

VISEDU LIGHT: VISUALIZADOR DE RAY TRACING

Daniel Rossato Martini, Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação

Departamento de Sistemas e Computação

Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

drmartini@furb.br, dalton@furb.br

Resumo: Este artigo descreve o desenvolvimento de um programa para aprendizado ray tracing, bem como os principais testes aplicados e os resultados obtidos. O programa foi desenvolvido em C# com Visual Studio 2019 e utilizando a ferramenta Unity. O programa tem o objetivo de ensinar ray tracing de forma mais prática, permitindo o estudante fazer alterações em tempo real para ver como elas afetam a renderização da imagem, assim como oferecendo explicações das técnicas que estão sendo demonstradas. De forma a avaliar o desempenho do programa, foram feitos alguns cenários de testes separados pelas áreas do programa. Com os testes realizados, o programa mostrou que cumpriu o esperado, explicando para os usuários como o ray tracing é aplicado em uma renderização.

Palavras-chave: Aprendizado. Computação gráfica. Ray tracing. Unity.

1 INTRODUÇÃO

O ray tracing não é apenas um algoritmo, mas sim a junção de vários algoritmos, os quais foram desenvolvidos a partir de um trabalho publicado por Appel em 1968. Em 1979 e 1980, Kay e Whitted expandiram a ideia, tendo adicionado cálculos mais corretos da iluminação especular e da refração de luz (LOPES, 2000, p. 1).

De acordo com Pacheco (2008, p. 3, tradução nossa), “Ray tracing, de forma geral, não é nada menos do que uma simulação perfeita da luz. [...] A única diferença é que algoritmos de ray tracing seguem o caminho da luz num caminho inverso, [...]. Assim, muitas vezes se torna difícil visualizar como o ray tracing está sendo utilizado dentro de uma cena, se tornando um algoritmo bem complexo. Muitas variáveis podem afetar renderizações com ray tracing, como a cor de um objeto refletir em outro objeto, ou então a textura de um objeto torna a luz difusa.

Um dos primeiros exemplos de uso de ray tracing em animações e filmes foi em Carros, pela Pixar/Disney, onde os carros eram bem curvos, com superfícies bem reflexivas (PACHECO, 2008, p. 5). Até a pouco tempo, apenas em cenas renderizadas offline era possível fazer o uso de ray tracing, pois, como nota Batali et al. (2006, p. 5), a renderização de algumas cenas pode levar muito tempo, chegando a mais de 100 minutos, o que impossibilita seu uso em qualquer aplicação em tempo real, como jogos.

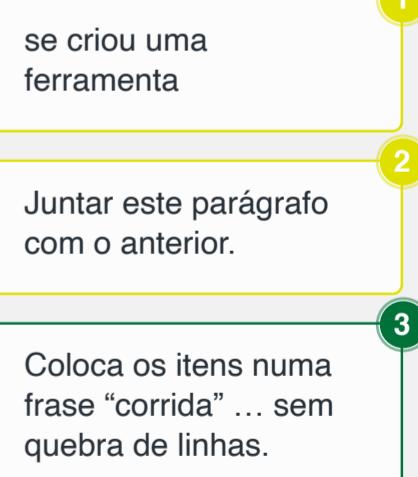
Em 2018, a NVIDIA lançou sua nova geração de placas de vídeo, a série RTX, que contam com núcleos específicos para calcular ray tracing (NVIDIA, 2018). Ainda de acordo com a NVIDIA (2018), esses núcleos de ray tracing dão às placas de vídeo poder suficiente para fazer uso de ray tracing em tempo real em jogos. Com isso há um novo foco em ray tracing, porém por não ter sido muito explorado recentemente, não há muitas formas de se entender como o ray tracing funciona e nem como ele afeta as renderizações.

Com base nesses argumentos, **1** propõe-se a criação de uma ferramenta para a visualização de como funciona o ray tracing, adicionando explicações para que estudantes da área de computação gráfica possam compreender essa tecnologia.

2 O objetivo deste trabalho é disponibilizar um ambiente para visualização e aprendizado de iluminação sobre a tecnologia de ray tracing.

3 Os objetivos específicos são:

- a) disponibilizar três cenas para simulação de ray tracing;
- b) apresentar uma explicação de como está ocorrendo o ray tracing na cena;
- c) permitir alterar textura dos objetos;
- d) permitir alterar cor dos objetos.



2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são apresentados os principais assuntos pertinentes à aplicação, sombras criadas com ray tracing, reflexos e refrações, path tracing e denoiser. Também são apresentados os trabalhos correlatos ao programa.

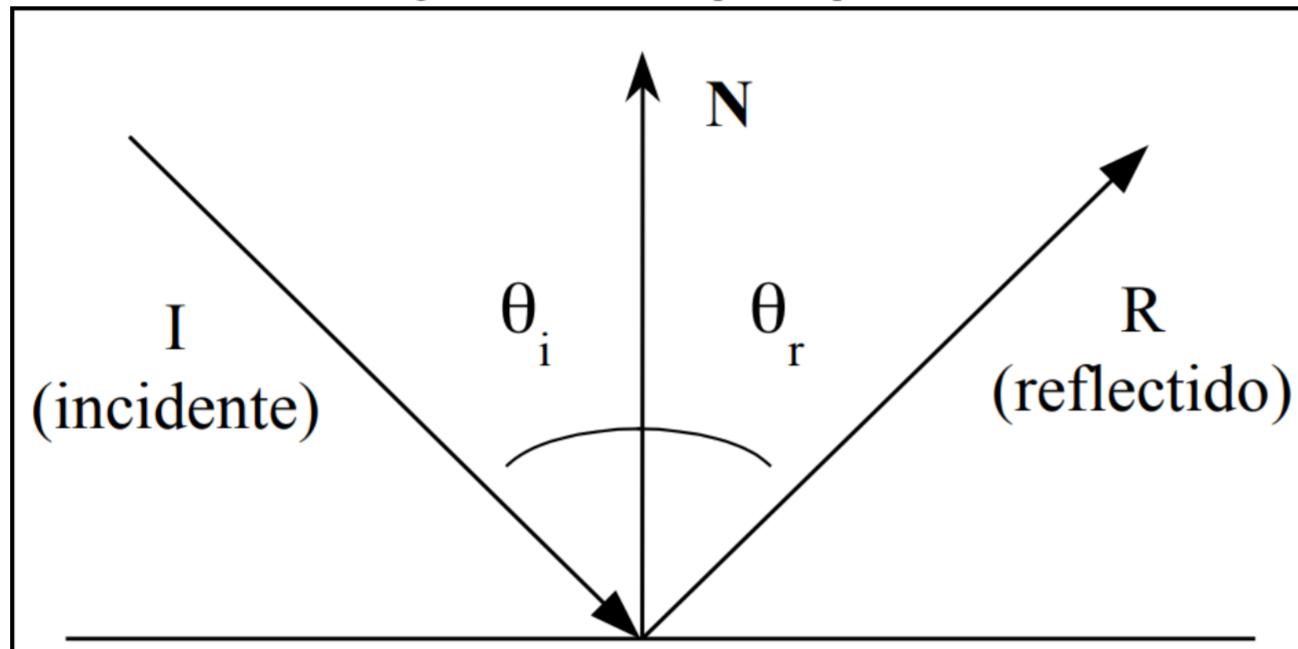
2.1 RAY TRACING

De acordo com Pacheco (2008, p. 1, tradução nossa), “O *ray tracing* pode ser visto como um algoritmo recursivo para calcular a cor de um pixel”. Ainda de acordo com Lopes (2000), existem três tipos de raios com funções distintas, sendo eles os raios refletivos, refratados e de iluminação direta ou de sombra. Dentro dos raios de reflexão, existem dois tipos, a reflexão especular perfeita e a reflexão difusa perfeita.

Reflexão especular perfeita ocorre em superfícies polidas e brilhantes, geralmente metálicas, de vidro ou espelhadas. Na realidade não existem superfícies refletivas perfeitas, porém o modelo de reflexão especular perfeita **providencia** uma aproximação suficientemente parecida com as reflexões que ocorrem em um ambiente real. A Figura 1 demonstra como ocorre a reflexão especular, onde um raio é refletido no mesmo ângulo de entrada, na imagem indicado como Θ (LOPES, 2000).

1
providência

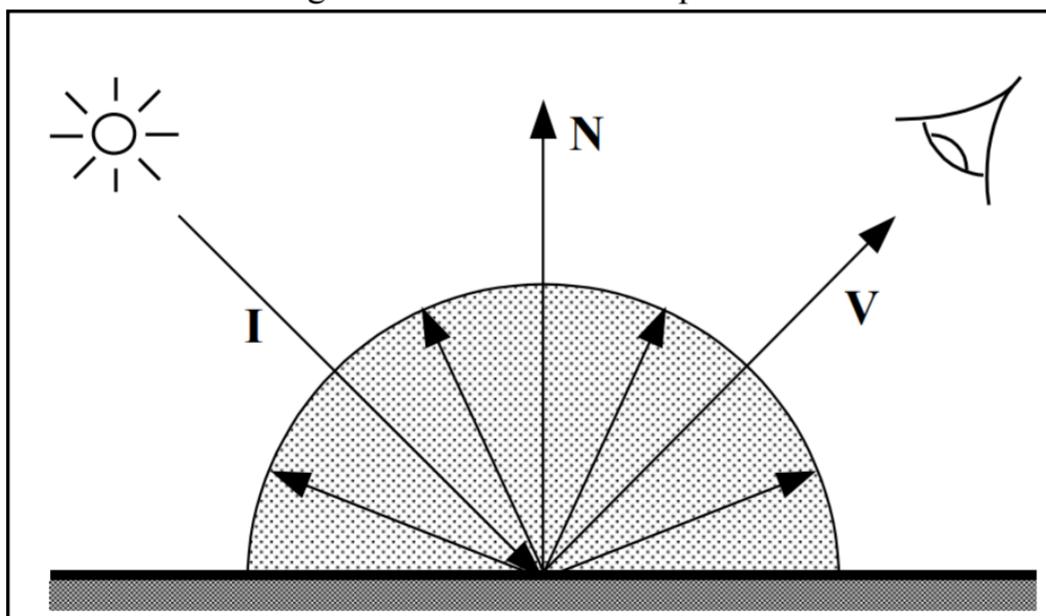
Figura 1 - Reflexão especular perfeita



Fonte: Lopes (2000).

Reflexão difusa perfeita ocorre em superfícies irregulares, nela não existe uma única direção que o raio de luz pode seguir, na verdade a luz se dispersa em igual densidade para todas as direções. A Figura 2 exemplifica como ocorre reflexão difusa perfeita (LOPES, 2000).

Figura 2 - Reflexão difusa perfeita



Fonte: Lopes (2000).

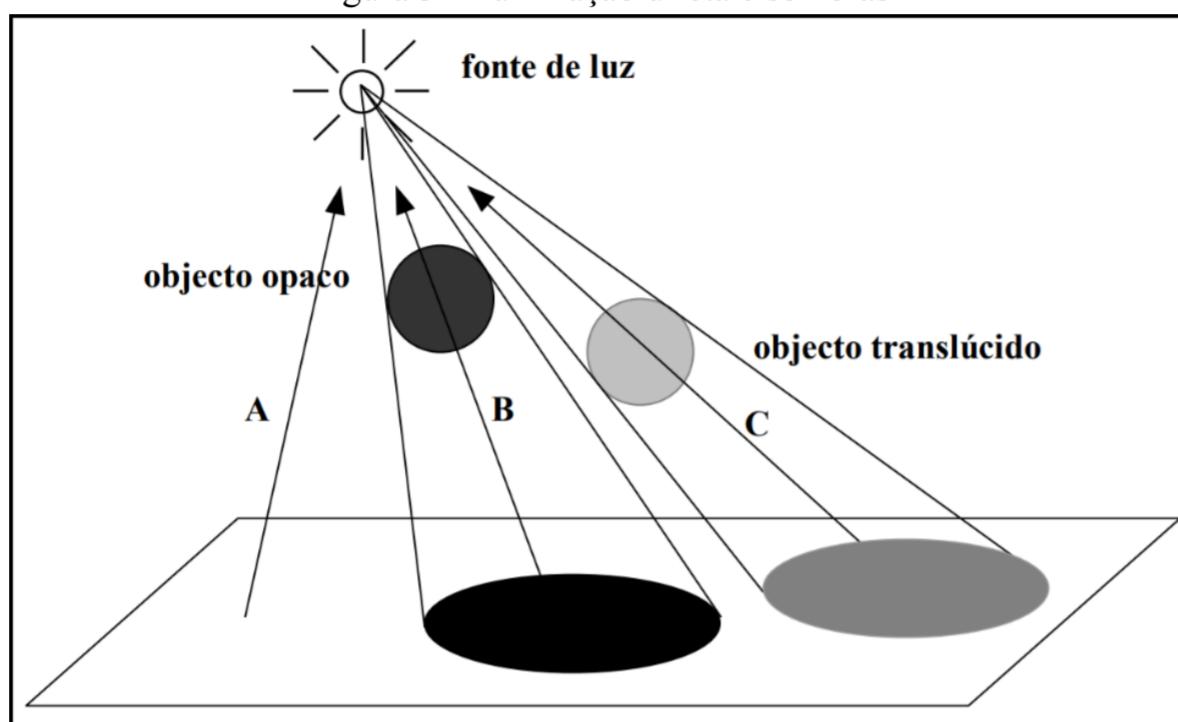
A refração é uma outra propriedade que deve ser levada em consideração no *ray tracing*. Segundo Lopes (2000, p. 19), a luz se propaga por objetos transparentes ou translúcidos, por esse motivo é possível ver através de vidros, mas isso também faz com que os objetos aparentam estar mais próximos do que realmente estão caso esteja dentro da água. Isso é causado pela diferença da velocidade de propagação da luz em diferentes meios, quando um raio vai de um meio para outro, a direção do raio é alterada. O grau de alteração da direção desse raio depende do índice de refração do meio que o raio acabou de entrar.

