Universidade Federal de Minas Gerais
EBA/Cinema de Animação e Artes Digitais

Comparação crítica entre renderização e renderização em tempo real

Aluno: Ricardo Viana Villela
Orientador: Arttur Ricardo Araujo Espindula

RESUMO

Foi feita uma comparação crítica entre uma cena tridimensional renderizada nos computadores do laboratório 3D-A da Escola de Belas Artes da UFMG utilizando o **Blender Cycles** e a **Unreal Engine 4**. Esta cena foi criada deliberadamente com o intuito de explorar as principais diferenças entre os resultados obtidos, contendo mostras de quantidade de **geometria**, nove exemplos de materiais básicos (**difuso**, **metálico**, **anisotrópico**, liso, vidro, **SSS**, **materiais texturizados**, **texturas procedurais** e um teste de brilho), dois tipos de iluminação (***point light*** e HDRi), além de uma pequena animação de 325 quadros feita para rodar a 25 quadros por segundo, criada para testar as diferenças situacionais de iluminação (moção, matiz e intensidade), exportações e tempo de execução. Encontramos no resultado da Unreal Engine 4 uma diferença substancial do tempo de finalização em detrimento da qualidade visual conseguida na imagem renderizada com o Blender Cycles.

Palavras-chaves: Blender Cycles. Unreal Engine 4. Renderização. *Realtime rendering*. UE4.

INTRODUÇÃO

São disponibilizados para os estudantes do curso de Cinema de Animação e Artes Digitais, na Escola de Belas Artes da UFMG, equipamentos que estão ultrapassados em relação às tecnologias atuais, pois se compararmos as configurações das peças de **GPU** e **CPU** da escola (GPU NVIDIA Quadro K600, CPU Intel Xeon E5-1620 v2) com as disponíveis no mercado atualmente (CPU Intel

Xeon E3-1285 v6, sendo que a GPU tem uma performance de até 4025% menor em relação à GPU mais recente) veremos a diferença, principalmente por se tratarem de equipamentos lançados no ano de 2013 (os comparativos estão disponíveis nas referências[1] e [2]). Essa falta de computadores atualizados implica diretamente nos membros deste curso que querem produzir um conteúdo com **modelagem tridimensional**, principalmente no que se trata de **renderização**.

São ensinados nas disciplinas relacionadas ao conteúdo de renderização 3D apenas os **renderizadores** nativos do Blender. Além disso, para poder utilizar um **renderizador externo** ou até mesmo um outro programa, é necessário fazer uma solicitação oficial aos responsáveis por essas disciplinas. Sendo assim, estão disponíveis imediatamente essas duas opções: o renderizador interno do Blender (descontinuado) e o Cycles. São renderizadores com um grande potencial, mas que dependem de muito processamento da máquina para conseguir um resultado rápido, se comparados com renderizadores em tempo real. Com essa dificuldade surge a busca de novos meios, como o utilizado por um grupo de alunos na matéria de ATELIÊ: CINEMA DE ANIMAÇÃO III de 2018, no filme Gota de Chuva, onde o grupo fez uso do programa Unreal Engine 4 para realizar as renderizações.

Este artigo busca fazer uma comparação crítica entre resultados obtidos em testes realizados utilizando os dois programas, Blender (Cycles) e Unreal Engine 4 (ou UE4 como também é chamada), a partir de uma cena criada deliberadamente. Esta cena expõe os principais materiais necessários para a criação de uma cena comum em animação, introduzindo uma alternativa de renderização para o público que possui um equipamento de trabalho ultrapassado, como é o caso do curso já citado, e que deseja obter um resultado de renderização com menos tempo de espera em relação aos renderizadores citados (Blender interno e Cycles).

MATERIAIS E MÉTODOS

Cena tridimensional modelada e texturizada no Blender versão 2.79b. A cena consiste em uma sala, com cinco planos (como um quarto), três modelos idênticos de **Suzanne** (objeto típico do Blender) para representar modelos com formas dinâmicas, quatro cavidades no plano do meio (parede frontal) onde se encontram uma esfera em cada cavidade, para apresentar diferentes materiais em cada esfera,

conforme consta na FIG. 1. A cena somou um total de 35.310 **vértices** e 70.078 **triângulos**.

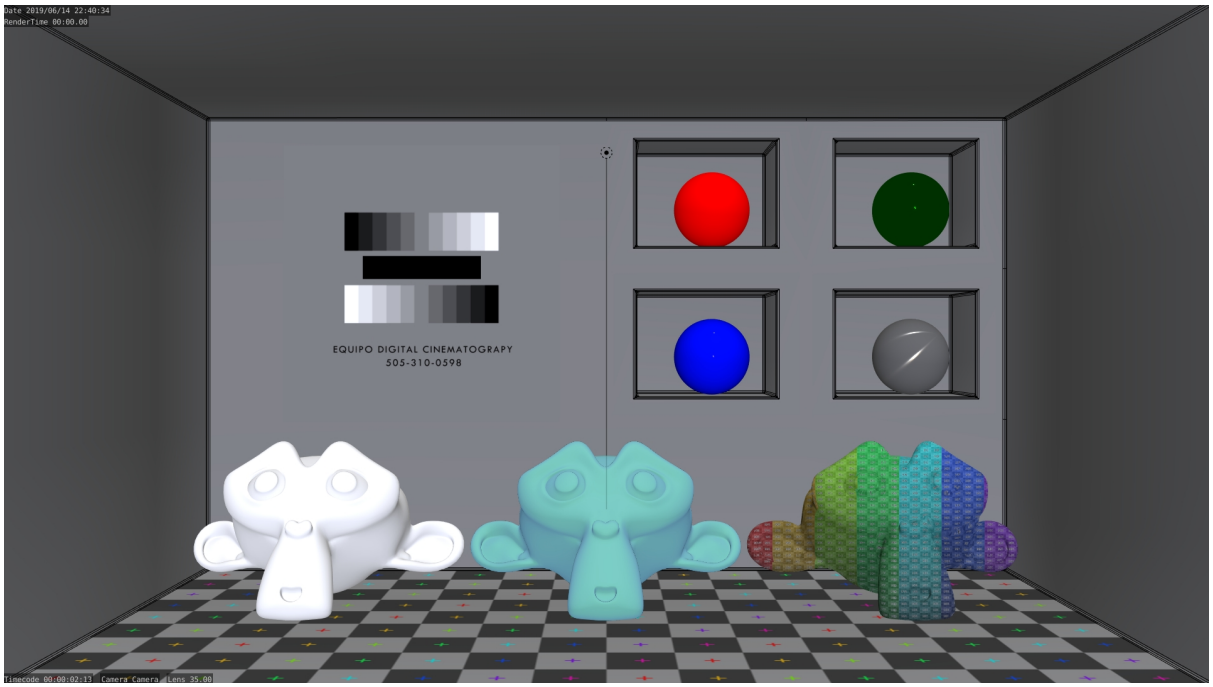

FIG. 1 - Captura de tela da cena no arquivo do Blender.
Fonte: elaborado pelo autor.

Relação de geometria dos objetos é apresentada no Quadro 1.

QUADRO 1
Relação de geometria dos objetos tridimensionais

Objeto	**Nº de vértices**	**Nº de faces triangulares**
Esfera	2.562	5.120
Plano	4	2
Sala	1.184	2.364
Suzanne	7.958	15.744

Fonte: elaborado pelo autor.

Os materiais dispostos nos modelos foram distribuídos de maneira deliberada: há um material difuso (de nome *Diffuse*), um material metálico (de nome *Metallic*), um material anisotrópico (de nome *Anisotropic*), um material liso (de nome ***Glossy***), um material de vidro (de nome *Glass*), um material **SSS** (***Subsurface Scattering*** de nome *SSS*), dois materiais texturizados (dois com texturas de cor

com resolução 4096x4096 pixels e outro no chão com textura de cor, de resolução 64x64 *pixels,* nomes *TextureColorTest* e *GridTest*), um material com **textura procedural** (no Blender “voronoi”, na Unreal “fractal” que apresentam resultados semelhantes, de nome *ProceduralTextureTest*), uma imagem de teste de brilho (textura com resolução de 2.048x2.048 *pixels*, nome *Bright_Test*). Foram escolhidos esses materiais (FIG. 2), pois eles são os materiais mais básicos e os mais utilizados em animações, sobretudo nos trabalhos realizados no curso. Além disso é possível fazer misturas desses materiais para simular vários outros derivados destes.

