



# Compilador de Funções de Distribuição de Refletância Bidirecional descritas em L<sup>A</sup>T<sub>E</sub>X para Linguagem de Shading

Everton Santos de Andrade Júnior  
Orientador(a): Dra. Beatriz Trinchão Andrade

Universidade Federal de Sergipe

Dez / 2024

# Sumário

① Introdução

② Conceitos

③ Metodologia

④ Desenvolvimento

⑤ Resultados

⑥ Conclusão

⑦ Extra

## Contexto

- Na **computação gráfica**, a representação realista de cenas tridimensionais depende da modelagem da interação entre a luz e os materiais que compõem os objetos.
- Profissionais e pesquisadores modelam essa interação por meio das **funções de distribuição de refletância bidirecional** (BRDFs)<sup>1</sup>.
- BRDFs são implementadas em programas especializados que rodam na GPU.

<sup>1</sup> Do inglês, *Bidirectional Reflectance Distribution Functions*. Referência: [PJH16]







## Motivação

Na renderização, as BRDFs são implementadas por meio de programas chamados de **shaders**. Mas, as BRDFs são comumente descritas por equações em  $\text{\LaTeX}^2$ . Dois problemas surgem com essa modelagem:

- ① Barreira técnica<sup>3</sup>: a visualização da modelagem de BRDFs exige um conhecimento especializado em programação na linguagem de *shading*.
- ② Baixo nível de iteração: qualquer alteração nas equações requer uma reescrita do *shader*, o que implica um trabalho adicional e a necessidade de baixar o nível de abstração.

---

<sup>2</sup>  $\text{\LaTeX}$  é um sistema de preparação de documentos para alta qualidade tipográfica. Assim como visto no início.

<sup>3</sup> Físicos ou matemáticos podem trabalhar com BRDFs sem necessariamente dominar programação de baixo nível.

## Motivação

Surge a necessidade de **simplificar** a **criação** de shaders para BRDFs.

Um **compilador** capaz de traduzir BRDFs escritas em  $\text{\LaTeX}$  para *shaders* permitiria uma maior **acessibilidade** e **agilidade** na criação de efeitos visuais complexos.

$$\rho_d = 0, \vec{1}, 1$$

$$\rho_s = 1, \vec{0}, 1$$

$$n = +2^8$$

$$f = \frac{\rho_d}{\pi} + \rho_s * \frac{n+2}{2*\pi} * \cos \theta_h^n$$



# Objetivo

Projetar e implementar um **compilador** capaz de:

- ① Processar **BRDFs** descritas em equações **TEX**.
- ② Gerar **código de shading** na linguagem-alvo da API **OpenGL**.
- ③ Visualizar o efeito dessas BRDFs usando o **código GLSL<sup>4</sup>** gerado.

Resultado esperado é um **shader** que reproduza as características de refletância da BRDF original.

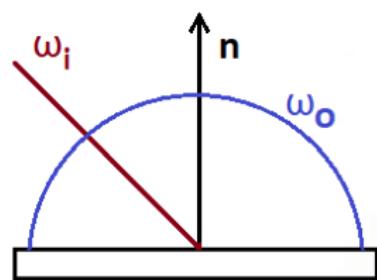
---

<sup>4</sup> GLSL é a linguagem de *shading* compatível com OpenGL.

## Conceitos - BRDFs

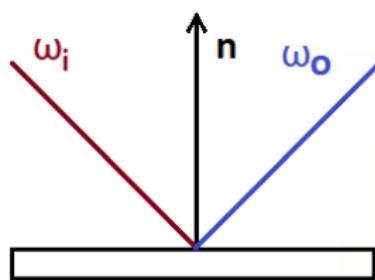
As BRDFs calculam a proporção entre a energia luminosa que atinge um ponto na superfície e como essa energia é:

- **Refletida**
- **Transmitida**
- **Absorvida**



$$f(\omega_i, \omega_o) = \frac{\rho_d}{\pi} \cdot \cos \theta$$

**BRDF Difusa Ideal**



$$f(\omega_i, \omega_o) = k_s \cdot \delta(\omega_i - \omega_o)$$

**BRDF Pura Especular**

# Equação de Renderização

Um renderizador estima a “quantidade” luz  $L_o$  que sai de um ponto em uma direção  $\omega_o$  usando a equação de renderização ([Kaj86]). Nela, a BRDF é encontrada:

Figure: [Mon21]

$$L_o(p, \omega_o) = L_e(p, \omega_o) + \int_{H^2} f(p, \omega_i, \omega_o) L_i(p, \omega_i) \cos(\theta_i) d\omega_i$$

$L_o$  é radiância de saída

$L_e$  é radiância emitida pela superfície (i.e. fonte de luz)

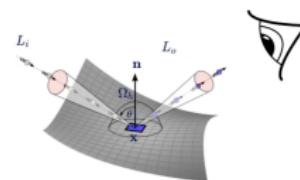
$L_i$  é radiância incidente na superfície

$\omega_i$  é a direção do raio incidente

$\omega_o$  é a direção do raio refletido

$f$  função de refletância

(1)

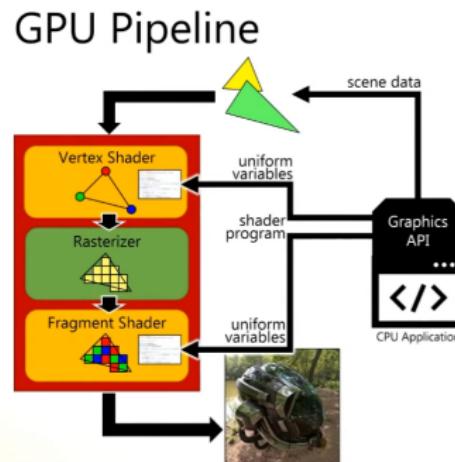


## Pipeline de GPU

Etapas Principais:

- ① **Shader de Vértice**: Processa e transforma vértices.
- ② **Rasterização**: Gera fragmentos a partir de primitivas.
- ③ **Shader de Fragmento**: Determina a cor final dos fragmentos.

Fluxo de dados são: CPU → GPU → *Shaders* → Imagem final.



