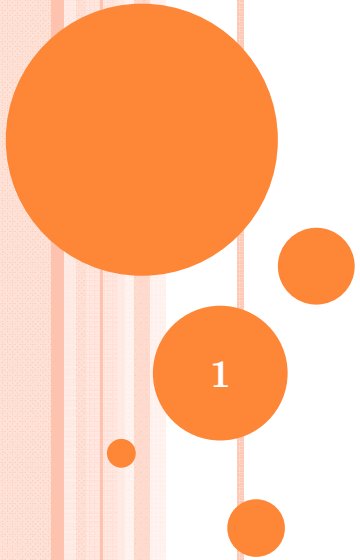


Introdução à OpenGL
Parte 3

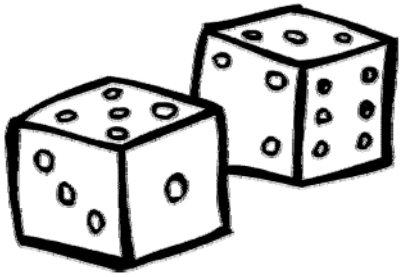
Pedro Henrique Bugatti

Roteiro

- Visualização
- Iluminação
- Tonalização
- Materiais
- Textura

VISUALIZAÇÃO

Em CG, gerar uma visão de uma cena tridimensional é similar ao ato de tirar uma fotografia



Alvo, Objetos, Atores...

+

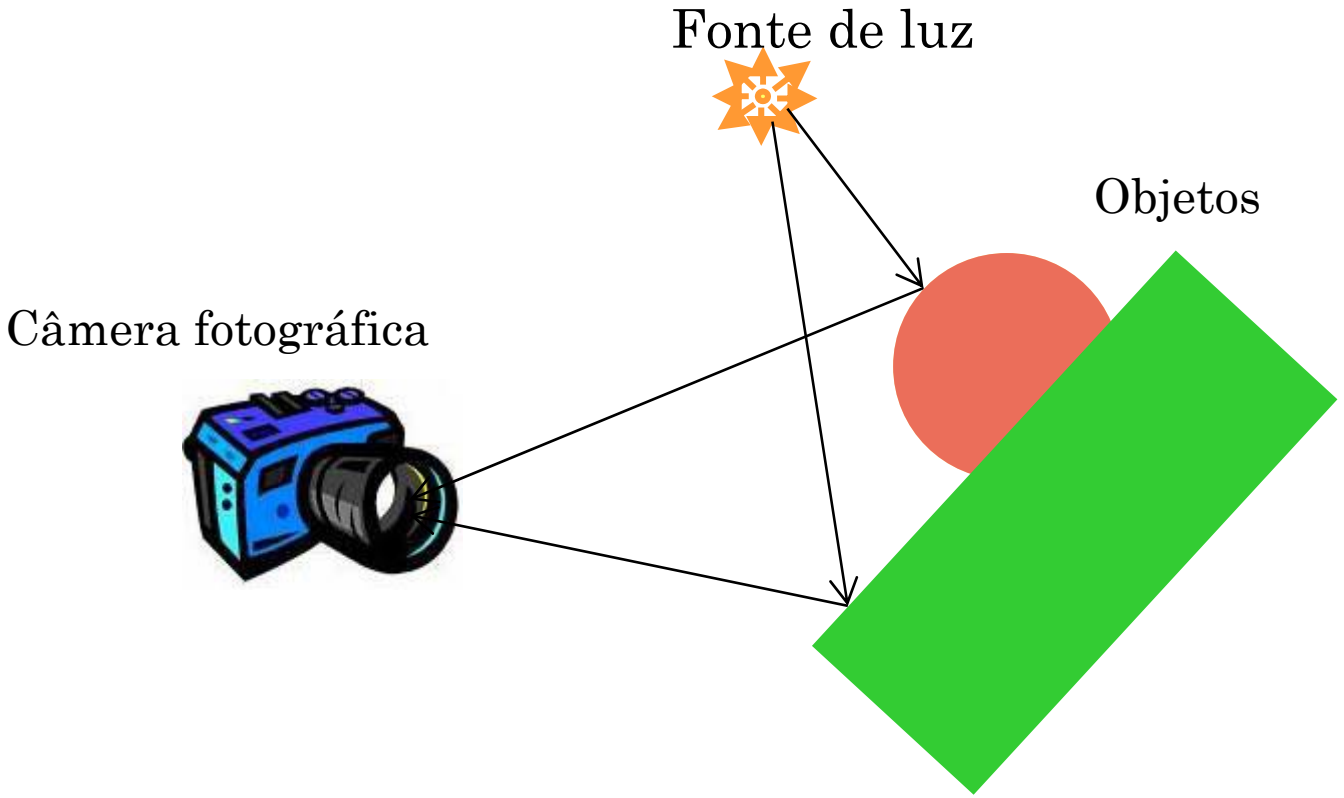
Câmera

+

Luz!

