido mundialmente por criar o jogo Tetris quando trabalhava para o Centro de Computação da Academia Soviética de Ciências, nos anos 1980.

[54] Shigeru Miyamoto (Sonobe, 16 de novembro de 1952) é um *designer* e produtor de jogos eletrónicos japonês, conhecido por ser o criador de algumas das mais bem-sucedidas séries de jogos eletrónicos de todos os tempos.

[55] Thomas Knoll é um engenheiro de *software* estadunidense nascido em Ann Arbor, Michigan, mais conhecido por iniciado o desenvolvimento do Photoshop (com o seu irmão John Knoll).

[56] John Knoll (Ann Arbor, 6 de outubro de 1962) é um supervisor de efeitos visuais americano e diretor criativo da Industrial Light & Magic. É um dos criadores originais do Adobe Photoshop (com o seu irmão, Thomas Knoll), também trabalhou como supervisor de efeitos visuais nas prequelas de Star Wars e nas edições especiais da trilogia original, também atuou como supervisor de efeitos visuais da ILM para Star Trek Generations e Star Trek: First Contact, assim como a série Piratas das Caraíbas.

tempo real, algo que, até atualmente, seja através de *Screen-Space Reflections*, *Ray Tracing*[57] ou outros truques, ainda é um sistema difícil de criar. Nesse ano, saiu o jogo conhecido atualmente como um dos melhores videojogos de todo o sempre, assim como um dos videojogos mais significativos na história dos videojogos, *DOOM* (id Software, 1993), ajudando a definir o género de FPS e sendo pioneiro em distribuição *online*, tecnologias como o uso de gráficos 3D, modo de jogo *multiplayer* através do uso da *internet*, e suporte para *mods*[58]. Imediatamente no ano seguinte, a sequela DOOM II: Hell on Earth (id Software, 1994) é lançado, assim como a Playstation, criada pela Sony, uma consola com grande importância na evolução tanto das consolas de videojogos como os videojogos, por ser uma das consolas que impulsionou a transição de cartuchos para CDs, assim como a transição para sistemas de *32-bits* (PlayStation | Console and Games, 2008), (ainda em uso por alguns sistemas atualmente, embora esteja a atingir o seu fim de vida, dado em grande parte devido às limitações de memória *RAM*, até quatro *gigabytes*) para além disto, ao longo da sua vida, a Playstation permitiu que inúmeros videojogos fossem criados, e alguns criaram impactos profundos nos videojogos que constatamos atualmente, como, por exemplo, Wipeout (Psygnosis, 1995), Final Fantasy VII (Square Enix, 1997), Gran Turismo (Polys Entertainment, 1997) assim como Metal Gear Solid (Konami Computer Entertainment Japan, 1998) e o seu impacto nos videojogos cinemáticos e narrativos dos dias de hoje.

Em 1996 é lançado o filme Rainbow (Bob Hoskins, 1996), um filme familiar de aventurar sobre a direção de Bob Hoskins[59], ficando este filme conhecido como o primeiro filme no mundo a utilizar técnicas de Pós-Produção digital extensivas, filmado inteiramente nas primeiras câmaras de Cinematografia Eletrónica de Estado Sólido da Sony, toda a pós-produção, efeitos de som, edição e música foram digitais. Nesse ano sai o Quake (id Software, 1996), que se torna pioneiro devido a ser primeiro jogo FPS a utilizar ambientes completamente modelados em 3D e mapas de luz pré-sintetizados (Abrash, 2000). Tomb Raider (Core Design, 1996) é também lançado nesse ano, por parte da Eidos Interactive, e impulsiona o uso de caracteres femininos invés de só masculinos, no que tocava ao papel de personagem principal de ação.

---

[57] Ray Tracing é um algoritmo de computação gráfica utilizado para a síntese (renderização ou sintetização) de imagens tridimensionais, ou CGIs, o método consiste na simulação do trajeto que os raiz de luz percorrem no mundo real, mas, neste caso de trás para a frente, ou seja, o inverso do mundo real onde os raios de luz são emitidos a partir de uma fonte de luz percorrem o espaço ate encontrarem um objeto, e apos encontrarem o objeto são refratados ou refletidos de acordo com as propriedades do objeto como a cor, textura e a opacidade, alterando assim a trajetória e fazendo com que apenas uma infinitésima minoria dos raios que partiram da fonte de luz atinjam, finalmente o olho ou ponto observador.

[58] Um mod ou modificação, é uma alteração realizada por fans ou jogadores a um ou mais aspetos de um videojogo, desde o aspeto deste, a mais níveis ou até pequenas alterações que resolvem pequenos problemas ou falhas existentes no mesmo.

[59] Robert William "Bob" Hoskins, Jr. (Bury St Edmunds, 26 de outubro de 1942 — Londres, 29 de abril de 2014) foi um ator britânico. A sua carreira incluiu papéis principais em Pennies from Heaven (1978), The Long Good Friday (1980), Mina Lisa (1986), Who Framed Roger Rabbit (1988), Mermaids (1990) e Super Mario Bros. (1993), e de apoio em Brazil (1985), Hook (1991), Nixon (1995), Enemy at the Gates (2001), Mrs. Henderson Presents (2005), A Christmas Carol (2009), Made in Dagenham (2010) e Branca de Neve e o Caçador (2012). Alem disso, ele também foi o diretor dos filmes The Raggedy Rawney (1988) e Rainbow (1996).

Em 1997 No Respect (Appeal, 1997), publicado pela Ocean Software, foi um dos primeiros videojogos a utilizar *lens flare*, em 1998 Metal Gear Solid (Konami Computer Entertainment Japan, 1998) é lançado e cria o início do que viriam a ser jogos cinemáticos assim como tendo um forte impacto na narrativa dos videojogos, e, no ano seguinte, em 1999, George Lucas[60] produz o filme Star Wars: Episode I — The Phantom Menace (George Lucas, 1999), onde ele misturou sequencias filmadas com câmaras digitais de alta-definição com sequências filmadas em película fotográfica, o que desafiou bastante a supremacia que existia na altura, no que tocava aos modelos de câmara a usar, especialmente quando decidiu filmar a sequela, Star Wars: Episode II — Attack of The Clones (George Lucas, 2002), inteiramente com câmaras digitais. Foi também neste ano que o videojogo de ação e aventura Outcast (Appeal, 1999) foi lançado para Windows pela infogrames, e com ele uma superabundância de efeitos e novas tecnologias, algumas de post-Processing, tais como, *Bloom*, (existente previamente em videojogos como um efeito que ocorria naturalmente devido aos ecrãs CRT, mas que com a evolução para ecrãs com painéis eletrónicos, deixou de acontecer) profundidade de campo e *Anti-Aliasing*.

Com o início do novo milénio, imensas alterações ocorreram na indústria dos videojogos, no desenvolvimento dos mesmos, e por si até alterações culturais e sociais da sociedade. A *internet* evolui de algo pequeno e pouco utilizado pela população em geral, para quase que um fator no dia a dia da sociedade, o lançamento da Playstation 2 por parte da Sony, a introdução de jogos *sandbox*[61] e o surgimento ou popularização de várias franquias como Counter Strike (Valve, 2000), The Sims (Maxis, 2000) e Grand Theft Auto (Rockstar Games, 1997) mais precisamente com os lançamentos de Grand Theft Auto: Vice City (Rockstar North, 2002) e Grand Theft Auto: San Andreas (Rockstar North, 2004). *Bloom*, o efeito de *post-Processing* teve também um grande impulso na sua utilização com o lançamento do jogo Tron 2.0 (Monolith Productions, 2003) pela Buena Vista Interactive e o artigo publicado pelos autores do videojogo em 2004, *Real-Time Glow,* o que levou a uma saturação no uso do mesmo. Crysis (Crytek, 2007), é lançado pela Eletronic Arts, e cria não só um *benchmark*[62] nos anos que seguiram mediante o poder de processamento necessário, assim como em realismo e aperfeiçoamento de imensos efeitos e técnicas nos videojogos. Outro

[60] George Walton Lucas Jr. (nascido a14 de maio de 1944) é um diretor de cinema, produtor, guionista e empresário americano. Lucas é mais conhecido por criar as franquias de Star Wars e Indiana Jones e fundar a Lucasfilm, LucasArts e Industrial Light & Magic. Ele atuou como presidente da Lucasfilm antes de a vender à The Walt Disney Company em 2012. É um dos cineastas de maior sucesso financeiro da história e foi indicado a quatro Oscars e é considerado uma figura significativa do movimento New Hollywood do século XX.

[61] Um jogo sandbox é um tipo de videojogo com um elemento de jogabilidade que permite ao jogador um alto grau de criatividade para completar tarefas em direção ao objetivo dentro do jogo, caso o objetivo exista. Alguns jogos existem como jogos de sandbox puros, sem objetivos, também conhecidos como brinquedos de software ou não-jogos.

[62] Em computação, benchmark é o ato de executar um programa de computador, um conjunto de programas ou outras operações, a fim de avaliar o desempenho relativo de um objeto, normalmente executando uma série de testes padrão e ensaios nele. Normalmente, benchmarking é associado com a avaliação de características de performance de um hardware de computador.

ponto marcante na década de 2000 foi também a introdução e popularização de *smartphones*, e videojogos sociais.

Além disto, a evolução na qualidade e realismo gráfico nos videojogos tem vindo a tornar-se cada vez mais realista, até ao ponto em que, atualmente, há videojogos que se apresentam como realistas ou muito perto de tal, e onde o foco tecnológico se vira para a criação de realidades virtuais e aumentadas, o uso de *Deep Learning*[63] e inteligência artificial, e inclusive o *streaming* de videojogos, que faz com que seja possível jogar videojogos com requisitos de hardware elevados, em dispositivos que nunca conseguiriam preencher esses requisitos, desde que o utilizador possua uma conexão estável à *internet*. Por fim, marca-se também a introdução ou transição que certos videojogos começaram a sofrer desde 2018, com o lançamento de placas gráficas capazes de *Ray Tracing* em tempo real, o que levou a que vários videojogos começassem a utilizar esta mesma tecnologia de sintetização[64], em vez do *standard* que era a rasterização, conhecida também como *Raster*. No capítulo **2.2 *Ray Tracing* em tempo real, *Ray Tracing* tradicional *e*** Lumen explica-se sucintamente o que é o *Ray Tracing*.

[63] *Deep Learning* é uma função da inteligência artificial que imita o funcionamento do cérebro humano no processamento de dados e na criação de padrões para uso na tomada de decisões.

[64] Sintetização neste caso refere-se à síntese da imagem, normalmente referido pela palavra brasileira, renderização, proveniente da palavra inglesa render + izar + -ção. Sintetização, vem como tal de sintetizar + -ção.

2 O post-Processing no desenvolvimento de videojogos.

Tendo, portanto, acompanhado a história e evolução do *post-Processing* nos seus vários meios, entende-se que o *post-Processing* é na sua essência, um conjunto de efeitos ou alterações aplicadas a algo, após a conclusão diga-se física do trabalho, ou seja, a aplicação de efeitos ou alterações a uma fotografia, filme, pintura, etc. E de certo modo isto aplica-se tanto a CGI, como aos videojogos. No entanto, no que toca a estes dois últimos, e por agora consideraremos que os videojogos fazem parte de CGI, assim como animações 3D e qualquer outra categoria de arte que utilize o poder de processamento de um sistema informática para criar e apresentar imagens, o *post-Processing* funciona de maneiras algo diferentes do habitual, especialmente porque atualmente muitos dos efeitos ou filtros de *post-Processing* podem ser ligados com um simples clique de rato, o que leva aos utilizadores a por vezes tomarem como garantido um efeito sem nunca pensarem como este é realizado, dando graças apenas aos desenvolvedores e programadores que o implementaram e como tal tratando a maneira como estes efeitos funcionam quase que "por magia". Com isto o que se quer dizer é, que enquanto muitos destes efeitos podem ser aplicados a uma CGI sem grandes estratagemas, há alguns onde o processo para tal necessita de um pouco mais de pensamento, e como tal, de implementação. Tome-se como exemplo o *Anti-Aliasing*, que abordaremos mais aprofundadamente a seguir, este possui várias formas de ser implementado, mas observe-se a forma mais simples, que consiste em sintetizar uma imagem com dezasseis vezes a sua resolução, ou seja, uma imagem que inicialmente teria a resolução de 1000×1000 pixeis (um megapíxel / 1MP), terá de ser sintetizada a 16000×16000 pixeis (duzentos e cinquenta e seis megapíxeis / 256MP) e depois reduzida novamente para a resolução inicial de 1MP, sendo, portanto, este filtro enquadrado em *post-Processing* de uma forma um pouco estranha, visto envolver passos iniciais necessários ainda antes de entrarmos na fase de *post-Processing*, visto que apenas a diminuição da imagem para a resolução original é que é feita no *post-Processing*.

De qualquer maneira e falando de um modo mais simples e não tão avançado, a verdade é que o *post-Processing* é, portanto, algo aplicado a uma obra concluída (independentemente de que esta não esteja na sua versão final de apresentação) nos termos em que não é necessário ou viável refazer a obra para corrigir ou alterar certos aspetos da mesma. O que não significa que seja incorreto ou desnecessário realizar a obra sem pensar no que se possa fazer no *post-Processing*, assim como não se deve realizar a mesma com o pensamento de que qualquer problema se possa resolver em *post-Processing*, o que se vê várias vezes atualmente, normalmente em conjunto com frases como "Resolve-se depois em *Post*".

2.1 Efeitos existentes e as suas funções.

Atualmente, dentro da Unreal Engine 4.26 (Epic Games, 2021) (**Figura 16**), existem cinco categorias de efeitos de *post-Processing*, as quais podem encontrar-se a seguir. Não quer, no entanto, dizer que esta os possua todos, pois há muitos outros motores, tanto de

criação de videojogos como esta, assim como de sintetização comum. Assim como, há outras formas de *post-Processing* que podem estar incluídas na Unreal Engine, mas, a maneira como estas funcionam, faz com que não possam ser parte do *post-Processing*, devido a ocorrerem, por exemplo, durante a sintetização da imagem, ou inclusive antes desse ponto.

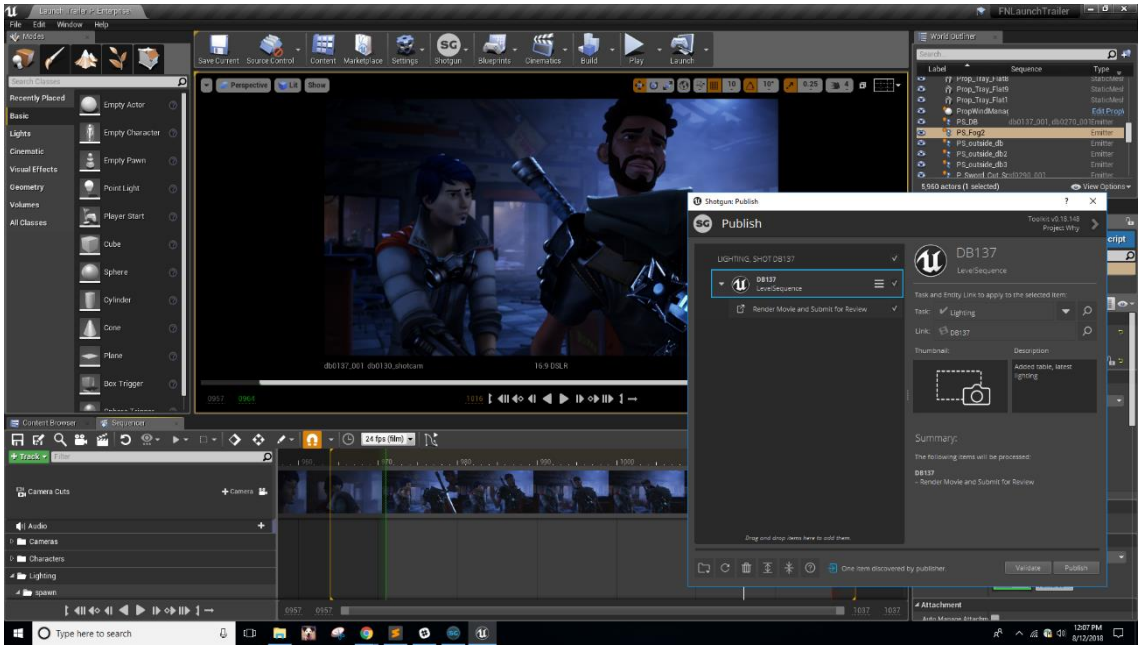

Figura 16 - Captura de tela da Unreal Engine 4.20 (Epic Games, 2018).

Isto não quer dizer, no entanto, que em versões passadas da *engine* (motor), ou versões futuras, que a quantidade de efeitos não possa mudar, devido principalmente à evolução tecnológicas e as mudanças constantes que esta traz.

Denote-se ainda que, como referido, estes efeitos não são exclusivos da Unreal Engine, e grande parte, pode-se encontrar noutros *softwares*, como, por exemplo, o *Color-Grading*, encontrado em praticamente todas as categorias de *software* cujo intuito seja a edição fotográfica, edição de vídeo ou sintetização de cenas 3D. Assim sendo, é possível que a maneira como estes funcionam, seja semelhante ou igual a outros *softwares*, validando, portanto, que a transição de um *software* para outro, no que toca a *post-Processing*, não seja obrigatoriamente simbólico de começar do zero.

Antes de mais, é necessário saber que, para utilizar efeitos de *post-Processing* dentro da Unreal Engine, o desenvolvedor, utilizador ou artista tem de adicionar à sua cena um *Post Process Volume* (**Figura 17**), pois é este volume que controla todas as configurações que se deseja, mais do que isso, podem-se ter vários volumes, o que permite ter várias configurações numa só cena, rapidamente, sem ter de andar a reconfigurar tudo novamente.

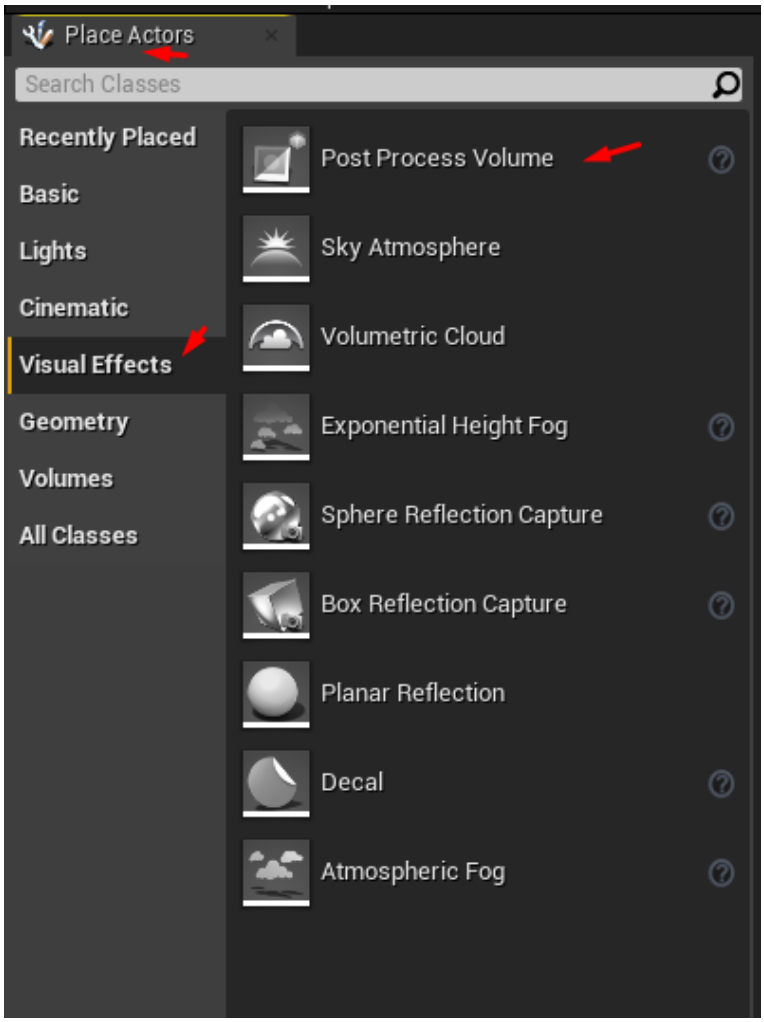

Figura 17 - Guia visual de como adicionar um *Post Process Volume* a uma cena na Unreal Engine 4.26, começando por ir ao separador *Place Actors*, seguindo de clicar em *Visual Effects* (Imagem do autor).

Mais ainda, denote-se que há mais dois elementos nesta lista que serão importantes mais à frente, nomeadamente o *Sphere Reflection Capture* e o *Box Reflection Capture*.

Finalmente dando uma vista de olhos ao que o *Post Process Volume* (**Figura 18**) apresenta, este contem várias categorias que podem ser abertas e fechadas, nomeadamente o *Transform*, que serve para controlar o tamanho rotação e local do volume, *Lens*, que apresenta uma quantidade variedade de efeitos de post-Processing relacionados à lente de uma câmara, seguido de *Color Grading* e *Film*, que mais à frente se explorará melhor o que contêm e para que servem, *Rendering Features*, que trata de elementos relacionados com a sintetização da cena, *Post Process Volume Settings*, onde se pode configurar propriedades do volume, assim como a maneira como este interage com outros volumes de post-Processing, e, se este é infinito no caso do I*nfinite Extent (Unbound,), ou seja*, caso só se utilize um único volume para a nossa cena, pode-se ligar esta propriedade, e assim deixa-se de haver preocupação com o tamanho do volume e da cena. De seguida tem-se o *Replication*, o *Brush Settings*, o *Actor*, o *Lod* e o *Cooking*, que servem vários propósitos, mas que para o caso de estudo deste documento não serão abordados.

Denota-se ainda que o foco é colocado essencialmente no desenvolvedor, e como tal, considerações no ponto de vista de um jogador não são incluídas, visto que não é incomum um desenvolvedor idealizar uma solução para o jogo, normalmente seguindo a sua visão, e o jogador alterar esta visão de modo a servir a sua, especialmente se o jogo resultante permitir a colocação de *mods*.

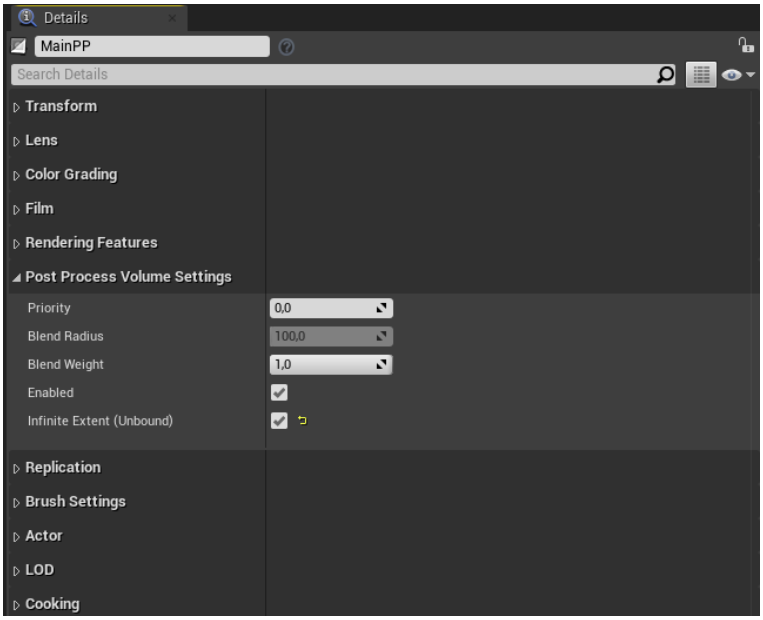

Figura 18 - Captura de tela das configurações e categorias do *Post Process Volume (Imagem do autor).*

Por último, e sendo este o primeiro filtro de que se fala, tem-se o Anti-Aliasing, que atualmente não se encontra no volume, devido a ser inerente à *engine*, como tal este pode ser encontrado nas configurações do Projeto (**Figura 19**), apenas necessário procurar por "*Aliasing"* na barra de pesquisa fornecida para que este nos seja apresentado.

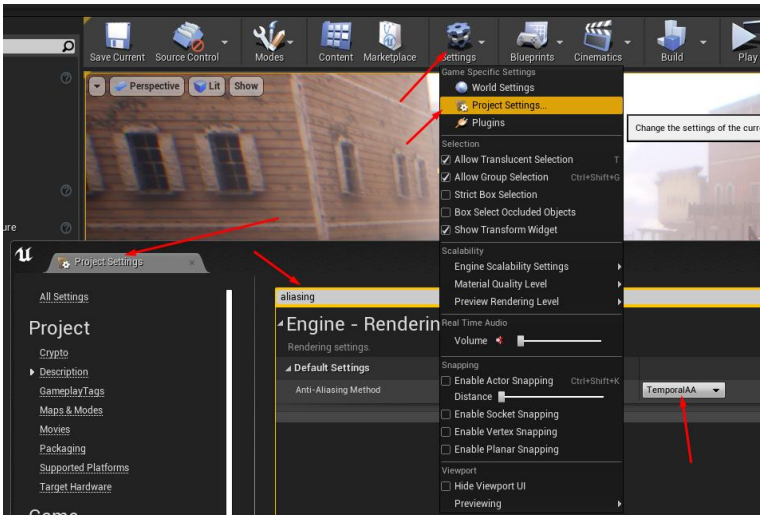

Figura 19 - Guia visual de como proceder, na Unreal Engine 4.26, para alterar o *Anti-Aliasing*, começando por clicar no botão *Settings*, seguido de *Project Settings*, e na janela que abre, procurar na barra de pesquisa por *aliasing* (Imagem do autor).

Anti-Aliasing

Anti-Aliasing (AA) é, resumidamente, um conjunto de técnicas usadas para remover as margens irregulares ou "pixelizadas" de objetos num ecrã. Ou seja, para remover o que é conhecido como *Aliasing*, e embora com a evolução tecnológica, os dias em que o *Aliasing* possa deixar de ser um problema, visto que este está, na verdade, intrinsecamente relacionada com a capacidade humana de detetar os "pixeis" num ecrã, portanto, a densidade de pixeis no ecrã, ou a resolução. Exemplo também disto é, por exemplo, um ecrã de um *smartphone*, é muito menos propicio a ter *Aliasing* notável, em comparação com o de uma televisão. Devido ao tamanho de cada píxel num ecrã de telemóvel ser imensamente menor do que é o caso da grande maioria das televisões, embora também houvesse tempos em que o *Aliasing* se notava num telemóvel, veja-se a **Figura 20**, onde a densidade de pixeis ainda não era suficientemente alta para que o *Aliasing* passasse despercebido. Denote-se ainda que o *Aliasing* não ocorre quando as linhas são completamente horizontais ou verticais. (**Figura 21**)

Figura 20 - Pormenor do ecrã de um Nokia N70 (Nokia, 2005), onde, se consegue observar o *Aliasing* dos vários objetos e textos do ecrã.

Figura 21 - Linha diagonal, vertical e horizontal, ilustrando o que acontece num ecrã devido a este ser constituído por pixeis (Imagem do autor).

A função do Anti-Aliasing?

Na sua essência, o Anti-Aliasing procura anular ou disfarçar os efeitos do *Aliasing*, que por si, é gerado devido à utilização de pixeis e à perceção humana. Daí que por vezes o *Aliasing* continua notável, inclusive após o uso de Anti-Aliasing, mas de forma suavizada.

Daí que os efeitos do *Anti-Aliasing* dependem também da forma como este é utilizado, visto que a técnica não é definida como algo preciso que só categoriza um único método. É por isto que há vários tipos e categorias de Anti-Aliasing, alguns que recorrem à sintetização da imagem em resoluções superiores, e depois à redução dessa sintetização para coincidir com a resolução do ecrã, ou à utilização de métodos de "ilusão ótica" para iludir a perceção do *Aliasing*, ou, no mínimo, suavizando esta perceção, etc.

É, no entanto, verdade que a melhor maneira de remover *Aliasing*, portanto, o que seria o melhor método de Anti-Aliasing, seria aumentar a densidade de pixeis nos ecrãs até ao ponto que a perceção fosse impossível de notar ao olho humano, mesmo que visto de muito perto. Isso, ou, uma revolução tecnológica que altere a forma como conhecemos os pixeis atualmente, ou como os ecrãs são construídos. Imagine-se, por exemplo, que, se no futuro, os ecrãs se comportassem através de vetores, é provável que o *Aliasing* não existisse. (**Figura 22**)

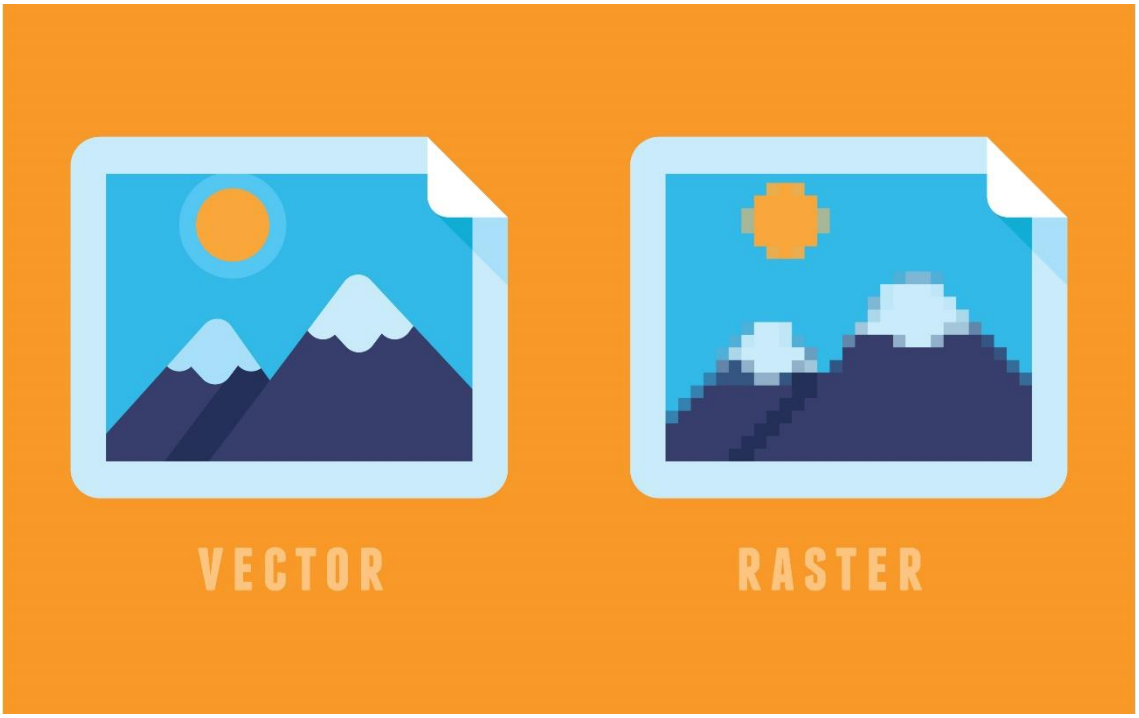

Figura 22 - Comparação entre vectorização e rasterização (s/i).

Tipos de Anti-Aliasing.

Antes de seguirmos com a análise e alguns dos tipos de Anti-Aliasing existentes até à data, assim como alguns exemplos, relembra-se que, o Anti-Aliasing está relacionado à perceção humana, e à resolução de um ecrã mediante a utilização de pixeis. Veja-se a **Figura 23** onde na parte superior temos duas capturas de ecrã do videojogo Need For Speed Heat (Ghost Games, 2019), sem utilização de *Anti-Aliasing*, e cuja parte inferior utiliza *Anti-Aliasing*, no entanto, a parte da direita (superior e inferior) são capturas a uma resolução superior, vê-se que, onde o AA está ligado, o *Aliasing* é reduzido, especialmente a grelha do carro, no entanto, nota-se também que numa resolução superior em comparação com uma inferior, o *Aliasing* é também menor.

Figura 23 - Comparação entre o uso de Anti-Aliasing e resoluções de sintetização diferentes no jogo Need For Speed Heat (Ghost Games, 2019) (Imagem do autor).

- **MSAA**

Multisample Anti-Aliasing é um dos tipos de *Anti-Aliasing* que, normalmente, estabelece o maior equilíbrio entre fidelidade visual e *desempenho*.

O que acontece com MSAA é que, usa múltiplas amostras de dois ou mais pixeis adjacentes para criar uma imagem de fidelidade superior, como se observa na **Figura 24**, quanto mais amostras forem usadas, melhor a fidelidade da imagem, no entanto, o uso de mais amostras, requer mais poder de processamento, ou seja, o impacto no desempenho aumenta. Normalmente MSAA é definido, portanto, em múltiplos de dois, ou seja, duas amostras, quatro, oito ou dezasseis.

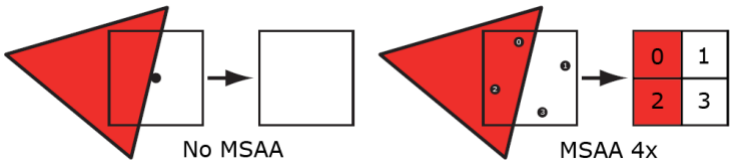

Figura 24 - Resultados de uma sintetização sem MSAA e uma com 4xMSAA, quando um triângulo cobre parcialmente um píxel. Baseado numa imagem da 3.ª edição do livro, Real-Time Rendering (Möller, Hoffman & Haines, 2008).

Na **Figura 25** vê-se a diferença entre utilizar MSAA com um múltiplo de 8, contra não utilizar nenhum tipo de Anti-Aliasing, e como tal, repare-se que a ponta arredondada da arma, é mais suave com MSAA, enquanto sem Anti-Aliasing, esta mesma ponta apresenta alguma *pixelização*.

Figura 25 - Comparação entre o uso ou não de MSAA no jogo Insurgency (New World Interactive, 2014).

- **SSAA**

Supersampling Anti-Aliasing, e é uma das técnicas de *Anti-Aliasing* mais básicas e menos eficientes que existem, basicamente, SSAA sintetiza a imagem a uma resolução superior, e, de seguida, reduz o tamanho para produzir uma imagem mais nítida, utilizando vários padrões de d*ownsampling,* como demonstrado na **Figura 26**.

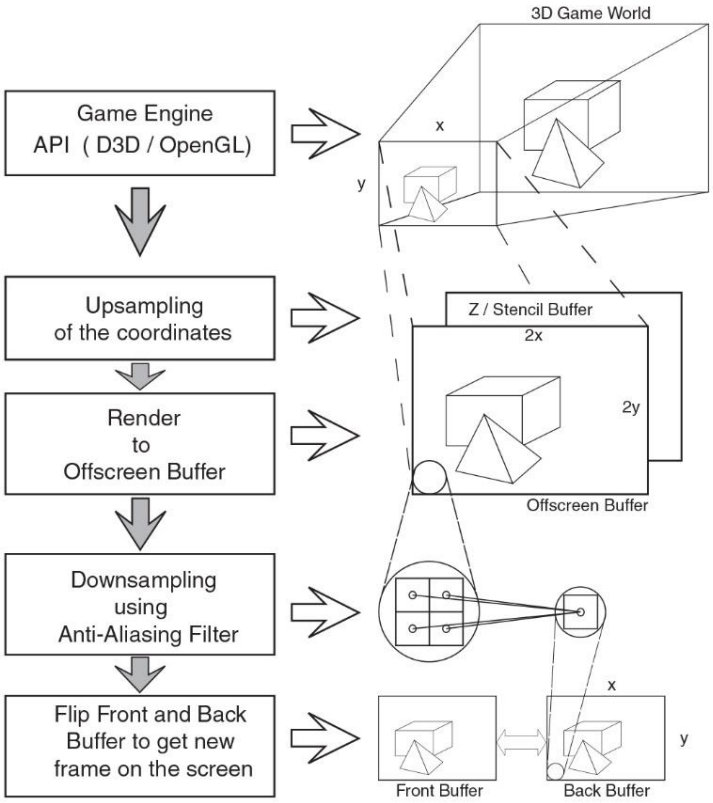

Figura 26 - Visão geral de Supersampling de grade ordenada, esquema retirado de Super-sampling Anti-aliasing Analyzed (Beets & Barron, 2000).

Geralmente, SSAA tende a produzir os melhores resultados no que toca a técnicas de *Anti-Aliasing*, mas, é também uma dos meios que mais afeta o desempenho do *hardware*. Um exemplo de SSAA em prática encontra-se na **Figura 27**.

Figura 27 - Comparação entre o uso ou não de SSAA no jogo Tomb Raider (Square Enix, 2013) (Imagem do autor).

- **FXAA**

Fast Approximate Anti-Aliasing (**Figura 28**), criado pela Nvidia, é atualmente o melhor tipo de *Anti-Aliasing* para computadores com *hardware* mais fraco, devido ao pouco impacto que este tem na GPU, visto que este suaviza a imagem à medida que esta aparece, em vez de se preocupar com a geometria 3D dos modelos de um videojogo, o seu ponto fraco é que as margens e as texturas podem ficar meio esborratadas, o que obviamente não fica tão bem como os resultados obtidos através do uso de MSAA ou SSAA. FXAA é também o modo

de *Anti-Aliasing* usado pela Unreal Engine 4 devido a ser um método eficiente em termos do uso da GPU. Na **Figura 29** encontra-se um exemplo de FXAA em utilização.

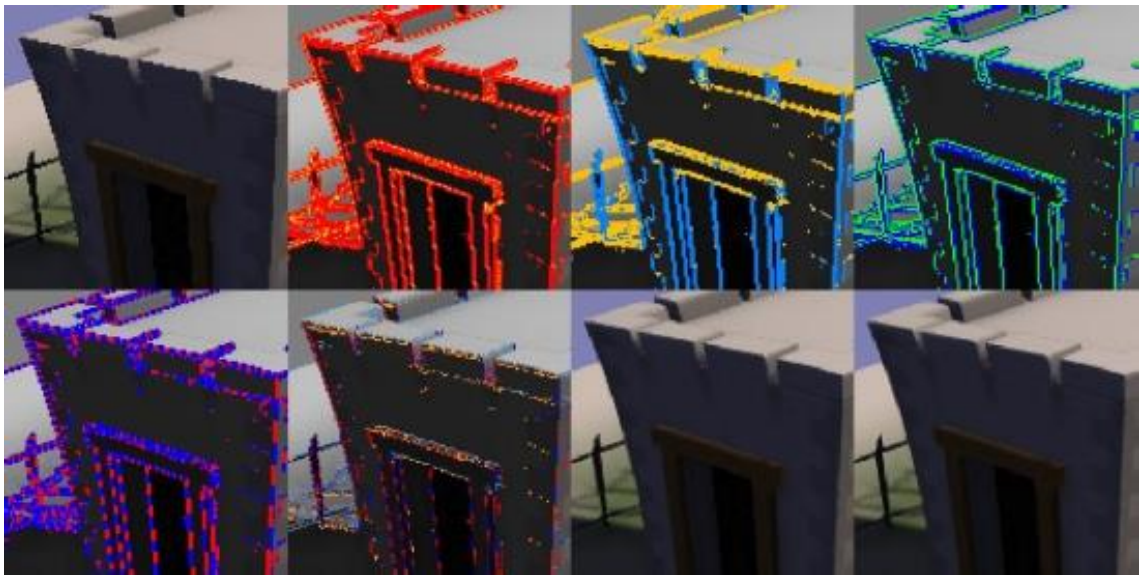

Figura 28 - Exemplo de como o algoritmo de FXAA funciona, da esquerda para a direita e de cima para baixo, retirado de FXAA (Lottes, 2009).