A definição de sombras é feita utilizando-se de raios secundários, criando-se um raio do ponto de intersecção do raio principal com um objeto, como o chão, até cada uma das fontes de luz. Na Figura 3, existe o raio A, que por não ter nenhuma intersecção, sabe-se que é um local iluminado. No raio B, há uma intersecção com a fonte de luz, então não há

2
Alinhamento da margem direita do parágrafo.

luz no local abaixo desse objeto, já o raio C intersecciona com um objeto translúcido, então há luz no local, porém reduzida.

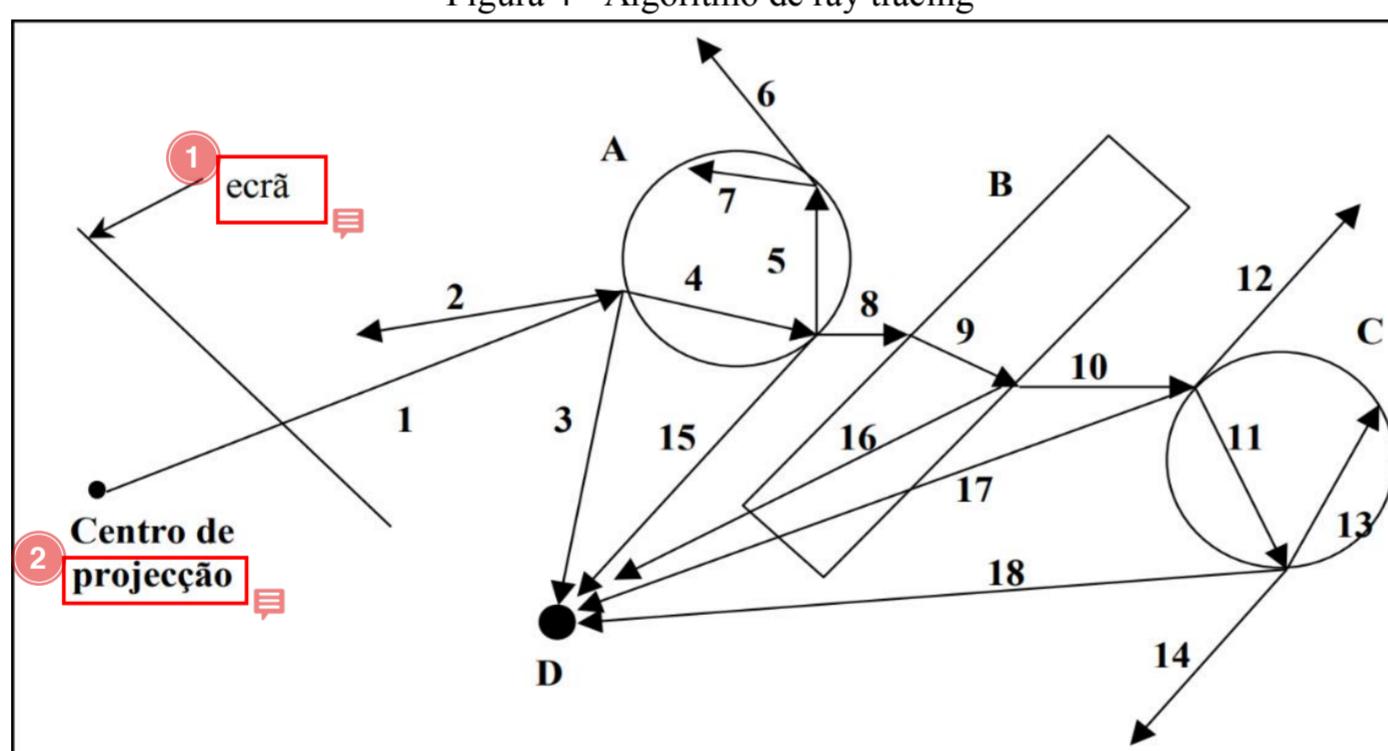
Figura 3 - Iluminação direta e sombras



Fonte: Lopes (2000).

De forma simples, o algoritmo completo de *ray tracing* funciona lançando um ou mais raios de cada pixel da tela, então para cada raio verifica-se as intersecções com os objetos, quando o raio se intersecciona com algum objeto, verifica-se se esse objeto gera reflexão difusa ou especular, gerando o raio secundário apropriado. De forma recursiva então são gerados novos raios, até que eles atinjam uma fonte de luz ou a quantidade máxima de novos saltos for atingida. A Figura 4 demonstra o algoritmo em um ambiente simples, contendo alguns objetos translúcidos, o número em cada um dos raios é a sequência em que eles foram criados e o item D é a fonte de luz.

Figura 4 - Algoritmo de ray tracing



Fonte: Lopes (2000).

2.2 PATH TRACING

3 *Path tracing* foi definido como sendo um único algoritmo para resolver uma grande variedade de algoritmos de renderização, ele 4 providênia um contexto unificado de aproximações para visualizar a solução de uma única equação (KAJIYA, 1986). Ainda de acordo com Kajiya (1986), isso não é uma surpresa, pois todos os métodos de renderização tentam modelar o fenômeno físico da luz refletindo sobre diversos tipos de superfícies.

Kajiya (1986), pegou como base a equação de radiosidade, balanceando o fluxo de energia de um ponto da superfície para outro. Kajiya (1986, p. 1, tradução nossa) define, “[...] o transporte da intensidade da luz de um ponto da superfícies para outro é simplesmente a soma da luz emitida e a intensidade total da luz que foi espalhada em direção à x vindo de todos os outros pontos de superfície”.

4 5

Trabalho de Conclusão de Curso - Ano/Semestre: 2020/1

Traduz este texto para “tela”.

Traduz este texto para “projeção”

renderização. Ele

Pouco conteúdo, mais uns dois parágrafos.

Faltou descrever sobre “denoiser”.

1

2

3

4

5

2.3 TRABALHOS CORRELATOS

São apresentados três trabalhos semelhantes ao trabalho proposto. O primeiro é o trabalho de conclusão de curso de Koehler (2015), que desenvolveu um visualizador de material educacional voltado à renderização. O segundo é o POV-Ray (PERSISTENCE OF VISION RAYTRACER PTY, 2013), uma ferramenta para criar imagens tridimensionais com *ray tracing*. Por último, é apresentado o trabalho feito por Peterrier, Thalmann e Vexo (2006) o Mental Vision, uma plataforma para ensino de computação gráfica.

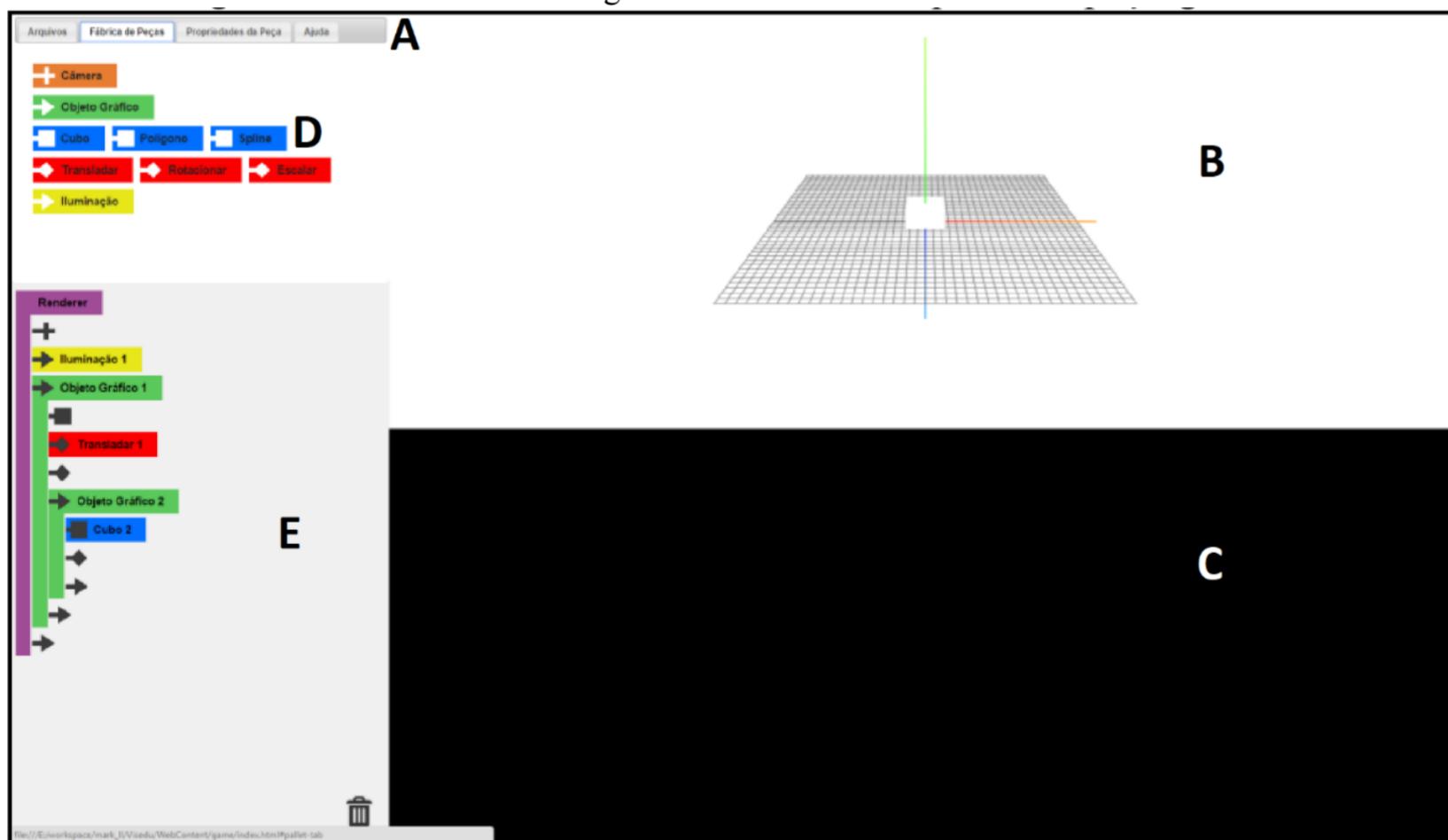
2.4 VISSEDU-CG 4.0

Koehler (2015) integrou dois outros trabalhos, sendo eles o VisEdu-CG 3.0 criado por Nunes (2014) e o projeto de Harbs (2013). A aplicação serve para facilitar o ensino de computação gráfica para alunos, na aplicação podem ser criadas cenas, que servem para facilitar a visualização das diversas propriedades existentes.

