Desenho de uma matriz de retângulos

Serve de base para o exercício das cores

Leandro Tonietto
Processamento Gráfico - Unisinos
ltonietto@unisinos.br
http://professor.unisinos.br/ltonietto

Sumário

- Estrutura geral do código
- Definição de uma câmera (projeção)
- Definições do main e call-backs da GLUT
- Desenho da matriz
- Função para desenho de um retângulo
- Tratamento de eventos
- Dicas para o exercícios do jogo das cores

Estrutura geral do código

- O código básico de um programa simples em OpenGL poderia ter:
  - método `main` para inicialização da GLUT, passagem das funções de *call-back* (desenho da tela, tratamento de teclado, tratamento de mouse e etc.), inicialização das variáveis e objetos do programa e chamada para o *main loop* (`glutMainLoop()`).
  - método `init`, invocado a partir do `main`, para fazer as inicializações dos dados e das propriedades da câmera.
    - `glClearColor(r,g,b); // cor de fundo`
    - `glOrtho(xi,xf,yi,yf); // limites da visualização`
    - `glViewport(x,y,w,h); // define pos e tamanho janela`
  - método de desenho da tela (`display`, p.e.), que contém as chamadas da OpenGL para desenhar todos os retângulos
  - método `drawRect` para desenhar apenas um retângulo na tela.

Funções de *call-back* para a GLUT

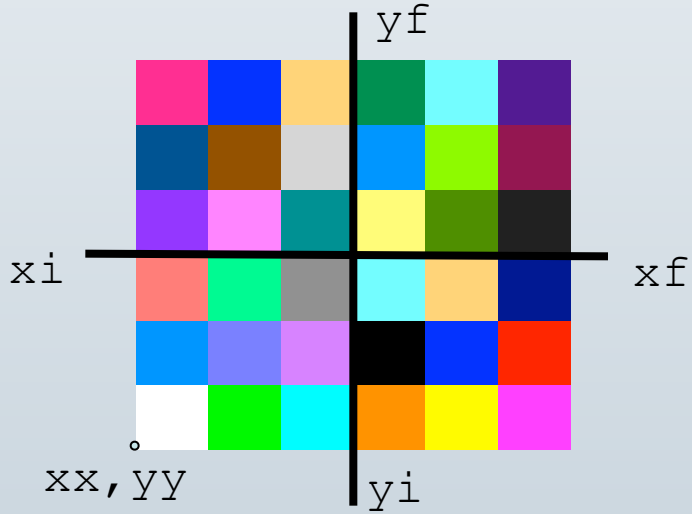
- Implemente outras funções para auxílio do programa, como call-backs para eventos de mouse, teclado e redimensionamento de tela:

```
int main(int argc, char** argv){

 // inicialização da GLUT
 glutInit(&argc, argv);
 glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
 glutInitWindowSize (width, height);
 glutInitWindowPosition (0, 0);
 glutCreateWindow ("Jogo das Cores");
 init ();

 // funções de call-back para a GLUT. Devemos passar o nome da função que
 // criamos para a GLUT. Os parâmetros estar no padrão (veja slides)
 glutDisplayFunc(display);   // função de desenho
 glutReshapeFunc(reshape);   // tratamento de redimensionamento da janela
 glutKeyboardFunc(keyboard); // tratamento de teclado
 glutMouseFunc(mouse);       // tratamento de mouse

 // inicializações do programa
 init();

 // Inicia loop do programa, o main loop
 glutMainLoop();
  return 0;
}
```

Deve-se implementar também uma função para inicialização do jogo. Nesta função, é interessante colocar o código que gera a cor aleatória para cada retângulo. O sorteio da cor deve ser feito por canal (R,G e B):
b = rand() % 256;

Método de desenho geral (matriz)

- Este método possui a visão geral sobre o desenho da cena e é invocado pela GLUT sempre for necessário redesenhar a tela:

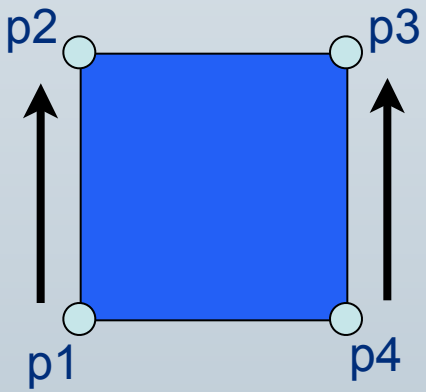



```
void display(void){
  glClear(GL_COLOR_BUFFER_BIT);
  //... outras definições
  float xx;
  float yy = yi; // y inicial da tela
  glBegin(GL_QUADS);
  for (int y=0; y < rows; y++) {
     xx = xi; // x inicial da tela
     for (int x = 0; x < cols; x++) {
        // definir qual é cor (r,g,b)...

        drawRect(xx, yy, w, h, r,g,b); // w e h são as dimensões do retângulo
        xx += w; // xx de base para o primeiro vértice do próximo retângulo
     }
     yy += h; // yy de base para o primeiro vértice do próximo retângulo
  }
  glEnd();
  glFlush(); // ou glutSwapBuffers();
}
```

