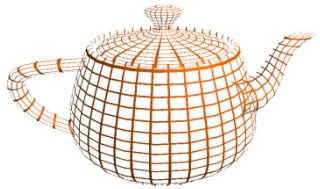
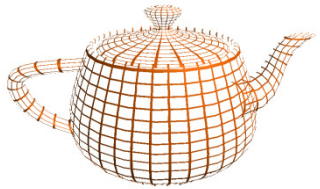
Computação Gráfica

Interactividade Básica com GLUT e Primitivas Geométricas



Interactividade Básica com GLUT

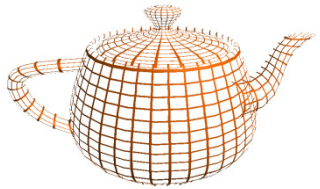
- O GLUT suporta de forma simples um conjunto alargado de dispositivos de entrada:
  - Rato
  - Teclado
  - Trackball
  - Tablet



Interactividade com GLUT

- A utilização de dispositivos de entrada em GLUT implica a definição de funções para processamento dos eventos gerados.

- Implica ainda o registo dessas funções no ambiente do GLUT.



Registo de Callbacks - Teclado

- **`glutKeyboardFunc(nome_função);`**

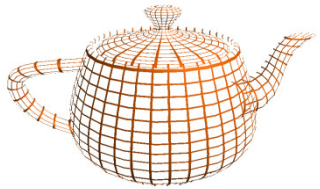
Registo da função que processa eventos relacionados com o teclado.

Esta função será invocada sempre que for premida uma tecla "normal".

A função é invocada com a indicação da tecla, e a posição do rato em coordenadas relativas da janela.

Assinatura da função registada:

```
void nome_função(unsigned char tecla, int x, int y);
```



Registo de Callbacks - Teclado

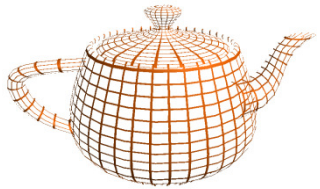
- **`glutSpecialFunc(nome_função);`**

Equivalente à função anterior mas para as teclas especiais: F1..F12, HOME,END, Setas, etc...

Os códigos das teclas são constantes definidas em GLUT com o prefixo GLUT_KEY, por exemplo: GLUT_KEY_F1.

Assinatura da função registada:

```
void nome_função(int tecla, int x, int y);
```



Registo de Callbacks - Rato

- **`glutMouseFunc(nome_função);`**

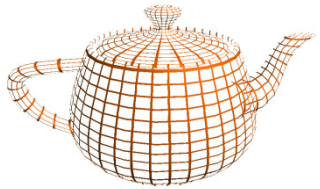
Regista uma função a invocar quando um dos botões é premido, ou solto.

A função registada devolve informação sobre:

- posição do rato (em coordenadas relativas da janela);
- qual o botão que gerou o evento (GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON);
- qual o estado do botão (GLUT_UP, GLUT_DOWN).

A função registada tem a seguinte assinatura:

- `void nome_função(int botão, int estado, int x, int y);`



Registo de Callbacks - Rato

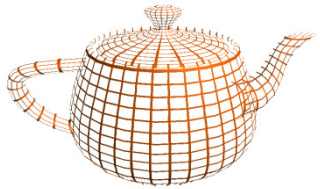
- **`glutMotionFunc(nome_função);`**
- **`glutPassiveMotionFunc(nome_função);`**

Regista uma função a invocar quando há movimento do rato. A função glutMotionFunc deve ser utilizada quando se pretende detectar movimento com um dos botões premidos.

A função registada devolve informação sobre a posição do rato (em coordenadas relativas da janela);

A função registada tem a seguinte assinatura:

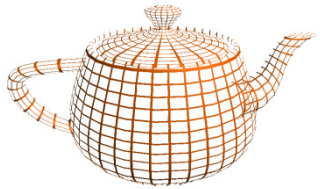
- `void nome_função(int x, int y);`



Interactividade com GLUT

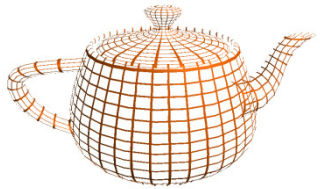
- Através do GLUT é ainda possível definir pop-up menus de uma forma simples.

- Os items dos pop-up menus podem por sua vez também ser menus.