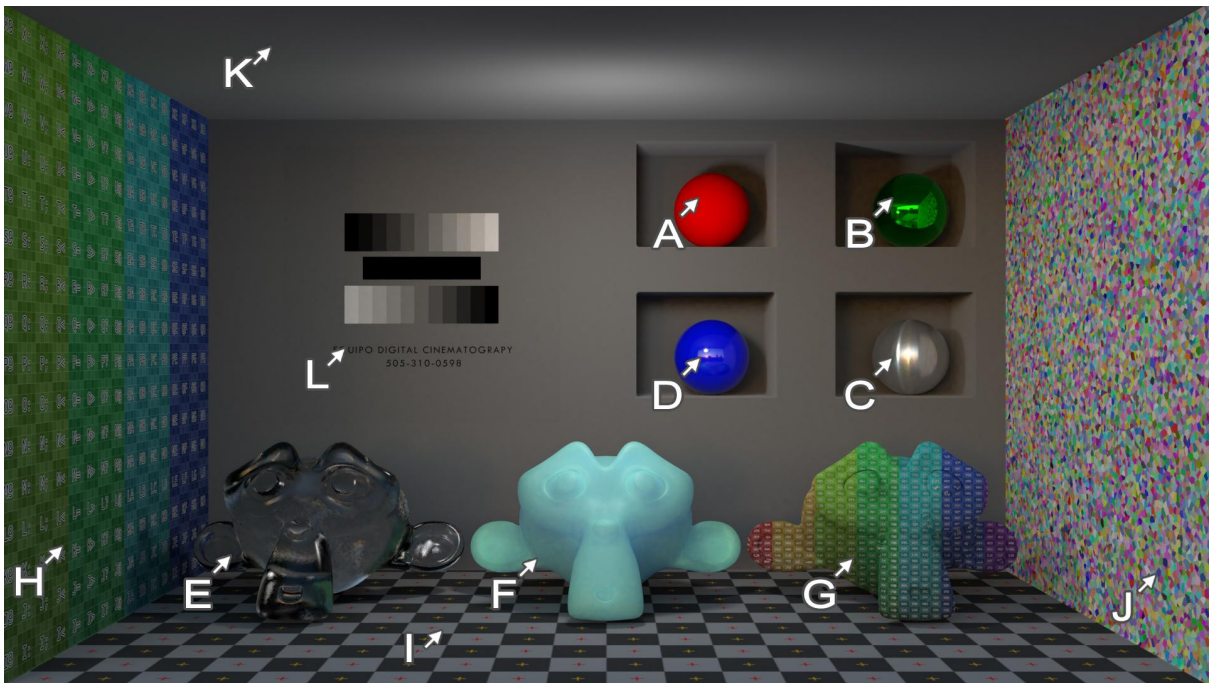

FIG. 2 - Cena renderizada utilizando o Blender Cycles, com letras utilizadas para indicar cada objeto. Fonte: elaborado pelo autor.

Relação dos objetos e seus materiais é apresentada no Quadro 2.

QUADRO 2
Objetos tridimensionais e seus materiais

Objeto	**Espaços de material**	**Nome do material**
Sala	4	K - *neutral*, I - *Grid_Test*, H - *TextureColorTest*,

		J - *ProceduralTextureTest*
Plane	1	L - *Bright_Test*
Icosphere	1	A - *Diffuse*
Icosphere.001	1	B - *Metallic*
Icosphere.002	1	D - *Glossy*
Icosphere.003	1	C - *Anisotropic*
Suzanne	1	F - *SSS*
Suzanne.001	1	E - *Glass*
Suzanne.002	1	G - *TextureColorTest*

Fonte: elaborado pelo autor.

A cena foi iluminada por uma **imagem HDRi** (com resolução de 3.200x1.600 *pixels*) e uma luz pontual (*point light*). A animação serviu para realizar a interação da cena com as luzes, sendo que as lâmpadas foram animadas. Foram testadas várias posições da lâmpada *point light*, com uma animação que a faz percorrer a sala, mostrando as variações dos materiais atingidos pela luz, além de haver animações de intensidade, matiz e saturação. Quanto à animação HDRi, ocorreu apenas a alteração de intensidade de luminosidade.

A cena foi renderizada na resolução 1920x1080 em cada quadro, no formato PNG (RGB, 8 bits, 0% de compressão). Após a exportação de cada *frame*, foram compiladas todas as imagens em um arquivo de vídeo, no formato MPEG-4 (codec H.264). Não houve canal de áudio no arquivo final, uma vez que o teste é exclusivamente visual, sem preocupações com som.

A animação durou 13 segundos, sendo 325 quadros dispostos a 25 fps (fotograma por segundo). Começou com uma cena escura, seguida da imagem HDRi, subindo sua intensidade de luminosidade em 1 segundo. Logo após foi a *point light* que acendeu, também em 1 segundo, em seguida, em meio segundo, a lâmpada foi movida até o topo da cena, onde ela iniciou uma volta pela sala que dura 6 segundos, parando exatamente no ponto onde começou, após uma descida de também meio segundo. Após essa animação, começou uma animação de saturação e matiz que durou 2 segundos. Assim que as interações de saturação e

matiz acabaram, a lâmpada se apagou novamente em 1 segundo, seguido do HDRi, que também durou mais 1 segundo para zerar a sua intensidade de luminosidade.

Essa animação com os mesmos objetos e mesmos materiais foi realizada em duas plataformas: *Blender* e *Unreal Engine 4*. Cada plataforma tem seu meio de criar essa animação, trazendo seus resultados específicos.

A cena foi feita num arquivo do Blender 2.79b, ocupando um espaço em disco de 14.809 KB, contendo todos os arquivos de imagem: LA_Downtown_ Helipad_GoldenHour_3k.hdr e bright_test.png. Como a cena foi criada no próprio Blender, não houve necessidade de importação de nenhum objeto 3D para a cena.

Além dos objetos citados anteriormente, há na cena uma lâmpada *point light* e uma câmera. A câmera foi configurada com perspectiva ativa, com o valor de distância focal padrão de 35 mm, que seguiu inalterada durante toda a animação. A lâmpada (convenientemente nomeada como *point*) teve seus valores alterados, pois essa, sendo essa animada.

Também foi animado o HDRi, textura utilizada no **nó** (*node*) *Background* para simular uma iluminação atmosférica. Desse modo, no Blender foi utilizado o sistema mais comum de aplicação de HDRi, um nó *Environment Texture* conectado ao valor de cor do nó *Background*. No *Backgroud* o valor força (*Strength*) também foi parametrizado, indo de 0.000 a 1.000 (FIG. 3).