## Shader de Vértice

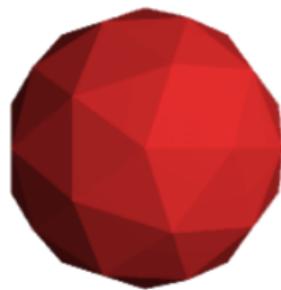
- Aplicação de transformações (ex.: rotação, projeção).
- Transmissão de dados (ex.: normais) ao *shader* de fragmento.

```
1 #version 330 core
2 layout(location = 0) in vec3 inPosition;
3 layout(location = 1) in vec3 inNormal;
4
5 uniform mat4 modelViewProjection;
6
7 out vec3 fragNormal;
8
9 void main() {
10     vec3 manipulatedPosition = inPosition + (sin(gl_VertexID * 0.1) * 0.1);
11     fragNormal = inNormal;
12     gl_Position = modelViewProjection * vec4(manipulatedPosition, 1.0);
13 }
14
```

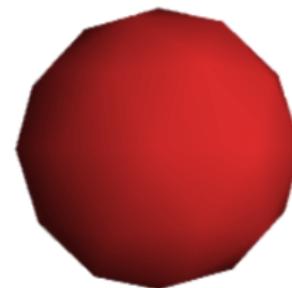
## Shader de Fragmento

Nesse *shader* é onde a cor final é calculada. Roda paralelamente 1 vez para cada "pixel".

**Shading para cada vértice**



**Shading para cada fragmento**



**Figure:** Diferença entre *shading* por vértice e por fragmento.

# Compilação: Visão Geral

Compilador  $C : L_1 \rightarrow L_2$  mapeia programa de  $L_1$  para  $L_2$  preservando semântica<sup>5</sup>.

## ① Análise Léxica (Lexing):

- Divide entrada em tokens (palavras-chave, identificadores)
- Reconhecível por máquinas de estado

## ② Análise Sintática (Parsing):

- Constrói árvore sintática baseado nas regras de produção

## ③ Verificação Semântica:

- Garante consistência de tipos, assinatura de funções, significado dos símbolos, etc.
- No nosso caso, previne erros na modelagem da BRDF

## ④ Emissão de Código:

- Gera código na linguagem alvo

<sup>5</sup> Mantém mesmo significado algorítmico

# Metodologia: Análise e Técnicas

## Estudo da Literatura:

- Nenhum trabalho similar encontrado em bases acadêmicas

**Table:** Resultados das bases após aplicar os critérios.

Bases	Filtrados
IEEE Xplore Digital Library	2
BDTD	0
CAPES Periódico	0
ACM Digital Library	1
Google Acadêmico	1

## Estudo Realizado nas seguintes áreas:

- Radiometria e BRDFs. Base teórica pelo livro "Physically Based Rendering" [PJH16].
- Linguagem de shader GLSL
- Técnicas de análise sintática (Pratt Parsing) e geração de código
- Análise de BRDFs existentes (Artigos)

# Metodologia: Design Inicial de Casos de Teste

Design inicial para validação da tradução BRDF em  $\text{\LaTeX} \rightarrow \text{GLSL}$ <sup>6</sup>

$$\text{cook\_torrance}(\omega_i, \omega_o) = \frac{D(h)F(\omega_i, h)G(\omega_i, \omega_o, h)}{4(\omega_i \cdot n)(\omega_o \cdot n)}$$

## $\text{\LaTeX}$ Fonte

```
\text{vec3 cook\_torrance}(\text{\omega\_i}, \text{\omega\_o})  
= \frac{D(h)F(\text{\omega\_i}, h)G(\text{\omega\_i},  
\text{\omega\_o}, h)}{4(\text{\omega\_i} \cdot n)  
(\text{\omega\_o} \cdot n)}
```

## GLSL a ser gerado

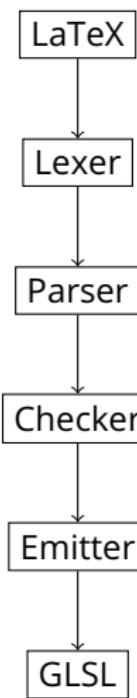
```
vec3 cook_torrance(vec3 wi, vec3 wo) {  
    float D_RESULT = D(h);  
    vec3 F_RESULT = F(wi, wo);  
    float G_RESULT = G(wi, wo, h);  
    float den = 4.0 * dot(n, wi) * dot(n, wo);  
    return D_RESULT * F_RESULT * G_RESULT / den;  
}
```

<sup>6</sup> Alguns símbolos não estão definidos para simplificar

# Metodologia: Implementação do Compilador

- **Linguagem Odin:**
  - Linguagem de propósito geral, orientada a dados
  - Sem dependências externas (bibliotecas padrão)
- **Etapas do Compilador:**

- Lexer e Parser (Pratt Parsing)
- Análise Semântica (checker)
- Travessia AST (walker)
- Geração GLSL (emitter)



# Metodologia: Especificação da Linguagem

Subconjunto do ambiente equation (`\begin{equation}`) essencial para descrever BRDFs

Descrição	Exemplos	Comando $\text{\LaTeX}$
Funções trigonométricas	<code>tan, sin, cos</code> e suas funções inversas	<code>\tan, \sin, \cos\arctan, \arcsin, ...</code>
Função raiz quadrada	$\sqrt{ }$	<code>\sqrt{ }</code>
Função exponencial	<code>exp</code>	<code>\exp</code>
Funções utilitárias	<code>max, min</code>	<code>\max, \min</code>
Definições de equações	$f = x$	<code>f = x</code>
Definições de funções	$f(x, y) = x^y$	<code>f(x, y) = x^y</code>
Constantes comuns	$\pi, \epsilon$	<code>\pi, \epsilon</code>
Constantes de radiometria	$\theta_i$ e outras presentes na 3	<code>\theta_i</code>
Indicadores de vetor	$\vec{n}$	<code>\vec{ }</code>
Identificadores aninhados	$f_{n_j}$	<code>f_{n_i}</code>
Chamadas de funções	$f(x + y)$	<code>f(x + y)</code>
Produto vetorial	$x \times y$	<code>x \times y</code>
Soma e Subtração	$x + y, x - y$	<code>x + y, x - y</code>
Negação	$-y$	<code>-y</code>
Multiplicação	$x \cdot y$	<code>x \cdot y</code>
Frações	$\frac{x}{y}$	<code>\frac{x}{y}</code>
Divisão	$x/y$	<code>x / y</code>
Potenciação	$x^y$	<code>x^y</code>

Table: Tabela de especificação da linguagem.

