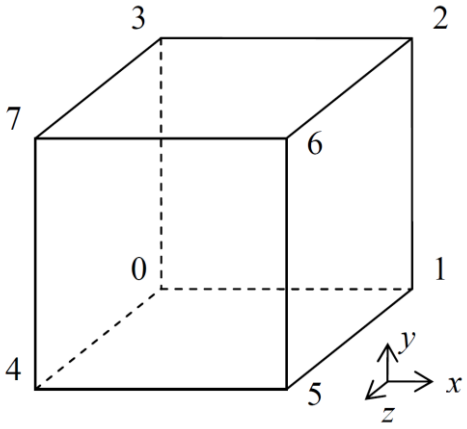
Aula Prática Laboratorial n.º 4

Sumário

Projecto “cubo”. Tutoriais Nate Robins “*transformation*” e “*projection*”.

Enunciado

Projecto “cubo”



1. Crie uma função **`void`** `cubo()`, que desenhe um cubo centrado na origem, alinhado com os eixos e de lado unitário, com cores diferentes nas suas diferentes faces. Baseie-se na função já existente, `cubo()`, a qual desenha já uma das faces (um polígono de quatro lados). Deverá alterar o *array* `vertices[][]`, de modo a definir os quatro vértices em falta; e deverá também ter em consideração que os vértices de cada face deverão ser especificados no sentido directo, isto é, contrário ao sentido dos ponteiros de um relógio.
2. Altere os valores de `modelo.theta[0]`, `modelo.theta[1]` e `modelo.theta[2]` usados nas chamadas à função na instrução `glRotatef()`, na função `Draw()`.
3. Insira o código necessário para criar e activar o *buffer* de profundidade, de modo a que as faces escondidas do cubo não sejam desenhadas:
   - Na função `main()`, acrescentar `GLUT_DEPTH` ao argumento da instrução `glutInitDisplayMode()`.
   - Na função `Init()`, remover o comentário associado à instrução `glEnable(GL_DEPTH_TEST);`

- Na função `Draw()`, acrescentar `GL_DEPTH_BUFFER_BIT` ao argumento da instrução `glClear()`.

4. Na função `Timer()`, insira o código necessário para rodar o cubo em torno dos três eixos principais (*x*, *y* e *z*). Use a função `Mouse()` para seleccionar o eixo de rotação com base no botão do rato premido (botão esquerdo: eixo *x*; botão central: eixo *y*; botão direito: eixo *z*). A variável `modelo.eixoRodar` especifica o índice do *array* `modelo.theta[]`). Se o utilizador premir a tecla correspondente ao eixo que está a rodar, o cubo deverá parar de rodar.
5. Insira o código necessário para programar as teclas '+' e '-' no sentido de aumentar e diminuir a dimensão (o lado) do cubo (instrução `glScalef()` na função `Draw()`).
6. Use a função `glPolygonMode()` para alterar a representação gráfica dos polígonos constituintes das faces do cubo e, desta forma, verificar se as mesmas estão correctamente orientadas. Outro teste consiste em usar a instrução `glEnable(GL_CULL_FACE)` para desenhar apenas as faces com o lado da frente voltado para o observador.
7. Crie uma função `eixos()`, que desenhe a componente positiva dos três eixos principais em três cores distintas (vermelho, verde e azul).
8. Use a função `eixos()` para replicar o sistema de eixos coordenados no canto inferior esquerdo da janela gráfica. Recorra às transformações geométricas (`glTranslatef()`, `glRotatef()` e `glScalef()`) e à pilha (*stack*) de matrizes disponibilizados pelo OpenGL (`glPushMatrix()` e `glPopMatrix()`).
9. Coloque um cubo de menor dimensão em cada um dos eixos do cubo maior.
10. Anime os três cubos de menor dimensão, de modo a que estes se aproximem e afastem do cubo maior (variável `modelo.translacaoCubo`).

11. Anime os três cubos de menor dimensão, de modo a que estes rodem em torno do eixo respectivo ao aproximarem-se do cubo maior; e evitar que os mesmos rodem quando se afastam (variável `modelo.thetaCubo`).
12. Crie uma função `cubo1()` que modele o mesmo cubo – centrado na origem, alinhado com os eixos e de lado unitário – recorrendo repetidamente ao desenho de apenas uma das suas faces e às transformações – translações e rotações – adequadas.

Tutoriais Nate Robins "*transformation*" e "*projection*"

Execute os tutoriais "*transformation*" e "*projection*"de Nate Robins.

Funções

- **void** `glFrontFace(`**GLenum** `mode)`

  Define o lado da frente do polígono: `GL_CCW` (sentido contrário ao dos ponteiros de um relógio; opção por omissão) ou `GL_CW` (sentido dos ponteiros de um relógio).

- **void** `glPolygonMode(`**GLenum** `face,` **GLenum** `mode)`

  Define o modo de representação das faces de um polígono:

  `face` – `GL_FRONT`, `GL_BACK` ou `GL_FRONT_AND_BACK`;

  `mode` – `GL_FILL`, `GL_LINE` ou `GL_POINT`.

- **void** `glLoadIdentity(`**void**`)`

  Carrega a matriz seleccionada com a matriz identidade, descartando eventuais transformações previamente efectuadas.

- **void** `glPushMatrix(`**void**`)`
- **void** `glPopMatrix(`**void**`)`

  Guarda numa pilha (*stack*) / restaura o estado da matriz seleccionada.

- **void** `glTranslatef(`**GLfloat** `tx,` **GLfloat** `ty,` **GLfloat** `tz)`

Promove a translação de um objecto de `tx` unidades em *x*, `ty` unidades em *y* e `tz` unidades em *z*.

- **`void`** `glRotatef(`**`GLfloat`** `angle,` **`GLfloat`** `rx,` **`GLfloat`** `ry,` **`GLfloat`** `rz)`

  Promove a rotação de um objecto de `angle` graus em torno do eixo definido pelo vector (`sx`, `sy`, `sz`).

- **`void`** `glScalef(`**`GLfloat`** `sx,` **`GLfloat`** `sy,` **`GLfloat`** `sz)`

  Promove o escalamento de um objecto dos factores de multiplicação `sx` em *x*, `sy` em *y* e `sz` em *z*.