

# VISEDU LIGHT: visualizador de ray tracing

Aluno(a): Daniel Rossato Martini

Orientador: Dalton Solano dos Reis

# Roteiro

- Introdução
- Objetivos
- Fundamentação Teórica
- Trabalhos Correlatos
- Requisitos
- Especificação
- Implementação
- Resultados e Discussões
- Conclusões e Sugestões
- Demonstração

# Introdução

- Apenas usado em renderizações *offline*.
- Poucos explorado para aplicações em tempo real;
- Nova ênfase em *ray tracing* em jogos;

# Objetivos

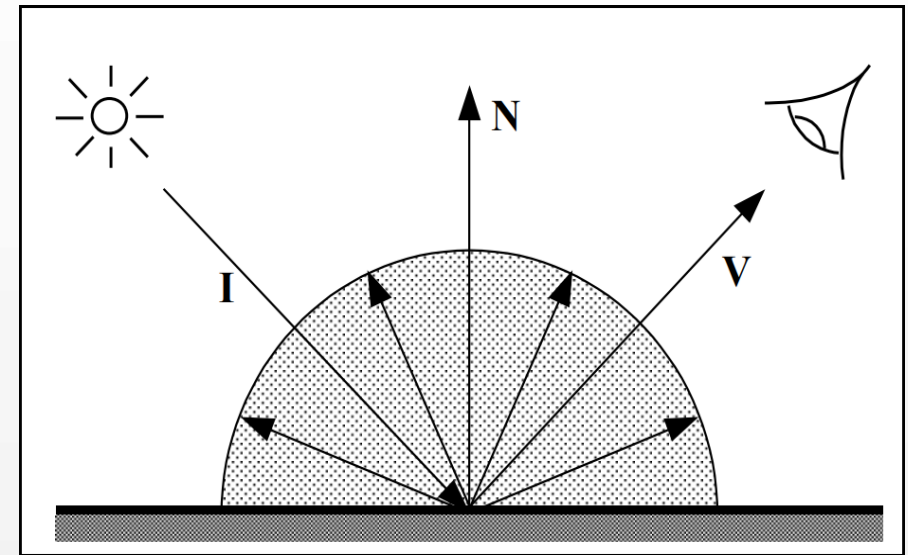
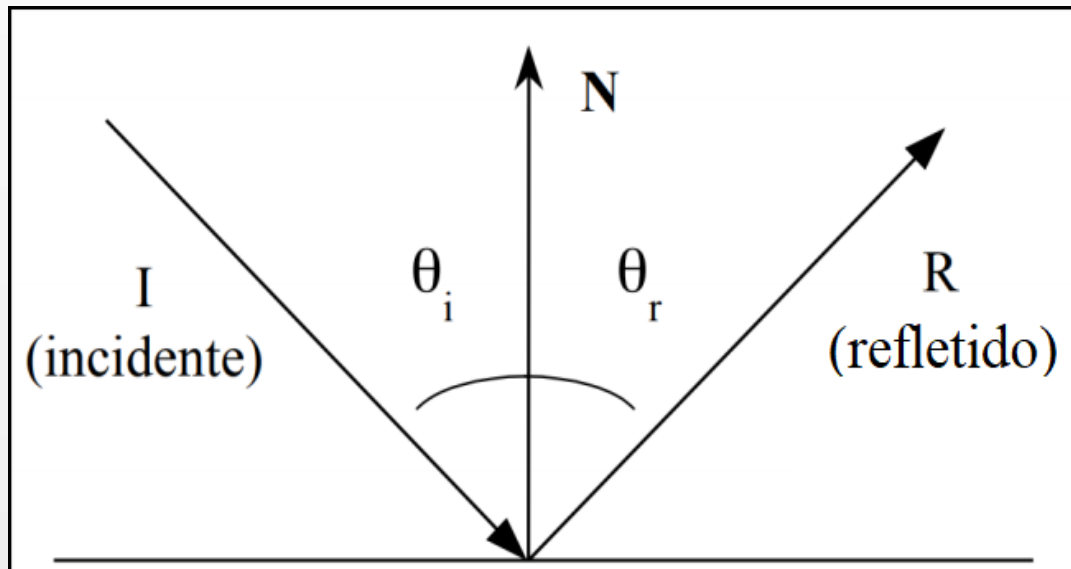
Desenvolver uma ferramenta para a visualização do *ray tracing*, com explicações das técnicas utilizadas.

