Textura e Iluminação

SCC0250/0650 - Computação Gráfica

Profª. Rosane Minghim
https://edisciplinas.usp.br/course/view.php?id=61213
https://edisciplinas.usp.br/course/view.php?id=61210
rminghim@icmc.usp.br
P.A.E. Diego Cintra e Fábio Felix
diegocintra@usp.br, f_diasfabio@usp.br

Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)
baseado no material de anos anteriores, vários autores

21 de maio de 2018

Sumário

Textura

Mesmo a textura sendo uma característica importante para descrever uma imagem, é difícil defini-la. Entretanto, de maneira simples, pode-se dizer que ela é um conjunto de variações de intensidade que formam padrões e repetições.

Figura: Obtida no site Pexels.

Figura: Obtida no site Pexels.

Textura

Figura: Obtida no site Pexels.

Figura: Obtida no site Pexels.

OpenGL - Textura 1D

```
1  ...
2  GLubyte texLine[16];
3  glTexImage1D(GL_TEXTURE_1D, 0, GL_RGBA, 4, 0, GL_RGBA, GL_UNSIGNED_BYTE, texLine↩
       );
4  glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
5  glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
6  glEnable(GL_TEXTURE_1D);
7
8  glBegin(GL_LINES);
9   glTexCoord1f(0.0);
10  glVertex2d(0.0, 0.0);
11
12  glTexCoord1f(1.0);
13  glVertex2d(1.0, 1.0);
14
15  glTexCoord1f(1.0);
16  glVertex2d(0.0, 0.0);
17
18  glTexCoord1f(0.0);
19  glVertex2d(0.0, -1.0);
20 glEnd();
21
22 glDisable(GL_TEXTURE_1D);
23 ...
```

OpenGL - Textura 1D

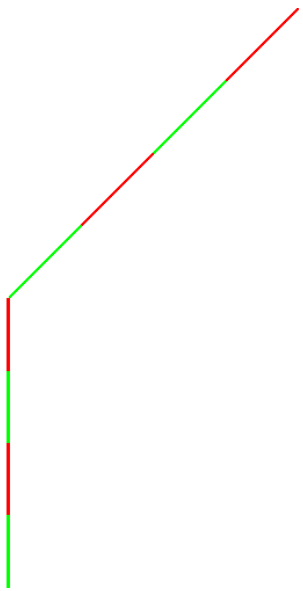


Figura: Textura em linhas.

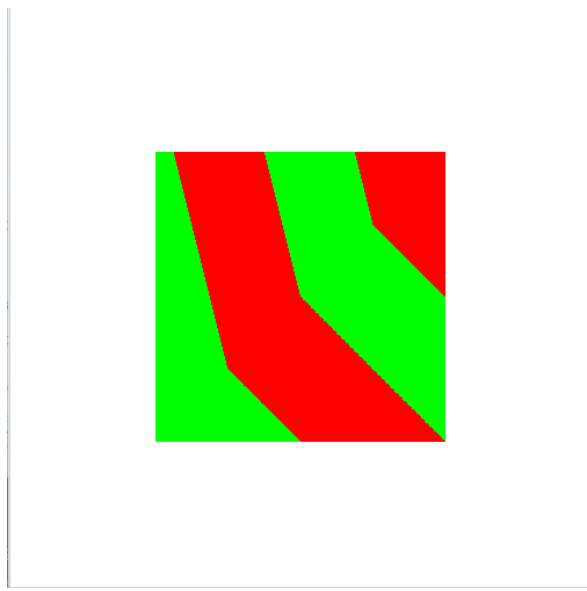


Figura: Textura em quadrilátero.

```
1  void glTexImage1D(GLenum target, GLint level, GLint internalFormat, GLsizei width, GLint border, GLenum format, ⟵
       GLenum type, const GLvoid * data);
2  void glTexParameteri(GLenum target, GLenum pname, GLint param);
3  void glTexCoord1f(GLfloat s);
```