Figura 29 - Comparação entre a utilização ou não de FXAA no jogo Tomb Raider (Square Enix, 2013) (Imagem do autor).

- **TXAA / TAA**

Temporal Anti-Aliasing, introduzido também pela Nvidia, é um tipo único e complexo de *Anti-Aliasing* que utiliza várias técnicas de maneira a lidar com as margens irregulares, mas também com *Aliasing* temporal, suavizando movimento, no entanto, TAA afeta bastante o desempenho, sendo, portanto, pouco utilizado até à data. Um exemplo de TAA encontra-se na **Figura 30**.

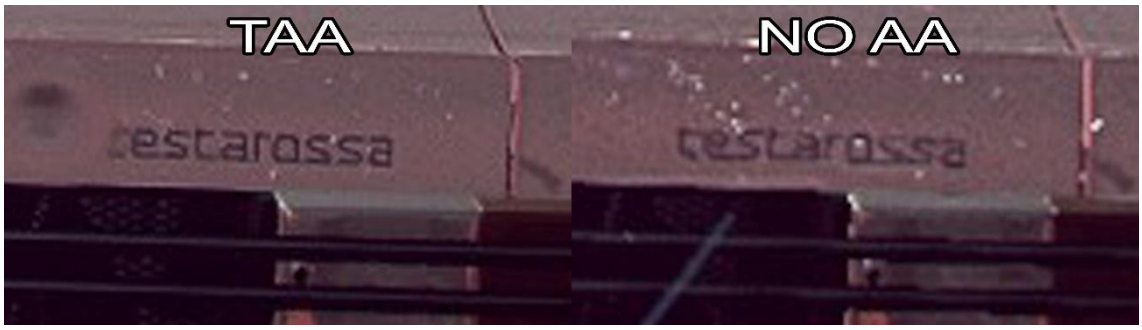

Figura 30 - Comparação entre a utilização ou não de TAA no jogo Need For Speed Heat (Ghost Games, 2019) (Imagem do autor).

No caso do TXAA este é semelhante ao TAA, mas adiciona ao algoritmo de *Anti-Aliasing*, alguns elementos provenientes do MSAA. O que proporciona um resultado superior em retorno de um impacto no desempenho ainda mais elevado ao TAA. Um exemplo de TXAA encontra-se na **Figura 31**.

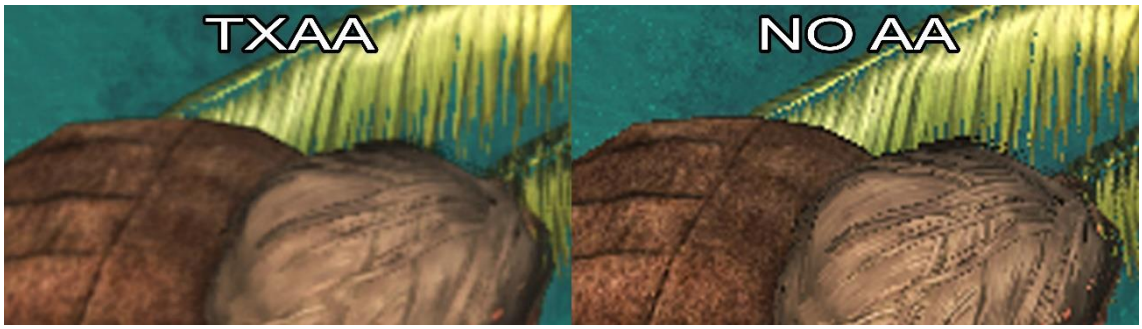

Figura 31 - Comparação entre a utilização ou não de TXAA no jogo Assassins Creed IV: Black Flag (Ubisoft, 2013) (Imagem do autor).

- **DLSS**

Deep learning supersampling desenvolvido pela Nvidia, e, atualmente, limitado a funcionar em *GPUs* Volta e Turing, visto, utilizar os *Tensor Cores* implementados nestas categorias de arquitetura GPU, é um método de *Anti-Aliasing* que usa modelos de *deep learning* construídos em supercomputadores da Nvidia, que permitem ao GPU gerar imagens mais definidas e detalhadas, ao aumentar a resolução da mesma, como SSAA.

Note-se que de momento, este tipo de Anti-Aliasing está apenas disponível para placas gráficas da Nvidia como mencionado previamente, do tipo RTX, ou seja, outras placas da Nvidia que não sejam RTX, ou de outras marcas como a AMD não podem usufruir deste tipo de *Anti-Aliasing*, embora a AMD esteja a desenvolver o seu próprio tipo de Anti-Aliasing deste género, conhecido como "*FidelityFX Super Resolution*" que permite que outros tipos de hardware sem serem RTX possam usufruir de uma tecnologia semelhante ao DLSS. Da mesma forma, a Intel possui a sua própria tecnologia, conhecida como Intel XeSS (Intel Xe Super Sampling).

Com isto, de certo modo os problemas causados por *Aliasing* são bastante reduzidos, e, na maioria das vezes, desaparecem totalmente em termos do que é percetível pelo olho humano, mas há ainda algumas vantagens com o uso do DLSS, sendo uma grande melhoria

do desempenho do videojogo, onde os *frames* por segundo podem passar de medíocres ou insuficientes para uma boa experiência, para uma experiência de jogo suave ou até excelente, exemplo disto encontra-se na **Figura 32** onde sem DLSS o jogo tem uma quantidade de fps (frames por segundo) insuficiente para uma boa jogabilidade, mas com este ligado, mesmo na opção de qualidade, este já se considera jogável, embora haja uma pequena diminuição na qualidade do *Aliasing*, e se o utilizar optar pela configuração de *ultraperformance* o jogo ainda mais jogável se encontra, aproximando-se do que é considerado ideal no que toca a fps em jogos de PC, que seriam os 60fps (Sean Whaley, 2018).

Figura 32 - Comparação entre as várias configurações disponíveis de DLSS no jogo Cyberpunk 2077 (CD Projekt RED, 2020) (Imagem do autor).

Qual é o método mais correto?

Tendo em consideração que, apenas alguns tipos de *Anti-Aliasing* foram mencionados e demonstrados acima, para se poder realizar uma verdadeira escolha de qual o melhor método a utilizar, era necessário agora analisarem-se todos os outros imensos tipos de *Anti-Aliasing* existentes e criados, ou, como foi feito no caso do *DOOM* (id Software, 2016), criar o próprio tipo de Anti-Aliasing para o produto em questão, neste caso, o TSSAA (*Temporal Super Sampling Anti-Aliasing*) que se pode observar na **Figura 33** e **Figura 34**.

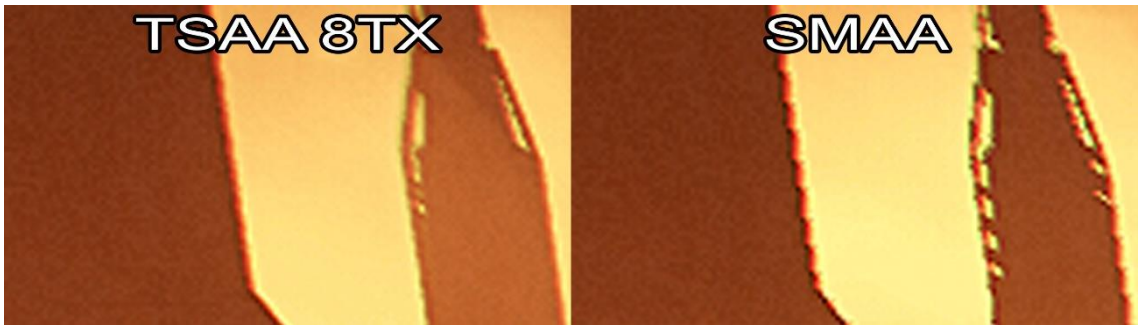

Figura 33 - Comparação entre o uso de TSAA 8TX e SMAA no jogo DOOM (id Software, 2016) (Imagem do autor).

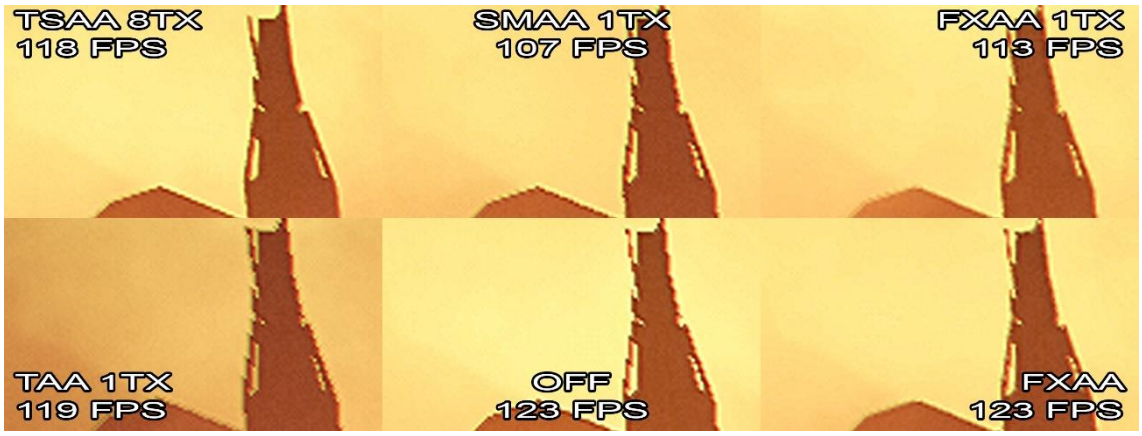

Figura 34 - Comparação entre as várias opções de *Anti-Aliasing* disponíveis no jogo DOOM (id Software, 2016) e o desempenho de cada opção (Imagem do autor).

Com isto, colocam-se desde já dois pontos muito importantes na escolha de qual é o melhor método, e a resposta de certa maneira muda consoante uma quantidade e de pontos como, por exemplo:

- A equipa desenvolvedora do videojogo tem a possibilidade de criar um tipo de Anti-Aliasing à medida para o videojogo?
- Que tipo de jogo será? Ou seja:
    - Será um jogo com o intuito de quebrar records gráficos e tem como sistemas alvo, computadores de última geração topo de gama?
    - Será graficamente razoável e como tal orientado para consolas e a maioria dos computadores?
    - Vai ser um jogo de telemóvel onde os recursos são mais fracos?
    - …

Mas se, de certo modo, excluirmos estas questões mais específicas, e apenas nos resumirmos aos tipos de Anti-Aliasing mais comercialmente conhecidos e disponíveis, estes resumem-se aos mencionados previamente:

- MSAA (com vários múltiplos)
- SSAA (com vários múltiplos)
- FXAA
- TAA
- DLSS
- AMD FSR

Destes, o SSAA é considerado um dos melhores tipos de *Anti-Aliasing* em termos de qualidade, mas em termos de desempenho é o pior de todos (Liberatore, Longazo, Pettinati, & Weise, 2014), o que leva a que este seja apenas uma boa opção para o caso de o videojogo que se realiza, estar orientado para sistemas com um desempenho exemplar, de modo a conseguirem aguentar com o mesmo. Outro tipo que se enquadra neste patamar tanto em termos de qualidade, mas também em impacto no desempenho é o TAA, sendo ligeiramente

menos abusivo nos recursos necessários, mas que, proporciona um resultado um pouco inferior, no entanto, em ambos os casos, a diferença é quase impercetível, tirando a diferença de um a dois *frames* por segundo do resultado.

Guardando, portanto, o SSAA e o TAA para videojogos orientados a sistemas superiores, o MSAA é utilizado como um meio-termo, proporcionando uma imagem inferior à de SSAA, mas com um impacto menor no desempenho.

Chegando ao FXAA, este é conhecido por ser a forma superior de *Anti-Aliasing*, se o sistema alvo for fraco no desempenho, como, por exemplo, um telemóvel.

Temos assim três tipos de *Anti-Aliasing*, cada um orientado respetivamente a sistemas de desempenho, alto, médio e fraco.

No fim, no entanto, a opção mais acertada, partindo de que os desenvolvedores são capazes de a implementar, é, preparar o ambiente do videojogo sem nenhum método de *Anti-Aliasing* ligado, e fazer com que este tenha o melhor aspeto possível, e de seguida, implementar os três tipos de *Anti-Aliasing* (TXAA, MSAA e FXAA), e posteriormente deixar que seja o utilizador a escolher, através de um *menu* de configurações visuais, como a maioria dos videojogos hoje em dia já possui, permitindo assim adaptar a qualidade e impacto de desempenho a qualquer dispositivo.

Então e o DLSS? Das três mencionadas previamente, o DLSS é o modo superior de *Anti-Aliasing*, visto apresentar a melhor imagem, e com um impacto de desempenho semelhante ao do FXAA, mas este sofre de ser uma solução limitada apenas a certos *hardwares*, nomeadamente, GPUs da NVIDIA do tipo RTX. Como tal, é uma excelente opção de implementação como as anteriores, mas, que deve ser deixada ao cargo do jogador poder ligá-la ou não. Em alternativa existe o AMD FSR, no entanto, este ainda se encontra em desenvolvimento, e os resultados obtidos ainda não possuem a qualidade dos obtidos com o DLSS, adicionando inclusive *Aliasing* ou outros artefactos à imagem em vários casos.

No entanto, no caso da Unreal Engine, atualmente, os modos de *Anti-Aliasing* que podem ser utilizados são o FXAA, o MSAA e o TAA, como se constata na **Figura 35**, visto serem aqueles que já estão incorporados e implementados na *engine*, no entanto, há a possibilidade de implementar outros, mas para tal seriam necessários vários passos, procedimentos e configuração adicionais.

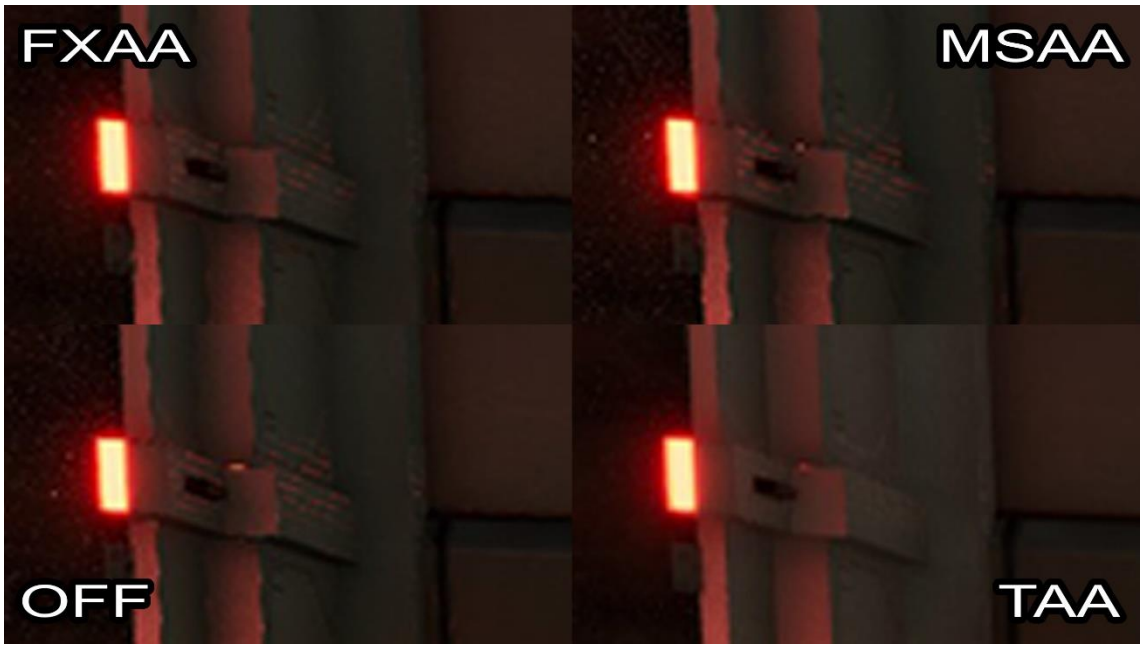

Figura 35 - Comparação entre as opções de AA disponíveis na Unreal Engine 4.26 (Imagem do autor).

Assim sendo, a questão de qual é o melhor volta a surgir, no caso da Unreal Engine, mas, a resposta acaba por ser a mesma, primeiro dependendo da habilidade do desenvolvedor de implementar os tipos de *Anti-Aliasing*, de conseguir preparar o ambiente de jogo para ter o melhor aspeto possível sem a utilização de *Anti-Aliasing*, e por fim, deixar sempre a escolha de qual tipo utilizar, ao jogador.

Lens

Na categoria de *Lens*, como foi referido anteriormente, encontram-se os filtros de *post-Processing* relacionados à lente de uma câmara, sejam estas configurações comuns como o controlo de exposição, ISO, abertura, etc., mas também alguns artefactos que podem acontecer na mesma como *Bloom*, *Lens Flare* e *Lens Dirt*.

Em baixo segue uma lista detalhada dos efeitos, desde o que cada um é e realiza, a como cada controlador se comporta e quais os resultados esperados de mexer nos controladores. A única exceção é o primeiro ponto de todos, o *Mobile Depth of Field*, que embora esteja relacionado com a profundidade de campo, apenas se aplica a projetos planeados para dispositivos móveis como telemóveis, que no caso desta dissertação não se tem como objetivo e daí não se fala do mesmo.

Bloom

Bloom é, portanto, o segundo *item* da lista, mas o primeiro de que falamos, e este é um fenómeno de luz no mundo real, que, quando adicionado a imagens sintetizadas, adiciona imenso realismo à mesma, com um impacto de desempenho reduzido.

Este efeito, adiciona franjas ou plumas de luz (**Figura 36**), que se estendem a partir das margens de áreas muito luminosas em imagens, proporcionando a ilusão de que essa área

luminosa, tem demasiada luz, ofuscando a câmara ou o olho. Estas áreas podem ser fontes de luz, como lâmpadas ou o sol, mas também reflexos destas fontes de luz.

Figura 36 - Comparação entre *bloom* ligado e desligado na Unreal Engine 4.26 (Imagem do autor).

Desde 2004, que o efeito é utilizado regularmente, devido principalmente ao facto de que os ecrãs mudaram nesta altura, passando a maioria a utilizar painéis eletrónicos em vez de CRTs, visto que os ecrãs CRT conseguiam criar efeitos de *bloom* involuntariamente. O fundamento físico do *bloom* é que, no mundo real, as lentes não conseguem focar perfeitamente, e até uma lente perfeita vai sempre convolver a imagem. Sobre condições normais, estas imperfeições não são percetíveis, mas, se a fonte de luz for muito intensa e brilhante, estas tornam-se visíveis. Como tal, a luz parece derramar para além da forma normal.

No entanto, é necessário um uso cuidado deste efeito, visto que em exagero pode levar a que a imagem acabe por parecer demasiado brilhante, exposta ou ofuscada (**Figura 37**), algo que acontecia comummente nos videojogos lançados nos anos 2000.

Figura 37 - Exemplo do uso excessivo de *bloom* no jogo Syndicate (Starbreeze Studios, 2012).

No caso da Unreal Engine, o *bloom* tem quatro controladores principais e um grande conjunto de controladores avançados. Neste caso vamos apenas falar dos quatro controladores principais, e o primeiro dos avançados.

- ***Method***

No controlador *Method* ou método, pode-se escolher entre *Standard*, que permite depois utilizar os controladores *Intensity* e *Threshold*, e o método *Convolution*, que desliga a *Intensity* e o *Threshold*, mas liga o *Convolution Kernel*.

- ***Intensity***

Intensity ou, intensidade controla quão forte o efeito de *bloom* é, no caso de este ter definido como método o Standard. Na **Figura 37** acima, o problema do exagero do *bloom* é não só, mas em grande maioria proveniente de uma intensidade demasiado elevada, o que faz com que tudo o que é considerado como fonte de *bloom* é convertido numa quantidade exagerada de brilho.

- ***Threshold***

Threshold controla o ponto em que uma determinada parte de uma cena é considerada para criar *bloom*, o que faz com que um *threshold* de 0 considere que tudo deve criar *bloom*, criando uma imagem demasiado brilhante e ofuscante, se o controlador for colocado ao máximo, nenhum objeto será considerado e como tal não irá existir *bloom*.

- ***Convolution Kernel***

Caso o método escolhido seja o *Convulution* em vez de Standard, o desenvolvedor pode agora adicionar um efeito ao seu tipo de *bloom*, que, em vez de este ser refletido como um desfoque dos pontos luminosos, este pode, por exemplo, ser uma estrela, ou um coração, isto pode assim, converter o habitual efeito de *bloom*, num efeito mais semelhante a, por exemplo um *flare* na lente (*lens flare*).

É, no entanto, preciso ter cuidado com o uso deste filtro, pois este acaba por desfocar a imagem a partir de um certo ponto. Para tal convém ajustar o controlador *Convolution Scale* encontrado nos parâmetros avançados, como se pode verificar na **Figura 38**, que ao utilizar a textura da **Figura 39**, dependendo da escala (*scale*), a imagem pode ficar totalmente desfocada.

Figura 38 - Comparação entre valores diferentes de convolução na Unreal Engine 4.26 (Imagem do autor).

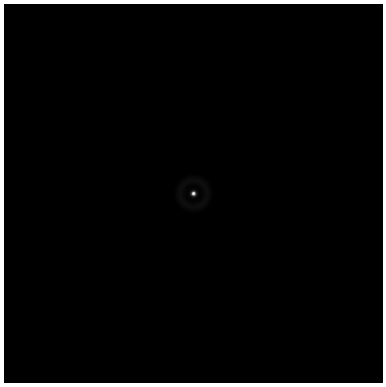

Figura 39 - Textura utilizada para o efeito de convolução na Figura 38 (Imagem do autor).

- ***Size Scale***

O último ponto em que se foca é o *Size Scale*, ou o tamanho que o *bloom* terá. Ao aumentar este controlado, o efeito de *bloom* aumenta de tamanho, ao ponto de que, uma lâmpada que antes tinha um *bloom* cuja aura tinha um centímetro de diâmetro em redor da mesma, pode vir a ter 1 metro.

Exposure

O segundo *item* de que se fala é o *Exposure*, em português, exposição, e este é também conhecido no mundo do *post-Processing* como *Autoexposure* ou *Eye Adaption*, previamente era também conhecido como *Bloom* e HDR, e este efeito, nomeadamente o *autoexposure* baseia-se na habilidade que a retina do olho possui, que, mediante vários níveis de luminosidade, se ajusta, da maneira a poder haver boa visibilidade independentemente de, por exemplo, estar demasiado sol, ou estar de noite, portanto, estar escuro. Este efeito eventualmente foi recriado em câmaras fotográficas e de vídeo, através de anos de evolução, embora, neste caso, o que acontece, é que o processador na câmara, calcula as configurações corretas da câmara, mediante, o nível de luz, para que, por exemplo, quando se fotografa algo de dia, não haja um risco que a fotografia tenha demasiada luz, portanto, haja demasiados clarões e brilho, mas poucos contrastes, detalhes e afins, como, se a foto fosse de noite, o que resulta numa foto muito escura, com exceção das luzes emitidas por candeeiro, por exemplo.

No entanto, em videojogos e animação 3D, por natureza, este efeito não existe, assim como nas câmaras fotográficas e de vídeo, o que faz com que o produto final seja pressentido como irrealista, ou seja, o jogador ou espetador, será incapaz de sentir a diferença da luz que existe numa gruta ou túnel, comparando com o que este sente num campo aberto durante a tarde, na verdade, o túnel parecerá demasiado claro, e o campo um pouco mais escuro do que o que seria real.

Com este filtro de *post-Processing*, o problema mencionado previamente é remediado, e, como tal, a animação ou jogo irá ser mais realista na forma como a luz é percecionada. Este efeito pode em alguns casos, no entanto, também ser conhecido como HDR, embora, recentemente a desambiguação entre HDR e *Autoexposure* esteja a ser tratada, visto que, HDR é um termo diferente, comummente associado ao método fotográfico onde uma cena é fotografada várias vezes em exposições diferentes, e posteriormente analisada para criar uma fotografia única com mais detalhes. Com isto dito, os componentes que fazem parte deste filtro são:

- ***Metering Mode***

O primeiro controlador de todos define qual é a mediação que controla o *Autoexposure* ou a exposição automática da cena. Existem três tipos diferentes de mediação:

- *Autoexposure Histogram* — este é conhecido com o modo padrão da *engine* visto que é o que oferece o controlo mais preciso das três mediações de exposição, baseando-se num histograma de 64 compartimentos.
- *Autoexposure Basic* — O segundo modo de exposição automática oferece menos controlo e menos configurações, mas que funciona mais rápido.
- Manual — O último modo, elimina a exposição automática e começa a necessitar de ser controlado efetivamente pelas câmaras espalhadas pela cena, mais semelhante ao que seria utilizado por um profissional sem exposição automática, e com várias câmaras ao seu dispor.

- ***Exposure Compensation***

Continuando, o segundo controlador, semelhante ao que acontece com câmaras fotografias, de filmar e até nas câmaras que os telemóveis dos dias de hoje possuem, aumenta ou diminui a quantidade de iluminação, ou seja, compensa a quantidade de exposição de uma cena. Neste caso da Unreal Engine, por cada unidade de valor, o dobro da exposição é aumentado ou diminuído, sendo então que 1 corresponde ao dobro, 2 corresponde ao quádruplo em diante. Em baixo, na **Figura 40** pode-se observar como a iluminação funciona entre -2 a 2 de compensação.

Figura 40 - Exemplo de cinco níveis diferentes de compensação de espoição na Unreal Engine 4.26 (Imagem do autor).

- ***Apply Physical Camera Exposure***

Ao ativar este *item,* a luminosidade da cena é afetada pelas configurações colocadas na categoria *Camera*, caso este esteja desativado, as configurações serão, ISO 100, abertura 1.0 e Velocidade do Obturador 1.0. Esta opção apenas funciona se a medição de exposição for manual.

- ***Exposure Compensation Curve***

Caso o artista deseje, este pode recorrer a um *asset* conhecido com *curve,* que permite um controlo mais detalhado e minucioso da compensação de exposição, neste caso é importante possuir a sobreposição do histograma sobre a cena (Show — Visualize — HDR) para se receber um gráfico semelhante ao apresentado na Figura 41, onde 1 corresponde ao EV100 aproximado da cena e 2 corresponde a compensação da curva, ambos correspondentes ao ponto focal da sobreposição, ou seja, o pequeno quadrado no centro da cena.

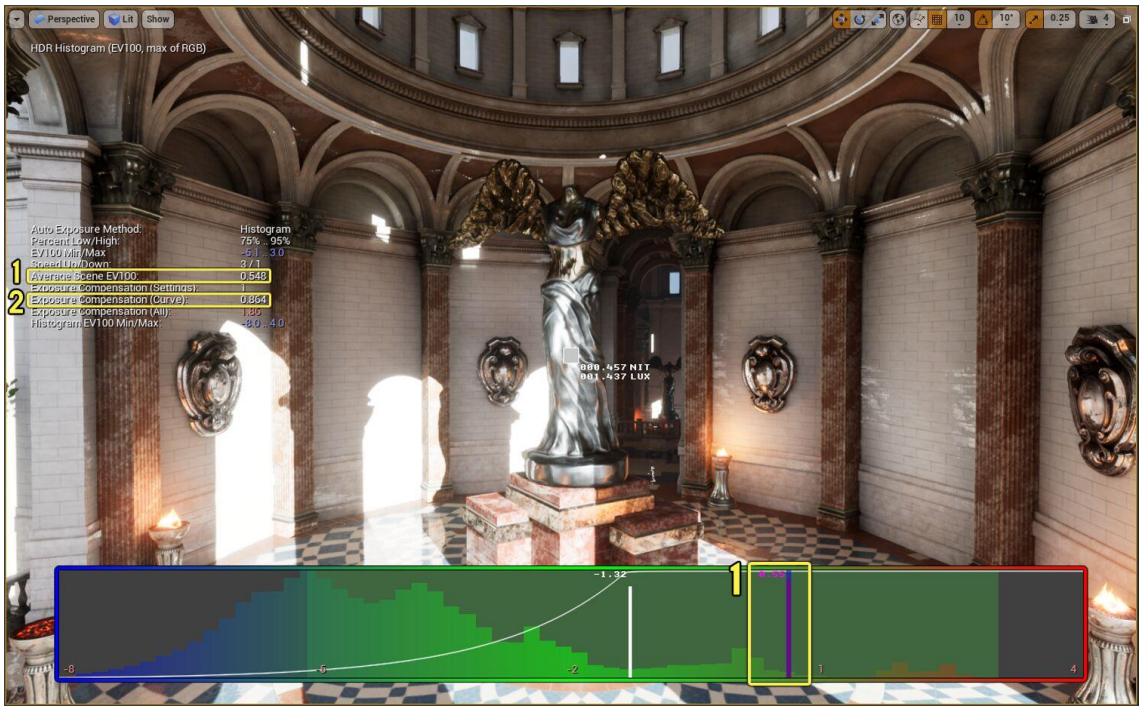

Figura 41 - Exemplo da sobreposição de HDR na Unreal Engine (Epic Games, 2020).

Assim sendo, durante a criação do gráfico ou da *curve*, os valores de X irão corresponder ao ponto 1 da figura anterior, ou seja, ao valor aproximado da cena em EV100, e os valores no eixo dos Y correspondem à compensação que o artista deseja aplicar nesse ponto. Sendo importante ver o valor devolvido no ponto em baixo do 2, *Exposure Compensation (All)* que corresponde a quanta exposição está nesse ponto focal, incluindo o valor de compensação coloca anteriormente.

- ***Exposure Metering Mask***

Adicionalmente, se o artista desejar, este pode ainda controlar a exposição da cena com uma máscara, ou seja, uma imagem onde os pontos mais claros da mesma resultando numa exposição maior e os pontos escuros numa exposição menor da imagem, uma demonstração do que ocorre pode ser constatado na **Figura 42** onde à esquerda tem-se a máscara utilizada, seguido da cena sem máscara e depois a mesma cena, com a máscara.

Figura 42 - (da esquerda para a direita) Mascara utilizada para a medição da exposição, cena sem máscara e cena com máscara (Epic Games, 2020).

- ***Min Brightness***

Neste ponto, é decidido como a exposição automática se comporta com zonas de fraca luminosidade, onde, valores menores resultam numa cena mais iluminada, após a exposição ser automaticamente adaptada, dito isto, devido à exposição automática e o modo como a Unreal Engine funciona, haverá sempre iluminação, mesmo que a zona possua absolutamente nenhuma fonte de luz e esteja enquadrada no volume do *post-Processing*. Valores superiores a 1 têm um resultado igual a 1.

- ***Max Brightness***

Semelhante à configuração anterior, o valor do *Max Brightness* decide como a exposição automática se comporta em zonas de luminosidade elevada, com isso, o valor tem de ser superior a 0, ou a iluminação aparecerá de tal ponto iluminada que apenas se vê luz, nada mais. Quanto maior o valor, menor a iluminação da cena em zonas muito iluminadas.

- ***Speed Up* e *Speed Down***

Neste ponto define-se, a velocidade da adaptação da exposição, de um ambiente escuro para um demasiado iluminado (*Speed Up)* e vice-versa *(Speed Down)* sendo que o valor corresponde a F-stops por segundo.

Chromatic Aberration

Mais um efeito que historicamente tinha origens indesejadas, neste caso em relação às cores de uma foto que devido á lente da camara fotográfica ou do método de revelação apresentava cores que não coincidiam como deve de ser, o que causava aberração em termos de cor no produto final, onde os canais vermelho, verde e azul do produto não se encontravam. No entanto este problema tem sido reduzido já há bastante tempo, especialmente com o evoluir das camaras fotográficas, mas dependendo da qualidade da lente implementada na camara, há ainda nos dias de hoje possibilidade de reparar neste efeito, mesmo que este seja extremamente reduzido e por vezes impercetível ao olho humano sem realizar um zoom da foto. É ainda possível presenciar este efeito no mundo real em casos que não envolvam camaras fotográficas ou cinemáticas, como por exemplo, quando a luz passa por um vidro, esta pode começar a ser refratada, e como tal a luz que entrou inicialmente no vidro começa a sofrer uma aberração de cor, especialmente quando este raio de luz passa nas extremidades do vidro, ou no caso das camaras, das extremidades da lente. Este efeito é inclusive globalmente conhecido através da experiência de passar um raio de luz por um prisma, onde entra um raio de luz branco, e do outro lado saírem as várias cores constituintes do espectro de cores visíveis do olho humano. Outra referencia a este efeito é a capa do álbum The Dark Side of the Moon (Pink Floyd, 1973) demonstrada na **Figura 43**.

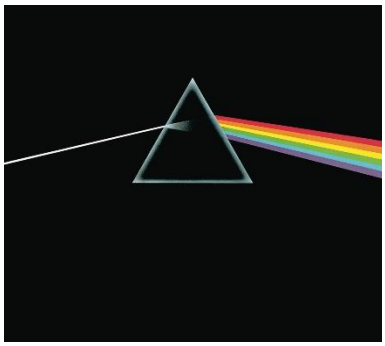

Figura 43 - Capa do álbum The Dark Side of the Moon (Pink Floyd, 1973).

Sendo, no entanto, este efeito algo que é conhecido como um artefacto indesejado, é normal surgirem questões relativas à utilização do mesmo, pois seria a princípio melhor não o utilizar, mas tal e qual como aconteceu com o *Lens Flare*, há um lado artístico e criativo do efeito que permite a que um determinado desenvolvedor ou artista criar certos tipos de imagens que necessitam deste efeito para mostrar as suas ideias. Mais do que isso, o efeito é comumente utilizado para tentar corrigir o que se pode considerar como um problema em termos de gráficos de computador. Este problema provém de que os gráficos de computador são tão "limpos" que de modo que estes pareçam um pouco mais realistas, aplicar um pouco de aberração cromática leva a que estes gráficos não sejam tão limpos, onde as margens ficam um pouco desfocadas, mas muito ligeiramente.

Sendo assim tem-se pelo menos já duas razões pela qual o uso de aberração cromática pode ser importante, para que a cena criada seja um pouco mais suave e realista em vez de ter um aspeto limpo e computadorizado, e o outro, como ferramenta criativa, por exemplo, a criação de um videojogo que retrata um documentário fílmico dos anos 20 onde as camaras eram de baixa qualidade. Exemplo disto é o videojogo Alien Isolation (Creative Assembly, 2014), que tentou recriar fielmente todas as variáveis e componentes do filme original, Alien (Ridley Scott, 1979), e assim sendo, para alem do ambiente e objetos 3D e da iluminação, estes recriaram todos os defeitos que existiam no original, como o *lens flare*, o tom da pelicula de filme e a aberração cromática das camaras utilizadas.

Em termos de *engine*, a Unreal Engine possui dois controladores para configurar o aspeto e intensidade da aberração cromática, nomeadamente, a *Itensity* que controla a intensidade de aberração cromática presente na cena, e o *Start Offset*, que controla a partir de que ponto a aberração aparece, seguindo a forma da cena. Como se vê na **Figura 44,** devido á cena ser retangular, a aberração cromática acontece com a mesma forma e proporção, e inclusive vê-se em o ponto onde a aberração desvanece e a imagem passar a deixar de possuir o efeito.

Figura 44 - Demonstração de como a aberração cromática é aplicada à cena, na Unreal Engine 4.26 (Imagem do autor).

Dirt Mask (Lens Dirt)

Dirt Mask ou como é conhecido comumente, *Lens Dirt* ou seja, sujidade na lente, é o primeiro efeito de post-Processing da lista cujas origens provêm de artefactos numa lente de camara, alias, é bastante provável que já tenha presenciado este problema mesmo que não se tenha apercebido dele, visto que este pode ser visto em vários outros objetos para alem da lente de uma camara, e com o avanço tecnológico que se tem nos dias de hoje e com os telemóveis a possuírem camaras fotográficas, é bem possível que, caso o seu telemóvel possua uma camara, que esta possua sujidade, e como tal sofra de *Lens Dirt*, mesmo que seja muito subtil e de tamanho bastante reduzido, como se vê na **Figura 45**.

Figura 45 - Sujidade presente numa lente de um telemóvel (Sony Xperia X Compact) assim como noutros elementos relacionados à função de filmar/fotografar do telemóvel (Imagem do autor).

Pois bem, como mencionado e entendido perfeitamente pelo nome, *Lens Dirt* é realmente apenas sujidade numa lente, seja de uma câmara fotográfica, de filmar, num capacete ou até nos óculos e embora regra geral seja desejável que quando se tira uma foto ou se grave um filme, que se tenha sempre atenção à limpeza das lentes de modo que este problema não exista ou não seja percetível pelo menos, este pode também ser usado como algo artístico e no que toca a jogos, este tem inclusive sido abusado desde que começou a surgir,

especialmente se o jogo for um FPS de guerra, onde cada vez que uma bomba cai relativamente perto de um jogador, ou uma bala atinge o solo perto dele, ou até quando a bala atinge o jogador, é comum aparecer sujidade no ecrã, como se observa na **Figura 46**, simbolizando haver sujidade a atingir a lente de uma câmara e como tal esta fica suja, tentando assim proporcionar a ideia de realismo, embora isto faça pouco sentido, pois se a ideia é que o jogador encara a personagem, a única maneira de este efeito ser realista, era se a personagem utiliza-se óculos, ou se esta fosse um repórter a segurar uma câmara, mas aí não fará sentido segurar uma câmara e uma arma simultaneamente, a não ser que a personagem tenha pelo menos quatro braços.