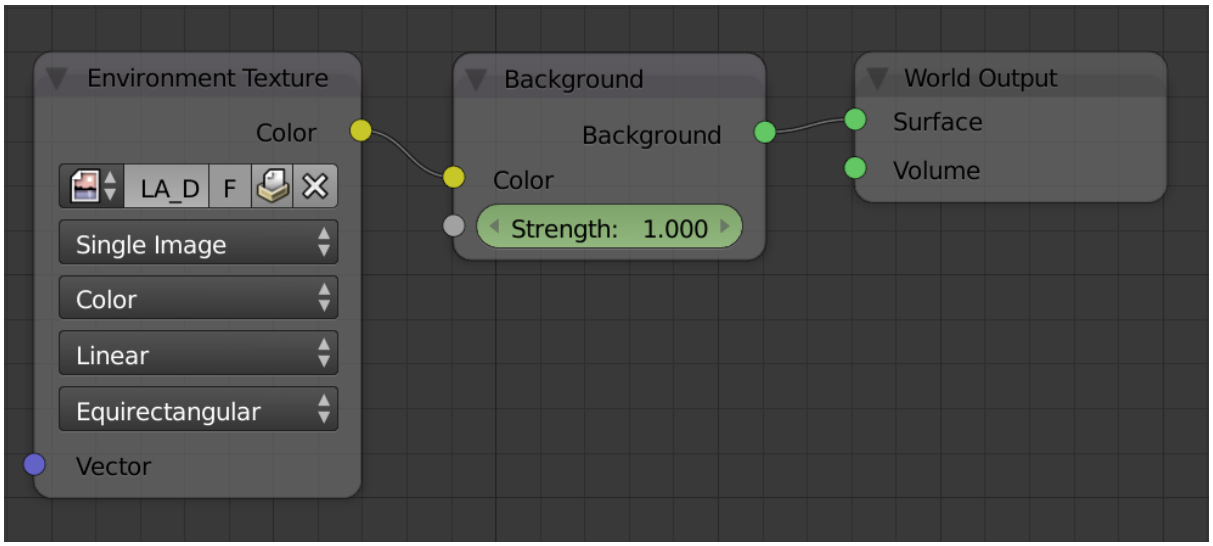

FIG. 3 - Captura de tela da configuração de nós no material do mundo da cena no Blender. Fonte: elaborado pelo autor.

Quanto aos materiais, foram utilizados na maioria deles o *nó* ***Principled BSDF***, que tem uma interface parecida com o *Output* das configurações de materiais da Unreal Engine 4. Nos materiais *TextureColorTest*, *SSS*, *Glossy*, *Metallic* e *Anisotropic,* foram usados o nó *Principled BSDF* (FIG. 4).

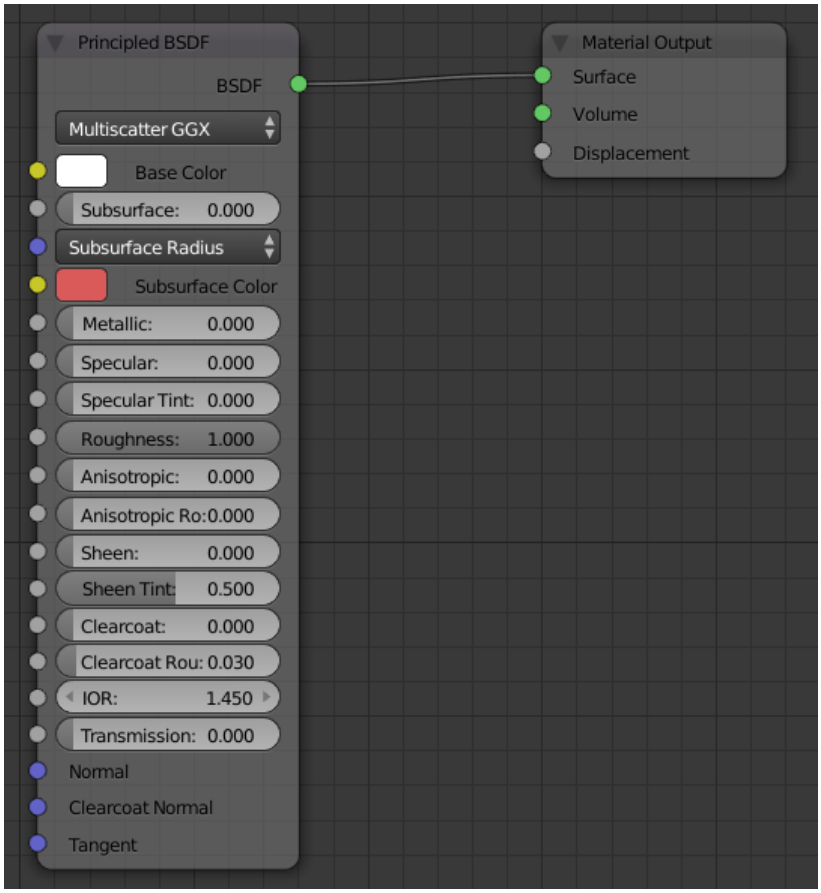

FIG. 4 - Exemplo de material com o nó *Principled BSDF*.

Nos outros materiais, que não tiveram necessidades de misturas, foram usados apenas um nó básico. Nesse caso, ficou o seguinte contexto:

- neutral: nó *Diffuse BSDF*;
- Grid_Test: nó *Image Texture, Diffuse BSDF*;
- Bright_Test: nó Image Texture e *Diffuse BSDF*;
- Diffuse: nó *Diffuse BSDF*;
- Glass: nó *Glass BSDF*;
- SSS: nó *Principled BSDF*;
- TextureColorTest: nó *Image Texture, Principled BSDF*;
- Glossy: nó *Principled BSDF*;

- Metallic: nó *Principled BSDF*;
- Anisotropic: nó *Principled BSDF.*

Toda a cena foi configurada para ter o melhor rendimento de tempo com qualidade. Qualidade nesse sentido seria a redução de **ruídos** e **vaga-lumes** na cena, além de manter as características visuais esperadas dos materiais citados. Nesse caso, o metal deve se parecer metálico, o vidro deve parecer vidro, e assim por diante.

A proposta foi da cena ser renderizada utilizando o renderizador Cycles, integrado no Blender 2.79b. Nesse experimento foi utilizada a função do renderizador Cycles: *Denoising,* onde o programa filtra o resultado da imagem para se livrar dos ruídos, enquanto preserva o máximo de detalhes visuais que puder. Apenas o valor do raio (*Radius)* no *Denoising* foi diminuído para 5 (que por padrão é 8). Desse modo, a função filtra partes menores da imagem, preservando mais detalhes em função de tempo de execução.

As configurações de *render* alteradas foram as de amostragem (*Sampling*), caminhos de luz (*Light Paths*) e *Performance*, todas na aba *Render*, em *Properties*. Como podemos observar na FIG. 5, em amostragem (*Sampling*) foi alterado o valor de "*Samples*>*Render*" para 128, que determina a quantidade de amostras que o renderizador irá gerar até finalizar um quadro da animação. 128 foi a quantidade mínima necessária para gerar um resultado com poucos ruídos com a ajuda do *denoising*.

Em *Light Paths* foram alterados diversos valores: "transparência (*Transparency*) Min e Max" alterados para 1; "*Bounces*> Min" alterado para 1; "Bounces>Max" alterado para 3; "Difusa (*Diffuse*)" alterado para 2; "Polimento (*Glossy*)" alterado para 2; "Transmissão (*Transmission*)" alterado para 3; "Volume" alterado para 1. Foram desativadas as opções de cáustica reflexiva (*Reflective Caustics*) e cáustica refrativa (*Refractive Caustics*). A opção de sombra (*Shadow*) se manteve ativada. Esses valores se referem à quantidade de caminhos de luz (*light paths*), que por padrão vem com valores mais altos, aumentando muito o tempo de processamento da renderização. Para o nosso estudo não era necessário usar as quantidades padrões, pois os números estavam configurados em excesso,

causando apenas maior tempo de espera por *render* sem ter alteração na imagem, sendo então diminuídos os valores de maneira a otimizar o tempo de renderização.

Em *Performance*, foi ativada a opção *Save buffers*. "*Tiles> Hilbert Spiral"* foi alterado para *Center* e o valor de X e Y configurados para X=128 e Y=64. Valores baseados em testes realizados por Andrew Price, onde ele obtém resultados de renderização mais rápidos sem afetar na qualidade da imagem final, registrados em seu *blog* Blender Guru [7].