OpenGL - Textura 2D

```
1  ...
2  GLubyte matrixTex[32][32][4];
3  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 32, 32, 0, GL_RGBA, GL_UNSIGNED_BYTE, ↩
       matrixTex);
4
5  // Definir GL_TEXTURE_MIN_FILTER e GL_TEXTURE_MAG_FILTER ↩
       para GL_TEXTURE_2D
6
7  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
8  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
9
10 glEnable(GL_TEXTURE_2D);
11
12 glBegin(GL_QUADS);
13   glTexCoord2f(0.0, 1.0); glVertex2d(-0.5, 0.5);
14
15   glTexCoord2f(0.0, 0.0); glVertex2d(-0.5, -0.5);
16
17   glTexCoord2f(1.0, 0.0); glVertex2d(0.5, -0.5);
18
19   glTexCoord2f(1.0, 1.0); glVertex2d(0.5, 0.5);
20 glEnd();
21
22 glDisable(GL_TEXTURE_2D);
23 ...
```

OpenGL - Textura 2D

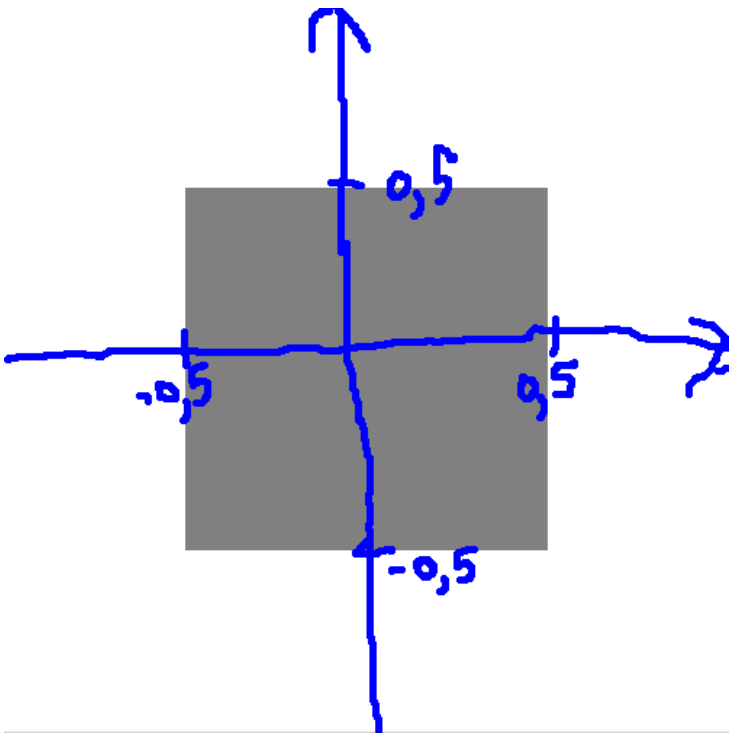


Figura: Coordenadas do objeto.

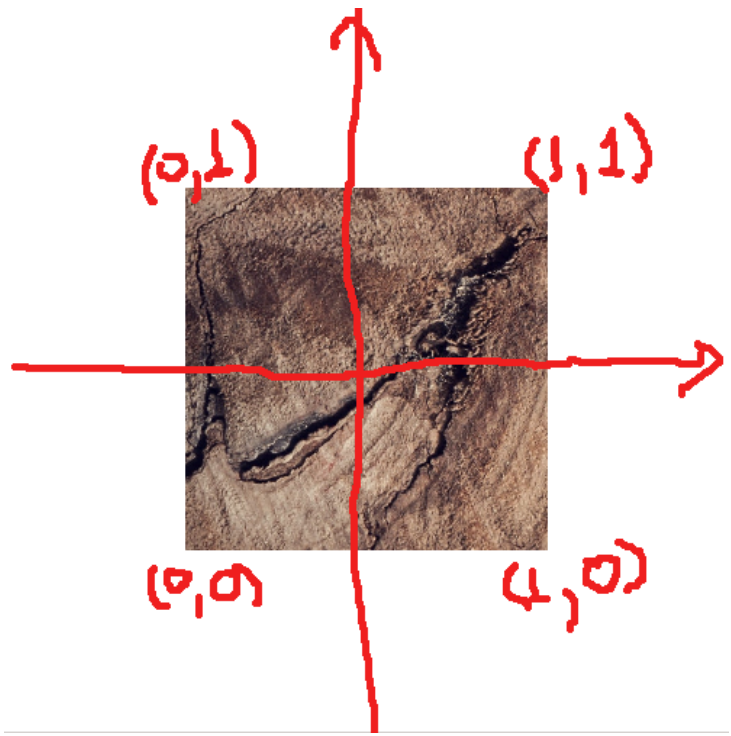


Figura: Coordenadas da textura.

```
1 void glTexImage2D(GLenum target, GLint level, GLint internalFormat, GLsizei width, GLsizei height, GLint border,↩
        GLenum format, GLenum type, const GLvoid * data);
2 void glTexCoord2f(GLfloat s, GLfloat t);
```