# Metodologia: Especificação de Convenções

## Convenções sobre BRDFs suportadas

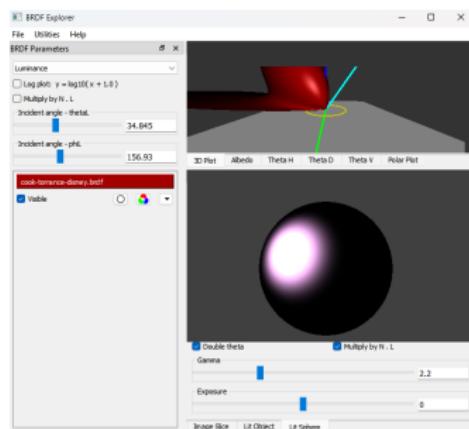
Símbolo	Comando $\text{\LaTeX}$	Descrição
$\theta_i$	<code>\thetaeta_i</code>	Ângulo de elevação da direção da luz incidente
$\theta_o$	<code>\thetaeta_o</code>	Ângulo de elevação da direção da luz refletida
$\phi_i$	<code>\phii_i</code>	Ângulo azimutal da direção da luz incidente
$\phi_o$	<code>\phii_o</code>	Ângulo azimutal da direção da luz refletida
$\omega_i$	<code>\omegai_i</code>	Direção da luz incidente
$\omega_o$	<code>\omegai_o</code>	Direção da luz refletida
$f$	<code>f</code>	BRDF de referência
$\vec{n}$	<code>\vec{n}</code>	Vetor normal à superfície
$\vec{h}$	<code>\vec{h}</code>	Vetor do meio entre $\omega_o$ e $\omega_i$
$\theta_h$	<code>\thetaeta_h</code>	Ângulo entre $\vec{n}$ e $\vec{h}$
$\theta_d$	<code>\thetaeta_d</code>	Ângulo entre $\omega_i$ e $\vec{h}$

Table: Tabela de símbolos e suas descrições

# Metodologia: Experimentos de Renderização

Escolha da ferramenta: **Disney BRDF Explorer:**

- Aceita shaders para renderizador em tempo real
- Interface interativa
- Ajuste dinâmico de parâmetros



## Desenvolvimento: Pacotes

- Organização modular para encapsulamento de responsabilidades:
  - **lexer**: Análise léxica e tokenização (feito com máquina de estados).
  - **parser**: Construção da AST com *Pratt Parsing*.
  - **walker**: Navegação e análise da AST.
  - **checker**: Inferência de tipos e validações semânticas.
  - **emitter**: Geração do código GLSL.

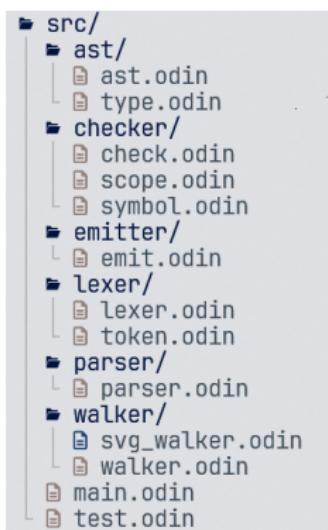


Figure: Estrutura de pacotes do compilador.

# Desenvolvimento: Arquitetura do Compilador

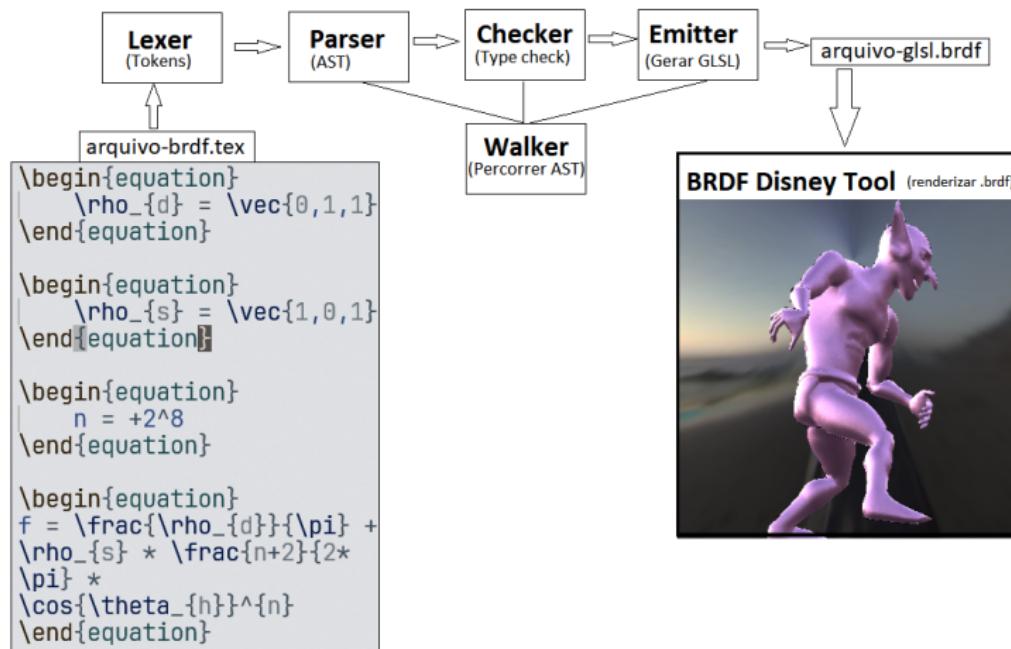


Figure: Estrutura geral da arquitetura do compilador.

## Desenvolvimento: Erros de Compilação (1/3)

Exemplificando as verificações realizadas pelo compilador por meio dos possíveis erros que podem ser gerados.

### Erro Léxico:

```
[excyber @ ~/code/tcc-lang]
$ .bin/unnamed test/should-error/equation.tex          ( new-lang-tcc-mds ) 1:4
14
test/should-error/equation.tex(8:8) error: You can't use a reserved word `arccos` without t
he \ in front, as in `\arccos`  

    7 | \begin{equation}
     8 |     f = arccos
        ^^^^^^
[excyber @ ~/code/tcc-lang]
```

Uso inválido de palavra reservada

### Erro Sintático:

```
[excyber @ ~/code/tcc-lang]
$ .bin/unnamed test/should-error/balanceamento.tex      ( new-lang-tcc-mds ) 2:17
`\\end`  
`)`test/should-error/balanceamento.tex(9:1) error: Expected `)` but got `\\end` instead.  

    8 |     f = ((1 * 2)
     9 |     \\end{equation}
        ^^^
   10 |
[excyber @ ~/code/tcc-lang]
```

Parênteses desbalanceados

## Erros de Compilação (2/3)

### Erro Sintático:

```
$ .bin/unnamed test/should-error/token.tex          ( new-lang-tcc-mds ) 1:54
test/should-error/token.tex(9:16) error: Expected an expression, but got token that can't m
ake a expresion got `'\end`'

8 | \begin{equation}
9 |     f = ----- \end
|           ^^^^
10 | \end{equation}
```

Expressão matemática inválida

### Erro Semântico:

```
[excyber @ ~/code/tcc-lang]
$ .bin/unnamed test/should-error/math.tex          ( new-lang-tcc-mds ) 1:52

parsed test/should-error/math.tex sucessfully
test/should-error/math.tex(8:9) error: Function `exp()` can only have a number type as argu
ment, but we got `R^3`.

7 | \begin{equation}
8 |     f = \exp(\vec{1,1,1})
|           ^^^^
9 | \end{equation}
[excyber @ ~/code/tcc-lang]
```