A Figura 5 apresenta a interface da aplicação, que contém 4 abas (item A). A primeira é Arquivos, ela serve apenas para importar e exportar cenas criadas. A segunda aba, Fábrica de Peças, é dividida em 4 partes, sendo a área B a cena que está sendo criada, e a área C mostra o que a câmera está captando, que é o que está sendo renderizado. O item D são as peças, como objetos e propriedades que podem ser adicionados à cena, e no item E é onde são adicionadas as peças à cena. Dentre os objetos que podem ser adicionados estão as fontes de luz, que funcionam de forma bem simples, apenas iluminando aquilo que atingem, sem reflexões. A terceira aba é a Propriedades da Peça, nela são definidas as propriedades, como posição e textura, da peça que está selecionada. Por último há a aba Ajuda, nela há apenas algumas informações de como o programa funciona.

O trabalho roda no navegador web onde uma tela é exibida utilizando HTML5 e Javascript. Já o JQuery é utilizado para definir os comportamentos mais complexos da tela, como arrastar e soltar as peças. Para criar e manipular os elementos gráficos foi utilizada a biblioteca Three.js que serve para abstrair o WebGL. O WebGL utiliza a *Graphics Processing Unit* (GPU) do computador para realizar a renderização (KHRONOS GROUP, 2019).

Figura 5 - ViseduCG 4.0



Fonte: Koehler (2015).

Koehler (2015) atingiu seus objetivos conseguindo adaptar o motor gráfico para utilizar o Three.js. Assim a aplicação suporta ambientes tridimensionais com muitos objetos gráficos. Mesmo com muitos objetos, a biblioteca se manteve estável mostrando assim que há espaço para aplicações gráficas exigentes na web. O autor destaca também que o HTML5, CSS, Javascript e JQuery permitiram um controle mais simples e eficaz sobre a tela em geral.

2.5 POV-RAY

O POV-Ray (PERSISTENCE OF VISION RAYTRACER PTY, 2013) é um editor livre, disponibilizado para Windows, que permite criar e renderizar cenas utilizando *ray tracing*. Para isso, é disponibilizada uma tela para edição

(2006), o

ATENÇÃO: os trabalhos correlatos devem ser descritos no artigo usando quadros.

Ver nos exemplos dos artigos.

Parei de revisar.

alunos. Nesta aplicação

cenas que

aplicação que

partes. Já a

B representa a cena

criada. A área

cena. E no item

textura da

Não revisei .. mudar formato para quadro.

1

2

3

4

5

6

7

8

9

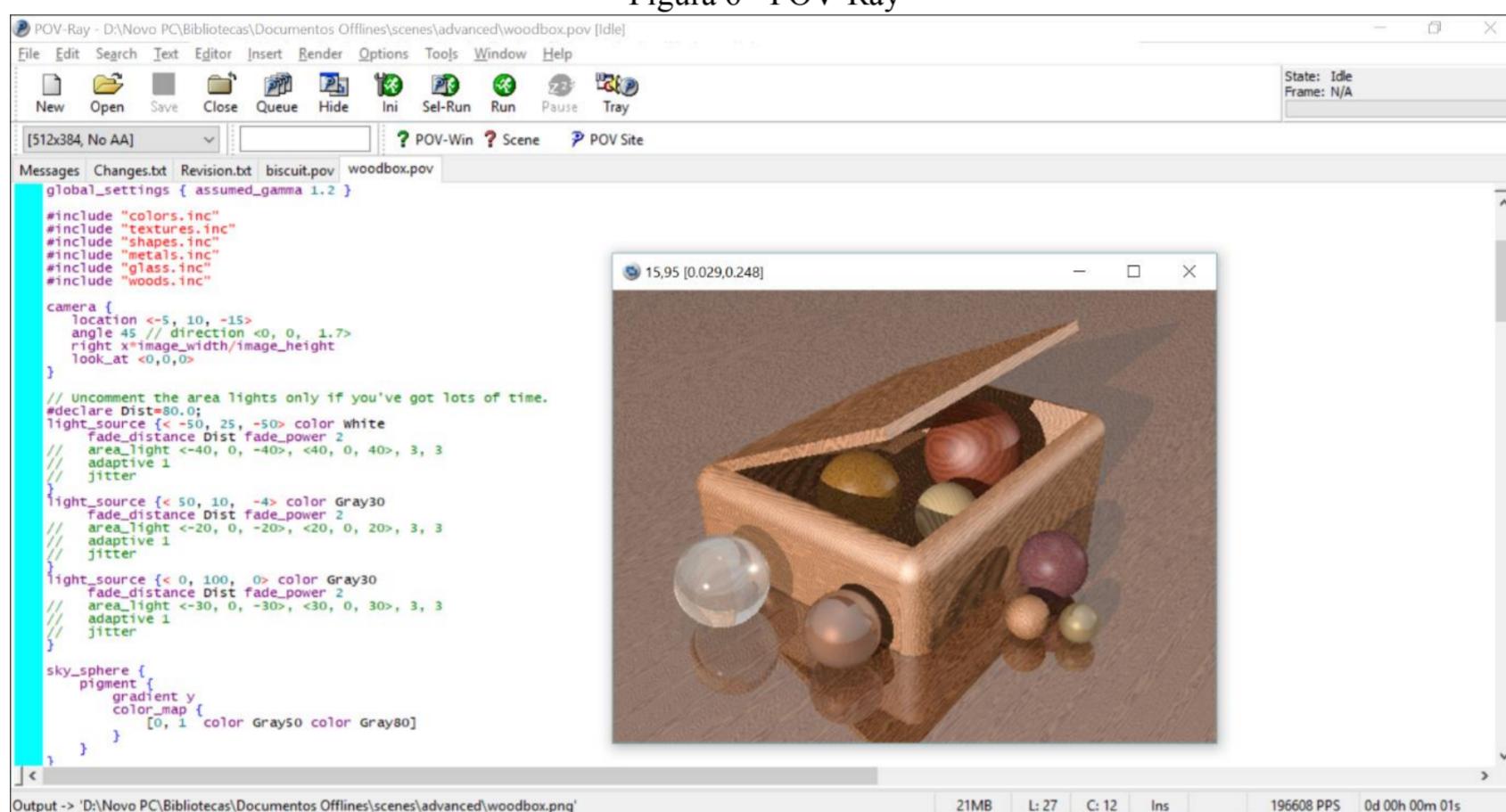
10

11

de código manualmente conforme a Figura 6, na qual é utilizada uma linguagem própria, que suporta diversos tipos de objetos, como esferas e retângulos, além de diversos tipos de fontes de luz, entre outros. Para que não seja necessário escrever tudo manualmente, é possível inserir códigos de objetos a partir de uma lista bem extensa de opções. Após inseridos, o usuário pode manualmente fazer alterações como preferir. A complexidade do programa é alta, visto que é necessário conhecer a linguagem própria e, como não há um editor visual, tudo é feito via código.

A renderização é feita utilizando-se de *ray tracing* em toda a cena, utilizando as fontes de luz definidas no projeto e levando como base as propriedades definidas para os objetos, como reflexibilidade e cor. A renderização pode por vezes demorar bastante para terminar, pois não é utilizada aceleração por GPU, sendo apenas utilizada a *Central Processing Unit* (CPU).

Figura 6 - POV-Ray



Fonte: Persistence of Vision Raytracer PTY (2013).

A alta complexidade do programa permite que se criem imagens bem complexas e realistas utilizando-se do *ray tracing*. O programa tem uma área de ajuda, porém não faz nenhuma explicação do que se está passando na hora da renderização.

1

2.6 MVISIO

1

Não revisei .. mudar formato para quadro.

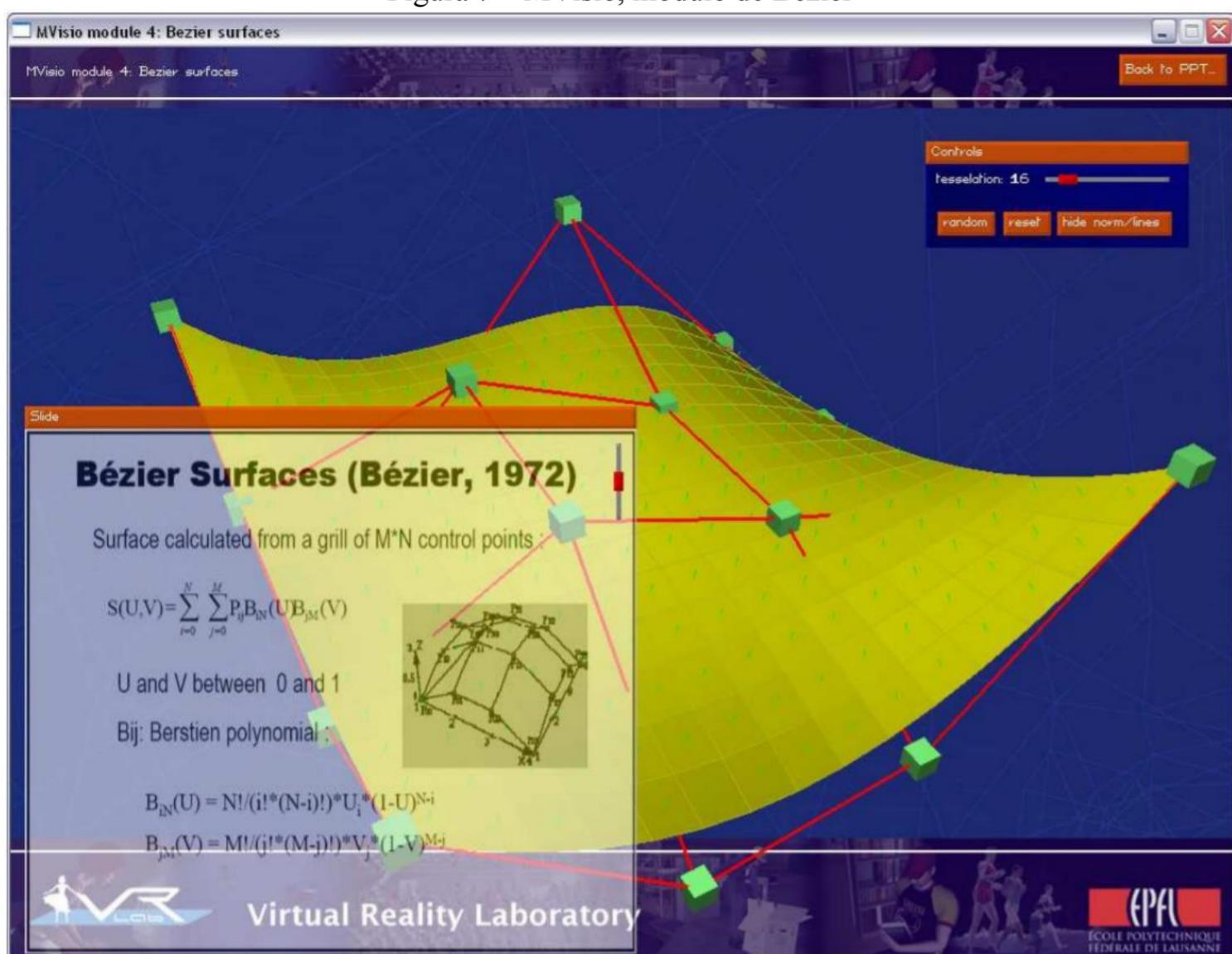
Petermier, Thalmann e Vexo (2006) criaram uma plataforma para simplificar e melhorar o ensino e prática de computação gráfica, denominada MVisio. A plataforma final é dividida em duas partes, uma é uma combinação de aplicações compactas que demonstram diversas técnicas e algoritmos de computação gráfica, denominada de módulos pedagógicos; a outra é um motor gráfico orientado à pedagogia para ser usado pelos alunos para projetos e trabalhos práticos.

Os módulos pedagógicos são usados para ilustrar de forma simples temas complexos da computação gráfica. A Figura 7 demonstra um módulo sobre a superfície de Bézier. Cada módulo fala sobre apenas um tópico para evitar confusão. Eles também permitem mudar diversas variáveis para ficar claro como elas funcionam e afetam a representação visual da cena. Para cada tópico há uma explicação detalhada sobre eles, mostrando fórmulas e desenhos sobre o tema.

O motor gráfico foi construído para ser fácil de usar e ser possível de rodar em vários tipos de *Personal Computers* (PC), inclusive *Personal Digital Assistant* (PDA), sem a necessidade de converter nada nos projetos. Para usuários mais avançados, ele abstrai as operações de baixo nível, permitindo que o usuário foque em criar aplicações em alto nível. É possível construir cenas complexas, com dezenas de objetos, texturas diferentes, porém não há vários pontos de luz, toda a cena é renderizada com uma iluminação uniforme e global.

