OpenGL, GLU e GLUT

http://www.opengl.org/resources
http://pyopengl.sourceforge.net

Carlos Hitoshi Morimoto (hitoshi@ime.usp.br) - DCC/IME

GLUT (OpenGL Utility Toolkit)

- O GLUT é uma biblioteca de rotinas independentes do SO, que fornecem as ferramentas necessárias para a requisição de janelas e registro de eventos.
- Instale o freeglut
  - http://freeglut.sourceforge.net/

OpenGL

- Open Graphics Library
  - Desenvolvida pela SGI
  -
  - é uma API gráfica que permite a criação de imagens através da definição de objetos (2D/3D) usando um conjunto de formas primitivas geométricas e rotinas para manipulação.

Linguagem

A API do OpenGL foi projetada para ser usada com C ou C++

PyOpenGL: Python binding para OpenGL
- é mais lento
- mas é mais fácil de aprender/usar

OpenGL

Foi desenvolvido para ser INDEPENDENTE do sistema operacional (SO) e do gerenciador de janelas.

Por isso o OpenGL NÃO oferece rotinas de display ou interface (pois requerem E/S)

Esqueleto de
um programa
em PyOpenGL

```
•# Definicao das funcoes de callback
•
•import sys
 glutInit(sys.argv)
 glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
 glutInitWindowSize(200,200)
 glutCreateWindow("Nome da Janela")

 # Inicializar Callbacks
 # myInit()

 glutMainLoop()
```

Callbacks mais comuns

events	callback	Prototipo
(Re)Display	glutDisplayFunc	display()
(Re)size Window	glutReshapeFunc	resize(w, h)
Idle event	glutIdleFunc	idle()
Mouse button	glutMouseFunc	mouse(bt, st, x, y)
Mouse motion	glutPassiveMotionFunc	motion(x, y)
Keyboard Key	glutKeyboardFunc	keyboard(c, x, y)

Rotinas de inicialização do GLUT

import sys
glutInit(sys.argv)
- Em C: glutInit (int argc, char** argv)

glutInitDisplayMode()
- single ou double buffering
  - GLUT_SINGLE ou GLUT_DOUBLE (front/back)
- Como as cores são exibidas
  - GLUT_RGB ou GLUT_RGBA
- Buffer de profundidade
  - GLUT_DEPTH

Exemplo de callbacks

```
def display ():
    # sua rotina de (re)display

def mouse( b, s, x, y):
    # sua rotina de tratamento do mouse

if __name__ == "__main__":

    # esqueleto do programa
    # inicializar Callbacks
    glutDisplayFunc(display);
    glutMouseFunc(mouse);
    – ...
```

```
·GlutInitWindowSize( largura, altura )
      ·# sugestão de tamanho ao WM
.
·GlutInitWindowPosition( x , y );
      ·# sugestão de posição ao WM
.
·GlutCreateWindow( "nome da janela" );
·glutMainLoop();
```

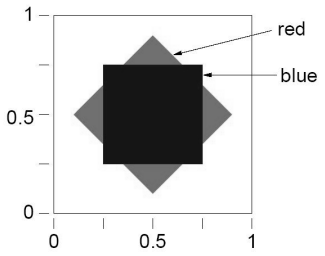
display() e reshape()

- A rotina de display é sempre necessária.
  - pode ser chamada pelo sistema ou pelo próprio programa.
    - Sistema: quando a janela é criada ou alguma outra "sai de cima".
    - Programa: animação
      - usar a rotina glutPostRedisplay(), pois outros eventos podem estar requerendo redisplay.
- Reshape() é chamada sempre que a janela muda de tamanho (tbm quando é criada).

mouse() e keyboard()

- mouse(but, st, x, y)
  - **but:** GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON
  - **st:** GLUT_DOWN, GLUT_UP
  - **(x,y):** coordenada onde o mouse foi clicado, relativo ao canto superior esquerdo da janela.
- keyboard(c, x, y): rotina chamada sempre que o teclado é acionado. c recebe o caracter, e (x,y) a posição do mouse (convenção X-window).