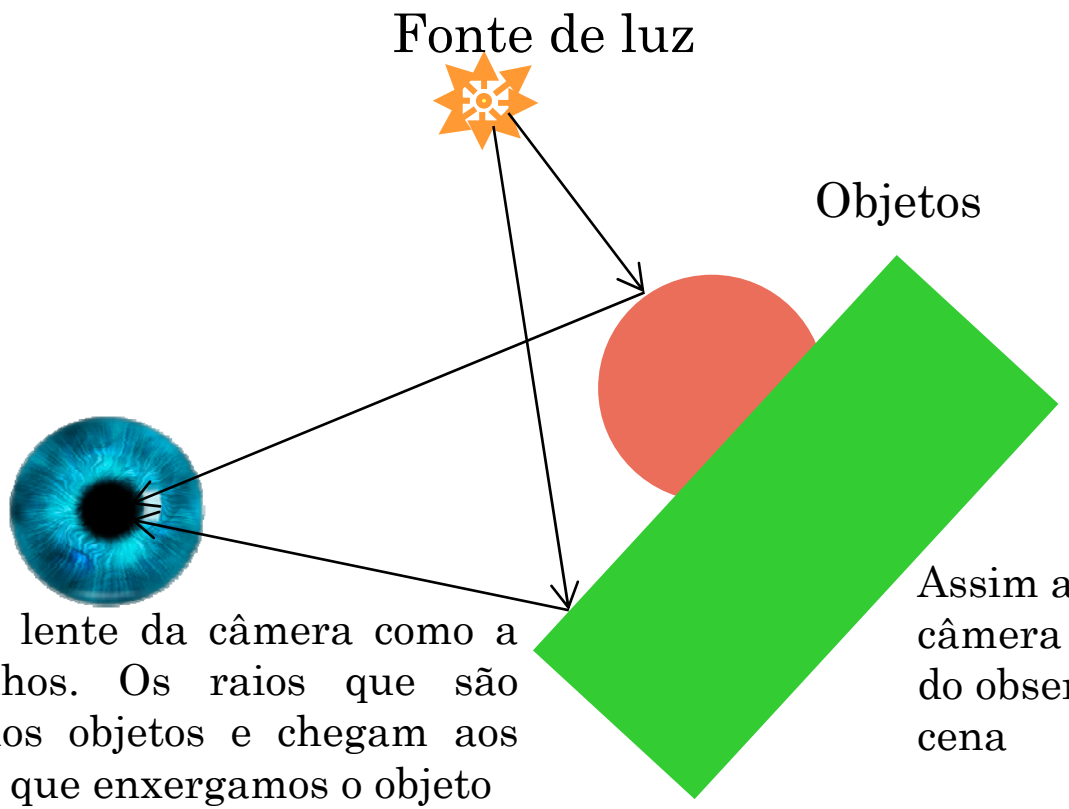
VISUALIZAÇÃO

Resumo do processo físico: os raios de luz atingem os objetos e são absorvidos (transmitidos) ou refletidos



VISUALIZAÇÃO

Resumo do processo físico: os raios de luz atingem os objetos e são absorvidos (transmitidos) ou refletidos



Podemos considerar a lente da câmera como a pupila de nossos olhos. Os raios que são refletidos/emitidos pelos objetos e chegam aos nossos olhos permitem que enxergamos o objeto

Assim a posição da câmera é a posição do observador da cena

VISUALIZAÇÃO

Passos para se tirar uma fotografia

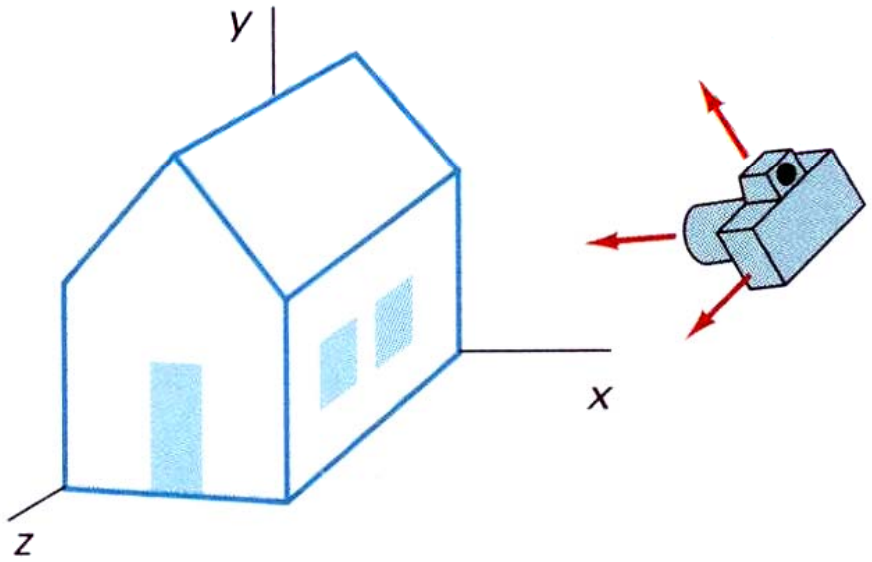
1 - Definir a posição da câmera;

