UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Métodos para Rendering de Superfície

Disciplina: Computação Gráfica (BCC35F)
Curso: Ciência da Computação

Prof. Walter T. Nakamura
waltertakashi@utfpr.edu.br

Campo Mourão - PR

Baseados nos materiais elaborados pelas professoras Aretha Alencar (UTFPR) e Rosane Minghim (USP)

Sumário

1) Introdução
2) Métodos de Rendering
   - Rendering de Superfície de Intensidade Constante
   - Rendering de Superfície de Gouraud
   - Rendering de Superfície de Phong
3) Programação OpenGL (Rendering)

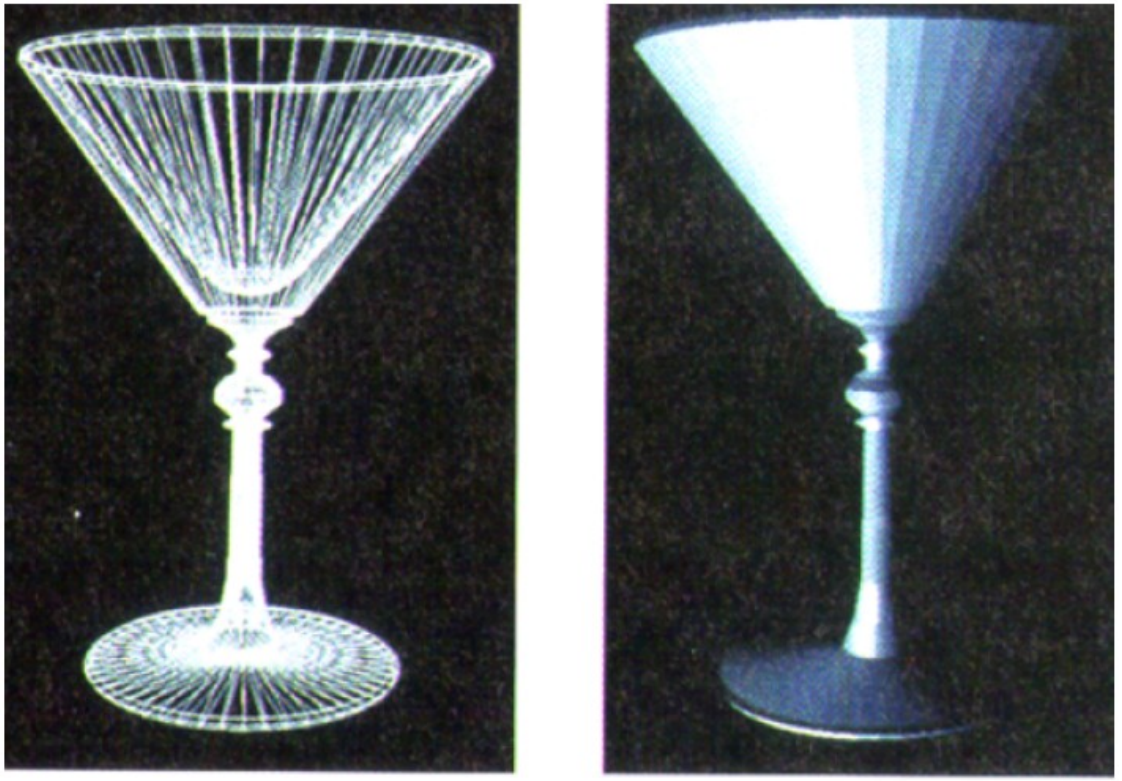
Sumário

Introdução

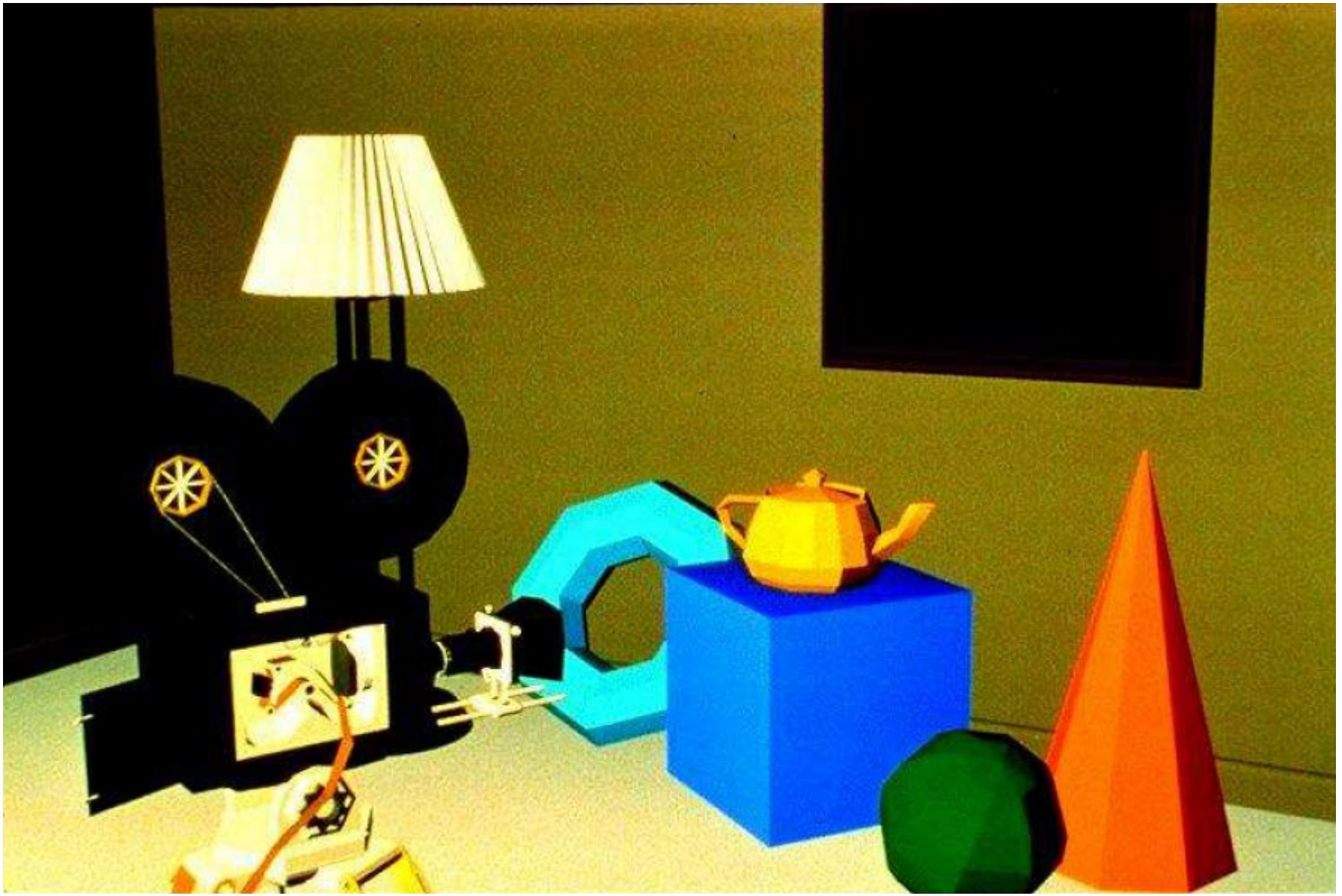
- Baseado no modelo de iluminação, um método de **rendering de superfície** é usado para determinar a **cor dos pixels**
- O modelo de iluminação pode ser usado de formas diferentes para definir a cor de uma superfície:
  - **Ray-tracing:** executado em cada pixel projetado (realismo)
  - **Scan-line:** executado em alguns pixels e interpolado no restante (tempo real)
- Maioria das APIs gráficas reduz o processamento usando algoritmos de **scan-line**
  - As **intensidades** são calculadas nos vértices e interpoladas nas posições restantes dos polígonos

Métodos de Rendering de Superfície