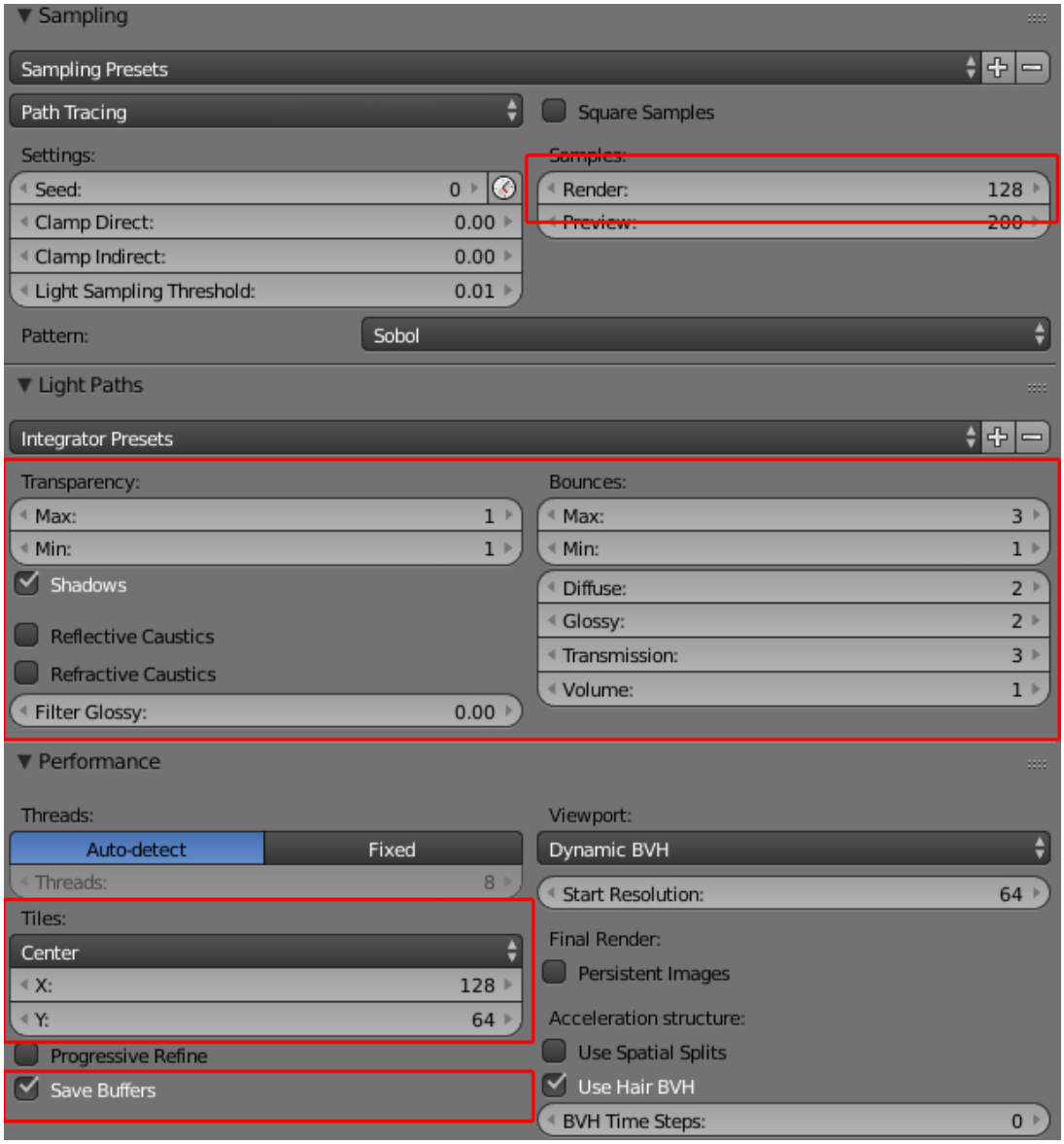

FIG. 5 - Captura de tela com as configurações finais de renderização para renderizar a animação no Blender Cycles (*Sampling*, *Light Paths* e *Performance*); Fonte: elaborado pelo autor.

Para a realização do teste foi replicada na ***game engine*** Unreal Engine 4 a mesma cena gerada no Blender (FIG. 6). O arquivo de projeto da Unreal Engine 4 não é compactado como o do Blender, ou seja, a *game engine* cria uma pasta de dados do projeto. A pasta do projeto do nosso experimento ocupou um espaço em disco de 57,5 megabytes, espaço contando com os mesmos arquivos de texturas existentes no arquivo do Blender. Inclui-se nesse projeto também os arquivos em FBX, pois são esses os arquivos dos modelos tridimensionais feitos e exportados da cena do Blender, além de 24,2 MB referentes a uma cópia de segurança automática gerada pela Unreal.

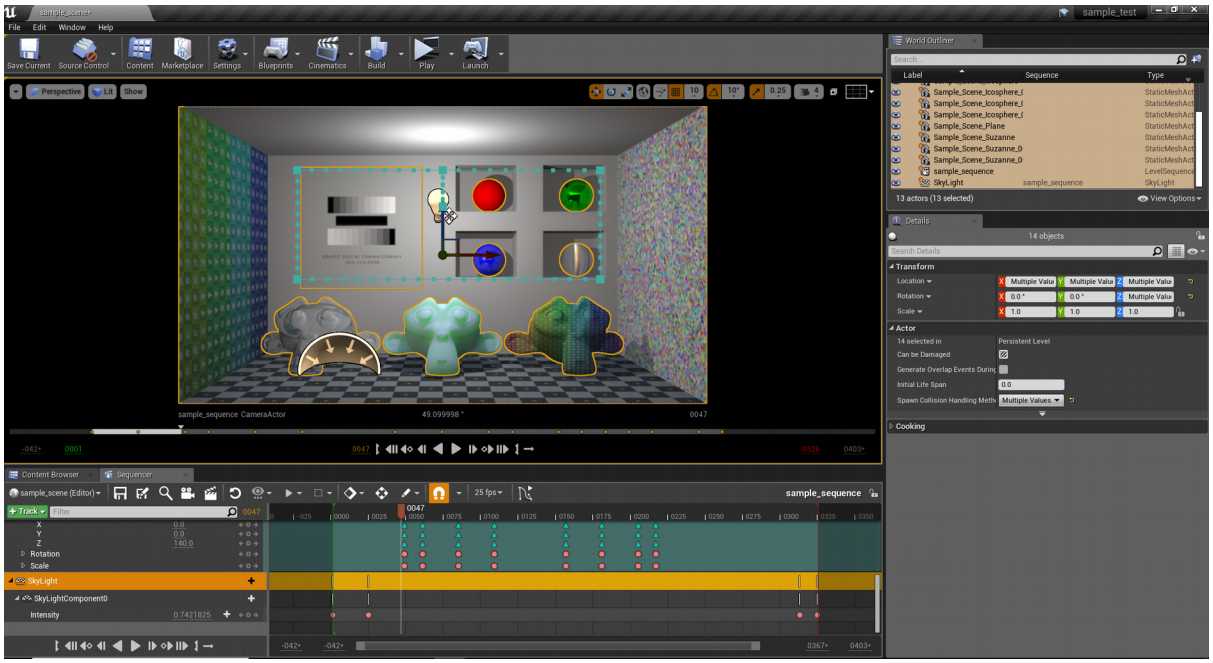

FIG. 6 - Captura de tela da cena final na Unreal Engine 4.
Fonte: elaborado pelo autor.

Como a Unreal é uma *game engine* e não um programa de modelagem 3D, foram importados os modelos gerados na cena do Blender já citados, com as mesmas nomenclaturas e disposição de materiais. Não foi adicionado nenhum outro modelo à cena, uma vez que o propósito do estudo é comparar a mesma cena renderizada nas duas plataformas. Sendo a cena primária gerada no Blender, essa cena serviu de referência para o resultado obtido na Unreal, respeitando as particularidades deste segundo programa. Não era o intuito gerar uma réplica perfeita do *render* no Cycles.

Para além dos modelos, houve a necessidade de adicionar objetos como a lâmpada e a câmera, pois a *engine* não suporta importação desses objetos quando exportados do Blender. A lâmpada utilizada também foi a *point light*, que mesmo com configurações diferentes da *point light* do Blender, possui a mesma função. Para simular iluminação de ambiente (Environment Light) é necessário, na Unreal, utilizar um *SkyLight*, opção de lâmpada que simula iluminação atmosférica. A partir disso, as lâmpadas foram animadas de maneira a reproduzir a cena criada no Blender. As animações ficaram levemente diferentes por conta das interpolações automáticas de quadros de cada programa (FIG. 7): no Blender a interpolação automática acaba a cada *frame*, enquanto na Unreal, as interpolações se sobrepõem a cada *frame*, causando um efeito indesejado na animação. Sendo assim, foi configurada na Unreal a interpolação linear nas animações. Para conseguir na UE4 um resultado de interpolação idêntico ao do Blender seria necessário um tempo adicional, tempo este que, nesse ponto da pesquisa, já não tínhamos mais disponível.

