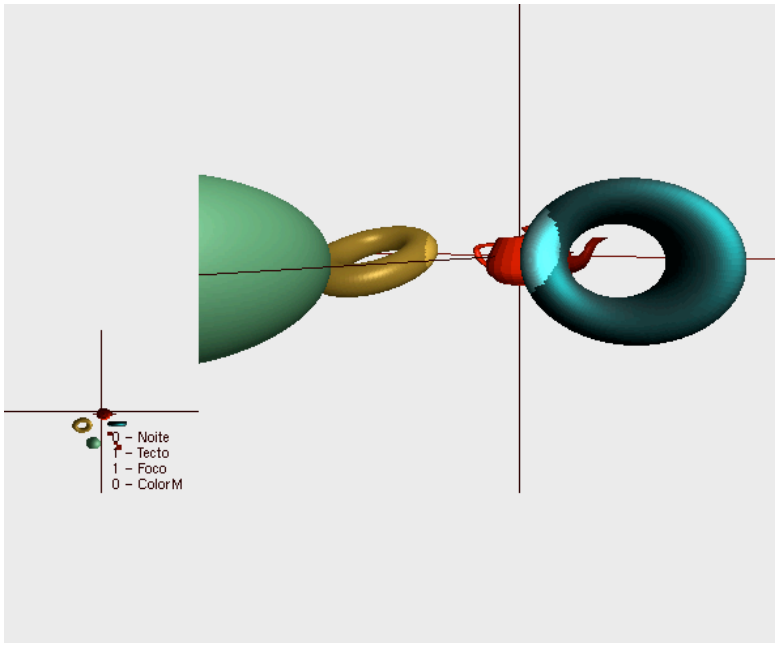
Trabalho **4**

Visita Virtual



Computação Gráfica

Aulas práticas

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

JHenriques/PCarvalho

Março de 2011

Índice

Trabalho para Avaliação

Período de Realização:	2 semanas
Data máxima para entrega:	Alunos da TP1: 26 de Abril (24h) Alunos da Tp2-TP7:28 de Abril (24h)
Local e formato de Entrega:	Cacifo do Prof. Paulo de Carvalho, em **CD** com o código fonte e o projecto compilado (não necessita de relatório).
Defesa:	obrigatória; a defesa será realizada nas aulas PL; inscrição obrigatória

1. Introdução

Este trabalho tem como objectivo principal o estudo e a aplicação de modelos de iluminação. Além disso pretende-se continuar a estudar a aplicação prática de projecções de forma a simular uma "visita Virtual".

Os objectos possíveis de visualizar/visitar podem ser implementados através de modelos pré-definidos em OpenGL (Cube, TeaPot, Sphere, ...). Em concreto pretende-se dispor, de uma forma mais ou menos aleatória, quatro ou cinco dos modelos acima referidos e simular uma "Visita Virtual", a ser efectuada por um observador.

A visita poderá ser realizada de dia ou de noite e além da visualização local dos objectos deve ser sempre disponibilizado ao observador a sua posição relativamente à cena (mapa).

1.1 Requisitos

Modelos (objectos) a visitar

O programa a implementar deve poder:

- Definir vários (mínimo de 5) modelos e posicioná-los na cena
  - Recorrer ao uso de glutWireCube( ), glutTeaPot( ), ...
  - Os diversos modelos não se devem intersectar
  - Os diversos modelos devem rodar em torno de um eixo perpendicular ao plano XZ
- Subdividir a janela de visualização em duas zonas distintas
  - A da esquerda, onde é visualizado um mapa. Conseguido à custa de uma projecção ortogonal, com o observador situado "em cima" da cena.
  - A da direita, onde é feia a visita propriamente dita aos objectos em exposição.

Iluminação

Assume-se que a visita pode ser feita de dia ou de noite. Se for de dia a iluminação existente será apenas ambiente. Se for de noite, e porque a luz existente (ambiente) é quase nula, o observador de forma a visualizar os objectos, terá de recorrer a uma de duas formas: ligando uma luz existente no tecto da sala ou utilizar uma lanterna (ou as duas em simultâneo).

Materiais

Assume-se que os materiais podem ser definidos através das suas características (componentes ambiente, especular, difusa).