OpenGL - Textura 2D

No lugar de criar uma matrix de 3 dimensões, o que é trabalhoso, é possível utilizar uma biblioteca qualquer que carregue uma imagem de textura. O exemplo abaixo utiliza OpenCV com essa finalidade.

```
1  ...
2  #include <opencv2/opencv.hpp>
3  ...
4  cv::Mat img = cv::imread("texture.jpg");
5  //efetuar o flip da imagem se for necessario
6  cv::flip(img, img, 0);
7
8  //A OpenCV representa as cores como BGR em vez de RGB
9  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, img.cols, img.rows, 0, GL_BGR, GL_UNSIGNED_BYTE, img.ptr());
10 ...
```

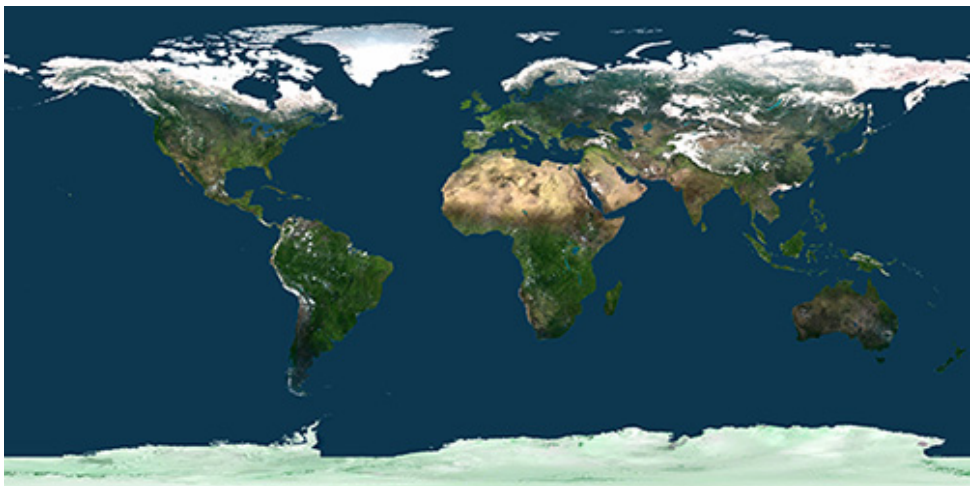
OBSERVAÇÃO

Texturas não são aplicadas a objetos criados com funções como *glutWireTeapot*, *glutWireCube*, *glutSolidCube* etc. Para isso, utilizar o objeto *GLUquadric* como nos próximos exemplos.

OpenGL - Textura 3D

```
1  ...
2  glTexImage3D(GL_TEXTURE_3D, 0, GL_RGBA, img.cols, img.rows, 1, 0, GL_BGR, ↩
       GL_UNSIGNED_BYTE, img.ptr());
3
4  // Definir GL_TEXTURE_MIN_FILTER e GL_TEXTURE_MAG_FILTER ↩
       para GL_TEXTURE_3D
5
6  glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_S, GL_REPEAT);
7  glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_T, GL_REPEAT);
8  glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_R, GL_REPEAT);
9
10 glEnable(GL_TEXTURE_3D);
11
12 GLUquadric *quadObj = gluNewQuadric();
13
14 gluQuadricTexture(quadObj, GL_TRUE);
15 gluSphere(quadObj, 0.5, 40, 50);
16 gluDeleteQuadric(quadObj);
17
18 glDisable(GL_TEXTURE_3D);
19 ...
```

OpenGL - Textura 3D



Perceba que nesse caso a OpenGL mapeou a textura sem a necessidade de associar as coordenadas dela com as do objeto.

```
1  void glTexImage3D(GLenum target, GLint level, GLint internalFormat, GLsizei width, GLsizei height, GLsizei depth
       , GLint border, GLenum format, GLenum type, const GLvoid * data);
```