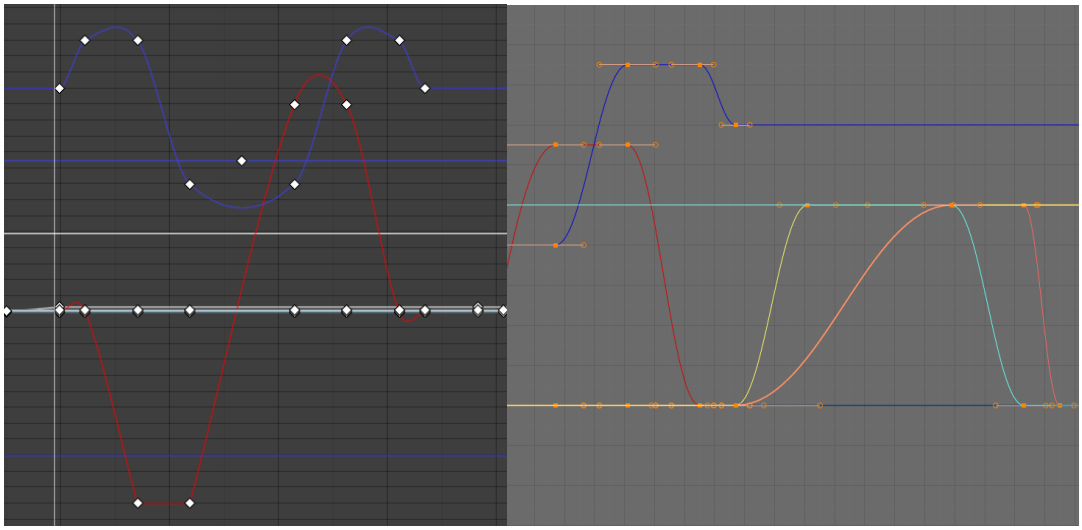

FIG. 7 - Recorte de captura de tela dos exemplos de interpolação: na esquerda a interpolação automática da Unreal Engine 4, à direita a interpolação automática do Blender.
Fonte: elaborado pelo autor.

Como se trata de uma *game engine*, toda a cena é pré-configurada de maneira a otimizar o projeto. Desse modo, é necessário mudar algumas configurações para o programa priorizar a qualidade gráfica (sem ruídos visuais e

vaga-lumes), consequentemente perdendo mais na *performance* (ampliando o tempo do *render*). Como nossa intenção não é fazer um jogo, e sim um arquivo de vídeo, a perda de *performance*, nesse sentido, não é um problema, contanto que não atrapalhe o processo de criação dessa cena. Por padrão a Unreal tem opções como oclusão de ambiente, ***raytracing***, reflexos, qualidade de textura, sombras, *anti-aliasing* e algumas configurações de pós-processamento configuradas com a qualidade média/baixa, sendo necessário alterar algumas dessas configurações e aumentar outras. Todas essas configurações citadas foram configuradas com as opções de maior qualidade que o programa oferece (FIG. 8), com exceção do *raytracing*, pois as configurações dos computadores disponíveis para a pesquisa não suportavam, além do raytracing ser uma opção da *engine* muito recente, sendo assim ainda pouco otimizada, elevando a demanda de processamento da máquina para poder funcionar.

FIG. 8 - Corte de captura de tela destacando a opção de configurações de gráficos gerais da Unreal Engine 4.
Fonte: elaborado pelo autor.

RESULTADOS

As renderizações realizadas no Blender Cycles demoraram um tempo médio de 5 minutos e 30 segundos de finalização por quadro, levando um total aproximado de 1788 minutos, ou seja, 30 horas de renderização. Foram utilizados 4 computadores para realizar as renderizações, totalizando 7 horas e 30 minutos por computador. O tempo exato de cada quadro está registrado no canto superior esquerdo de cada um deles, podendo ser visto nos arquivos da pesquisa (FIG. 9).

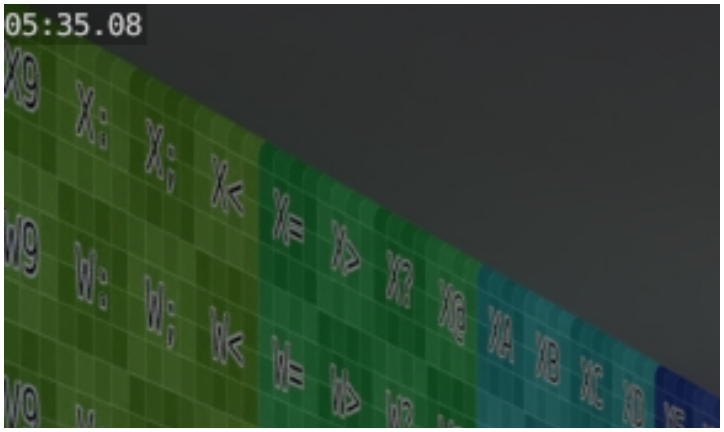

FIG. 9 - Recorte de quadro com tempo de renderização de quadro específico.
Fonte: elaborado pelo autor.

No Blender o resultado visual foi satisfatório. Os materiais configurados ficaram fiéis, condizentes com suas descrições (metal parecendo metal, vidro parecendo vidro e assim por diante). Toda a interação da animação de iluminação funcionou sem apresentar nenhum erro ou resultado inesperado, como quadros apresentando mais ruídos ou **vaga-lumes** (*fireflyes*) [8].

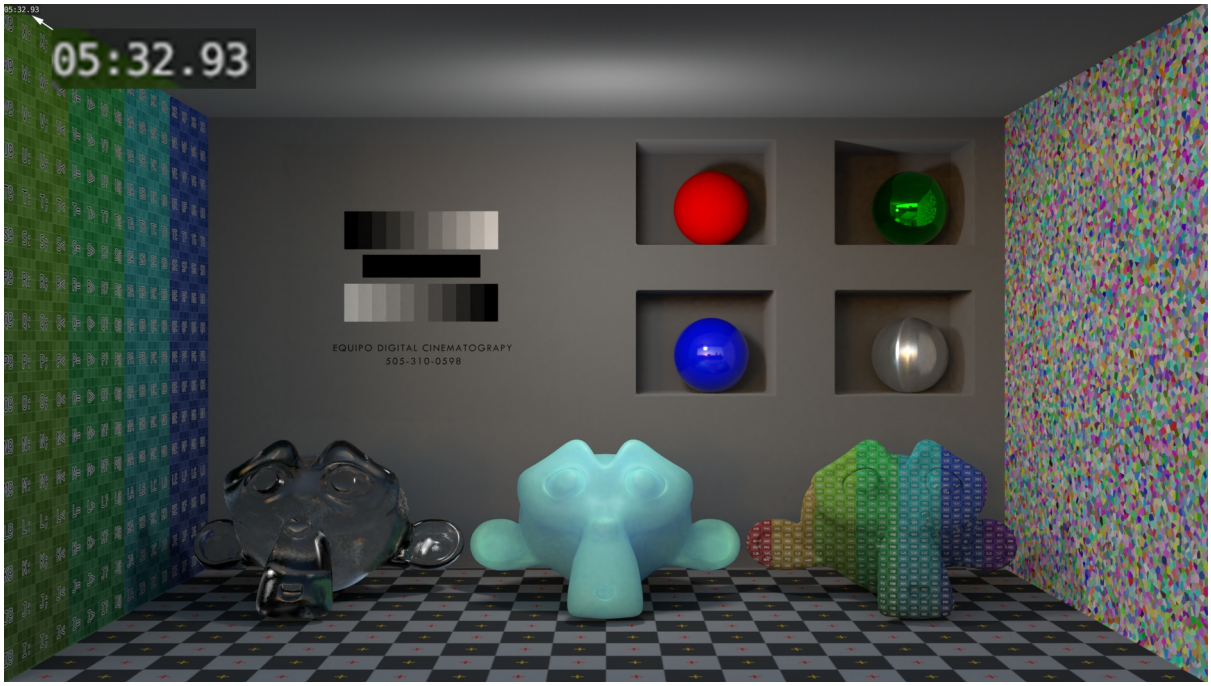

FIG. 10 - Resultado da renderização final da cena no Blender, quadro 41 da animação (seta indica ampliação da imagem).
Fonte: elaborado pelo autor.