- O método mais simples para renderizar uma superfície é usar a mesma cor para todos seus pontos (**flat surface rendering**)
- Emprega-se o modelo de iluminação para determinar a intensidade das 3 componentes RGB em uma única posição da superfície.
  - Vértice ou centroide do polígono



Métodos de Rendering de Superfície



Sumário

1) Introdução
2) Métodos de Rendering
   - Rendering de Superfície de Intensidade Constante
   - Rendering de Superfície de Gouraud
   - Rendering de Superfície de Phong
3) Programação OpenGL (Rendering)

Métodos de Rendering de Superfície

- O **flat surface rendering** normalmente define resultados precisos se:
  - O polígono é uma face de um poliedro e **não** uma seção de uma **superfície curva**
  - Todas as **fontes de luz** estão **distantes** o suficiente da superfície de forma que $N \cdot L_l$ e a função de **atenuação** são **constantes**
  - A **posição visão** é distante o suficiente do polígono de forma que $V \cdot R_l$ é constante

$$I = k_a I_a + \sum_{l=1}^{n} I_l \left[k_d (N \cdot L_l) + k_s (V \cdot R_l)^{n_s} \right]$$

- Mesmo se alguma dessas condições for falsa, uma boa aproximação pode ser conseguida se os polígonos empregados forem pequenos.

