VISEDU LIGHt: visualizador de Ray tracing

Daniel Rossato Martini

Prof. Dalton Solano dos Reis – Orientador

# 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 núcleosespecíficos para calcular *ray tracing* (NVIDIA, 2018). Ainda de acordo com a NVIDIA(2018), esses núcleosde *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, 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.

## OBJETIVOS

O objetivo deste trabalho é disponibilizar um ambiente para visualização e aprendizado de iluminação sobre a tecnologia de *ray tracing*. Os objetivos específicos são:

1. disponibilizar três cenas para simulação de *ray tracing*;
2. apresentar uma explicação de como está ocorrendo o *ray tracing* na cena;
3. permitir alterar textura dos objetos;
4. permitir ligar e desligar o *ray tracing*.

# 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 Peternier, Thalmann e Vexo (2006) o *Mental Vision*, uma plataforma para ensino de computação gráfica.

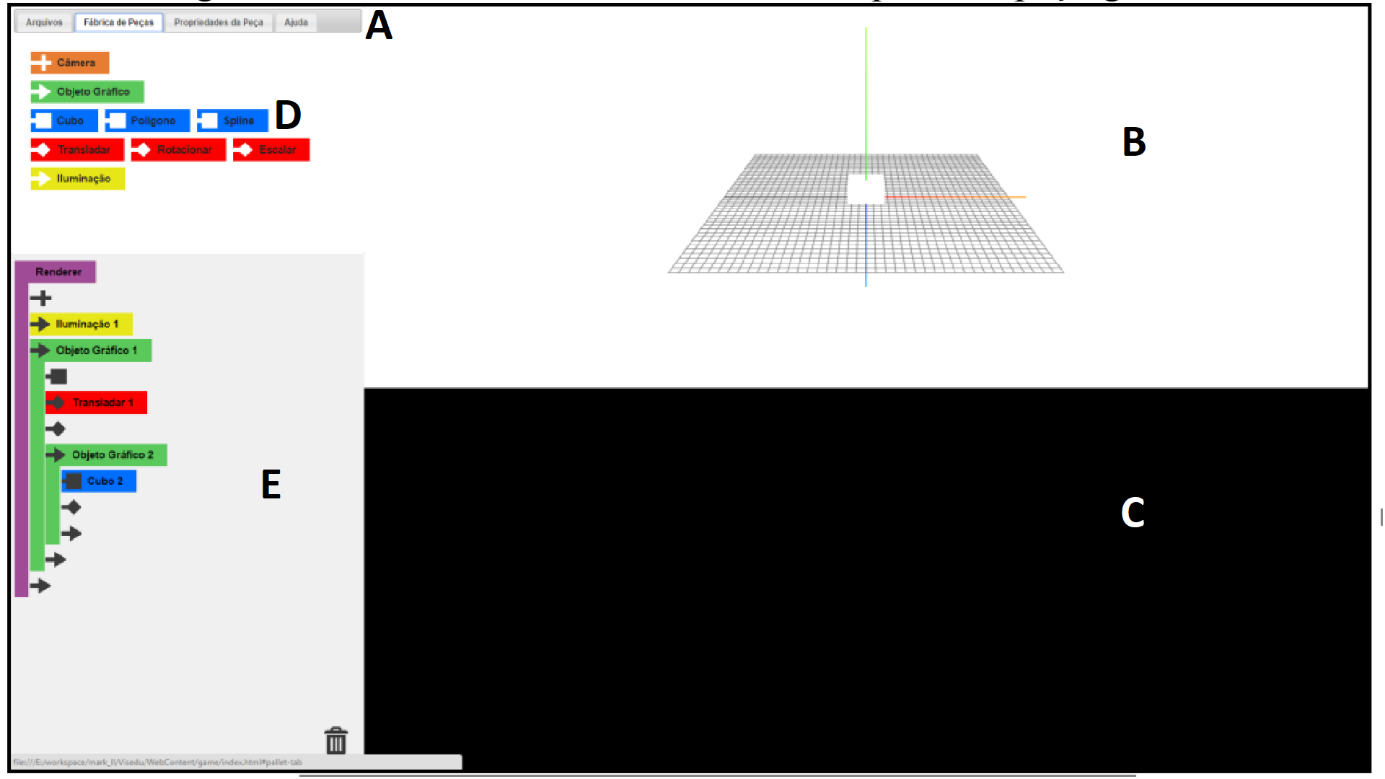
## VISEDU-CG 4.0

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, podem ser criadas cenas, que servem para facilitar a visualização das diversas propriedades existentes.

O programa tem uma tela conforme a Figura 1, no item *A* há 4 abas*.* A primeira é Arquivos, ela serve apenas para importar e exportar cenas criadas. A segunda, 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 todo 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).

Koehler (2015) atingiu seus objetivos conseguindo adaptar o motor gráfico para utilizar o Three.js. Assim a aplicação suporta ambientes tridimensionais com um grande número de objetos gráficos. Mesmo com um grande número de 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.