Exemplo de uma rotina de display()



Desenho básico

- Antes de desenhar uma cena, o OpenGL precisa saber sobre os objetos a serem desenhados, a forma de projeção, e sobre os reflexos e sombreamento.
- Apenas como exemplo, considere apenas objetos 2D, sem reflexos ou sombreamento, e com pouca interação.

Descrição das funções

- janela virtual
  - quadrado sobre o intervalo real [0,1]
- glClear: limpa a janela com a cor definida por glClearColor(float R, float G, float B, float A);
- O argumento de glClear define o buffer a ser limpo (GL_DEPTH_BUFFER_BIT, etc).

Exemplo de uma rotina de display()

```
def display():
  glClearColor(1.0, 1.0, 1.0, 1.0)
  glClear(GL_COLOR_BUFFER_BIT)

  glColor3f(1.0, 0.0, 0.0)
  glBegin(GL_POLYGON)
     glVertex2f(0.90, 0.50)
     glVertex2f(0.50, 0.90)
     glVertex2f(0.10, 0.50)
     glVertex2f(0.50, 0.10)
  glEnd()

  glColor3f(0.0, 0.0, 1.0)
  glRectf(0.25, 0.25, 0.75, 0.75)
  glutSwapBuffers()
```

glColor3f: recebe 3 floats, RGB

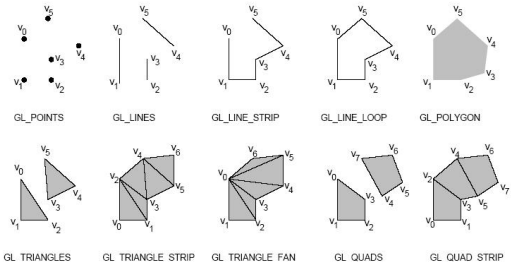
- glColor3ui, glColor3d, glColor3fv, etc.
- floats e doubles variam entre [0,1]
- para tipos inteiros (char, int, short, long) o valor pode variar até o máximo permitido

dica: no PyOpenGL use glColor

- essa função "escolhe" a opção apropriada

Entendendo melhor
o OpenGL

Exemplos de objetos no OpenGL



- OpenGL funciona como uma máquina de estados. Uma vez definido um atributo (cor, tamanho de um ponto, largura de linha, etc), ele desenha usando esses atributos.
- O OpenGL pode desenhar diversos tipos de objetos. O mais simples é o glRectf(), que desenha um retângulo.
- Objetos complexos são definidos com o auxílio das funções glBegin(mode) e glEnd(). A lista de vértices entre esses comandos define o objeto.
- O tipo do objeto é definido pelo argumento de glBegin.
  - GL_POINTS, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP, GL_POLYGON

Desenhando em OpenGL

- Apenas para o momento, vamos ignorar os efeitos de luz e sombreamento
- Acabamos de ver como podemos criar objetos em OpenGL. Vamos utilizar uma região de desenho virtual para criar os objetos
- Mas:
  - - como definir os parâmetros da câmera?
  - - E como o OpenGL mapeia a figura na região virtual para a janela real (como ele mostra a imagem)?

- glVertex2f
- glRectf
- Outros atributos possíveis:
  - glPointSize()
  - glLineWidth()
  - glLineStipple()
  - glPolygonStipple()
- 2D/3D?
- caso não especificado, um ponto é criado com z=0.
- glRect sempre desenha no plano z=0

Viewport

- Primeiro, precisamos definir onde o desenho será exibido.
- O viewport corresponde a uma região na janela realmente criada, onde o nosso desenho é exibido.
- Sempre que a janela muda de tamanho/forma, não é necessário mudar o desenho, apenas o viewport.
- Em geral, a região corresponde a toda a janela, mas nem sempre esse é o caso

Viewport

•glutInitWindowSize()
  • fornece apenas uma "sugestão" de tamanho

•def reshape(width, height):
    glViewport (0, 0, width, height)