Sumário

1) Introdução
2) Métodos de Rendering
   - Rendering de Superfície de Intensidade Constante
   - Rendering de Superfície de Gouraud
   - Rendering de Superfície de Phong
3) Programação OpenGL (Rendering)

Rendering de Superfície de Gouraud

- O esquema de **Gouraud surface rendering** interpola linearmente as intensidades nos vértices por toda face do polígono de um objeto iluminado.
- Desenvolvido para **aproximar superfícies curvas**, amenizando as transições de intensidades entre polígonos adjacentes
  - Elimina as descontinuidades de intensidades do **flat surface rendering**

Rendering de Superfície de Gouraud

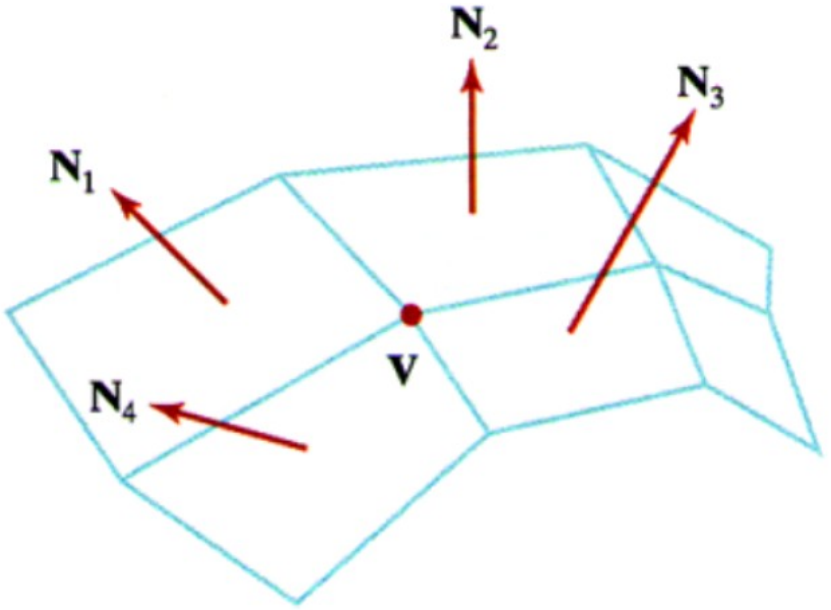
- Cada polígono de uma superfície é processado usando o seguinte procedimento:
  1) Determina o vetor unitário normal médio em cada vértice do polígono
  2) Aplica o modelo de iluminação em cada vértice para obter as intensidades
  3) Interpola linearmente as intensidades dos vértices sobre a área projetada do polígono

Rendering de Superfície de Gouraud

- O vetor normal médio em um vértice é obtido fazendo a média das normais de todos os polígonos que compartilham esse vértice:

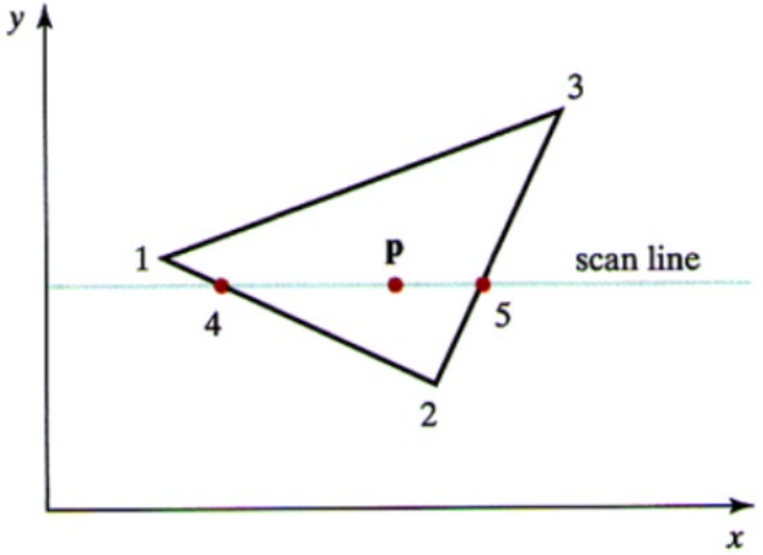
$$N_v = \frac{\sum_{k=1}^{n} N_k}{\left|\sum_{k=1}^{n} N_k\right|}$$

- Usando essas normais o modelo de iluminação é então executado para calcular as intensidades em cada vértice.



Rendering de Superfície de Gouraud

- Esses **valores de intensidade** são então interpolados para se obter as intensidades ao longo de *scan-lines* que intersectam a área projetada do polígono.



