Aula – Computação Gráfica

OpenGL 3D

1

O Fluxo Gráfico Padrão

- Malha de objetos 3D (geralmente com triângulos ou quadrados)
- Aplicação de propriedades dos materiais
- Mapeamento de textura de conforme necessário
- Iluminação da cena
- Posicionamento da câmera

2

Por que usar OpenGL para 3D?

- Amplamente utilizado na indústria e na academia
- Usaremos API de função fixa (Fixed-function API, OpenGL 1.x)
  - Permite prototipação rápida
  - Permite utilização de efeitos clássicos de iluminação
  - Abstrai a programação paralela para placas gráficas

3

OpenGL 3D

- Utiliza a mesma plataforma do 2D
- Modificações devem ser feitas para se trabalhar em 3D
  - Habilitar tratamento de profundidade
  - Definir objetos
  - Definir fontes de Luz
  - Definir a câmera

4

4

Tratamento de Profundidade

- Permite o OpenGL definir qual objeto está na frente
  - Incluindo suas partes
- A teoria será vista em aulas futuras
- A habilitação deve ser feita em 3 partes do código
  - Na inicialização

```
glEnable(GL_DEPTH_TEST); //Habilitação do teste de profundidade
```

  - Na callback de renderização

```
glClear ( GL_COLOR_BUFFER_BIT |
          GL_DEPTH_BUFFER_BIT); //Limpeza do buffer de profundidade
```

  - Na criação da janela

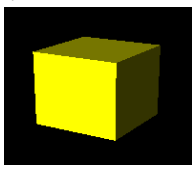
```
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB |
          GLUT_DEPTH); //Criação do buffer
```

5

5

Polígono 3D

- Especificação de material
  - Descreve as propriedades refletivas da luz (cor, brilho, etc.)



```
float materialColor[] = { 1.0, 1.0, 0.0, 1.0}; //Amarelo
glMaterialfv(       GL_FRONT, //Indica a face frontal do objeto
                    GL_DIFFUSE, //Indica a componente difusa da luz
                    materialColor); //Passa os valores de cor R/G/B/A
```

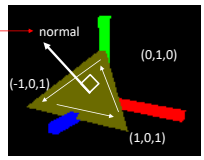
6

6

Polígono 3D

- OpenGL usa o sistema de coordenadas da mão direita

Normal para fins de Frontal e Trazeiro



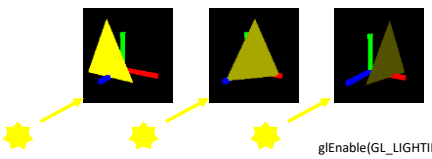

```
glBegin(GL_TRIANGLES);
   glVertex3f(0, 1, 0);
   glVertex3f(-1, 0, 1);
   glVertex3f(1, 0, 1);
glEnd();
```

7

7

Iluminação

- Iluminação no mundo real é complexa
  - Necessita de muitas interações entre raios e objetos
- Modelagem é feita via aproximações
  - Modelos clássicos de iluminação datam dos anos 70
- Ainda hoje, uma modelagem completa está além do calculável



```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
```

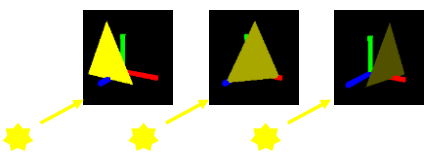
8

8

Iluminação

- Definição da fonte de luz

```
float light_position[] = { 0.0, 0.0, 10.0, 1.0 }; //Luz posicionada ao longo do eixo z
glLightfv(          GL_LIGHT0, //Indica que esta definindo a fonte de luz 0
                    GL_POSITION, //Indica que esta definindo a posição
                    light_position); //Passa a posição
```



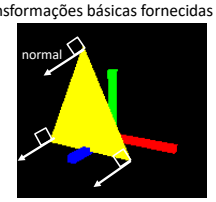
9

9

Normais 3D para Iluminação

- As normais para o cálculo da iluminação precisam ser definidas
- Normais definidas de forma errada ou incoerente
  - Causam comportamentos estranhos nos objetos
- Sugestão
  - Defina a normal em um sistema de coordenadas conhecido
  - Modifique através das transformações básicas fornecidas

```
glBegin(GL_TRIANGLES);
    glVertex3f(0, 2, 1);
    glNormal3f(0, 0, 1);
    glVertex3f(-1, 0, 1);
    glNormal3f(0, 0, 1);
    glVertex3f(1, 0, 1);
    glNormal3f(0, 0, 1);
glEnd();
```

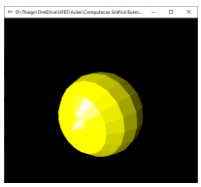


10

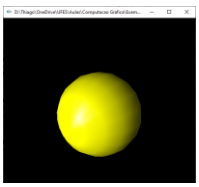
10

Shading

- Objetivo final da iluminação da cena é permitir o cálculo de cor de cada pixel
- O modelo de iluminação é caro para recomputar
  - Shading permite aproximar/simplificar o cálculo por pixel



glShadeModel (GL_FLAT);



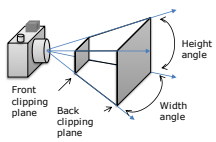
glShadeModel (GL_SMOOTH);

11

11

Câmera Virtual

- A câmera padrão do OpenGL está
  - Posicionada na origem
  - Olhando para o negativo do eixo z
  - Com o up alinhado com o eixo y
- Seria inconveniente desenhar tudo nessas coordenadas
  - Com isso, é necessário ajustar a câmera
- A câmera é definida por
  - Posição
  - Direção
  - Orientação
  - Planos de corte (frontal e trazeiro)
  - Ângulo de abertura (width)
  - Aspect ratio



12

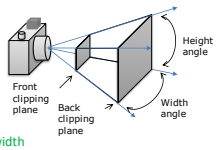
12

Câmera Virtual

- Posição, Direção e Orientação

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(      3,2,5,  // Posição
                0,0,0,  // Direção para onde está olhando
                0,1,0); // Orientação (up)
```

- Planos de corte (frontal e trazeiro)
- Ângulo de abertura (width)
- Aspect ratio




```
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
gluPerspective (    90, // Ângulo de abertura width
                    1, // Aspect ratio ()
                    1, // Distância do plano de corte frontal (near)
                    15); // Distância do plano de corte trazeiro (far)
```

13

13

Transformação Genérica em OpenGL

- OpenGL provê uma função para multiplicar uma matriz qualquer na pilha de transformações
  - *glMultMatrixf*
  - Ela recebe como parâmetro um ponteiro para a primeira posição de um vetor de 16 posições representando uma matriz 4x4
  - OpenGL usa a transposta em sua representação
    - Entao, em um array M[4][4], temos M[C][L], sendo C o índice da coluna e L o da linha se considerar a matriz vista nas aulas anteriores

```
GLfloat M[4][4] = {     // Matriz genérica de transformacao configurada para uma
    1,  0,  0,  0, // translacao de 10 em x, 20 em y e 30 em z
    0,  1,  0,  0, // Em OpenGL ela é representada pela transposta
    0,  0,  1,  0, // Portanto, os itens referentes a parte da translacao estão na
    10, 20, 30,  1 // ultima linha
    };
glMultMatrixf(&M[0][0]); // Multiplica a matriz atual A da pilha pela matriz M, v'= A M v
```

14

14

Perguntas ?????

15

15