2 - Apontar a câmera para uma direção ;

3 - Girar a câmera em torno da linha de visão;

4 - Definir zoom óptico (mecânico) e/ou zoom digital;

5 - Acionar o botão.



Em CG, emprega-se o conceito de câmeras virtuais

Definindo a Visualização

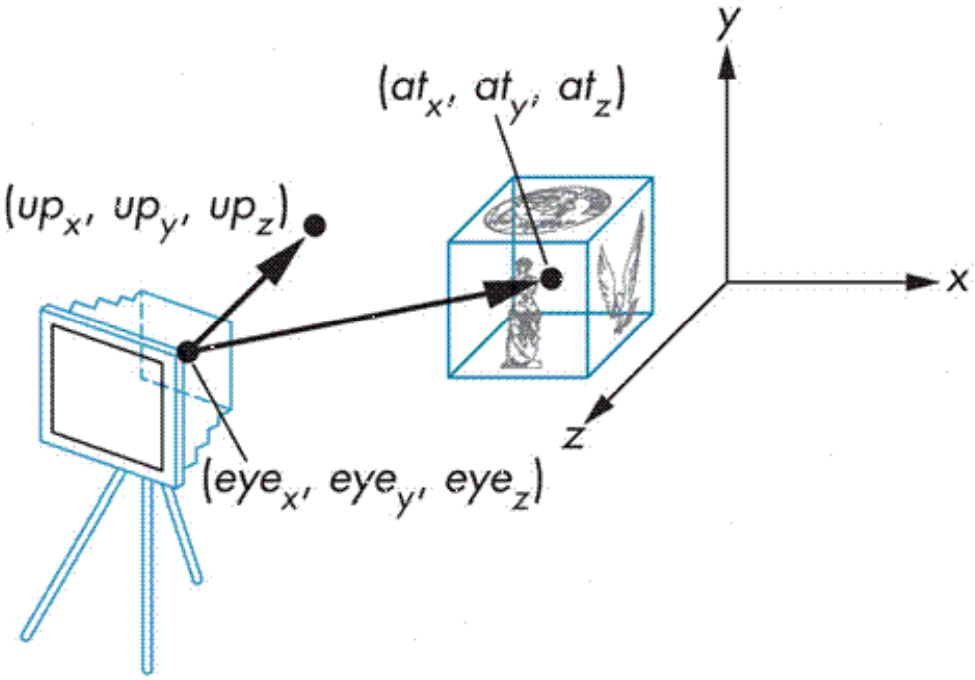
- Para visualizar cenas 3D é necessário especificar a posição e orientação da câmera. A GLU fornece a seguinte função:

gluLookAt(GLDouble eyex ,eyey, eyez, atx, aty, atz, upx, upy, upz)

eye: definir posição da câmera (observador);

at: ponto para onde observador está olhando;

up: coordenadas do vetor *up* (vetor que indica a "vertical" da câmera);