Tipos incompatíveis

## Erros de Compilação (3/3)

### Erro Semântico:

```
$ .bin/unnamed test/should-error/another.tex          ( new-lang-tcc-mds ) 1:52
parsed test/should-error/another.tex sucessfully
test/should-error/another.tex(8:4) error: Trying to use undefined symbol `\'text_{var}' inside equation `f'
    7 | \begin{equation}
    8 |     f = 2^{\exp(\text{var})}
    |
    9 | \end{equation}
[excyber @ ~/code/tcc-lang]
```

Símbolo não definido

### Erro Semântico:

```
test/should-error/redefinition.tex(12:7) error: Redefinition of symbol `f'
    11 | \begin{equation}
    12 |     f = 2
    |
    13 | \end{equation}
```

Redefinição de símbolo

# Resultados

Experimentos são realizados com diferentes BRDFs para validar o compilador

- Metodologia padronizada:
  - O código fonte de entrada são equações no ambiente `equation` do  $\text{\LaTeX}$
  - Tradução para GLSL pelo compilador desenvolvido neste trabalho
  - Visualização no Disney BRDF Explorer
- Condições controladas:
  - Ângulos de luz fixos ( $\theta_i = 33.89$ ,  $\phi_i = 145.83$ )
  - Gamma = 2.112, Exposição = -1.248

# Resultados: Experimentos

- **11 Experimentos Realizados:**

- ① Blinn-Phong
- ② Cook-Torrance
- ③ Ward
- ④ Ashikhmin-Shirley
- ⑤ Oren-Nayar
- ⑥ Cook-Torrance<sub>2</sub>
- ⑦ Ashikhmin-Shirley<sub>2</sub>
- ⑧ Dür
- ⑨ Edwards-2006
- ⑩ Kajiya-Kay-1989<sub>\*</sub>
- ⑪ Minnaert

- Cada experimento ilustra:

- ① Código fonte de entrada ( $\text{\LaTeX}$ ) e saída (GLSL)
- ② Gráficos 3D e 2D de distribuição de reflexão
- ③ Renderização de objetos 3D

# Experimento Blinn-Phong: Equações da BRDF em documento L<sup>A</sup>T<sub>E</sub>X

$$\rho_d = 0, \vec{1}, 1 \quad (1)$$

$$\rho_s = 1, \vec{0}, 1 \quad (2)$$

$$n = +2^8 \quad (3)$$

$$f = \frac{\rho_d}{\pi} + \rho_s * \frac{n+2}{2*\pi} * \cos \theta_h^n \quad (4)$$

## Experimento Blinn-Phong: Código fonte L<sup>A</sup>T<sub>E</sub>X da BRDF

```
\begin{equation}
    \rho_d = \vec{0,1,1}
\end{equation}
```

```
\begin{equation}
    \rho_s = \vec{1,0,1}
\end{equation}
```

```
\begin{equation}
    n = +2^8
\end{equation}
```

```
\begin{equation}
f = \frac{\rho_d}{\pi} + \rho_s * \frac{n+2}{2*\pi} *
\cos{\theta_h}^n
\end{equation}
```

# Experimento Blinn-Phong: Código GLSL da BRDF deste experimento (1 de 2)

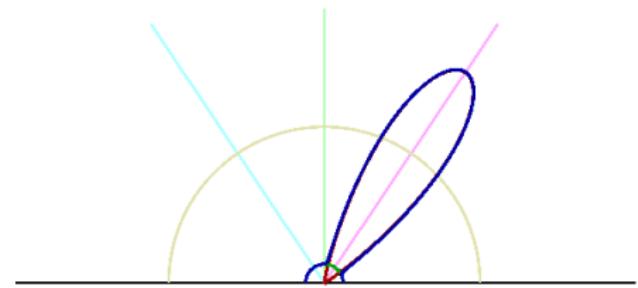
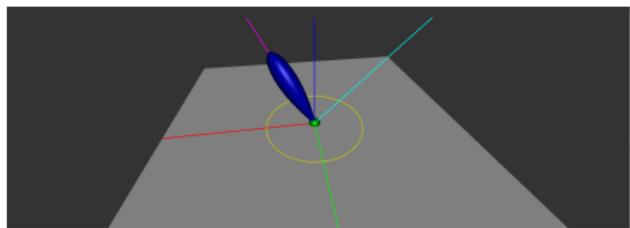
```
1 analytic ::begin parameters
2 #[type] [name] [min val] [max val] [default val]
3 ::end parameters
4 ::begin shader
5 //////////// START OF BUILTINS DECLARTION ///////////
6 vec3 var_0_vec_h;
7 vec3 var_3_vec_n;
8 float var_10_theta_h;
9 float var_11_theta_d;
10 float var_1_pi;
11 float var_2_epsilon;
12 vec3 var_4_vec_omega_i;
13 float var_5_theta_i;
14 float var_6_phi_i;
15 vec3 var_7_vec_omega_o;
16 float var_8_theta_o;
17 float var_9_phi_o;
18 //////////// END OF BUILTINS DECLARTION ///////////
19 //////////// START OF USER DECLARED ///////////
20 vec3 var_12_rho_s;
21 float var_13_n;
22 vec3 var_14_rho_d;
23 vec3 var_15_f;
24 //////////// END OF USER DECLARED ///////////
```

# Experimento Blinn-Phong: Código GLSL da BRDF deste experimento (2 de 2)

```
1 vec3 BRDF(vec3 L, vec3 V, vec3 N, vec3 X, vec3 Y) {
2     ////////////////// START OF BUILTINS INITIALIZATION //////////////////
3     var_0_vec_h = normalize(L + V);
4     var_3_vec_n = normalize(N);
5     var_1_pi = 3.141592653589793;
6     var_2_epsilon = 1.192092896e-07;
7     var_4_vec_omega_i = L;
8     var_5_theta_i = atan(var_4_vec_omega_i.y, var_4_vec_omega_i.x);
9     var_6_phi_i = atan(sqrt(var_4_vec_omega_i.y * var_4_vec_omega_i.y +
10                         var_4_vec_omega_i.x * var_4_vec_omega_i.x),
11                         var_4_vec_omega_i.z);
12    var_7_vec_omega_o = V;
13    var_8_theta_o = atan(var_7_vec_omega_o.y, var_7_vec_omega_o.x);
14    var_9_phi_o = atan(sqrt(var_7_vec_omega_o.y * var_7_vec_omega_o.y +
15                         var_7_vec_omega_o.x * var_7_vec_omega_o.x),
16                         var_7_vec_omega_o.z);
17    var_10_theta_h = acos(dot(var_0_vec_h, N));
18    var_11_theta_d = acos(dot(var_0_vec_h, var_4_vec_omega_i));
19    ////////////////// END OF BUILTINS INITIALIZATION //////////////////
20    var_12_rho_s = vec3(1.0, 0.0, 1.0);
21    var_13_n = pow(2.0, 8.0);
22    var_14_rho_d = vec3(0.0, 1.0, 1.0);
23    var_15_f = ((var_14_rho_d / var_1_pi) +
24                ((var_12_rho_s * ((var_13_n + 2.0) / (2.0 * var_1_pi))) *
25                 pow(cos(var_10_theta_h), var_13_n)));
26    return vec3(var_15_f);
27 }
```

# Experimento Blinn-Phong: Plots de Distribuição de Reflexão

Figure: *Plots* da distribuição de reflexão especular e difusa do experimento Blinn-Phong.