O MVisio foi construído em C++ e utiliza o OpenGL, porém não utiliza a GPU, com isso as renderizações podem acabar sendo lentas, especialmente nos PDAs. Diversas funções necessárias para o motor gráfico, como carregamento de imagens foram construídas diretamente dentro dele. E isso somado ao fato que ele é orientado a objeto, permitiu que a aplicação ficasse leve.

Figura 7 – MVisio, módulo de Bézier



Fonte: Peternier, Thalmann e Vexo (2006).

3 DESCRIÇÃO DO PROGRAMA

Este capítulo visa apresentar os detalhes de especificação e desenvolvimento do programa. O capítulo apresenta quatro seções, sendo a primeira responsável por exibir algumas características do desenvolvimento. Como o programa foi criado com três salas, as próximas seções detalham cada uma delas: a segunda é destinada à uma visão geral do menu inicial e da sala Sombras. A terceira seção explica o que foi feito na sala Técnicas e a quarta seção apresenta explicações sobre a sala PathTracing. Para realizar o desenvolvimento do programa foram utilizados os requisitos apresentados no Apêndice A.

3.1 IMPLEMENTAÇÃO

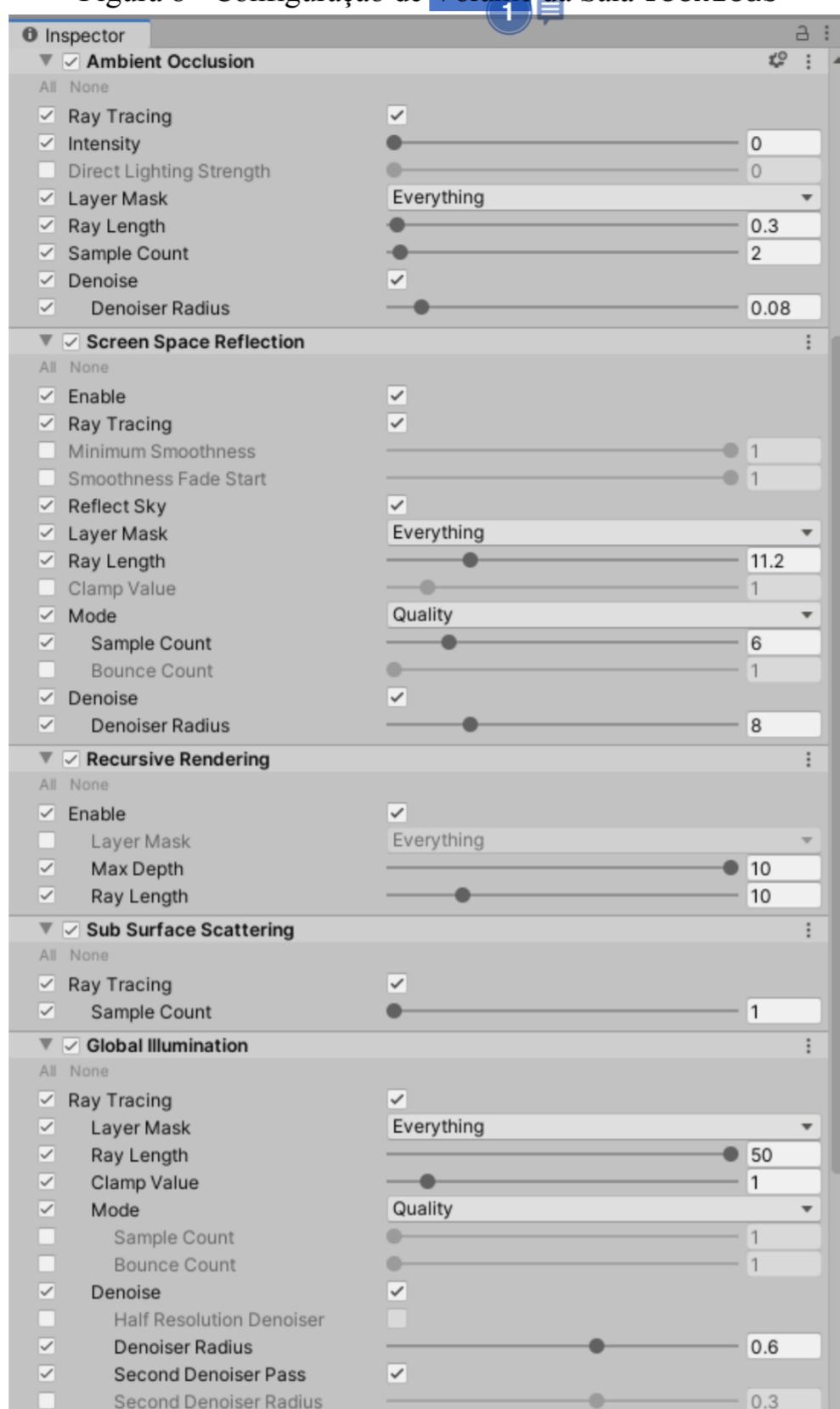
Para realizar a implementação do programa, foi utilizada o Unity, com scripts em C#. Para fazer uso do ray tracing foi utilizada uma versão beta do Unity, a versão 2019.3b. O Unity que permitiu fazer uso da API de ray tracing do DirectX12. Foi criado um projeto de *High Definition Render Pipeline* (HDRP), que é o tipo de projeto no Unity que permite fazer uso das API de ray tracing do DirectX12. Dentro do Unity foram utilizados os componentes Volume, no qual são adicionadas sobrecargas nas opções padrões de renderização, e com essas opções é possível fazer o uso do ray tracing.

As sobrecargas utilizadas no programa foram de *Ambient Occlusion* (AO), *Screen Space Reflection* (SSR), *Recursive Rendering* (RR), *Global Illumination* e *Path Tracing*. O AO é responsável de verificar o quanto escuro as pequenas áreas da cena devem ser, com ray tracing isso pode ser feito de forma mais precisa. O SSR é responsável por calcular os reflexos da cena, ele define que será refletido ou não, com o ray tracing ele se torna mais preciso. O RR é utilizado para fazer reflexos e refrações nas salas, ele é utilizado apenas com ray tracing. O *Global Illumination* é utilizado para verificar o quanto iluminado a cena como um todo deve ser, o ray tracing permite uma aproximação muito mais próxima à real. O *Path Tracing* sobrescreve a renderização de todos os outros componentes, dessa forma sempre é utilizado sozinho, ele é o mais pesado dos tipos que são utilizados nesse programa. Ele permite renderizar uma cena completa e de forma mais realista o possível, apenas utilizando ray tracing.

Cada uma das salas utilizou uma combinação diferente dessas sobrecargas para poder mostrar da melhor forma como cada uma delas afetam uma cena. Na Figura 8 há um exemplo do uso dessas sobrecargas na sala Técnicas.

- 1 delas. A segunda
- 2 Fonte courier.
- 3 programa se utilizou o Unity com
- 4 tracing se utilizou uma
- 5 2019.3b. Versão está do Unity
- 6 O próximo passo foi criar um
- 7 Esta tradução está certa?
- 8 não, mas com
- 9 salas, e ele
- 10 ser. E assim o ray
- 11 componentes, e dessa
- 12 sozinho. O Path Tracing é mais
- 13 custoso de processamento
- 14 O Path Tracing permite renderizar

Figura 8 - Configuração de Volume da Sala Técnicas



1
Está tradução está certa?

Fonte: elaborado pelo autor.

4 Foram também utilizados diversos assets disponíveis na Unity Store. O Quadro mostra quais assets foram utilizados para a implementação do programa.

Quadro 1 - Assets utilizados

Asset	Uso
MS Sports Car	Foi utilizado um carro em duas das salas.
Snaps Prototype Asian Residencies	Utilizado para adicionar as casas visíveis na sala PathTracing.
Snaps Prototype Construction Site	Assets leves utilizados para ver onde cada objeto deve ficar.
Snaps Art HD Construction Site	Assets de alta qualidade utilizados na sala PathTracing para ter uma cena realista.
Asset Swap Tool	Ferramenta utilizada para fazer a troca entre os assets leves para os assets de alta qualidade

Fonte: elaborado pelo autor.

2
Junta este parágrafo com o anterior.

3
alguns

4
Também se utilizou assets

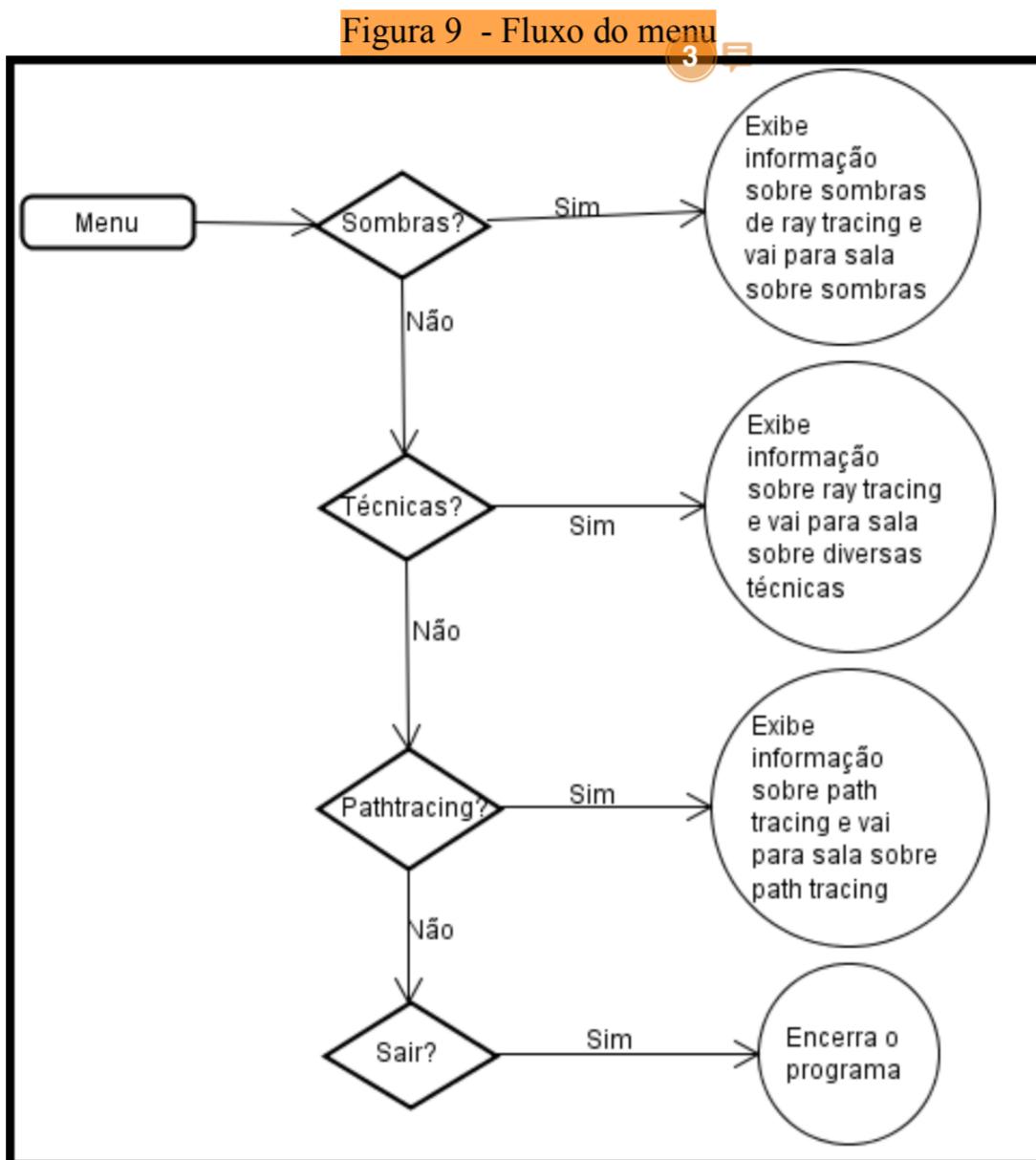
5
Store (Quadro 1).