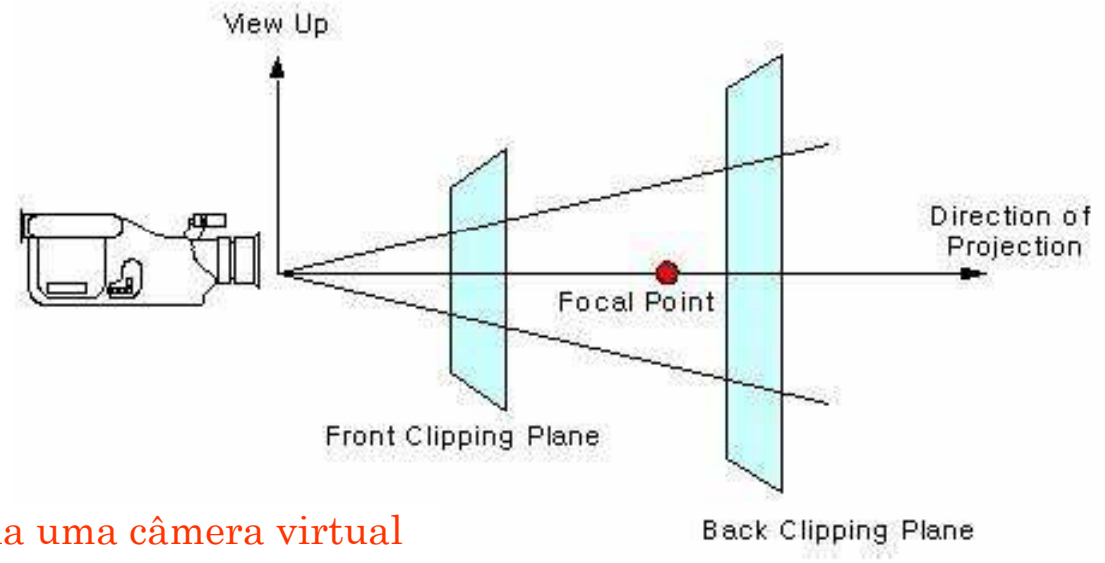
Definição de parâmetros difícil. Dica: estabelecer a posição da câmera na origem.

Definir cena paralela ao plano x,y (definindo vetor *view up* de mesma direção do eixo y).

VISUALIZAÇÃO - PROJEÇÕES

void gluPerspective(GLdouble fovy, aspect, zNear, zFar)

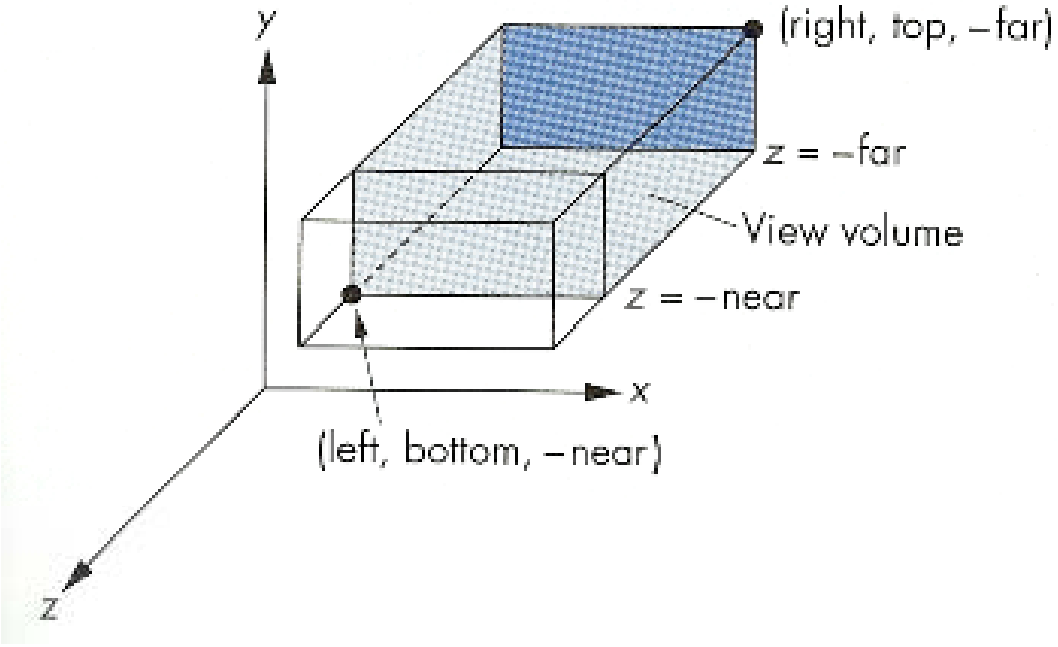
- Projeção perspectiva:
  - *fovy*: ângulo de abertura da câmera;
  - *aspect*: razão entre x (width) e y (height);
  - *zNear*: distância do observador ao plano de corte frontal;
  - *zFar*: distância do observador ao plano de corte trazeiro.



Atributos da uma câmera virtual

VISUALIZAÇÃO - PROJEÇÕES

- gluOrtho(GLdouble left,right,bottom,top, near, far)
  - Projeção paralela, onde os parâmetros definem os limites mínimos e máximos da janela de projeção em x , y e z, onde z negativo indica valor do lado oposto da posição do observador.