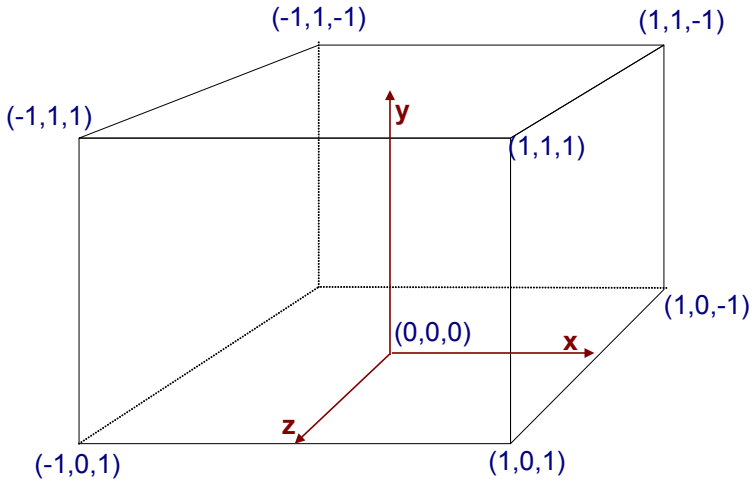
Em OpenGl, tal tarefa pode ser simplificada recorrendo ao uso de **glColorMaterial**.

Acções

O visitante deve poder:

- Percorrer a exposição dos modelos
  - Teclas '↓', ' ↓' para a frente e para trás
  - Teclas '←', ' →' para orientação, isto é, para a esquerda e para a direita.
- Comutar entre materiais definidos por 'cores' ou 'propriedades'
  - Tecla: '**M**' - liga/desliga alternadamente
- Activar o modo de visita noite/dia
  - Tecla '**N**' - noite/dia alternadamente
- Ligar/desligar a luz existente no tecto da sala
  - Tecla '**T**' - liga/desliga alternadamente
- Ligar/desligar a lanterna (foco)
  - Tecla: '**F**' - liga/desliga alternadamente
- Orientar a lanterna
  - Tecla: '**a**'/'**d**' - esquerda/direita
  - Tecla: '**w**'/'**s**' - cima/baixo

1.2 Mundo, observador e objectos

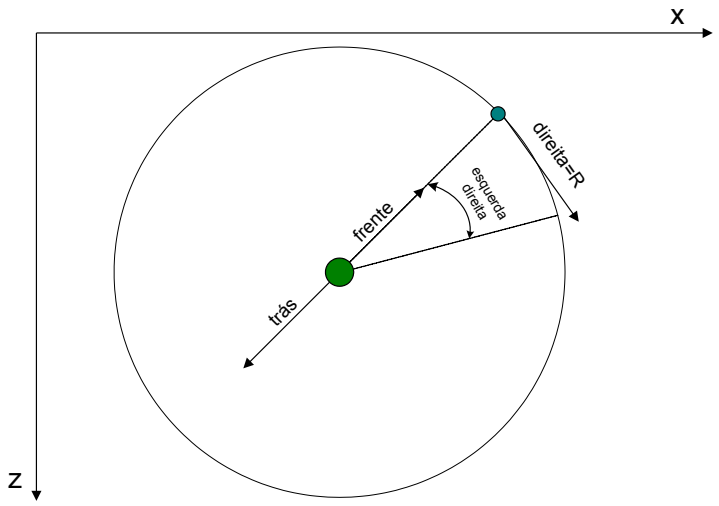


Mundo

Assuma que o mundo é um paralelepípedo, por exemplo de dimensões, [-1,1, -1,1, 0,1]

Observador

O observador encontra-se inicialmente numa determinada posição, por exemplo (x,y,z)=(0.0, 0.5, 1.0), podendo depois movimentar-se mas apenas segundo as coordenadas (x , z), isto é, a sua altura (posição segundo y) é fixa. Este pode andar apenas para a frente e para trás (usando as setas ↑ e ↓) e alterar a direcção do seu movimentos (teclas ← e →), como se mostra na figura (visto de cima):



Objectos

Assume-se que os modelos (objectos) estão "assentes" no plano y=0, isto é no plano x-z.