OpenGL - Cubemap Texture

```
1  struct Point
2  {
3    GLfloat x, y, z;
4  };
5  ...
6  std::vector<Point> POINTS;
7  std::vector<std::string> faces;
8  std::vector<GLenum> faces_index;
9  ...
10 faces_index.push_back(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z); //front
11 faces_index.push_back(GL_TEXTURE_CUBE_MAP_POSITIVE_Z); //back
12 faces_index.push_back(GL_TEXTURE_CUBE_MAP_POSITIVE_X); //right
13 faces_index.push_back(GL_TEXTURE_CUBE_MAP_NEGATIVE_X); //left
14 faces_index.push_back(GL_TEXTURE_CUBE_MAP_POSITIVE_Y); //top
15 faces_index.push_back(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y); //bottom
16 ...
```

OpenGL - Cubemap Texture

```
1  for(int i = 0; i < faces.size(); i++)
2  {
3    cv::Mat img = cv::imread(faces[i].c_str());
4  //glTexImage1D e glTexImage3D nao aceitam os valores definidos em ↩
       faces_index
5    glTexImage2D(faces_index[i], 0, GL_RGBA, img.cols, img.rows, 0, GL_BGR, ↩
         GL_UNSIGNED_BYTE, img.ptr());
6  }
7
8  // Definir GL_TEXTURE_MIN_FILTER, GL_TEXTURE_MAG_FILTER, ↩
       GL_TEXTURE_WRAP_S, GL_TEXTURE_WRAP_T e ↩
       GL_TEXTURE_WRAP_R para GL_TEXTURE_CUBE_MAP
9
10 glEnable(GL_TEXTURE_CUBE_MAP);
11 glBegin(GL_QUADS);
12
13 for(int i = 0; i < POINTS.size(); i++)
14 {
15   glTexCoord3f(POINTS[i].x, POINTS[i].y, POINTS[i].z);
16   glVertex3f(POINTS[i].x, POINTS[i].y, POINTS[i].z);
17 }
18
19 glEnd();
20 glDisable(GL_TEXTURE_CUBE_MAP);
21 ...
```