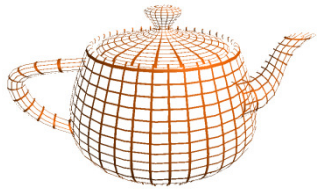
Interactividade com GLUT - Menus

- Primeiro passo: Criar um menu.
  - No processo de criação de um menu é necessário especificar qual a função que irá processar as opções do menu.
    - **`int glutCreateMenu(nome_função);`**
  - O valor devolvido por *glutCreateMenu* é o identificador do menu.
  - A função registada receberá como parâmetro o identificador da opção.
  - Assinatura da função registada:
    - `void nome_função(int op);`



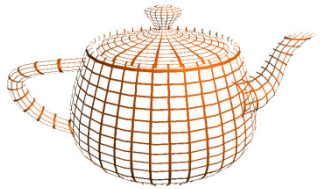
Interactividade com GLUT - Menus

- Segundo passo: Adicionar opções ao menu.
  - **`void glutAddMenuEntry(char *op, int valor);`**
  - Ex: `glutAddMenuEntry("Vermelho",1);`
  - As opções são adicionadas ao fim do menu. Não é possível inserir a meio ou no princípio.
- Terceiro passo: Associar o menu a um botão do rato
  - **`glutAttachMenu(int botão);`**
  - botão = GLUT_LEFT_BUTTON, GLUT_RIGHT_BUTTON, ou GLUT MIDDLE_BUTTON



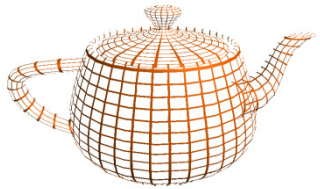
Interactividade com GLUT - Menus

- Notas:
  - Ao adicionar elementos a um menu não se especifica qual o menu. Quando se cria um menu este passa a ser o *menu actual*, e é neste menu que as opções são inseridas.
  - Pode-se alterar o menu actual através da função
    - `glutSetMenu(int menuId);`



Gestão de Recursos

- Ao utilizar a idleFunc, o GLUT está a redesenhar a cena continuamente.
- Em cenas estáticas, só há necessidade de redesenhar quando a câmara se move.
- Para evitar a utilização desnecessária de recursos do sistema utiliza-se *glutPostRedisplay* sempre que a câmara se mova.
- Esta função irá pedir ao GLUT que refresque o conteúdo da janela.



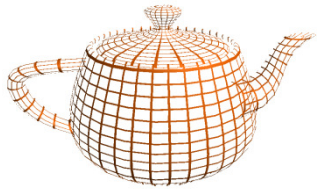
Modelação 3D

- Definição de um ponto em 3D

  glVertex3f(x,y,z);

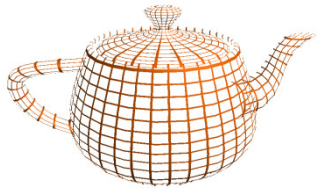
- Para desenhar triângulos fazer:

```
glBegin(GL_TRIANGLES);
        glVertex3f(0.0f, 0.0f, 10.0f);
        glVertex3f(1.0f, 1.0f, 10.0f);
        glVertex3f(0.0f, 1.0f, 10.0f);
glEnd();
```



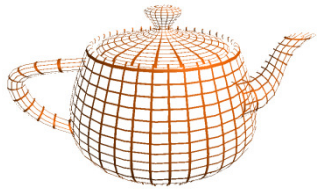
Modelação 3D

- Orientação dos polígonos
  - Por omissão, os pontos devem ser definidos pela ordem inversa à do movimento dos ponteiros do relógio



Modelação 3D

- *Back Face Culling*
  - `glEnable(GL_CULL_FACE);`
  - `glCullFace(GL_FRONT ou GL_BACK);`
- Podemos definir a orientação para a frente dos polígonos
  - `glFrontFace(GL_CW ou GL_CCW);`

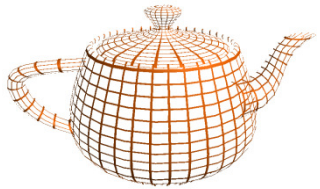


Modelação 3D

- `glPolygonMode(face,modo);`

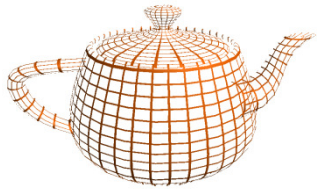
- face:
  - GL_FRONT;GL_BACK;GL_FRONT_AND_BACK

- modo:
  - GL_LINE;GL_FILL;GL_POINT



Algumas Funções Necessárias

- glTranslatef(x,y,z);
- glRotatef(alpha,x,y,z); // ângulo em graus
- glColor3f(r,g,b);
- gluLookAt(px,py,pz, lx,ly,lz, ux,uy,uz);



Exercício

- Completar o esqueleto fornecido de modo a criar uma aplicação interactiva que desenhe 1 pirâmide (uma face de cada cor).
- O teclado deve permitir mover a pirâmide no plano XZ, rodá-la em torno do eixo dos YY, e ainda alterar a altura da pirâmide.
- Testar glutPostRedisplay;
- Deve ser criado um menu para escolha da cor das faces da pirâmide.