O OpenGL, mais precisamente a GLUT, implementa uma série de modelos tipo:

- void **glutWireSphere**(GLdouble radius, GLint slices, GLint stacks);
- void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);
- void **glutWireCube**(GLdouble size);
- void glutSolidCube(GLdouble size);
- void **glutWireTorus**(GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings);
- void glutSolidTorus(GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings);
- void **glutWireIcosahedron**(void);
- void glutSolidIcosahedron(void);
- void **glutWireOctahedron**(void);
- void glutSolidOctahedron(void);
- void **glutWireTetrahedron**(void);
- void glutSolidTetrahedron(void);
- void **glutWireDodecahedron**(GLdouble radius);
- void glutSolidDodecahedron(GLdouble radius);
- void **glutWireCone**(GLdouble radius, GLdouble height, GLint slices, GLint stacks);
- void glutSolidCone(GLdouble radius, GLdouble height, GLint slices,GLint stacks);
- void **glutWireTeapot**(GLdouble size);
- void glutSolidTeapot(GLdouble size);

1.3 Janela de visualização

Divida a janela de visualização em duas zonas distintas:

- **Mapa**: projecção ortogonal
- **Visita** virtual aos modelos: projecção em perspectiva

A divisão da janela de visualização pode ser conseguida com o comando **viewport**:

```
void glViewport(GLint x, GLint y, GLsizei width, GLsizei height);
```

Defines a pixel rectangle in the window into which the final image is mapped. The (x, y) parameter specifies the lower-left corner of the viewport, and width and height are the size of the viewport rectangle. By default, the initial viewport values are (0, 0, winWidth, winHeight), where winWidth and winHeight are the size of the window.

1.4 Depth Buffer

Não esquecer o buffer de profundidade. É importante para a correcta visualização de objectos 3D, como por exemplo visibilidade.

Activar

```
glEnable(GL_DEPTH_TEST);
```

Iniciar

```
glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH );
```

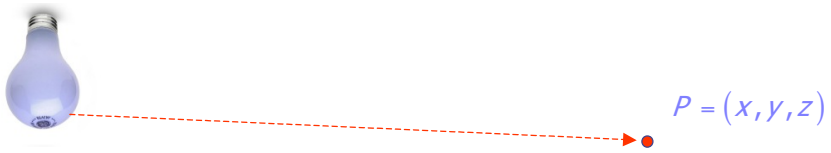
Limpar

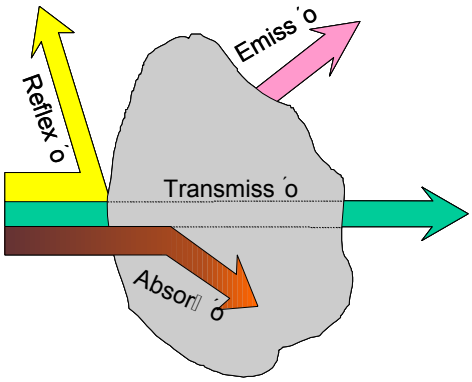
```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
```

2. Iluminação em OpenGL

2.1 Luzes e materiais

O OpenGl apresenta diversas funcionalidades relacionadas com a definição e implementação de modelos de iluminação. Além de um modelo de iluminação global de ambiente, em OpenGL a iluminação em cada ponto é determinado em função da definição de fontes de luz.





Assim, considerando a interacção entre uma fonte de luz e um objecto, a iluminação num ponto **P** é dada pelas seguintes componentes:

- **Ambiente**: com origem na fonte de luz propriamente dita
- **Reflexão**: resultante da luz que é reflectida no material e alcança o ponto P. Por sua vez é subdividida em duas componentes:
  - Difusa
  - Especular
- **Emissão**: do próprio material
- **Transmissão**: que atravessa o material

Existe uma outra componente de absorção, retida pelo material.

2.2 Modelo de Phong

O **OpenGL implementa o modelo de Phong**, em que são consideradas as componentes:

- Ambiente
- Reflexão difusa
- Reflexão especular
- Emissão.

Não é portanto possível considerar as componentes de transmissão e absorção.