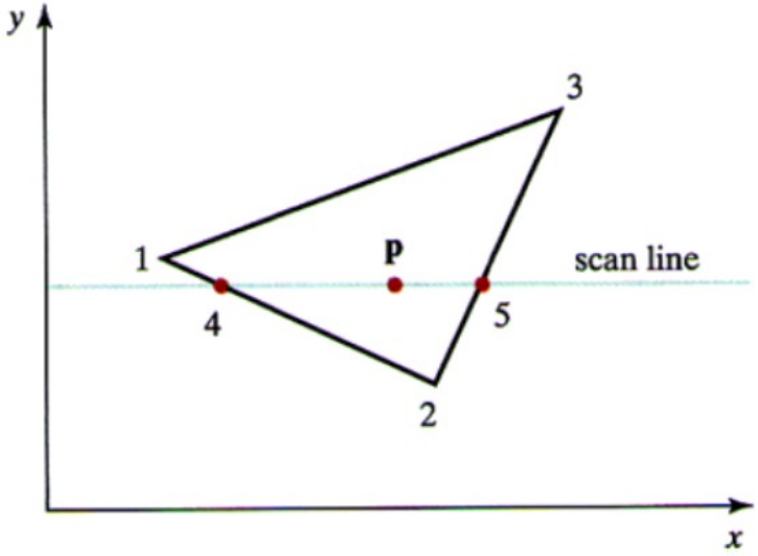
- As intensidades das intersecções das *scan-lines* com as arestas dos polígonos são calculadas interpolando linearmente as intensidades dos pontos finais das arestas.

Rendering de Superfície de Gouraud

- Por exemplo, a intensidade em 4 pode ser calculada considerando somente o deslocamento vertical da scan-line:

$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

$$I_5 = \frac{y_5 - y_2}{y_3 - y_2} I_3 + \frac{y_3 - y_5}{y_3 - y_2} I_2$$

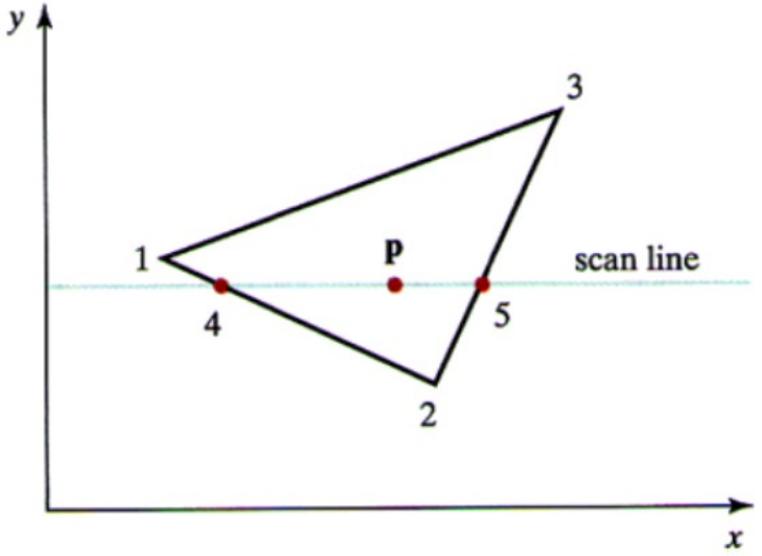


- A intensidade em 5 pode ser obtida da mesma forma interpolando verticalmente as intensidades em 2 e 3.

Rendering de Superfície de Gouraud

- Considerando as intensidades obtidas em 4 e 5, as intensidades em qualquer ponto p da *scan-line* são obtidas interpolando na horizontal:

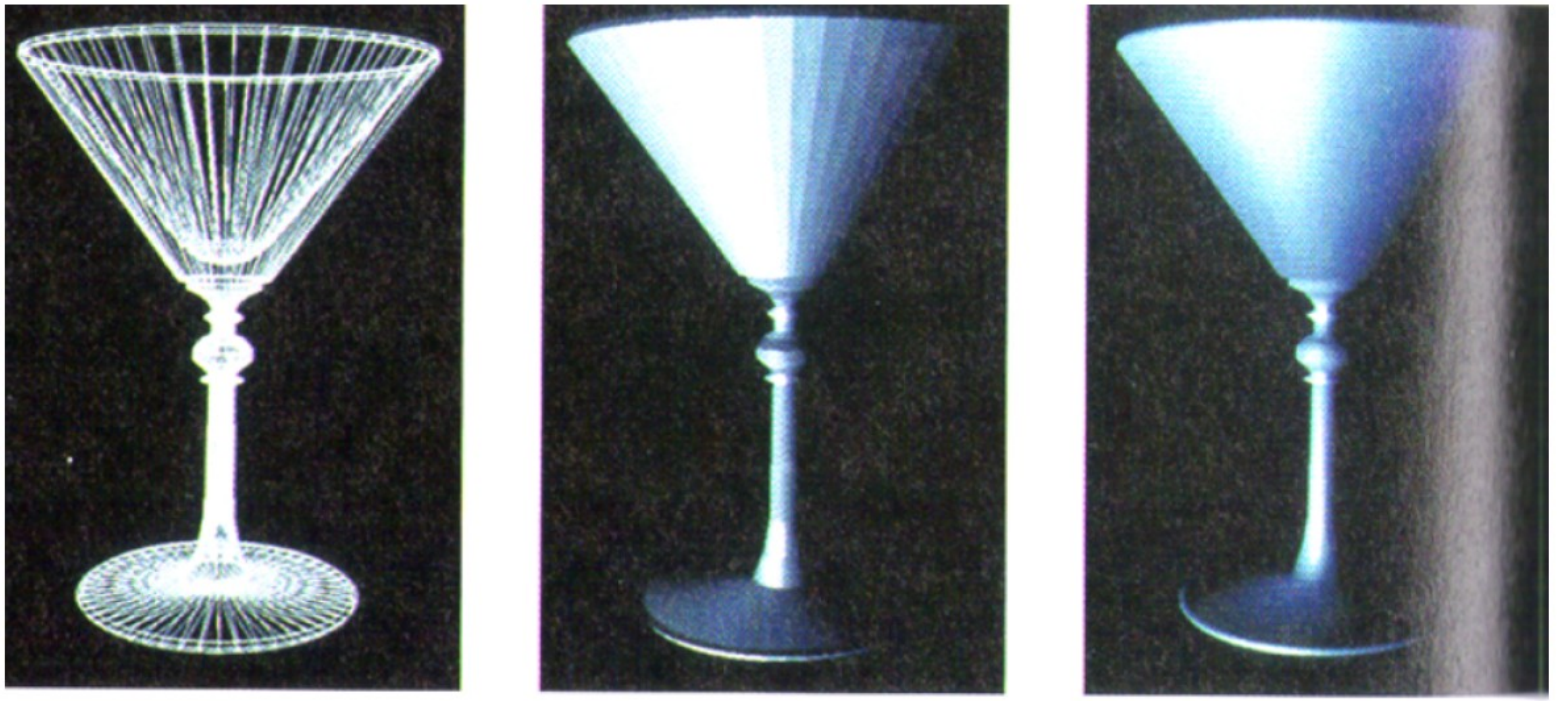
$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$



- Esse método é conhecido como **interpolação bilinear** e é executado para os 3 componentes RGB separadamente