Figura 46 - Exemplo de *Dirt Lens* no jogo Battlefield 3 (DICE, 2011).

No caso da Unreal Engine, como seria de esperar, existe então a possibilidade de adicionar este efeito, e para tal possui-se três controladores para tal. No entanto, é importante saber que, este efeito é influenciável pelo *bloom* e pela iluminação da cena, ou seja, dependendo da quantidade de iluminação, uma parte da sujidade pode sobressair em comparação a outras partes com menos luz que a primeira.

- ***Dirt Mask Texture***

O primeiro controlador de todos permite adicionar a sujidade desejada à lente, através de uma imagem, que será, portanto, representativa de como a sujidade aparece na nossa cena, por exemplo, como se observa na **Figura 47**.

Figura 47 - Exemplo de como a Unreal Engine 4.26 aplica *Lens Dirt* (Imagem do autor).

É importante também denotar que a proporção da textura deve corresponder ao tamanho do ecrã onde o produto final será apresentado, caso contrário, pode haver uma deformação da mesma, assim como é importante saber que a textura deve possuir um fundo preto, pois tudo o que seja superior a isso começa a aparecer no ecrã, e, caso a textura possua cores para além das neutras, estas iram ser contabilizadas, tal e qual como se pode denotar na **Figura 48**, onde ao utilizar a imagem representada numa escala menor no canto inferior esquerdo da figura, o efeito para além de distorcer a imagem, cria um efeito indesejável, bem distante do que seria considerado como sujidade na lente, cobrindo inclusive o ecrã quase na totalidade, tirando onde se tem os limites da esfera, pois estes tinham a cor preta.

Figura 48 - Exemplo do que acontece à cena, se a imagem utilizada para *Lens Dirt*, na Unreal Engine 4.26, não for adequada (Imagem do autor).

- ***Dirt Mask Intensity***

De seguida tem-se o controlador da intensidade, que controla a intensidade da sujidade que aparece na lente. Sendo que um valor de 0 faz com que nada apareça, e um de 8 que apareça já um bom bocado, este pode, no entanto, ser levado acima do valor de 8, escrevendo manualmente o valor em vez de arrastar o controlador, para aumentar a intensidade, caso 8 não seja o suficiente para criar a ideia desejada, uma comparação entre várias intensidades pode ser examinada em baixo na **Figura 49**.

Figura 49 - Comparação entre várias intensidades de *Dirt Mask* na Unreal Engine 4.26 (Imagem do autor).

- ***Dirt Mask Tint***

Por fim, pode-se adicionar um tom colorido à sujidade se assim o desejarmos, através do *Tint*, por exemplo, para dar a ideia de que em vez de se ter pó na lente, se tem sangue, como é exemplificado na **Figura 50**, ou outro líquido colorido qualquer.

Figura 50 - Exemplo de como a Unreal Engine 4.26 aplica *Tint* à *Dirt Mask* (Imagem do autor).

Camera

Entrando na categoria de *Camera*, esta apresenta as configurações base que uma câmara fotografia ou de filmar normalmente apresenta, um triângulo, nomeadamente a velocidade do obturador, o ISO e a abertura do diafragma, uma representação gráfica deste triângulo pode ser conferida na **Figura 51**, assim como o que cada ajuste influencia. Para além destes três pontos encontra-se ainda a abertura máxima do diafragma e a quantidade de lâminas que constituem o diafragma. Para além destes controlos e componentes, as câmaras costumam ainda ter a opção de compensação de exposição, no entanto, visto já termos falado desta denotamos apenas que os filtros *Camera* e *Exposure* funcionam em conjunto, e inclusive o *Depth of Field* que encontraremos mais à frente faz parte também deste conjunto de filtros que se relacionam uns com os outros.

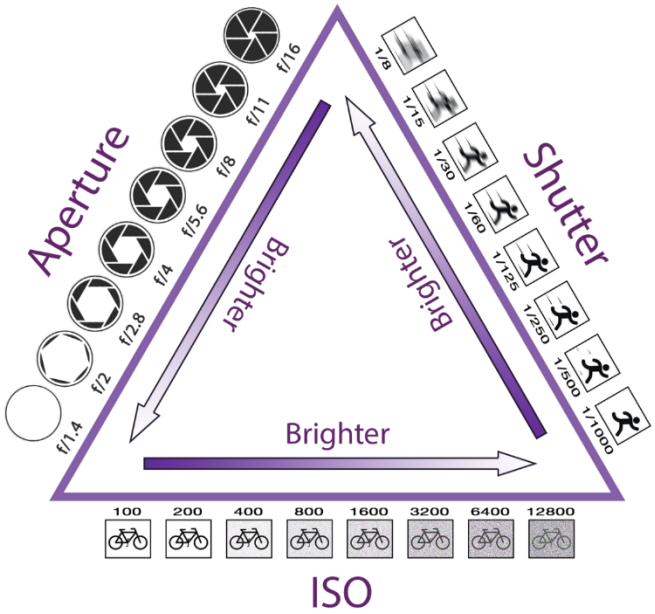

Figura 51 - Esquema dos três princípios da fotografia (s/i).

Visto, no entanto, que este trabalho se foca nos videojogos e no *post-Processing*, não se aborda a fundo como uma câmara fotográfica funciona, mas antes de seguirmos com a explicação das propriedades de cada controlador, é importante ter em atenção que apenas podemos ligar os controladores do ISO e da velocidade do obturador, se o controlador *Metering Mode* falado previamente na exposição, estiver definido como manual.

- ***Shutter Speed***

O primeiro *item* de todos controla a velocidade do obturador, este no caso da *engine* serve principalmente para controlar a iluminação, no entanto, em termos de fotografia, este controlador serve também e principalmente para controlar o tempo que o obturador está aberto, ou seja, durante quanto tempo a luz entra na lente, isto causaria a que no caso de uma fotografia, a imagem apareça ou muito esborratada, ou como se o tempo tivesse sido congelado totalmente, como se pode conferir na **Figura 52**.

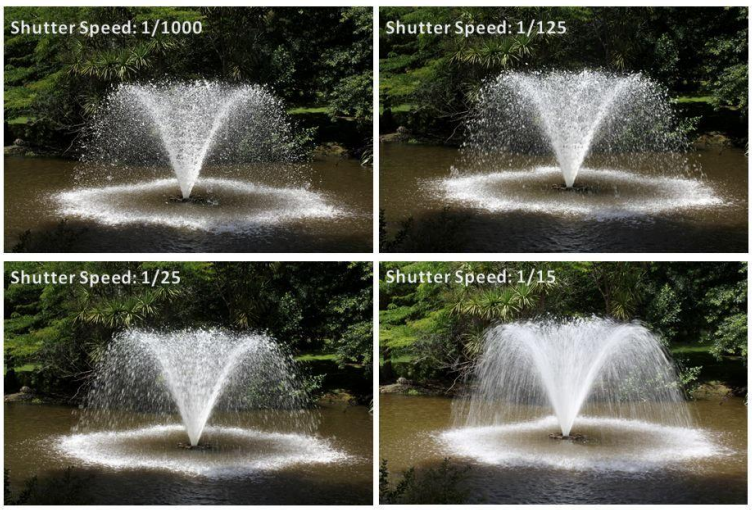

Figura 52 - Comparação entre velocidades de obturador, diferentes (s/i).

Exemplos totalmente opostos em termos da velocidade de abertura do obturador são, no caso de o obturador estar muito tempo, ou seja, a velocidade for baixa, as fotos variadas que se vêm de cidades, onde as luzes dos carros deixam um rasto como na **Figura 53**.

Figura 53 - Timelapse noturno no Japão, exemplificando o que acontece se a velocidade do obturador for demasiado longa, especialmente em condições noturnas com várias fontes luminosas em constante movimento (faróis dos automóveis) (s/i).

Já no caso oposto, se a velocidade for muito alta, ou seja, a luz tiver um tempo extremamente reduzido para entrar, o movimento parece congelado ou parado dependendo de quão alta a velocidade é, semelhante ao que se pode observar nas fotografias de corridas de carro como, por exemplo a da **Figura 54**.

Figura 54 - Fotografia demonstrativa de uma velocidade de obturação alta, onde o carro é capturado totalmente congelado, incluindo as rodas (McGrath, 2013).

Regressando à *engine*, e a como o controlador funciona na mesma, este controla essencialmente a quantidade de luz da cena, em termos de *gameplay*, no entanto, caso a *engine* esteja a ser usada para criar um produto que não seja um jogo ou somente um, como, por exemplo uma animação ou uma curta-metragem ou até uma sintetização fotográfica, o efeito já mostra mais das suas capacidades, congelando ou esborratando itens em movimento numa cena. Semelhante ao que se encontra, por exemplo no modo de fotografia da série de jogos de Gran Turismo (Polyphony Digital, 1997) (Grixti, 2017) representado na **Figura 55**, entre outros videojogos que começaram também a adaptar o modo de fotografia.

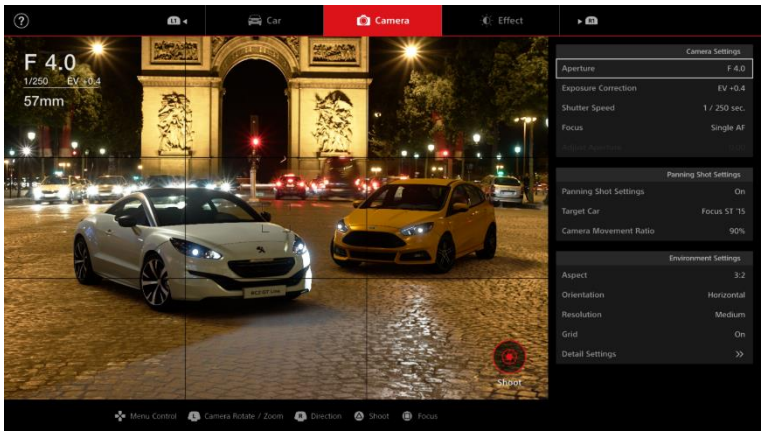

Figura 55 - Modo de fotografia no jogo Gran Turismo Sport (Polyphony Digital, 2017)

- **ISO**

O segundo *item* da lista e ainda pertencente ao triângulo fotográfico é o ISO, este que em termos fotográficos serve para controlar a iluminação, e regra geral é o último em que se mexe, isto porque este tem uma grande tendência a adicionar grão e ruído às fotos se este for demasiado alto como se pode presenciar na **Figura 56** onde mantendo um valor de exposição de 0, a imagem começa a ganhar ruído à medida que o ISO sobe.

Figura 56 - Demonstração do aumento do ruído na imagem à medida que o ISO aumenta (Imagem do autor).

Felizmente, no caso da *Unreal Engine*, este apenas permite controlar a luz, sem o artista ter de se preocupar com a possibilidade de a sua cena começar a possuir ruído. É possivelmente também por esta razão que no caso da *engine*, o ISO e a velocidade do obturador sejam desligados quando se liga o modo de metragem automático em vez de manual no controlador de exposição.

- ***Aperture***

O terceiro *item* da lista, último do triângulo e o primeiro que funciona independentemente da metragem da exposição, a *aperture* ou abertura do diafragma controla o tamanho de luz que entra, como tal mais uma fonte que pode controlar a quantidade de luz que entra numa cena, mas também o que define a profundidade de campo, como exemplificado na **Figura 57**, embora mais uma vez, de modo a poder utilizar devidamente este para controlar a profundidade de campo. É também importante saber que o valor apresentado corresponde ao F-Stop, ou seja, quanto maior o valor for, na verdade, menor a abertura, como se pode ver abaixo na figura.

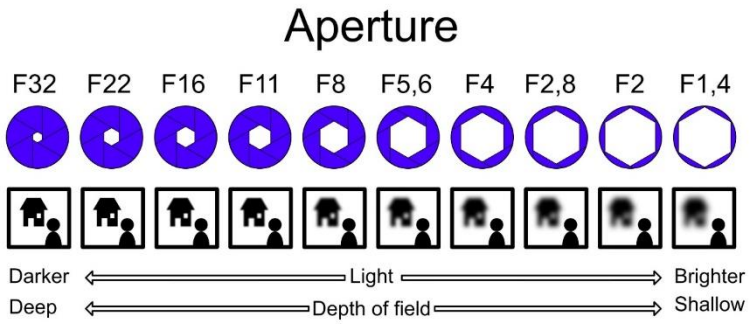

Figura 57 - Esquema demonstrativo do que acontece à imagem mediante valores de abertura do diafragma, diferentes (s/i).

- ***Maximum Aperture***

Relacionado ao controlador anterior, este define o tamanho máximo que a abertura pode ter, ou seja, se eu definir um valor de 8 neste controlador, mesmo que o anterior possua o valor de 32, este é anulado e reduzido para 8, pois não pode ser superior ao definido neste controlador. Este controla ainda se as laminas que se definem a seguir, são direitas ou curvas.

- ***Number of diaphgram blades***

Por último entra-se nas propriedades do diafragma no que toca à maneira como este é construído, e como tal, quantas laminas constituem o diafragma, como exemplificado na **Figura 58**, este efeito é especialmente importante, pois controla como será o look dos *bokehs*, como se observa na **Figura 59**.

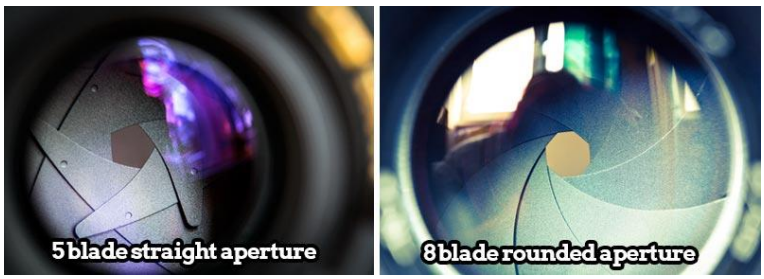

Figura 58 - Exemplo de diafragmas com uma quantidade diferente de lâminas e a forma como estes deixam a luz entrar (s/i).

Figura 59 - (três) *Bokeh* diferentes devido ao diafragma das lentes ter quantidades diferentes de lâminas (s/i).

Lens Flares

No mundo real, quando raios de luz, provenientes de uma ou mais fontes muito brilhantes, atingem a lente de uma câmara, estes raios têm tendência a serem refletidos em várias partes no interior de uma lente, até finalmente atingirem o sensor, o que leva à criação de artefactos conhecidos com *Lens Flare* e *Ghosting*, como se vêm na **Figura 60**. Este problema é, no entanto, mais comum em câmaras fotográficas ou de filmar que utilizem lentes com várias peças no interior, as lentes comummente conhecidas como lentes profissionais, que permitem ao utilizador ajustar uma quantidade enorme de configurações, desde ampliação à abertura etc. No entanto, este acontecimento também existe noutras categorias de lentes fotográficas, como, por exemplo as câmaras dos telemóveis dos dias de hoje, e até em capacetes, viseiras e óculos.

Figura 60 - Exemplo de *Lens Flare* e *Ghosting* na lente (s/i).

Como tal, estes artefactos podem ser considerados como indesejados, visto que por vezes chegam até a reduzir o contraste de uma imagem, ou a criar uma nébula luminosa demasiado grande que ofusca a maioria da imagem, e é devido a isto que no caso dos inícios do cinema, as equipas de produção tentavam ao máximo reduzir estes artefactos, através de imensos truques, o mais conhecido proveniente de pintar as lentes das câmaras com vernizes e camadas que reduzem o *glare* ou *bloom* e o *lens flare*. Inclusive atualmente vemos esta técnica a ser aplicada a óculos e até às viseiras dos capacetes de condutores profissionais, visto que o *lens flare* nestes é reduzido devido a não haver tantos pontos de refração e reflexão, mas ainda assim serem o suficiente para causarem vários transtornos. Para além destes truques há outros como coberturas de lentes, que reduzem a quantidade de luz indesejada ou fora de cena que entra na lente, como é ilustrado na **Figura 61**.

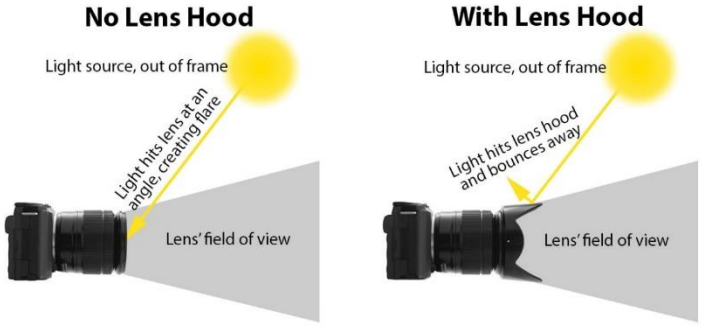

Figura 61 - Ilustração de como uma cobertura de lente previne *Lens Flare* (s/i).

No entanto, a opinião que se tinha de que um filme ou fotografia com *lens flare* era considerado o trabalho de um amador ou de baixa qualidade veio com os tempos a mudar, pois, começou-se a argumentar que, para um filme ser real, ou seja, conduzido no exterior e não dentro de quatro paredes, este terá *lens flare*, e como tal nos anos 60 os filmes começaram a sair com mais *lens flare*. Anos depois até os filmes realizados no interior de estúdios e como tal nas tais quatro paredes começaram a possuir *lens flares*, por vezes adicionados à *posteriori*. Isto começou-se a ver especialmente em filmes de ficção científica, onde *lens flare* era um efeito predominante (Vox, 2016). Como tal atualmente o efeito pode ser considerado um erro, mas é mais comummente considerado como algo necessário para representar realismo ou até como uma ferramenta criativa e artística para que o resultado atinja a ideia original do criador. Especialmente quando se considera que atualmente, imensos filmes de cinema têm uma abundância de *flares* feita a partir de CGI, onde, o *Lens Flare* não existe naturalmente, especificamente porque não existe uma lente para começar, e como tal, muitos dos *flares* existentes são criados em pós-produção, como exemplificado na **Figura 62,** numa cena retirada do filme Avengers: Infinity War (Anthony Russo e Joe

Russo, 2018) que ocorre no espaço, numa nave espacial, criada através de CGI, mas onde se têm vários *lens flares* provenientes das luzes da nave.

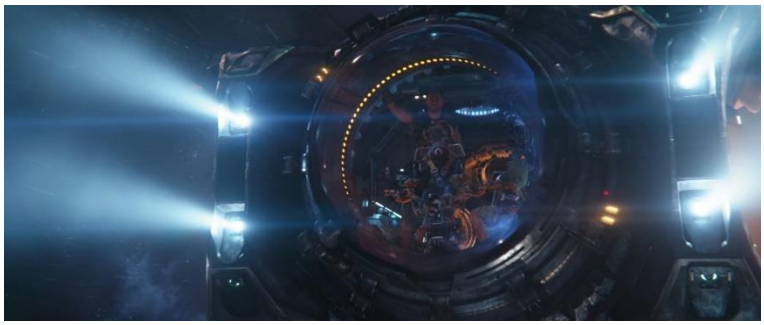

Figura 62 - Exemplo de *Lens Flare* artificiais no filme Avengers: Infinity War (Marvel, 2018) durante uma cena onde as personagens estão a navegar no espaço.

Sendo então este efeito comummente relacionado ao mundo do cinema atual, e com o mundo dos videojogos a possuírem cada vez mais cenas cinemáticas com o intuito de se aproximarem com o que acontece precisamente no mundo do cinema, a Unreal Engine disponibilizou este efeito como parte das suas ferramentas de post-Processing para que os criadores que utilizem a *engine* possam também criar estas cenas cinemáticas de alta qualidade, como se observa na **Figura 63**, até ao ponto em que hoje em dia, há inclusive cenas de televisão e cinema que recorrem à *engine* para sintetizar certas partes das suas criações devido à elevada qualidade que esta possui.

Figura 63 - Exemplo de *Lens Flare* no jogo Horzion Zero Dawn (Guerrilla Games, 2017).

Tenha-se, no entanto, atenção que o efeito presentemente no que toca a um videojogo, pode até ser indesejado, no caso de se querer, por exemplo que o videojogo dê a ideia de realismo a partir da perspetiva não de uma camara, mas sim de imersão em carácter, como se observa no caso da **Figura 64**, pois nesse caso, o olho humano não produz *lens flare*, com a exceção de este utilizar, por exemplo um capacete e daí haver uma quantidade de *flares* dependente da quantidade de camadas de proteção de vidro ou plástico do visor do capacete, ou, por exemplo o uso de óculos onde começa a haver cada vez mais um foco na venda de tratamentos de lente (dos óculos) para que estes não produzam *flares*, mas que nem sempre é evitável, especialmente se estes estiverem um pouco sujos ou engordurados.

Figura 64 - Exemplo do *Lens Flare* no jogo Battlefield 4 (Dice, 2013)

Para se conseguir então usufruir deste efeito deve-se ter em atenção aos seguintes componentes constituintes do painel relacionado ao *Lens Flare*:

- ***Intensity***

Como o próprio nome indica, a *Intensity* ou intensidade, controla precisamente a intensidade ou o brilho do *lens flare*, sendo que o valor de 0 desliga totalmente qualquer *lens flare*, e o 16 faz com que o *lens flare* ocorre sempre que este tenha oportunidade de acontecer.

- ***Tint***

Tint, coloriza o efeito de *lens flare*, dependendo da cor escolhida e de quão saturada esta é, ou seja, ao escolher um vermelho vivo, o *lens flare* é totalmente vermelho, se eu colocar a saturação a meio, o vermelho que agora é esbranquiçado acaba por dar uns tons avermelhados à cor natural do *lens flare,* e quanto mais esbranquiçada a cor for mais natural a cor do *lens flare* será, no entanto, se começar-mos a diminuir o valor, ou seja, a escurecer a cor, o *flare* começa a ser reduzido, podendo até ser desligado, misturando-se assim um pouco com a propriedade anteriormente referida da intensidade. Há também que ter noção que, dependendo da cor original do ponto luminoso, a cor do *flare* vai também ser alterada, ou seja, uma fonte de luz vermelha produz *lens flares* avermelhados, e uma branca *lens flares* esbranquiçados.

- ***Bokeh Size***

O *bokeh size* define o tamanho das formas de *bokeh* criadas pelo *lens flare*, sendo que valores de 0 desligam o *lens flare* mais uma vez, observe-se de como os valores do controlador influenciam os *flares* através da **Figura 65**.

Figura 65 - Comparação entre tamanhos de *bokeh* diferentes na Unreal Engine 4.26 (Imagem do autor).

- ***Threshold***

O *threshold* controla a quantidade mínima de brilho necessária para que um *flare* seja criado, sendo que valores mais baixos criam mais *flares* e valores mais elevados diminuem ou eliminam totalmente a quantidade de *lens flare* no ecrã.

- ***Bokeh Shape***

Como o nome indica, o *bokeh shape* controla a forma do *bokeh*, para uma que o desenvolvedor ou artista escolha, partindo de uma imagem, como demonstrado na **Figura 66**, que deve ser restritamente a preto ou branco, sendo branco a forma. Caso a imagem possua cor, por exemplo, vermelho em vez de branco, o *flare* sofre uma coloração também.

Figura 66 - Exemplo de como um *bokeh* pode ter formas variadas e até artificiais (que não dependem da quantidade de lâminas da lente) na Unreal Engine 4.26 (Imagem do autor).

- ***Tints***

A última configuração possível ao *lens flare*, encontra-se em *Tints*, uma categoria com 8 controladores, cada um responsável por colorir um dos *flares*, individualmente. Podendo assim ajudar a melhorar o look do *lens flare*, no caso, por exemplo o desenvolvedor querer apenas ter dois *flares* em vez dos oito que a *engine* cria, este pode designar que seis dos *flares* têm a cor preta, assim fazendo com que apenas dois dos *flares* sejam adicionados.

Image Effects

Estando quase a chegar ao fim, o penúltimo *item* da lista consiste na verdade em dois itens, *Vignette* ou vinheta e *Grain* ou grão.

- ***Vignette***

O efeito de *vignetting* ou de vinheta é uma ocorrência comum na fotografia até nos dias de hoje, e o resultado desta ocorrência é uma imagem onde os cantos são ligeiramente mais escuros, como o que acontece na **Figura 67**. Normalmente tem a forma exterior de um círculo e pode ser subtil ou extremamente notável, isto devido a uma quantidade variada de origens, sejam essas mecânicas, proveniente normalmente dos acessórios montados à lente da câmara, podem ser de origem ótica, e neste caso, o *vignetting* ocorre naturalmente e é impossível de remover devido à sua relação com a forma como a luz passa na lente sendo gravada no sensor, e por fim pode ser do píxel, onde este está mais relacionado à forma como o ângulo da luz entra nos sensores dos pixeis das câmaras digitais (Mansurov, 2013).

Figura 67 - Exemplo de um efeito extremo de *Vignetting* numa foto (s/i).

De modo a corrigir estes problemas, imensas tecnologias e formas surgiram para que este efeito não aconteça, e inclusive alguns *softwares* de tratamento de imagem apenas precisam que o utilizador coloque o a câmara e lente para que o *software* resolva o problema automaticamente.

No entanto, o efeito pode também ser utilizado como uma forma criativa e artística, muitas das vezes servindo para colocar o foco da atenção de quem vê a nossa imagem ou cena, escurecendo (como consta na **Figura 68**) ou totalmente obstruindo detalhes nas extremidades da imagem que iriam distrair o espetador.

Figura 68 - Exemplo de um *vignetting* subtil e eficaz, que concentra a atenção nas flores (s/i).

No caso da Unreal Engine, tem-se apenas um controlador para o efeito de vinheta, e este, em contraste com o que pode ser criado noutros *softwares* ou o que ocorre com certas lentes e câmaras, não consegue ser extremamente exagerado, como demonstrado na **Figura 69** (a não ser que o utilizador utilize valores acima dos predefinidos como máximo pela *engine*) e como tal é fácil de ver como este pode ser rapidamente utilizado precisamente para focar o espetador ou jogador no centro do ecrã.

Figura 69 - Comparação entre o uso de *vignetting* ou não, na Unreal Engine 4.26 (Imagem do autor).

- ***Grain***

Outro efeito que a *engine* permite adicionar à cena é o grão que é mais uma das ocorrências ou artefactos comummente vistos em fotografia e cinema, especialmente se este não for recente, se for proveniente de câmaras de baixa qualidade e de gravações realizadas em condições menos ideais, entre outras fontes variadas, um exemplo de grão pode ser visto, embora subtilmente no filme Underworld (Len Wiseman, 2003), demonstrado em pormenor na **Figura 70**. Este é também relacionada à película de filme e como a maioria destas películas produziam grão devido a uma multitude de razões, mas normalmente associada à luz e à velocidade de gravação. Hoje em dia o efeito tem vindo a ser eliminado gradualmente, primeiro devido à mudança de película para fotos digitais, e também com o aumento tecnológico em volta dos componentes constituintes das câmaras dos dias de hoje. No entanto, se pegar numa câmara atual, e aumentar o ISO da mesma, o grão aparece novamente, e quão maior o ISO for, mais notável este será.

Figura 70 - Pormenor do grão ou ruído existente no filme Underworld (Wiseman, 2003).

Como tal, este é um efeito que a maioria dos cinematógrafos e fotógrafos de outros tempos tentavam ao máximo reduzir de modo a criar um trabalho profissional, no entanto, presentemente o efeito tem vindo a começar a ser utilizado para servir vários finais, desde tentar emitir as câmaras de baixa qualidade dos tempos passados, servir apenas como nostalgia, ou até tentar induzir a ideia de vida, ou de alucinogénio visto que este está sempre em movimento constante, mas de uma forma aleatória.

Dentro da Unreal Engine este efeito possui dois controladores, nomeadamente, *jitter* e *intensity*, e embora o primeiro, tenha vindo a criar problemas, e como tal, esteja constantemente ligado, ou seja, qualquer grão que se aplique à cena, estará em constante movimento,

o segundo, *intensity,* como é expetável pelo nome, controla a intensidade do grão que aparece na cena. Um exemplo de utilizar grão ou não numa cena pode ser constatado na **Figura 71**.

Figura 71 - Comparação entre o uso ou não de grão na Unreal Engine 4.26 (Imagem do autor).

Depth of Field

Baseado no efeito real, quer nas câmaras fotografias, vídeo, e ainda na visão humana, dependendo da distância, e o ponto focal, o resto tem tendência a ficar desfocado. A este efeito dá-se o nome de *depth of field* (DOF ou Profundidade de Campo), este consiste na distância entre os objetos mais próximos e os mais distantes numa foto ou no olhar, deixando os objetos em foco nítidos, e os outros, cada vez mais desfocados mediante a distância que estes têm em relação ao ponto focal. No entanto, ao contrário do que acontece com o olho humano, este acontecimento pode ser controlado, de maneira a conseguir controlar o ponto a partir de que o desfoque começa a acontecer, podendo inclusive chegar ao ponto de que tudo parece nítido e nada desfocado, um tópico conhecido como Distancia Hiperfocal.

Deste modo, quando maior for o *Depht of Field*, maior será a parte da imagem nítida em contraste com a desfocada, assim como o inverso, onde um *Depth of Field* pequeno irá causar a uma parte maior da imagem a estar desfocada, em alguns casos causando até que esta se desfoque.

Na Unreal Engine 4, há vários métodos de utilizar este efeito de *Depth of Field*, onde os desenvolvedores separam em duas categorias, cinemático e *mobile*, sendo que a primeiro

oferece uma aparência cinematográfica onde os ajustes são semelhantes às opções de câmaras comuns disponíveis em fotografia e cinematografia, sendo que esta opção funciona para consolas e computadores, a segunda oferece opções de *Depth of Field* otimizadas e de baixo impacto no *hardware*, que funcionam para plataformas móveis como telemóveis e computadores portáteis com fraco *hardware*. No caso desta dissertação iremos apenas focar nos controladores da versão normal, ou de computador / consola.

Antes de seguir com a explicação de cada controlador uma das práticas recomendadas é ligar a camada de visualização de profundidade de campo, de movo a ver melhor o que está desfocado e o que faz parte do foco da imagem. Para isso segue-se a hierarquia *Show — Visualize — Depth of Field Layers* (**Figura 72**). Ao ativar esta camada, note-se que tudo o que é apresentado a verde é considerado como perto, a azul longe e o preto conta com o que é considerado ser o foco, para alem das *layers* é importante saber que as configurações da câmara colocadas previamente, têm uma influência aqui, nomeadamente a ***Aperture***.

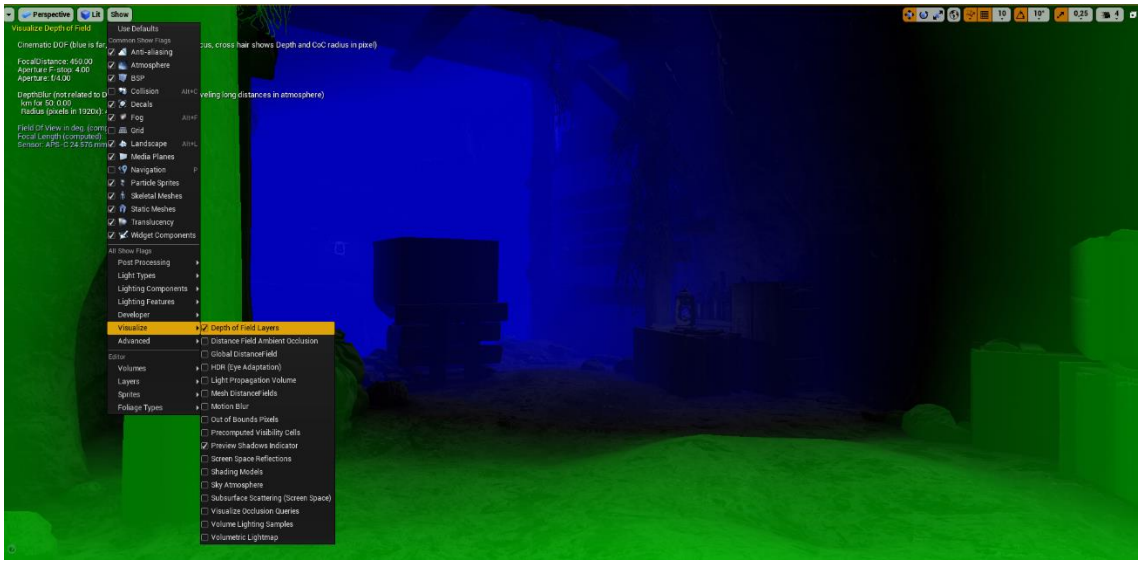

Figura 72 - Demonstração de como ligar *Depth of Field Layers*, assim como o resultado de as ativar. (Imagem do autor)

- ***Focal Distance***

No primeiro ponto do *Depth of Field* define-se a distância focal, em centímetros, e através das *Depth of Field Layers*, pode-se facilmente visualizar onde este ponto focal é, relembrando que, um valor de 100 corresponde a 1 metro.

- ***Depth Blur km for* 50%**

O segundo ponto define a partir de que distância, os pixeis começam a ser desfocados, no entanto, este valor corresponde a quilómetros, sendo que os valores mais recomendados sejam perto do 0,001, ou seja, 1 metro.

- ***Depth Blur Radius***

O último ponto define o raio do desfoque em pixels, em relação a uma resolução de 1920x1080 pixéis, neste caso, valores inferiores a 3 tendem a que a cena não possua desfoque focal.

Color Grading

Na sua essência, *color grading*, é o processo de melhorar o aspeto de uma determinada imagem, algo, feito, de forma estandardizada, desde meados dos anos 60.

Atenção que, como temos vindo a ver até agora, embora o *color grading* seja considerado como uma ferramenta de melhorar o produto ou imagem final, como é exemplificado na **Figura 73**, não quer dizer que não possa efetuar o oposto, alias, é bastante simples deixar-se levar pela criatividade não restrita ou pela falha de conhecimento e fazer com que a imagem final seja pior que a original, efetivamente estragando qualquer trabalho que a imagem original possuía, fosse esse muito ou pouco, bom ou medíocre. Para além disto, chama-se também à atenção de que as vezes, *color grading* é falado como *color correction* e vice-versa, efetivamente misturando os dois processos, mas isto é errado, embora estes de certo modo possuam ferramentas semelhantes, o resultado pretendido em cada um difere como veremos de seguida.

Figura 73 - Exemplo de uma cena antes e depois de *color grading* (s/i).

Começando então com color *correction*, o termo refere-se ao processo de corrigir a cor, luz, contraste e afins, da imagem, de maneira que esta coincida com a ideia do realizador na altura da gravação, e/ou com as cores originais no cenário, antes de a câmara, alterar a mesma. Exemplo disto é o resultado de, utilizando o próprio telemóvel, fotografar o que vemos agora à nossa frente, e comparar o resultado da foto com o que vemos, onde, regra geral, há diferenças, cores claras que acabam por ficar totalmente brancas, superfícies que por serem brancas e estarem iluminadas perdem o detalhe e são agora uma mancha branca, ou uma imagem quase preta, visto estar muito escuro. Isto tem a ver com o facto de o olho humano, ser, de um modo figurativo, a lente ou câmara fotográfica/vídeo mais potente que conhecemos, não só na resolução, mas também como esta funciona na captação de cor, exposição automática, etc., com exceções como daltonismo, ou problemas oculares (normalmente compensados com óculos, lentes ou cirurgias). Exemplo disto pode ser observado quando num concerto noturno, onde uma câmara precisa de vários ajustes para conseguir realizar uma captura com um detalhe semelhante ao presenciado pessoalmente, e ainda assim é comum haver problemas de ruído devido a um ISO muito elevado, ou movimentos arrastados para compensar a falha de luz (estes efeitos podem ser reduzidos, dependendo da câmara e da sua qualidade, dai concertos gravados para DVD ou TV que utilizam câmaras de valores muito elevados, acabam por conseguir reduzir bastantes defeitos de filmagem). Como tal, este processo costuma e deve ser feito antes de se seguir com o *color grading*. Ignorar *color correction* e partir para *grading* seria como realizar um desenho, a cores, numa folha cor-de-rosa, o que ia causar que todos os brancos do desenho ficassem cor-de-rosa, criando algo estranho, o *color correction* neste caso corresponderá a, ou trocar a folha por uma branca, ou fornecer ao artista um lápis branco de modo que este pudesse pintar todos os pontos do desenho que deveriam ser brancos, com o lápis branco, corrigindo assim a imagem.

Já o termo *color grading*, serve para criar emoções e atmosfera numa determinada cena, de maneira que esta possa induzir o espetador a sentir algo, este processo está normalmente ligado à teoria da cor, por exemplo, o vermelho é comummente associado a raiva, paixão, desejo, excitação, velocidade, força, poder, calor, amor, fogo, sangue, violência, guerra, energia, entre outros, mas o roxo ou o violeta, também costumam estar relacionados com poder, erotismo, intimidade e sensibilidade, assim como o cor-de-rosa pode ser associado com amor, romance e charme (Fusco, 2016), deste modo, se um autor de uma foto, filme ou jogo, tivesse o intuito de criar uma história romântica, possivelmente até erótica, alterará as cores da imagem original de maneira que esta se aproxime de tons de vermelho, roxo, violeta e cor-de-rosa, induzindo assim o espetador a sentir estas emoções, para além do que se passa no contexto da imagem. Mais uma vez, realizar sempre estas tonalizações com alguma restrição, de modo que a imagem continue a possuir um aspeto realista (caso seja o aspeto desejado), pois uma imagem levada ao extremo começa a perder o efeito desejado e torna-se até incomodativo e é claro, esta ideia, sempre que possível, deve partir do início da

criação, ou seja, neste exemplo, tentar que, quando se filmassem as cenas originais, que os moveis e roupa tivessem tons avermelhados, porque se estes fossem todos cada um da sua cor, muito vivos, quase como se fossem todos brinquedos, o resultado sofre um impacto relevante, que pode ser complicado ou impossível de reverter.

Assim sendo, o *color grading* possui vários atributos que podem ser melhorados ou alterados numa imagem, tais como, o contraste, a cor, a saturação, o detalhe, os níveis de pretos e brancos, exposição, entre outros, como é demonstrado na **Figura 74**, onde o ambiente original se torna num ambiente mais frio e escuro, alusivo até à noite, e onde cores que eram avermelhadas, se tornam amareladas ou esverdeadas.