# Objetivos Específicos

- Permitir o usuário alterar variáveis de ambiente;
- Criar três salas com diferentes técnicas em cada;
- Explicar cada etapa que o usuário passa;

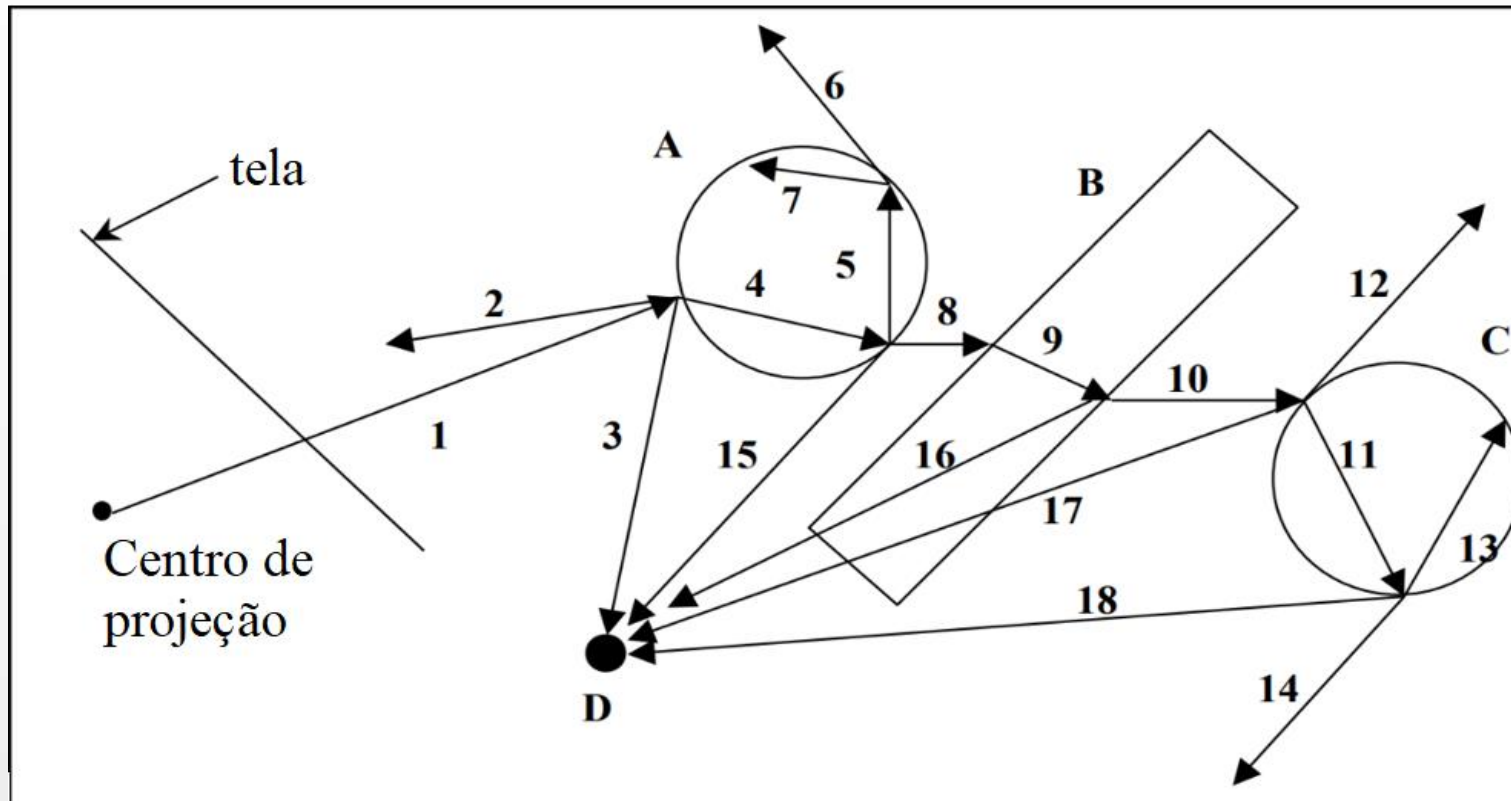
# Fundamentação Teórica – *Ray Tracing*

- Reflexão Perfeita;
- Reflexão Difusa;
- Refração;
- Sombras.