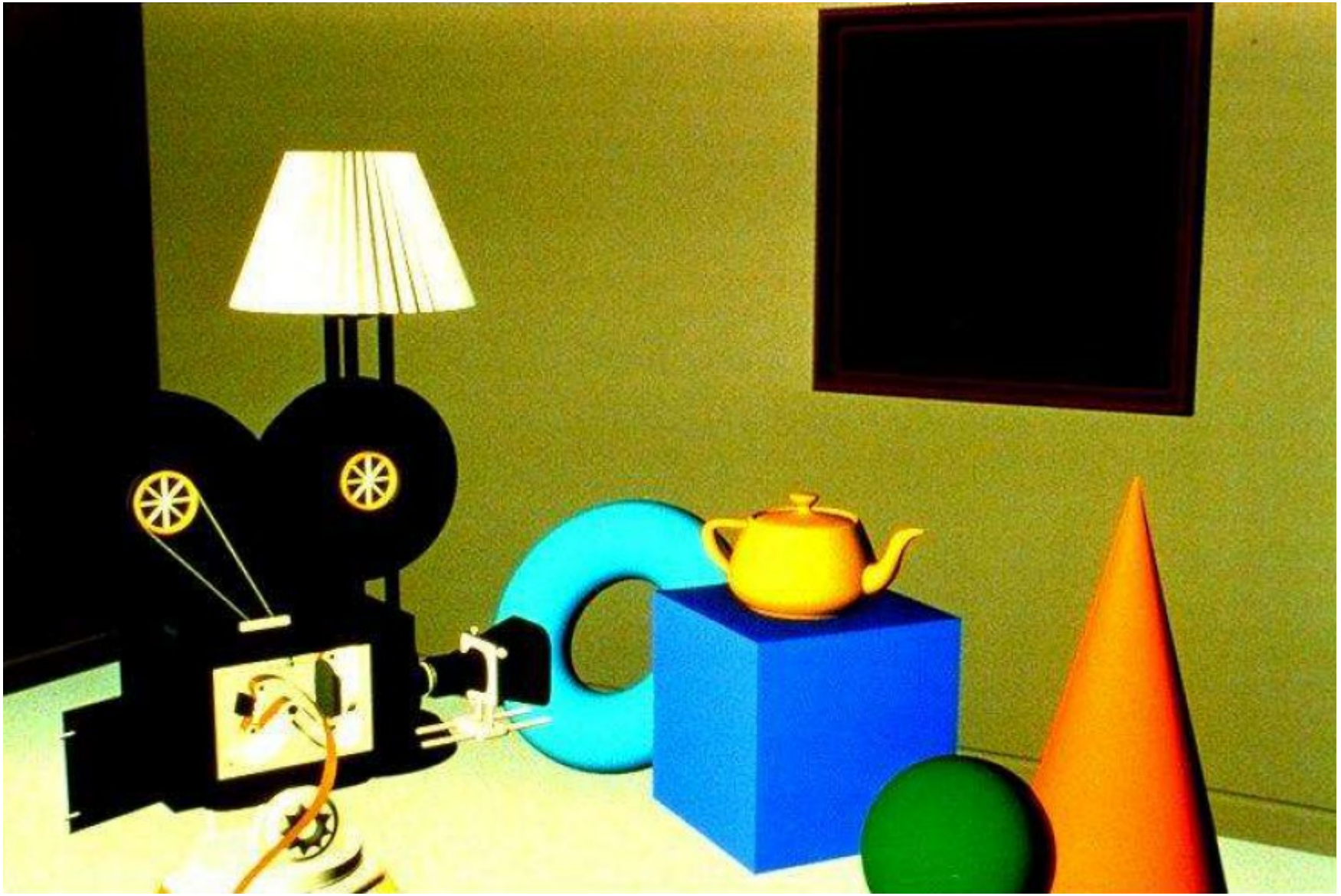
Rendering de Superfície de Gouraud

- Essa interpolação de intensidades **elimina descontinuidades** mas tem alguns problemas:
  - Brilhos na superfície podem apresentar formatos estranhos
  - Intensidades claras ou escuras podem parecer riscadas (mach bands)



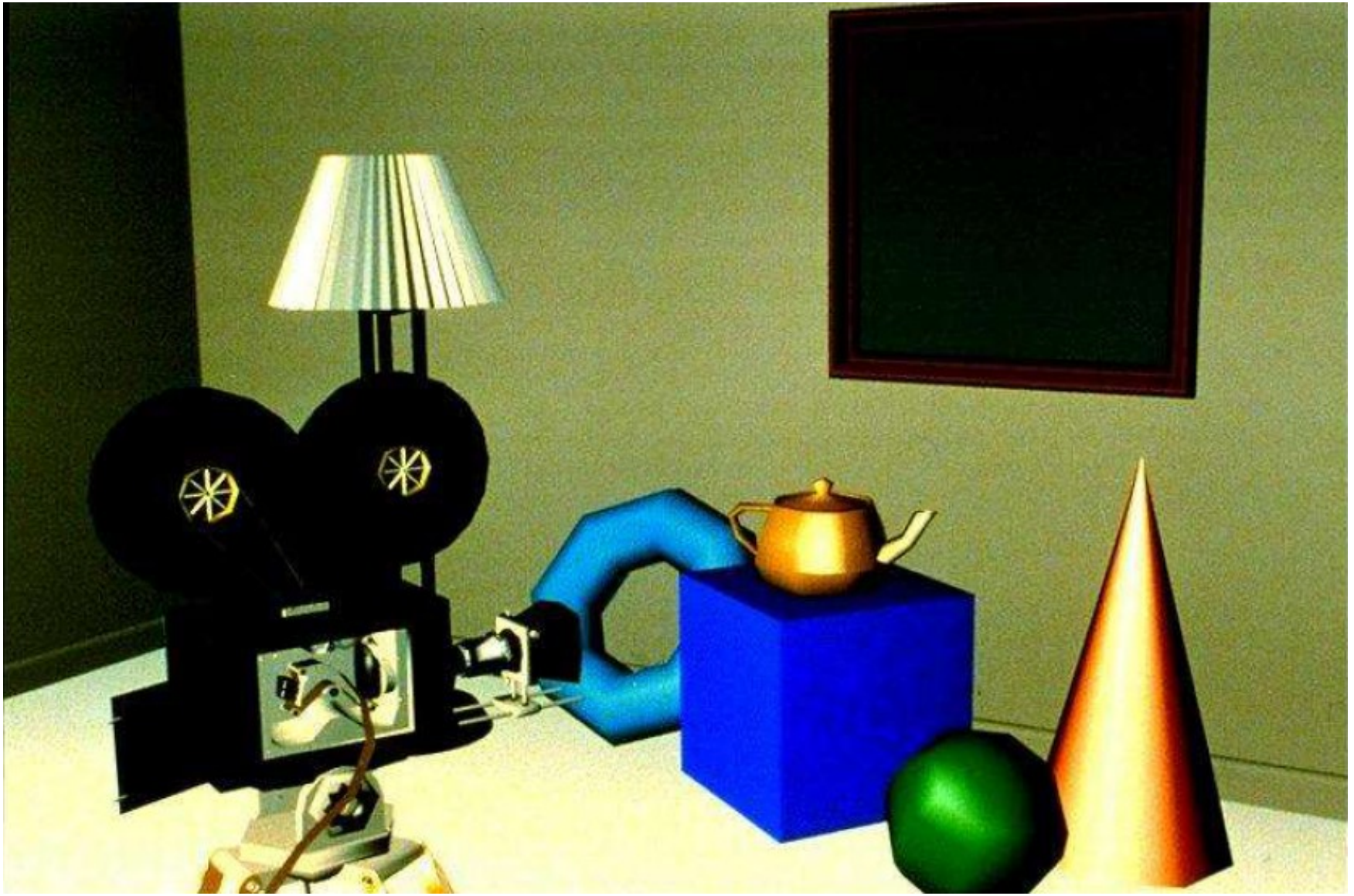
(a) (b) (c)