OpenGL - Cubemap Texture

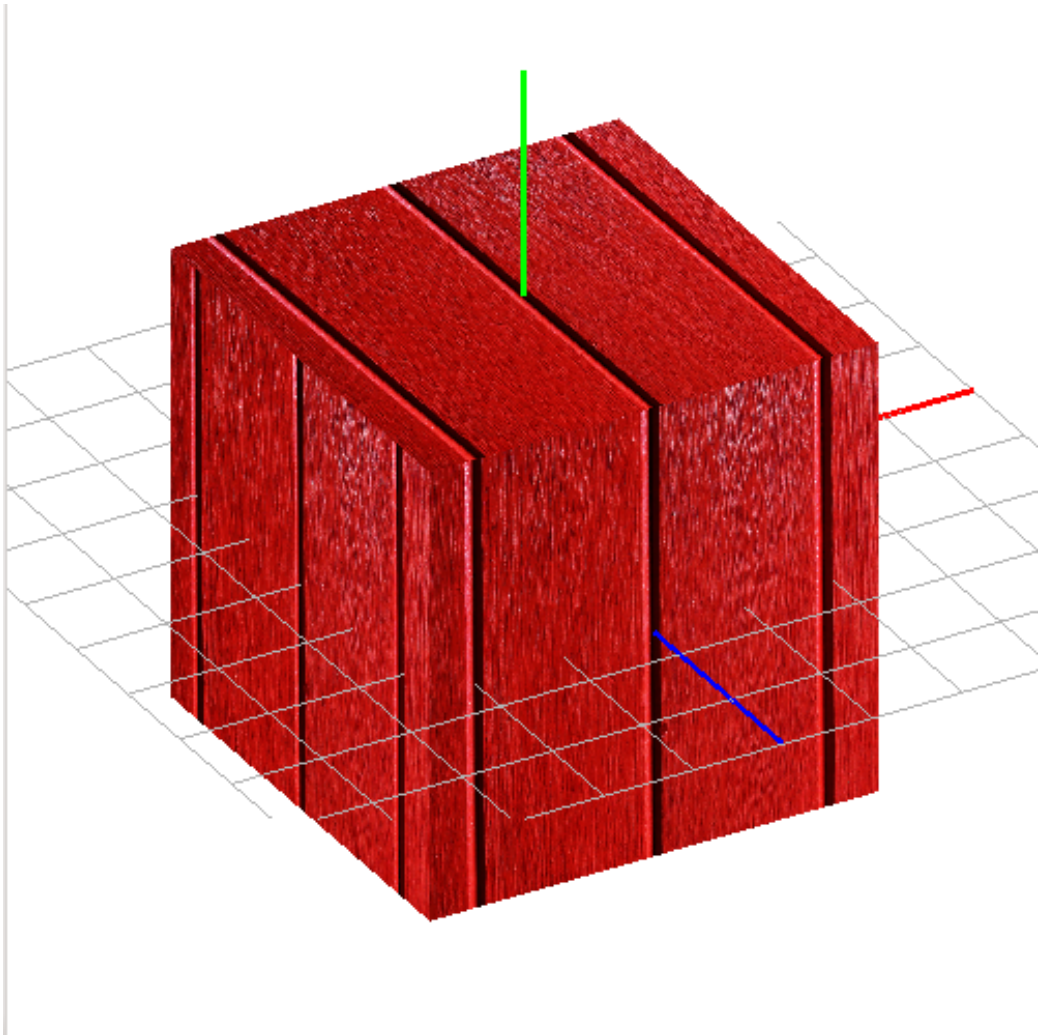


Figura: Obtida no site Pexels.

OpenGL - Aplicando Cubemap para criar Skybox

Skybox é cubo grande o baste para conter toda uma cena. Esse cubo possui 6 imagens que rodeiam a "câmera". Essa aplicação proporciona uma noção de profundidade do ambiente.

Para que funcione corretamente, as faixas de valores das coordenadas do cubo devem ser maiores que as das coordenadas dos objetos da cena.

Os valores de *near* e *far* das projeções utilizadas, sejam elas ortogonais ou de pesrpectiva, devem ser grandes o bastante para conter o cubo e não o recortar quando alguma rotação for realizada na cena.

Lembrar que o ponto de visualização padrão está em $(0, 0, 0)$. Então qualquer objeto que for desenhado nessa posição pode não ser visualizado corretamente. Os objetos devem ser transladados para que possam ser apresentados, ou as coordenadas do *gluLookAt* devem ser modificadas.

OpenGL - Aplicando Cubemap para criar Skybox

```
1  gluPerspective(90.0, aspect, 0.1, 50);
2  ...
3  gluLookAt(0.0, 0.0, 1.0,
4            0.0, 0.0, -1.0,
5            0.0, 1.0, 0.0);
6  ...
7  //Pontos dos quadrilateros com coordenadas definidas entre −5 e 5.
```

OpenGL - Outras funções

```
1  //Gera uma quantidade n de identificadores de texturas
2  void glGenTextures(GLsizei n, GLuint * textures);
3
4  //Remove da memora n padores de texturas
5  void glDeleteTextures(GLsizei n, const GLuint * textures);
6
7  //Atribui um identificar numerico a um determinado tipo de textura.
8  void glBindTexture(GLenum target, GLuint texture);
9
10 //Carrega uma textura e substitui uma parte de uma textura existente
11 void glTexSubImage*D(...)
```

Iluminação

Modelos de iluminação são utilizados para calcular a cor de uma região da superfície de um objeto que está sendo iluminado. Esse processo garante maior grau de realismo a uma cena.