Figura 74 - Exemplo comparativo entre uma cena na Unreal Engine 4.26, antes e depois de levar *color grading* (Imagem do autor)

Resumidamente, *color correction* é um processo técnico que corrige problemas de cor no intuito de a imagem parecer o mais natural possível, como se fosse visto pelo olho humano, já *color grading*, é um processo técnico criativo, usado para adicionar emoções, atmosfera, etc. à imagem, mesmo que o resultado não seja natural.

Retrocedendo, foram referenciadas algumas das propriedades que costumam fazer parte do *color correction* e do *color grading*, mas no caso da *unreal engine*, quais são as propriedades que estão disponíveis e o que cada uma delas faz?

- ***White Balance***

A primeira categoria na *engine* é o *white balance*, ou seja, o balanço de brancos da cena, e este, é comum ser utilizado mais para corrigir problemas com as cores do que dar propriamente um tom criativo à cena, como tal é mais considerado que este faça parte do *color correction* do que do *color grading*, mas sem exclusividade de um ou do outro.

Temp

Dentro do White Balance o controlador Tempo serve para controlar a temperatura da luz, que por si, varia entre laranjas ou cores quentes, e azuis ou cores frias, a maneira mais fácil de entender o alcance do controlador é a partir da **Figura 75** e **Figura 76**, mas, vendo-o invertido, visto que no caso do controlador da *Unreal Engine*, quanto mais baixo o valor, mais fria a temperatura em vez de mais quente. Em suma, quanto menor o valor, mais fria e azulada a cena parece, e quanto maior o valor, mais quente e avermelhada.

Figura 75 - Ilustração de como temperaturas de cor diferentes, em Kelvin, produzem cores diferentes, assim como algumas fontes que produzem essas luzes (s/i).

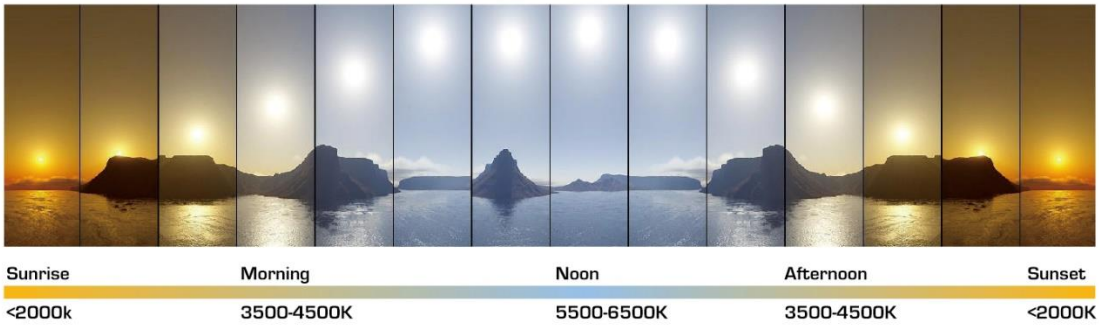

Figura 76 - Ilustração de como a temperatura da luz solar (e lunar) corresponde aos momentos do dia (s/i).

Tint

O controlador de *tint*, em semelhança com o anterior, altera o tom da cena entre dois tipos de cor, mas, neste caso, em vez de variar entre azuis e laranjas, este varia entre verdes e magentas. Regra geral o *tint* é mais reservado para corrigir problemas causados com a iluminação de uma cena, em que este seja muito esverdeada, por exemplo, por motivos variados, e ao movimentar o controlador para o lado da magenta, a cena poderá começar a ficar com uma cor mais natural, parecida com branco. Não quer, no entanto, dizer que não possa ser utilizado para fins criativos, embora, de um certo modo seja melhor realizar essas alterações mais criativas com os componentes seguintes.

- ***Global***

Como o próprio nome indica, o Global serve para controlar a cor globalmente, sem ser demasiado específico nas sombras ou iluminação, parâmetros estes que veremos a seguir. No entanto, no parâmetro global, assim como nos parâmetros *Shadows, Midtones* e *Highlights*, há algumas opções que podem ser controladas.

Saturation

Saturation, ou, saturação, refere-se à cor e a sua pureza, como tal, ao ajustarmos este ponto, as cores da nossa cena serão mais puras, ou vivas, ou, pelo contrário, cada vez menos vivas, podendo no limite, ficar a nossa cena a preto e branco. Em semelhança aos controladores que encontramos em *Saturation*, estes são iguais quando falamos de *Contrast, Gamma* e *Gain*. Ou seja, temos primeiramente um círculo de cor onde podemos ajustar a cor e o quão desejamos que esta seja mais clara ou menos. De seguida um controlado que influencia a intensidade, que neste caso é o principal controlador do efeito, que ao ser alterado faz com que a cena fica com mais saturação (cores mais vivas) ou sem saturação (pretos e brancos). Do lado direito podemos ainda escolher entre RGBY e HSVY, para melhor ajustarmos a cor escolhida, algo que neste caso tem um impacto limitado e como tal é recomendado deixar que os valores se mantenham em RGBY e todos a 1.

Contrast

Ao entrar no campo de *Contrast*, vê-se que como referido previamente, os controlos são iguais aos de *Saturation*, sendo então a única diferença, no resultado obtido ao mexer nestes, visto que, agora, ao, por exemplo escolhermos um tom laranja, as partes mais iluminadas da nossa cena tornam-se alaranjadas, mas, as sombras, começam a ficar mais azuladas, visto que, neste caso, o azul seria a cor oposta ao laranja. Como tal, o *Contrast* ou contraste em português, afeta duas tonalidades na nossa cena, uma dependente da outra, a iluminação e as sombras.

Gamma

Gamma controla os tons médios de uma cena, e, ao alterarmos os valores dos controladores neste ponto, a nossa cena irá aproximar-se de uma certa tonalidade de cor, muito semelhante ao que se fazia com o *Tint*, no White Balance, mas que agora, a opção de cor é mais ampla, e não apenas restrita a verdes e magentas.

Gain

No caso do *Gain*, a ideia é controlar os pontos mais luminosos da cena, no entanto, é necessário ter algum cuidado pois, este pode afetar também as sombras, criando assim a necessidade de ir fazendo um balanço entre os campos *Gamma, Gain* e *Offset*.

Offset

Finalmente, entramos no último campo, Offset, este que corresponde ao controlo das sombras da cena.

- ***Shadows, Midtones* e *Highlights***

Continuando com o esquema usado no ponto Global, cada um destes três pontos possui outros 5 dentro de cada, mas o importante a saber agora é que estes são complementares ao anterior, servindo para polir possíveis pontos do *color grading* que não tenham ficado 100% como desejado, por exemplo, se toda a cena estivesse com a tonalidade e correção desejada, exceto a iluminação, onde o ideal era esta ter tons avermelhados, mas está com tons amarelados, ao entrar no ponto *Highlights* e ajustar os controladores no mesmo, podemos então levar esses tons amarelados para os avermelhados que queremos.

HighLights Min

No caso das propriedades das *Highlights*, ou seja, as zonas mais iluminadas da cena, há um controlador extra que *Shadows* e *Midtones* não possuem, o *HighLights Min*, que controla as propriedades do que é definido no painel dos *Highlights* e controla o alcance que os *Highlights* abrangem, indo de -1 onde a imagem é toda considerada como *Highlights* a 1 onde só as partes mais extremamente iluminadas é que sofrem alterações.

- ***Misc***

Finalmente chega-se ao último painel do *Color Grading*, o *Misc*, este serve para alterar várias propriedades da cena que não tenham sido alteradas anteriormente, ou para aperfeiçoar mais minuciosamente as anteriores, dentro desta categoria encontram-se então os controladores:

Blue Correction

O controlador *Blue Correction* permite realizar algumas alterações a todas as tonalidades de azul da cena, tornando-as mais azuis ainda com o valor a 0 ou mais esbranquiçadas no caso de o valor estar a 1.

Expand Gamut

Este controlador expande cores que estejam muito saturadas que se encontrem fora da game sRGB ara falsificar uma sintetização de gama ampliada. É, no entanto, um efeito quase impercetível, dependendo do ecrã onde se realiza o *grading*.

Tone Curve Amount

O *Tone Curve Amount*, controla os efeitos criados pela curvatura de tons das cores, reduzindo o contraste e aumentando a quantidade de tons da imagem, de modo que esta não tenha transições de tons tão extremos.

Scene Color Tint

Por fim, reservando os dois controladores referentes a LUTS para mais à frente, o controlador de *Scene Color Tint* fornece ao utilizador mais uma oportunidade de converter o tom geral de uma cena relativamente à cor escolhida.

***Lookup Tables* (LUTs).**

Previamente falou-se de *color grading* e como este podia melhorar a imagem, fornecendo cores, contrastes e afins mais corretos, e até proporcionando emoções à cena. No entanto, o processo de aprender a trabalhar e usar corretamente estas técnicas, pode ser bastante longo e complicado.

É assim introduzido o termo LUT (*Lookup Table*), um termo usado para descrever uma matriz predeterminada de números que fornecem uma maneira mais rápida e fácil para cálculos específicos, que, no contexto de *color grading*, transforma os valores das cores introduzidos nos valores desejados, proporcionando assim uma forma mais fácil e rápida de realizar o *color grading* de uma cena, através de uma imagem ou aparência criado por um artista previamente, uma espécie de modelo pré-feito. No entanto deve a imagem à qual o LUT se aplica, que já tenha o *color correction* feito tal e qual como vimos previamente no que tocava ao *color grading*, senão, podem surgir problemas onde as cores não correspondam ao esperado.

Geralmente, há dois tipos de LUTs, unidimensionais (1D) e tridimensionais (3D):

- LUTs 1D (**Figura 77**) são assim conhecidos devido a serem controlados por um único valor, e, podem ser reduzidos à predefinição de uma curva de *gamma*. Este tipo de LUT consegue atingir o objetivo de um LUT, sendo o único problema, que não oferece tanto controlo como outros.

Figura 77 - LUT 1D, capaz de controlar o valor de *gamma* de uma cena (Imagem do autor)

- Já os LUTs 3D (**Figura 78** e **Figura 79**) controlam a matiz, a saturação e o brilho, de maneira que haja muito mais controlo sobre valores de cor específicos. Este é o caso do tipo de LUTs utilizados pela Unreal Engine 4.

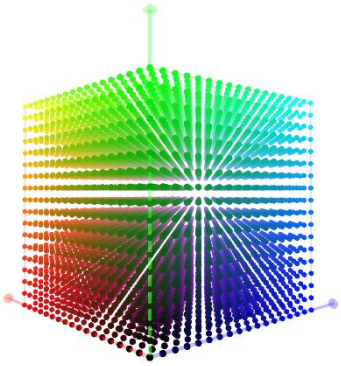

Figura 78 - Representação de um LUT 3D inalterado, onde cada ponto equivale a um pixel (s/i).

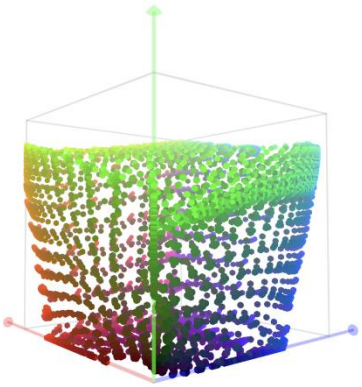

Figura 79 - Representação de um LUT 3D após este ser alterado (s/i).

Falando então no tipo de LUT 3D, como referido o utilizado na Unreal Engine 4, é essencialmente um conjunto de 16 quadros de 16×16 com uma gama de cores, que no caso de um LUT neutro (**Figura 80**), ou seja, um LUT que não altera a imagem original, que contém várias cores, desde vermelhos, verdes, azuis, roxos, amarelos, pretos e brancos, mas que pode depois ser alterado para, por exemplo, esbater as cores e dar um tom mais sépia à cena (**Figura 81**Figura 82), resultando num ficheiro que no fim tem um tamanho total de 256×16 pixeis (**Figura 82**), ou imaginando um espaço 3D, um cubo de 16×16x16 pixeis.

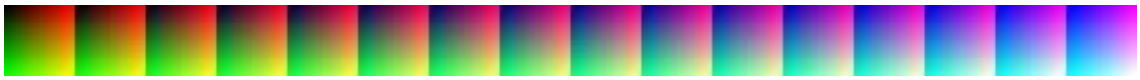

Figura 80 - LUT 3D neutro como é retirado da Unreal Engine 4.26 (Epic Games, 2020).

Figura 81 - Comparação entre uma cena antes e depois de ter um LUT 3D alterado aplicado, na Unreal Engine 4.26 (Imagem do autor).

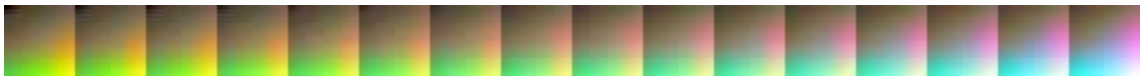

Figura 82 - LUT 3D alterado de modo a criar imagens sépia (Amplify Creations, 2016).

Assim sendo pode-se considerar inclusive que um LUT é um filtro rápido para mudar as cores de uma imagem, tanto saturação como contraste, luminosidade, tom etc. semelhante ao que se observa atualmente em várias aplicações de tonalização de fotos como, por exemplo as histórias do Instagram e do Facebook, que com um rápido deslize do dedo antes de finalizar um *upload*, o utilizador pode aplicar tonalidades e cores diferentes das originais.

Agora falando especificamente em termos da Unreal Engine 4, esta não tem muito que se lhe diga, e na aplicação e utilização é um dos efeitos de post-Processing mais simples, isto porque possui apenas dois controladores, a intensidade, que controla quão grandes as alterações à cena vão ser mediante o LUT introduzido, e qual é o LUT que se utiliza, sendo tão simples como arrastar do *Content Broswer*, o LUT pretendido para a caixa.

Um pouco mais complicado, mas ainda assim fácil de realizar, é a criação de um LUT. Neste caso há imensas maneiras de criar um LUT para aplicar à nossa cena, mas possivelmente uma das mais simples consiste em pegar captura de alta qualidade da cena (**Figura 83**), e utilizando o editor de imagens preferido, como, por exemplo, o Adobe Photoshop, aplicar os filtros desejados, mas sempre em *layers*, e após se obter o look pretendido, pegar nestas *layers*, e arrastar para um ficheiro LUT neutro, finalmente guardar o LUT com as

alterações aplicadas e está concluída a criação de um LUT. Um passo, no entanto, que pode levar a um resultado superior sem criar muito mais trabalho é usar *software* que converta imagens em LUTs, e, a partir novamente de uma captura de tela de alta qualidade, após o *color correction* e *tone mapping* estarem aplicados, converter a captura num LUT, e seguir como se mencionou previamente, abrir a imagem no editor, criar os ajustes e arrastar os ajustes para este LUT neutro, mas com a cor e o tom já corrigidos da nossa cena.

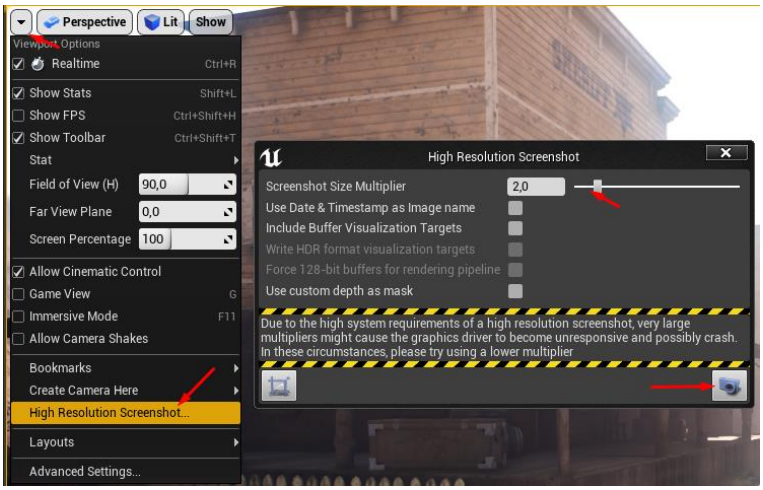

Figura 83 - Ilustração de como retirar uma captura de tela de alta resoulão na Unreal Engine 4.26 (Imagem do autor).

Film

Tone mapping, *filmic tonemapper* ou como é apresentado na Unreal Engine apenas como *Film*, tem o objetivo de converter uma ampla gama de cores de alta faixa dinâmica, conhecida como uma imagem HDR (*High Dynamic Range*) numa imagem com uma faixa dinâmica baixa, conhecida como LDR (*Low Dynamic Range*) de maneira que a imagem consiga ser reproduzia corretamente numa ampla gama de monitores, sejam estes num ecrã de telemóvel ou de televisão, independentemente da tecnologia que estes ecrãs usem, ou seja, não interessa se são CRTs, LED, OLED entre outros, obtendo assim, uma imagem com uma tonalidade de cores muito semelhante nos vários ecrãs, sem problemas como uma imagem ter uma boa visualização num ecrã, e má em todos os outros.

Com isto, é prática comum que este seja o primeiro processo a ser realizado quando se começa a trabalhar no post-Processing de uma cena, podendo quando muito ser apenas superado pelo Anti-Aliasing, de maneira que quando se comece a realizar as alterações criativas, que a imagem esteja corrigida em termos de cor, como se falou previamente no que toca ao *color correction*. É também pratica comum no caso dos videojogos, que todos os níveis e cenas possuam o mesmo *tonemapping* e *color correction* antes de se partir para qualquer outra categoria de post-Processing, de forma que a base seja igual para todos.

Antes de continuarmos a analisar o *tonemapping* dentro da Unreal Engine, note-se que, assim como o *color grading*, é recomendando que o ecrã onde se realizam estas duas alte-

rações que sejam o melhor possível no que toca à cor, contraste e iluminação, nomeadamente, uma profundidade de cor de 10 ou 12 bits em vez do tradicional 8 bit, e 300 nits de iluminação (*brightness*), sendo até altamente recomendado que estes possuam tecnológicas como HDR10, HDR10+ ou preferencialmente, Dolby Vision.

- ***Slope***

O controlador *Slope* ajusta a inclinação da curva S usada para mapear os tons da imagem, e como tal, valores maiores, fazem com que as imagens possuam um contraste maior, parecendo mais viva, e valores menores começam a aproximar-se de uma imagem mais acinzentada, menos saturada e com menos contraste, onde, no caso de o valor ser 0, a imagem fica totalmente preta.

- ***Toe***

O *Toe* controla as cores escuras da cena, como tal, valores superiores aumentam a quantidade de zonas escuras na cena, podendo fazer com que estas fiquem totalmente pretas, valores inferiores reduzem as zonas pretas e escuras da cena, sem as remover totalmente, mas fazendo com que o contraste destas seja menor também, revelando assim mais detalhes nas zonas escuras da cena, mas tornando-as mais acinzentadas e desaturdas.

- ***Shoulder***

Ao contrário do Toe, este controlador mexe com as zonas luminosas da cena, dando a estas, com um valor menor, um ar mais acinzentado, ou até enublado, e com valores maiores, fazendo com que estas fiquem mais esbranquiçadas e até ofuscantes e sobressaturadas.

- ***Black Clip***

Este controlador controla o momento em que o cruzamento acontece, e como tal, o preto começa a sobrepor-se a várias cores e pontos da cena, semelhante ao que o Toe efetua, mas começando logo a sobrepor-se às zonas mais escuras e queimando muita da cor à volta das mesmas. Normalmente, este valor não é alterado, mantendo-se no 0.

- **White Clip**

Em semelhança ao que foi dito no *Black Clip*, este segue o mesmo efeito, mas desta vez para os pontos luminosos, como é o caso do *Shoulder*, este efeito, no entanto, é bastante subtil e pode até ser impercetível quando alterado.

Rendering Features

A última categoria que se encontra no painel de *post-Processing* é relacionada aos efeitos de sintetização, onde, alguns têm inclusive pouco ou nada a ver com o termo de *post-Processing*, mas sim mais com a maneira como a *engine* realiza uma sintetização, como é o caso

de todas as categorias relacionadas a *Ray Tracing*, que não serão analisadas devido a merecerem um pequeno artigo à parte de modo a não só explicar as diferenças dos modos de sintetização, vantagens e desvantagens, necessidades de hardware entre outros pontos.

Sendo assim, os filtros excluídos são:

- *Ray Tracing Ambient Occlusion*
- *Ray Tracing Global Illumination*
- *Reflections*
- *Ray Tracing Reflections*
- *Translucency*
- *Ray Tracing Translucency*
- *Path Tracing*

Post Process materials

Os *Post Process Materials* são provavelmente os mais complicados de entender, em como são criados, e como os utilizar, honestamente este ponto merecia um documento exclusivo para realmente se poder explicar como usufruir deste, mas no que toca ao perceber o que este efetua, a tarefa torna-se algo mais simples, desde que não se tente aprofundar demasiado, aí a explicação de como este funciona seria bastante mais complicada e novamente será merecedora de um documento exclusivo para si.

Na essência, estes são filtros criados a partir do editor de materiais (e dai que estes são categorizados precisamente como materiais) que ao serem aplicados à cena podem alterar o look da mesma drasticamente, por exemplo, transformando o aspeto de uma cena que até ao momento é realista, para a de um desenho animado, ou um desenho técnico, como é exemplificado na **Figura 84**. A quantidade de filtros que podem ser aplicados é também elevada, mas há que ter algum cuidado com as compatibilidades entre eles, visto que alguns filtros em vez de se misturarem com outros, vão sobrepor-se, e eliminar os efeitos anteriores.

Figura 84 - Exemplo demonstrando um antes e depois da uma cena com um *Post Process Material* aplciado (Imagem do autor).

Há que denotar que *Post Process Materials* atualmente é uma categoria de post-Processing restrito à Unreal Engine em como este é utilizado e aplicado, no entanto, outros *softwares* podem realizar finais semelhantes, tanto em *game engines* como editores de fotografia, edição de filme etc. Por exemplo, a Unity (Unity Technologies, 2005) possui *Shader Graphs*, que, são um exemplo bastante semelhante ao utilizado na Unreal Engine.

Ambient Cubemap

Este efeito serve para adicionar à cena mais um pouco de luz, proveniente de uma imagem conhecida como um HDRi ou textura *Cubemap*, e para além de adicionar alguma iluminação, vai também adicionar mais detalhes aos reflexos de certos objetos, fazendo com que possa ser um filtro capaz de adicionar ou remover um realismo superior, como se observa no exemplo da **Figura 85** em que ao adicionar este efeito, a cena ganha mais um pouco de iluminação, especialmente em zonas que antes estariam mais escurecidas, assim como uma pequena variação no tom da iluminação, no entanto, esta é refletida na esfera onde se nota o reflexo de algumas árvores, que na cena não existem visto esta estar situada num deserto.

Figura 85 - Exemplo de uma cena na Unreal Engine 4.26, onde, os reflexos correspondem a uma floresta verde, e não ao *Wild West* da cena (Imagem do autor).

Mas caso o HDRi utilizado fosse mais semelhante ao ambiente utilizado na cena, neste caso algo no género dos *westerns* americanos, com deserto e alguns salões, este reflexo já não seriam incomodadores.

- ***Tint***

O primeiro controlador serve para adicionar um tom de cor ao *cubemap* que se escolherá de seguida, servindo como controlo adicional caso a iluminação proveniente do *cubemap* não corresponda a 100% à desejada, e habilitando assim que o desenvolvedor possa adicionar um tom, por exemplo, sépia, à iluminação.

- ***Intensity***

A *intensity* permite, mais uma vez, como tem vindo a ser costume, definir qual é o impacto que o *cubemap* tem na cena, especialmente na iluminação e de reflexos como se pode constatar na **Figura 86.**

Figura 86 - Comparação entre intensidades diferentes na Unreal Engine 4.26 (Imagem do autor).

- ***Cubemap Texture***

Por fim, temos a opção de escolher o nosso HDRi, este sendo proveniente, portanto de um ficheiro de imagem, normalmente com a extensão de HDR ou EXR, ficheiros estes que conseguem armazenar várias exposições da mesma foto ou imagem numa só. No exemplo da **Figura 87** vemos que ao alterar o *cubemap* para o HDRI da **Figura 88**, a cena acaba por ficar um pouco mais realista, visto este estar mais conforme o ambiente em que a cena do jogo se encontra. Em especial nota-se que no reflexo da esfera, já não existem árvores ou elementos estranhos à cena.

Figura 87 - Exemplo onde a cena anterior, possui agora um reflexo correto, semelhante ao que é considerado como *Wild West*, na Unreal Engine 4.26 (Imagem do autor).

Figura 88 - HDRi utilizado na cena anterior (Figura 87) de modo a alterar os reflexos do *Ambient Cubemap* de modo a refletir algo como o *Wild West* (s/i)

Ambient Occlusion (AO ou SSAO de *Screen Space Ambient Occluion)*

Em português, oclusão ambiental, é um filtro que adiciona à cena uma profundidade superior e de mais qualidade no que toca a sombras, adicionando então sombras à cena, em sítios onde previamente não havia, ou, onde estas eram demasiado subtis ou escassas, como se consegue observar na **Figura 89**.

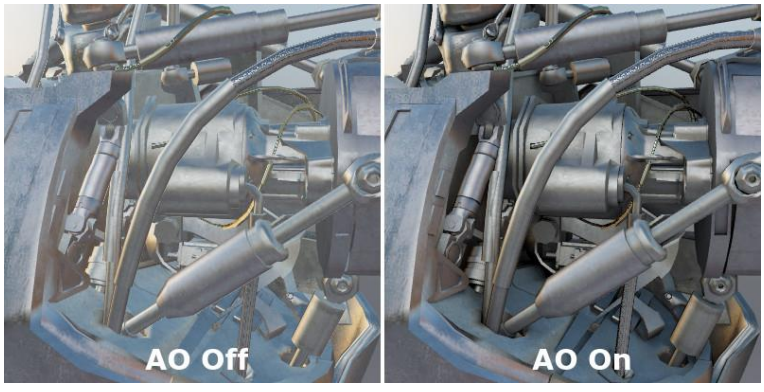

Figura 89 - Exemplo de oclusão ambiental ligada e desligada (s/i).

No caso da Unreal Engine este filtro possui apenas dois controladores.

- ***Intensity***

O primeiro controlador é responsável pela quantidade de oclusão ambiental que a cena possui, variando entre ligado e desligado, com algumas suavizações de cada no meio caso o artista decida que totalmente ligado ou desligado seja demasiado, uma comparação entre estes valores pode ser observada na **Figura 90**.

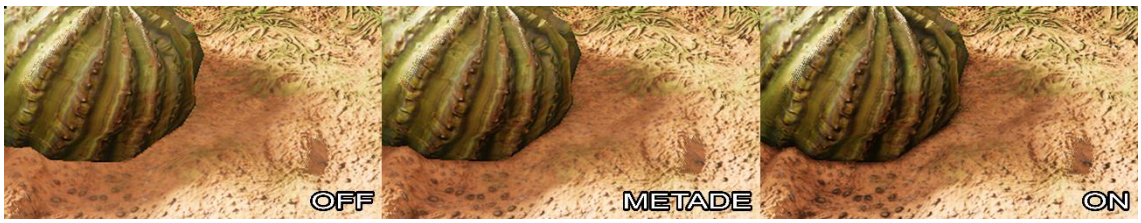

Figura 90 - Comparação de valores diferentes em termos de intensidade de oclusão ambiental na Unreal Engine 4.26 (Imagem do autor).

- ***Radius***

O segundo controlador, define então o tamanho que a oclusão tem, e este pode melhor ou totalmente quebrar o realismo e o efeito numa cena, isto porque, um valor demasiado pequeno, cria uma oclusão muito fina, quase inexistente, como se houvesse apenas uma linha delimitadora onde a oclusão existe, mas um valor elevado pode criar uma sombra tão ampla que acaba por não se ter basicamente oclusão. Para verificar a comparação entre estes valores veja-se o exemplo na **Figura 91.**

Figura 91 - Comparação de valores de raio diferentes na oclusão ambiental na Unreal Engine 4.26 (Imagem do autor).

Global Illumination (ou *Screen Space Global Illumination*, SSGI)

Iluminação global, ou em alternativa, iluminação indireta, é um método de iluminação que embora não consiga fornecer iluminação a 100% a uma cena, e como tal fugindo um pouco ao que o nome indica, pode, para além da iluminação já existente numa cena, adicionar mais um pouco de iluminação, de modo que esta fique mais completa e como tal apresentar melhores resultados.

Este filtro possui apenas dois controladores, um correspondente à cor da luz que ilumina a cena (**Figura 92**), e o segundo que controla a intensidade da iluminação (**Figura 93**).

Figura 92 - Comparação entre cores de iluminação global diferentes, na Unreal Engine 4.26 (Imagem do autor).

Figura 93 - Comparação entre valores máximos e mínimos de iluminação global na Unreal Engine 4.26 (Imagem do autor).

Motion Blur

Motion blur, é um efeito comum no cinema e na fotografia, e é correspondente ao desfoque ou movimento esborratado obtido numa determinada cena quando algo se move demasiado rápido para que a câmara ou as suas configurações consigam captar a 100% o objeto em movimento, isto está como tal normalmente interligado entre os *frames* por segundo que um filme é gravado e com a velocidade do obturador. Inclusive, isto acontece com o olho humano, podendo soar presuntuoso, quem nunca abanou, por exemplo a mão em frente do olhar e reparou que caso o olhar não esteja a focar ou a acompanhar a mão, que esta cria um rasto por onde passa.

Na *engine*, mais uma vez a equipa por trás da Unreal Engine adicionou a possibilidade de introduzir *motion blur* nos projetos desenvolvidos a partir da mesma, e dentro de momentos exploraremos os vários controladores que constituem o filtro de post-Processing referente a *motion blur*, no entanto, antes de começar há dois pontos que temos de preparar de modo a poder visualizar melhor onde o *blur* acontece.

Como tal ao clicar na **lista do botão *play***, devemos selecionar o ***Simulate*** como se observa na **Figura 94**, e de seguida ativar o *overlay* de visualização clicando em **Show** seguido de ***Visualize*** e por fim ***Motion Blur*** como se observa na **Figura 95**, com isto deve-se obter uma figura semelhante à **Figura 96** onde tudo o que seja considerado estar

em movimento terá uma cor amarelada invés do cinzento que cobre a maioria da cena atualmente. Para além disto, é importante saber que o *motion blur* também é aplicado à câmara do jogador, ou seja, cada vez que este olhe em volta, o *motion blur* também se ativa, como tal é preciso ter cuidado de modo que este ocorra quando é necessário, mas que não se oponha à experiência do jogador.

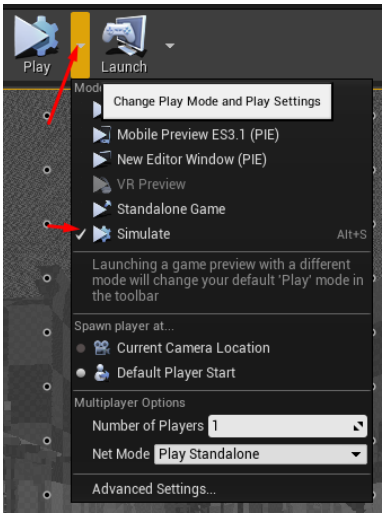

Figura 94 - Captura de tela de como modificar o *play* para *simulate* na Unreal Engine 4.26 (Imagem do autor).

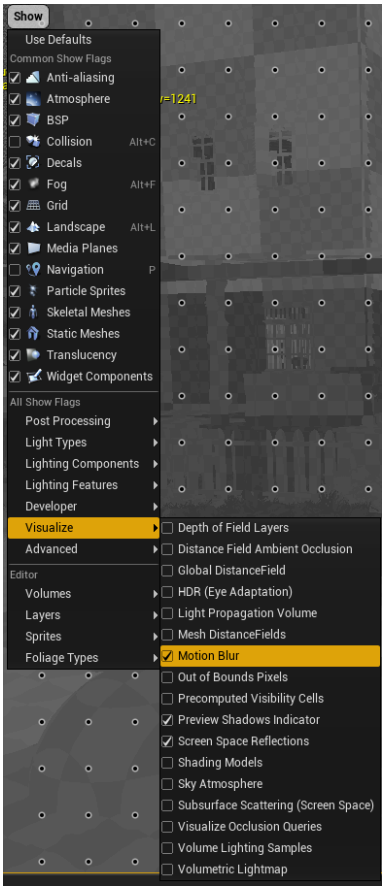

Figura 95 - Captura de tela de como ativar a sobreposição visual de *Motion Blur* na Unreal Engine 4.26 (Imagem do autor).

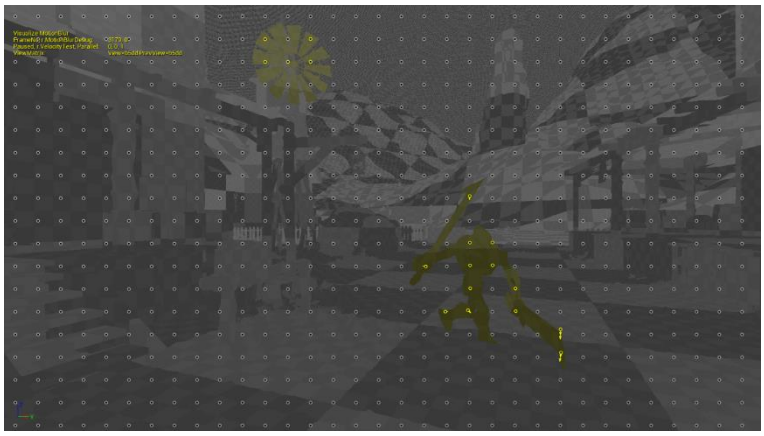

Figura 96 - Exemplo de como a sobreposição visual de *Motion Blur* altera a imagem, na Unreal Engine 4.26 (Imagem do autor).

Tendo ativado o *overlay* devidamente, analisemos agora os quatro controladores pertencentes ao efeito, sempre também em mente que é importante alternar entre o *overlay* e a vista normal para se ter noção do *blur*. Mais ainda, veja-se que com o *overlay* ligado, para além de ser possível analisar os objetos que vão ser contabilizados para o *motion blur*, vemos também a direção do movimento através de pequenas setas indicadoras.

- ***Amount***

O primeiro controlador controla geralmente a intensidade do *motion blur*, sendo que 0 significa que o *motion blur* está totalmente desativado. No entanto, como veremos mais à frente, este pode estar totalmente ligado, portanto, com o valor de 1, mas se outros valores estiverem configurados para tal, o *motion blur* pode ser desativado ou até continuar ativado, mas de uma maneira quase impercetível. Na **Figura 97** apresenta-se uma comparação entre vários valores e o resultado destes.

Figura 97 - Comparação entre valores de intensidade ou quantidade diferentes de *motion blur* na Unreal Engine 4.26 (Imagem do autor).

- **Max**

O segundo controlador, controla a distorção máxima criada pelo *motion blur*, como percentagem em relação à largura do ecrã, daí que um valor de 0 desliga totalmente o *motion blur*, mas o valor de 1, ou seja, 1% da largura do ecrã (no caso de um ecrã *Full HD*, portanto de 1920×1080 pixeis, o valor é de 19 píxeis) dependendo das outras configurações, pode já

ser o suficiente para que ativar ou até exagerar o *motion blur*. Na **Figura 98** apresenta-se uma comparação entre três valores diferentes e os seus resultados visuais, onde no caso de 100 partes do movimento começam inclusive a desaparecer, semelhante aos rastos de movimento visto em vários desenhos animados.

Figura 98 - Comparação entre valores de distorção de *motion blur* diferentes na Unreal Engine 4.26 (Imagem do autor).

- **Target fps**

Este campo controla, portanto quando é que o *motion blur* deve acontecer, tal e qual como acontece no cinema, ou seja, se um objeto se estiver a mover demasiado rápido em relação às configurações da câmara, e como tal, da velocidade do obturador, o objeto é capturado com *motion blur*, no entanto, se for o oposto, ou seja, o objeto move-se mais lentamente que o ratio de *frames* ou velocidade de obturação da câmara, o objeto não possui *motion blur*, como tal, em termos da *engine*, este controlador permite definir-nos se o *motion blur* é ou não ativado e se sim, a partir de quantos frames por segundo é que o objeto tem de se estar a mover de modo a possuir *motion blur*.

Como tal, um valor muito baixo como 1 ou 2 é mais que suficiente para *ativar motion blur* em praticamente tudo o que acontece na nossa cena, já à medida que nos aproximamos de valores superiores a 25, o *motion blur* começa a aparecer apenas em objetos que se movam rápido, e os que se movem lentamente não vão possuir *motion blur,* e à medida que o valor aumenta, mais rápido tem de ser o movimento de modo que este sofra *motion blur*.

Por fim é importante saber que, se o valor do controlador for 0, este não aplica *motion blur* a tudo, mas sim, o *motion blu*r depende de quantos *frames* por segundo o produto final vai correr, como tal um computador mais fraco que apresente um rácio de *frames* menor, vai conter mais *blur* que outro sistema com um ratio superior.

Veja-se a **Figura 99** para se ter uma ideia do resultado através da utilização de quatro valores diferentes.

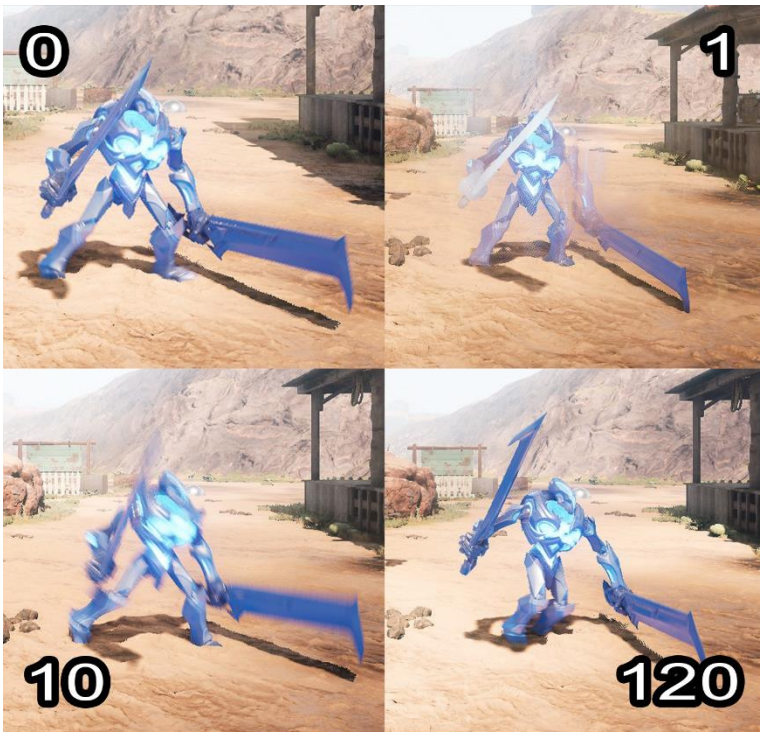

Figura 99 - Comparação do que acontece ao *motion blur* quando o alvo de fps tem valores diferentes, na Unreal Engine 4.26 (Imagem do autor).