6
utilizado o modelo gráfico de um carro

7
adicionar os modelos gráficos das casas

3.2 VISÃO GERAL DO MENU E SALA SOMBRAS

O programa tem por objetivo auxiliar o entendimento de *ray tracing* no contexto de renderização, ensinando principalmente alunos iniciantes na área de computação gráfica. O programa é composto pelo menu inicial e três salas, o menu tem três botões, cada qual quando clicados apresentam informações sobre a técnica escolhida e então leva o usuário à uma sala diferente. As salas apresentam explicações e exemplos sobre diferentes técnicas de *ray tracing* que podem ser aplicadas durante a renderização de uma imagem. Na Figura 9 há um fluxo do menu e na Figura 10 há uma imagem da tela do menu.



- 1 programa desenvolvido tem
- 2 Este programa
- 3 Diminui um pouco as figuras 9 e 10 e juntas ela numa só (lado a lado).

Fonte: elaborado pelo autor.

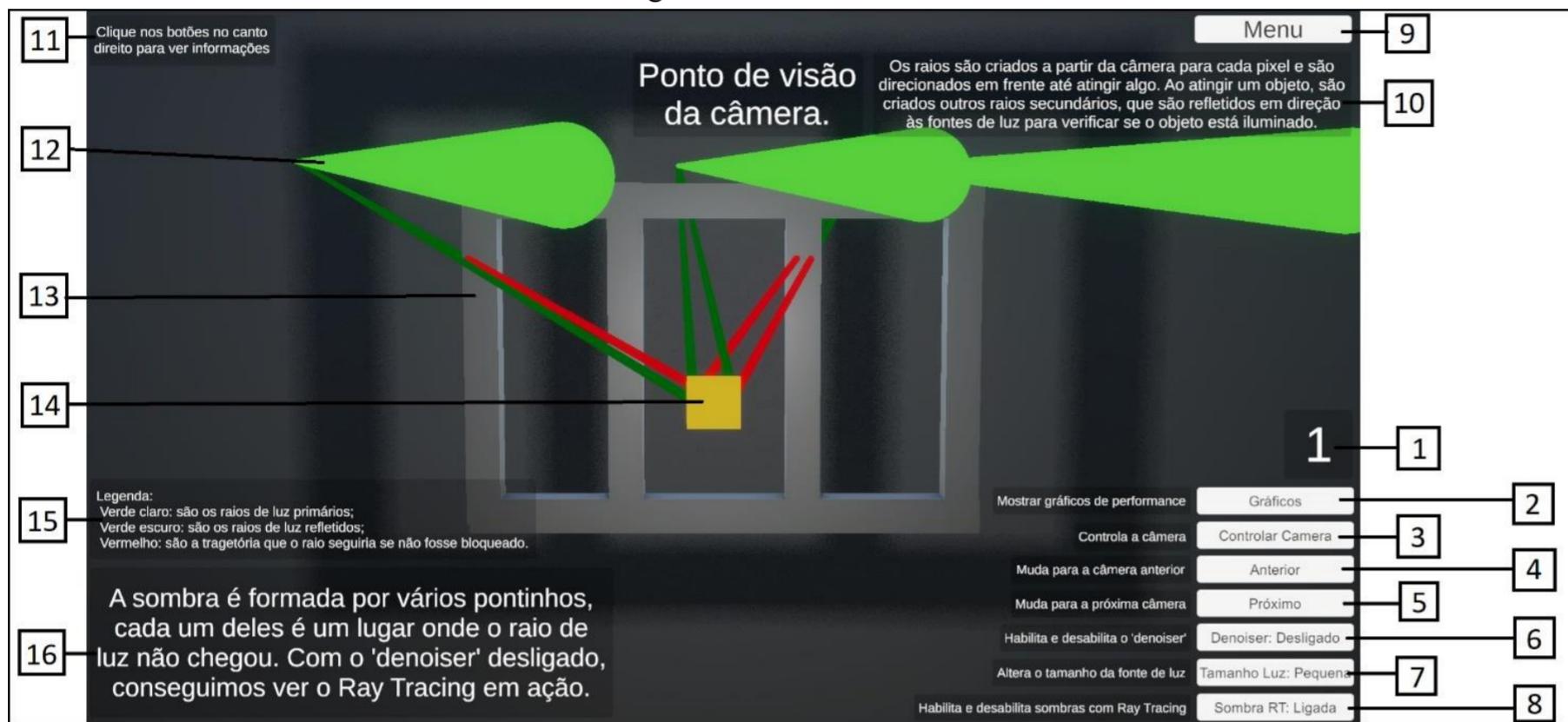
Figura 10 - Menu



Fonte: elaborado pelo autor.

Ao clicar no primeiro botão, o usuário é levado à sala sobre sombras de *ray tracing*, nesta sala o usuário é apresentado com uma cena exemplificando o uso de *ray tracing* para gerar sombras. O usuário pode fazer diversas alterações, como ligar e desligar as sombras de *ray tracing*, alterar o tamanho da fonte de luz, ligar e desligar o *denoiser*, alterar qual o ponto de visão e exibir gráficos de performance do sistema. A seguir a Figura 11 mostra essa sala com seus componentes identificados por números e o Quadro 2 apresenta a descrição de cada um dos componentes enumerados.

Figura 11 - Sala Sombras



Fonte: elaborado pelo autor.

Quadro 2 - Descrição da tela inicial da sala Sombras

Identificador	Descrição
1	Indica qual o ponto de visão atual.
2	Botão para exibir os gráficos sobre performance.
3	Botão para controlar a câmera.
4	Botão para ir ao ponto de visão anterior.
5	Botão para ir ao próximo ponto de visão.
6	Botão para ligar e desligar o <i>denoiser</i> .
7	Botão para alterar o tamanho da fonte de luz.
8	Botão para ligar e desligar as sombras de <i>ray tracing</i> .
9	Botão para voltar ao menu.
10	Caixas de texto explicando o ponto de visão atual.
11	Caixa de texto dando instrução sobre os botões.
12	Guia para demonstrar um raio de luz.
13	Objeto que gera a sombra.
14	Objeto para representar a fonte de luz.
15	Caixa de texto com a legenda das cores dos raios de luz.
16	Caixa de texto com explicação do <i>ray tracing</i> .

3 Fonte: elaborado pelo autor 4

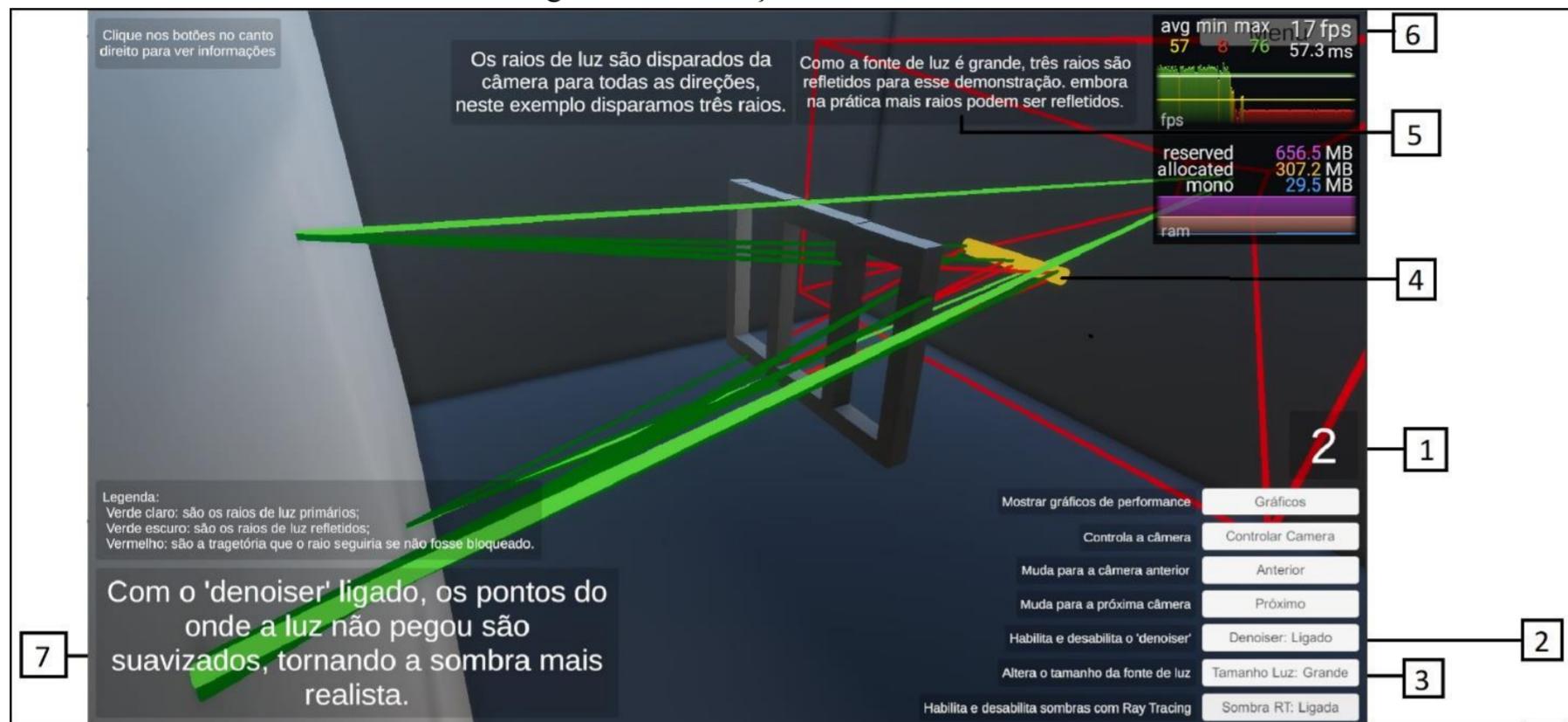
Conforme o usuário passa de um ponto de visão para o próximo, as explicações vão mudando para refletir aquilo que está sendo dado ênfase no momento, conforme o item 5 da Figura 12. Ao alterar o tamanho da fonte de luz, a cena muda para se adaptar ao novo tamanho da luz, criando novos raios para a demonstração e alterando a explicação exibida, pode ser visto no item 4 da Figura 12 a nova fonte de luz maior, em amarelo, junto com novos raios, e no item 3 a descrição do botão também é alterada. Caso o usuário clique no botão para exibir os gráficos, pode ser visto no item 6 da Figura 12 que aparece na tela um gráfico dos Frames Por Segundo (FPS) junto com uma informação da memória que está sendo alocada, com o FPS é possível analisar o impacto na performance com cada alteração feita na cena. Ao clicar no botão para ligar o *denoiser*, a explicação no canto direito inferior é alterada, conforme item 7 da Figura 12, e o botão de *denoiser* também muda sua descrição.

- 1 tracing. Nesta
- 2 algumas
- 3 Diminuir espaço entre figura e fonte.
- 4 Terminar frase com ponto final.
- 5 momento (Figura 12-5).
- 6 exibida. Na Figura 12-4 se pode ver a nova fonte
- 7 raios. Já no item 3 a descrição
- 8 gráficos (Figura 12-6) aparece na tela um gráfico
- 9 alocada. Com
- 10 denoiser a
- 11 alterada (Figura 12-7), e o

O usuário também pode alterar o ponto de vista, na Figura 12 é possível ver outro ponto de visão, bem como no item 1 há um outro número indicando que é outro ponto de visão. No ponto de visão 6 o usuário pode controlar a câmera, dessa forma permitindo visualizar algo que o tenha interessado mais de perto.

vista (na Figura 12 é possível ver outro ponto de visão), bem

Figura 12 - Alterações na sala de Sombras



Fonte: elaborado pelo autor.