# Fundamentação Teórica – *Path Tracing*

- Forma única de renderização;



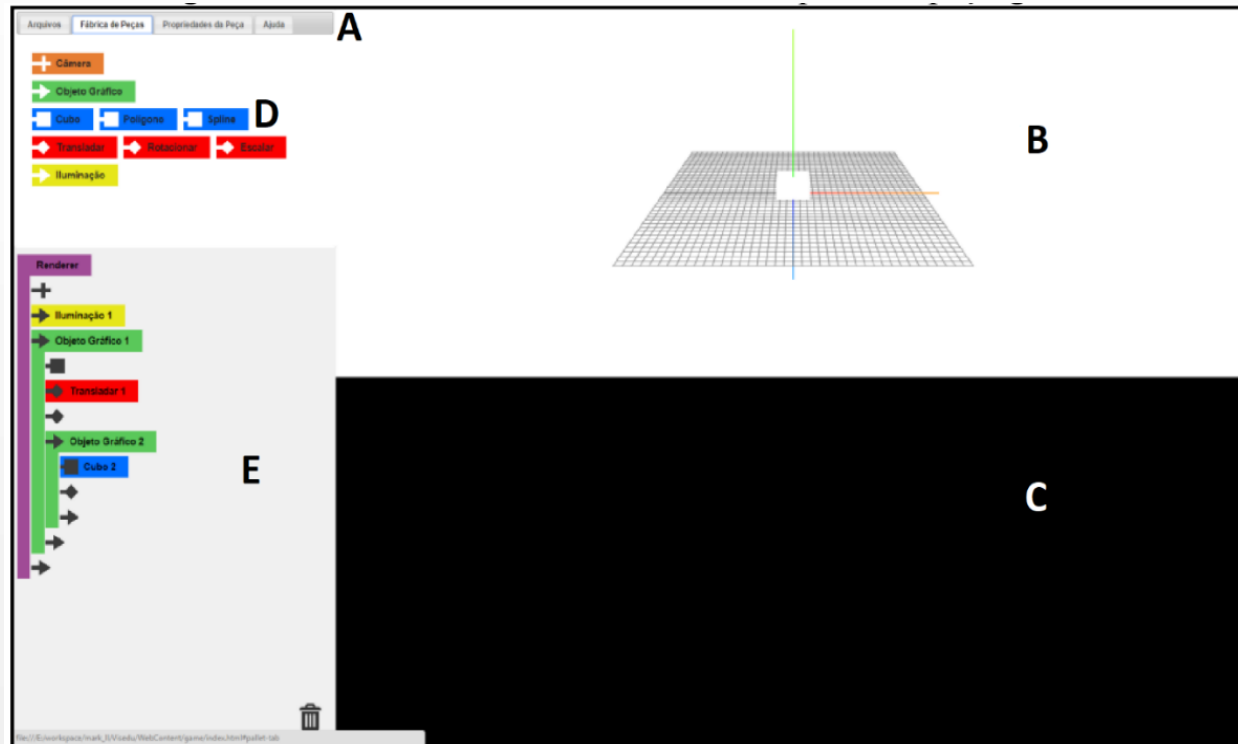
# Fundamentação Teórica – *Denoiser*

- Integral ao *ray tracing*;
- Limpa a imagem gerada;
- Diversos tipos.