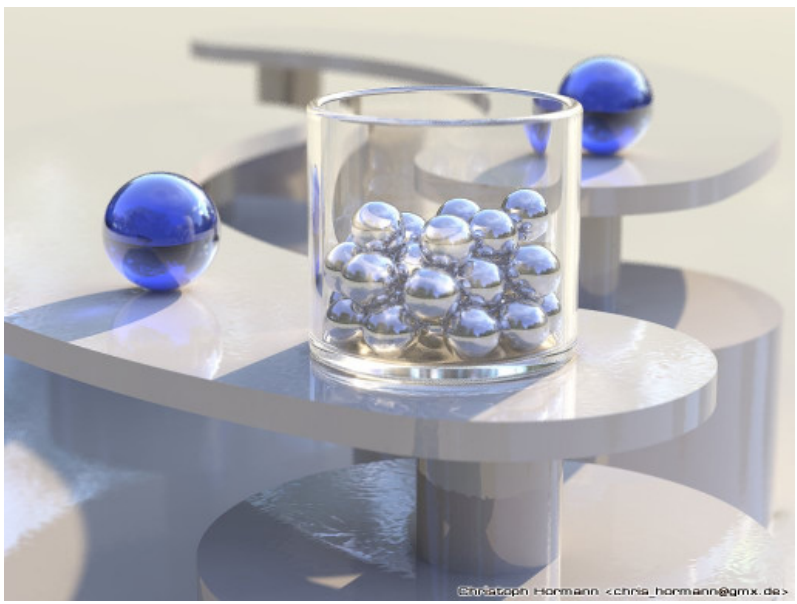


Figura: Obtida no site Povray.

Figura: Obtida no site Povray.

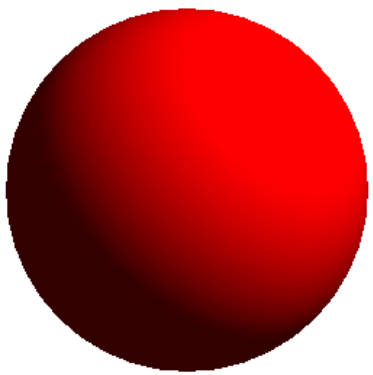
Iluminação

Figura: Obtida no site Povray.

Figura: Obtida no site Povray.

OpenGL - Iluminação

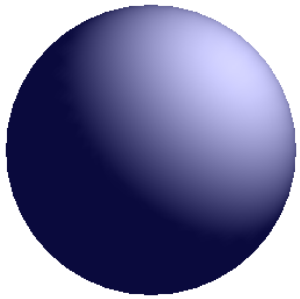
```
1  ...
2  GLfloat position_type[] = { 1.0, 1.0, 1.0, 0.0 };
3
4  //GL_LIGHT0, GL_LIGHT1, ... , GL_LIGHT7
5  glLightfv(GL_LIGHT0, GL_POSITION, position_type);
6
7  glEnable(GL_COLOR_MATERIAL);
8  glEnable(GL_LIGHTING);
9  glEnable(GL_LIGHT0);
10 ...
```



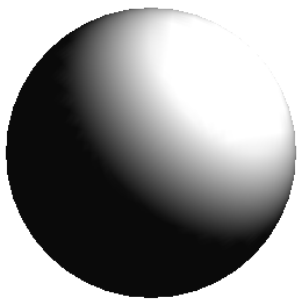
```
1  void glLightfv(GLenum light, GLenum pname, const GLfloat * params);
```

OpenGL - Iluminação

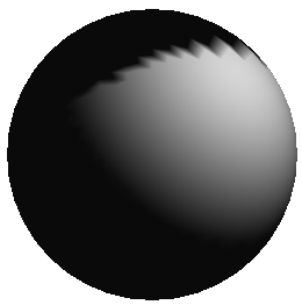
```
1  GLfloat light_color[] = { 0.0, 0.0, 1.0, 1.0 };
2
3  //GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR
4  glLightfv(GL_LIGHT0, GL_AMBIENT, light_color);
```



```
1  // GL_CONSTANT_ATTENUATION, ↩
       GL_LINEAR_ATTENUATION, and ↩
       GL_QUADRATIC_ATTENUATION
2  // OBS.: o tipo em position_type deve ser diferente de 0
3  glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 0.6);
```

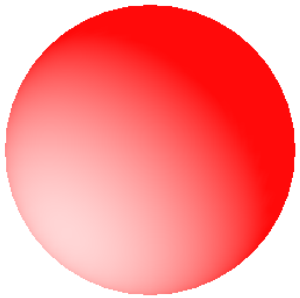


```
1  //x, y, z (0, 0, -1)
2  GLfloat dirVec[] = {0.0, -1.0, 0.0};
3  ...
4  // GL_SPOT_DIRECTION, GL_SPOT_CUTOFF, ↩
       GL_SPOT_EXPONENT
5  glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, dirVec);
6  //Valores entre 0 e 90. O valor padrao (180) emite a luz ↩
       em todas as direcoes.
7  glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 60);
8  //Valores entre 0 (default) e 128
9  glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 0.0);
```

