Exercício

Inicialmente faça o *download* do arquivo iluminacao.cpp.

Substitua a chamada para a função *glutWireTeapot(50.0f)* (usada para desenhar o *wire-frame* de um *teapot*) por *glutSolidTeapot(50.0f)*, compile e execute o programa. Olhando a imagem gerada se observa que a mesma parece 2D. Isto ocorre porque a versão *Solid* deve ser usada somente quando se está trabalhando com iluminação.

Para utilizar a iluminação em OpenGL, passe mais um parâmetro para as funções *glClear* e *glutInitDisplayMode*, da seguinte maneira:

```
 :
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 :
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
 :
```

Estas alterações definem que o o programa atual necessita de um *buffer* de profundidade, também chamado de *z-buffer*, e que ele deve ser inicializado antes de cada redesenho.

Depois, **substitua** a função *Inicializa()* do seu código pela função *Inicializa()* apresentada abaixo.

```
// Inicialização
void Inicializa(void)
 {
       // Define a cor de fundo da janela de visualização como branca
       glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

       // Habilita a definição da cor do material a partir da cor corrente
       glEnable(GL_COLOR_MATERIAL);
       //Habilita o uso de iluminação
       glEnable(GL_LIGHTING);
       // Habilita a luz de número 0
       glEnable(GL_LIGHT0);
       // Habilita o depth-buffering
       glEnable(GL_DEPTH_TEST);

       // Habilita o modelo de colorização de Gouraud
       glShadeModel(GL_SMOOTH);

       // Inicializa a variável que especifica o ângulo da projeção
       // perspectiva
       angle=50;
       // Inicializa as variáveis usadas para alterar a posição do
       // observador virtual
       rotX = 30;
       rotY = 0;
       obsZ = 180;
```

```
}
```

Na seqüência, inclua a função abaixo antes da função *Desenha()*.

```
// Função responsável pela especificação dos parâmetros de iluminação
void DefineIluminacao (void)
{
      GLfloat luzAmbiente[4]={0.2,0.2,0.2,1.0};
      GLfloat luzDifusa[4]={0.7,0.7,0.7,1.0}; // "cor"
      GLfloat luzEspecular[4]={1.0, 1.0, 1.0, 1.0};// "brilho"
      GLfloat posicaoLuz[4]={0.0, 50.0, 50.0, 1.0};

      // Capacidade de brilho do material
      GLfloat especularidade[4]={1.0,1.0,1.0,1.0};
      GLint especMaterial = 60;

      // Define a refletância do material
      glMaterialfv(GL_FRONT,GL_SPECULAR, especularidade);
      // Define a concentração do brilho
      glMateriali(GL_FRONT,GL_SHININESS,especMaterial);

      // Ativa o uso da luz ambiente
      glLightModelfv(GL_LIGHT_MODEL_AMBIENT, luzAmbiente);

      // Define os parâmetros da luz de número 0
      glLightfv(GL_LIGHT0, GL_AMBIENT, luzAmbiente);
      glLightfv(GL_LIGHT0, GL_DIFFUSE, luzDifusa );
      glLightfv(GL_LIGHT0, GL_SPECULAR, luzEspecular );
      glLightfv(GL_LIGHT0, GL_POSITION, posicaoLuz );
}
```

Em seguida, chame a função *DefineIluminacao()* na função *Desenha()*, logo após à chamada da função *glClear()*. Também inclua uma chamada para esta função dentro da função *PosicionaObservador()*, como exemplificado abaixo. Finalmente, compile e execute o programa para ver a nova imagem gerada.

```
void Desenha(void)
{
      // Limpa a janela de visualização com a cor
      // de fundo definida previamente
      glClear(GL_COLOR_BUFFER_BIT);

      // Chama a função que especifica os parâmetros de iluminação
      DefineIluminacao(); // INCLUA ESTA LINHA
       ...

// Função usada para especificar a posição do observador virtual
void PosicionaObservador(void)
{
      // Especifica sistema de coordenadas do modelo
      glMatrixMode(GL_MODELVIEW);
      // Inicializa sistema de coordenadas do modelo
      glLoadIdentity();
      DefineIluminacao(); // INCLUA ESTA LINHA
       ...
```

Agora vamos estudar algumas funções chamadas nos códigos apresentados acima. Primeiro, vamos considerar a função *glShadeModel()*, que é usada para especificar a técnica de tonalização desejada. Experimente trocar o parâmetro GL_SMOOTH por GL_FLAT e responda a questão 1.