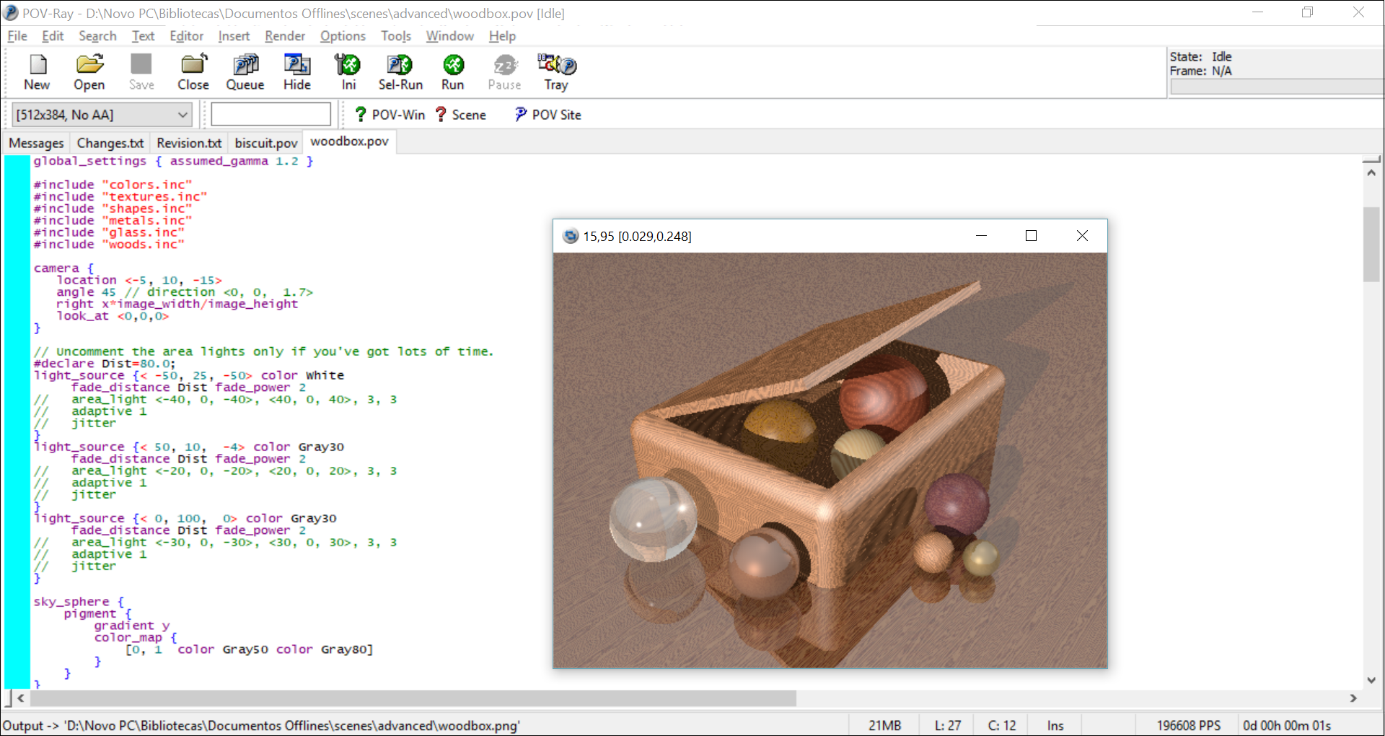
Figura 1 – Visedu-CG 4.0

Fonte: Koehler (2015).

## 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 de código manualmente conforme a Figura 2, 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 2 – 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.

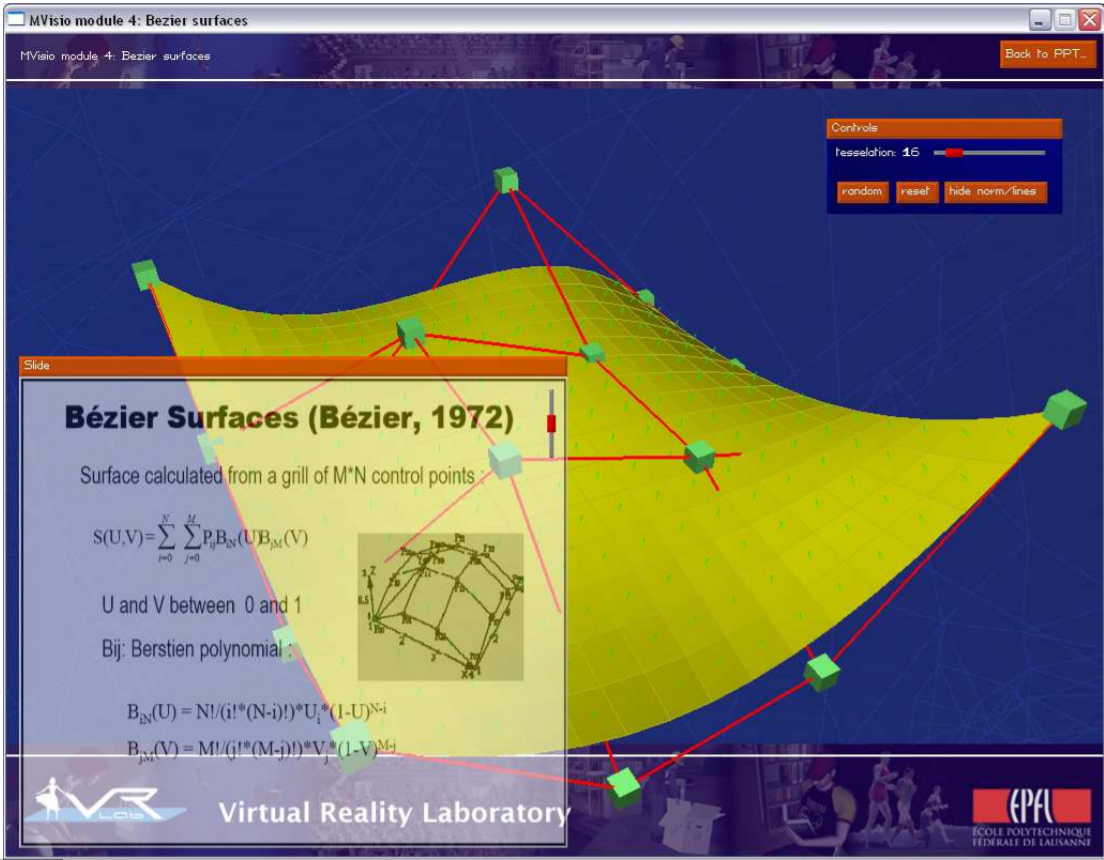
## MVisio

Peternier, 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 3 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 Assistent* (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 3 – MVisio, módulo de Bézier

aaaFonte: Peternier, Thalmann e Vexo (2006).