- ***Per Object Size***

Por fim, o último ponto que controla o *motion blur*, define o valor mínimo necessário que um objeto tem de ter em termos de raio, em comparação com a percentagem da largura do ecrã, podendo assim fazer com que objetos que apareçam menores na cena, nem que seja devido a estarem demasiado longe, não possuam *motion blur,* podendo assim melhorar o desempenho do projeto ao livrar recursos que seriam necessários para realizar os cálculos e aplicar o *motion blur*.

Uma comparação entre três valores diferentes e os seus resultados pode ser observada na **Figura 100** e na **Figura 101**, neste caso, através do *overlay* de *Motion Blur*.

Figura 100 - Comparação de valores raios diferentes nas propriedades do *motion blur* da Unreal Engine 4.26 (Imagem do autor).

Figura 101 - Comparação dos valores na Figura 100 com a sobreposição do visualizador de *motion blur* ativo, demonstrando os objetos que sofrem motion blur mediante os valores aplicados (Imagem do autor).

Screen Space Reflections

Screen Space Reflections, é um dos efeitos de *post-Processing* que tem vindo a sofrer grandes melhorias ao longo dos anos, começando com truques simples nos primeiros videojogos onde apareceu, nos quais o nome era outro, e onde basicamente qualquer superfície que fosse refletora, possuía uma câmara secundária que sintetizava uma segunda vez o ambiente de jogo, o que levava a reflexões que não podiam ser imperfeitas, portanto não continham sujidade nem reflexos, outra maneira era a de incluir duas versões da personagem, e duplicar o nível, no entanto, a personagem que seria supostamente o reflexo, possuía controlos invertidos, como é o caso do Super Mario 64 (Nintendo EAD, 1996), apresentado na **Figura 102**, mais uma vez dando a ideia de que a sala tinha uma parede com um espelho.

Figura 102 - Captura de tela do jogo Super Mario 64 (Nintendo EAD, 1996) onde se observa um dos primeiros exemplos de reflexos em videojogos.

Outra técnica que, entretanto, surgiu, funcionava de modo em que uma imagem era projetada no interior de um cubo, que utilizava a cor do cubo para tentar dar uma aproximação do que refletir mediante a câmara do jogador, mas ainda assim os reflexos não eram perfeitos.

Finalmente, entramos no que hoje em dia é conhecido com *Screen Space Reflections* ou abreviado como SSR, que permitia reflexos mais corretos, mas cujo impacto era superior no desempenho em relação aos dois anteriores.

Como tal, não é surpresa que hoje em dia várias técnicas são utilizadas para além do SSR de modo a criar reflexos realistas nos videojogos. Mas focando-nos agora apenas no filtro de post-Processing de SSR, em termos de como este se comporta na Unreal Engine 4, este possui três controladores.

- ***Intensity***

O primeiro controlador de todos controla a intensidade com que o SSR influencia os objetos com materiais refletores, como se pode constatar na **Figura 103**, especialmente no cubo da esquerda e na parte da esfera direita que se encontra mais próxima do chão.

Figura 103 - Comparação entre o mínimo e o máximo possível na intensidade do uso de SSR na Unreal Engine 4.26 (Imagem do autor).

- ***Quality***

O segundo controlador define a qualidade dos reflexos, especialmente em materiais onde estes não sejam 100% espelhados, ou cromados. No entanto, quanto maior o valor, melhor os reflexos destas superfícies, mas também mais recursos de hardware são necessários, para além disso, como se pode constatar na Figura 104, esta mudança nem sempre resulta em algo visível, independentemente dos valores utilizados, sendo, portanto, um dos principais pontos de debate no que toca à restrição de utilização na qualidade máxima.

Figura 104 - Comparação entre três valores diferentes de qualidade de reflexos na Unreal Engine 4.26 através do uso de SSR (Imagem do autor).

- ***Max Roughness***

Por fim, o último controlador define a partir de que ponto os reflexos devem aparecer em superfícies onde a rugosidade não é perfeita. Isto pode constatar-se Figura **105**, onde a esfera da esquerda, perto do ponto de contacto com o chão, e o cubo da direita apresentam alguns reflexos em vez de manchas de cor e sombras.

Figura 105 - Comparação de valores no que toca à rugosidade máxima dos reflexos de SSR na Unreal Engine 4.26 (Imagem do autor).

- **Considerações**

Por fim veja-se que como mencionado previamente, a melhor maneira de incluir reflexos num projeto, e neste caso, de Unreal Engine 4 é através da utilização de SSR, mas também de outras técnicas como, por exemplo *reflection captures*, que podem adicionar mais alguns detalhes nos reflexos ou até melhorar estes como é demonstrado na **Figura 106**.

Figura 106 - Comparação de reflexos na Unreal Engine 4.26, através de SSR, mas adicionalmente de *reflection captures* numa das cenas (Imagem do autor).

Misc

A última categoria serve para controlar de um método geral todos os filtros relacionados ao ecrã, como qualidade contra desempenho. Sendo então que valores inferiores a 100 irão aumentar o desempenho do projeto visto os efeitos utilizarem uma percentagem menor do ecrã, e só depois serem aumentados para o ecrã final, enquanto ao contrário, portanto valores superiores a 100 vão causar uma qualidade superior, mas com um desempenho inferior visto que estes efeitos são sintetizados a partir de um tamanho superior. Semelhante ao que acontece com o Anti-Aliasing geralmente relacionados a *Super Sampling*. No entanto, este efeito é apenas visível quando se corre o produto final, ou seja, o jogo.

2.2 *Ray Tracing* em tempo real, *Ray Tracing* tradicional e *Lumen*.

Como referido previamente, não será explorado a fundo os pontos referentes ao *ray tracing* como *post-Processing*, mas considera-se importante ter uma breve ideia do impacto visual que este pode criar num projeto. Do mesmo modo, admite-se como necessário uma pequena análise de mercado no impacto que este começa a ter atualmente.

Comece-se por observar a diferença entre Rasterização e *ray tracing*. A rasterização funciona através de uma câmara (virtual) que olha para todos os triângulos constituintes da cena 3D e determina quais é que são visualizados nesse momento. Através dessa informa-

ção, o motor examina as fontes de luz, assim como outros detalhes ambientais para adicionar luz e cor aos pixeis em cada triângulo. Isto resulta numa situação em que vários detalhes da imagem não apareçam, devido a serem provenientes de partes da cena que a câmara ignora e como tal não são contabilizados na sintetização. Assim sendo, sombras, reflexos, refrações entre outros detalhes são reduzidos ou totalmente inutilizados.

Já o *ray tracing,* funciona através do cálculo de todos os caminhos, de todos os raios de luz numa cena, seguindo-os até estes chegaram à câmara. Com isto, cada vez que um raio é refletido, refratado, transmitido ou absorvido, a luz e o caminho são alterados, tal e qual como acontece na física (mundo real). Por exemplo, se uma luz branca incidir numa superfície colorida, esta é refletida, com uma tonalização diferente, causando outros objetos a serem iluminados de uma maneira diferente do primeiro. Exemplo deste fenómeno pode ser constatado na **Figura 107,** onde a luz, sala e o cubo são brancos, e as esferas são coloridas. Devido à cor das esferas, quando a luz atinge as mesmas, as superfícies do cubo e da sala em redor destas sofre uma tonalização, azul ou vermelha. No fim da sintetização, o resultado é superior ao da Rasterização, visto as sombras, reflexos e outros detalhes possuírem uma fidelidade física maior, como se pode observar no exemplo da **Figura 108**.

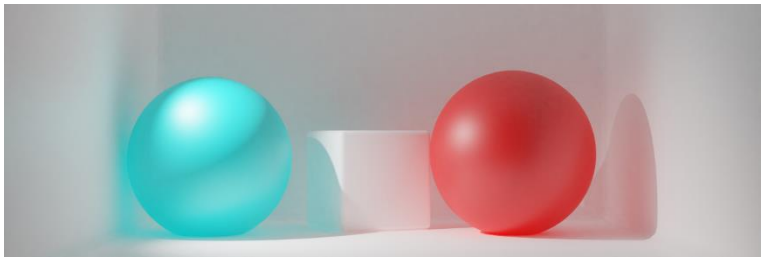

Figura 107 - Imagem ilustrativa de como a luz muda de cor ao atingir objetos coloridos (Imagem do autor).

Figura 108 - Imagem ilustrativa das diferenças entre rasterização (sem truques ou filtros adicionais) e *Ray Tracing (*Intel, s.d).

No entanto, um fator importante a considerar entre os dois é o impacto no desempenho do hardware que cada um tem, pois, o *ray tracing* requer hardware mais potente, caso contrário o tempo necessário para se alcançar o resultado da sintetização, quando comparado com o da rasterização, é bastante longo. Felizmente, devido ao avanço tecnológico, e à velocidade deste avanço, o poder do *hardware* dos sistemas informáticos atualmente já é considerável, e a possibilidade de realizar sintetizações em *ray tracing* em meros segundos passa a ser uma possibilidade até em computadores portáteis, ao contrário do que acontecia há 20 anos atrás. Exemplo deste avanço pode ser observado no filme Toy Story (Pixar,

1995), onde, através de 117 (cento e dezassete) computadores, um único *frame* de animação podia levar entre 45 (quarenta e cinco) minutos a 30 (trinta) horas a finalizar a sua sintetização, dependendo da cena (Insider, 2019). Atualmente, segundo o produtor, Jonas Rivera, o tempo necessário para sintetizar o filme novamente, seria inferior ao tempo necessário para ver o filme, cuja duração é de 77 (setenta e sete) minutos (Insider, 2019).

Mas o aumento do desempenho não provem totalmente do avanço tecnológico do *hardware*, visto o próprio software e os algoritmos utilizados para sintetização de cenas em 3D tendo vindo a evoluir. Um exemplo destas melhorias pode ser observado, através da análise da **Figura 109** e **Figura 110** onde, apesar do uso do mesmo sistema informático, software de edição, sintetização 3D, e com o método *Cycles* do *Blender* (Blender Foundation, 1994) as diferenças no tempo de execução da sintetização são consideráveis, especialmente entre a primeira (á esquerda) e a ultima (á direita) onde o aumento do desempenho é superior a 2000% (dois mil por cento). Numa fase inicial de análise, nota-se imediatamente a diferença entre um algoritmo que só utiliza o poder do processador, em contraste com um que utiliza o processador e a placa gráfica do computador. De seguida vê-se que, a diferença entre o algoritmo *cuda* ou o *optiX* já não é tão grande, mas, ainda assim, é de 125% (cento e vinte e cinco por cento). Finalmente observa-se que ao utilizar um filtro de remoção de ruido, invés de configurações elevadas nas várias propriedades do *Cycles,* que o tempo volta a diminuir, e o aumento do desempenho é de 318% (trezentos e dezoito por cento).

Figura 109 - Comparação do tempo necessário para sintetizar uma cena, com algoritmos diferentes (Imagem do autor).

Figura 110 - Pormenor da Figura 109 onde se observa que a qualidade do resultado das sintetizações, é, eficazmente, o mesmo (Imagem do autor).

Assim sendo é importante denotar que o resultado de uma sintetização com *ray tracing* depende da quantidade de raios de luz emitidos ou de amostras, e é aqui que a diferença entre *ray tracing* em tempo real e *ray tracing* para filmes, ou *ray tracing* total começa-se a denotar. No caso do *ray tracing* em tempo real, normalmente a quantidade de raios de luz emitidos costuma ser 16, podendo raramente ir até aos 64, dependendo do resultado de

qualidade/desempenho que se pretende. Para se melhor se entender a relação entre a qualidade e o desempenho veja-se a **Figura 111** em baixo, que demonstra uma comparação de uma cena simples, mas com várias amostras diferentes.

Figura 111 - Comparação entre valores de amostragem diferentes e o tempo necessário para lidar com as amostras durante a sintetização da imagem (Imagem do autor).

Outra variável a ter em conta durante este processo é dependente da quantidade de vezes que os raios de luz são refratados ou refletidos, que resulta num grande impacto visual, caso este processo seja efetuado numa quantidade elevado. Sendo necessário sublinhar que eventualmente o impacto começa a diminuir, resultando numa relação entre o resultado visual e desempenho cada vez pior. Assim sendo, é comum utilizar uma quantidade de saltos no valor de 16 ou 32. Como se pode, constatar na **Figura 112**, a qualidade visual é diminuta quando abaixo de 4 saltos, sendo que mesmo com 4 saltos o lado esquerdo do cubo não demonstra transparência, no entanto, com 16 ou mais saltos, a diferença começa a ser difícil de se notar, enquanto o tempo necessário para o processo começa a ser bastante semelhante (no caso da cena utilizada).

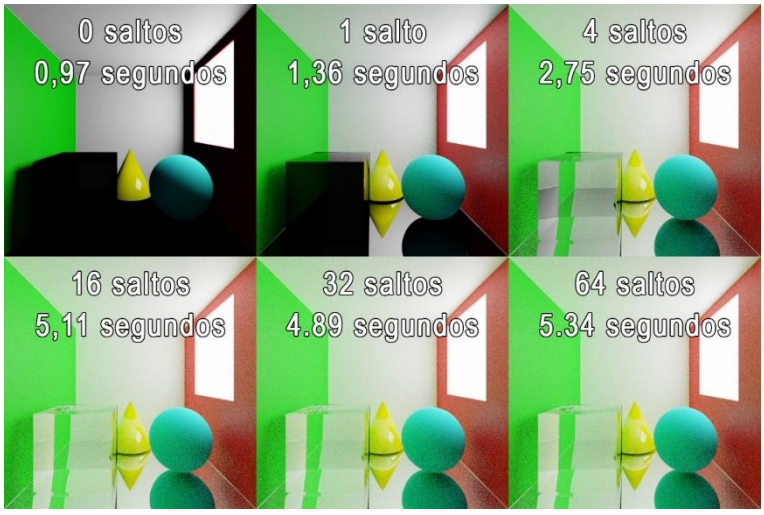

Figura 112 - Comparação entre valores de saltos diferentes, e o tempo necessário para sintetizar a imagem mediante esses valores (Imagem do autor).

Por fim existe ainda outro grande elemento que diferencia as duas categorias de *ray tracing*, embora este comece a ser utilizado em ambos os tipos devido ao impacto visual que proporciona, aumento do desempenho e resultados visuais. Este processo é conhecido por

denoising, e tem como funcionalidade a redução do ruído causado na imagem final após esta ser sintetizada. Isto faz com que uma imagem com um quarto das amostras possa ter uma qualidade final igual ou semelhante à que utiliza a totalidade das amostras, demorando esta muito menos tempo a ser apresentada. Sendo que, a utilização de *denoising* apenas adiciona alguns milissegundos à sintetização

Vejam-se as seguintes comparações da **Figura 113** e **Figura 114** e o tempo necessário para sintetizar cada imagem. Como já foi referido, aumentar a quantidade de amostras reduz a quantidade de ruído que esta possui. No entanto, quando se aplica o filtro de ruído, o valor de saltos necessários para imagens semelhantes é bem menor, assim como o tempo final de sintetização. No entanto, é necessário ter cuidado, pois se as amostras forem poucas, a imagem final irá apresentar alguns artefactos. Destaca-se ainda que o filtro utiliza poucos recursos, o que apenas aumenta alguns milissegundos à sintetização da imagem, e assim sendo, como se observa no fim, na **Figura 115**, o tempo ganho é bastante percetível, quando inclusive a qualidade da imagem é superior.

Isto faz com que o *denoising* seja extremamente importante, devido ao resultado final possuir a possibilidade de apresentar resultados superiores, em menos tempo.

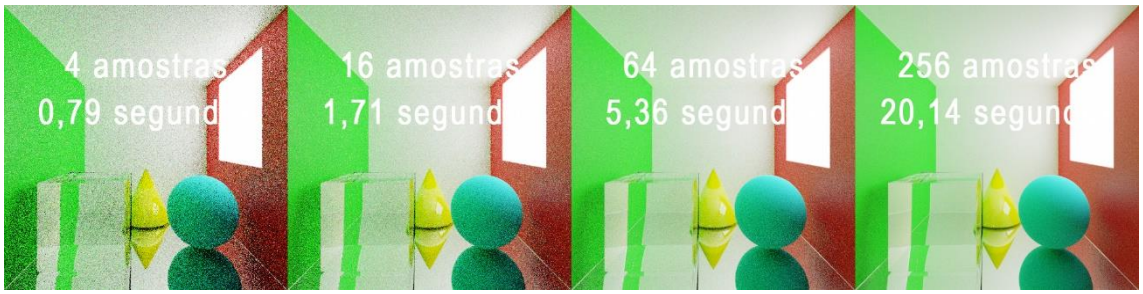

Figura 113 - Comparação entre valores de amostragem diferentes e o tempo necessário para sintetizar uma imagem mediante estes (Imagem do autor).

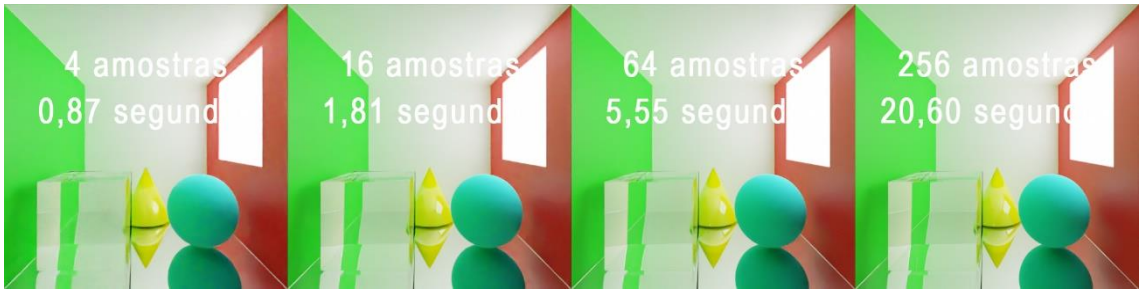

Figura 114 - Repetição da comparação da Figura 113, mas agora com um filtro de *denoising* ligado (Imagem do autor).

Figura 115 - Pormenor comparativo entre uma sintetização com um valor de amostras inferior, mas com denoising ligado, e outra sintetização com um valor de amostras superior e sem denoising, assim como a diferença de tempo necessária para sintetizar cada uma consoante os valores (Imagem do autor).

Esta é uma das técnicas utilizadas na Unreal Engine 5 (Epic Games, s.d), que apresenta uma das variantes do *ray tracing* conhecido como *Lumen*. Variante esta que possibilita a utilização de *ray tracing*, em tempo real. Pode-se também considerar este método de *ray tracing* como sendo focado para o desenvolvimento de videojogos ou projetos do género, sem a necessidade de hardware dispendioso. No entanto, este é ainda um sistema em desenvolvimento, e no momento da escrita deste documento, os resultados obtidos através de *Lumen* são voláteis e podem por vezes apresentar resultados indesejados, cabendo assim aos desenvolvedores a decisão relativa à utilização de *Lumen*. O mesmo se aplica à decisão da escolha entre *ray tracing* em tempo real ou Rasterização. Como se observa na **Figura 116** as diferenças visuais entre rasterização e *ray tracing* são poucas atualmente, onde se vê a maior diferença é nos *frames* por segundo, visto o *ray tracing* ter um impacto de dez *frames* por segundo a menos, o equivalente a 33% de desempenho. No entanto é de denotar que caso o fator que reina na decisão da ferramenta forem apenas os *frames* por segundo, *Lumen* seria a escolha acertada devido a proporcionar um aumento relativo de dez *frames* por segundo, ou seja, um aumento de 33% quando comparado com a imagem *rasterizada*, e o dobro dos *frames* por segundo em comparação com o Ray Tracing. Mais ainda, neste caso apresentado, os reflexos são superiores, onde nos outros dois casos os reflexos estão esborratados, quase como manchas, no Lumen estes correspondem mais à forma dos objetos.

Figura 116 - Comparação do desempenho dos três tipos de sintetização disponíveis com a Unreal Engine 5 acesso antecipado (Epic Games, s.d) (Imagem do autor).

3 A influencia do *post-Processing* no desenvolvimento de um videojogo.

Até agora foram mencionadas várias maneiras nas quais, a utilização do *post-Processing,* durante o desenvolvimento de um jogo, tende a melhorar o resultado, mas, este estudo foi feito de uma maneira objetiva e essencialmente para servir de exemplo para cada efeito de *post-Processing*. Analise-se agora uma lista mais detalhada, não exaustiva nem exclusiva, de como o *post-Processing* pode criar resultados rápidos na implementação de várias ideias durante o desenvolvimento de um videojogo.

Comecemos por ver, provavelmente, uma das formas onde o *post-Processing* tende a ser mais associado. A visão noturna ou infravermelha, é bastante comum nos jogos de FPS, em especial em níveis noturnos, onde a iluminação é pouca. E, fora recorrer ao uso de lanternas de modo a iluminar a cena, a alternativa são binóculos com visão noturna infravermelha. Ora, na implementação, há inúmeras maneiras de colocar a opção de visão noturna, mas, possivelmente uma das mais simples, no caso da Unreal Engine, seria a de utilizar precisamente o post-Processing. Portanto, adicionando uma câmara que funcionaria como binóculos noturnos, onde se modifica a iluminação global, a exposição e o *color grading,* de modo que a cena tenha uma tonalidade quase exclusivamente verde (ou outra cor, visto que, comummente o verde é a cor mais associada com visão noturna). Para complementar este efeito, ainda podem ser adicionados alguns ajustes, como, o uso de *bloom*, de modo que este cegue o jogador quando se olha para um ponto luminoso, o uso de ruído, e ainda a adição de um *Post Process Material* para enfatizar ainda mais o efeito. Podendo até utilizar este *Post Process Material* como uma instância, e assim criando uma maneira rápida de alterar a cor da visão noturna. Ao observar a **Figura 117** pode-se constatar os resultados obtidos com uma implementação simples (iluminação global, exposição e *color grading)* e uma mais avançada.

Figura 117 - Cena sem tratamento de visão noturna (esquerda), com tratamento base (centro) e tratamento avançado (direita).

Este é, como referido, um dos primeiros exemplos que aqui falamos, mas mais do que como este pode ser utilizado é importante denotar que este não é um efeito meramente visual, pois a sua função pode servir várias outras funções como jogabilidade e narrativa. Assim sendo, vemos que o uso do *post-Processing* não está somente ligado ao grafismo de um jogo, e pode ter uma função mais abrangente.

Continuando com a exploração de como o post-Processing pode influenciar, adicionar ou melhorar certos e determinados meios no desenvolvimento de um jogo, é importante

falar de um método utilizado não só para o desenvolvimento de jogos, mas também no cinema e da fotografia. O uso da psicologia e teoria da cor, de modo a promover emoções, ideias e várias outras associações ao espetador. Dar um exemplo para cada cor, no entanto, seria extremamente extensivo, e mesmo exemplos exaustivos de apenas uma cor, tem bastante que se lhe diga. Como tal o melhor mesmo é a leitura e estudo de livros sobre o tema como:

The Complete Color Harmony, Pantone Edition: Expert Color Information for Professional Results (Eiseman, 2017) escrito por Leatric Eiseman, especialista em cor e conhecida como "a guru internacional da cor", é um livro que fala sobre vários pontos da cor como, a harmonia da cor, terminologias básicas da cor, discórdia e dissonância, a psicologia da, a relação entre cor e disposição/espirito, a relação entre a personalidade e a cor, entre outros assuntos. Contudo o livro é mais orientado ao uso de cores pigmentadas, associado á marca Pantone, mas que não deixa de ser uma leitura essencial para quem deseje ter mais conhecimento sobre a cor.

A Psicologia das Cores: Como as cores afetam a emoção e a razão (Eva Heller, 2012) relata a pesquisa realizada por Eva Heller sobre as cores e como estas são percebidas por uma variedade de indivíduos. As cores são assim divididas em capítulos, onde cada uma é estudada nos seus significados assim como esta é equiparada a outras cores que podem ser associadas a esse significado.

Com isto, o uso da cor é uma das formas mais completas de criar um leque superior de emoções, como euforia, alegria, tristeza entre outros no desenvolvimento de um jogo. Seja na narrativa, no ambiente visual, na *interface*, etc., as potencialidades de utilizar a cor como uma vantagem são bem definidas. Mas é esta capacidade de influenciar os meios nos quais a cor interage, que existe a complexidade. Ora abrir um dos livros anteriores, consultar que cor vai dar um determinado sentimento é fácil, já entender a matéria em si de modo a chegar ao ponto de compreensão em que a consulta dos livros é rara torna-se uma tarefa complicada e que requer muito estudo e tempo. É devido a esse facto que *color grading* é provavelmente a técnica mais difícil de dominar do *post-Processing* e não se esqueça que não é só saber o que cada cor significa ou o tom que esta dá, pois, um conjunto de cores pode alterar o seu significado.

Associado à teoria da cor, de certo modo, temos o controlo da exposição, também utilizado no cinema e na fotografia. A exposição pode proporcionar não só emoções como ideias, alterando rapidamente um ambiente e com isso a narrativa e o género. Por exemplo, uma mansão antiga, bem conservada, de dia, bem iluminada e com uma exposição alta, pode dar a ideia de algo fantasioso e maravilhoso, uma casa da realeza ou uma casa de sonho. Já ao diminuir a exposição (e aumentarmos o contraste assim como aplicar uma tonalidade verde, devido à associação do verde com o paranormal), até que a escuridão na casa seja imensa, o que antes era maravilhoso passa a ser fantasmagórico, alterando o ambiente para algo do

género de horror, terror, ansiedade, mistério, etc. Passa-se então, de uma forma algo linear e simples, do encantador para o aterrador, apenas com a alteração da exposição. Exemplo disso pode ser observado na **Figura 118**.

Figura 118 - Cena de ficção científica, bem iluminada, simbólica de um ambiente seguro e familiar (esquerda) em contraste com a mesma cena, mas cuja iluminação é reduzida através da exposição, resultando num aspeto mais obscuro e possivelmente perigoso (direita).

Ainda na análise da exposição, algo que começa a ser comum atualmente, é a utilização da exposição automática, associada ao olho humano e a sua capacidade de, até um certo ponto, se adaptar à quantidade de luz da situação em que se encontra. Caso comum de quando deste fenómeno num jogo ocorre quando a personagem entra ou sai de uma área escura, como um túnel subterrâneo, para uma zona bem iluminada no exterior durante o dia. Nesse momento, inicialmente, este tem demasiada luminosidade e tem dificuldade em ver, mas rapidamente se adapta e é novamente capaz de ver o que o rodeia sem problemas, acrescentando assim mais uma forma onde o *post-Processing* pode ajudar com a jogabilidade.

Outra forma de adicionar elementos que não sejam apenas visuais, como referenciado previamente, é a utilização de ***Post Process materials*** para alterar o aspeto visual do jogo. Por exemplo, tornar uma cena realista, numa cena semelhante a uma banda desenhada antiga, incluindo todos os defeitos e problemas com as impressões de bandas desenhas antigas. Isto pode ainda ser utilizada ainda com o objetivo de conectar mais facilmente com o publico alvo. Por exemplo, um publico mais infantil, em princípio, prefere uma cena menos realista, mais animada e estilizada, tal e qual os desenhos animados ou jogos que estes consomem. Este pode ser utilizado ainda de outras formas, agora realistas, como adicionar pixelização ou certos artefactos a uma imagem, na ideia de, por exemplo, o jogador estar a ver uma cena através de uma câmara de CCTV, ou através da ampliação digital cuja qualidade não é a melhor. Continuado ando no ponto de utilização de PPM em câmaras e origens digitais, pode-se ainda adicionar efeitos de falhas e erros à imagem, como se, por exemplo, houvessem interferências, ou o material fosse já antigo e tivesse defeitos, de modo a proporcionar, por exemplo, alguma tensão no caso de um jogo de terror, onde a personagem é um repórter do sobrenatural, e quando este se aproxima de algo dessa mesma natureza, a imagem começa a ter falhas, e quanto mais ocorrentes e graves os defeitos, mais perto este se encontra de algo sobrenatural, podendo até estar em risco de vida. Um exemplo

de como a utilização de PPMs pode alterar o look de uma cena pode ser constatado na **Figura 119**, onde a cena passou de ter um aspeto realista para o aspeto de uma banda desenhada, através da utilização dos filtros demonstrados na **Figura 120**.

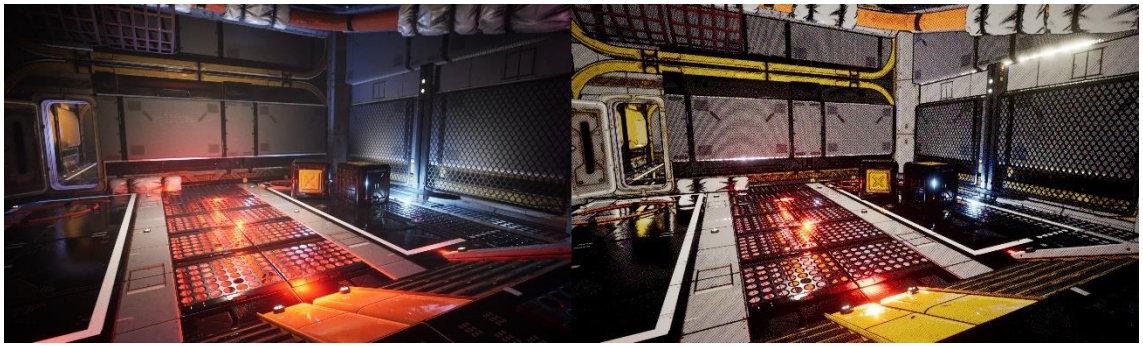

Figura 119 - Comparação entre uma cena normal (esquerda) e uma com vários Post Process Materials aplicados, resultando num aspeto de banda desenhada impresso (direita).

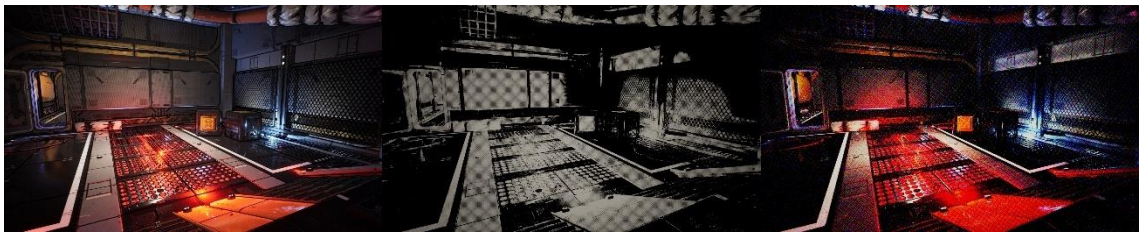

Figura 120 - Os Post Process Materials utilizados na imagem anterior, por ordem da esquerda para a direita. Linhas de contorno, Crosshatch de sombras e filtro Halftone para simular os problemas de impressão de muitas bandas desenhadas antigas.

Continuando no campo da imagem digital, o uso de aberração cromática e *vignetting* pode ajudar a reforçar a ideia de que o espetador vê a cena através de um ecrã antigo, como. Este aspeto pode ainda ser reforçado com a introdução de grão e um PPM que crie linhas de varredura e interferências, como se pode constatar na **Figura 121**.

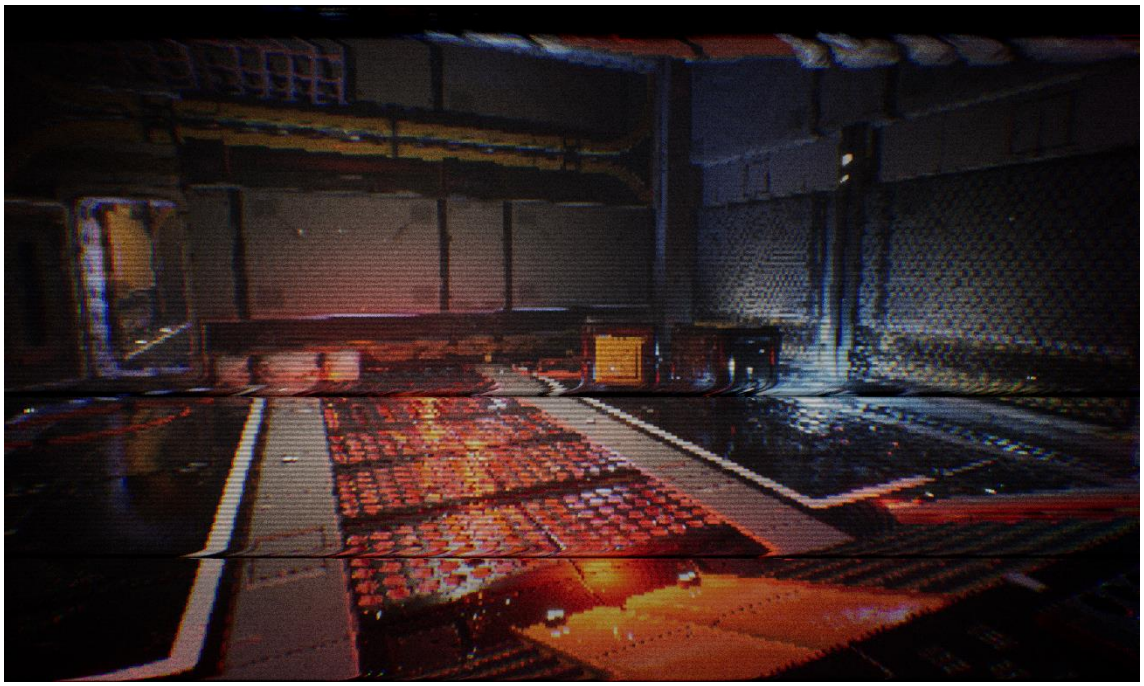

Figura 121 - Demonstração do efeito de CRT.

Por si, o efeito de aberração cromática pode ser utilizado também para, por exemplo, proporcionar a ideia de que a personagem está sobre o efeito de substâncias, como álcool ou estupefacientes, devido à associação que uma aberração cromática exagerada tem com alucinações, e assim sendo, mais uma vez, a implementação desta ideia deixa de ser apenas algo visual e ser algo com a capacidade de proporcionar elementos de jogabilidade, narrativa e até de *interface*.

Assim sendo, consegue-se observar que o *post-Processing* pode ter um impacto que trespassa a função visual e pode inclusive servir outras finalidades como a narrativa, *interface*, jogabilidade, etc. e estes foram apenas alguns exemplos do que é possível realizar, de uma forma muitas das vezes diegética, com o *post-Processing*, mais alguns exemplos que se poderiam aqui colocar seriam os controlos de câmara para criar momentos cinemáticos durante a história ou narrativa de um jogo, ou os vários efeitos que se podem utilizar num jogo de FPS, especialmente com o uso de profundidade de campo, exposição, sujidade na lente entre outros, o *motion blur* para realçar a perceção da velocidade de certos movimentos (popularizado no cinema devido às restrições técnicas das camaras, na captura de objetos em movimento a uma grande velocidade), etc.

Resumidamente, o importante a retirar é que o *post-Processing* serve várias finalidades para além de adicionar detalhe visual ao videojogo, mas há alguns pontos que têm de ser considerados durante a implementação do *post-Processing*, porque mesmo quando a ideia é a de criar um produto realista, há algumas opções que têm de ser contornadas, devido a haver sempre vários fatores, como o facto do público-alvo, a decisão no caso do entretenimento, a funcionalidade, etc.

É devido a estas várias outras decisões durante o desenvolvimento, que é comum ver efeitos de *post-Processing* a serem utilizados de formas que não deveriam ser, como é o caso já aqui falado várias vezes, dos jogos de FPS, no que toca a sujidade nas lentes, em personagens que não têm nenhuma proteção que possa ser suja, porque devido a tantos FPS já utilizarem esse efeito da forma que utilizam, acaba por ser estranho na perspetiva da maioria dos jogadores não o ter, e isto acontece com quase todos os filtros e efeitos de *post-Processing*, aliás há inclusive uma certa discórdia entre jogadores no que toca ao uso de *lens flares* e de *bloom*, visto que estes são demasiado ofuscantes e/ou distraírem demasiado o jogador do que ocorre na cena.

Com isto dito é importante considerar que, quando se tomam as decisões artísticas de como se implementa o *post-Processing* no produto, seja com o intuito de ter um resultado realista ou não, existem vários elementos a ter em consideração. Exemplo como o fator da funcionalidade, evitar a alienação do público-alvo e o que este já está habituado.

4 Conclusão da importância do *post-Processing*.

"Qual é a importância do *post-Processing* no desenvolvimento de videojogos?". Esta era a pergunta inicial que se colocava na abertura desta parte (teórica), assim como a pergunta a que, fulcralmente, se pretendia responder com esta dissertação. Neste ponto, após a análise histórica de como este tem vindo a evoluir, o que cada componente é capaz de fazer, e as finalidades que podem facilmente ser atingidas com o *post-Processing*, finalidades estas que ultrapassam largamente o meramente visual, a importância encontra-se estabelecida. No entanto, existem exceções. Podemos tomar como exemplo o tempo disponível para desenvolvimento do videojogo e as capacidades da equipa envolvida, bem como o objetivo do próprio videojogo. Estes fatores, mesmo durante a utilização do *post-Processing*, determinam a forma como os problemas e as soluções são colocados.

Nesses casos, conceitos como o fisicamente correto são sobrepostos por todas as outras decisões pertencentes ao desenvolvimento do videojogo. Estas decisões podem ser criativas. Por exemplo, se a ideia do videojogo estiver relacionada a uma paródia na forma de desenho animado, então elementos como *lens dirt,* exposição automática, reflexos fisicamente corretos e *lens flares,* entre outros, podem ser desnecessários. Poderão também estar em jogo fatores de ordem financeira e/ou condicionamentos de tempo; estes muito provavelmente impossibilitarão o tratamento minucioso do grafismo do videojogo de modo a este ser realista.

Parte Prática

Demo - The Last Star

Inicia-se aqui o relatório do desenvolvimento do projeto até ao demo disponível online, relatando todas as dificuldades, questões, alterações implementações e soluções das várias fases de desenvolvimento do projeto, no ponto de vista dos autores e mediante tudo o que foi examinado e compreendido consoante os temas falados na parte anterior.