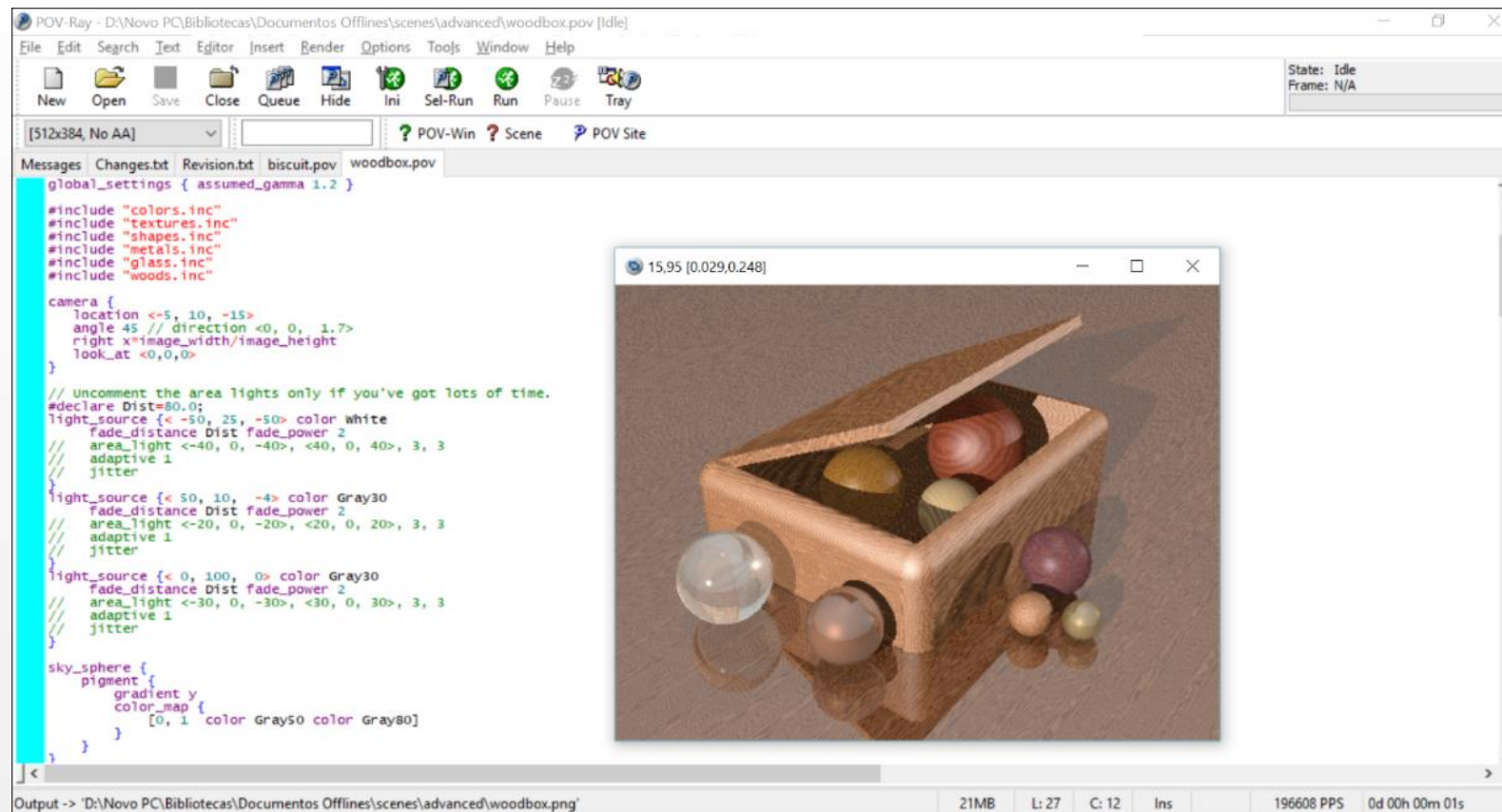
# Trabalhos Correlatos

VISEDU-CG 4.0: visualizador de material educacional



# Trabalhos Correlatos

## POV-Ray



# Trabalhos Correlatos

## Mental Vision

MVisio module 4: Bézier surfaces

MVisio module 4: Bézier surfaces

Back to PPT...

Controls

Resolution: 16

random reset hide norm./lines

Slide

### Bézier Surfaces (Bézier, 1972)

Surface calculated from a grill of  $M \times N$  control points :

$$S(U,V) = \sum_{i=0}^N \sum_{j=0}^M P_{ij} B_{iN}(U) B_{jM}(V)$$

U and V between 0 and 1

Bij: Bernstein polynomial :

$$B_{iN}(U) = \frac{N!}{i!(N-i)!} U^i (1-U)^{N-i}$$
$$B_{jM}(V) = \frac{M!}{j!(M-j)!} V^j (1-V)^{M-j}$$