Questão 1) O que aconteceu quando você trocou o parâmetro passado para a função*glShadeModel*? Explique com as suas palavras para que serve esta função.

Em OpenGL a cor de uma fonte de luz é caracterizada pela quantidade de vermelho (R), verde (G) e azul (B) que ela emite, e o material de uma superfície é caracterizado pela porcentagem dos componentes R, G e B que chegam e são refletidos em várias direções. No modelo de iluminação a luz em uma cena vem de várias fontes de luz que podem ser "ligadas" ou "desligadas" individualmente. A luz pode vir de uma direção ou posição (por exemplo, uma lâmpada) ou como resultado de várias reflexões (luz ambiente - não é possível determinar de onde ela vem, mas ela desaparece quando a fonte de luz é desligada). Também pode haver uma luz ambiente genérica em uma cena que não vem de uma fonte de luz específica.

No modelo de iluminação OpenGL a fonte de luz tem efeito somente quando existem superfícies que absorvem e refletem luz. Assume-se que cada superfície é composta de um material com várias propriedades. O material pode emitir luz, refletir parte da incidente luz em todas as direções, ou refletir uma parte da luz incidente numa única direção, tal com um espelho. Então, OpenGL considera que a luz é dividida em quatro componentes independentes (que são colocados juntos).

- **Ambiente**: resultado da luz refletida no ambiente; é a luz que vem de todas as direções;
- **Difusa**: luz que vem de uma direção, atinge a superfície e é refletida em todas as direções; assim, apresenta a mesma intensidade independente do local no qual o observador está posicionado;
- **Especular**: luz que vem de uma direção e tende a ser refletida numa única outra direção;
- **Emissiva**: simula a luz que se origina de um objeto; a cor emissiva de uma superfície adiciona intensidade luminosa ao objeto, mas não é afetada por qualquer fonte de luz e também não introduz luz adicional na cena.

A cor do material de um objeto depende do percentual de luz vermelha, verde e azul incidente que este reflete. Assim como as luzes, o material possui componentes ambiente, difusa e especular diferentes, que determinam como será a luz refletida. Isto é combinado com as propriedades das fontes de luz, de tal maneira que a reflexão ambiente e difusa definem a cor básica do material. A cor da luz especular é geralmente cinza ou branca.
Crie três variáveis globais (r, g e b) do tipo *float* que devem ser inicializadas com 1.0 na função *Inicializa()*. Estas variáveis devem ser usadas na criação do vetor *luzEspecular* na função *DefineIluminacao()*, como mostrado abaixo:

GLfloat luzEspecular[4]={1.0, 1.0, 1.0, 1.0};

Depois, inclua o tratamento de eventos adequado para que os valores destas variáveis possam ser alterados para valores entre 0.0 e 1.0. Compile e execute o programa para alterar os valores das variáveis criadas.

Questão 2) O que acontece quando os valores destas variáveis são alterados? Porque você achaque acontecem estas alterações.

Os componentes de cor especificados para a luz possuem um significado diferente dos componentes de cor especificados para os materiais. Para a luz, os números correspondem a uma porcentagem da intensidade total para cada cor. Se os valores R, G e B para a cor da luz são 1, a luz é branca com o maior brilho possível. Se os valores são 0.5 a cor ainda é branca, mas possui metade da intensidade, por isso parece cinza. Se R=G=1 e B=0, a luz parece amarela.

Para os materiais, os números correspondem às proporções refletidas destas cores. Se R=1, G=0.5 e B=0 para um material, este material reflete toda luz vermelha incidente, metade da luz verde e nada da luz azul. Assim, simplificadamente, a luz que chega no observador é dada por (LR.MR, LG.MG, LB.MB), onde (LR, LG, LB) são os componentes da luz e (MR, MG, MB) os componentes do material [Woo 1999].

Questão 3) Altere o programa para que a variável *especMaterial* seja uma variável global. Depois, inicialize o valor desta variável com 60 na função inicializa e adicione o código necessário paraque esta variável tenha o seu valor aumentado de 5 unidades cada vez que o usuário pressiona atecla F1, e diminuído de 5 unidades cada vez que o usuário pressiona a tecla F2. Cuide para queela nunca seja menor que 0 e nem maior que 160. Compile e execute o programa alterando ovalor desta variável até chegar a valores muito baixos (por exemplo, 10) e muito altos (porexemplo, 100). Explique com as suas palavras qual a função desta variável.