1 Introdução ao projeto.

The Last Star é um videojogo em primeira pessoa do tipo *First-Person Shooter*, de sobrevivência e de horror, num contexto de ficção científica, com elementos de exploração, apresentado sobre a forma atual de uma demo com a duração de 4 a 8 minutos dependendo do conhecimento do jogador e da quantidade de exploração realizada.

A ação decorre trezentos triliões de anos no futuro, numa nave espacial, construída para conduzir o que resta da humanidade, através de um portal temporal. Este portal, no entanto, precisa de absorver a energia da última estrela do universo, uma anã branca, visto que todas as outras estrelas se tornaram buracos negros, ou estrelas de neutros. Devido à viagem ser longa e distante, esta leva anos, o que resulta nos humanos serem preservados em cápsulas criogénicas. A manutenção e segurança da nave e do seu conteúdo é assegurada por uma tripulação de autómatos, cujo objetivo é o de certificarem que tudo ocorre sem falhas. Contudo, o programa dos autómatos é corrompido com um vírus maligno, o que os tornou máquinas de destruição.

O jogador encara o papel de um dos humanos, que acorda da sua cápsula de conservação pelo capitão da nave. O objetivo do jogador é salvar a humanidade da escuridão perpétua do universo, sobrevivendo aos autómatos, agora com intenções homicidas, e de ativar o portal que permitirá prolongar a sobrevivência da humanidade ao fazer com que esta viagem atrás no tempo para uma época onde os problemas de sobrevivência e de escuridão perpétua estejam longe.

Os controlos do jogo (**Figura 122**) seguem o que é considerado quase globalmente como padrão, e, como tal, as teclas WASD servem para que o jogador se desloque no ambiente, nomeadamente W para a frente, S para trás, A para a esquerda e D para a direita; mais ainda, pode-se recorrer ao uso da tecla Shift para correr e ao Ctrl para se agachar. Com a tecla E, pode-se interagir com vários objetos e pontos ao longo do jogo, e a tecla R serve para recarregar o carregador da arma. Através do rato, consegue-se olhar em redor e apontar quando se obtém a arma. Ao pressionar o botão do lado direito do rato, consegue-se fazer mira com a arma, e ao clicar no botão esquerdo consegue-se disparar e, ao pressionar sem

largar, dispara-se em modo automático. Como bónus, pode-se pressionar a tecla C para assobiar, e o F para atacar de modo físico, batendo, portanto, nos inimigos com a arma em vez de a disparar.

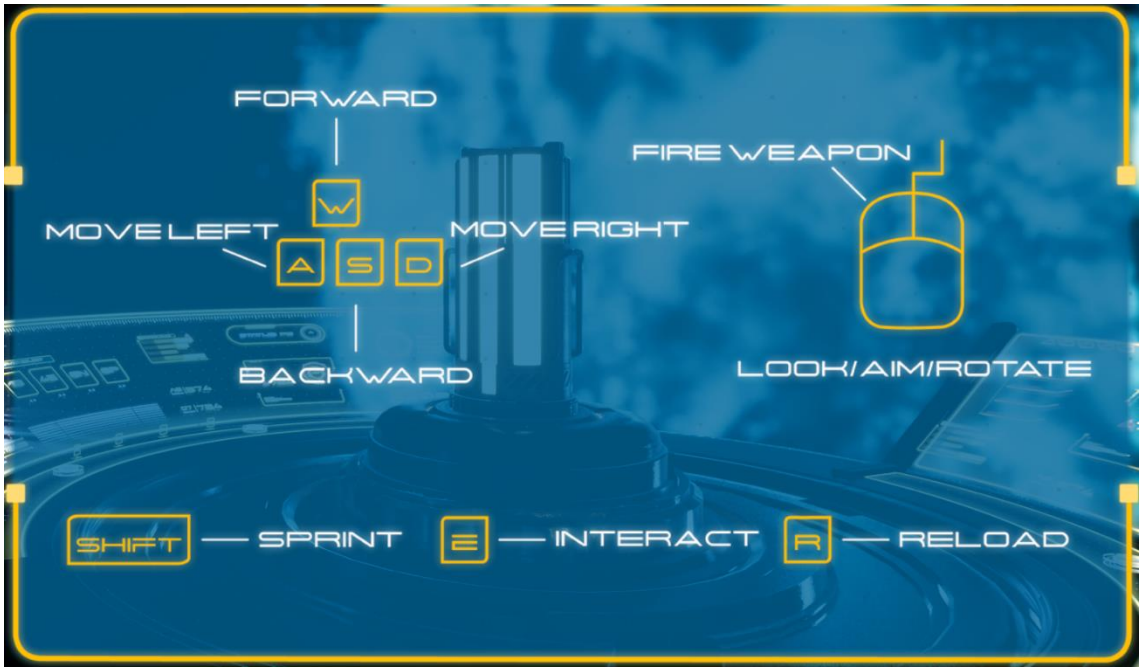

Figura 122 - Captura de tela do esquema de controlos do projeto (Imagem do autor).

2 Global Game Jam *2019* e o início do projeto.

O desenvolvimento do projeto teve início na Global Game Jam *2019*, com sede na Universidade da Beira Interior entre os dias 25 e 27 de janeiro de 2019. A equipa, na altura, estava constituída pelo *sound designer, storyteller* e programador Henrique Candeias, a artista conceptual, game designer e *level designer* Carolina Angelino e o artista de 3D e iluminação Sandro Gonçalves. O tema era *What home means to you*.

Para responder ao tema colocado, decidiu-se criar uma história intitulada de "White Dwarf", com semelhanças à história atual em que o projeto se encontra. O conceito então da história, na altura, resumia-se a um futuro pós-apocalíptico, onde todas as estrelas do universo já se tornaram em buracos negros ou estrelas de neutrões, e apenas uma estrela se mantinha viva, uma anã branca, onde se situava a última estação espacial humana ainda ativa. No entanto, todos os humanos mais importantes encontravam-se num estado de *suspensão criogénica*, à espera de que um dos robôs construídos para a manutenção da estação pudesse também arranjar uma solução para a sobrevivência da humanidade, antes que os últimos raios de luz da anã acabassem e toda a vida humana fosse perdida para sempre.

Houve ainda algum debate, que consistia na implementação de opções, em dois pontos da história. O primeiro ponto consistia em, quando o jogador, no papel de um robô, quase no fim da sua vida devido a vários danos causados ao longo dos séculos que este operou a estação, encontrasse um *item* que possuía poder Infinito, em vez de o utilizar mais à frente, absorveria o poder deste *item*, transformando-se num ser como Deus, com poder infinito, mas colocando um fim à tripulação humana da estação devido a não haver agora maneira ou necessidade de os reviver, criando-se assim um final que se considerava como mau. Caso contrário, o jogador leva este *item* até um portal, e neste portal tem duas opções: a primeira era de colocar o *item* no pedestal e ativar o portal, enviando a humanidade atrás no tempo, para uma época em que não havia problemas relacionados com a morte das estrelas, atingindo assim o final considerado como normal. No entanto, isto iria matar o robô. No entanto, caso o jogador examinasse o *item*, este poderia ser alterado, e quando fosse colocado no pedestal, o portal em vez de servir para viajar no tempo, emite uma radiação infinita, do género do *Big Bang*, que altera geneticamente os humanos, fazendo com que estes se tornassem semideuses, capazes de criar plantas, vegetações e animais, sem a necessidade de uma luz solar normal. Estes, em conjunto, seriam ainda capazes de reparar o robô, tornando-o num semideus humano como o resto da tripulação, e alterando as propriedades físicas da anã, para que esta nunca morresse, começando assim uma nova era na história da humanidade e atingindo o final perfeito.

Havia também a intenção de colocar outros robôs ao longo da estação, alguns com defeito, que atacavam o jogador, e outros que já foram desligados devido à falta de manutenção, ao qual o próprio jogador poderia retirar peças para se tentar arranjar a si próprio.

Tendo o plano todo delineado em como se queria o conceito do jogo, começou-se a planear a realização, mas visto que, a equipa era apenas constituída por um *sound* e *game designer*, uma artista 2D e um artista 3D, o primeiro obstáculo no desenvolvimento do projeto surgiu devido à falta de um programador. Foi nesse ponto que se decidiu utilizar a Unreal Engine, que na altura se encontrava ainda na versão 4.21, por esta possuir um sistema de programação através de diagramas, facilitando assim o problema relacionado com a falha de um programador.

No entanto, outros obstáculos surgiram ao longo do evento e da realização do projeto, nomeadamente que um dos três elementos da equipa teria de aprender a programar através do sistema de diagramas, para o qual o Henrique disponibilizou-se, mas com o infortúnio de ter de se ausentar do recinto físico onde o evento decorria devido a compatibilidades de *hardware* com a *engine*. Outro obstáculo também relacionado com a programação foi proveniente do facto de que, como era necessário aprender a programar, muitos conteúdos foram colocados de parte e não chegaram a ser implementados, como foi o caso das opções de fim, robôs inimigos, e arranjar-se o robô do jogador através de peças de outros robôs. Um obstáculo que já era esperado foi o tempo, considerando que todo o projeto teria de ser realizado em menos de 48 horas.

Outros obstáculos acabaram por surgir, mas de menor importância, como foi o caso da inexperiência da artista 2D que aprendeu a realizar arte conceptual de ficção científica para o projeto, assim como a desenhar mapas, mas também a tenra experiência na criação de ambientes 3D do género de ficção científica do artista 3D.

No final, após muito trabalho, o projeto tomou a forma de um pequeno demo, onde a história era exposta ao jogador através de um intercomunicador que se ativava em vários pontos do jogo, e cuja jogabilidade consistia em explorar o nível à procura de quatro chaves necessárias para abrir uma porta que continha o *item* que seria depois colocado no pedestal do portal e concluiria assim o jogo.

Em baixo, na **Figura 123**, pode-se consultar algumas imagens retiradas do projeto White Dwarf (Gonçalves, Angelino, & Candeias, 2019), como este se encontrava no final da *Global Game Jam* 2019, que pode também ser encontrado no sítio web da *Global Game Jam*, assim como todos os outros projetos realizados nessa edição.

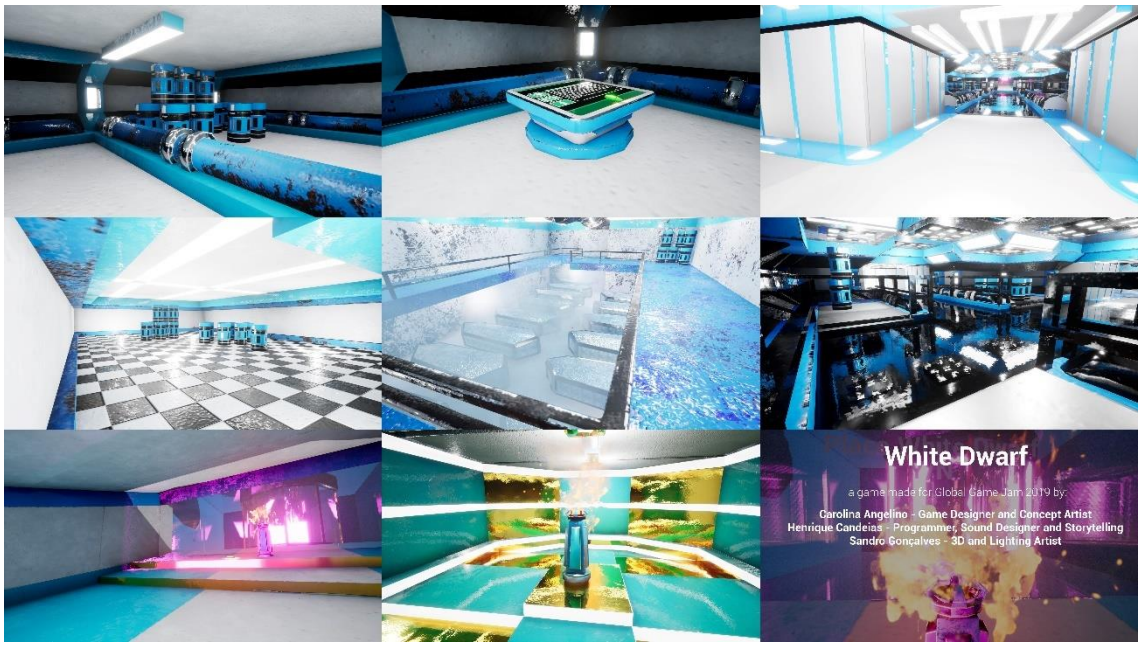

Figura 123 - Compilação de várias capturas de tela do jogo criado durante a Global Game Jam 2019, White Dwarf (Angelino, Candeias & Gonçalves, 2019) (Imagem do autor).

3 Desenvolvimento intermédio.

Com a necessidade de realizar um projeto em conjunto com a dissertação, houve alguma hesitação durante algum tempo no que respeitava ao desenvolvimento, nomeadamente com quem este seria desenvolvido, como e que forma teria, tudo isto associado também à necessidade de que este tinha de possuir a capacidade de estudo e aplicação de post-Processing.

No entanto, mediante debate, dois dos membros da equipa, nomeadamente Henrique Candeias e Sandro Gonçalves, acordaram que o projeto da White Dwarf seria um bom começo para o desenvolvimento das respetivas dissertações em que estes trabalhavam, e como tal, se o utilizador estivesse interessado em saber mais sobre como o projeto foi desenvolvido no ponto de vista do Henrique Candeias, este poderia consultar a sua dissertação futuramente no repositório digital da UBI[65], tendo então o projeto (base) decidido várias propostas de como se poderia melhorar o que se tinha nos mais variados aspetos da mesma, desde o grafismo proveniente das texturas e modelos 3D constituintes do ambiente, mas também programação, *storytelling*, *level* design, som etc.

Como tal, inicialmente começou-se a modelar novos *assets* 3D para criar um mapa que se tinha feito entretanto, mas havia sempre algo em falta no que tocava ao aspeto destes, para além do tempo necessário para a criação destes e, eventualmente, após o mapa final estar totalmente estabelecido e sem a necessidade de sofrer mais alterações, pelo menos de momento, optou-se por utilizar packs de *assets* disponíveis gratuitamente no Marketplace da Unreal Engine, e criar o ambiente através desses, tendo assim de se modelar apenas os *assets* que não se conseguiram encontrar no Marketplace.

Mediante este progresso e decisões, o videojogo sofreu bastante alterações, nomeadamente o *storytelling*, onde, em vez de numa estação espacial, ocorreu numa nave, não sendo já a personagem um robô, mas sim um dos humanos em suspensão criogénica. Alguns dos humanos ainda estavam ativos, como se pode verificar ao longo de alguns *tablets* espalhados pelo ambiente com mensagens relativas ao vírus que infetou os autómatos e os tornou hostis perante o jogador. Há também o comandante da nave a narrar a história à medida que o jogo progride, estando este ainda vivo durante parte do *gameplay*. Com isto, foi esquecida a opção de se ter mais do que um final, sendo então o único final atual o jogador conseguir fornecer energia ao portal para que este e todos os outros ainda em suspensão criogénica consigam viajar atrás no tempo para prolongarem a sua vida e escaparem à escuridão total do universo.

Com isto, construiu-se uma versão de demonstração do projeto para se expor como este corria, mas também para demonstrar as alterações feitas desde então e começar a receber

[65] https://ubibliorum.ubi.pt

também alguns comentários dos vários jogadores que experimentaram a demo. Algumas capturas de ecrã do projeto nesta fase seguem na **Figura 124**.

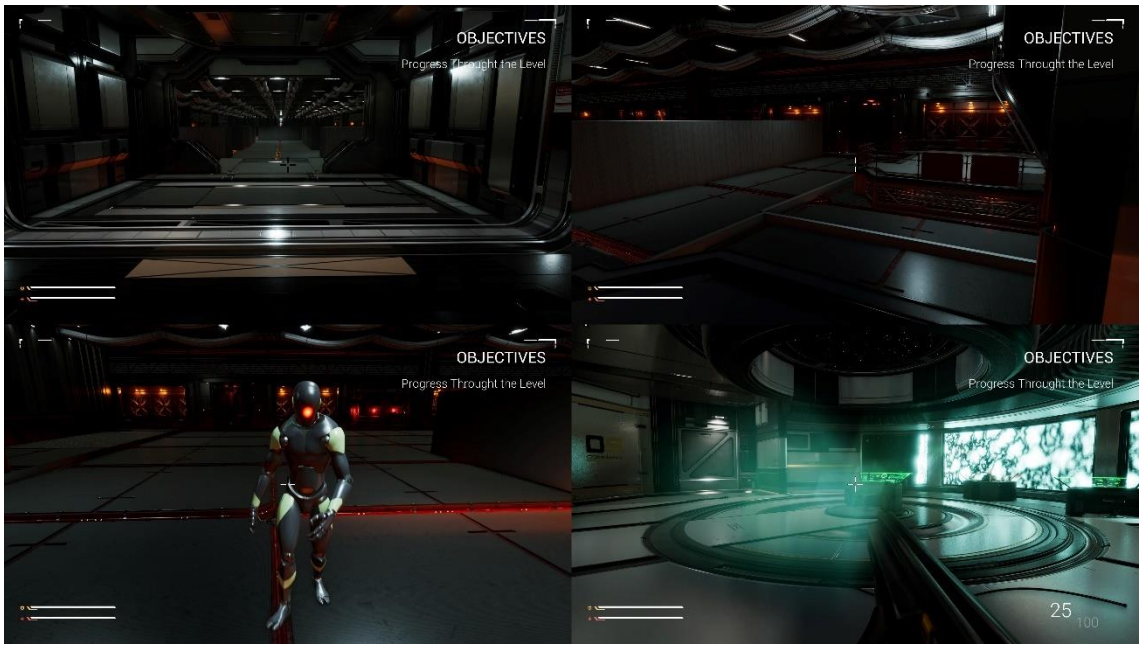

Figura 124 - Compilação de capturas de tela do demo The Last Star (Candeiras & Gonçalves, s.d) durante o seu desenvolvimento inicial (Imagem do autor).

4 Base para testes de post-Processing.

Após a demonstração do demo acima, outras alterações surgiram mediante comentário do público, assim como implementação de fatores que ainda não tinham sido possíveis de implementar. Estas alterações consistiram em:

- Alterar a disposição do mapa, de modo a criar um ambiente maior e mais orgânico do que a nave deveria ser, especialmente adicionando corredores adicionais, evitando problemas relacionados com o primeiro devido a corredores demasiado compridos e vazios;
- Colocação de *assets* que estavam em falta, substituindo assim os *placeholders* colocados previamente, como as paredes e divisões agora encontradas no nível intermédio (nível dois) com a ideia de criar o alojamento da tripulação. (Cantina, balneários, etc.) (**Figura 125**);
- Criação de *assets* não incluídos nos packs utilizados:
    - Portal (**Figura 126**);
    - Cápsulas criogénicas (**Figura 127**),
- Alteração de *assets* de modo a melhor se enquadrarem com o ambiente geral do jogo:
    - Remodelação e texturização da arma do jogador (**Figura 128**);
    - Texturização dos autómatos (**Figura 129**);
- Implementação de animações e sequências em relação a:
    - Absorção de energia proveniente da estrela (**Figura 130**);
    - Ligar o portal e conseguintemente a sequência de conclusão do jogo (**Figura 131**);
- Melhoria na resposta dos autómatos em relação à personagem do jogador;
- Implementação de ataques ao jogador no momento em que este apanha a arma, assim como adição de alguns autómatos em várias partes do ambiente.
- Pequenas alterações e implementações em relação ao menu inicial e ao de pausa.

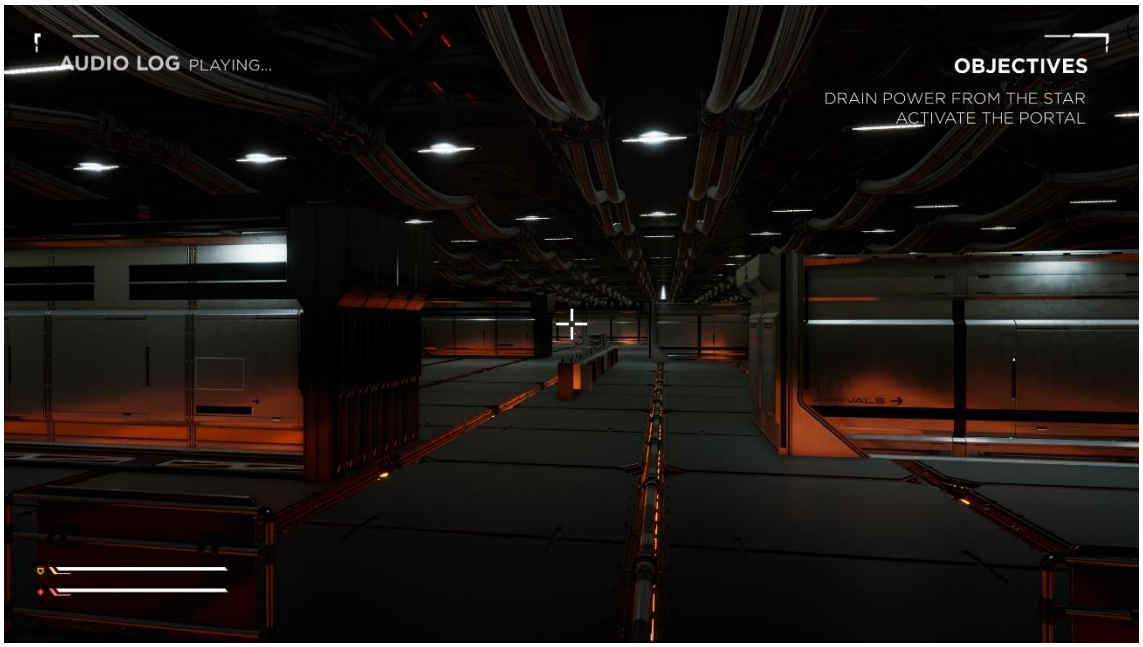

Figura 125 - Captura de tela, demonstrando a colocação de divisões devidamente modeladas e texturizadas invés de blocos cinza no projeto (Imagem do autor).

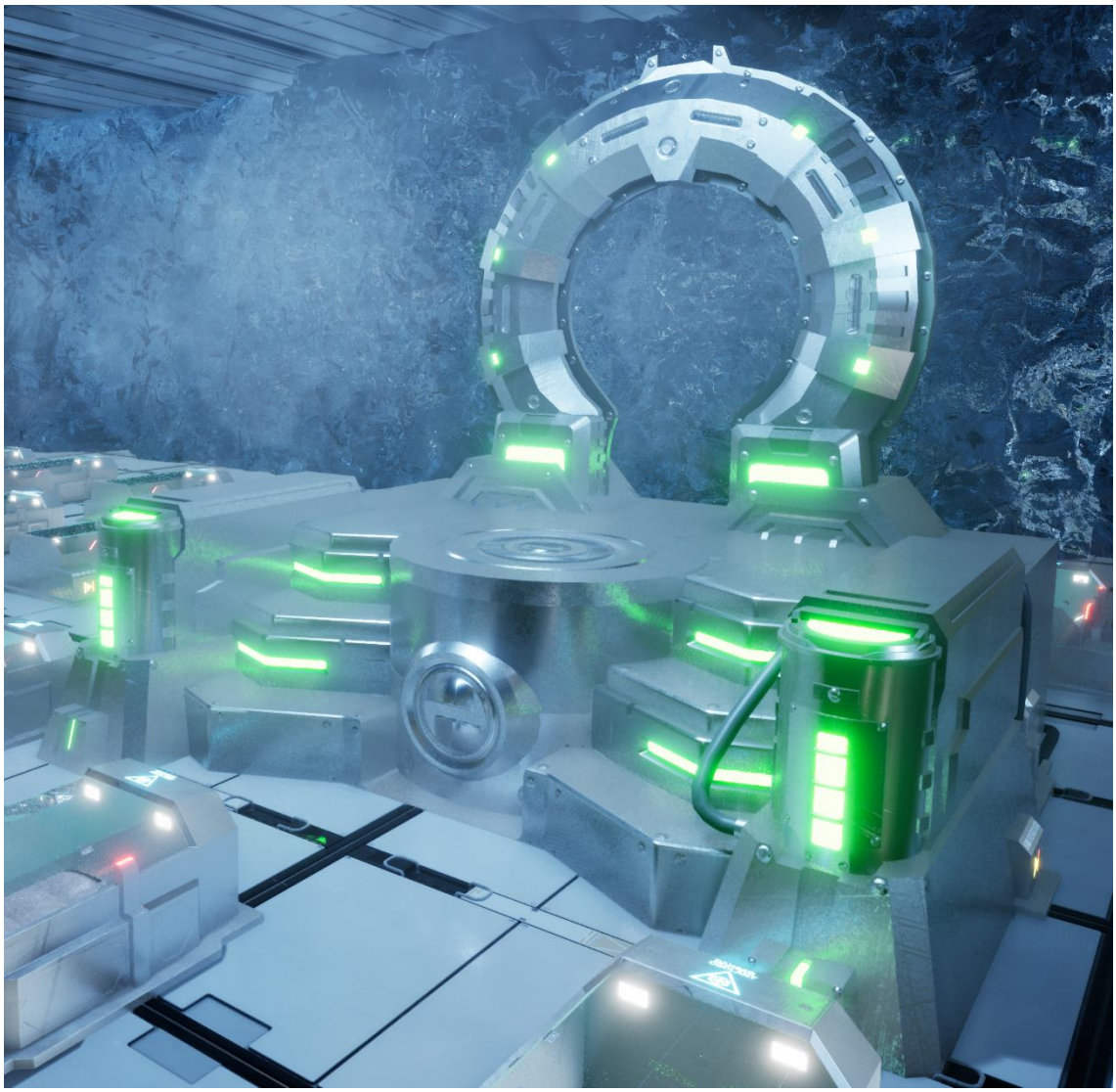

Figura 126 - *Asset* criado para servir de Portal no projeto (Imagem do autor).

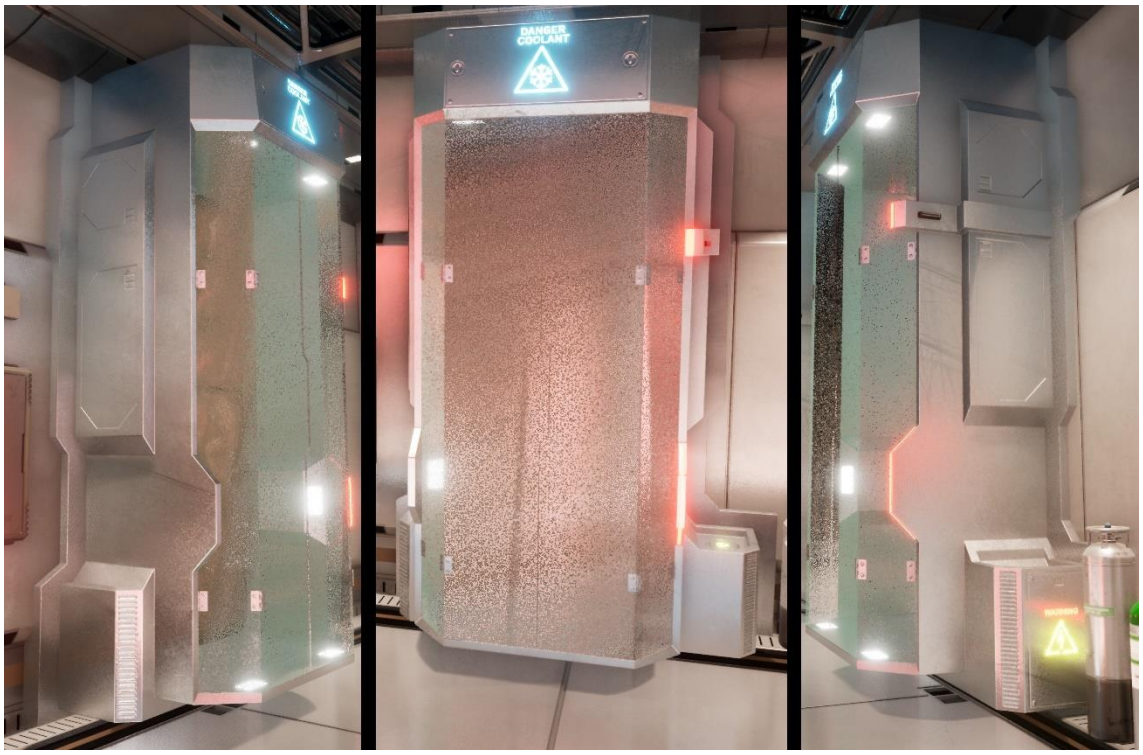

Figura 127 - *Asset* criado para servir de cápsula criogénica no projeto (Imagem do autor).

Figura 128 - Arma *retexturizada* para se enquadrar melhor com o resto dos *assets* do projeto (Imagem do autor).

Figura 129 - Autómato *retexturizado* para se assemelhar mais ao T-800 do filme The Terminator (James Cameron, 1984) (Imagem do autor).

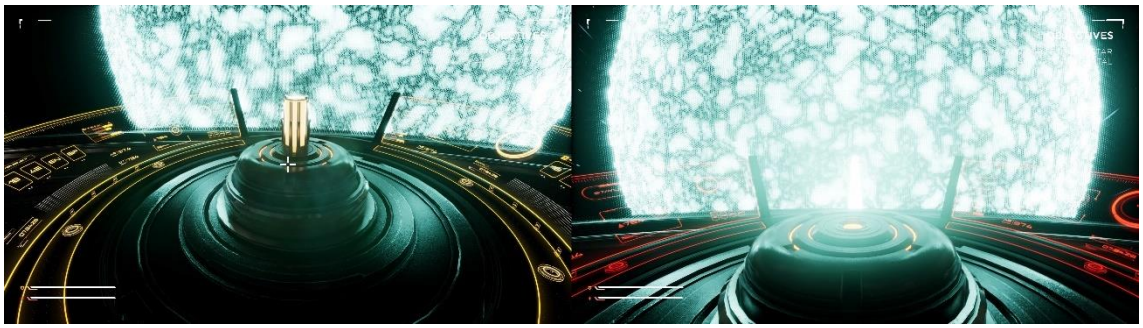

Figura 130 - Cenas do projeto onde se suga energia da estrela para encher a cápsula de energia (Imagem do autor).

Figura 131 - Cenas do projeto onde se observa o fim jogável do demo e a animação que leva a personagem através do portal (Imagem do autor).

Foi neste ponto que se realizou a entrega da dissertação do Henrique Candeias e onde se vai agora trabalhar, não só no que toca à devida implementação do post-Processing, mas também na implementação e melhoria de outros itens dentro do que for exequível e necessário.

5 Implementação do *post-Processing*.

Tendo o projeto sido realizado de forma a que cada um dos elementos participativos desenvolvia as suas devidas partes individualmente, e, ocasionalmente, ou quando a necessidade pedia, partilhavam-se as alterações, que eram incorporadas. Como tal, quando o projeto como está definido no capítulo anterior chegou às nossas mãos, havia alguns detalhes que precisavam de ser adicionados ou alterados de modo que este pudesse ser aberto para edição e adição de post-Processing.

Mediante isto, o primeiro passo necessário foi a obtenção do plugin *Assets Cleaner — Project Cleaning Tool* e do *plugin* FMOD, pois, sem estes plugins, o projeto não poderia ser aberto. Após se adquirirem os plugins e se instalarem na *engine*, importou-se o projeto que se encontrava na versão 4.24, para a versão 4.26 da Unreal Engine. Com o *upgrade* vieram, no entanto, algumas dificuldades relacionadas ao plugin FMOD no que tocava à configuração e realização devida do *upgrade* proveniente das maneiras como este pode ser colocado. No final, a maneira utilizada foi realizar a transferência do plugin e substituir todos os ficheiros no projeto, após se ter criado uma cópia. Assim sendo, de momento, encontrávamo-nos com duas versões do projeto: a 4,24, que se considera como *backup*, e a 4,26 que se utiliza a partir deste momento.

Com isto, a primeira alteração realizada foi de resolver um problema que ocorreu entre versões e envios de ficheiros, onde num dos níveis, havia um *item* em falta, nomeadamente um *decal*[66], como se pode observar na **Figura 132**. Assim sendo, foi necessário abrir uma versão extra do projeto, utilizada maioritariamente para testes e desenvolvimento do mapa e dos *assets* 3D, onde se encontrava o *decal* necessário.

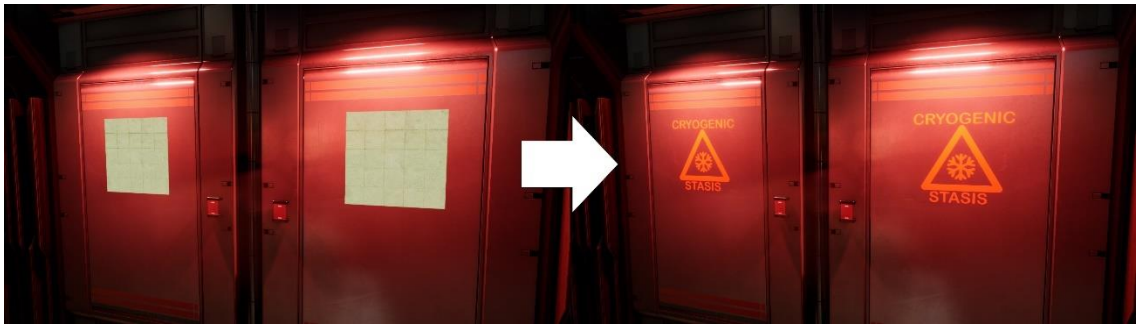

Figura 132 - Imagem ilustrativa do antes e após recolocar o decalque em falta (Imagem do autor).

[66] *Decals* ou decalques são materiais projetados em objetos 3D e que alteram ou adicionam detalhes ao mesmo, tendo em conta a própria geometria do objeto, assim como geometrias provenientes de *tesselation* e *normal maps*. Estes podem ser utilizados para, por exemplo, se adicionar um grafiti a uma parede de tijolo, sem que esta esteja fixa, ao contrário do que aconteceria se o grafiti fosse adicionado à textura do objeto 3D; caso fosse necessário modificar a localização, seria necessário recriar a textura.

De seguida, analisou-se novamente o projeto para se analisar todas as alterações que o projeto ainda poderia sofrer de modo a este ser melhorado, independentemente da possibilidade e capacidade de implementação. Assim sendo, apresenta-se a seguinte lista delineando essas alterações:

1. Remover a restrição de movimento no início do jogo;
2. Diminuir a distância de interação com os elementos;
3. Aumentar a velocidade dos inimigos para torná-los mais desafiantes e aumentar a sensação de perigo e tensão durante o jogo;
4. Reduzir a velocidade da personagem do jogador, especialmente quando esta não anda para a frente;
5. Rever as animações de movimento da personagem e dos inimigos;
6. Melhorar a leitura do texto nos *Tablets*;
7. Implementar um botão de saltar;
8. Melhorar controlos:
    - ESCAPE para poder sair do modo de leitura dos *tablets*;
    - Remover o C para assobiar;
9. Melhorar o *Loading* entre níveis;
10. Rever as ativações dos sons e músicas de fundo;
11. Implementar um mapa ou ajudas visuais de orientação;
12. Adicionar mais *assets* à área da tripulação para aumentar o realismo e utilidade da mesma mediante a *Lore*;
13. Rever os elementos de post-Processing e de iluminação.

Deste modo colocam-se como prioritários dois elementos no que toca a melhorias e/ou alterações: o post-Processing, devido ser o foco principal de estudo desta dissertação, e a iluminação, que está interligada com o post-Processing.

Para tal, abriram-se todos os níveis criados e removeram-se todos os volumes de post-Processing e *Reflection Captures*, e adicionou-se um elemento a cada nível conhecido como *SM_ColorCalibrator*[67], que iria ajudar à calibração de cores posteriormente.

De seguida, readicionou-se um volume de post-Processing, e, nas propriedades (*Settings*), confirmou-se a opção *Infinite Extent (Unbound)*, de modo a este se aplicar ao nível inteiro. Consequentemente, no painel de exposição, escolheu-se o método de metragem manual, e com o valor de compensação a 15, de modo a se poder utilizar a iluminação já incluída com o projeto sem a ter de refazer de raiz, o que levaria imenso tempo. Houve também a necessidade de corrigir o valor da iluminação da estrela anã branca utilizada no projeto.

[67] Para adicionar este elemento é necessário ter os elementos da *engine* disponíveis. Se estes não estiverem disponíveis, na janela de *Content Browser*, clique em *View Options*, e marque o item *Show Engine Content*.

Surge então o primeiro problema.

Quantos lumens produz a estrela, neste caso uma anã branca, especialmente à distância a que a mesma se encontra da nave?

Para tal, adicionou-se o primeiro ponto de luz necessário, uma *Directional Light*, visto que esta ajuda à ideia de que a estrela projeta realmente luz. Para tal, realizou-se uma pequena pesquisa onde se procurou o valor em lux do sol, e o tamanho de uma anã branca em comparação com o sol, tendo-se assim determinado que a luz solar fornece uma luminância de cerca de 120 000 (cento e vinte mil) lux (Jones, 2019), e que uma anã branca possui um tamanho ligeiramente maior que a terra (https://www.if.ufrgs.br/oei/stars/wd/wd_evol.htm), que, por si, possui cerca de 0,9% do tamanho do sol ("Solar System Sizes | NASA Solar System Exploration", 2003). Assim sendo, optou-se por ter uma anã com cerca de 1,44% do tamanho do sol, resultando numa anã com 10.050Km de raio (Sharp, 2017), e com isto acabou-se por aumentar o tamanho do *asset* referente à mesma e criar mais alguma distância da nave. Logo, a iluminação colocada na *Directional Light* foi de 1,44% dos 120 000 lux, resultando em 1728 Lux. No entanto, visto que se tem a compensação a 15, e há um vidro protetor que reduz a quantidade de luz que conseguiria penetrar na nave, o valor foi mais uma vez dividido, agora por 15, resultando em 115,2 Lux Ainda assim, a quantidade de luz era excessiva: afinal, apenas se dividiu uma vez, o que corresponderia a dividir pela compensação. Faltava agora considerar o vidro, que se queria que reduzisse a luminosidade em 99%, resultando assim no valor final de 1,152 (116,2×99%=114,048 — 115,2-114,048=1,152) Lux. Ajustou-se ainda a cor que se encontrava dentro de um tom branco para um tom mais ciano, aproximando-se assim de uma temperatura de cor dentro dos 25 000 (vinte cinco mil) kelvin. Para se ter uma ideia das cores que uma anã branca poderia ter, pode-se ainda observar a **Figura 133**.