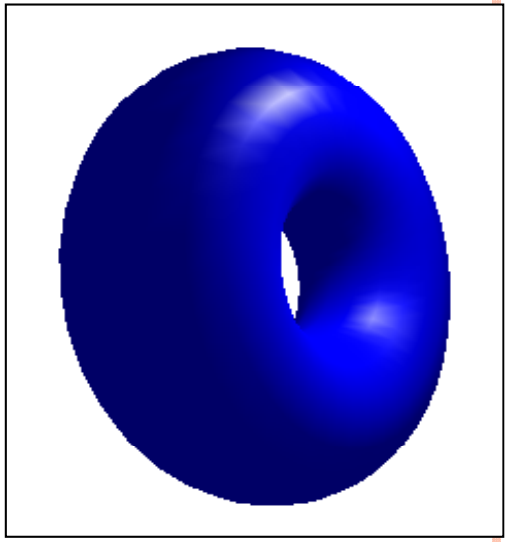


Visualização - Exemplo

- Explore o exemplo teapot3D.cpp
  - Modifique a posição do observador
    - parâmetros da função - gluLookAt()
  - Modifique os parâmetros da projeção
    - parâmetros da função - gluPerspective()

Iluminação - Sensação de Realismo

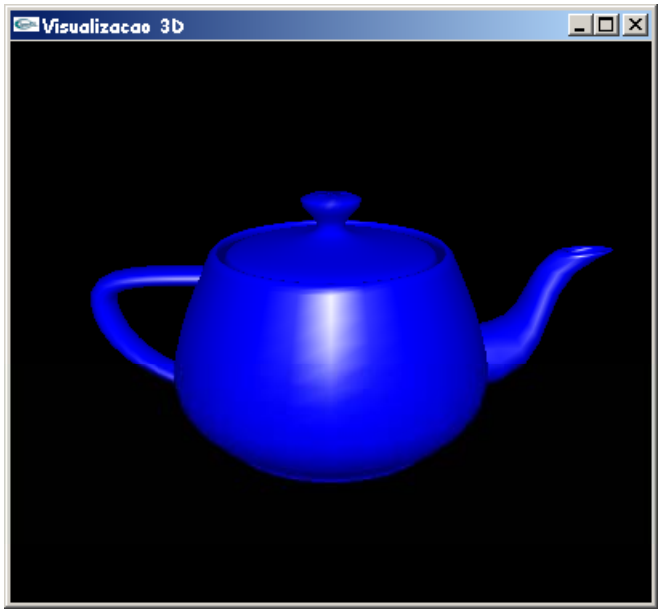
- Objetos são visíveis porque absorvem e refletem luz;
- A noção de realismo em uma cena 3D está ligada à idéia de iluminação;
- Para simular a realidade deve-se incluir fonte de luz na cena e descrever a interação dessas fontes de luz com os objetos da cena.



EXEMPLO – ILUMINAÇÃO

Cena A - Sem Iluminação

Cena A - Com Iluminação



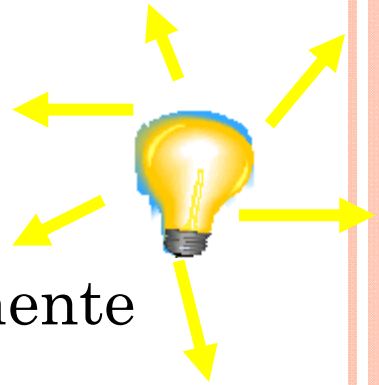
Como podemos observar a imagem sem iluminação não possui uma qualidade satisfatória. Ao adicionar uma ou mais fontes de luz obtém maior realismo.

ILUMINAÇÃO

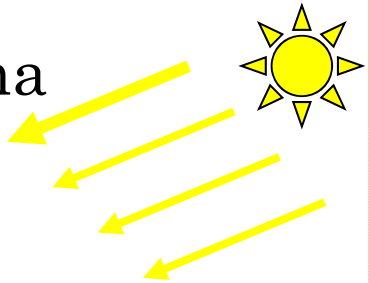
- Iluminação é a simulação de como os objetos refletem luz. A iluminação de um objeto em geral depende:
  - Material e cor do objeto;
  - Posição e cor da fonte de luz;
  - Parâmetros de iluminação de luz ambiente.

Iluminação – Tipos de Luz

- Tipos de luz:



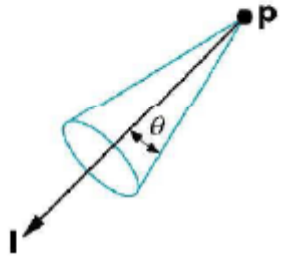
  - Pontual (local): raios emanam uniformemente em todas as direções;



  - Direcional: (infinita): raios em apenas uma direção;
  - Spotlight: raios emitidos na forma de um cone apontando para uma determinada direção.