# proposta Da ferramenta

Neste capítulo é apresentada a justificativa para elaboração do trabalho, bem como os requisitos e metodologia de desenvolvimento.

## JUSTIFICATIVA

O Quadro 1 apresenta um comparativo entre os trabalhos correlatos apresentados, onde as linhas representam as características e as colunas os trabalhos.

Quadro 1 - Comparativo entre os trabalhos correlatos

|  |  |  |  |
| --- | --- | --- | --- |
| características / trabalhos | Visedu 4.0  (KOEHLER, 2015) | POV-Ray (2013) | MVisio  (PETERNIER, THALMANN, VEXO 2006) |
| plataforma | Web | Windows | Diversos |
| tecnologia principal utilizada | WebGL | Própria | OpenGL |
| permite edição de cenas | Sim | Sim | Sim |
| várias fontes de luz | Sim | Sim | Não |
| renderiza *ray tracing* | Não | Sim | Não |
| dá explicações sobre os temas | Não | Não | Sim |
| utiliza GPU | Sim | Não | Não |

Fonte: elaborado pelo autor.

É possível observar que cada um dos trabalhos roda em um ambiente diferente, porém o Visedu 4.0 e o MVisio tem maior flexibilidade de uso, pois o primeiro pode ser rodado a partir de qualquer navegador. Já o segundo foi feito para diversas plataformas, permitindo diversas pessoas pudessem trabalhar com ele. Em relação à tecnologia utilizada, cada um deles utiliza uma diferente, sendo que o POV-Ray utiliza uma própria.

Todos os três permitem que sejam feitas edições no ambiente que se está criando, adicionando objetos, texturas entre outros, sendo o POV-Ray o mais complexo entre eles. Na questão de utilizar várias fontes de luz, ambos o Visedu 4.0 e o POV-Ray permitem que sejam adicionadas várias luzes, porém o MVisio não permite, ele apenas tem uma iluminação global igual para toda a área. Ainda no tema de iluminação, apenas o POV-Ray faz uso do *ray tracing*, criando assim uma iluminação extremamente realista, o Visedu 4.0 faz uso apenas de iluminação simples, sem nada de reflexão.

Apenas o MVisio tem explicações completas dentro do programa, com vários módulos cada um explicando algum tema específico. Já o Visedu 4.0 é uma ferramenta voltada para entender como funcionam algumas funções da computação gráfica, porém não há explicações propriamente ditas na ferramenta.

A utilização da GPU para renderizações é muito importante para agilizar o processo, porém apenas o Visedu 4.0 a utiliza, devido ao WebGL que ele utiliza. O POV-Ray como tem um motor de renderização próprio acaba não utilizando a GPU. Por fim, o MVisio não utiliza a GPU também devido a necessidade de compatibilidade entre várias ferramentas em ambientes diferentes.

Com base nas informações acima, conclui-se que nenhum deles apresenta uma explicação satisfatória sobre como o *ray tracing* afeta uma cena, e os que usam o *ray tracing* não apresentam uma performance satisfatória pelo fato de a GPU não ser utilizada. Assim, propõe-se a criação de uma ferramenta que irá mostrar como o *ray tracing* afeta a renderização de cena, explicando cada situação. Isso será feito utilizando-se de tecnologias novas, como o Unity, para permitir seu uso em diversos dispositivos com facilidade e performance. Também será feito uso dos núcleos *Ray Tracing* (RT) das GPUs da NVIDIA*,* para mostrar a diferença que eles podem fazer numa renderização. Com esse trabalho, espera-se que estudantes possam vir a entender como funciona essa tecnologia que está se popularizando atualmente.

## REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

A aplicação desenvolvida deve:

1. 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);
2. 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);
3. 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);
4. possuir um menu que direcione para cada uma das cenas (RF);
5. permitir alterar a textura e a cor dos objetos (RF);
6. adicionar caixas de texto com explicações sobre como o *ray tracing* está sendo utilizado na cena (RF);
7. possuir duas telas de visualização, uma com a visão da câmera e outra em terceira pessoa mostrando a cena como um todo (RF);
8. criar traços na visualização em terceira pessoa mostrando o caminho dos raios de luz (RF);
9. criar três tipos de texturas, reflexiva, opaca e transparente (Requisito Não Funcional - RNF);
10. permitir escolher três cores, vermelho, verde e azul (RNF);
11. utilizar aceleração de GPU quando disponível (RNF);
12. permitir ligar e desligar o *ray tracing* (RNF).

## METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

1. levantamento bibliográfico: realizar pesquisa sobre *ray tracing*;
2. elicitação de requisitos: reavaliar os requisitos, caso necessário fazer ajustes neles ou adicionar novos;
3. especificação: criar os diagramas de classe e de casos de uso utilizando a Unified Modeling Language (UML) com a ferramenta Astah Community;
4. implementação: implementar o trabalho proposto utilizando Unity, com a linguagem de programação C# (C-*sharp*) no ambiente *Visual Studio*;
5. testes: realizar testes para validar se o *ray tracing* está funcionando de forma correta, se está respeitando a textura e a cor dos objetos, realizar testes com alunos de computação gráfica.

As etapas serão realizadas nos períodos relacionados no Quadro 2.

Quadro 2 - Cronograma

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2019 | | | | | | | | | |
|  | ago. | | set. | | out. | | nov. | | dez. | |
| etapas / quinzenas | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| levantamento bibliográfico |  |  |  |  |  |  |  |  |  |  |
| elicitação de requisitos |  |  |  |  |  |  |  |  |  |  |
| especificação |  |  |  |  |  |  |  |  |  |  |
| implementação |  |  |  |  |  |  |  |  |  |  |
| testes |  |  |  |  |  |  |  |  |  |  |

Fonte: elaborado pelo autor.

# REVISÃO BIBLIOGRÁFICA

Nesta seção é apresentada uma introdução inicial ao tema principal do projeto. O tema apresentado é *ray tracing*, técnica para simular a representação realista da luz em renderizações dentro da área de computação gráfica.

## 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.

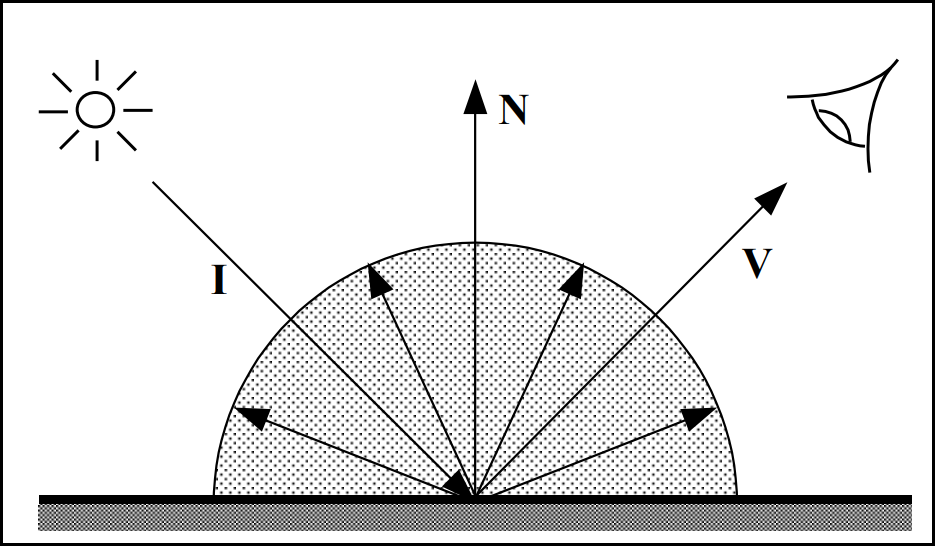
Reflexão especular perfeita ocorre em superfícies polidas e brilhantes, geralmente metálicas, de vidro ou espelhadas. Na realidade não existem superfícies refletivas perfeitas, porém o modelo de reflexão especular perfeita providencia uma aproximação suficientemente parecida com as reflexões que ocorrem em um ambiente real. A Figura 4 demonstra como ocorre a reflexão especular, onde um raio é refletido no mesmo ângulo de entrada, na imagem indicado como Θ (LOPES, 2000).