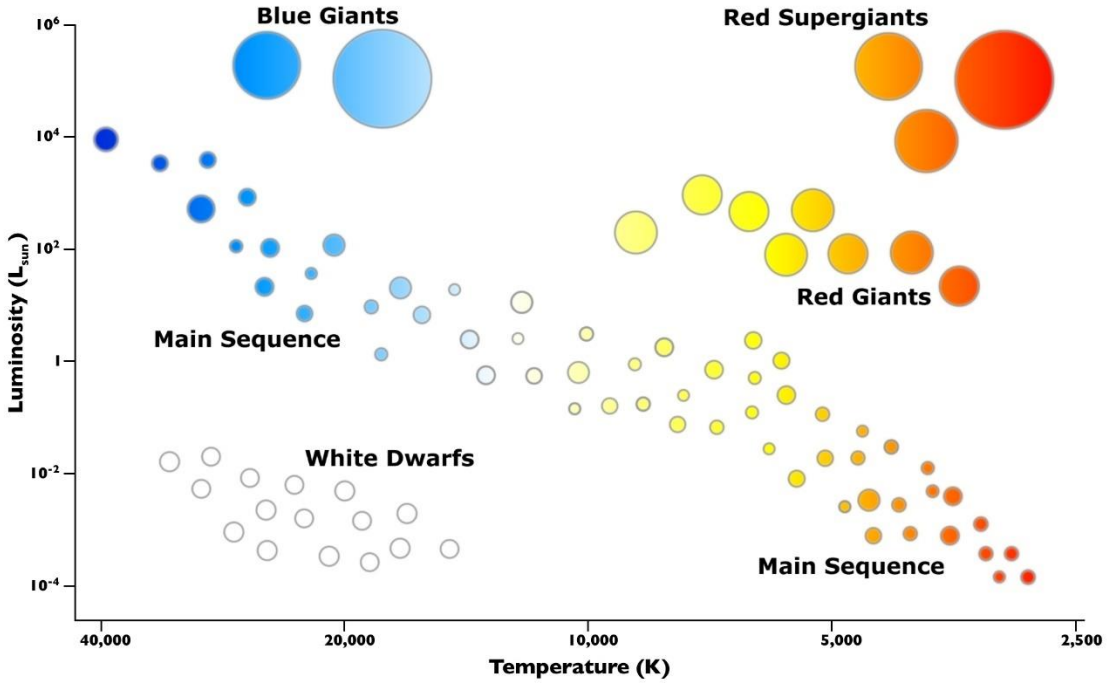

Figura 133 - Diagrama de Hertzsprung-Russell (s/i)

Realizaram-se ainda vários outros detalhes, obtendo-se assim, finalmente, a iluminação base desejada, como se pode observar na **Figura 134.** Através da **Figura 135** e da **Figura 136,** pode-se ainda constatar as diferenças entre o antes e o depois destas alterações, quer no interior, quer na estrela em si.

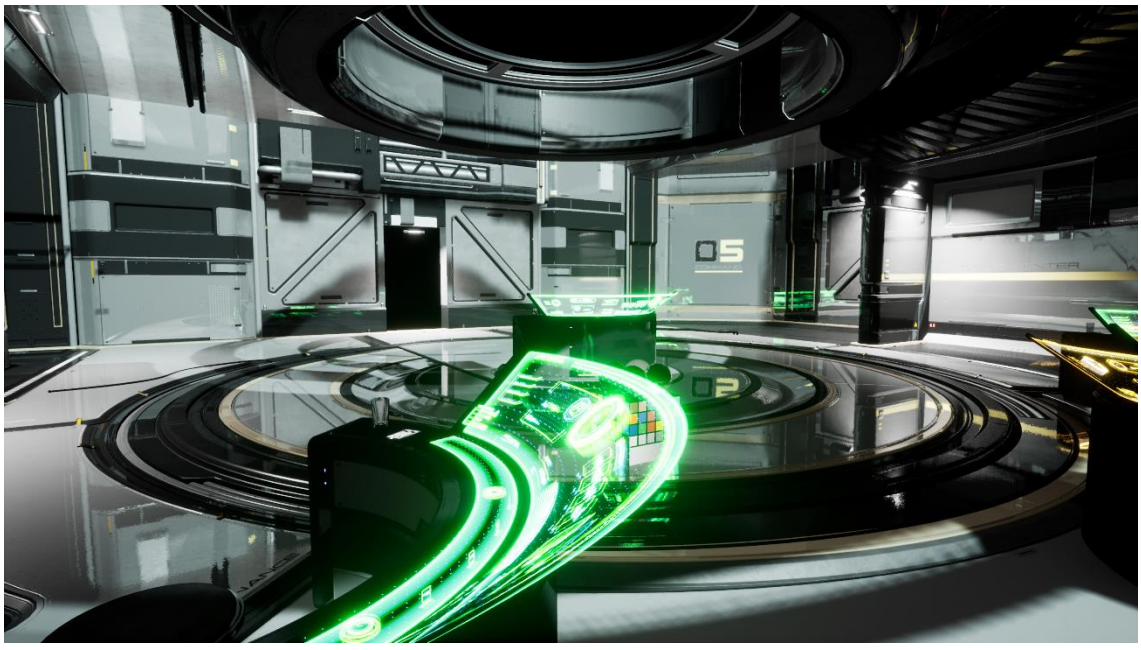

Figura 134 - Captura de tela do resultado da iluminação do projeto (Imagem do autor).

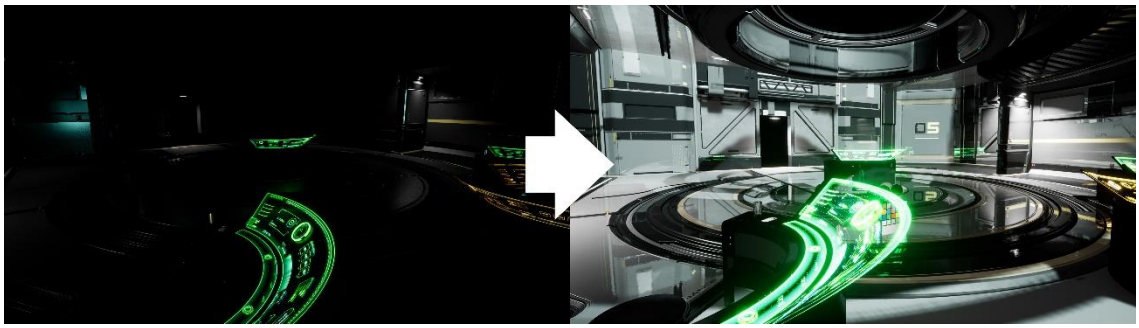

Figura 135 - Comparação da iluminação antes e após esta ser trabalhada (Imagem do autor).

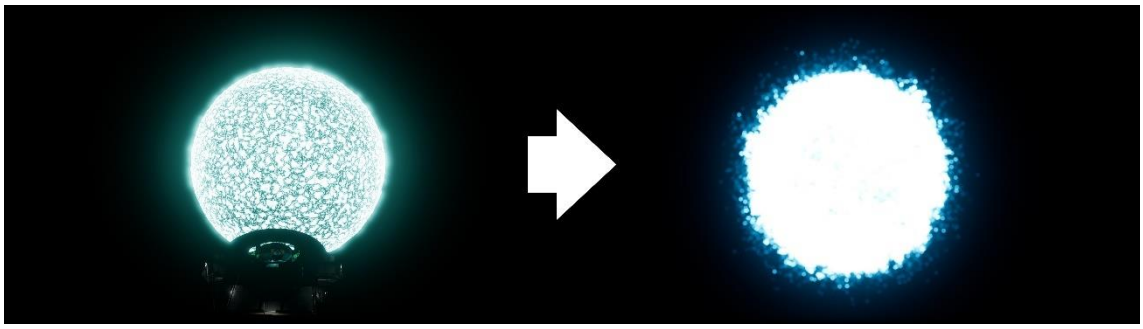

Figura 136 - Comparação da anã branca antes e após (Imagem do autor).

De seguida, ajustou-se o *Bloom*, de modo que este exista especialmente em torno de objetos e pontos muito luminosos, mas sem exageros. Para tal, mudou-se a intensidade para 0,5. A exposição alterou-se agora para automática através do histograma, visto que se pretende que o *gameplay* ocorra através dos olhos da personagem. Como tal, esta deve ser capaz de se adaptar à luminosidade de uma cena como o olho humano. Alteraram-se ainda os valores *de Speed Up* e *Down* para 10, de modo que adaptação não fosse totalmente imediata nem demasiado duradoura. A aberração cromática manteve-se desligada, assim como a *Dirt Mask*, câmara, *Lens Flare* e *Image Effects*. Isto tudo, devido a estes componentes serem relacionados a efeitos que ocorreriam em lentes fotográficas e não com o olho humano da personagem. Como tal, finalizou-se esta categoria com a profundidade de campo onde se optou por uma distância focal de 150 com um desfoque de 0,055 e (um) raio de 4, mantendo assim tudo o que se encontra a uma distância curta até média (focado, mas começando a desfocar ligeiramente a partir dai, sem desfocar demasiado ao ponto em que ficasse irreconhecível tudo o que estivesse a uma distância mais longa, isto dentro da distância máxima que se pode atingir com o jogo).

Prossegue-se para as propriedades de sintetização, onde se começou por ligar a oclusão ambiental com uma intensidade de 1 e um raio de 10, colocando-se a iluminação global com uma intensidade de 2, com a cor a assemelhar-se ao ciano emitido pela anã branca. Chegou-se então ao *motion blur*, um dos pontos que precisava de ser bastante melhorado mediante algumas das críticas recebidas pelos jogadores que testaram previamente o projeto. Neste ponto, queria-se então dar alguma ideia de *motion blur* proveniente do olho humano, pois este também sofre o efeito em certos momentos, como é o caso de olhar muito rapidamente à volta, ou no caso de tentar acompanhar vários pontos de interesse em movimento. Com

isto, decidiu-se colocar o Target fps a 42, sendo este um valor intermédio mediante o que foi encontrado como um valor comum para o olho humano, tendo em conta que, embora o olho humano apenas consiga ver verdadeiramente entre 7 a 13Hz, ou seja, entre o equivalente a 7 a 13 fps (Wiltshire, 2017). No entanto, outros estudos defendem que o olho humano pode chegar até aos 72 fps ("How many frames per second can the human eye see?", 2021). Estas diferenças costumam, no entanto estar relacionados com o facto de que o olho humano tem velocidades de perceção diferentes mediante o seu campo de visão, focal ou periférico, assim como o estilo de vida do indivíduo. Todavia, considerando que a personagem foi recentemente descongelada da cápsula criogénica, optou-se por um valor ligeiramente inferior, e acabou por se ficar com o Target fps a 37. Em relação ao resto, colocou-se o *Amount* a 1 e o *Max* a 0,5.

Segue-se para o *Screen Space Reflections*, onde se manteve a intensidade a 100, tendo-se porém aumentado a qualidade para 100, assim como o *Max Roughness* para 1, visto que o ambiente possui vários materiais refletores, e para não haver problemas de desempenho que se obriga a reduzir o valor, por exemplo, da qualidade. Continua-se com a percentagem do ecrã em termos do tamanho de sintetização, sobre a qual se encontrou um bom equilíbrio nos 115%, realizando-se assim uma sintetização com um pouco mais de qualidade, sem impacto notável no desempenho. Porém, este valor terá de ser revisto mais à frente, não só consoante o desempenho obtido individualmente em cada nível, mas também através do teste do projeto noutros sistemas informáticos para avaliar o impacto de desempenho que este possui. Concluiu-se este passo ao adicionar alguns *scene captures* e fazer a construção da iluminação do nível. (Alterações visíveis através da comparação na **Figura 137**)

Figura 137 - Comparação dos reflexos através das várias configurações testadas (Imagem do autor).

Finalizou-se com o *Ambient Cubemap*, que proporciona mais alguns detalhes nos reflexos da cena, mas apenas se o *cubemap* utilizado for correto. Para tal, criou-se um HDRi da cena, que se aplicou à textura do *Cubemap*, com a intensidade de 1,1. De seguida, criou-se novamente o HDRi da cena e substituiu-se o *Cubemap*, de modo a se obter um pouco mais de detalhes. Na **Figura 138** é possível observar as diferenças entre o uso dos HDRis e na **Figura 139** os HDRis criados.

Figura 138 - Comparação do uso de vários HDRis para melhorar os reflexos (Imagem do autor).

Figura 139 - HDRis utilizados na Figura 138 (Imagem do autor).

Seguiu-se então para o passo de *Color Correction* e *Tone Mapping* da cena e, como tal, abriram-se os painéis *Color Grading — WhiteBalance* e *Film* no volume de post-Processing e, finalmente, focou-se a câmara no objeto *SM_ColorCalibrator*, de modo a ver devidamente as alterações realizadas. O primeiro passo foi então mudar o valor de *Temp* em *White Balance* para 5250, dando-se assim um tom mais azulado à cena, correspondente à cor da luz proveniente da estrela, mas de forma subtil. No caso do *Tint*, colocou-se a -0,05 de modo a ganhar um tom ligeiramente esverdeado proveniente das luzes verdes das várias consolas em redor, e tendo em consideração também que muitos dos materiais e metais na cena têm tons azuis ou esverdeados, o que leva a que a cor da luz refletida mude ligeiramente. Finalizou-se com a alteração do *Shoulder* em *Film* para 0,475, de modo que os brancos na cena fossem menos acinzentados, sem necessidade de alterar o *Slope* ou o *Toe*, realizando-se assim pequenos ajustes à cena base, sem exageros, e sempre com possibilidade de, se estes mais à frente se revelassem incorretos, poder desligar os mesmos ou desligar os outros todos e regressar a estes, em vez de demorar a adicionar, configurar, remover e adicionar novamente pontos de luz ou mudar texturas consoante necessário.

No entanto, isto consistia em apenas um dos níveis. Como tal, este processo seria necessário ser realizado para todos os outros níveis, mas visto que ainda havia definições a alterar que iram ter impacto em todos os outros níveis, continuou-se com o melhoramento e com o elenco do processo e as alterações realizadas. Deste modo, todos os níveis seguintes apenas necessitam de "copiar" o que é feito a este nível inicial.

Foi neste ponto que se iniciou o trabalho nos restantes níveis, onde foi necessário adicionar alguns pontos extra de trabalho, nomeadamente no que toca à criação do *ambient cubemap*, visto que, para este ser correto, vários volumes tiveram de ser criados e colocados nas várias salas para se criarem vários *cubemaps*, Se este passo não fosse efetuado, os reflexos não coincidiriam com o que está dentro de cada sala. Mais ainda, quaisquer objetos que

se propusesse adicionar aos níveis deveria ser feito nesse ponto, sob risco de os reflexos não serem tão realistas quando o desejado. Logo, colocaram-se os vários objetos que estavam em falta em várias partes dos níveis, de forma a dar mais realismo. Só após isto se criaram os *cubemaps*.

A primeira destas alterações estava relacionada com a arma utilizada durante o *gameplay* do jogo, devido principalmente ao facto de esta não coincidir com o ambiente e a arte. Assim, essa foi substituída por outra pertencente a um *asset pack* disponível gratuitamente no mercado da Unreal Engine, como consta na **Figura 140**.

Figura 140 - Comparação do modelo da arma antes (à direita) e após (à esquerda) (Imagem do autor).

Continuando as alterações ao jogo, a próxima recaiu sobre um ponto de luz roxo na sala onde se apanha a arma, sendo a função deste, assim, sinalizar ao jogador onde a arma se encontrava. Embora constituindo uma pequena ajuda visual, esta não fazia sentido por ser a única luz roxa na sala, quando todas as outras eram simplesmente brancas, com a exceção das consolas. Como tal, a luz foi reposta de maneira a que ficasse branca como todas as outras, todavia perdendo-se com isso um ponto importante, que era conseguir dar uma ajuda ao jogador para este conseguir encontrar a arma facilmente. De forma a resolver esse problema, adicionou-se um efeito às luzes, assim como um efeito sonoro referente a uma luz a falhar, semelhante ao que é comummente utilizado em filmes e jogos de terror para dar um ar mais sombrio a certas partes dos ambientes. Desta forma (para resumir), eliminou-se uma luz que não coincidia com a arte visual do jogo, que foi substituída por uma que não só segue o estilo visual, mas que em termos diegéticos faz todo o sentido. Note-se, no entanto, que esta solução surgiu de modo a que não se perdesse demasiado tempo na resolução do problema, pois idealmente, e seguindo a diegese, vários pontos luminosos ao longo do jogo deveriam sofrer do efeito. No entanto, aí teria de se arranjar outros modos para fornecer um apoio visual ao jogador para auxiliar que este encontrasse a arma (como, por exemplo, ser o único ponto luminoso que não piscava / falhava, e que não produzia som nenhum).

Na sala de comando, foram ainda colocados mais alguns objetos no lado esquerdo da mesma, opostos ao lado onde a arma se encontrava para se preencher um pouco mais o espaço e não deixar tantas dúvidas ao jogador perante onde a arma se encontraria. Fez-se o mesmo para os corredores que levam o jogador entre a sala de comando e a sala da tripulação, de modo a evitar que os jogadores se perdessem, como era frequente até ao momento; inclusive, apagaram-se algumas luzes e partes do mapa que eram desnecessárias.

No segundo nível, ou seja, nos alojamentos da tripulação, foram adicionados alguns objetos, como pratos e comida, itens de higiene, e roupas, entre outros, alguns dos quais foram obtidos online[68][69].

Para além disso, ainda se alterou um material de vidro que confundia sempre os jogadores, pois era demasiado transparente.

Por fim, no último e terceiro nível, adicionou-se uma personagem[70] ao interior das cápsulas criogénicas que eram visíveis ao longo do jogo, mantendo vazias os que se encontravam na sala criogénica, visto ser quase impossível o jogador conseguir olhar em detalhe para dentro das mesmas; ademais, no caso de as cápsulas referidas estarem preenchidas, surgiria um impacto de desempenho severo.

Neste ponto, pode-se agora aplicar o *color grading* ao projeto. Assim sendo, fez-se uma análise de outros projetos e videojogos semelhantes no que toca ao *color grading*, assim como o tom geral do ambiente, contraste, luz, etc.

Após a análise dos jogos e filmes *System Shock* (Nightdive Studios, s.d), *Dead Space* (EA Redwood Shores, 2008), *Dead Space 2* (Visceral Games, 2011), *Dead Space 3* (Visceral Games, 2013), *Event Horizon* (Paul W. S. Anderson, 1997), *Alien Isolation* (Creative Assembly, 2014), *Alien* (Ridley Scott, 1979) e *Aliens* (James Cameron, 1986), eis o que se denotou:

Em System Shock (Nightdive Studios, s.d), denotou-se um ambiente escuro, mas sem ser em demasia, com tonalidades de cor geralmente azuladas, algum *bloom* e uma grande profundidade de campo, que passava totalmente despercebida. A exceção a isto revia-se em partes onde a quantidade de inimigos era mais elevada e, como tal, a tonalidade de cor mudava mais para tons de laranjas e vermelhos.

Dead Space (EA Redwood Shores, 2008) apresenta diferenças acentuadas, possuindo muito mais *bloom* do que System Shock (Nightdive Studios, s.d) e tonalidades de cor mais acastanhadas e ambientes menos escuros em certos momentos ou perto de estarem totalmente escuros, onde a única fonte de luz viável é a lanterna colocada na arma da personagem. No entanto, nas partes mais iluminadas, a tensão mantém-se devido a vários outros elementos que ajudam a obstruir os ambientes, como, por exemplo vapores. Há alguns momentos em que as tonalidades se afastam ligeiramente dos castanhos e mudam para outros como, por exemplo, tons esverdeados em zonas com uma densidade de vegetação elevada,

---

[68] "Oxygen Farm 1" (https://skfb.ly/6Tr6n) de Eliott com a licença de *Creative Commons Attribution* (http://creativecommons.org/licenses/by/4.0/).

[69] "Galactic Cola Vending Machine" (https://skfb.ly/onoSO) de ChrisCopeland3D com a licença de CC *Attribution-NonCommercial-ShareAlike* (http://creativecommons.org/licenses/by-nc-sa/4.0/).

[70] Scifi Girl v.01" (https://skfb.ly/HpVV) de patrix com a licença CC *Attribution-NonCommercial-ShareAlike* (http://creativecommons.org/licenses/by-nc-sa/4.0/)

ou tonalidades esbranquiçadas onde os inimigos se assemelham a fantasmas ou a gravidade é reduzida e possibilita que inimigos normais aparentem voar como fantasmas.

Ao transitarmos para Dead Space 2 (Visceral Games, 2011), o *bloom* é reduzido em comparação com o primeiro Dead Space (EA Redwood Shores, 2008), deixando assim de ser tão ofuscante, com tonalidades neutras, à exceção de alguns momentos como, por exemplo quando há um alarme e o jogador tem de se focar em sobreviver ao evento, mudando os tons para um laranja acastanhado. Em semelhança, o Dead Space *3* (Visceral Games, 2011) pouco muda em relação ao Dead Space 2 (Visceral Games, 2011).

Segue-se Event Horizon (Paul W. S. Anderson, 1997), o primeiro de três filmes analisados: No início, enquanto a situação se encontra normal, ou seja, ainda não há perigo, existe uma boa iluminação e, como tal, é fácil captar detalhes na cena. No entanto, à medida que a situação começa a ser agravada, a iluminação começa a diminuir, começando a ficar cada vez mais escuro, até que se chega ao ponto em que a escuridão é mais profunda, sendo iluminada a cena ocasionalmente através da forte iluminação dada pela tempestade, para mostrar detalhes como, por exemplo, as paredes que estão cobertas de sangue, mas que, devido à escuridão, não representam um detalhe que as personagens denotem imediatamente. Mais para o fim, depois de já estar instalado o terror e a ação começar a ser mitigada face ao mistério e a tensão, que crescem, começam-se a ver tons azuis para ajudar a transmitir o sentimento de calma, podendo-se também assim criar momentos de susto superiores, visto agora o espetador não estar à espera que aconteçam. Eventualmente, também estas tonalidades de azul são substituídas por vermelhos e laranjas, assim que o final se aproxima e o perigo começa a ser cada vez superior. É também importante saber que Dead Space (EA Redwood Shores, 2008) teve uma forte inspiração neste filme.

Passamos agora para os filmes Alien (Ridley Scott, 1979) e Aliens (James Cameron, 1986), antes de falarmos do videojogo Alien Isolation (Creative Assembly, 2014), por este ser inspirado nos filmes que o precedem. Começando com o primeiro filme, Alien (Ridley Scott, 1979), este começa com tons de azul, apoiando assim o sentimento de calma, mas também de distância, visto o filme se passar num futuro distante. Nota-se também um ambiente bastante escuro, que, após a introdução da tripulação, é substituído por tons brancos e áreas bem iluminadas, mantendo assim um sentimento de paz, limpeza, inocência entre outros. No entanto, não demora muito até que a iluminação volte a descer e o ambiente volte a ser escuro. Há, no entanto, uma cena breve em tons de amarelos e laranjas numa sala que aparenta ser relacionada com um computador, proporcionando assim uma ideia também de entusiasmo, extravagância, conhecimento entre outros. Entretanto, na cena em que a nave realiza uma aterragem, o tom de azul mantém-se. Neste caso, pode até considerar-se que este simboliza a confiança e união que a tripulação possui numa situação de turbulência. Algum tempo depois, após o primeiro ataque, a vítima é levada para uma sala semelhante a um bloco operatório, e novamente a boa iluminação reaparece durante alguns

minutos antes de a escuridão regressar gradualmente. Este contraste entre breves momentos de boa iluminação e a maioria de fraca iluminação propaga-se ao longo do filme. Para o fim do filme, existe ainda uma cena tonalizada em torno de tons amarelos e laranjas, proporcionados maioritariamente pela chama que ilumina a cena que podemos interpretar como um sinal de perigo. A partir desta cena, a luta entre os tons azuis e alaranjados é notável durante algum tempo, enquanto os azuis gradualmente superam os amarelos, desvanecendo estes, eventualmente, até um tom neutro que permanece até ao final.

Em Aliens (James Cameron, 1986), é notável que os tons azulados e a fraca iluminação dão a abertura do filme, ainda ocorrente no espaço onde a personagem principal, Ripley, se encontrava em suspensão criogénica. Assim que a personagem é acordada, os tons neutros tomam posse, assim como a boa iluminação. Após algum tempo, os tons azuis regressam e eventualmente também a fraca iluminação. Mais à frente, aproximadamente quando se chega a dois terços do filme, os tons mudam bruscamente para vermelho, enquanto as personagens tentam sobreviver na escuridão aos *xenomorphs* que as perseguem. Algum tempo depois, os tons de vermelho voltam a dar mais uma vez lugar aos tons azulados.

No caso do videojogo Alien Isolation (Creative Assembly, 2014), os tons de azul também são predominantes, assim como a fraca iluminação, que neste caso acaba por ser mais forte que a existente nos filmes onde amiúde a iluminação era tão fraca que quase toda a cena era preta. Há também algumas partes em que a tonalidade se torna mais para os laranjas, normalmente durante zonas de alerta, mas de um modo muito subtil.

Relativamente ao projeto, decidiu-se que o aspeto geral exigia, antes de mais, o tratamento da quantidade de luz que este possui, que é imensa, algo que até serviria caso o estilo fosse de ação orientada para exploração ou aventura, mas não quando este se enquadra nos géneros de ação em contexto de furtividade e terror. Começa-se por aumentar o contraste, reduzir a saturação e o *gamma*, ligeiramente, e o ganho (este, de forma dramática). Isto aplica-se também a todos os níveis do jogo. Já nas tonalidades, o *menu* e o primeiro nível receberam uma tonalização entre o azul e o roxo, para simbolizar o frio do espaço e dos corredores em volta, assim como para realçar o ar de mistério. O segundo nível, por ser onde a maioria dos robôs se encontra e, como tal, onde a maioria do perigo existe, aumentaram-se os tons de amarelos e vermelhos, visto estes representarem perigo. Por fim, o terceiro e último nível recebeu um tom ainda mais azulado do que o primeiro, visto este se desenrolar na sala criogénica e, como tal, onde a maioria do frio se concentra. No entanto, estas alterações de tonalidade de cores, saturação, *gamma* e contraste são mais enquadradas na dimensão artística do que no realismo. Isto não significa que estes domínios sejam totalmente opostos, ou que não possa haver uma visão artística para haver realismo e vice-versa: neste caso, as duas até estão bastante interligadas, visto que, para além das cores utilizadas que vão proporcionar uma certa emoção ao jogador, estas aliam-se também ao ambiente, pelo que tonalidades mais quentes se encontram em partes do jogo em que a temperatura é realmente superior, e tonalidades mais frias se encontram em partes da nave normalmente

mais frias. No final, ainda se ajustou a quantidade de luz das cenas, de modo a que ficassem ainda mais escuras. As alterações podem ser vistas através da **Figura 141**, onde a primeira (à esquerda) parte se enquadra sem *color grading*, a segunda (ao centro) com *color grading* e a última (à direita) com a luz ajustada.

Figura 141 - Comparação das alterações realizados no projeto como *color grading* e posteriormente de iluminação (Imagem do autor).

No final, ainda foram revistos alguns outros assuntos, nomeadamente a *interface* do *menu*, que foi alterada de modo a se enquadrar melhor na estética já existente, assim como a estrela que também foi refeita, agora utilizando o sistema de partículas *Niagara*, de modo que esta se assemelhasse mais a uma estrela e menos a um planeta gasoso com uma forte emissão de luz. Estas alterações podem ser observadas na **Figura 142**. Mais ainda se reduziu a *screen percentagem*, que se colocou previamente a 115% e foi agora reduzida para 100% no *menu* e 85% durante o decorrer dos níveis, fazendo com que o impacto não seja tão grande no desempenho, já que se observou que o desempenho sofreu bastante após todas as alterações realizadas. Com isto, conseguiu-se, por exemplo, passar, no editor, de cerca de 3fps para 23fps, ou seja, passando de uma experiência praticamente impossível de jogar para uma que já possibilitava algum controlo, sem se perder demasiada fiabilidade gráfica como se constata na **Figura 143**. Foram ainda feitos mais alguns ajustes menores de modo a que o desempenho melhorasse mais ainda, na tentativa de que se chegasse ou, idealmente, superasse os 30fps em todas as áreas do projeto, algo atingido com sucesso visto ter-se atingido uma média de 40fps ou mais.

Foi ainda colocada a hipótese de implementar *ray-tracing* no projeto, mas, devido à exigência do projeto no *hardware* para que o desempenho fosse aproveitável, e a falta de *hardware* potente o suficiente para se conseguir trabalhar, o *ray-tracing* permanece desligado.

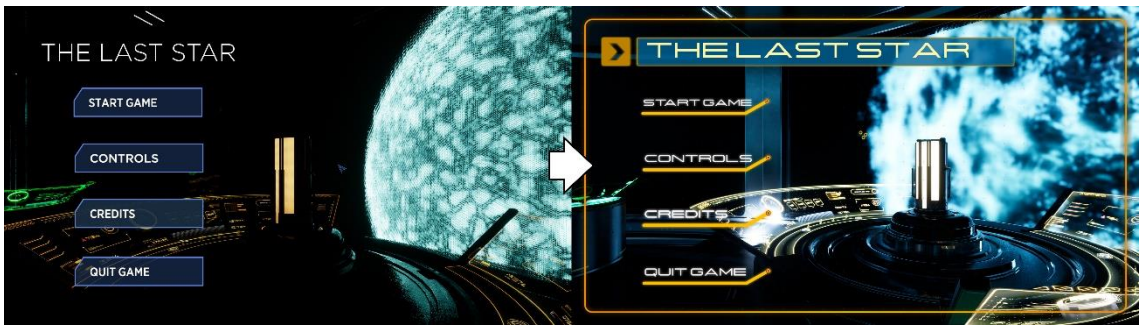

Figura 142 - Comparação das alterações realizadas ao menu e à Anã Branca. (Imagem do autor).

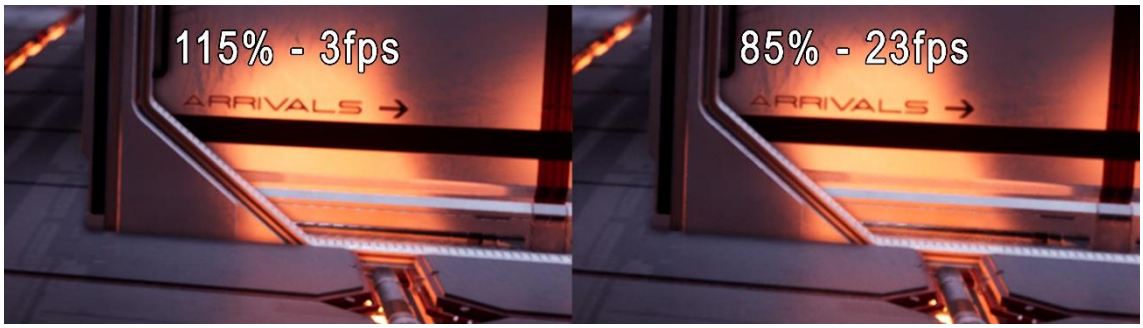

Figura 143 - Comparação da percentagem e a relação com o aumento de frames por segundo (Imagem do autor).

Com isto, nem todos os pontos de melhoria foram realizados, por falta de tempo, capacidade e conhecimento. Apresentam-se os que ficam por alterar ou implementar:

1. Diminuir a distância de interação com os elementos;
2. Aumentar a velocidade dos inimigos para se transmitir melhor a ideia de terror;
3. Reduzir a velocidade da personagem do jogador, especialmente quando esta não anda para a frente;
4. Rever as animações de movimento da personagem e dos inimigos;
5. Implementar um botão de saltar;
6. Melhorar alguns dos controlos:
    - ESCAPE para poder sair do modo de leitura dos *tablets*;
    - Remover o C de assobiar;
7. Melhorar o *Loading* entre níveis;
8. Rever as ativações dos sons e músicas de fundo;
9. Implementar um mapa;

Tendo-se realizado então 4 das 13 alterações necessárias, deu-se por concluído o desenvolvimento atual do projeto, no que toca essencialmente ao necessário para o seu contexto na dissertação, sabendo que este ainda tem um longo caminho a percorrer até se conseguir considerar um projeto com qualidade suficiente para ser globalmente distribuído sem a necessidade de atualizações ou resolução de problemas.

Presentemente, o demo jogável do projeto pode ser encontrado em https://sardinelord.itch.io/the-last-star ou através do código QR ao lado, assim como versões anteriores do mesmo, a par com os detalhes das alterações produzidas ao longo do desenvolvimento do mesmo até este atingir a sua forma final.

6 Conclusão do projeto e alterações futuras.

Após a conclusão de todas as alterações e atualizações possíveis de realizar mediante o tempo e capacidades da equipa, o projeto foi novamente disponibilizado de modo a receber-se comentários não só sobre a implementação do post-Processing, mas também com outras alterações como a implementação de mais objetos 3D nas cenas.

Assim sendo, adiciona-se à lista de itens a implementar:

1. Implementação de otimizações de desempenho para que este corra melhor e numa maior quantidade de *hardware*;
2. Implementação de DLSS ou preferencialmente FSR devido ao segundo não precisar de *hardware* específico;
3. Itens destruíveis;
4. Mais interações para além dos *tablets* de informação;
5. Revisão total do *menu* inicial e das suas opções tendo em consideração uma melhor experiência de utilizador;
6. Revisão da *interface* de modo a esta se alinhar melhor com a experiência do utilizador;
7. Não remover imediatamente os corpos dos inimigos;
8. Pontos de dano diferentes nos inimigos;
9. Maior variedade de inimigos;
10. Trocar o efeito do portal por partículas do *niagara;*

Mais ainda fica a possibilidade de atualizar o projeto para unreal engine 5 de modo a usufruir do *Lumen*, assim como os seus grafismos e desempenhos superiores. Para tal, será necessário que os plugins utilizados ao longo do desenvolvimento sejam atualizados para suportar a atualização da *engine*. Em alternativa, poderá ser necessário substituir os plugins por outros ou criar conteúdo para essa tarefa.

Conclusão

Como foi falado de uma forma breve e sucinta no capítulo 4 da parte teórica (Conclusão da importância do *post-Processing*), a conclusão é de que a importância do *post-Processing* está mais que estabelecida, sendo a exceção a esta importância fatores limitados como o objetivo pretendido, capacidades da equipa, e tempo, mas estes são fatores que se aplicam a todos os elementos constituintes do desenvolvimento de um jogo. Relembre-se, por exemplo, o videojogo Cyberpunk 2077 (CD Projekt RED, 2020), que, durante a sua fase de desenvolvimento, prometia ter uma vasta quantidade de elementos; no momento de lançamento, todavia, vários desses elementos não chegaram a ser implementados, como por exemplo o sistema de comboio metropolitano e a personalização de veículos (Dane, 2020).

Excluindo então esses fatores, a sua importância, mais uma vez, é bem definida. Efeitos pertencentes ao *post-Processing* estão inclusive presentes em situações do dia a dia de um indivíduo comum sem este sequer se aperceber. Por exemplo, veja-se que o *Anti-Aliasing* é utilizado numa vasta gama de situações para eliminar artefactos gráficos, mais especificamente na suavização da pixelização de certos itens. Para dar um exemplo cabal, este é utilizado em quase todos os sistemas operativos informáticos na apresentação de letras no ecrã, suavizadas, em vez de as apresentar pixelizadas.

Realizando uma análise conclusiva mais profunda do *post-Processing*, seja o *Anti-Aliasing*, o *color grading*, a *color correction* ou outros efeitos do post-Processing, as capacidades do *post-Processing* como uma ferramenta versátil e essencial estão bem estabelecidas. Ainda assim, a questão da sua importância no desenvolvimento não desvanece totalmente. Como foi estudado, usufruir da ferramenta para obter bons resultados não é simples. Esta afirmação pode ser comprovada pela quantidade de videojogos que exageram na quantidade de efeitos deste género, sendo a utilização de alguns destes efeitos aplicados em situações sem sentido, o que acaba por dar origem um julgamento relativo à importância desta ferramenta. Portanto, o poder da ferramenta não está em questão, mas sim a aprendizagem requerida para se dispor de qualidades técnicas suficientes para usufruir de maneira produtiva e consistente do post-Processing.

Tendo em conta que o problema se encontra relacionado com a aprendizagem da metodologia necessária para a boa utilização, e não com a utilização prática, considere-se, por exemplo, que de modo a criar outros algoritmos de Anti-Aliasing cada vez melhores, se requer um conhecimento das linguagens de programação relevantes ao produto, suficientes para se conseguir criar um algoritmo cuja qualidade e desempenhos sejam superiores aos utilizados atualmente. Outro exemplo vê-se na utilização de filtros relacionados com as câmaras, como a exposição, o ISO e a profundidade de campo, entre outras, que requerem que se tenha algum conhecimento prévio de como utilizar uma câmara fotográfica no modo manual.

Mais ainda, é importante saber que o post-Processing é sempre aplicado a imagens já existentes, não conseguindo este criar imagens do nada. Isto pode levar a pensar-se que a importância do post-Processing seja inferior a outros pontos do desenvolvimento dos videojogos, como programação, modelos 3D, gráficos da interface, som, etc. Mais especificamente, pode ser considerado que a sua importância é inferior nos campos visuais de desenvolvimento de um videojogo. No entanto, considerar a sua importância como inferior apenas porque este precisa de algo ao qual seja aplicado, não é correto. Veja-se que um jogo funciona tipicamente sobre várias linhas de código, e por muito poucas linhas que sejam, são estas que proporcionam a interação ao jogador. Sem código, um jogo seria uma experiência *media* sem interatividade, resultando em narrativas semelhantes às de um filme, animação ou apresentação de imagens. Mas não é por isso que se pode dizer que a programação é superior a todos os outros campos do desenvolvimento de um videojogo. Um caso análogo é imaginar um videojogo sem qualquer informação gráfica; aliás, a necessidade de haver algo gráfico é subentendida no próprio nome "videojogo". Sem vídeo, seria apenas um jogo, neste caso, desprovido do sentido da visão.