Rendering de Superfície de Gouraud



Gouraud Shading (sem highlight especular)

Rendering de Superfície de Gouraud



Gouraud Shading (com highlight especular)

Rendering de Superfície de Gouraud

- Efeito de **mach bands** consiste em faixas claras ou escuras que são percebidas próximo das fronteiras entre duas regiões de diferentes gradientes de luz

Sumário

Rendering de Superfície de Phong

- Um método **mais preciso** de interpolação é conhecido como **Phong surface rendering**
- Ao invés de interpolar valores de intensidades, **normais** são interpoladas:
  - Cálculos mais precisos de intensidades
  - Brilhos mais realísticos nas superfícies
  - Redução do efeito mach-band
- Computacionalmente **mais caro** que o método de Gouraud

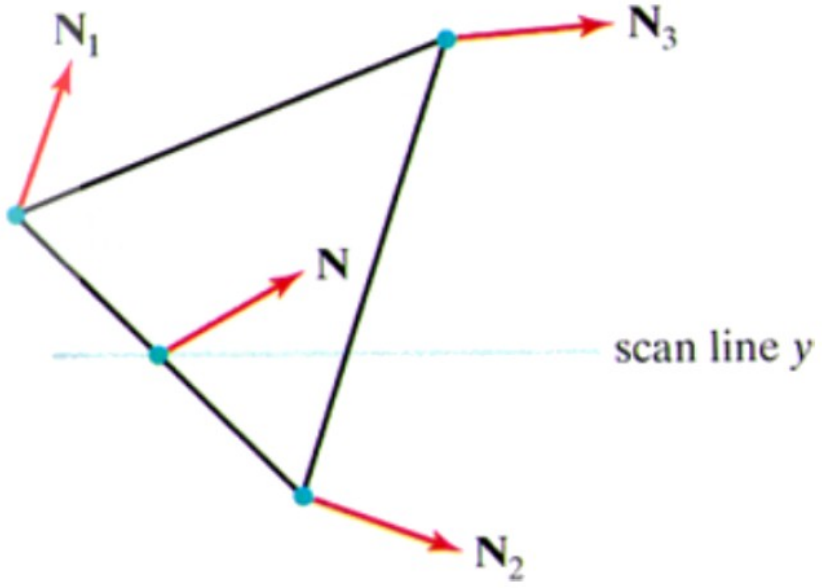
Rendering de Superfície de Phong

- Cada polígono é processado usando o seguinte procedimento:
  1) Determina o **vetor unitário normal médio** em cada vértice do polígono
  2) **Interpola linearmente** as normais dos vértices sobre a área projetada do polígono
  3) **Aplica o modelo de iluminação** nas posições ao longo da scan-line para calcular as intensidades dos pixels usando as normais interpoladas

Rendering de Superfície de Phong

- O procedimento de interpolação das normais é o mesmo da interpolação das intensidades do método de Gouraud
- Por exemplo, o vetor normal N é verticalmente interpolado a partir das normais nos vértices 1 e 2 fazendo:

$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$



- Normais precisam ser renormalizadas!

OpenGL Shading Language

- **OpenGL Shading Language (GLSL)** é uma linguagem de alto nível que permite aos desenvolvedores maior controle sobre o pipeline gráfico sem precisar se preocupar com aspectos específicos de hardware
- Usando a GLSL é possível usar o modelo de rendering de Phong

Sumário

1) Introdução
2) Métodos de Rendering
   - Rendering de Superfície de Intensidade Constante
   - Rendering de Superfície de Gouraud
   - Rendering de Superfície de Phong
3) Programação OpenGL (Rendering)

Funções de Rendering de Superfície

- Podemos usar dois métodos de rendering da superfície:
  - (1) de intensidade constante (**flat**);
  - (2) e o modelo de Gouraud (**smoothing**)
  - Não existe suporte para o modelo de Phong!
    - Phong requer que as normais sejam passadas ao longo do rendering pipeline para o 'screen space'
    - OpenGL tonaliza os vértices em viewing coordinates e em seguida descarta as normais: impossível fazer Phong shading
- Para definir o método de rendering usamos:

```
glShadeModel(rendering_method);
```

- Onde rendering_method pode ser `GL_FLAT` ou `GL_SMOOTH`

Funções de Rendering de Superfície

- Para se **definir a normal** usamos:

```
glNormal3*(Nx, Ny, Nz);
```

- Com o sufixo dependendo do tipo de parâmetro `b`, `s`, `i`, `f` e `d`, ou com a adição de `v` caso o parâmetro seja um vetor
  - Valores *byte*, *short* e *integer* são convertidos para valores de ponto flutuante na faixa de −1.0 a 1.0
- A normal é um valor de estado da OpenGL e tem valor padrão igual a (0.0, 0.0, 1.0).