## Experimento Blinn-Phong: Objetos 3D renderizados para esta BRDF.



# Experimento Cook-Torrance: Equações da BRDF em documento L<sup>A</sup>T<sub>E</sub>X

$$m = 0.13 \quad (1)$$

$$\rho_d = 0.3, 0.05, 0.05 \quad (2)$$

$$\rho_s = 0.0, 0.2, 1.0 \quad (3)$$

$$f = \frac{\rho_d}{\pi} + \frac{\rho_s}{\pi} * \frac{D * G}{(\vec{n} \cdot \vec{\omega}_i) * (\vec{n} \cdot \vec{\omega}_o)} \quad (4)$$

$$G = \min(1, \min(\frac{2 * (\vec{n} \cdot \vec{h}) * (\vec{n} \cdot \vec{\omega}_o)}{(\vec{h} \cdot \vec{\omega}_o)}, \frac{2 * (\vec{n} \cdot \vec{h}) * (\vec{n} \cdot \vec{\omega}_i)}{(\vec{h} \cdot \vec{\omega}_o)})) \quad (5)$$

$$D = \frac{1}{(m^2) * (\cos \theta_h)^4} * \exp -((\tan \theta_h)/m)^2 \quad (6)$$

# Experimento Cook-Torrance: Código fonte L<sup>A</sup>T<sub>E</sub>X da BRDF

```
\begin{equation}
m = 0.13
\end{equation}

\begin{equation}
\rho_d = \vec{0.3, 0.05, 0.05}
\end{equation}

\begin{equation}
\rho_s = \vec{0.0, 0.2, 1.0}
\end{equation}

\begin{equation}
f = \frac{\rho_d}{\pi} + \frac{\rho_s}{\pi} *
\frac{D * G((\vec{n}) \cdot \vec{\omega_i})) * ((\vec{n}) \cdot \vec{\omega_o}))}{(\vec{n} \cdot \vec{\omega_i}) * ((\vec{h}) \cdot \vec{\omega_o}))}
\end{equation}

\begin{equation}
G = \min(1, \min(\frac{2 * ((\vec{n}) \cdot \vec{h}) * ((\vec{n}) \cdot \vec{\omega_o}))}{((\vec{n}) \cdot \vec{\omega_o}))}, \frac{2 * ((\vec{n}) \cdot \vec{h}) * ((\vec{n}) \cdot \vec{\omega_i}))}{((\vec{n}) \cdot \vec{\omega_i}))})
\end{equation}

\begin{equation}
D = \frac{1}{(m^2) * (\cos(\theta_h))^4} * \exp(-((\tan(\theta_h)) / m)^2)
\end{equation}
```

# Experimento Cook-Torrance: Código GLSL da BRDF deste experimento (1 de 2)

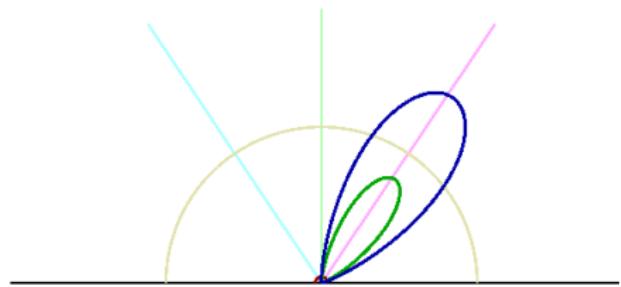
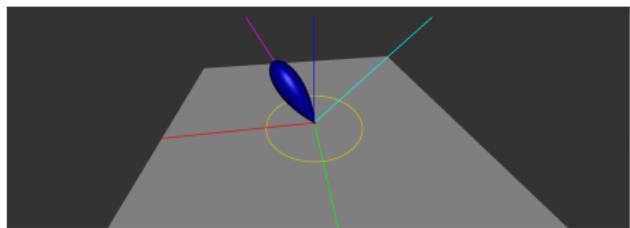
```
1 analytic ::begin parameters
2 #[type][name][min val][max val][default val]
3 ::end parameters
4 ::begin shader
5 //////////// START OF BUILTINS DECLARTION ///////////
6 vec3 var_0_vec_h;
7 vec3 var_3_vec_n;
8 float var_10_theta_h;
9 float var_11_theta_d;
10 float var_1_pi;
11 float var_2_epsilon;
12 vec3 var_4_vec_omega_i;
13 float var_5_theta_i;
14 float var_6_phi_i;
15 vec3 var_7_vec_omega_o;
16 float var_8_theta_o;
17 float var_9_phi_o;
18 //////////// END OF BUILTINS DECLARTION ///////////
19 //////////// START OF USER DECLARED ///////////
20 float var_12_G;
21 vec3 var_13_rho_s;
22 float var_14_m;
23 float var_15_D;
24 vec3 var_16_rho_d;
25 vec3 var_17_f;
26 //////////// END OF USER DECLARED ///////////
27 //////////// START FUNCTIONS DECLARATIONS ///////////
28 //////////// END FUNCTIONS DECLARATIONS ///////////
```