3.3 SALA DE TÉCNICAS

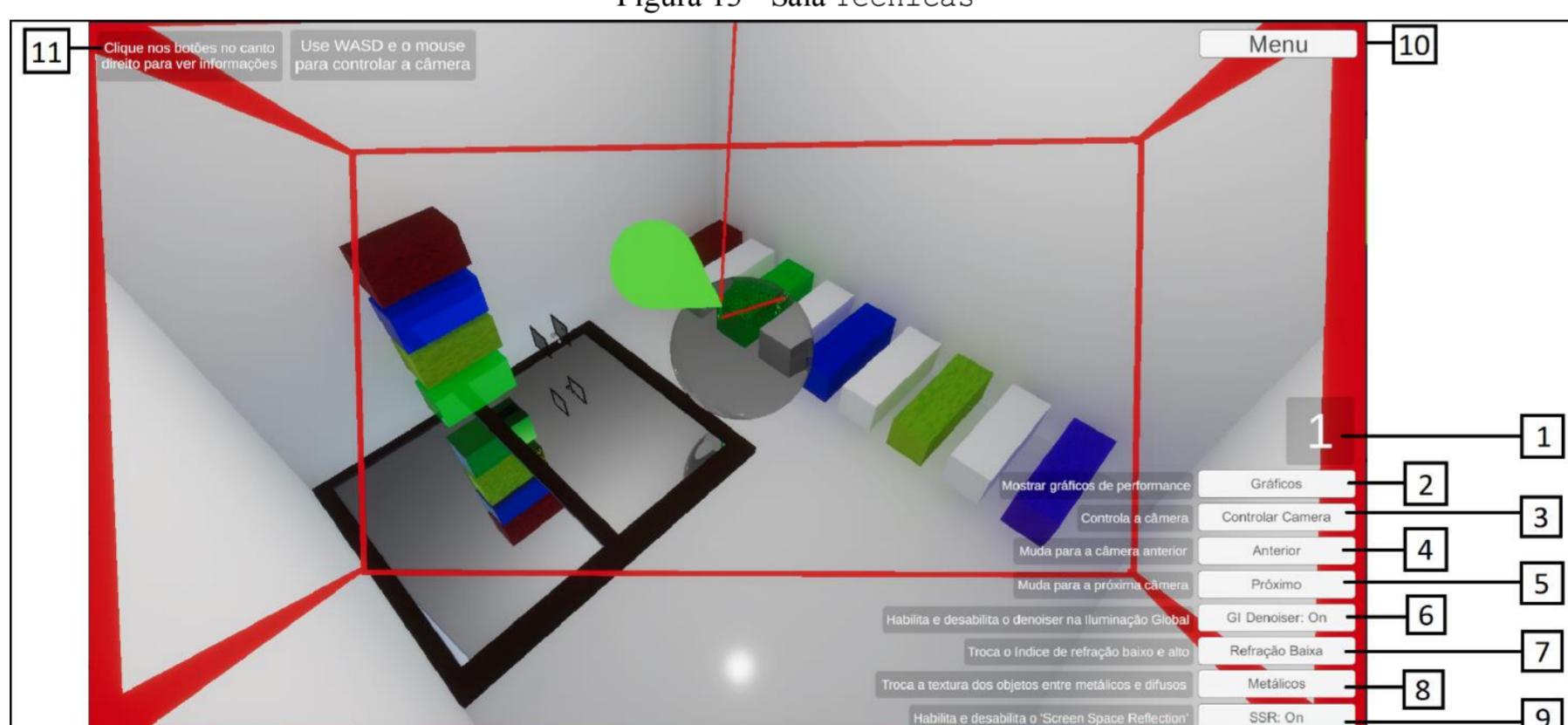
Quando o usuário clica no menu na sala Técnicas ele é direcionado à uma sala onde vão ser explicadas as técnicas de reflexos, refrações, iluminação global e dispersão de luz, todas utilizando *ray tracing*. Esta é a sala com mais explicações e pontos de visão do programa, porém sua estrutura é parecida com a sala anterior, com os botões de ações no canto direito inferior e com as caixas de texto aparecendo na tela do usuário conforme ele vai passando ponto de visão em ponto de visão. As opções de Gráficos, Controlar Câmera, Anterior e Próximo funcionam da mesma forma que já explicado na sala Sombras. A Figura 13 enumera cada um dos pontos importantes da tela e o Quadro 3 descreve esses pontos.

programa. Porém

refrações, iluminação

luz (todas utilizando ray tracing).

Figura 13 - Sala Técnicas



Fonte: elaborado pelo autor.

Quadro 3 - Descrição da tela inicial da sala Técnicas

Identificador	Descrição
1	Indica qual o ponto de visão atual.
2	Botão para exibir os gráficos sobre performance.

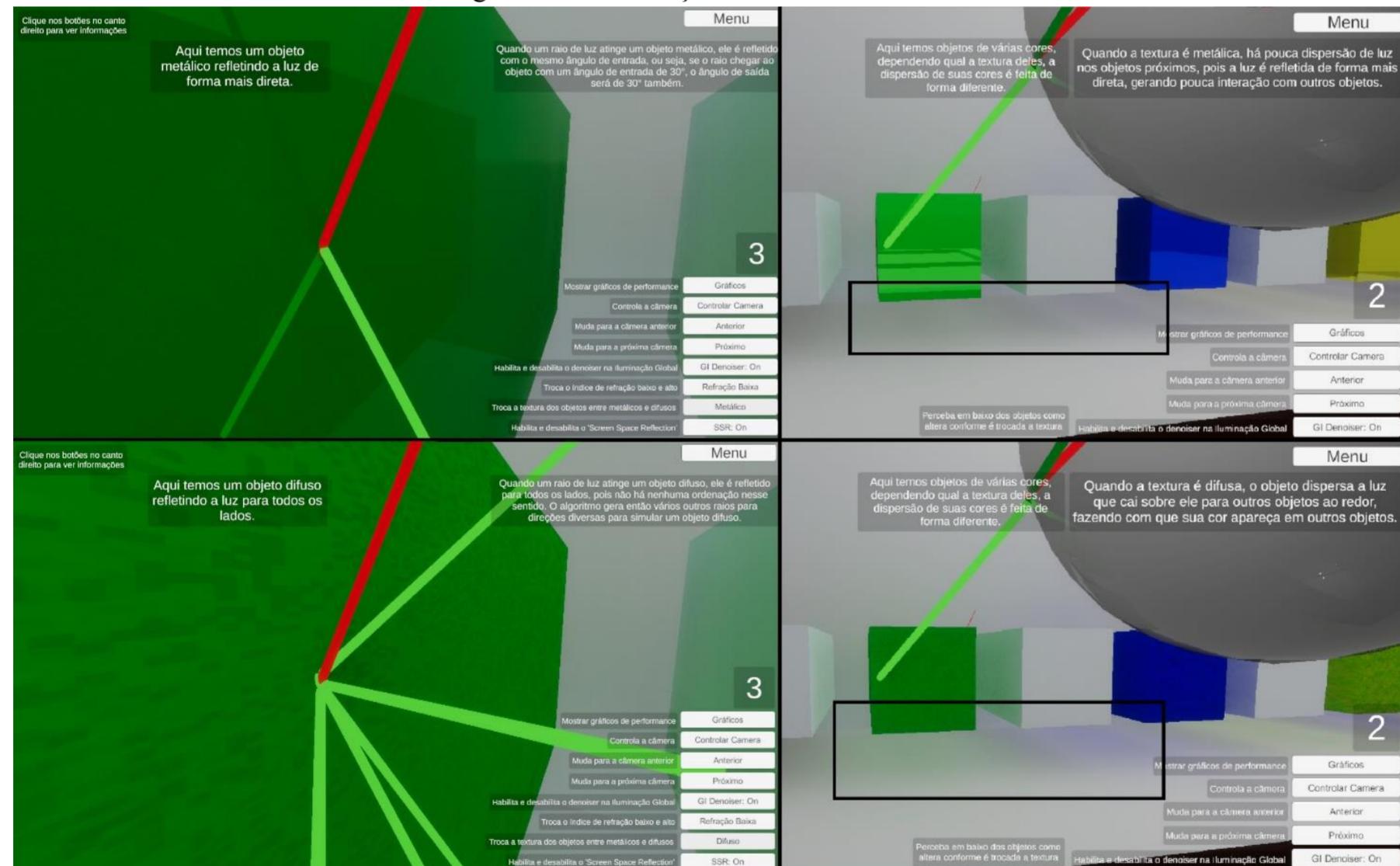
Cuidar, na versão final
não separar quadros,
figuras, legendas e
fontes pela quebra de
página.

3	Botão para controlar a câmera.
4	Botão para ir ao ponto de visão anterior.
5	Botão para ir ao próximo ponto de visão.
6	Botão para ligar e desligar o <i>denoiser</i> da Iluminação Global.
7	Botão para alterar o índice de refração dos vidros.
8	Botão para alterar o tipo dos objetos na cena entre metálicos e difusos.
9	Habilita e desabilita o <i>Screen Space Reflection</i> .
10	Botão para voltar ao menu.
11	Caixa de texto dando instrução sobre os botões.

Fonte: elaborado pelo autor.

As explicações da sala são feitas por etapa, conforme o usuário vai avançando clicando no botão **Próximo**, o ponto de vista vai mudando, e para cada novo ponto de vista há uma explicação nova. A primeira é sobre a difusão da cor de um objeto em outros objetos próximos e como objetos metálicos e difusos afetam a renderização de uma cena de formas diferentes. Na Figura 14 há uma demonstração de o que o usuário vê e como a cena é alterada quando o usuário muda os objetos de metálicos para difusos. Após essa explicação, é apresentado ao usuário algumas explicações sobre os reflexos com *ray tracing*, na tela é mostrado ao usuário dois espelhos, um usando o SSR e outro utilizando apenas *ray tracing*, junto com explicações sobre cada um deles e qual a diferença entre eles. O SSR pode também fazer o uso de *ray tracing*, e com o botão SSR o usuário pode habilitar e reabilitar ele para ver a diferença. Se o usuário seguir para o próximo ponto de vista é explicado mais a fundo sobre os reflexos utilizando apenas o RR, que faz uso apenas de *ray tracing*, fazendo uso de dois espelhos um de frente para o outro, onde o usuário pode ver como eles são refletidos um no outro diversas vezes.

Figura 14 – Diferença entre metálico e difuso



Fonte: elaborado pelo autor.

A terceira explicação é sobre a iluminação global com *ray tracing*, é dada ao usuário uma explicação de como é aplicado a iluminação global na cena e como o *ray tracing* pode melhorar bastante a sua qualidade, mesmo utilizando poucas amostras por pixel. O usuário tem a opção de desligar o *denoiser* da iluminação global, podendo dessa forma ver como o *ray tracing* está sendo aplicado na cena inteira.

A última explicação da sala é sobre a refração, e para essa explicação é mostrado ao usuário, utilizando-se de gizmos novamente, como o raio é afetado ao passar por um objeto que refrata a luz. Junto com a explicação visual há mais alguns textos explicando o que está acontecendo. O usuário tem a opção de alterar o índice de refração do objeto e ao fazer isso os textos que são exibidos para ele mudam novamente. Durante todo o percurso o usuário tem a opção de controlar a câmera, permitindo então que ele veja algo que o tenha interessado mais de perto.

1 Arrumar para fonte courier ... parece que ficou diferente.
2 primeira etapa é
3 do
4 tracing. Na
5 pode habilitar e desabilitar
6 vista é
7 RR. Esta técnica faz uso

8 tracing, onde é dada

9 sala Técnicas é

10 refração. E para

3.4 SALA DE PATHTRACING

Ao clicar no botão PathTracing no menu, o usuário é direcionado à sala que irá lhe mostrar diversas explicações sobre como *path tracing* é utilizado para realizar a renderização de uma imagem. Essa é a sala que contém a maior quantidade e qualidade de assets, bem como a técnica que mais demora para ser processada, então ela requer um computador bem forte para conseguir rodar bem.¹ A estrutura dos botões na tela é parecida com as duas outras salas, com os botões Gráficos, Controlar Câmera, Anterior e Próximo tendo os mesmos funcionamentos que nas outras salas.² A Figura 15 enumera os principais pontos e o Quadro 4 explica cada um dos pontos enumerados.

3

Figura 15 – Sala PathTracing



processada. Então esta sala requer

um computador com uma performance maior para garantir uma taxa de renderização mínima.

Próximo. E o funcionamento destes botões também são iguais aos das outras salas.

Fonte: elaborado pelo autor.

Quadro 4 - Descrição da tela inicial da sala PathTracing

Identificador	Descrição
1	Botão para voltar ao menu.
2	Indica qual o ponto de visão atual.
3	Botão para controlar a câmera.
4	Botão para ir ao ponto de visão anterior.
5	Botão para ir ao próximo ponto de visão.
6	Botão para ligar e desligar as luzes espalhadas pela cena.
7	Botão para alterar entre dia e noite.
8	Botões para alterar a quantidade de saltos que os raios de luz podem fazer.
9	Botão para exibir os gráficos sobre performance.
10	Botão alterar a configuração de <i>ray tracing</i> para a padrão.
11	Botão para que apenas iluminação direta seja mostrada.
12	Botão para que apenas iluminação secundário seja mostrada.
13	Botão para que apenas iluminação terciária seja mostrada.
14	Botão para que sejam utilizadas 8 amostras por pixel.
15	Botão para que sejam utilizadas 512 amostras por pixel.
16	Botão para que sejam utilizadas 2048 amostras por pixel.
17	Botão para que sejam utilizadas 4096 amostras por pixel.
18	Caixa de texto dando instrução sobre os botões.
19	Caixas de texto explicando sobre as luzes na cena.
20	Caixa de texto que altera sempre que o usuário muda qual a configuração ele está usando.