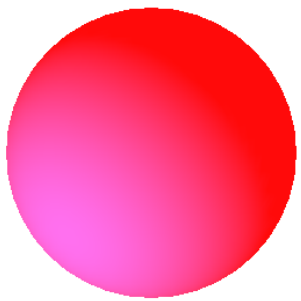


OpenGL - Iluminação

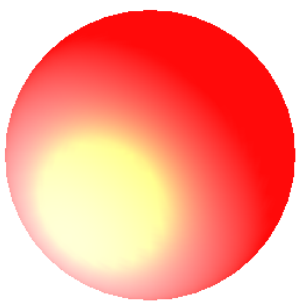
```
1  GLfloat material_color[] = { 1.0, 0.0, 0.0, 0.0 };
2
3  //Invocar antes de criar o objeto que possui o material
4  //GL_FRONT, GL_BACK, GL_FRONT_AND_BACK
5  glMaterialfv(GL_FRONT, GL_EMISSION, material_color);
```



```
1  GLfloat diffuse[] = { 0.2, 0.4, 0.9, 1.0 };
2  //GL_AMBIENT, GL_DIFFUSE, ↩
       GL_AMBIENT_AND_DIFFUSE, ↩
       GL_SPECULAR
3  glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuse);
```



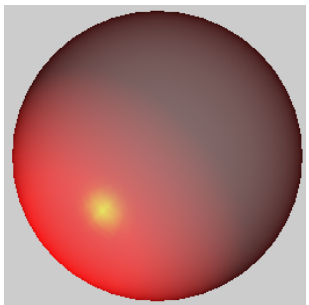
```
1  GLfloat specular[] = { 0.0, 1.0, 0.0, 1.0 };
2  glMaterialfv(GL_FRONT, GL_SPECULAR, specular);
3  //Expoente da reflexao especular
4  glMaterialf(GL_FRONT, GL_SHININESS, 10.0);
```



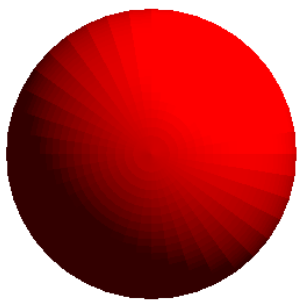
Quando *GL_COLOR_MATERIAL* é habilitado as cores do material seguem o que for definido com *glColor*. A função *glColorMaterial* define quais face e reflexão devem seguir essas cores.

OpenGL - Iluminação

```
1  GLfloat fog_color[] = {0.8, 0.8, 0.8, 1.0};
2
3  glFogfv(GL_FOG_COLOR, fog_color);
4  //Atenuacao
5  // GL_LINEAR, GL_EXP, GL_EXP2
6  glFogi(GL_FOG_MODE, GL_EXP);
7  //Atributo para os modos exponenciais
8  glFogf(GL_FOG_DENSITY, 1.5);
9
10 glEnable(GL_FOG);
```



```
1  //GL_FLAT, GL_SMOOTH (default)
2  glShadeModel(GL_FLAT);
3
4  //Define as componentes do vetor normal de uma superficie
5  //glNormal3f(x, y, z);
6
7  //Normaliza os vetores normais dos vertices
8  //glEnable(GL_NORMALIZE);
```



Bibliografia

- **Básica:**
  - Hearn, D. Baker, M. P. Computer Graphics with OpenGL, Prentice Hall, 2004. **(livro texto)**
  - Neider, J. Davis, T. Woo, M. OpenGL programming guide, 2007.
  - Angel, E. Interactive computer graphics: a top-down approach with OpenGL, Addison Wesley, 2000.
  - Foley, J. et. al. Introduction to Computer Graphics, Addison-Wesley, 1993.
  - Angle, E. and Shreiner, D., 2011. Interactive computer graphics: A topdown approach with shader-based opengl.

Bibliografia

- **Complementar:**
  - Computer Graphics Comes of Age: An Interview with Andries van Dam. CACM, vol. 27, no. 7. 1982
  - The RenderMan – And the Oscar Goes to... IEEE Spectrum, vol. 38, no. 4, abril de 2001.
  - Material do ano passado: `https://sites.google.com/site/computacaograficaicmc2017t2/`
  - Apostilas antigas da disciplina Computação Gráfica
    - `http://www.gbdi.icmc.usp.br/material?q=system/files/apostilas.pdf`
  - Curso da ACM SIGGRAPH (on line)