Iluminação – Tipos de Luz

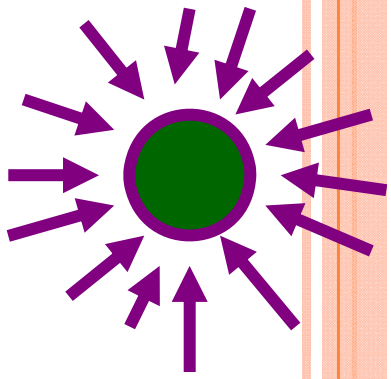
- Spotlight:
  - Direção (vector);
  - Cutoff (ângulo de abertura do cone);
  - Atenuação com o ângulo.



Iluminação - Modelos de Reflexão

- Um modelo de reflexão descreve a interação dos raios de luz com a superfície e a natureza da fonte de luz.
  - Luz Ambiente
  - Difusa
  - Especular

ILUMINAÇÃO - MODELOS DE REFLEXÃO



- Reflexão da Luz Ambiente
- A luz ambiente origina-se da reflexão da fonte de luz em todas as superfícies da cena;
- Ela permite a visibilidade de superfícies que não recebendo diretamente raios de luz;
- Por definição, ela depende apenas da cor do objeto e gera uma iluminação constante para todas as faces do objeto

Iluminação - Modelos de Reflexão

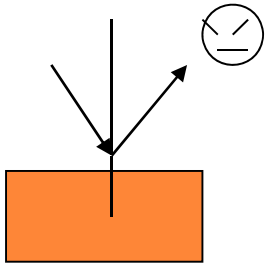
- Reflexão Difusa
- Ocorre na maioria dos objetos opacos;
- O objeto absorve parte da luz que recebe e reflete igualmente em todas as direções, porém com intensidade menor do que a recebida.

Iluminação - Modelos de Reflexão

- Reflexão Especular

- A luz é refletida em uma direção. O raio deixa a superfície com o mesmo ângulo do raio incidente e a normal a superfície.

- Assim, a medida em que o observador altera sua posição, altera-se a intensidade de brilho que ele visualiza.



ILUMINAÇÃO – TIPOS DE LUZ

- Funções OpenGL utilizadas na especificação das fontes de luz (Define as propriedades de uma fonte de luz):
  - void glLightfv(GLenum light, GLenum pname, TYPE * param)
- light: indica a fonte de luz desejada (GL_LIGHT0 a GL_LIGHT7)
- pname: define a característica da fonte de luz que se deseja configurar e pode receber valores GL_AMBIENT, GL_SPECULAR, GL_POSITION, etc...
- param: vetor do tipo GLfloat, determina o valor da característica definida em pname.

ILUMINAÇÃO – TIPOS DE LUZ

- Ativa o modelo de iluminação e o configura:
  - void glLightModelfv(GLenum pname, TYPE * param)
- pname: define a característica a ser configurada (pode ser GL_LIGHT_MODEL_AMBIENT, GL_LIGHT_MODEL_LOCAL_VIEWER entre outros)
- param: determina o valor de da característica definida em pname.

Iluminação – Luz Exemplo

Definição da cor componente ambiente, da cor da componente difusa, da cor da componente especular, e da posição da luz.

```
//RGB e alpha (transparência)
GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_specular[]= { 1.0, 1.0, 1.0, 1.0 };
GLloat light_position[]= { 2.0, 5.0, 5.0, 0.0 };


//definindo todos os componentes da luz 0)
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR,light_specular);
glLightfv(GL_LIGHT0, GL_POSITION,light_position);
```

Iluminação – Luz Exemplo

Define a cor da luz ambiente:

```
GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
```

Ativa o uso de uma luz ambiente na cena:

```
glLightModel(GL_LIGHT_MODEL_AMBIENT,
  light_ambient)
```

ILUMINAÇÃO

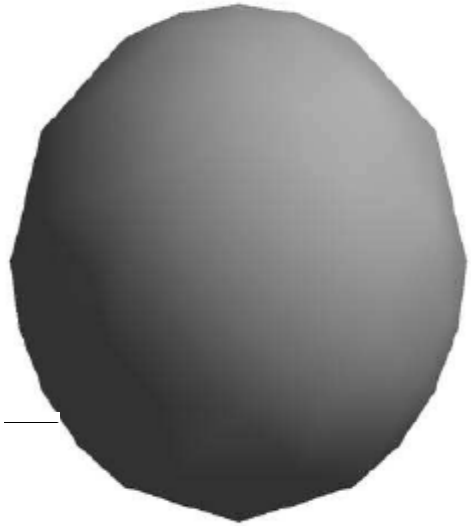
- Explore o exemplo teapot3d-iluminacao.cpp
  - Acrescente novas luzes de diferentes tipos
  - .....