# Experimento Cook-Torrance: Código GLSL da BRDF deste experimento (2 de 2)

```
1 vec3 BRDF(vec3 L, vec3 V, vec3 N, vec3 X, vec3 Y) {
2
3     ////////////////// START OF BUILTINS INITIALIZATION //////////////////
4     var_0_vec_h = normalize(L + V);
5     var_3_vec_n = normalize(N);
6     var_1_pi = 3.141592653589793;
7     var_2_epsilon = 1.192092896e-07;
8     var_4_vec_omega_i = L;
9     var_5_theta_i = atan(var_4_vec_omega_i.y, var_4_vec_omega_i.x);
10    var_6_phi_i = atan(sqrt(var_4_vec_omega_i.y * var_4_vec_omega_i.y +
11                           var_4_vec_omega_i.x * var_4_vec_omega_i.x), var_4_vec_omega_i.z);
12    var_7_vec_omega_o = V;
13    var_8_theta_o = atan(var_7_vec_omega_o.y, var_7_vec_omega_o.x);
14    var_9_phi_o = atan(sqrt(var_7_vec_omega_o.y * var_7_vec_omega_o.y + var_7_vec_omega_o.x * var_7_vec_omega_o.x),
15                         var_7_vec_omega_o.z);
16    var_10_theta_h = acos(dot(var_0_vec_h, N));
17    var_11_theta_d = acos(dot(var_0_vec_h, var_4_vec_omega_i));
18    ////////////////// END OF BUILTINS INITIALIZATION //////////////////
19
20    var_12_G = min(1.0, min(((2.0 * (dot(var_3_vec_n, var_0_vec_h))) * (dot(var_3_vec_n, var_7_vec_omega_o))) / (dot(
21        var_0_vec_h, var_7_vec_omega_o)), (((2.0 * (dot(var_3_vec_n, var_0_vec_h))) *
22            (dot(var_3_vec_n, var_4_vec_omega_i))) /
23            (dot(var_0_vec_h, var_7_vec_omega_o)))));
24    var_13_rho_s = vec3(0.0, 0.2, 1.0);
25    var_14_m = 0.13;
26    var_15_D = ((1.0 / ((pow(var_14_m, 2.0)) * pow((cos(var_10_theta_h)), 4.0))) *
27                 exp((-pow(((tan(var_10_theta_h)) / var_14_m)), 2.0)));
28    var_16_rho_d = vec3(0.3, 0.05, 0.05);
29    var_17_f = ((var_16_rho_d / var_1_pi) + ((var_13_rho_s / var_1_pi) * ((var_15_D * var_12_G) / ((dot(var_3_vec_n,
30                     var_4_vec_omega_i)) * (dot(var_3_vec_n, var_7_vec_omega_o))))));
31    return vec3(var_17_f);
32 }
```

# Experimento Cook-Torrance: Plots de Distribuição de Reflexão

Figure: Plots da distribuição de reflexão especular e difusa do experimento cook-torrance.



# Experimento Cook-Torrance: Objetos 3D renderizados para esta BRDF



# Experimento Edwards-2006: Equações da BRDF em documento L<sup>A</sup>T<sub>E</sub>X

7

Edwards halfway-vector disk (2006)

$$n = 10 \quad (1)$$

$$R = 1 \quad (2)$$

$$\text{lump}(\vec{h}, R, n) = (n + 1)/(\pi * R * R) * (1 - (\vec{h} \cdot \vec{h})/(R * R)^n) \quad (3)$$

Scaling projection

$$uH = \vec{\omega}_i + \vec{\omega}_o \quad (4)$$

$$h = (\vec{n} \cdot \vec{\omega}_o) / (\vec{n} \cdot uH) * uH \quad (5)$$

$$huv = h - (\vec{n} \cdot \vec{\omega}_o) * \vec{n} \quad (6)$$

Specular term (D and G)

$$p = \text{lump}(huv, R, n) \quad (7)$$

$$f = p * ((\vec{n} \cdot \vec{\omega}_o)^2) / (4 * (\vec{n} \cdot \vec{\omega}_i) * (\vec{\omega}_i \cdot \vec{h}) * ((\vec{n} \cdot \vec{h})^3)) \quad (8)$$

<sup>7</sup> Experimento adicionado aos slides para demonstrar definição de funções no formato  $f(x, y) = x + y$ .

# Experimento Edwards-2006: Código fonte L<sup>A</sup>T<sub>E</sub>X da BRDF

```
\begin{equation}
n = 10
\end{equation}
\begin{equation}
R = 1
\end{equation}

\begin{equation}
\text{lump}(\vec{h}, R, n) = (n+1)/(\pi * R * R) * (1 - (\vec{h} \cdotdot \vec{h}) / (R * R)^n)
\end{equation}

Scaling projection
\begin{equation}
uH = \vec{\omega_i} + \vec{\omega_o} \% // unnormalized H
\end{equation}
\begin{equation}
h = (\vec{n} \cdotdot \vec{\omega_o}) / (\vec{n} \cdotdot uH) * uH
\end{equation}
\begin{equation}
huv = h - (\vec{n} \cdotdot \vec{\omega_o}) * \vec{n}
\end{equation}
\begin{equation}
p = \text{lump}(huv, R, n)
\end{equation}
\begin{equation}
f = p * ((\vec{n} \cdotdot \vec{\omega_o})^2
/ (4 * (\vec{n} \cdotdot \vec{\omega_i}))
* (\vec{\omega_i} \cdotdot \vec{h}) * ((\vec{n} \cdotdot \vec{h})^3))
\end{equation}
```

# Experimento Edwards-2006: Código GLSL da BRDF deste experimento (1 de 2)

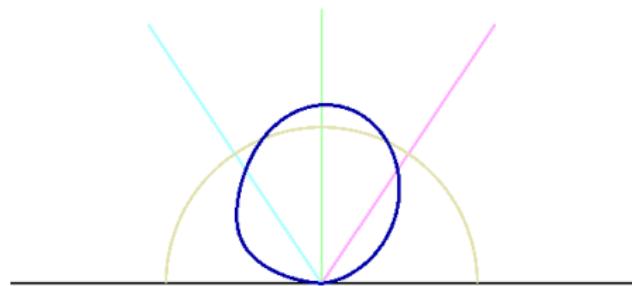
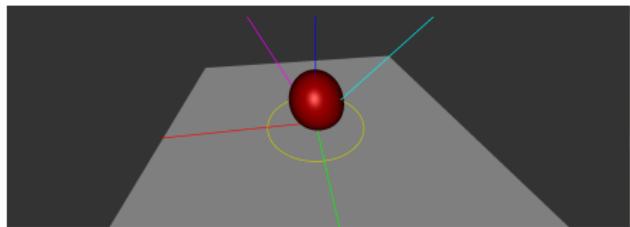
```
1 analytic ::begin parameters
2 #[type][name][min val][max val][default val]
3 ::end parameters
4 ::begin shader
5 //////////// START OF BUILTINS DECLARTION ///////////
6 vec3 var_0_vec_h;
7 vec3 var_3_vec_n;
8 float var_10_theta_h;
9 float var_11_theta_d;
10 float var_1_pi;
11 float var_2_epsilon;
12 vec3 var_4_vec_omega_i;
13 float var_5_theta_i;
14 float var_6_phi_i;
15 vec3 var_7_vec_omega_o;
16 float var_8_theta_o;
17 float var_9_phi_o;
18 //////////// END OF BUILTINS DECLARTION ///////////
19 //////////// START OF USER DECLARED ///////////
20 vec3 var_15_uH;
21 vec3 var_16_h;
22 float var_14_n;
23 vec3 var_17_huv;
24 float var_13_R;
25 float var_18_p;
26 float var_19_f;
27 //////////// END OF USER DECLARED ///////////
28 //////////// START FUNCTIONS DECLARATIONS ///////////
29 float var_12_text_lump(vec3 var_0_vec_h, float var_13_R, float var_14_n) {
30     return (((var_14_n + 1.0)) / ((var_1_pi * var_13_R) * var_13_R)) *
31         ((1.0 - ((dot(var_0_vec_h, var_0_vec_h)) /
32             pow((var_13_R * var_13_R), var_14_n))));
```