O único problema visual apresentado nos quadros foi um aspecto meio embaçado nos objetos, efeito causado por consequência do uso da função

Denoising. Mas para detectar esse problema durante a animação, é necessário alguém mais atento (FIG. 11).

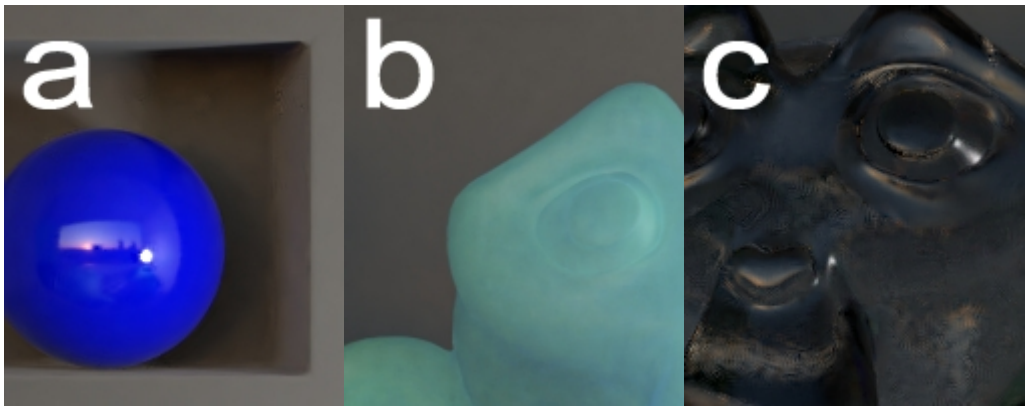

FIG. 11 - Recorte de efeitos indesejados encontrados no quadro 154 da animação renderizada no Blender Cycles. Nenhuma imagem deveria ter apresentado essa textura indefinida, mas sim um aspecto de efeito gradiente comum:
a) Efeito indesejado encontrado em toda a sombra da esfera
b) Efeito indesejado encontrado em todo o material SSS
c) Efeito indesejado encontrado em todo o material Glass
Fonte: elaborado pelo autor.

Na UE4 o tempo de renderização foi de 13 segundos para os 325 *frames*, ou seja, 25 quadros renderizados por segundo, ou seja, 40 milissegundos por quadro. O tempo de renderização da Unreal Engine 4 foi de aproximadamente 8.200 vezes mais rápido do que o tempo de renderização utilizando o Blender Cycles. Como a UE4 se trata de um motor de jogos (*game engine*), é natural que o resultado de renderização seja rápido. Por esse motivo o tipo de renderização é denominado renderização em tempo real. Contudo, foram apresentadas algumas insatisfações no resultado visual obtido na UE4.

Como foi citado anteriormente, não foi possível utilizar a recém implementada tecnologia de *raytracing* na Unreal Engine 4. Com o *raytracing* desabilitado, a cena perdeu bastante qualidade gráfica se comparada com a cena renderizada no Cycles. Testes de render com *raytracing* na UE4 são apresentadas na FIG. 12 e 13.

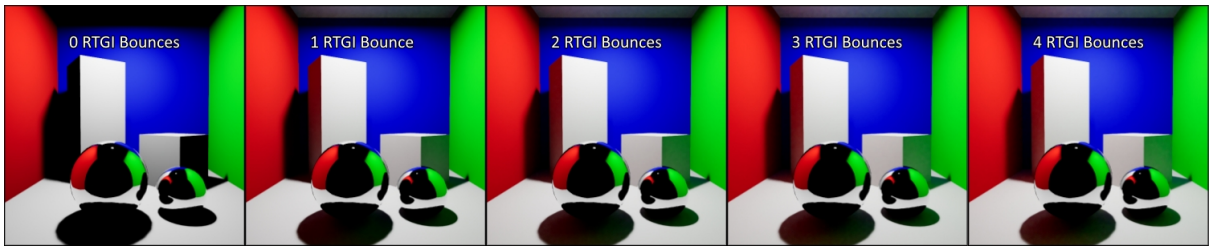

FIG. 12 - Teste de **RTGI** (*Ray Tracing Global Illumination*) utilizando a UE4.
Fonte: LENZ, 2019, #ue4realtimeraytracingguide. [9]

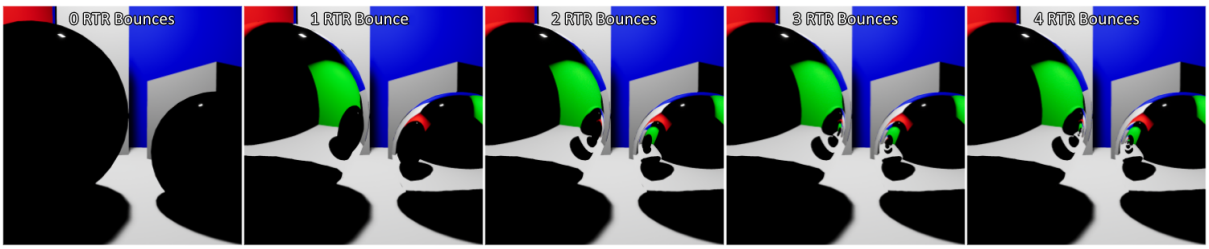

FIG. 13 - Teste de **RTR** (*Ray Tracing Reflections*) utilizando a UE4.
Fonte: LENZ, 2019.

Para conseguir um resultado gráfico idêntico ao gerado no Blender, seriam necessárias várias alterações em configurações básicas, além de geração de várias alternativas que estão além das condições nativas dos recursos da Unreal Engine 4. Sendo assim, alguns desses aspectos foram mantidos, inclusive serviram de objeto de análise em nossa comparação, identificados como características intrínsecas deste renderizador, como por exemplo os efeitos realizados por pós-processamento e as paletas de cores. O resultado final de renderização da cena utilizando a Unreal foi apresentado na FIG. 14.

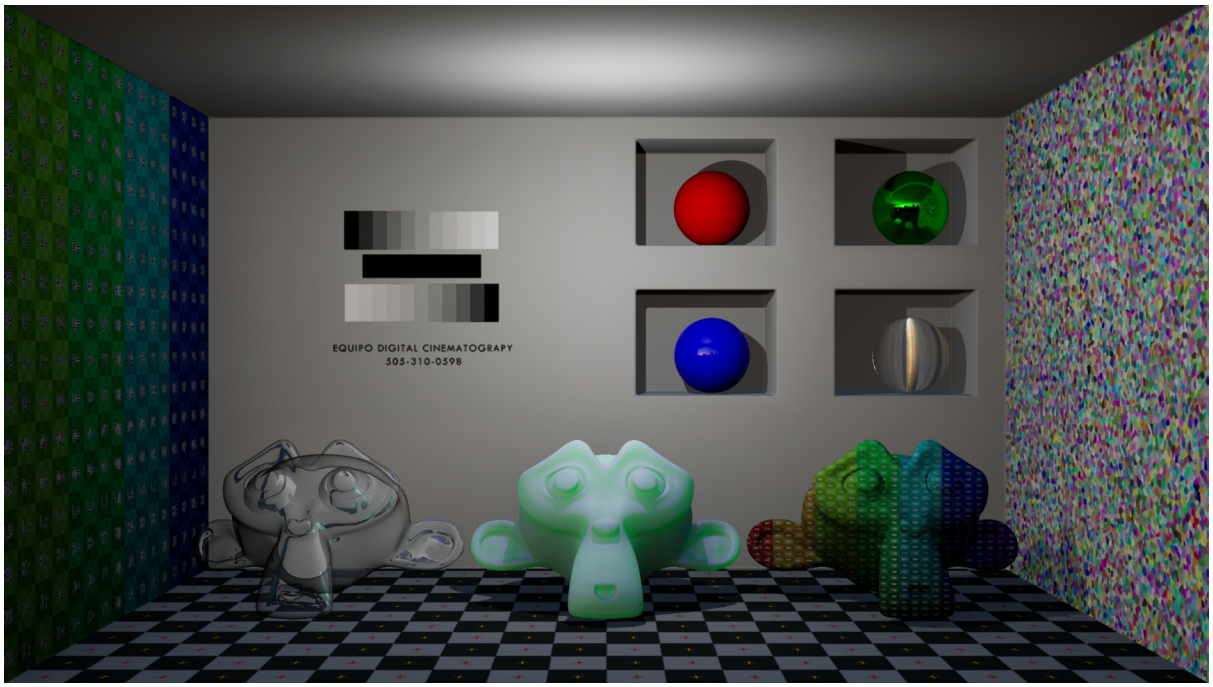

FIG. 14 - Resultado da renderização final da cena na UE4, quadro 41 da animação.
Fonte: elaborado pelo autor.