Virtual Reality Laboratory

EPFL  
ÉCOLE POLYTECHNIQUE  
FÉDÉRALE DE LAUSANNE

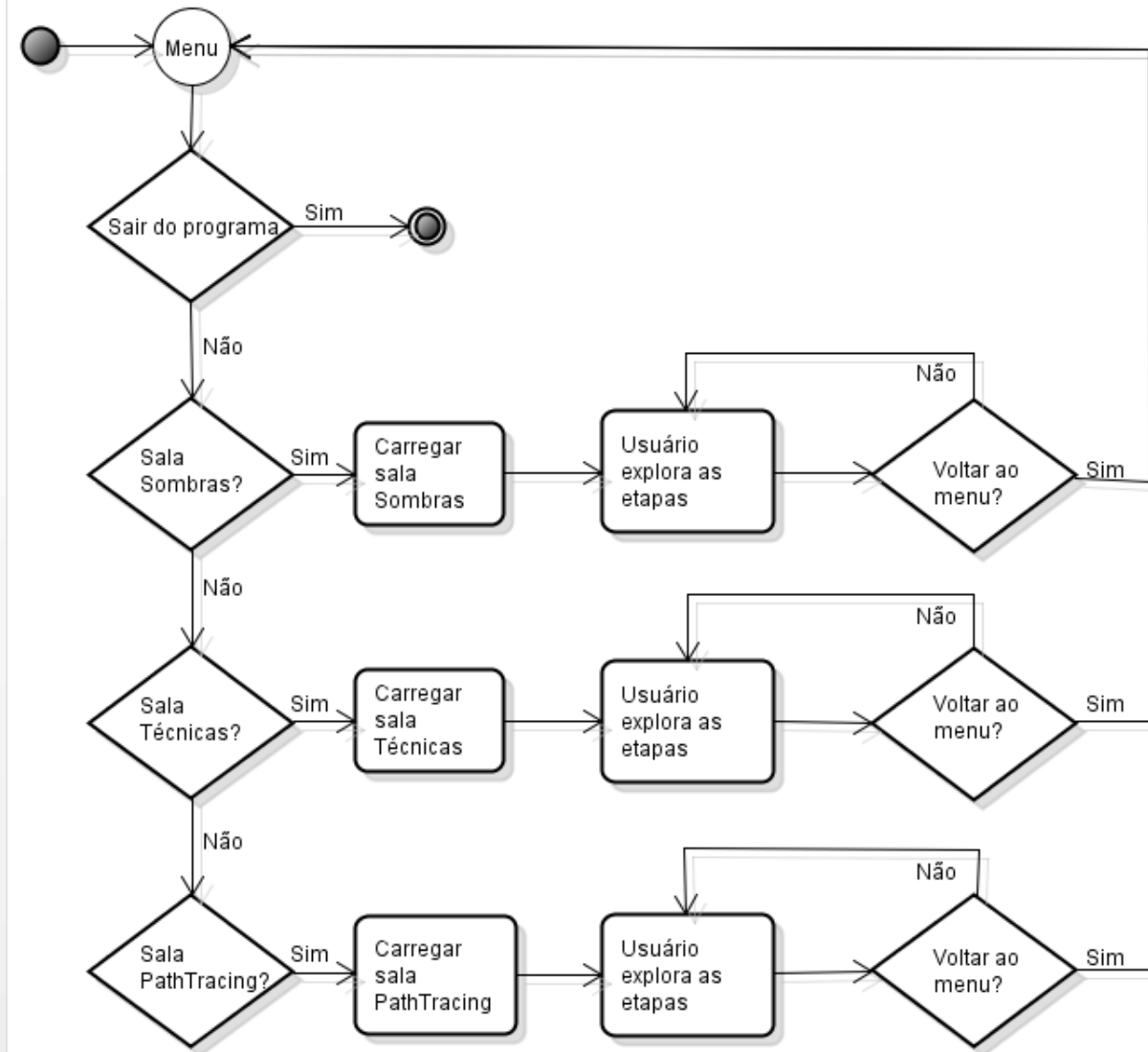
# Requisitos

- Requisitos Funcionais:
  - possuir uma cena com um quarto branco, com dois cubos dentro e um holofote como fonte de luz apontando para um dos cubos;
  - 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;
  - 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;
  - permitir o usuário alterar a textura e a cor dos objetos;
  - permitir o usuário ligar e desligar o *ray tracing*;
  - possuir duas telas de visualização, uma com a visão da câmera e outra em terceira pessoa mostrando a cena como um todo;
  - criar traços na visualização em terceira pessoa mostrando o caminho dos raios de luz;