2.3 Definição de luzes e materiais

Além de um modelo de iluminação ambiente global o OpenGl permite definir e activar até 8 fontes de luz (GL_LIGHT0, GL_LIGHT1, …, GL_LIGHT7). Para cada uma delas é possível caracterizar independentemente cada uma das componentes anteriores (ambiente, difusa e especular).

O mesmo se passa para as características dos materiais dos objectos existentes, isto é, podem ser individualmente especificadas as suas características.

Iluminação em geral

Para activar os modelos de iluminação de uma forma geral deve-se fazer

```
glEnable(GL_LIGHTING);
```

Modelo global

Para definir/activar o modelo ambiente global pode-se, por exemplo, fazer

```
GLfloat intensidadeCor[4]={ iR, iG, iB, 1.0};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, intensidadeCor);
```

em que iR, iG, iB, definem as intensidades de cada uma das componentes da cor: vermelha, verde e azul.

Luzes individuais

Cada uma das luzes (i=0,...,7)pode ser activada como

```
glEnable(GL_LIGHTi);
```

Quanto à sua definição das suas propriedades

```
glLightf*(GL_LIGHTi, propriedade, valor );
```

em que **propriedade** diz respeito às características da luz e **valor** ao valor respectivo da propriedade em causa (para a luz anteriormente activada GL_LIGHTi)

Materiais

De forma análoga as características de um material podem ser definidas como:

```
glMaterialfv( face, propriedade, valor);
```

em que **face** define a face em questão (frente/trás), **propriedade** diz respeito às características do material e **valor** ao valor respectivo da propriedade em causa.

glColorMaterial

A tarefa de definir/atribuir as propriedades de um material é um processo repetitivo e moroso. Neste contexto, uma particularidade interessante disponível em OpenGL, consiste **GL_COLOR_MATERIAL**. Basicamente é possível **definir cores em vez definir propriedades de materiais**.

Para activar esta funcionalidade

```
glEnable( GL_COLOR_MATERIAL)
```

Assim, basta definir as propriedades para um dos objectos e as suas propriedades são herdados para os restantes excepto a cor (que é necessário definir). Caso nenhum valor seja definido os valores por omissão são:

- GL_AMBIENT (0.2, 0.2, 0.2, 1.0) Cor ambiente do material
- GL_DIFFUSE (0.8, 0.8, 0.8, 1.0) Cor difusa do material
- GL_SPECULAR (0.0, 0.0, 0.0, 1.0) Cor especular do material
- GL_SHININESS 0.0 coeficiente de especularidade do material
- GL_EMISSION (0.0, 0.0, 0.0, 1.0) Cor emissiva do material

Ver com mais detalhe a utilização desta funcionalidade na parte final.

3. Propriedades

3.1 Cor/Intensidade

Nesta relação luz/material/ponto uma das características principais é a definição de cor/intensidade. Assim:

- A uma fonte de luz está associada a cor/intensidade da luz que emite.
- A um objecto está associado a cor/intensidade da luz que é capaz de reflectir
- A quantidade de luz num ponto é função das duas anteriores: luz que é e emitida pela fonte de luz e que é reflectida pelo material.

Luz

Por exemplo a seguinte instrução define uma fonte de luz com componentes vermelha e verde, de intensidade máxima (100%) vermelha e (50%) verde, num modelo de iluminação ambiente.

```
GLfloat luzCor[4]={1.0, 0.5, 0.0,1.0};
glLightfv(GL_LIGHT0, GL_AMBIENT, luzCor );
```

Material

A seguinte instrução define um material que reflecte apenas as cores verde e vermelho, respectivamente 30% e 100%, considerando a componente de reflexão difusa.

```
GLfloat matCor[4]={0.3, 1.0, 0.0,1.0};
glMaterialfv(GL_FRONT, GL_DIFFUSE, matCor);
```

Ponto

A quantidade luz de um Ponto P (iP) é função da cor/intensidade de emissão da fonte luz (iL) e cor/intensidade de reflexão do material (iM):

- iP (R)= iL(R)*iM(R) componente vermelha
- iP (V)= iL(V)*iM(V) componente verde
- iP (B)= iL(B)*iM(B) componente azul

3.2 Tipo de fontes de luz

Quanto ao tipo e localização, uma fonte de luz pode-se ser considerada no infinito ou a uma distância finita. Neste último caso poderá ser **pontual** (a luz propaga-se em todas as direcções) ou um **foco** (a luz propaga-se numa direcção bem definida).