TONALIZAÇÃO

- A linguagem OpenGL oferece diretamente dois modelos de tonalização:
- Modelo Flat: a cor de uma face é aquela calculada para o primeiro vértice da face;
- Modelo Gouraud: a cor calculada nos vértices é interpolada para a obtenção da cor no restante da face;

FLAT

GOURAD



TONALIZAÇÃO

- A função para selecionar a tonalização é:
  - void glShadeModel(GLenum *mode*)

- Onde *mode* é
  - GL_FLAT (modelo Flat)
  - GL_SMOOTH (modelo Gouraud)

TONALIZAÇÃO

- Explore o exemplo teapot3d-iluminacao.cpp
  - Modifique os para parâmetros de glShadeModel()
  - .....

MATERIAIS

- As superfícies dos objetos em OpenGL são compostas de materiais que podem emitir, refletir luz em todas as direções ou em uma única direção

- Os materiais, dependendo de quanto refletem de luz podem ser:
  - Especular: 
  - Difuso: 

MATERIAIS

- Função usada para definir propriedades do material:

```
glMaterialfv(GLenum face, GLenum pname, TYPE * param)
```

estabelece os parâmetros do material que serão usados pelo modelo de iluminação

- Face: especifica quais faces do objeto terão as propriedades que serão especificadas GL_FRONT, GL_BACK e GL_FRONT_AND_BACK;

- Pname: propriedade do material a ser especificada (GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR)

- Param: determina o valor de da característica definida em pname.

Materiais

- Exemplo
  - cor da luz refletida especularmente

```
GLfloat specular[]= { 1.0, 1.0, 1.0, 1.0 };

glMaterialfv(GL_FRONT,GL_SPECULAR,specular);
```

  - concentração do brilho

```
glMateriali(GL_FRONT, GL_SHININESS, 60);
```

Materiais

- Explore o exemplo teapot3d-iluminacao.cpp
  - Modifique os parâmetros da função glMaterialfv()
  - .....

Obtenção do Realismo

- Os passos necessários obter realismo em uma cena são:
  - Habilitar e selecionar o modo de tonalização;
  - Especificar as propriedades dos materiais;
  - Especificar as luzes.

TEXTURA

- Em OpenGL, obtém-se o efeito de textura através da "aplicação" de uma imagem às faces do objeto.
- A relação entre os vértices das faces e os pontos na imagem de textura deve ser indicada explicitamente.
- Utiliza-se, portanto, uma técnica conhecida como "mapeamento de textura" ou "*texture mapping*".

TEXTURA

- Um pixel de textura é denominado *texel*
- O mapeamento de textura é habilitado e desabilitado explicitamente através da função
  - **glEnable("*texture mapping mode*") e glDisable().**
- O "*texture mapping mode*" pode ser:
  - GL_TEXTURE_1D
  - GL_TEXTURE_2D
  - GL_TEXTURE_3D_EXT ,dependendo do tipo da função **glTexImage*D()** usado

TEXTURA

- Os passos para gerar apresentar um objeto com a aplicação de uma textura, na forma de uma imagem "colada" às suas faces são os seguintes:

1. Identificação de um objeto textura (através da criação de um nome) para representar a imagem de textura

   - **void glGenTextures(GLsize n, GLuint *textnames);**
   - Retorna **n nomes não usados como nomes de textura no *array textnames***

TEXTURA

2. Criação de um objeto de textura propriamente dita, associando-a a um nome textura criado anteriormente.

   - **void glBindTexture(GLenum *target*, GLuint *textureName*);**
   - Associa o nome indicado em *textureName* a um objeto de textura do tipo especificado por *target,* criado com valores default para as propriedades e para a imagem.

3. Especificação da imagem para o objeto de textura criado anteriormente

   - **void glTextImage1D(GLenum target, GLint level, Glint internalformat, GLsizei width, Glint border, GLenum format, GLenum type, const GLvoid *pixels)**

   - **void glTextImage2D(GLenum target, GLint level, GLint internalformat, GLsizei width, Glsizei height, GLint border, GLenum format, Glenum type, const GLvoid *pixels);**

TEXTURA - PASSOS

- Cria uma textura 1D ou 2D conforme o caso
  - (*target = GL_TEXTURE_1D, GL_TEXTURE_2D)*
  - ***Level dever ser 0 para uma única imagem de textura.***

- Será diferente de zero quando usando a técnica de *mip-mapping* para obter níveis de detalhe diferente nas texturas.

- ***Internalformat*** descreve a representação interna da textura *(GL_RGBA, GL_ALPHA,* GL_LUMINANCE, etc).

- Tamanho especificado por **width ou por width e height.**

- ***Border*** indica se desejamos uma borda de 0 ou mais pixels.

- ***Fomat*** e ***type*** *correspondem aos mesmos parâmetros vistos para imagem (GL_RGBA, GL_RED,* etc, e GL_INT, GL_SHORT_INT, etc, respectivamente).