Agora coloque as linhas de código que aparecem abaixo em comentário:

```
// GLfloat especularidade[4]={1.0,1.0,1.0,1.0};
 ...
// glMaterialfv(GL_FRONT,GL_SPECULAR, especularidade);
 ...
// glMateriali(GL_FRONT,GL_SHININESS,especMaterial);
 ...
// glLightfv(GL_LIGHT0, GL_SPECULAR, luzEspecular );
```

Questão 4) O que aconteceu com a imagem final quando as linhas de código acima foramcolocadas em comentário?

Altere o valor de inicialização da variável *luzAmbiente* para *{0.1,0.1,0.1,1.0}*, compile e execute o programa. Altere o valor desta mesma variável para *{0.4,0.4,0.4,1.0}*, compile e execute o programa. Agora altere o valor desta mesma variável para *{0.8,0.8,0.8,1.0}*, compile e execute o programa.

Questão 5) O que acontece quando o valor desta variável é alterado? Explique porque você achaque acontecem estas alterações.

Agora modifique o programa para incluir mais uma fonte de luz que pode ter sua posição alterada. Para isto, crie três variáveis (x, y e z) do tipo *float* para serem usadas para alterar a posição da fonte de luz, inclua mais uma fonte de luz na função *DefineIluminacao()* e habilite a utilização desta nova fonte de luz na função *Inicializa()*. Adicione também o tratamento de eventos necessário para que as três variáveis criadas tenham seus valores modificados. Compile e execute o programa mudando a posição desta fonte de luz para ver o efeito na imagem final.

Conforme estudado em aula, a maioria dos objetos em Computação Gráfica são representados através de uma malha de polígonos. Para exemplificar a utilização de uma malha de polígonos, elimine a chamada para a função *glutSolidTeapot(50.0f);* e coloque o código abaixo no seu lugar.

```
// Desenha um cubo
glBegin(GL_POLYGON); // Face posterior
      glNormal3f(0,0,1); // Normal da face
      glVertex3f(50.0, 50.0, 50.0);
      glVertex3f(-50.0, 50.0, 50.0);
      glVertex3f(-50.0, -50.0, 50.0);
      glVertex3f(50.0, -50.0, 50.0);
glEnd();

glBegin(GL_POLYGON); // Face frontal
      glNormal3f(0,0,-1); // Normal da face
      glVertex3f(50.0, 50.0, -50.0);
      glVertex3f(50.0, -50.0, -50.0);
      glVertex3f(-50.0, -50.0, -50.0);
      glVertex3f(-50.0, 50.0, -50.0);
glEnd();

glBegin(GL_POLYGON); // Face lateral esquerda
      glNormal3f(-1,0,0); // Normal da face
      glVertex3f(-50.0, 50.0, 50.0);
      glVertex3f(-50.0, 50.0, -50.0);
      glVertex3f(-50.0, -50.0, -50.0);
      glVertex3f(-50.0, -50.0, 50.0);
glEnd();
```

```
glBegin(GL_POLYGON); // Face lateral direita
      glNormal3f(1,0,0); // Normal da face
      glVertex3f(50.0, 50.0, 50.0);
      glVertex3f(50.0, -50.0, 50.0);
      glVertex3f(50.0, -50.0, -50.0);
      glVertex3f(50.0, 50.0, -50.0);
glEnd();

glBegin(GL_POLYGON); // Face superior
      glNormal3f(0,1,0); // Normal da face
      glVertex3f(-50.0, 50.0, -50.0);
      glVertex3f(-50.0, 50.0, 50.0);
      glVertex3f(50.0, 50.0, 50.0);
      glVertex3f(50.0, 50.0, -50.0);
glEnd();

glBegin(GL_POLYGON); // Face inferior
      glNormal3f(0,-1,0); // Normal da face
      glVertex3f(-50.0, -50.0, -50.0);
      glVertex3f(50.0, -50.0, -50.0);
      glVertex3f(50.0, -50.0, 50.0);
      glVertex3f(-50.0, -50.0, 50.0);
glEnd();
```

Questão 6) Explique o que deveria ser feito para exibir o cubo se o mesmo fosse modelado comouma malha de triângulos.