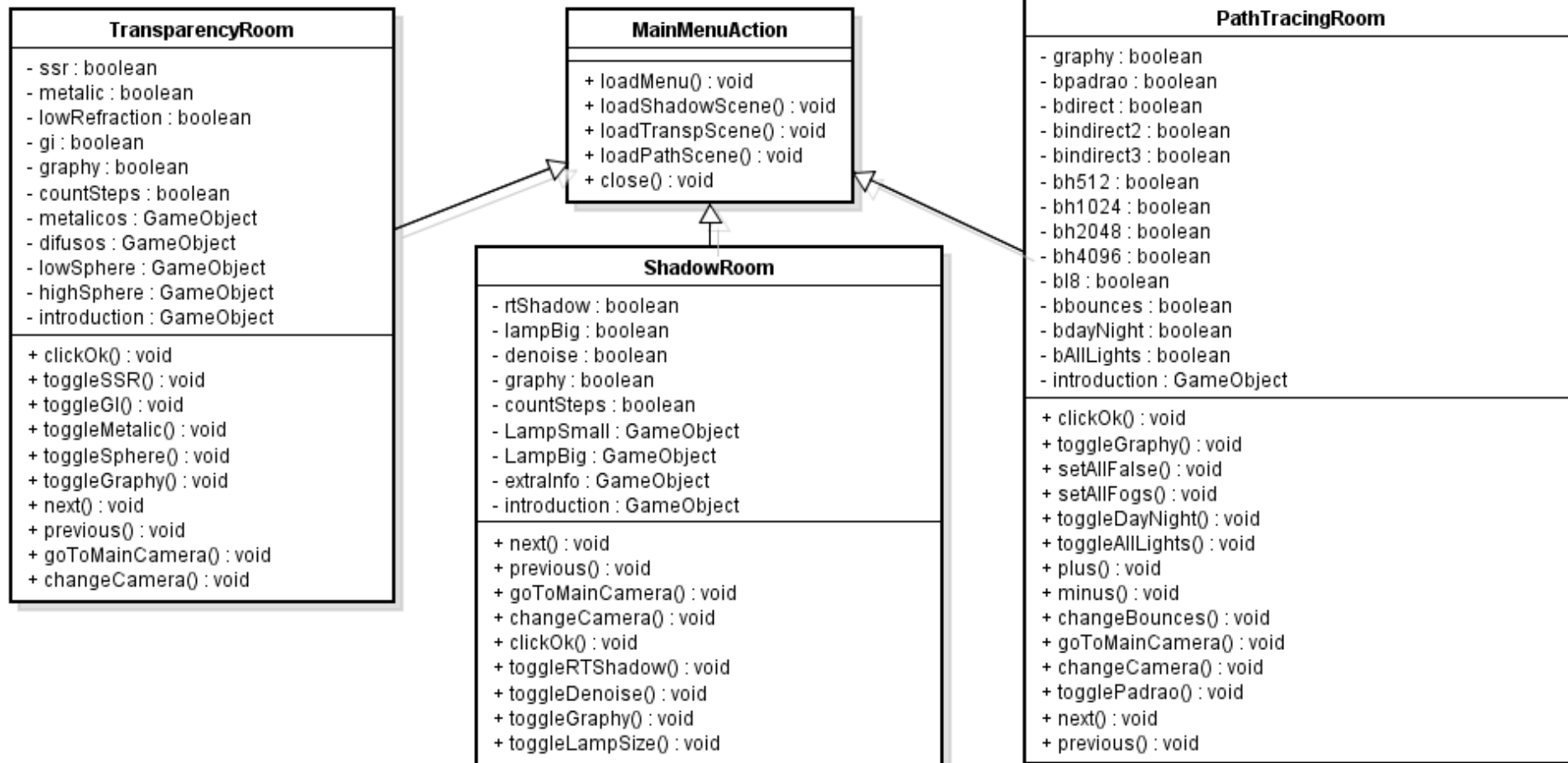
# Requisitos

- Requisitos Não Funcionais:
  - criar três tipos de texturas, reflexiva, opaca e transparente;
  - possuir um menu que direcione para cada uma das cenas;
  - adicionar caixas de texto com explicações sobre como o *ray tracing* está sendo utilizado na cena;
  - permitir escolher três cores, vermelho, verde e azul;
  - utilizar aceleração de GPU quando disponível.