Desta forma, contabilizar a importância do post-Processing pode recair sobre o produto em desenvolvimento. Este precisa de *color grading*? Ou de *Anti-Aliasing*? Ou de *Post Process Materials*? Para o caso de as respostas serem negativas, então a importância do post-Processing nesse produto é pouca ou nula. Por outro lado, caso os desenvolvedores queiram adicionar um visual mais realista ou estilizado, a importância já tem de ser considerada. E se o produto necessitar de um determinado efeito por razões para além do visual, como, por exemplo, constituir parte da narrativa e da jogabilidade, aí o post-Processing é imprescindível.

Há ainda que ter em linha de conta que um artista nesta área, ao longo da sua carreira, irá muito provavelmente desenvolver uma variedade de videojogos, pelo que este deve, pelo menos, ter o conhecimento básico mínimo sobre *post-Processing*, pois os videojogos em que trabalhará podem ou não necessitar de post-Processing, e, especialmente em equipas pequenas, a hipótese de haver outro artista para tratar desse assunto são pequenas.

Resumindo, é importante que qualquer artista visual, no desenvolvimento de videojogos, saiba utilizar o post-Processing, nem que seja de uma forma básica. Este conhecimento básico pode consistir apenas na consideração de se um efeito deve ser utilizado ou não (mesmo com as configurações pré-definidas). Idealmente, o passo seguinte seria o estudo do *color grading* e de *color correction*, algo que, em norma, se o desenvolvedor se enquadrar mais na parte artística visual de um projeto (invés de programação ou som), já deveria existir no seu arsenal de conhecimento, faltando apenas adotar esse conhecimento à técnica do *post-Processing*.

Com isto, a sua importância acaba por parecer relativa, dependendo do ponto de vista do videojogo e do artista. Porém, a sua importância não pode ser negada, mesmo que não se

recorra à sua utilização. Ademais, no caso dos artistas, esta pode não ser uma prioridade de aprendizagem inicial, quando estes ainda estão a começar a sua carreira e a aprender a desenhar, programar, modelar, animar, etc. Contudo, assim que estes se encontrem perto de dominar a sua área de foco, a aprendizagem do post-Processing passa a ser um ponto importante. No entanto, é claro que, quando e se o artista aprende a utilizar e a recorrer ao *post-Processing*, a utilização do mesmo vai depender sempre de si e dos objetivos que este tem de futuro. Afinal de contas, o artista pode querer dedicar-se apenas a programar, ou apenas a modelar modelos 3D sem sequer os animar, texturizar etc. Mas aqui fica também outra questão, mais de autoanálise: quais são os objetivos de futuro do artista? O que é que este pretende aprender e o que é que pretender alcançar? Veja-se que, normalmente, se o artista tiver como objetivo na sua carreira supervisionar uma equipa, este não pode ser bom numa só área; pode até ser um especialista numa determinada área, mas convém que possua bases de outras áreas, de modo a conseguir comunicar eficientemente com todos os membros da sua equipa e de outras com que trabalhe.

A conclusão é, portanto, que, no desenvolvimento de videojogos, mesmo que não seja utilizado, o *post-Processing* é uma ferramenta importante, e questões relacionadas à sua aprendizagem merecem um estudo dedicado e independente.

Referências

Abrash, M. (2000). Ramblings in Realtime. Acedido a 18 de setembro de 2021, https://www.bluesnews.com/abrash/contents.shtml

ACMI. (s.d.). A Visit to the Seaside | 1908 | ACMI collection. Acedido a 5 de outubro de 2021, https://www.acmi.net.au/works/60833--a-visit-to-the-seaside/

Aldredge, J. (2019). Explore the Basics of Color Grading. Acedido a 3 de outubro de 2021, https://www.rocketstock.com/blog/color-grading-basics/

Anagnostou, K. (2019). Hybrid screen-space reflections. Acedido a 30 de setembro de 2021, https://interplayoflight.wordpress.com/2019/09/07/hybrid-screen-space-reflections/

Arvi VR INC. (2018). The Meaning of Anti-Aliasing: FXAA, SMAA, MSAA, SSAA, TXAA Algorithms. Acedido a 30 de setembro de 2021, https://vr.arvilab.com/blog/anti-aliasing

Beets, K., & Barron, D.L. (2000). Super-sampling Anti-aliasing Analyzed.

Beggan, P. (2017). This is Why You Should Post-Process Your Photography. Acedido a 30 de setembro de 2021, https://petapixel.com/2017/01/04/post-process-photography/

Bernacki, J. (2020). Automatic exposure algorithms for digital photography. Multimedia Tools and Applications, 79(19–20), 12751–12776. https://doi.org/10.1007/s11042-019-08318-1

Blender Guru. (2018). Lessons from 7 years of VFX. An interview with Wren Weichman of Corridor Digital [Video file]. https://www.youtube.com/watch?v=XOXeRNw9XX4

BorrowLenses. (2018). A Guide to Color Grading Your Video in 5 Easy Steps. Acedido a 30 de setembro de 2021, https://www.borrowlenses.com/blog/color-grading/

Brackeys. (2020). EVERY Image Effect in Unity Explained - Post Processing v2 Tutorial [Vídeo]. https://www.youtube.com/watch?v=9tjYz6Ab0oc

Brown, J. (2016). An Introduction to Color Grading. Acedido a 30 de setembro de 2021, https://www.bhphotovideo.com/explora/video/tips-and-solutions/introduction-color-grading

Bunyan, M. (2013). Exhibition: 'faking it: manipulated photography before photoshop' at the metropolitan museum of art, new york. Acedidod a 3 de outubro de 2021, https://artblart.com/tag/henry-peach-robinson-fading-away/

Cabading, Z. (2019). Anti-Aliasing: Everything You Need to Know. Acedido a 30 de setembro de 2021, https://www.hp.com/us-en/shop/tech-takes/what-is-anti-aliasing

Cambridge in Colour. (s.d.). Understanding Camera Lens Flare. Acedido a 30 de setembro de 2021, https://www.cambridgeincolour.com/tutorials/lens-flare.htm

CaseGuard. (2021). How many frames per second can the human eye see? Acedido a 3 de outubro de 2021, https://caseguard.com/articles/how-many-frames-per-second-can-the-human-eye-see/

Cellini, M. (2019). Introduction to Ray-Tracing in Unreal Engine. Acedido a 3 de outubro de 2021, https://80.lv/articles/introduction-to-ray-tracing-in-unreal-engine/

Chia, N., Cant, R. & Al-Dabass, D. (2001). New Anti-Aliasing and Depth of Field Techniques for Games Graphics.. 115-.

Chikhani, R. (2015). TechCrunch is now a part of Verizon Media. Acedido a 21 de setembro de 2021, https://techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/?guccounter=1

Cilia, D. (2020). Four friars. . . not five. Acedido a 30 de setembro de 2021, https://timesofmalta.com/articles/view/four-friars-not-five.805299

Cladera, A. (2015). Depth of Field: The Definitive Photography Guide. Acedido a 30 de setembro de 2021, https://www.photopills.com/articles/depth-of-field-guide

Coe, B. (1978). Colour Photography: The First Hundred Years, 1840–1940 (1ª ed.). Ash & Grant.

Cook, D. A. (1996). A History of Narrative Film (3ª ed.). http://144.214.21.63/CCS/etexts/more/film_culture/history_of_narrative_film/A_History_of_Narrative_Films.pdf

Cox, S. (2019). Hyperfocal Distance Explained. Acedido a 30 de setembro de 2021, https://photographylife.com/landscapes/hyperfocal-distance-explained

Dane, K. (2020). Cyberpunk 2077 loses content: Developers cut metro and car tuning – FREEMMORPG.TOP. Acedido a 5 de outubro de 2021, https://freemmorpg.top/cyberpunk-2077-loses-content-developers-cut-metro-and-car-tuning/

Davison, J. (2016). How 'Quake' Changed Video Games Forever. Acedido a 3 de outubro de 2021, https://www.rollingstone.com/culture/culture-news/how-quake-changed-video-games-forever-187984/

Denning, R. (2021). The origins of colour grading. Acedido a 30 de setembro de 2021, https://www.redsharknews.com/the-origins-of-grading

DigitalRev / In-Focus. (2016). A Visual History of Adobe Photoshop [Video file]. https://www.youtube.com/watch?v=cwk3O19rItQ

Donovan, T. (2010). Replay: The History of Video Games. Acedido a 21 de dezembro de 2021, https://booksvooks.com/fullbook/replay-the-history-of-video-games-pdf.html

Dunlop, J. What Is Lens Flare? (And How to Use It for Creative Photos!). Acedido a 30 de setembro de 2021, https://expertphotography.com/lens-flare/

E. (2016). Understanding Depth of Field – A Beginner's Guide. Acedido a 30 de setembro de 2021, https://photographylife.com/what-is-depth-of-field

Eiseman, L. (2017). The Complete Color Harmony, Pantone Edition: Expert Color Information for Professional Results. [ePub] Obtido em https://www.amazon.com/Complete-Color-Harmony-Pantone-Professional/dp/1631592963

Epic Games, Inc. (s.d.). Ambient Cubemaps. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/PostProcessEffects/AmbientCubemap/

Epic Games, Inc. (s.d.). Anti-Aliasing. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/AntiAliasing/

Epic Games, Inc. (s.d.). Auto Exposure (Eye Adaptation). Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/AutomaticExposure/

Epic Games, Inc. (s.d.). Bloom. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/Bloom/

Epic Games, Inc. (s.d.). Cinematic Depth of Field Method. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/DepthOfField/CinematicDOFMethods/

Epic Games, Inc. (s.d.). Color Grading and Filmic Tonemapper. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/ColorGrading/

Epic Games, Inc. (s.d.). Depth of Field. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/DepthOfField/

Epic Games, Inc. (s.d.). Lens Flare. Acedido a 30 de Setembro de 2021, https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/LensFlare/#:%7E:text=Simulating%20scattered%20light%20from%20bright,to%20imperfections%20in%20camera%20lenses

Epic Games, Inc. (s.d.). Lit Translucency. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.26/en-US/BuildingWorlds/LightingAndShadows/LitTranslucency/

Epic Games, Inc. (s.d.). Mesh Distance Fields. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.26/en-US/BuildingWorlds/LightingAndShadows/MeshDistanceFields/

Epic Games, Inc. (s.d.). Mobile Depth of Field Method. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/DepthOfField/MobileDOFMethods/

Epic Games, Inc. (s.d.). Path Tracer. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/RayTracing/PathTracer/

Epic Games, Inc. (s.d.). Post Process Effects. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/

Epic Games, Inc. (s.d.). Post Processing Content Examples. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.27/en-US/Resources/ContentExamples/PostProcessing/

Epic Games, Inc. (s.d.). Ray Tracing Features Settings. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/RayTracing/RayTracingSettings/

Epic Games, Inc. (s.d.). Real-Time Ray Tracing. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/RayTracing/

Epic Games, Inc. (s.d.). Screen Space Reflections. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/ScreenSpaceReflection/

Epic Games, Inc. (s.d.). Using Lookup Tables (LUTs) for Color Grading. Acedido a 30 de setembro de 2021, https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/UsingLUTs/

Epic Games, Inc. [Unreal Engine]. (2019, November 1). Color Correction | Tips & Tricks | Unreal Engine [Video file]. https://www.youtube.com/watch?v=0HMFczWSRig

Ferreira, M. (2014). Os tipos de Anti-Aliasing existentes e a sua diferença - FXAA, SMAA, MSAA e outros. - PCManias.com. Acedido a 18 de setembro de 2021, http://www.pcmanias.com/os-tipos-de-anti-aliasing-existentes-e-a-sua-diferenca-fxaa-smaa-msaa-e-outros/

Flueckiger, B. (2012). Kodachrome used for duplication. Acedido a 30 de setembro de 2021, https://filmcolors.org/timeline-of-historical-film-colors/

Fusco, J. (2016). The Psychology of Color in Film (with examples). Acedido a 3 de outubro de 2021, https://nofilmschool.com/2016/06/watch-psychology-color-film

GameSpot. (2017). Bloom, HDR, And HDR Displays - PC Graphics Settings Explained [Video file]. https://www.youtube.com/watch?v=02t_75b90cg

Gonçalves. S. & Almeida. F. (2019) The Unnoticed Importance of post-Processing in Videogame Development. Senses & Sensibility'19: Lost in (G)localization. Proceedings of the UNIDCOM 10th International Conference. 445-454

Grixti, S. (2017). GT Sport's 'Scapes' Is The Most Insane Photo Mode Imaginable. Acedido a 18 de setembro de 2021, https://press-start.com.au/features/2017/10/17/gt-sports-scapes-insane-photo-mode-imaginable/

Hannavy, J. (2007). Encyclopedia of Nineteenth-Century Photography (1ª ed.). http://home.fa.utl.pt/~cfig/Anima%E7%E3o%20e%20Cinema/Fotografia/Enciclopedia%20of%20the%2019th%20Century%20Photography.pdf

Hearn, P. (2019). What is Path Tracing and Ray Tracing? And Why do They Improve Graphics?. Acedido a 18 de setembro de 2021, https://www.online-tech-tips.com/computer-tips/what-is-path-tracing-and-ray-tracing-and-why-do-they-improve-graphics/

Heller, E. (2012). A psicologia das cores: Como as cores afetam a emoção e a razão (1a edição, 8a impressão ed.). Barcelona, Espanha: Editorial GG, SL.

Hellerman, J. (2019). How a Film Color Palette Can Make You a Better Filmmaker [W/ Infographics]. Acedido a 30 de setembro de 2021, https://nofilmschool.com/Film-color-theory-and-color-schemes

Hellerman, J. (2020). What's the Difference Between Color Correction and Color Grading? Acedido a 30 de setembro de 2021, https://nofilmschool.com/color-grading-vs-color-correction-process

Hirsch, R. (2017). Seizing the Light: A Social & Aesthetic History of Photography (3ª ed.). Abingdon, Reino Unido: Routledge.

History of Art: History of Photography. (s.d.). Acedido a 30 de setembro de 2021, http://all-art.org/history658_photography1.html

Hummel, A. W., & Needham, J. (1962). Science and Civilization in China. Volume IV, Physics and Physical Technology. Part I, Physics. The American Historical Review, 4(6), 463. Retirado de https://web.archive.org/web/20170703010030/https://monoskop.org/images/7/70/Needham_Joseph_Science_and_Civilisation_in_China_Vol_4-1_Physics_and_Physical_Technology_Physics.pdf

Inside Gaming. (2021). How mirrors work in video games [Video file]. Retrieved from https://www.youtube.com/watch?v=QV-7ck1Fe8E

Insider. (2019). How Pixar's Animation Has Evolved Over 24 Years, From 'Toy Story' To 'Toy Story 4' | Movies Insider [Vídeo]. https://www.youtube.com/watch?v=qTPKGVrFtQU&

Jambusaria, U., Katwala, N., & Deulkar, K. (2014). God Rays in Modern Gaming. International Journal of Computer Applications, 108, 31-33.

Jimenez, J., Gutierrez, D., Yang, J., Reshetov, A., Demoreuille, P., Berghoff, T., ... Sousa, T. (2011). Filtering Approaches for Real-Time Anti-Aliasing. ACM SIGGRAPH Courses.

Jones, R. (2019). Using physically-based lighting values with Enlighten and UE4 | Enlighten - Real Time Global Illumination Solution | Silicon Studio. Acedido a 16 de setembro de 2021, https://www.siliconstudio.co.jp/middleware/enlighten/en/blog/2019/20190322/

jrprockwell & jam-master jim. (2001). supersampling. https://www.everything2.com/index.pl?node_id=1028947

Kalmus, N. (1935) Color Consciouness. Journal of the Society of Motion Picture Engineers. 139-147

Knighty, K. (2018). Motion Blur in Video Games - knighty. Acedido a 3 de outubro de 2021, https://medium.com/@knighty/motion-blur-in-video-games-a5aadcffb2c7

Eastman Kodak Company. (s.d.). George Eastman. Acedido a 21 de setembro de 2021, https://www.kodak.com/en/company/page/george-eastman-history

Kogalla. (2019). How Many Lumens is the Sun? Acedido a 3 de outubro de 2021, https://kogalla.com/blogs/tech-trail/how-many-lumens-is-the-sun

Larson, J. (2020). How Many Frames Per Second Can the Human Eye See? Acedido a 3 de outubro de 2021, https://www.healthline.com/health/human-eye-fps#human-vs-animal-vision

Lev, P. (2006). The Fifties: Transforming the Screen, 1950–1959 (Volume 7) (History of the American Cinema) (1a ed.). Carlifornia, EUA: University of California Press.

Lettier, D. (2019). Screen Space Reflection | 3D Game Shaders For Beginners. Acedido a 30 de setembro de 2021, https://lettier.github.io/3d-game-shaders-for-beginners/screen-space-reflection.html

Liberatore, F., Longazo, J., Pettinati, S., & Weise, D. (2014) Comparison of Anti-Aliasing Techniques for Real-Time Applications. Acedido a 21 de dezembro de 2021, http://jacoblongazo.com/documents/Anti-Aliasing%20Paper.pdf

Lindblad, M. (2020). The History of Photoshop – Photoshop Through the Years. Acedido a 3 de outubro de 2021, https://filtergrade.com/history-of-photoshop-through-the-years/

Lottes. T (2009) FXAA.

Loew, K. (2021). Special Effects and German Silent Film: Techno-Romantic Cinema (Film Culture in Transition). Amsterdão, Holanda: Amsterdam University Press.

Magdics, M., Sauvaget, C., García, R. J., & Sbert, M. (2013). Post-processing NPR effects for video games. Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry - VRCAI '13. Published. https://doi.org/10.1145/2534329.2534348

Maio, A. (2019). What is Depth of Field? Examples of Shallow vs Deep Depth of Field. Acedido a 30 de setembro de 2021, https://www.studiobinder.com/blog/what-is-depth-of-field-definition/

Maio, A. (2021). What is Lens Flare? How to Get It & How to Avoid It. Acedido a 30 de setembro de 2021, https://www.studiobinder.com/blog/what-is-lens-flare/

Manaia, M. (s.d.) Luz, cor e percepção. Lume Arquitetura. 53, 72-78

Mansurov, N. (2013). What is Vignetting? Acedido a 30 de setembro de 2021, https://photographylife.com/what-is-vignetting

Mansurov, N. (2014). Understanding Lens Flare. Acedido a 30 de setembro de 2021, https://photographylife.com/what-is-ghosting-and-flare

Mansurov, N. (2018). How to Deal with post-Processing Pains. Acedido a 30 de setembro de 2021, https://photographylife.com/post-processing-pains

McDonald, A. (2020). What is CGI (Computer-Generated Imagery) & how does it work? Acedido a 3 de outubro de 2021, https://discover.therookies.co/2020/04/05/what-is-cgi-computer-generated-imagery-how-does-it-work/

Mishra, J. (s.d.). How to Understand Depth of Field in Photography. Acedido a 30 de setembro de 2021, https://expertphotography.com/understanding-depth-of-field-photography/

MJP. (2012). A Quick Overview of MSAA. Acedido a 30 de setembro de 2021, https://mynameismjp.wordpress.com/2012/10/24/msaa-overview/

Möller, T., Hoffman, N., & Haines, E. (2008). Real-Time Rendering [Ebook] (3rd ed.).

Monteiro, R. (2015). Veja como os gráficos dos jogos em 3D evoluíram nos principais consoles. Retrieved 3 October 2021, from https://www.techtudo.com.br/noticias/noticia/2015/11/veja-como-os-graficos-dos-jogos-em-3d-evoluiram-nos-principais-consoles.html

Nally, R. (2018). Tips on Making Games: From Modeling to post-Processing. Acedido a 21 de setembro de 2021, https://80.lv/articles/tips-on-making-games-from-modeling-to-post-processing/

NBC Bay Area. (2019). Evolution of Video Games Documentary [Part 1 of 6] [Vídeo]. https://www.youtube.com/watch?v=A-aO3hpCfvQ

NBC Bay Area. (2019). Evolution of Video Games Documentary [Part 2 of 6] [Vídeo]. https://www.youtube.com/watch?v=4yW28c8Le88

NBC Bay Area. (2019). Evolution of Video Games Documentary [Part 3 of 6] [Vídeo]. https://www.youtube.com/watch?v=1Qw9n9zfzsQ

NBC Bay Area. (2019). Evolution of Video Games Documentary [Part 4 of 6] [Vídeo]. Acedido em https://www.youtube.com/watch?v=oaweY1XmSgY

NBC Bay Area. (2019). Evolution of Video Games Documentary [Part 5 of 6] [Vídeo]. Acedido em https://www.youtube.com/watch?v=Kr8VgSgvjAk

NBC Bay Area. (2019). Evolution of Video Games Documentary [Part 6 of 6] [Vídeo]. Acedido em https://www.youtube.com/watch?v=i599VxhLgic

Nvidia. (2007). Part II: Light and Shadows. Acedido a 3 de outubro de 2021, https://developer.nvidia.com/gpugems/gpugems3/part-ii-light-and-shadows

O que significa Anti-Aliasing ou AA? | 2 A.M. Gaming. Acedido a 18 de setembro de 2021, https://blog.2amgaming.com/2020/02/o-que-e-anti-aliasing-aa/

O'Rorke, J. (2021). Real-Time Glow. Acedido a 3 de outubro de 2021, https://www.gamedeveloper.com/programming/real-time-glow

Oh, N. (2018). The NVIDIA Turing GPU Architecture Deep Dive: Prelude to GeForce RTX. Acedido a 18 de setembro de 2021, https://www.anandtech.com/show/13282/nvidia-turing-architecture-deep-dive/2

Ou, Y., Ambalathankandy, P., Takamaeda, S., Motomura, M., Asai, T., & Ikebe, M. (2020). Real-time Tone Mapping A State of the Art Report. IEEE Transactions on Circuits and Systems for Video Technology, 1. https://doi.org/10.1109/TCSVT.2021.3060143

Pagin, S. (s.d.). Adobe Photoshop History - 25 Years in the Making. Acedido a 3 de outubro de 2021, https://www.fastprint.co.uk/blog/the-evolution-of-photoshop-25-years-in-the-making.html

Perez, L. (s.d.). Game production cycle | Game Production. Acedido a 21 de setembro de 2021, http://leonardperez.net/game-production-cycle/

Pickell, D. (2019). Attention Required! | Cloudflare. Acedido a 21 de setembro de 2021, https://www.g2.com/articles/stages-of-game-development

PlayStation | Console and Games. (2008). Acedido a 21 de setembro de 2021, https://www.britannica.com/topic/PlayStation

Polygon University. (2019, July 24). How to Get High Quality Reflections in UE4 : Quality and Performance Tutorial [Video file]. Retrieved from https://www.youtube.com/watch?v=Fqw5c6Ow1xA

Propriedades de Estrelas. (2010). Acedido a 16 de setembro de 2021, https://www.if.ufrgs.br/oei/stars/wd/wd_evol.htm

Rechsteiner, A. The history of video games. Acedido a 21 de setembro de 2021, https://blog.nationalmuseum.ch/en/2020/01/the-history-of-video-games/

Redheffer, R. (1948). A Machine for Playing the Game Nim. The American Mathematical Monthly, 55(6), 343–349. https://doi.org/10.1080/00029890.1948.11999249

Ritson, J. (2018). 1D vs 3D LUTs. Acedido a 30 de setembro de 2021, https://affinityspotlight.com/article/1d-vs-3d-luts/

Sean Whaley, S. (2018, November 20). What is Frame Rate and Why is it Important to PC Gaming? Acedido a 19 de Dezmebro de 2021, https://www.hp.com/us-en/shop/tech-takes/what-is-frame-rate

Sears, J. (2016, July 28). How Photo Retouching Worked Before Photoshop. Acedido a 19 de Dezmebro de 2021, https://www.mentalfloss.com/article/83262/how-photo-retouching-worked-photoshop

Sharp, T. (2017). How Big is the Sun? | Size of the Sun. Acedido a 16 de setembro de 2021, https://www.space.com/17001-how-big-is-the-sun-size-of-the-sun.html

Skylum. (s.d.). Understanding HDR and Tone Mapping. Acedido a 30 de setembro de 2021, https://skylum.com/blog/what-is-tone-mapping

Solar System Sizes | NASA Solar System Exploration. (2003). Acedido a 16 de setembro de 2021, https://solarsystem.nasa.gov/resources/686/solar-system-sizes/

Stefyn, N. (2019). How Video Games Are Made | The Game Development Process | CG Spectrum. Acedido a 21 de setembro de 2021, https://www.cgspectrum.com/blog/game-development-process

Steiner, B. (2011). Post Processing Effects.

Stewart, S. (2021). What is Anti-Aliasing? [Ultimate 2021 Guide]. Acedido a 30 de setembro de 2021, https://www.gamingscan.com/what-is-anti-aliasing/

StudioBinder. (2020). Color Grading vs. Color Correction Process for Video: A Complete Guide. Acedido a 30 de setembro de 2021, https://www.studiobinder.com/blog/color-grading-vs-color-correction-process/

StudioBinder. (2020). What are LUTs? The Ultimate Guide to Color Grading. Acedido a 30 de setembro de 2021, https://www.studiobinder.com/blog/what-is-lut/

Stylized Rendering Post Processing. (s.d.). Acedido a 21 de setembro de 2021, https://docs.unrealengine.com/4.27/en-US/Resources/Showcases/Stylized/PostProcessing/

SvennoJ. (2014). Graphics Effects, Love em or hate em. Acedido a 3 de outubro de 2021, https://gamrconnect.vgchartz.com/thread.php?id=182932

Tabora, V. (2018). Tone Mapping To Achieve High Dynamic Range (HDR) - High-Definition Pro. Acedido a 30 de setembro de 2021, https://medium.com/hd-pro/tone-mapping-to-achieve-high-dynamic-range-hdr-2463bf5cd9fa

"The Cathode Ray Tube Amusement Device," Probably the Oldest Interactive Electronic Game : History of Information. (s.d.). Acedido a 21 de dezembro de 2021, https://www.historyofinformation.com/detail.php?entryid=3573

The Metropolitan Museum of Art. (s.d.). Fading Away. https://www.metmuseum.org/art/collection/search/302289

The New York Times. (1917, February 22). MOVING PICTURES IN COLOR; Dr. Kalmus Exhibits Technicolor Films at Engineering Building. The New York Times. https://www.nytimes.com

Unity Technologies. (s.d.). Unity - Manual: Post-processing. Acedido a 30 de setembro de 2021, https://docs.unity3d.com/Manual/PostProcessingOverview.html

Video Game History Timeline. Acedido a 21 de setembro de 2021, https://www.museumofplay.org/about/icheg/video-game-history/timeline

Video Game History. (2017). Acedido a 21 de setembro de 2021, https://www.history.com/topics/inventions/history-of-video-games

Vogt, N. (s.d.). White Dwarfs. Acedido a 3 de outubro de 2021, http://astronomy.nmsu.edu/geas/lectures/lecture24/slide03.html

Vox. (2016, March 28). We've hit peak lens flare. Here's how it started. [Video file]. https://www.youtube.com/watch?v=_IesAvesFUo

Vox. (2017, December 1). How Technicolor changed movies [Video file]. https://www.youtube.com/watch?v=Mqaobr6w6_I

Wagner, F. & Silva, S. (2021). Introdução ao Ray Tracing.

Wilde, T. (2019). PC graphics options explained. Acedido a 30 de setembro de 2021, https://www.pcgamer.com/pc-graphics-options-explained/2/

Wiltshire, A. (2017). How many frames per second can the human eye really see? Acedido a 3 de outubro de 2021, https://www.pcgamer.com/how-many-frames-per-second-can-the-human-eye-really-see/

Winchester, H. (2019). Real-time, ray-traced and rasterized rendering explained. Acedido a 18 de setembro de 2021, https://www.chaosgroup.com/blog/real-time-ray-traced-and-rasterized-rendering-explained

Wired. (2019, March 22). How Animators Created the Spider-Verse | WIRED [Video file]. https://www.youtube.com/watch?v=l-wUKu_V2Lk

Wirtz, B. (s.d.). Video Games History: From Magnavox Odyssey to Wii, There's No Stopping the Gaming Industry. Acedido a 21 de setembro de 2021, https://www.gamedesigning.org/gaming/history/

Wronski, B. (2015). Anamorphic lens flares and visual effects. Acedido a 30 de setembro de 2021, https://bartwronski.com/2015/03/09/anamorphic-lens-flares-and-visual-effects/

Yanarova, S. (2018). History of Retouching: Photographers and Retouchers Synergy in the Analog Photography Era – Retouching Academy. Acedido a 30 de setembro de 2021, https://retouchingacademy.com/history-of-retouching-photographers-and-retouchers-synergy-in-the-analog-photography-era/

Young, A. (s.d.). History of Photo Editing and Photo Manipulation. Acedido a 30 de setembro de 2021, https://fixthephoto.com/blog/retouch-tips/history-of-photo-retouching.html

Zhang, M. (2016). The History and Explosion of Lens Flare. Acedido a 30 de setembro de 2021, https://petapixel.com/2016/03/28/history-explosion-lens-flare/

Referências Multimédia

Adams, B. (1982). Tron [Arcade]. Chicago, EUA: Bally Midway.

Appeal. (1999). Outcast [Windows]. Paris, França: Infogrames.

Boon, E., & Tobias, J. (1992). Mortal Kombat [Arcade, Amiga, Game Boy, MS-DOS, Sega CD, Game Gear, Genesis, Master System, SNES, Super Famicom, TV game, LCD game, PS1 NTSC-J, PS2, PS3, PSP, Xbox 360 e GameCube]. EUA: Midway.

Bushnell, N., & Dabney, T. (1972). Pong [Arcade]. São Francisco, EUA: Atari.

Cameron, J. (1986). Aliens [Film]. EUA: 20th Century Fox.

Carmack, J., & Abrash, M. (1996). Quake [Sega Saturn]. Texas: Id Software Colley, S., Thompson, G., & Palmer, H. (1973). Maze War [Imlac PDS-1]. California, EUA.

Carmack, J., Romero, J., & Taylor, D. (1993). Doom [MS-DOS]. Texas, EUA: id Software.

Crowther, W., & Woods, D. (1977). Colossal Cave Adventure [DEC PDP-10]. Kentucky, EUA.

Desaulniers, J. (2013). Assassin's Creed IV: Black Flag [Microsoft Windows/ PlayStation 3/ PlayStation 4/ Xbox One/ Xbox 360 / Wii U]. Canadá: Ubisoft Montreal.

Digital, P. (1997). Gran Turismo [PlayStation, PlayStation 2, PlayStation 3, PlayStation Portable, PlayStation 4 e PlayStation 5]. Tóquio, Japão: Sony Interactive Entertainment.

Douglas, R. (2014). Insurgency [Microsoft Window/ OS X/ Linux]. Colorado, EUA: New World Interactive.

Douglas, S. (1952). OXO [Electronic Delay Storage Automatic Calculator]. Cambridge, Inglaterra.

Eidos Interactive. (1996). Tomb Raider [Xbox One]. Londres, Inglaterra.

Epic Games. (2021). Unreal Engine 5 [PlayStation 5]. Carolina do Norte, EUA.

Fowler, A., Vermeij, O., & Roger, A. (2004). Grand Theft Auto: San Andreas [PlayStation 2]. Escócia: Rockstar North.

General Computer Corporation. (1982). Ms. Pac-Man [Arcade]. Massachusetts, EUA.

Gratto, C. (2014). Alien: Isolation [Microsoft Windows/ PlayStation 3/ PlayStation 4/ Xbox 360/ Xbox One]. Inglaterra: Creative Assembly.

Graves, J. (2011). Dead Space 2 [PlayStation 3/ Xbox 360/Microsoft Windows]. Califór-nia, EUA: Visceral Games.

Graves, J., & Hannigan, J. (2013). Dead Space 3 [Xbox 360/ PlayStation 3 /Microsoft Windows]. Califórnia, EUA: Visceral Games.

Gygax, G., & Arneson, D. (1974). Dungeons & Dragons [Boxed set]. Wisconsin, EUA: Tactical Studies Rules.

Higinbotham, W. (1958). Tennis for Two [Computador analógico]. Brookhaven, EUA.

Hoskins, B. (1996). Rainbow [Film]. Reino Unido: First Independent Films.

Iwatani, T. (1980). Pac-Man [Arcade]. Tóquio, Japão: Namco.

Jones, D., & Dailly, M. (1997). Grand Theft Auto [PS1/Windows/MS-DOS/GBC]. Escócia: DMA Design.

Khan, B. (2016). Doom [Microsoft Windows/ PlayStation 4 / Xbox One]. Texas, EUA: id Software.

Kivling, O., & Gill, F. (2012). Syndicate [Microsoft Windows/ PlayStation 3 /Xbox 360]. Suécia: Starbreeze Studios.

Knoll, T. (1987). ImagePro [Macintosh Plus]. Michigan, EUA.

Knoll, T., & Knoll, J. (1990). Photoshop [X86 e x64]. Califórnia, EUA: Adobe Systems.

Kojima, H., Fukushima, T., & Uehara, K. (1998). Metal Gear Solid [Playstation e Microsoft Windows]. Tóquio, Japão: Konami.

Krotz, S. (2013). Tomb Raider [Microsoft Windows/ PlayStation 3 /Xbox 360]. Califórnia, EUA: Crystal Dynamics.

Kunitz, M., & Larsen, S. (2008). Wipeout [Video]. EUA: ABC.

Le, M., & Cliffe, J. (1999). Counter-Strike [Windows]. Washington, EUA: Valve Corporation.

Lucas, G. (1999). Star Wars: Episódio I – A Ameaça Fantasma [Film]. EUA: 20th Century Fox.

Lucas, G. (2002). Star Wars: Episódio II – Ataque dos Clones [Film]. EUA: 20th Century Fox.

Mason, N., Waters, R., Wright, R., & Gilmour, D. (1973). The Dark Side of the Moon [Vinyl/ Cassette/8-Track]. Londres, Inglaterra: Harvest.

Mercier, N. (2019). Need for Speed Heat [Microsoft Windows/ PlayStation 4 / Xbox One]. Suécia: Ghost Games.

Miyamoto, S. (1981). Donkey Kong [Arcade]. Quioto, Japão: Nintendo.

Miyamoto, S., & Tezuka, T. (1986). The Legend of Zelda [Family Computer Disk System]. Quioto, Japão: Nintendo.

Mystique. (2021). Custer's Revenge [Atari 2600].

Narita, K. (1997). Final Fantasy VII [PlayStation]. Tóquio, Japão: SquareSoft.

Nishikado, T. (1978). Space Invaders [Arcade]. Tóquio, Japão: Taito Corporation.

Nygren, V. (2013). Battlefield 4 [Microsoft Windows/PlayStation 3/ Xbox 360/ PlayStation 4/ Xbox One]. Suécia: DICE.

Nygren, V., & Romell, P. (2011). Battlefield 3 [Microsoft Windows/ PlayStation 3 / Xbox 360]. Suécia: DICE.

Ohtsu, T., Kaneko, H., Ishihara, M., Tohma, K., Hasegawa, M., & Nakagawa, S. et al. (1993). Dark Edge [Sega System 32]. Japão: Sega.

Pajitnov, A., Pavlovsky, D., & Gerasimov, V. (1984). Tetris [Game Boy]. Moscovo, Russia.

Petersen, S., Green, S., & McGee, A. (1994). Doom II [IBM Cassette BASIC / CP/M / PC-DOS 1.0]. Texas, EUA: id Software.

Popławski, M. (2020). Cyberpunk 2077 [Google Stadia/Microsoft Windows/ PlayStation 4 / Xbox One]. Polônia: CD Projekt.

Roosendaal, T. (1994). Blender [Linux, macOS, Windows, Android, FreeBSD, OpenBSD, NetBSD, DragonFly BSD e Haiku]. Amsterdão, Holanda: Blender Foundation.

Russell, S. (1962). Spacewar! [Computador PDP-1]. EUA: Massachusetts Institute of Technology.

Russo, A., & Russo, J. (2018). Avengers: Infinity War [Film]. EUA: Marvel Studios.

Scott, R. (1979). Alien, o Oitavo Passageiro [Film]. EUA: 20th Century Fox.

Spector, W. (1994). System Shock [Microsoft Windows /DOS/ MacOS]. Massachusetts, EUA: Looking Glass Technologies.

Stephens, K., & Lambert, K. (2003). Tron 2.0 [Microsoft Windows]. Washington, EUA: Monolith Productions.

Sweeney, T. (2014). Unreal Engine 4 (Version Unreal Engine 4, v4.18) [Microsoft Windows/ Linux/ Mac OS X/ Xbox One/ WiiU/ PlayStation 4/ HTML5/ iOS/ Android/Nintendo Switch]. Carolina do Norte, EUA: Epic Games.

Timson, S. (2008). Dead Space [Xbox 360/ Microsoft Windows/ PlayStation 3]. Califórnia, EUA: EA Redwood Shores.

Unity Technologies. (2005). Unity (Version 2021.1.6) [Windows/ Xbox 360/ MacOS/ Linux/Android/ iOS]. Califórnia, EUA.

van der Leeuw, M. (2017). Horizon Zero Dawn [PlayStation 4]. Países Baixos: Guerrilla Games.

Vermeij, O., & Fowler, A. (2002). Grand Theft Auto: Vice City [PlayStation 2]. Escócia: Rockstar North.

Wiseman, L., Grevioux, K., & McBride, D. (2003). Underworld [Film]. Califórnia, EUA: Screen Gems.

Wright, W. (2000). The Sims [Microsoft Windows]. Califórnia, EUA: Maxis.

WS Anderson, P. (1997). O Enigma do Horizonte [Film]. EUA: CIC Video.

Yajima, H., Iwamoto, D., & Iwawaki, T. (1996). Super Mario 64 [Nintendo 64]. Quioto, Japão: Nintendo.

Yamauchi, K. (1997). Gran Turismo [PlayStation]. Tóquio, Japão: Polyphony Digital.

Zur, I., & Diemer, B. (2007). Crysis [CryEngine 2: Microsoft Windows]. Frankfurt, Alemanhã: Crytek.

Gonçalves, S., Angelino, C., & Candeias, H. (2019). White Dwarf [Microsoft Windows]. Game Prototype. Acedido em https://v3.globalgamejam.org/2019/games/white-dwarf

Arnold, B., Guggenheim, R., & Lasseter, J. (1995). Toy Story [Filme]. Estados Unidos da America: Pixar Animation Studios.