A close up of a clock

Description automatically generatedFigura 4 – Reflexão especular perfeita

Fonte: Lopes (2000).

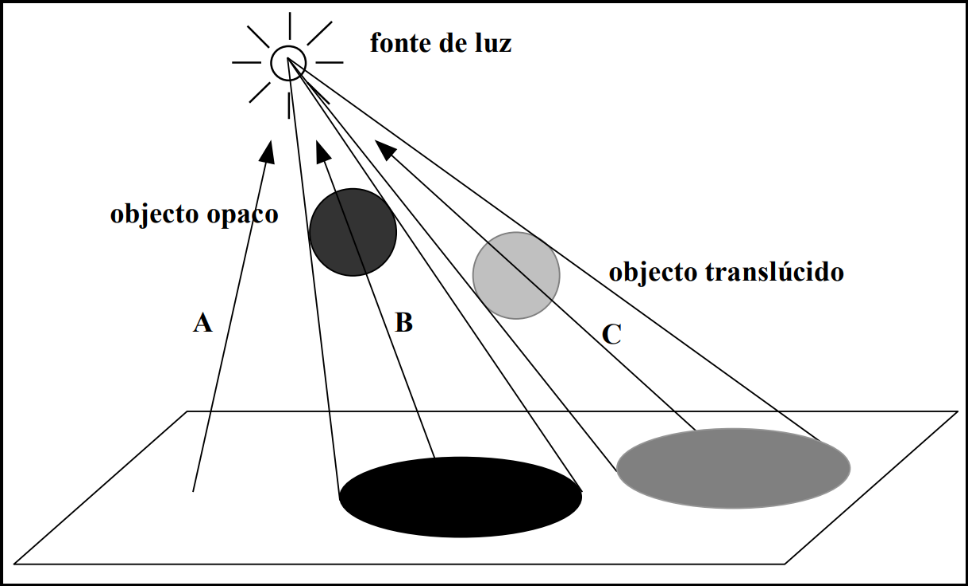
Reflexão difusa perfeita ocorre em superfícies irregulares, nela não existe uma única direção que o raio de luz pode seguir, na verdade a luz se dispersa em igual densidade para todas as direções. A Figura 5 exemplifica como ocorre reflexão difusa perfeita (LOPES, 2000).

Figura 5 – Reflexão difusa perfeita

Fonte: Lopes (2000).