•o tamanho real da janela em pixels é passado pela rotina reshape.
•protótipo: glViewport(x, y, w, h) # inteiros
  – (x,y) define o canto inferior esquerdo
  – (w,h) define a largura e altura do viewport em pixels

Comandos básicos

·glMatrixMode(mode);
  •mode= GL_MODELVIEW (default), GL_PROJECTION, GL_TEXTURE
  •define o tipo de matriz a ser utilizada
·glLoadIdentity();
·glLoadMatrixf(M);
·glMultMatrixf(M);
·glPushMatrix() and glPopMatrix();

Projeção do desenho sobre a imagem

- Agora sabemos desenhar em um espaço virtual, e definir o tamanho da região onde queremos exibir o desenho (gerar uma imagem)
  - Mas como a gente mapeia o desenho para a imagem?
  - Essa transformação é conhecida como projeção
- O OpenGL mantem 3 conjuntos de matrizes de transformação:
  - ***Modelview matrix:*** transformação de objetos
  - ***Projection matrix:*** transformações perspectiva e paralela
  - ***Texture matrix:*** define como as texturas são mapeadas no objeto

Projeções

- Para projetar um objeto 3D sobre o plano de imagem, é necessário:
  - 1. representar o objeto 3D em um sistema de coordenadas apropriado (para o OpenGL, centrado na câmera).
    - Modelview matrix: transforma pontos entre sistemas de coordenadas
  - 2. projetar os pontos
    - Projection matrix: define o tipo de projeção e a região a ser projetada

Operação de Matrizes em OpenGL

· Para cada um dos 3 tipos, o OpenGL mantem uma pilha de matrizes, sendo que a matriz no topo da pilha é aplicada primeiro.

· O OpenGL permite que uma matriz seja empilhada e desempilhada.

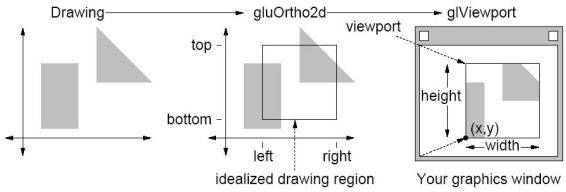
· O modo inicial é ModelView, e em geral se assume que essa é a matriz sendo utilizada. Portanto, sempre que mudar de modo, certifique-se de retornar ao ModelView.

Caso simples: Cena 2D

- Uma cena 2D não requer projeção, mas precisamos definir a região retangular no espaço do desenho virtual que contem a parte a ser exibida no viewport
  - gluOrtho2d(left, right, bottom, top)
  - exemplo de programa:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0, 1.0, 0.0, 1.0);
glMatrixMode(GL_MODELVIEW);
```

Projeção e Viewport



O que você deve saber

- Glut
  - Esqueleto de um programa (em PyOpenGL)
  - Callbacks
  - Animação
  - Menus
- OpenGL
  - Desenho como máquina de estados
  - Viewport
  - Matrizes de Transformação

Menus no GLUT

- Veja mais no paper OpenGL and X Column 1, e no manual do GLUT.
- Os programas gráficos devem permitir ao usuário alterar os seus modos de operação
- Os menus são instrumentos simples que permitem esse tipo de interação do programa com o usuário.
- No GLUT, os menus podem ser criados, modificados, e associados a um botão do mouse (funções básicas).
- Quando associado a um botão, o menu aparece quando o botão é pressionado.

Comandos básicos

```
submenu = glutCreateMenu(quit_menu);
glutAddMenuEntry("Continue", 1);
glutAddMenuEntry("Realy Quit!", 2);
glutCreateMenu(main_menu);
glutAddMenuEntry("OK", 1);
glutAddSubMenu("Quit", submenu);
glutAttachMenu(GLUT_RIGHT_ BUTTON);
```

```
def main_menu(val):
    if (val == 1):
      return
    glutPostRedisplay()
                »
            »
```

```
def quit_menu(val):
      if val == 1:
          return
      if val == 2:
          sys.exit(0)
      glutPostRedisplay()
```