Infinito

Por exemplo, caso se localize no infinito interessa definir a sua direcção (note-se que o ultimo parâmetro é 0.0-notação para vector de direcção)

```
GLfloat    direccao[ ]= { x, y, z, 0.0 };
```

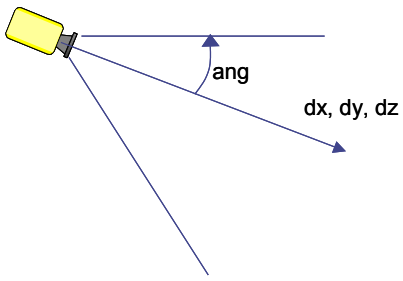
Pontual

Caso seja pontual (distância finita) interessa definir a sua posição (note-se que o ultimo parâmetro é 1.0-notação para ponto)

```
GLfloat    ponto[ ]= { x, y, z, 1.0 };
```

Foco

Caso seja do tipo foco, além da posição pode-se definir ângulo ([0..90]), direcção e concentração de luz ([0..128]).



Por exemplo

```
GLfloat direccao[ ] = { dx, dy, dz };
GLfloat concentracao = 0.3;
GLfloat angulo = 10.0;
```

Para activar

```
glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, direccao );
glLightf (GL_LIGHT1, GL_SPOT_EXPONENT , concentracao);
glLightf (GL_LIGHT1, GL_SPOT_CUTOFF, angulo);
```

3.3 Atenuação atmosférica

Considerando as quatro componentes:

Ambiente global: IAG

Ambiente local: IAL

Reflexão difusa: ID

Reflexão Especular: IE

O valor da iluminação num dado ponto é dada por:

$$I = I_{AG} + I_{AL} + I_D + I_E$$

Este valor pode ser ajustado, isto é, calculado com mais realismo, assumindo três tipos de atenuação atmosférica em função da distância da fonte de iluminação : constante (**kc**), linear (**kl**) e quadrática (**kd**):

$$I_a = \frac{I_0}{k_c + k_l d + k_q d^2}$$

Estes tipos de efeitos (atenuação) não têm sentido em modelos ambiente global e de emissão.

4. Componentes

Como se referiu são 4 os componentes considerados em OpenGL:

- Iluminação ambiente (global)
- Iluminação ambiente (local)
- Reflexão difusa
- Reflexão especular

4.1 Iluminação ambiente (Global)

Neste caso não tem sentido falar na localização do ponto de luz (surge de todas as direcções). Assim, é apenas necessário definir a cor/intensidade da luz.

Por exemplo a seguinte instrução

```
GLfloat luzGlobalCor[4]={0.1, 0.1, 0.1, 1.0};
```

define para as componentes (R,G,B) da fonte de luz uma intensidade igual=0.1, portanto contendo as componente vermelha, verde e azul, contudo bastante fraca!

Para **activar** o modelo de iluminação ambiente:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, luzGlobalCor);
```

4.2 Iluminação ambiente (Local)

É também possível definir uma componente iluminação local tipo ambiente. Depende da cor/intensidade da luz e da sua posição. No entanto, este tipo de iluminação é independente da posição, caso não se considerem atenuações da luz devido a efeitos atmosféricos.

Definição de alguns parâmetros

```
GLfloat localCor[4] ={0.1, 0.1, 0.1, 1.0};
GLfloat localPos[4] ={0.5, 1.0, 0.5, 1.0};
GLfloat localAttCon =1.0;
GLfloat localAttLin =0.05;
GLfloat localAttQua =0.0;
```

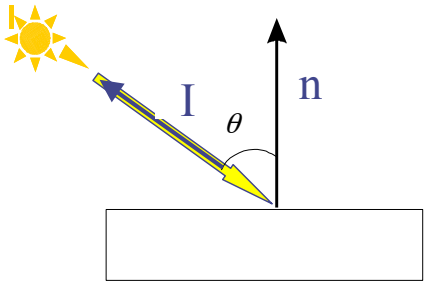
define para as componentes (R,G,B) da fonte de luz uma intensidade igual=0.1, localizada no ponto (x,y,z)=(0.5,1.0, 0.5).
Considera-se atenuação atmosférica de valores (C,L,Q)=(1.0, 0.05, 0.0).

Assumindo GL_LIGHT0 como definindo o ponto de luz, pode-se activar o modelo de iluminação local ambiente através de:

```
glLightfv(GL_LIGHT0, GL_POSITION, localPos );
glLightfv(GL_LIGHT0, GL_AMBIENT, localCor );
glLightf (GL_LIGHT0, GL_CONSTANT_ATTENUATION, localAttCon);
glLightf (GL_LIGHT0, GL_LINEAR_ATTENUATION, localAttLin) ;
glLightf (GL_LIGHT0, GL_QUADRATIC_ATTENUATION,localAttQua) ;
```

4.3 Reflexão Difusa

Neste caso considera-se a luz que é reflectida num dado objecto. Esta é dependente do ângulo entre a fonte de luz e a normal ao corpo e não depende da localização do ponto de observação.



Por exemplo

```
GLfloat ponto[ ]= { 1.0, 0.4, 1.6, 1.0 };
GLfloat CorDif[4] ={ 0.8,0.8,0.8, 1.0};
```

define a sua localização e intensidade de cor.

```
glLightfv(GL_LIGHT1, GL_POSITION, ponto );
glLightfv(GL_LIGHT1, GL_DIFFUSE, CorDif );
```

Da forma idêntica, segue a caracterização da reflexão difusa de um material

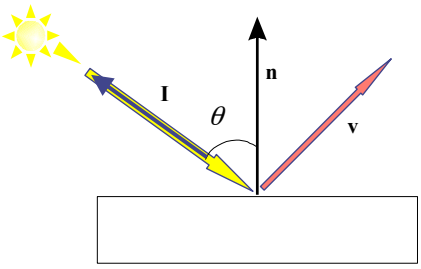
```
GLfloat matDifusa[ ] = {1.0,1.0,1.0,1.0};
```

e respectiva activação

```
glMaterialfv(GL_FRONT,GL_DIFFUSE, matDifusa);
```

4.4 Reflexão Especular

A reflexão especular é máxima na direcção especular atenuando-se acentuadamente (dependente do material) à medida que o ponto de observação se afasta desta direcção. Depende portanto da posição do observador.



Definição da componente especulares

```
GLfloat    ponto[ ]= { x, y, z, 1.0 };
GLfloat    CorEsp[4] ={ 1.0, 1.0, 1.0, 1.0};
```

e sua activação

```
glLightfv(GL_LIGHT1, GL_POSITION, pontoi );
glLightfv(GL_LIGHT1, GL_SPECULAR, CorEsp );
```

Para um material

```
GLfloat matEspecular[ ]= {1.0, 1.0, 1.0, 1.0};
GLint especMaterial = 3;
```

e sua activação

```
glMaterialfv(GL_FRONT,GL_SPECULAR, matEspecular);
glMateriali (GL_FRONT,GL_SHININESS,especMaterial);
```

O último parâmetro (GL_SHININESS) define o grau de especularidade (grau de polimento) do material. Varia de [0..128]. Quanto maior mais centrada será a reflexão.

5. Trabalho

5.1 Objectos: poligno

Além de alguns objectos pré-definidos deve definir um polígono (quadrado, por exemplo).

Tenha a tenção que além dos vértices é necessário **definir a normal**.

Tente perceber, do ponto de vista teórico, os resultados alcançados.

5.2 Lanterna: Foco

Deve simular, alem da luz natural e da luz do tecto já definidas, o efeito de uma lanterna transportada pelo observador. Por outras palavras uma luz do tipo foco.

Lanterna (foco)

- Cor/intensidade
- **Componentes difusas e especular**
- Posição (do observador que se desloca)
- Direcção (a ser controlada pelas teclas 'a','d','w','s')
- Ângulo e concentração de luz

5.3 Materiais

Além dos objectos já definidos recorrendo ao uso de glColorMaterial deve permitir implementar matérias recorrendo à definição das suas propriedades (ver anexo).

Pode utilizar um menu, mas não é necessário que assim seja.

As propriedades dos materiais podem ser definidas usando duas abordagens

- GL_Color_Material **activo**

- GL_Color_Material **inactivo**

GL_COLOR_MATERIAL inactivo

Neste caso é necessário definir exaustivamente as características de cada uma das componentes do material

- Ambiente
- Difuso
- Especular
- Emissão

Veja em anexo algumas propriedades para alguns materiais.

GL_COLOR_MATERIAL activo

Neste caso é possível 'definir cores' em vez de 'propriedades de materiais'.

A cor final é calculada pela combinação das características da fonte de luz e da cor dos materiais.

Note-se que é possível definir quais as componentes da cor que serão afectadas. Exemplo, para que a componente Emissão seja alterada usando glColor:

```
glColorMaterial ( GL_FRONT_AND_BACK, GL_EMISSION ) ;
glEnable ( GL_COLOR_MATERIAL ) ;
```

Um vez efectuado a função

```
glColor (Cor)
```

tem o mesmo efeito que especificar directamente a cor do material (componente emissão).

Um comando especialmente útil é o seguinte:

```
glColorMaterial ( GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE ) ;
```

Permite alterar em simultâneo as componentes ambiente e difusa, componentes muito usado em ambientes reais.

5.3 Anexo

Name	Ambient			Diffuse			Specular			Shininess
emerald	0.0215	0.1745	0.0215	0.07568	0.61424	0.07568	0.633	0.727811	0.633	0.6
jade	0.135	0.2225	0.1575	0.54	0.89	0.63	0.316228	0.316228	0.316228	0.1
obsidian	0.05375	0.05	0.06625	0.18275	0.17	0.22525	0.332741	0.328634	0.346435	0.3
pearl	0.25	0.20725	0.20725	1.0	0.829	0.829	0.296648	0.296648	0.296648	0.088
ruby	0.1745	0.01175	0.01175	0.61424	0.04136	0.04136	0.727811	0.626959	0.626959	0.6
turquoise	0.1	0.18725	0.1745	0.396	0.74151	0.69102	0.297254	0.30829	0.306678	0.1
brass	0.329412	0.223529	0.027451	0.780392	0.568627	0.113725	0.992157	0.941176	0.807843	0.21794872
bronze	0.2125	0.1275	0.054	0.714	0.4284	0.18144	0.393548	0.271906	0.166721	0.2
chrome	0.25	0.25	0.25	0.4	0.4	0.4	0.774597	0.774597	0.774597	0.6
copper	0.19125	0.0735	0.0225	0.7038	0.27048	0.0828	0.256777	0.137622	0.086014	0.1
gold	0.24725	0.1995	0.0745	0.75164	0.60648	0.22648	0.628281	0.555802	0.366065	0.4
silver	0.19225	0.19225	0.19225	0.50754	0.50754	0.50754	0.508273	0.508273	0.508273	0.4
black plastic	0.0	0.0	0.0	0.01	0.01	0.01	0.50	0.50	0.50	0.25
cyan plastic	0.0	0.1	0.06	0.0	0.50980392	0.50980392	0.50196078	0.50196078	0.50196078	0.25
green plastic	0.0	0.0	0.0	0.1	0.35	0.1	0.45	0.55	0.45	0.25
red plastic	0.0	0.0	0.0	0.5	0.0	0.0	0.7	0.6	0.6	0.25
white plastic	0.0	0.0	0.0	0.55	0.55	0.55	0.70	0.70	0.70	0.25
yellow plastic	0.0	0.0	0.0	0.5	0.5	0.0	0.60	0.60	0.50	0.25
black rubber	0.02	0.02	0.02	0.01	0.01	0.01	0.4	0.4	0.4	0.078125
cyan rubber	0.0	0.05	0.05	0.4	0.5	0.5	0.04	0.7	0.7	0.078125
green rubber	0.0	0.05	0.0	0.4	0.5	0.4	0.04	0.7	0.04	0.078125
red rubber	0.05	0.0	0.0	0.5	0.4	0.4	0.7	0.04	0.04	0.078125
white rubber	0.05	0.05	0.05	0.5	0.5	0.5	0.7	0.7	0.7	0.078125
yellow rubber	0.05	0.05	0.0	0.5	0.5	0.4	0.7	0.7	0.04	0.078125

Observação: multiplicar o valor de *shininess* por 128.