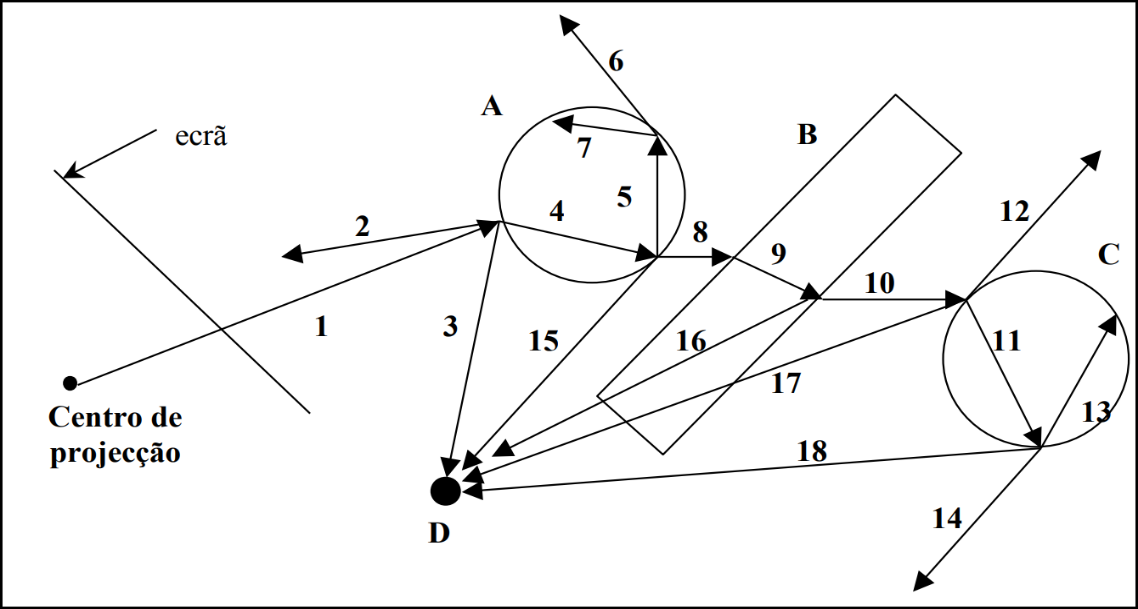
A refração é uma outra propriedade que deve ser levada em consideração no *ray tracing.* Segundo Lopes (2000, p. 19), a luz se propaga por objetos transparentes ou translúcidos, por esse motivo é possível ver através de vidros, mas isso também faz com que os objetos aparentam estar mais próximos do que realmente estão caso esteja dentro da água. Isso é causado pela 10 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 6, temos o raio A, que por não ter nenhuma intersecção, sabemos que é um local iluminado. No raio B, há uma intersecção com a fonte de luz, então não há luz no local abaixo desse objeto, já o raio C intersecciona com um objeto translúcido, então há luz no local, porém reduzida.

Figura 6 – 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 7 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 7 – Algoritmo de *ray tracing*

Fonte: Lopes (2000).

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.

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

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: <https://www.researchgate.net/publication/227186762\_Mental\_Vision\_A\_Computer\_Graphics\_Teaching\_Platform>. Acesso em: 23 mar. 2019.

ASSINATURAS

(Atenção: todas as folhas devem estar rubricadas)

Assinatura do(a) Aluno(a): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Assinatura do(a) Orientador(a): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Assinatura do(a) Coorientador(a) (se houver): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| Observações do orientador em relação a itens não atendidos do pré-projeto (se houver): |

FORMULÁRIO DE avaliação – PROFESSOR TCC I

Acadêmico(a):