Imediatamente percebemos a diferença entre o resultado visual obtido com o *render* do Blender Cycles e o resultado da renderização em tempo real da Unreal Engine 4. O resultado da renderização em tempo real ficou mais escuro e não apresentou o problema com ruídos que surgiu no resultado do Blender Cycles, devido ao uso do *denoising*. A diferença do resultado entre objetos que tem os materiais de vidro, SSS, metal e anisotrópico também ficou nítida. A iluminação apresentou resultado visual insatisfatório, mas esperado, devido ao fato da não utilização da função *raytracing*. Comparação mais detalhada dos materiais obtidos nos dois sistemas é apresentada na FIG. 15.

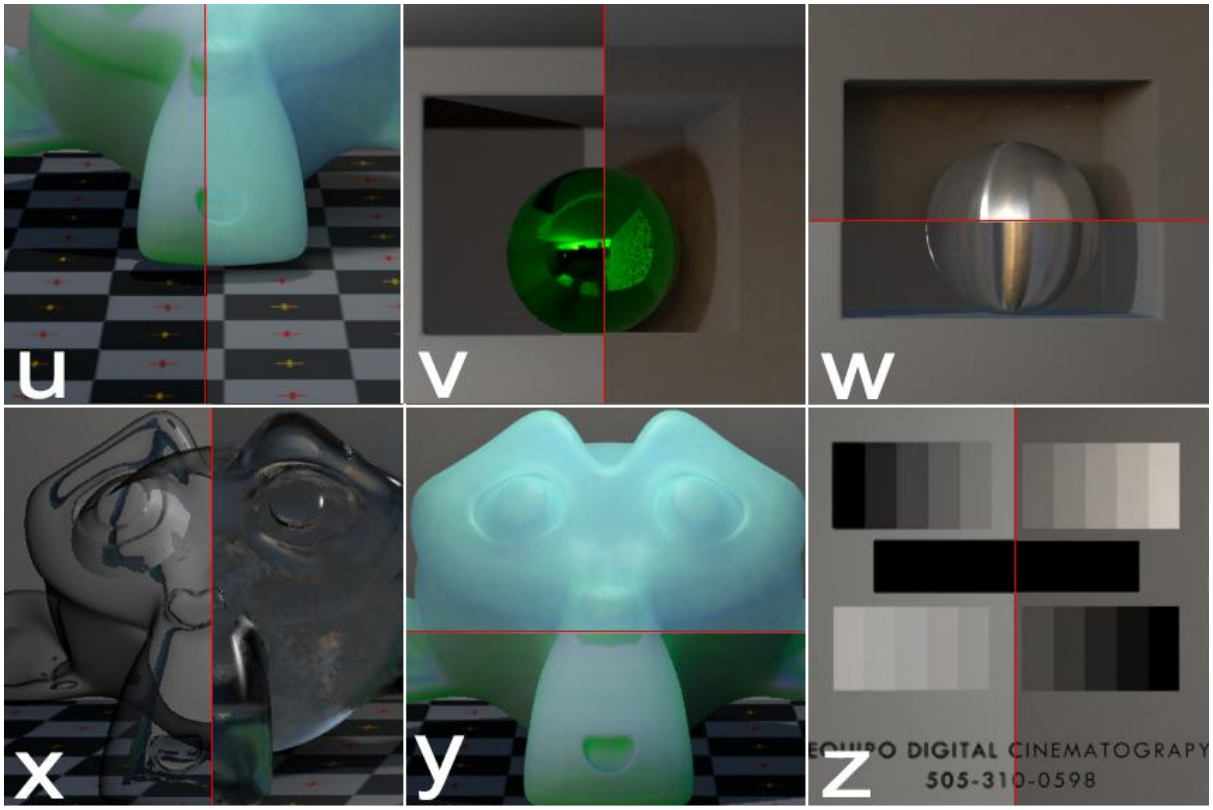

FIG. 15 - Montagem de comparação entre os resultados obtidos nas renderizações com o Blender Cycles e Unreal Engine 4:
u) Esquerda: resultado da UE4; Direita: resultado do Cycles
v) Esquerda: resultado da UE4; Direita: resultado do Cycles
w) Abaixo: resultado da UE4; Acima resultado: do Cycles
x) Esquerda: resultado da UE4; Direita: resultado do Cycles
y) Abaixo: resultado da UE4; Acima resultado: do Cycles
z) Esquerda: resultado da UE4; Direita: resultado do Cycles; deixando claro a diferença entre os valores de preto e branco obtidos em cada sistema
Fonte: elaborado pelo autor.

Conforme exposto na FIG. 15, podemos ver nítida diferença das imagens obtidas. A parte “u” da FIG. 15 mostra grande diferença no resultado da sombra do objeto Suzanne: o resultado da esquerda (UE4) apresenta uma sombra mais sólida, pouco difusa, azulada e com aspecto mais escuro, com menos influência do ambiente, enquanto a da direita (Cycles) apresenta um resultado de sombra mais suave e com mistura de várias cores, pois há influência do ambiente.

Nas partes “v”, “w” e “x” podemos ver a diferença dos materiais metálico, anisotrópico e vidro. Isso ocorreu porque, diferente do Blender Cycles, na Unreal Engine 4 não haviam opções nativas de *shadings* que representassem esses

materiais. Por isso, foi necessário fazer adaptações com texturas e misturas de configurações para obter resultados semelhantes.

A parte “y” também apresenta diferenças nos resultados de SSS, mas dessa vez essa faz parte de uma característica própria do programa. Como a intenção da pesquisa não foi fazer uma réplica da cena renderizada utilizando o Blender Cycles, e sim fazer as duas cenas renderizadas com configurações semelhantes, alguns resultados apresentaram diferenças referentes ao que o programa proporciona. E esse resultado do SSS se trata de um desses resultados, uma vez que o *Subsurface Scattering* foi configurado com valores de configurações semelhantes aos configurados no Blender Cycles.

CONCLUSÃO

Os experimentos tiveram os resultados esperados, tanto no tempo de finalização quanto no visual. Mesmo apresentando grandes diferenças nas qualidades de resultado visual, o uso da renderização em tempo real se torna viável em projetos que não necessitam de materiais muito complexos, devido ao tempo em renderização que se ganha. Não foram feitos mais testes envolvendo os *shaders* na Unreal Engine 4, devido ao curto tempo que tivemos para realizar este artigo. Mas ainda é possível, em um outro projeto, uma equipe configurar materiais específicos às necessidades do projeto, conseguindo resultados complementares que foram obtidos nesse experimento, uma vez que esse resultado não necessariamente tem ligação com o potencial do computador envolvido. A renderização é um tema ainda pouco explorado academicamente, mas que vem tendo a atenção da indústria de jogos e cinema. É interessante tanto para os professores, quanto para os alunos da Escola de Belas Artes a implementação de meios de explorar essa mídia ainda recente, tanto para acompanhar, quanto para contribuir com seu desenvolvimento.