Fonte: elaborado pelo autor.

Nessa sala há dois tipos de luzes, o Sol e algumas outras luzes pequenas espalhadas pela cena, o usuário tem a opção de deixar a cena de noite ou apagar as luzes pequenas, podendo assim ver como as luzes e o Sol afetam a cena.⁵

Terminar a frase com ponto final.

cena. O usuário

Conforme o usuário apaga ou acende as luzes a descrição que aparece nas caixas de texto muda para refletir estado atual das luzes. A Figura 15 mostra como essa sala fica de noite, porém com as luzes pequenas acesas.

O usuário também pode alterar várias propriedades do *path tracing*, entre elas a quantidade de amostras e a quantidade máximas e mínimas de saltos que os raios podem realizar. Conforme cada botão que o usuário clica a forma que a cena é renderizada é alterada, permitindo assim o usuário ver as diversas formas que o *path tracing* pode ser utilizado em uma cena.

Umas das configurações que o usuário pode realizar é mudar entre iluminação direta, secundária e terciária. Caso o usuário clique na opção de iluminação direta, os raios de luz serão limitados à no máximo 1 salto, dessa forma apenas ficarão iluminados os objetos que estão diretamente na linha de alguma fonte de luz. Outra opção que o usuário tem é ver a iluminação secundária, nesse modo os raios são obrigados a realizar apenas dois saltos, não podendo realizar menos ou mais. Esse tipo de iluminação permite o usuário ver para onde os raios estão sendo refletidos, e o quanto esses raios refletidos interferem na iluminação de áreas que não estão diretamente na linha de alguma fonte de luz. Há também a opção de iluminação terciária, que limita à três a quantidade máxima e mínima de saltos dos raios, com essa opção o usuário pode ver todos os locais que são iluminados por raios que fizeram três saltos, a partir dessa ponte a iluminação fica bem mais fraca. Para cada uma das opções o texto de explicação muda, para permitir o usuário entender quais as diferenças. Essa quantidade de saltos está representada no fluxo do Apêndice B.

Outra configuração que o usuário pode alterar é a quantidade de amostras que são coletadas para cada pixel da tela. Cada amostra é um raio disparado, por padrão essa sala utiliza 256 amostras por pixel, porém o usuário pode alterar para utilizar 8, 512, 2048 e 4096 amostras. Quando o usuário seleciona 8 amostras, são disparados apenas 8 raios por pixel, fazendo dessa forma que a cena seja renderizada muito mais rapidamente, porém com uma qualidade muito ruim. Para essa opção o usuário é informado sobre as suas características de ser mais rápida, porém mais lenta. O usuário pode selecionar 512 amostras, o que resulta em uma imagem melhor, porém ao custo de um tempo maior para ser concluída. Há também as opções de 2048 amostras e 4096 amostras, essas opções trazem a melhor qualidade no resultado final, porém podem demorar muitos minutos para serem concluídas, mesmo com hardware específico para isso. O usuário é informado nessas duas sobre suas características de serem melhores para renderização de animações do que para aplicações em tempo real. Para essa configuração da quantidade de amostras que o *path tracing* utiliza, há o Apêndice B que é possível ver onde ela se encaixa.

4 RESULTADOS

Durante a fase de implementação foram sendo realizados diversos testes para garantir que o que estava sendo implementado estava correto. Os testes foram divididos em três partes, uma para cada sala implementada, e foram aplicados separadamente conforme a implementação de cada sala foi terminada. O Quadro 5 mostra quais testes foram realizados para a sala Sombras, bem como o resultado esperado e se o teste foi bem-sucedido.

5 Quadro 5 - Testes realizados para validar Sala Sombras

Teste	Esperado	Obtido
Clicar no botão Menu	Voltar ao menu principal	OK
Clicar nos botões Anterior e Próxima	Alterar o ponto de vista que está sendo exibido na tela	OK
Clicar no botão Gráficos	Caso os gráficos de performance não estejam ativos, mostrar eles, caso contrário esconder ele.	NOK
Clicar no botão Denoiser	Caso denoiser esteja ativo, desativar ele, caso contrário ativar ele, alterar a descrição do botão para refletir o estado dele e alterar uma das caixas de texto para descrever o que está ocorrendo com ele	OK
Clicar no botão Tamanho Luz	Caso a fonte de luz esteja pequena, alterar o tamanho para grande, caso contrário, alterar para pequeno. Mudar explicação nas caixas de texto e alterar os gizmos dos raios de luz para refletirem o novo tamanho da luz.	OK
Clicar no botão Sombra RT	Caso as sombras estejam sendo formadas com ray tracing, desativar elas, caso contrário, ativar elas. Alterar caixas de texto para explicar a diferença entre a sombra padrão do Unity com a sombra ray tracing.	OK
Controlar câmera	Permitir controlar a câmera com os botões WASD e com o mouse	NOK

Fonte: elaborado pelo autor.

Utilizando o Quadro 5, é possível identificar dois pontos de falha, em relação ao controle da câmera e outro em relação aos gráficos de performance. O controle da câmera estava sendo feito por um código criado pelo autor, porém ele não estava funcionando corretamente, ao tentar usar um dos botões de controle junto com o mouse ele acabava ignorando um dos inputs. Foi alterado para utilizar um código padrão do Unity, chamado SimpleCameraControl.cs, com esse novo código o controle da câmera funcionou corretamente, mesmo se o usuário utilizar vários inputs ao mesmo tempo.

- 1 disparado, onde por
- 2 Arrumar.
- 3 amostras, onde essas
- 4 duas opções sobre
- 5 A primeira e segunda coluna usar alinhamento a justificado para o texto.
- 6 Terminar frase com ponto final.
- 7 Terminar frase com ponto final.
- 8 Terminar frase com ponto final.
- 9 Quadro 5 é possível
- 10 falha, um em
- 11 SimpleCameraControl.cs. Com

O outro problema encontrado foi com os gráficos de performance, caso eles fossem ativados e depois o usuário voltasse ao menu principal, os gráficos continuavam presentes e não sumiam conforme esperado. Para corrigir o erro, os gráficos foram adicionados dentro de um novo `GameObject` e foi criado um novo trecho de código no momento de voltar ao menu, nesse trecho o novo objeto está sendo escondido, a Figura 16 demonstra o novo trecho e código. Após corrigidos os erros encontrados na sala `Sombras`, foi realizado o desenvolvimento e subsequente teste da sala `Técnicas`, o Quadro 6 mostra os testes e resultados da sala `Técnicas`.

Figura 16 – Trecho de código para carregar Menu.

```
public void loadMenu()
{
    graphy = !graphy;
    graphyGO.SetActive(graphy);
    SceneManager.LoadScene(0);
}
```

Fonte: elaborado pelo autor.

Quadro 6 - Testes realizados para validar Sala Técnicas

Teste	Esperado	Obtido
Clicar no botão Menu	Voltar ao menu principal	OK
Clicar nos botões Anterior e Próxima	Alterar o ponto de vista que está sendo exibido na tela.	OK
Clicar no botão Gráficos	Caso os gráficos de performance não estejam ativos, mostrar eles, caso contrário esconder ele.	OK
Clicar no botão SSR	Caso <i>ray tracing</i> no SSR esteja ativo, desativar ele, caso contrário ativar ele. Alterar a descrição do botão para refletir o estado dele e alterar uma das caixas de texto para descrever o que está ocorrendo com ele.	OK
Clicar no botão Refração	Caso a refração esteja baixa, alterar para ficar alta, caso contrário tornar a refração baixa. Alterar a descrição do botão e alterar a caixa de texto para explicar a diferença entre refração baixa e alta.	NOK
Clicar no botão Metálico/Difuso	Caso os objetos em tela sejam metálicos, alterar eles para ficarem difusos, caso contrário alterar para que fiquem metálicos. Alterar a descrição do botão, alterar as caixas de texto para que façam a explicação pertinente ao tipo de objeto que está sendo exibido e alterar os gizmos de raio de luz que demonstram a interação dos raios de luz com os objetos.	OK
Clicar no botão GI Denoiser	Caso o <i>denoiser</i> da iluminação global esteja ativo, desativar ele, caso contrário, ativar ele. Alterar a descrição do botão e alterar as caixas de texto que explicam sobre a Iluminação Global por <i>ray tracing</i> .	OK
Controlar câmera	Permitir controlar a câmera com os botões WASD e com o mouse.	OK

Fonte: Elaborado pelo autor.

Com a execução dos testes foi possível perceber que apenas havia um problema com a refração de objetos. Como o índice de refração é definido por objeto e não algo global, não estava sendo possível alterar a refração do objeto em questão em tempo de execução, não acontecia nada, possivelmente devido à algum problema com a versão do Unity. Então foi alterado para que existam objetos com índice de refração diferentes, e sempre apenas um deles está visível de cada vez, alterando qual está visível com base no que o usuário escolheu.

Não houve mais problemas com a sala pois algumas das correções que foram aplicadas na sala anterior já foram aplicadas à essa sala já durante o desenvolvimento, evitando assim a geração de outros erros. Após a correção do erro encontrado, foi realizado um novo teste para validar as correções e não foi encontrado mais nenhum erro. Foi realizado então o teste da terceira sala, `PathTracing`, o Quadro 7 mostra os testes e os resultados obtidos.

Quadro 7 - Testes realizados para validar Sala PathTracing

Teste	Esperado	Obtido
Clicar no botão Menu	Voltar ao menu principal	OK
Clicar nos botões Anterior e Próxima	Alterar o ponto de vista que está sendo exibido na tela.	OK
Clicar no botão Gráficos	Caso os gráficos de performance não estejam ativos, mostrar eles, caso contrário esconder ele.	OK
Clicar no botão Dia	Trocar entre dia e noite na cena e alterar alguns textos de explicações.	OK

- 1 não eram desabilitados conforme
- 2 menu. Nesse
- 3 sendo desabilitado, conforme pode ser observado na Figura 16.
- 4 Sombras foi
- 5 Técnicas. O Quadro
- 6 Remover o ponto final.
- 7 Terminar a frase com ponto final.
- 8 A primeira e segunda coluna usar alinhamento a justificado para o texto.
- 9 Terminar a frase com ponto final.
- 10 Terminar a frase com ponto final.
- 11 execução. Neste caso não
- 12 terceira sala PathTracing (Quadro 7).
- 13 Terminar a frase com ponto final.

Clicar no botão Luzes	Caso as luzes espelhadas pela cena estejam ligadas, desligar elas, caso contrário ligar elas. Alterar a descrição de um dos textos em tela para refletir o estado atual das luzes.	OK
Clicar nos botões + e = 1	Alterar a quantidade de saltos que os raios podem realizar. Alterar a label entre os botões para mostrar quantos saltos os raios estão realizando atualmente.	OK
Clicar no botão Default	Alterar a configuração do ray tracing para ficar com os valores padrões.	OK
Clicar no botão Iluminação Direta	Alterar a configuração do ray tracing para utilizar apenas iluminação direta. Alterar as caixas de texto para explicar sobre iluminação direta.	OK
Clicar no botão Iluminação Secundária	Alterar a configuração do ray tracing para utilizar apenas iluminação secundária. Alterar as caixas de texto para explicar sobre iluminação secundária.	OK
Clicar no botão Iluminação Terciária	Alterar a configuração do ray tracing para utilizar apenas iluminação terciária. Alterar as caixas de texto para explicar sobre iluminação terciária.	OK
Clicar nos botões de amostras	Alterar a quantidade de amostras que o ray tracing está utilizando conforme o botão clicado. Alterar as caixas de texto para refletir qual opção foi utilizada.	OK
Controlar câmera	Permitir controlar a câmera com os botões WASD e com o mouse 2	OK

Fonte: elaborado pelo autor.