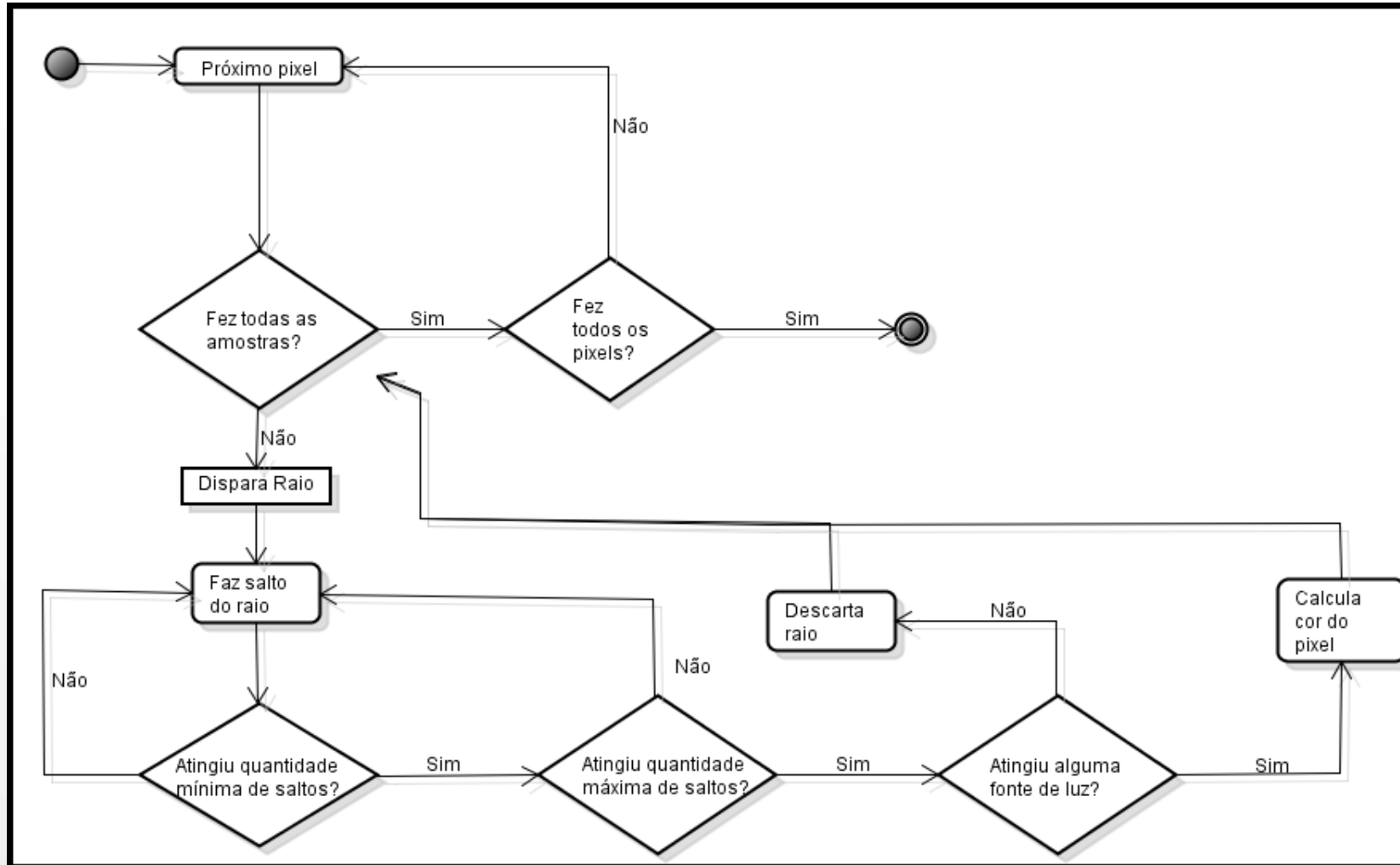
# Especificação



# Especificação



# Especificação



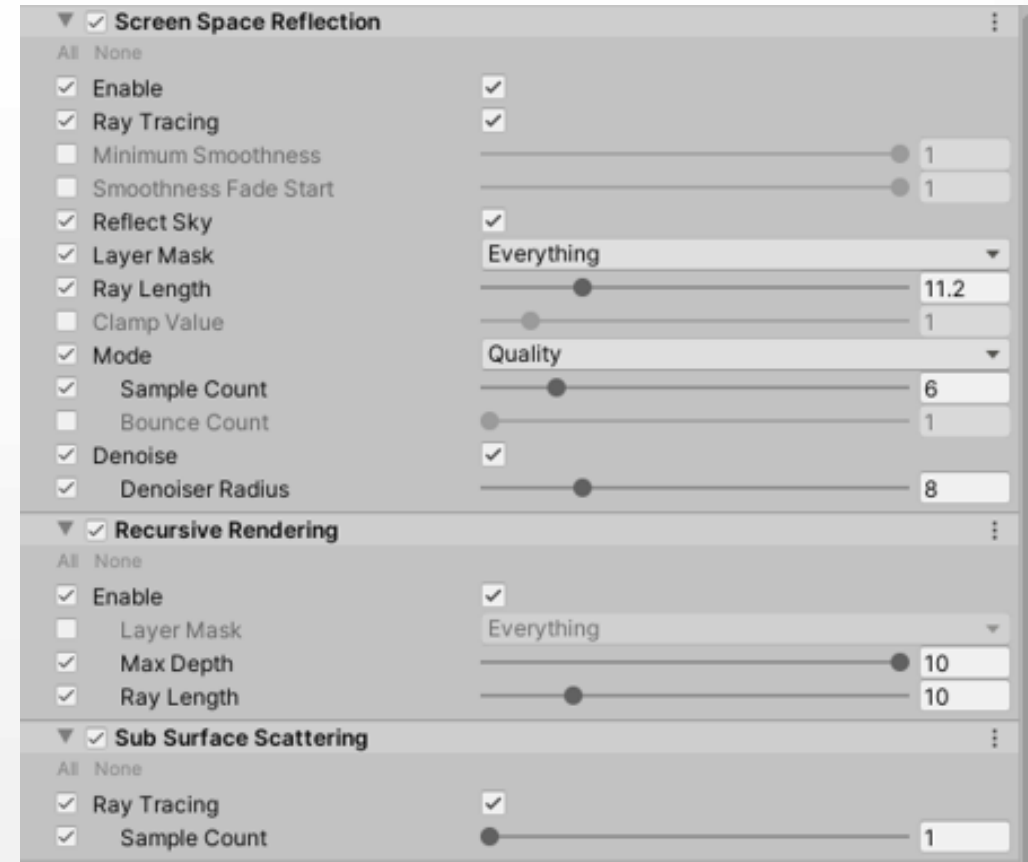


# Implementação

- Assets
  - MS Sports Car;
  - Snaps Prototype | Asian Residencies;
  - Snaps Prototype | Construction Site;
  - Snaps Art HD | Construction Site;
  - Asset Swap Tool.

# Implementação

- Unity;
- Projeto HDPR;
- 4 cenas;
- 4 scripts;



# Implementação

**11** Clique nos botões no canto direito para ver informações

**12**

**13**

**14**

**15** Legenda:  
Verde claro: são os raios de luz primários;  
Verde escuro: são os raios de luz refletidos;  
Vermelho: são a trajetória que o raio seguiria se não fosse bloqueado.

**16** A sombra é formada por vários pontinhos, cada um deles é um lugar onde o raio de luz não chegou. Com o 'denoiser' desligado, conseguimos ver o Ray Tracing em ação.

**Ponto de visão da câmera.**

Os raios são criados a partir da câmera para cada pixel e são direcionados em frente até atingir algo. Ao atingir um objeto, são criados outros raios secundários, que são refletidos em direção às fontes de luz para verificar se o objeto está iluminado.

**Menu**

**9**

**10**

**1**

**1**

**2** Mostrar gráficos de performance Gráficos

**3** Controla a câmera Controlar Camera

**4** Muda para a câmera anterior Anterior

**5** Muda para a próxima câmera Próximo

**6** Habilita e desabilita o 'denoiser' Denoiser: Desligado

**7** Altera o tamanho da fonte de luz Tamanho Luz: Pequena

**8** Habilita e desabilita sombras com Ray Tracing Sombra RT: Ligada

# Resultados e Discussões

- Testes realizados
  - Sala Sombras;
  - Sala Técnicas;
  - Sala PathTracing;
- Correções efetuadas
  - Controle da câmera;
  - Gráficos de performance;
  - Refração;

# Conclusões e Sugestões

- É possível utilizar *ray tracing* com Unity;
- Performance insatisfatória;
- Introduz o *ray tracing* à estudantes;

# Conclusões e Sugestões

- Alterar e adicionar objetos nas cenas;
- Permitir uso em computadores com menor processamento;
- Explicações mais complexas;
- Adicionar as explicações aos objetos;

# Demonstração