REFERÊNCIAS

1. TECHPOWERUP. GPU-Specs (Comparação GPU), 15 de jun. 2019. Disponível em: <www.techpowerup.com/gpu-specs/quadro-k600.c1839>. Acesso em: 14 de jun. de 2019.

2. INTEL. Comparação de especificações de produtos Intel, 15 de jun. 2019. Disponível em: <ark.intel.com/content/www/br/pt/ark/compare.html?productIds=99978,75779>. Acesso em: 14 de jun. de 2019.

3. BLENDER FOUNDATION. Blender Wiki, Denoise, 2018. Disponível em: <docs.blender.org/manual/de/dev/render/layers/denoising.html?highlight=denoise>. Acesso em: 15 de jun. 2019.

4. BLENDER FOUNDATION. Blender Wiki, Sampling, 2018. Disponível em: <docs.blender.org/manual/de/dev/render/cycles/render_settings/sampling.html>. Acesso em: 15 de jun. 2019.

5. BLENDER FOUNDATION. Blender Wiki, Light paths, 2018. Disponível em: <docs.blender.org/manual/de/dev/render/cycles/render_settings/light_paths.html>. Acesso em: 15 de jun. 2019.

6. BLENDER FOUNDATION. Blender Wiki, Performance, 2018. Disponível em: <docs.blender.org/manual/de/dev/render/cycles/render_settings/performance.html>. Acesso em: 15 de jun. 2019.

7. ANDREW PRICE. 13 Ways To Reduce Render Times.Andrew Price, 07 de set. 2010. Disponível em <www.blenderguru.com/articles/13-ways-to-reduce-render-times>. Acesso em: 15 de jun. 2019.

8. BLENDER FOUNDATION. Blender Wiki, Firefly, 2018. Disponível em: <docs.blender.org/manual/de/2.79/render/cycles/optimizations/reducing_noise.html?highlight=firefly>. Acesso em: 15 de jun. 2019.

9. COPYRIGHT © 2019 JOEY LENZ. Tech Art Portfolio. Joey Lenz, 2019.

Disponível em: <www.polyplant.co/tech-art-portfolio.html#ue4realtimeraytracingguide>. Acesso em: 15 de jun. 2019.

10. BLENDER FOUNDATION. Blender Wiki, Subsurface Scattering, 2018. Disponível em: <docs.blender.org/manual/de/2.79/render/blender_render/materials/properties/subsurface_scattering.html?highlight=sss>. Acesso em: 15 de jun. 2019.

11. BIRN, Jeremy. "*Lighting & Rendering*". Berkley, CA, EUA. 2006.

12. BLENDER FOUNDATION. Blender Wiki, Vertices e Faces, 2018. Disponível em: <https://docs.blender.org/manual/de/dev/modeling/meshes/structure.html#vertices>. Acesso em: 15 de jun. 2019.

GLOSSÁRIO

Anisotrópico (*Anisotropic*): São materiais que têm diferentes propriedades visuais dependendo da posição de onde são vistos. No Cycles, apresenta reflexos direcionados pela superfície e não pela fonte da luz [11].

Blender: Um programa de computador livre, de código aberto, que oferece diversas funções relacionadas à Computação Gráfica. Nesta pesquisa foram utilizadas apenas as opções de modelagem, texturização, desenvolvimento visual, animação e renderização [6].

CPU (*Central Processing Unit*): Peça de processamento principal dentro de um computador, responsável por todas as operações básicas realizadas pela máquina [6].

Cycles: Renderizador disponível no programa Blender.

Difuso (*Diffuse*): Tipo de material caracterizado por quando a luz é dispersa uniformemente em todas as direções, deixando o objeto opaco e sem reflexos

diretos.

Distância focal (*Focal Lenght*): Propriedade da fotografia, apropriada na Computação Gráfica, que simula o visual de uma imagem determinada pela distância entre o objeto e a lente.

***Game Engine*:** Programa de computador desenvolvido e utilizado para criação de jogos.

Geometria: Campo de estudos da Matemática. No campo da modelagem tridimensional se refere à quantidade e disposição de polígonos nos objetos e na cena como um todo.

***Glossy*:** Tipo de material caracterizado por quando a luz preserva a direção dos raios luminosos, mas ainda causa um pouco de dispersão na superfície, apresentando uma sutileza, deixando o objeto ainda com aparência opaca, mas apresentando reflexos do ambiente.

GPU (*Graphics Processor Unit*): Unidade de processos gráficos. Geralmente se refere a equipamentos descritos como placas de vídeo.

HDRi (*High Dynamic Range Image*): Imagens geradas por meio fotográfico, para alargar o alcance dinâmico, contendo informações de fontes luminosas. Pode se tratar também da extensão do formato de arquivo dessas imagens (.hdr), que são usadas como textura de iluminação.

Materiais: São configurações que definem o visual do objeto, determinando a estes como reagir com a luz.

Material Texturizado: Materiais que fazem uso de arquivos, de imagens ou não, como textura.

Metálico: Material configurado de maneira a ter um visual metálico, geralmente reconhecido por reflexos da cor do material.

Modelagem Tridimensional: Técnica em Computação Gráfica de manipulação de vértices e faces para gerar objetos tridimensionais.

Nó: Unidade de configuração de materiais no Blender e na Unreal Engine 4.

Point Light: Tipo de lâmpada virtual que simula uma lâmpada comum, com raios partindo de um ponto luminoso e seguindo em todas as direções.

***Raytracing*:** É uma parte opcional do processo de renderização que simula o reflexo, refração e sombreamento natural pelas superfícies 3D, mas trabalha de forma diferente, pois no *raytracing* os raios de luz começam da câmera para a cena.

Renderizador: Programa integrado ou não no programa de modelagem tridimensional, responsável pela renderização.

Renderizador Externo: Renderizador que não se encontra integrado a um programa de modelagem tridimensional.

Renderização: É o processo de finalização de uma imagem em qualquer mídia.

RTGI (*Ray Tracing Global Illumination*): Função de *raytracing* que simula a iluminação global, os rebatimentos de luz nos objetos da cena, causando emissões secundárias [11].

RTR (*Ray Tracing Reflections*): Função de *raytracing* que simula os reflexos de objetos pela luz, fazendo os objetos refletirem uns aos outros na cena tridimensional. Na Unreal Engine 4 com o *raytracing* desativado, os objetos usam um procedimento chamado "*screencap reflections*" onde o objeto apenas reflete as lâmpadas e os objetos que estão sendo capturados pela câmera [11].

Ruído: Efeito visual gerado por algoritmos que aparenta vários pontilhados aleatórios, lembrando os "chiados" de televisão. Entre renderizações *raytracing*, que o renderizador gera um ponto na imagem por raio de luz lançado, ruídos é como chamamos o aspecto visual gerado em imagens que não tiveram raios de luz o suficiente para formar uma imagem uniforme [8].

***Subsurface Scattering* (SSS):** Material que simula objetos semi-translúcidos onde os raios de luz entram no objeto, se espalham ao redor de seu interior e depois saem em um local diferente. Geralmente utilizado em materiais orgânicos como pele, frutas ou até inorgânicos, como algumas gemas e minerais [10].

Suzanne: Nome do objeto tridimensional disponível no programa Blender que se assemelha à cabeça de um macaco, usado como objeto com formas dinâmicas.

Textura Procedural: Textura gerada por algoritmos, diferente de texturas que utilizam arquivos de imagens.

Triângulo: Área criada entre três pontos (vértices), representando a superfície de um objeto tridimensional. Triângulos são sempre achatados, por tanto fáceis de calcular [12].

Unreal Engine 4: *Game engine* criada e gerida pela Epic Games™. Nessa pesquisa foi apenas abordada como renderizador, ignorando todas as outras ferramentas responsáveis para a criação de um jogo.

Vaga-lumes (*Fireflyes*): São pequenos pontos de brilho intenso que aparecem eventualmente no renderizador Cycles. *Fireflyes* é o termo popular na comunidade de usuários do Blender [8].

Vértice: É a parte mais simples de um objeto 3D, representando um ponto no espaço tridimensional [12].