Método de desenho de um retangulo

- Este método deve desenhar um retângulo na tela, de acordo com os parâmetros definidos:

```
void drawRect(float x, float y, float w, float h, float r, float g, float b){
  glColor(r, g, b);
  glVertex2d(x, y);          // p1
  glVertex2d(x, y + h);      // p2
  glVertex2d(x + w, y + h);// p3
  glVertex2d(x + w, y);      // p4
}
```



Desenho de texto na tela

- Para desenhar um texto no OpenGL precisamos de uma biblioteca auxiliar, uma vez que um texto não é uma primitiva gráfica.
- A biblioteca converte uma letra uma representação bitmap (imagem) para ser desenhada na tela e numa determinada posição. A GLUT, por exemplo, faz isto
  - Lembre-se: cada caractere deve ser desenhado numa nova posição x, subseqüente.

```
float drawText(char *msg, int msgLength, float x, float y, float charWidth){
   for (int i = 0; i < msgLength; i++) {
     glRasterPos2d(x, y);
     // A fonte, neste caso, é uma constante da GLUT. Que pode ser um param
     glutBitmapCharacter(GLUT_BITMAP_9_BY_15, *(msg++)); // inc ponteiro msg
     x += charWidth;
   }
   return x;
 }
```

Tratamento de eventos

- Implemente outras funções para auxílio do programa, como *call-backs* para eventos de mouse, teclado e redimensionamento de tela são úteis para que possamos implementar a lógica do programa:
  - Lembre-se: a GLUT vai gerenciar o loop principal do programa. O trabalho do programador agora é lidar com os eventos para mudar o estado do jogo.

```
void mouse(int button, int state, int x, int y){
  //faz alguma coisa dado que algum botão (button) foi pressionado,
  //um estado do botão (state) e a posição de tela (x,y) que foi clicada.
  //Não é coordenada do OpenGL, portanto, devemos converter o
  //clique de tela em coordenada do OpenGL. Dica: y é invertido.
}
void keyboard(unsigned char key, int x, int y){
  //faz alguma coisa dado que alguma tecla (key) foi pressionada,
  //e a posição de tela atual (x,y) do mouse.
  //Não é coordenada do OpenGL, portanto, devemos converter o
  //clique de tela em coordenada do OpenGL. Dica: y é invertido.
}
void reshape(int w, int h) {
  // redefinição da viewport e de proporções da tela, quando o usuário
  // aumenta ou diminui a tela do programa.
  // w e h são as novas dimensões da janela
  glViewport (0, 0, width=w, height=h);
 }
```

Tratando clique na tela

- O clique do mouse informado pela GLUT esta em coordenadas de tela e precisamos converter para coordenadas do OpenGL:

```
void mouse(int button, int state, int x, int y){

   // width é a largura da janela em pixels (ver glViewport)
   float xx = x / (float) width; // normaliza click: xx = [0..1)

   // transformar xx em coordenadas da janela OpenGL:
   // xi é a coordenada inicial da janela e w = xf - xi (largura da janela)
   // Veja as definições de janela no comando glOrtho
   xx = xi + xx * w; // xx está em coordenadas do OpenGL (xx=[xi..xf));


   // para o y temos que considerar que o sistema OpenGL o y cresce para cima
   // e no da tela (que veio o click) o y cresce para baixo. Então, devemos
   // primeiro inverter o y e depois convertê-lo para janela OpenGL
   // height é a altura da janela em pixels (ver glViewport)
   y = height - y;
   float yy = y / (float) height; // normaliza click: yy = [0..1)

   // transformar xx em coordenadas da janela OpenGL:
   // xi é a coordenada inicial da janela e h = yf - yi (altura da janela)
   // Veja as definições de janela no comando glOrtho
   yy = yi + yy * h; // yy está em coordenadas do OpenGL (yy=[yi..yf));

}
```

Outras dicas

- No caso dos retângulos para o jogo, é bom implementar uma classe para o objeto retângulo, contendo, um estato do retângulo (para saber se desenha ele ou não e também se ele deve entrar na contagem de pontos do round) e a cor que ele está sendo desenhado.
- No método init, você pode randomizar as cores dos retângulos e também inicializa-los com `visible=true` (estado do retâgulo).
  - É bom criar um método reset para redefinir as cores e estados do retângulo, para permitir ao usuário jogar novamente.
- Também não esqueça do contador de rounds para saber o fim do jogo.
- No fim, informe a pontuação total ao jogador.