Após a realização dos testes, 3 foi identificado que todas as ações estavam funcionando corretamente. Cada um dos botões estava realizando suas ações conforme esperado, porém foi percebido que caso fossem desligadas as luzes e trocado para ficar de noite, não era renderizado nada por falta de alguma fonte de luz, então 4 foi adicionado um texto quando está de noite e as luzes estão desligadas explicando que não é possível ver nada devido ao fato de que não há nenhuma fonte de luz na cena. A Figura 17 demonstra o trecho de código que foi adicionado para mostrar esse texto.

Figura 17 - Código para mostrar texto

```
public void toggleAllLights()
{
    bAllLights = !bAllLights;
    AllLights.SetActive(bAllLights);

    LightsInstructions.SetActive(bAllLights || bdayNight);
    SunInstructions.SetActive(bAllLights || bdayNight);

    if (!bAllLights && !bdayNight)
    {
        allLightsInstructions.text = "De noite e sem outras luzes, não é possível ver " +
            "nada na cena pela falta de alguma fonte de luz.";
        return;
    }

    if (bAllLights)
    {
        AllLightsTextText.text = "Luzes: Ligadas";
        allLightsInstructions.text = "Com as luzes ligadas, podemos ver como elas geram ou
    }
    else
    {
        AllLightsTextText.text = "Luzes: Desligadas";
        allLightsInstructions.text = "Com as luzes desligadas, apenas o Sol está iluminand
    }
}
```

Fonte: elaborado pelo autor.

5 CONCLUSÕES

O uso de ray tracing em aplicações de tempo real ainda 5 está em sua infância, com desempenho limitado mesmo com hardware feito para esse uso. O suporte de ray tracing no software também ainda 6 está sendo melhorado, no início desse projeto o Unity para ray tracing ainda estava em versão alpha, e no momento de encerrar o projeto ainda se encontra na versão beta. Durante o desenvolvimento houve diversos problemas de instabilidade do software, com o Unity dando crash com frequência e até certo ponto parando de funcionar completamente, resultando em precisar de uma reinstalação do Windows para que ele voltasse a funcionar. Com 7 o Unity saindo do alpha para o beta a estabilidade foi aumentando gradualmente até ela chegar no ponto atual que está aceitável, com a próximas versões e com ele saindo do beta ficará muito mais fácil utilizar o ray tracing no Unity.

São os botões + e = ou + e -?

Terminar a frase com ponto final.

testes foi

luz. Então

Melhorar ... não usa palavras do tipo "infância" em textos técnicos.

melhorado. No

aceitável. Acredasse que com

A maioria dos objetivos para a implementação foram cumpridos como especificados, porém as cenas descritas nos objetivos a, b e c foram modificadas de modo a ficarem com melhores exemplos de *ray tracing*, sendo que a cena descrita no objetivo c foi completamente refeita utilizando-se de assets complexos e de alta qualidade para que fosse mostrado um exemplo mais realista. O programa se mostrou como uma boa forma de exemplificar o *ray tracing* à alunos que estão iniciando seu aprendizado.

Embora o programa tenha cumprido os seus objetivos, é possível aprimorar sua eficiência no ensino do *ray tracing*. As possíveis extensões encontradas para o aprimoramento do programa são:

- a) permitir ao usuário alterar e adicionar objetos nas cenas;
- b) permitir o uso do programa em computadores mais fracos;
- c) adicionar explicações mais a fundo, com fórmulas;
- d) melhorar o controle da câmera;
- e) adicionar as explicações nos objetos ao invés de diretamente na tela, dessa forma apresentar as explicações conforme o usuário se aproxima dos objetos.

REFERÊNCIAS

BATALI, Dana et al. **Ray Tracing for the Movie ‘Cars’**. 2006. Disponível em: <<https://graphics.pixar.com/library/RayTracingCars/paper.pdf>>. Acesso em 06 abr. 2019.

CALISTO, Vincent S. **Towards real-time ray tracing in video games**. 2018. Disponível em: <<https://vincentcalisto.com/wp-content/uploads/2018/08/Towards-real-time-ray-tracing-in-video-games.pdf>>. Acesso em: 07 abr. 2019.

KHRONOS. **WebGL 2.0 Specification**. 2019. Disponível em: <<https://www.khronos.org/registry/webgl/specs/latest/2.0/>>. Acesso em: 24 mar. 2019.

HANSEN, Charles et al. **Interactive Ray Tracing**. 1999. Disponível em: <<https://www.ppsloan.org/publications/rtrt99.pdf>>. Acesso em: 06 abr. 2019.

HARBS, Marcos. **Motor para Jogos 2D Utilizando HTML5**. 2013. 78 f. Trabalho de Conclusão de curso (Bacharelado em Ciência da Computação) – Centro de Ciências e Exatas Naturas, Universidade Regional de Blumenau, Blumenau.

KAJIYA, James T. **The Rendering Equation**. 1986. 9 f. California Institute of Technology. Disponível em: <http://www.cse.chalmers.se/edu/year/2016/course/TDA361/rend_eq.pdf>. Acesso em: 20 jun. 2020.

KOEHLER, William F. **VISEDU-CG 4.0**: visualizador de material educacional. 2015. 89 f. Trabalho de Conclusão de curso (Bacharelado em Ciência da Computação) – Centro de Ciências e Exatas Naturas, Universidade Regional de Blumenau, Blumenau.

LOPES, João M B. **Ray Tracing**. 2000. Disponível em: <<http://disciplinas.ist.utl.pt/leic-cg/textos/livro/Ray%20Tracing.pdf>>. Acesso em: 13 abr. 2019.

NUNES, S. A. **VisEdu-CG 3.0**: Aplicação didática para visualizar material educacional – Módulo de Computação Gráfica. 2014. 89 f. Trabalho de Conclusão de curso (Bacharelado em Ciência da Computação) – Centro de Ciências e Exatas Naturas, Universidade Regional de Blumenau, Blumenau.

NVIDIA. **NVIDIA RTX™ platform**. 2018. Disponível em: <<https://developer.nvidia.com/rtx>>. Acesso em: 13 abr. 2019.

PACHECO, Hugo. **Ray Tracing in Industry**: An up-to-date review of industrial ray tracing applications and academic contributions. 2008. Disponível em: <<https://paginas.fe.up.pt/~aas/pub/Aulas/DiCG/HugoPacheco.pdf>>. Acesso em: 10 abr. 2019.

PERSISTENCE OF VISION RAYTRACER PTY. **POV-Ray**. 2013. Disponível em: <<http://www.povray.org/>>. Acesso em: 24 mar. 2019.

PETERNIER, Achille; THALMANN, Daniel; VEXO, Frederic. **Mental Vision**: A Computer Graphics Teaching Platform. 2006. Disponível em:

APÊNDICE A – REQUISITOS DA APLICAÇÃO

Neste apêndice serão exibidos os principais requisitos da aplicação utilizados no desenvolvimento. Os requisitos estabelecidos foram:

- a) possuir uma cena com um quarto branco, com dois cubos dentro e um holofote como fonte de luz apontando para um dos cubos (Requisito Funcional - RF);
- b) possuir uma cena com um chão branco e uma parede branca ao fundo, um objeto cúbico e outro objeto esférico, um holofote apontando para a esfera e uma luz dispersa logo acima dos objetos (RF);
- c) possuir uma cena com dois quartos ligados por uma porta, com uma fonte de luz dispersa num dos quartos, com dois objetos esféricos no quarto com a luz, e um objeto cúbico no quarto sem a luz (RF);
- d) permitir o usuário alterar a textura e a cor dos objetos (RF);
- e) permitir o usuário ligar e desligar o *ray tracing* (RF);
- f) possuir duas telas de visualização, uma com a visão da câmera e outra em terceira pessoa mostrando a cena como um todo (RF);
- g) criar traços na visualização em terceira pessoa mostrando o caminho dos raios de luz (RF);
- h) criar três tipos de texturas, reflexiva, opaca e transparente (Requisito Não Funcional - RNF);
- i) possuir um menu que direcione para cada uma das cenas (RNF);
- j) adicionar caixas de texto com explicações sobre como o *ray tracing* está sendo utilizado na cena (RNF);
- k) permitir escolher três cores, vermelho, verde e azul (RNF);
- l) utilizar aceleração de GPU quando disponível (RNF).



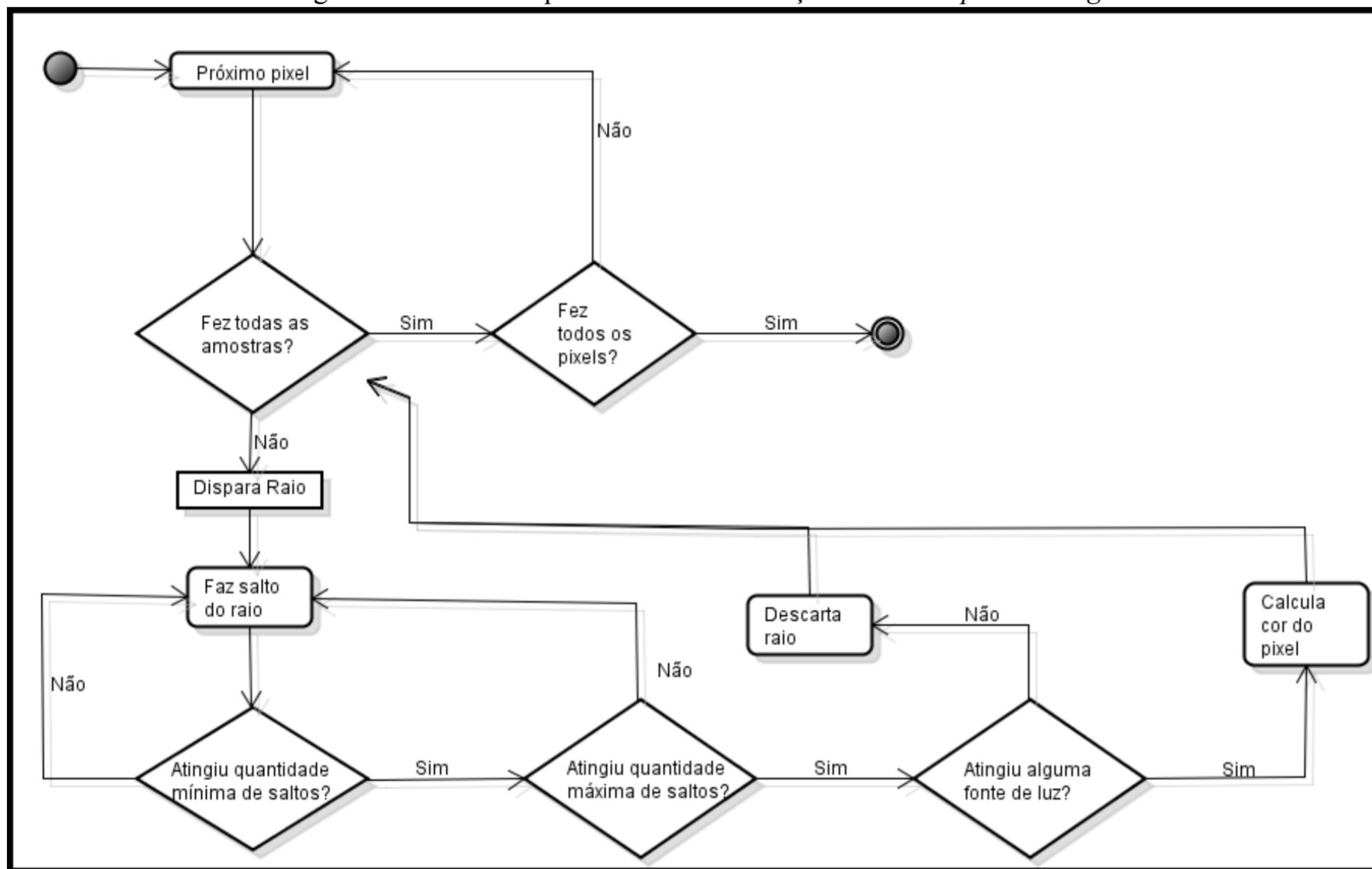
Terminar a frase com ponto final.

APÊNDICE B – FLUXOGRAMA DA RENDERIZAÇÃO COM *PATH TRACING*

Neste **apêndice 1** é exibido um fluxo simplificado de como é feita a renderização de uma cena utilizando o *path tracing*.

Apêndice

1

Figura 18 – Fluxo simplificado da renderização utilizando *path tracing*

Fonte: elaborado pelo autor.