- E ***pixels*** *corresponde ao array (unidimensional ou bidimensional, conforme o caso).*

TEXTURA - PASSOS

4. Indicação de eventual mudança de escala que a imagem de textura deve sofrer.

   - **GLint gluScaleImage(GLenum format, GLsizei widthin, Glsizei heightin, GLenum typein, const void *datain, Glsizei widthout, GLsizei heightout, GLenum typeout, void *dataout)**

- Esta função permite que se faça uma escala na imagem de textura, gravando-a novamente

TEXTURA - PASSOS

5. Especificação da forma de mapeamento da textura ao objeto.
   - **glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_MODE, modo);**

- Modo pode ser:
  - GL_MODULATE (default), que modula (multiplica) a cor do pixel obtido do objeto com as informações da textura;
  - GL_DECAL, onde cor atual do pixel obtido do objeto não influenciam na imagem final (ou seja, a imagem de textura é colada), exceto pelo parâmetro *alpha do* objeto que é mantido;
  - GL_BLEND, que usa a cor da textura e faz *blending* com a cor do objeto e uma cor especificada num atributo apropriado;
  - GL-REPLACE, quando o valor de *alpha do pixel é apenas substituído pelo alpha da* textura.

TEXTURA - PASSOS

6. Habilitação do mapeamento de textura:

   - **glEnable(GL_TEXTURE_1D); ou**
   - **glEnable(GL_TEXTURE_2D);**

TEXTURA - PASSOS

7. Especificação da relação entre coordenadas de vértices e coordenadas da imagem de textura.
   - **void glTextCoord{1234}{sifd}(Type coords);**
   - **void glTextCoord{1234}{sifd}v(Type *coords);**

- Uma textura 2D é tratada como um quadrado, com coordenadas variando de (0,0) a (1,1). Veja o exemplo abaixo:

```
// near face
glBegin (GL_POLYGON);
glTexCoord2f (0.0f,0.0f); /* lower left corner of image */
glVertex3f (-5.0f, -5.0f, 5.0f);
glTexCoord2f (1.0f, 0.0f); /* lower right corner of image */
glVertex3f (5.0f, -5.0f, 5.0f);
glTexCoord2f (1.0f, 1.0f); /* upper right corner of image */
glVertex3f (5.0f, 5.0f, 5.0f);
glTexCoord2f (0.0f, 1.0f); /* upper left corner of image */
glVertex3f (-5.0f, 5.0f, 5.0f);
glEnd ();
```

- É possível especificar coordenadas de textura fora dos limites 0 e 1. Mas, nesse caso, é necessário indicar de as texturas serão "podadas", mantendo os valores dos texels ou se haverá repetição do padrão. Isto é obtido com a inicialização de parâmetros.

- **glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);**
- **glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);**

TEXTURA - PASSOS

8. Especificação de como a textura será interpolada quando houver diferença de dimensão entre a face e a imagem de textura.

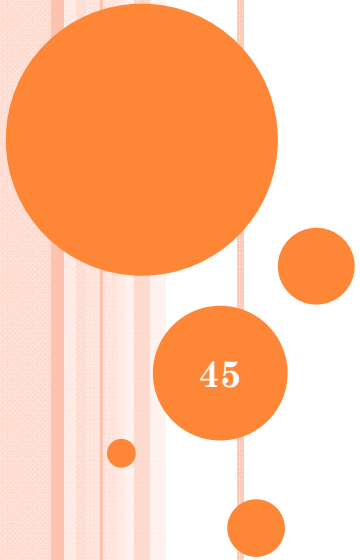
- OpenGL usa filtros para interpolar os texels no cálculo da textura de um pixel:
  - GL_TEXTURE_MIN_FILTER, quando o pixel representa uma área maior que um texel;
  - GL_TEXTURE_MAG_FILTER, quando o pixel representa uma área menor que um texel.
  - **glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FLTER, filter);**
  - onde *filter* pode ser um dos filtros usuais como GL_NEAREST (usa o texel cujas coordenadas sejam mais próximas do centro do pixel a ser exibido) e GL_LINEAR (faz uma média de um conjunto de 2x2 texels).

TEXTURA

- Exemplos a serem explorados e utilizados como templates para outras implementações:
  - Ármario-textura
  - Exemplo-tartaruga
  - Tabuleiro

Conclusões

- Nas aulas apresentadas foram explicitados os conceitos primordiais, características e recursos da OpenGL.
- No entanto a OpenGL possui inúmeros outros recursos, cabe a vocês explorarem essa extensa e poderosa biblioteca.
- Explorem a imaginação e combinem os conceitos aprendidos.



Introdução à OpenGL
Parte 3

Pedro Henrique Bugatti