Funções de Rendering de Superfície

- Para rendering de **intensidade constante** definimos apenas uma normal para cada polígono:

```
glNormal3fv(normal_vector);
glBegin(GL_TRIANGLES);
   glVertex3fv(vertex1);
   glVertex3fv(vertex2);
   glVertex3fv(vertex3);
glEnd();
```

Funções de Rendering de Superfície

- Para o rendering de **Gouraud** uma normal deve ser definida para cada vértice:

```
glBegin(GL_TRIANGLES);
   glNormal3fv(normal_vector1);
   glVertex3fv(vertex1);

   glNormal3fv(normal_vector2);
   glVertex3fv(vertex2);

   glNormal3fv(normal_vector3);
   glVertex3fv(vertex3);
glEnd();
```

Funções de Rendering de Superfície

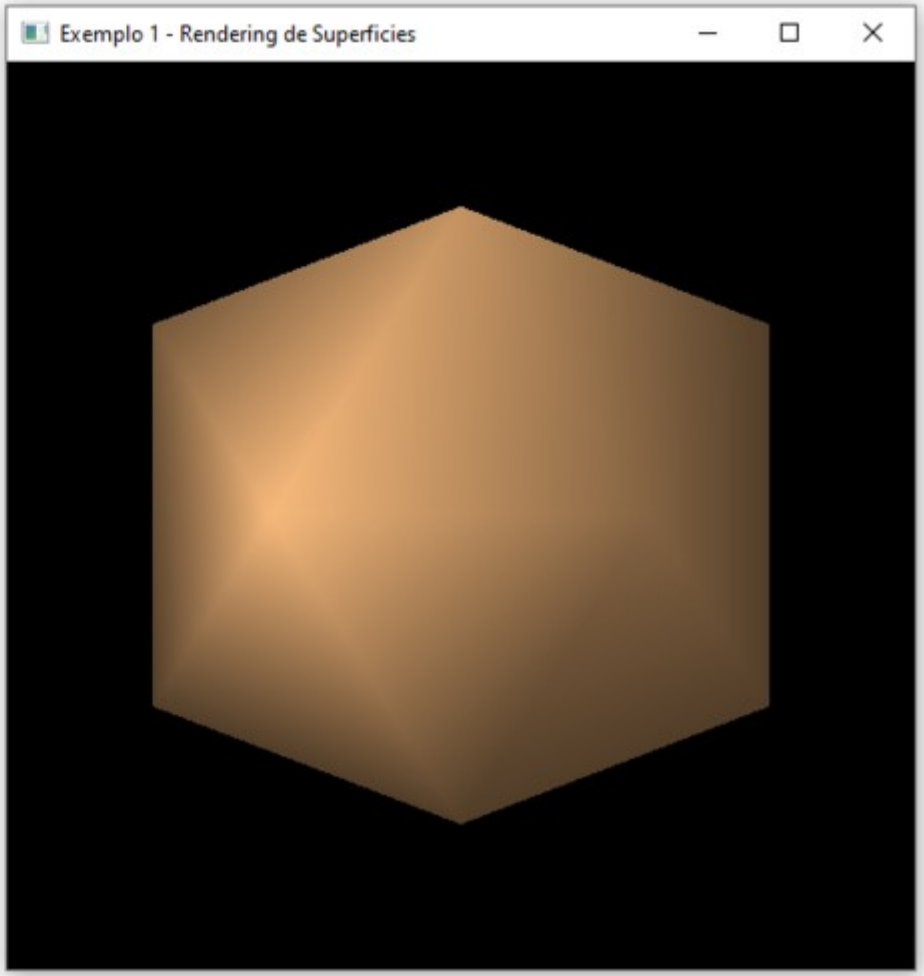
- Apesar dos vetores normais não precisarem ser especificados com tamanho unitário, fazendo isso **reduzimos** o custo computacional
- É possível solicitar a OpenGL **normalizar** qualquer vetor normal que **não seja unitário** chamando:

```
glEnable(GL_NORMALIZE);
```

- Esse comando **renormaliza** todas as normais às superfícies incluindo as que foram **modificadas** por transformações geométricas de escala e cisalhamento

Funções de Rendering de Superfície

- Com menos faces: flat / smooth



Funções de Rendering de Superfície

- Com mais faces: flat / smooth