# Experimento Edwards-2006: Código GLSL da BRDF deste experimento (2 de 2)

```
1 ////////////////////////////////////////////////////////////////// END FUNCTIONS DECLARATIONS //////////////////////////////////////////////////////////////////
2 vec3 BRDF(vec3 L, vec3 V, vec3 N, vec3 X, vec3 Y) {
3     ////////////////////////////////////////////////////////////////// START OF BUILTINS INITIALIZATION //////////////////////////////////////////////////////////////////
4     var_0_vec_h = normalize(L + V);
5     var_3_vec_n = normalize(N);
6     var_1_pi = 3.141592653589793;
7     var_2_epsilon = 1.192092896e-07;
8     var_4_vec_omega_i = L;
9     var_5_theta_i = atan(var_4_vec_omega_i.y, var_4_vec_omega_i.x);
10    var_6_phi_i = atan(sqrt(var_4_vec_omega_i.y * var_4_vec_omega_i.y +
11                           var_4_vec_omega_i.x * var_4_vec_omega_i.x), var_4_vec_omega_i.z);
12    var_7_vec_omega_o = V;
13    var_8_theta_o = atan(var_7_vec_omega_o.y, var_7_vec_omega_o.x);
14    var_9_phi_o = atan(sqrt(var_7_vec_omega_o.y * var_7_vec_omega_o.y +
15                           var_7_vec_omega_o.x * var_7_vec_omega_o.x), var_7_vec_omega_o.z);
16    var_10_theta_h = acos(dot(var_0_vec_h, N));
17    var_11_theta_d = acos(dot(var_0_vec_h, var_4_vec_omega_i));
18    ////////////////////////////////////////////////////////////////// END OF BUILTINS INITIALIZATION //////////////////////////////////////////////////////////////////
19    var_15_uH = (var_4_vec_omega_i + var_7_vec_omega_o);
20    var_16_h = (((dot(var_3_vec_n, var_7_vec_omega_o)) /
21      (dot(var_3_vec_n, var_15_uH))) * var_15_uH);
22    var_14_n = 10.0;
23    var_17_huv = (var_16_h - ((dot(var_3_vec_n, var_7_vec_omega_o)) * var_3_vec_n));
24    var_13_R = 1.0;
25    var_18_p = var_12_text_lump(var_17_huv, var_13_R, var_14_n);
26    var_19_f = ((var_18_p * (pow((dot(var_3_vec_n, var_7_vec_omega_o)), 2.0))) /
27      (((4.0 * (dot(var_3_vec_n, var_4_vec_omega_i))) *
28        (dot(var_4_vec_omega_i, var_0_vec_h))) * (pow((dot(var_3_vec_n, var_0_vec_h)), 3.0))));;
29    return vec3(var_19_f);
30 }
```

# Experimento Edwards-2006: Plots de Distribuição de Reflexão

Figure: Plots da distribuição de reflexão especular e difusa do experimento edwards-2006.



# Experimento Edwards-2006: Objetos 3D renderizados para esta BRDF



## Resultados: Visão Simplificada dos Outros Experimentos

Experimento	Blinn-Phong	Cook-Torrance	Ward	Ashikhmin-Shirley
Visualização				

Experimento	Oren-Nayar	Ashikhmin-Shirley <sub>2</sub>	Cook-Torrance <sub>2</sub>	Dür
Visualização				

Experimento	Edwards-2006	Kajiya-Kay-1989_*	Minnaert
Visualização			

## Conclusão

Este trabalho alcançou seu objetivo de **desenvolver um compilador capaz de traduzir BRDFs descritas em L<sup>A</sup>T<sub>E</sub>X para código GLSL**. De forma que:

- Reduz a **barreira técnica** para a criação de *shaders*, permitindo que profissionais de áreas não relacionadas à programação explorem efeitos visuais complexos.
- Aumenta a **interatividade** no design das BRDFs.
- Suporta a **integração** com a ferramenta gratuita da Disney para renderização visual das BRDFs compiladas.
- Permite que os usuários **focalizem na modelagem** de refletância, sem a necessidade de lidar com detalhes de baixo nível.
- Demonstra-se capaz por meio de experimentos de renderização bem-sucedidos.

## Conclusão: Conhecimentos Usados

Utilizamos conhecimentos interdisciplinares para realizar este trabalho, tais como:

- Gramáticas livres de contexto e geração de código.
- Algoritmos com árvores e grafos.
- Renderização projetiva, *raytracing* e programação com *shaders*.
- Fundamentos teóricos de refletância e radiometria.

## Conclusão: Contribuições

- Tradução de equações  $\text{\LaTeX}$  com suporte as convenções mais usadas na definição de BRDFs ( $\omega_i, \omega_o, \theta_i, \vec{n}, \pi\dots$ ):
- Visualização das árvores sintáticas geradas (SVG).
- Geração de mensagens de erro informativos para facilitar depuração.
- Suporte a operações fundamentais em computação gráfica:
  - Produto interno, vetorial.
  - Funções trigonométricas e exponenciação.

## Conclusão: Perspectivas Futuras

- Ampliar suporte a novas construções matemáticas:
  - Somatórios ( $\Sigma$ ) e produtos acumulados ( $\Pi$ ).
  - Vetores de mais dimensões do que apenas 3.
- Adicionar derivadas ( $\frac{d}{dx}$ ) e integrais ( $\int$ ) para resolução numérica.
- Suporte a outras linguagens de *shading* como a usada por Unity<sup>8</sup> e Unreal.
- Desenvolver um editor  $\text{\LaTeX}$  integrado com visualização simultânea.
- Melhorar o tratamento de erros para maior clareza e contextualização.

---

<sup>8</sup>HLSL

# Conclusão: Encerramento

**Fim<sup>9</sup>**

---

<sup>9</sup>evertonse.junior@gmail.com.

## Referências

- [Kaj86] James T Kajiya. "The rendering equation". In: *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 1986, pp. 143–150.
- [Mon21] Shaders Monthly. *Microfacet BRDF: Theory and Implementation of Basic PBR Materials* [Shaders Monthly #9]. Accessed: 2023-10-03. 2021. URL: <https://www.youtube.com/watch?v=gya7x9H3mV0>.
- [PJH16] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation* (3rd ed.) 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Nov. 2016, p. 1266. ISBN: 9780128006450.

# Modelos de BRDFs

- BRDF (Bidirectional Reflectance Distribution Function): Função que descreve como a luz é refletida por uma superfície.
- Apresentaremos modelos clássicos de BRDFs:
  - BRDF Pura Especular.
  - BRDF Difusa Ideal.
  - BRDF Brilhante.
  - BRDF Retro-Refletora.

## BRDF Pura Especular

- Superfície reflete luz apenas em uma direção.
- Segue a Lei da Reflexão:

$$f(\omega_i, \omega_o) = k_s \cdot \delta(\omega_i - \omega_o)$$

- Materiais típicos: metal polido, vidro.

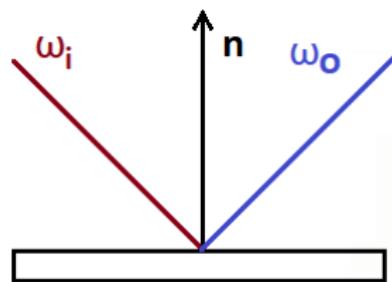


Figure: Reflexão especular. Raio incidente em vermelho e raio refletido em azul.

## BRDF Difusa Ideal

- Reflexão uniforme em todas as direções.
- Função BRDF:

$$f(\omega_i, \omega_o) = \frac{\rho_d}{\pi} \cdot \cos \theta$$

- Exemplos: tinta fosca, papel.

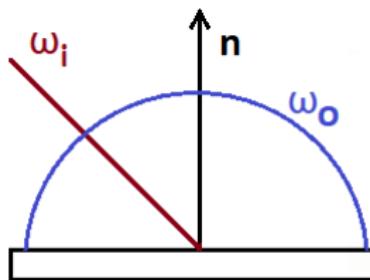


Figure: Reflexão difusa. Raios refletidos independem do ângulo de entrada.

## BRDF Brilhante

- Combina reflexões especulares e difusas.
- Modelo típico: Blinn-Phong.

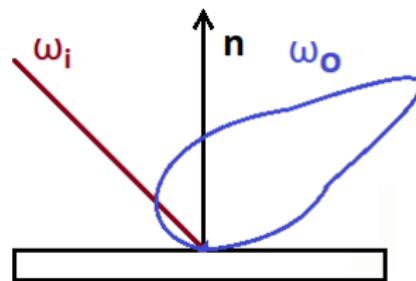


Figure: Reflexão *glossy*.

## BRDF Retro-Refletora

- Redireciona a luz de volta à fonte.
- Utilizado em superfícies como placas de trânsito e sinais de segurança.

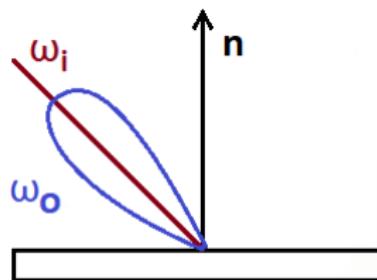


Figure: Reflexão retro-refletora.

# BRDF: Função de Refletância Bidirecional

- BRDF ( $f$ ) descreve como a luz reflete em diferentes direções:

$$f(\omega_i, \omega_o) = \frac{dL_o(\omega_o)}{L_i(\omega_i) \cos(\theta_i) d\omega_i}$$

- Propriedades:

- ① Positividade:  $f \geq 0$ .
- ② Conservação de energia:  $\int_{\Omega} f \cos(\theta_i) d\omega_i \leq 1$ .
- ③ Reciprocidade de Helmholtz:  $f(\omega_i, \omega_o) = f(\omega_o, \omega_i)$ .

# Radiometria: Introdução

- Estuda a interação da luz com superfícies na computação gráfica.
- Quantifica energia luminosa:
  - Brilho da fonte de luz.
  - Iluminação e refletância da superfície.
- Fundamenta a renderização realista de cenas tridimensionais.

# Energia Radiante $Q$

- Quantifica a energia total dos fótons atingindo uma superfície.
- Fórmula:

$$Q = \frac{hc}{\lambda}$$

onde:

- $h$ : Constante de Planck.
- $c$ : Velocidade da luz.
- $\lambda$ : Comprimento de onda.

# Fluxo Radiante e Irradiância

- Fluxo Radiante: Energia por unidade de tempo ( $J/s$ ):

$$\phi = \frac{dQ}{dt}$$

- Irradiância: Fluxo radiante por unidade de área:

$$E(p) = \frac{d\phi(p)}{dA}$$

- Quantifica os impactos de fótons em uma superfície.<sup>10</sup>

---

<sup>10</sup>Saber quantos fótons "tocam" uma superfície por segundo, é saber quão iluminado é aquela parte do objeto.

# Radiância

Radiância ( $L$ ): Fluxo radiante por unidade de área e ângulo sólido:

$$L = \frac{d^2\Phi}{dA d\omega \cos(\theta)}$$

Onde:

- $d\omega$ : Ângulo sólido (em sr).
- $\theta$ : Ângulo entre direção de incidência e normal da superfície.

## Visualização da Radiância

- Radiância considera direção específica no hemisfério sobre a superfície.

Figure: Visualização da radiância em uma direção específica do hemisfério.

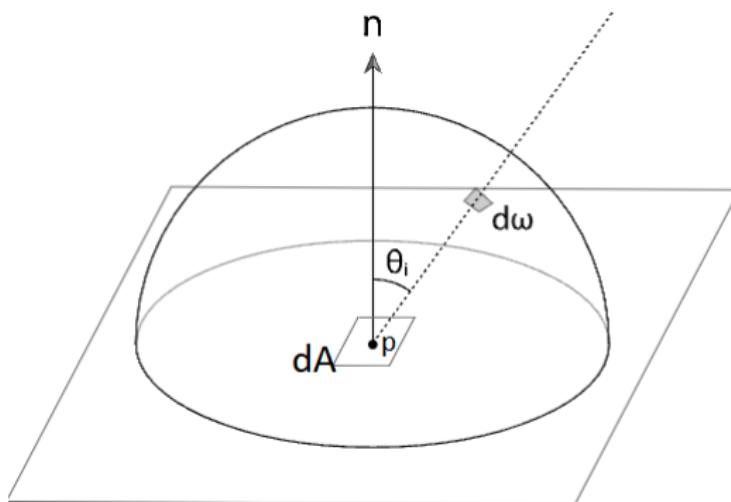


Figure: Ângulo sólido  $s$  do objeto B visto pelo ponto  $p$ .

# Árvore SVG