Avaliador(a):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ASPECTOS AVALIADOS1 | | atende | atende parcialmente | não atende |
| ASPECTOS TÉCNICOS | 1. INTRODUÇÃO   O tema de pesquisa está devidamente contextualizado/delimitado? |  |  |  |
| O problema está claramente formulado? |  |  |  |
| 1. OBJETIVOS   O objetivo principal está claramente definido e é passível de ser alcançado? |  |  |  |
| Os objetivos específicos são coerentes com o objetivo principal? |  |  |  |
| 1. TRABALHOS CORRELATOS   São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos? |  |  |  |
| 1. JUSTIFICATIVA   Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada? |  |  |  |
| São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta? |  |  |  |
| São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta? |  |  |  |
| 1. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO   Os requisitos funcionais e não funcionais foram claramente descritos? |  |  |  |
| 1. METODOLOGIA   Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC? |  |  |  |
| Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta? |  |  |  |
| 1. REVISÃO BIBLIOGRÁFICA (atenção para a diferença de conteúdo entre projeto e pré-projeto)   Os assuntos apresentados são suficientes e têm relação com o tema do TCC? |  |  |  |
| As referências contemplam adequadamente os assuntos abordados (são indicadas obras atualizadas e as mais importantes da área)? |  |  |  |
| ASPECTOS METODOLÓGICOS | 1. LINGUAGEM USADA (redação)   O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica? |  |  |  |
| A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)? |  |  |  |
| 1. ORGANIZAÇÃO E APRESENTAÇÃO GRÁFICA DO TEXTO   A organização e apresentação dos capítulos, seções, subseções e parágrafos estão de acordo com o modelo estabelecido? |  |  |  |
| 1. ILUSTRAÇÕES (figuras, quadros, tabelas)   As ilustrações são legíveis e obedecem às normas da ABNT? |  |  |  |
| 1. REFERÊNCIAS E CITAÇÕES   As referências obedecem às normas da ABNT? |  |  |  |
| As citações obedecem às normas da ABNT? |  |  |  |
| Todos os documentos citados foram referenciados e vice-versa, isto é, as citações e referências são consistentes? |  |  |  |

PARECER – PROFESSOR DE TCC I ou COORDENADOR DE TCC

**(preencher apenas no projeto):**

|  |  |  |
| --- | --- | --- |
| O projeto de TCC será reprovado se:   * qualquer um dos itens tiver resposta NÃO ATENDE; * pelo menos **4 (quatro)** itens dos **ASPECTOS TÉCNICOS** tiverem resposta ATENDE PARCIALMENTE; ou * pelo menos **4 (quatro)** itens dos **ASPECTOS METODOLÓGICOS** tiverem resposta ATENDE PARCIALMENTE. | | |
| **PARECER**: | ( ) APROVADO | ( ) REPROVADO |

Assinatura: Data:

FORMULÁRIO DE avaliação – PROFESSOR AVALIADOR

Acadêmico(a):

Avaliador(a):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ASPECTOS AVALIADOS1 | | atende | atende parcialmente | não atende |
| ASPECTOS TÉCNICOS | 1. INTRODUÇÃO   O tema de pesquisa está devidamente contextualizado/delimitado? |  |  |  |
| O problema está claramente formulado? |  |  |  |
| 1. OBJETIVOS   O objetivo principal está claramente definido e é passível de ser alcançado? |  |  |  |
| Os objetivos específicos são coerentes com o objetivo principal? |  |  |  |
| 1. TRABALHOS CORRELATOS   São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos? |  |  |  |
| 1. JUSTIFICATIVA   Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada? |  |  |  |
| São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta? |  |  |  |
| São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta? |  |  |  |
| 1. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO   Os requisitos funcionais e não funcionais foram claramente descritos? |  |  |  |
| 1. METODOLOGIA   Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC? |  |  |  |
| Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta? |  |  |  |
| 1. REVISÃO BIBLIOGRÁFICA (atenção para a diferença de conteúdo entre projeto e pré-projeto)   Os assuntos apresentados são suficientes e têm relação com o tema do TCC? |  |  |  |
| As referências contemplam adequadamente os assuntos abordados (são indicadas obras atualizadas e as mais importantes da área)? |  |  |  |
| ASPECTOS METODOLÓGICOS | 1. LINGUAGEM USADA (redação)   O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica? |  |  |  |
| A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)? |  |  |  |

PARECER – PROFESSOR AVALIADOR:

**(preencher apenas no projeto)**

|  |  |  |
| --- | --- | --- |
| O projeto de TCC ser deverá ser revisado, isto é, necessita de complementação, se:   * qualquer um dos itens tiver resposta NÃO ATENDE; * pelo menos **5 (cinco)** tiverem resposta ATENDE PARCIALMENTE. | | |
| **PARECER**: | ( ) APROVADO | ( ) REPROVADO |

Assinatura: Data: