Fundamentos Matemáticos para Computação Gráfica

Márcio Sarroglia Pinho
Isabel Harb Manssour

SEQUÊNCIA DE TRANSFORMAÇÕES GEOMÉTRICAS

Sequência de Transformações Geométricas

- Representação Tradicional de um Objeto

```
typedef struct Objeto {
    Ponto Pos, Rot, Esc;
    setRot(.....);
    setPos(.....);
    setEsc(.....);
    Draw();
} ;
Objeto Carro, Pessoa;
```


Sequência de Transformações Geométricas

```
void Objeto::Draw() {
   glPushMatrix();
        glTranslatef (TX,TY,TZ);
        glScalef(EX,EY,EZ);
        glRotatef(AngX,1,0,0);
        glRotatef(AngY,0,1,0);
        glRotatef(AngZ,0,0,1);
        DesenhaCarro();
    glPopMatrix();
}
```

COMO ALTERAR A ORDEM DAS TRANSFORMAÇÕES ??

COMO ACRESCENTAR MAIS UMA TRANSFORMAÇÃO ??

COMO ESCOLHER UM EIXO DE ROTAÇÃO ??

Sequência de Transformações Geométricas

- Utilizam-se Matrizes de Transformação
- glTranslate, glRotate ou glScale, são sempre armazenadas em uma matriz chamada MODELVIEW

glTranslatef(-3, 2, -8)
glVertex3f(10, 15, 20)

$$\begin{bmatrix}10 & 5 & 20 & 1\end{bmatrix} \times \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ -3 & 2 & -8 & 1\end{bmatrix} == \begin{bmatrix}7 & 17 & 12 & 1\end{bmatrix}$$

Sequência de Transformações Geométricas

- Nova representação dos objetos

```
typedef struct Objeto {
   GLFloat M[4][4];
   void Rotate(.....);
   void Translate(.....);
   void Scale(.....);
   Draw();
} ;
Objeto Carro, Pessoa;
```

Os métodos SET se alteram pois não há mais parâmetros de instanciamento

Sequência de Transformações Geométricas

- Novo método SetRot(ang, ax, ay, az)

```
void Objeto::Rotate() {
    glPushMatrix();
    // carrega a matriz do objeto na OpenGL
        glLoadMatrixf(&M[0][0]);
    // altera a matriz fazendo a rotação
        glRotated(ang, ax, ay, az);
    // guarda a matriz novamente no objeto
        glGetFloatv(GL_MODELVIEW_MATRIX, &M[0][0]);
    glPopMatrix();
}
```


Sequência de Transformações Geométricas

- Novo método Draw()

```
void objeto::Draw{
  glPushMatrix();
    // carrega a matriz do objeto na OpenGL
      glLoadMatrix (&M[0][0]);
    // faz o desenho
      DesenhaCarro();
    glPopMatrix();
}
```

Sequência de Transformações Geométricas

- Funções OpenGL para Matrizes de Transformação
  - glLoadIdentity(): matriz de transformação converte-se na matriz identidade
  - glGetFloatv (GL_MODELVIEW_MATRIX, Matriz): obtém a matriz corrente
  - glLoadMatrixf(Matrix): define uma nova matriz de transformação
  - glMultMatrixf(Matrix): multiplica a matriz de transformação atual por uma segunda matriz

TRANSFORMAÇÕES GEOMÉTRICAS HIERÁRQUICAS

Transformações Geométricas Hierárquicas

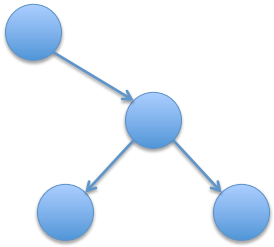
```
void DesenhaCena ()
{
   Carro.Draw();
   Pessoa.Draw();
}
```

COMO MOVER A PESSOA JUNTO COM O CARRO ??

GRAFO DE CENA

Transformações Geométricas Hierárquicas

- Grafo de Cena
  - Nodos
    - Objetos
    - Transformações
  - Arcos
    - Relações entre objetos
  - Permite a composição de transformações Geométricas
  - Permite modelar relações de hierarquia entre objetos



Transformações Geométricas Hierárquicas

- Nova representação dos objetos

```
typedef struct Objeto {
   GLFloat M[4][4];
   Objeto ListaDeFilhos[…];
   setRot(…..);
   setPos(…..);
   setEsc(…..);
   Draw();
} ;
```


Transformações Geométricas Hierárquicas

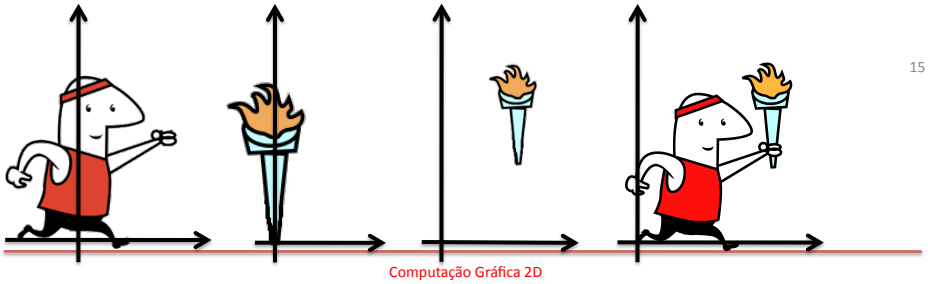
- Novo método Draw

```
void objeto::Draw{
   glPushMatrix();
      glMultMatrixf(&M[0][0]);      // glLoadMatrix ??
      DesenhaCarro();
      while(ListaDeFilhos[i])
         ListaDeFilhos[i].Draw();
   glPopMatrix();
}
```

Se usarmos glLoadMatrix, as transformações anteriores deixariam de afetar os objetos-filho

Transformações Geométricas Hierárquicas

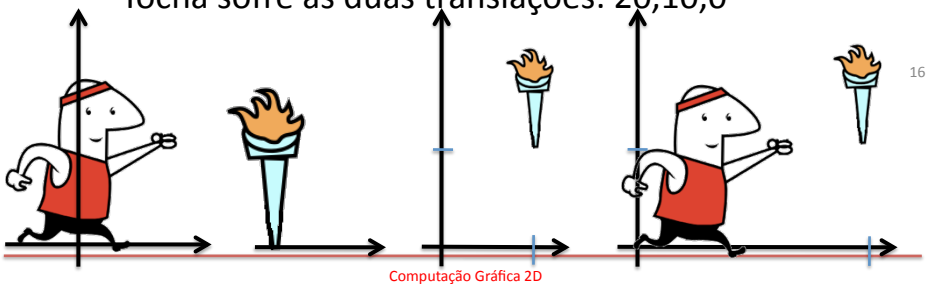
- Como anexar um objeto a outro ?
  - Transladar a tocha para sua posição
  - Anexar a tocha como 'filha' do atleta
- No momento da anexação a tocha irá 'saltar'



15

Transformações Geométricas Hierárquicas

- Antes da Anexação
  - Tocha: Translação de 10,10,0
  - Atleta: Translação de 10, 0, 0
- Após a anexação
  - Tocha sofre as duas translações: 20,10,0



16

Transformações Geométricas Hierárquicas

- Para evitar o 'salto'
  - É preciso **desfazer** a transformação do 'pai' antes de anexar o 'filho' **- Antes da Anexação**
  - Para isto aplica-se a inversa da transformação do pai ao filho

$$MFilho = MFilho \times MPai^{-1}$$

- Com isto, ao desenhar o filho, ele fica somente com a sua própria transformação

$$MFilho = MFilho \times MPai^{-1} \times MPai$$

CÁLCULO DAS COORDENADAS DE UM PONTO

Cálculo das coordenadas de um ponto

- Instanciamento
  - Como calcular as coordenadas GLOBAIS de um ponto do objeto após a execução de uma sequência de transformações?

$$CoordGlobal = CoordLocal \times MatrizGlobal$$

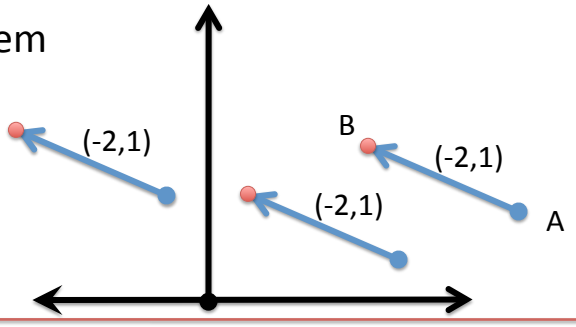
$$MatrizGlobal = MPai \times MPai \times .... \times MFilho$$

```
GLfloat MG[4][4];  // matriz global
glGetFloatv(GL_MODELVIEW_MATRIX, &MG[0][0]);
XGlobal = MG [0][0] * X + MG[1][0] * Y+MG[2][0] * Z +MG[3][0];
YGlobal = MG [0][1] * X + MG[1][1] * Y+MG[2][0] * Z +MG[3][1];
ZGlobal = MG [0][2] * X + MG[1][2] * Y+MG[2][0] * Z +MG[3][2];
```


VETORES, RETAS, ETC...

Vetores

- V = B-A
- Definem uma direção
- Possuem um tamanho(módulo)
- Não tem origem



Vetores

- Operações
  - Módulo

$$|V| = \sqrt{x^2 + y^2 + z^2}$$

  - Versor (vetor unitário)

$$V = \frac{(x,y,z)}{|V|}$$

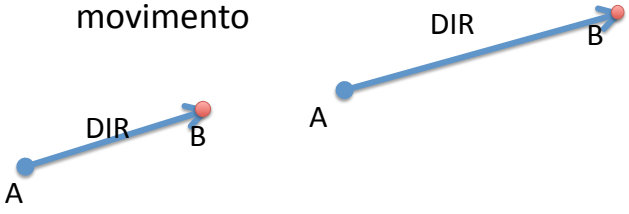
Vetores

- Operações
  - Multiplicação por um escalar
  - Permite alterar o tamanho do vetor e manter a direção

$$V(x,y,z) * n = (x * n, \quad y * n, \quad z * n)$$

Vetores

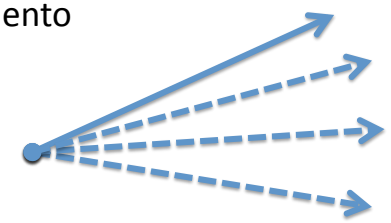
- Operações
  - A soma de um vetor a um ponto permite o deslocamento do ponto na direção do vetor
  - O tamanho do vetor fornece a velocidade do movimento



$$B = A + DIR * veloc$$

Vetores

- Operações
  - A rotação permite a mudança de direção de um deslocamento



$$xr = x * \cos(alfa) - y * sen(alfa)$$
$$yr = x * sen(alfa) + y * \cos(alfa)$$

Vetores

- Operações
  - Produto Escalar

$$V1 \bullet V2 = x1 * x2 + y1 * y1 + z1 * z2$$

  - Pode ser usado para calcular o ângulo entre 2 vetores pois, se V1 e V2 são unitários então:

$$V1 \bullet V2 = \cos(\alpha)$$

Vetores

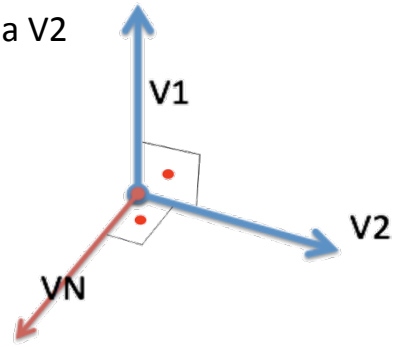
- Operações
  - Produto Vetorial V1 X V2 = (Xn,Yn,Zn)
  - Vetor perpendicular a V1 e a V2

$$V1 \times V2 = \det \begin{bmatrix} Xn & Yn & Zn \\ X1 & Y1 & Z1 \\ X2 & Y2 & Z2 \end{bmatrix}$$

$$Xn = Y1 * Z2 - Z1 * Y2$$

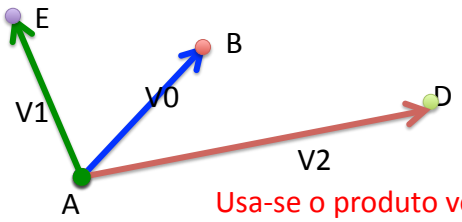
$$Yn = Z1 * X2 - X1 * Z2$$

$$Zn = X1 * Y2 - Y1 * X2$$



Vetores

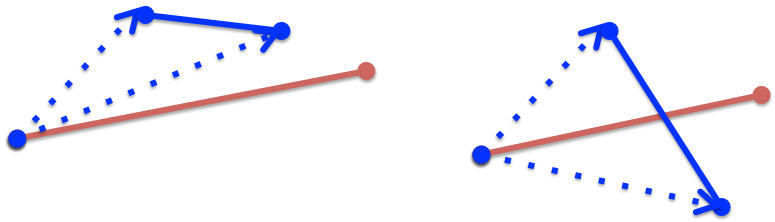
- Determinação de lateralidade
- De que lado está o ponto en relação a AB



Usa-se o produto vetorial entre os vetores
V0 x V1
V0 x V2

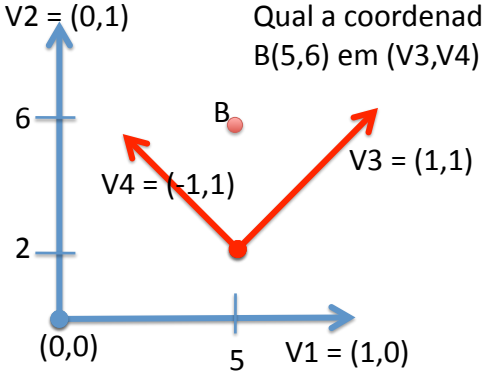
Vetores

- Determinação de lateralidade
  - Útil para saber se pode haver intersecção entre segmentos de reta



Vetores

- Definem Sistemas de Coordenadas



Qual a coordenada de B(5,6) em (V3,V4) ??

Azul: (V1,V2)

Vermelho: (V3,V4)

$$M1 = \begin{bmatrix} V1 \\ V2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$M2 = \begin{bmatrix} V3 \\ V4 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

Vetores

- O sistema de coordenadas deve ter eixos descritos por vetores unitários

$$|V3| = |V4| = \sqrt{2}$$

$$\begin{bmatrix} V3 \\ V4 \end{bmatrix} \Rightarrow \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

Vetores

- Para calcular as coordenadas de B no outro sistema faz-se a seguinte igualdade

$$O1 + M1 * B = O2 + M2 * B'$$

$$B' = (B - O2) * M2^{-1}$$

- O1 : Origem do sistema 1
- M1 : Matriz do sistema 1
- B: ponto original
- O2 : Origem do sistema 2
- M2 : Matriz do sistema 2
- B': ponto no sitema 2

$$M2^{-1} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

Vetores

- Continuando os cálculos...

$$B' = (B - O2) * M2^{-1}$$

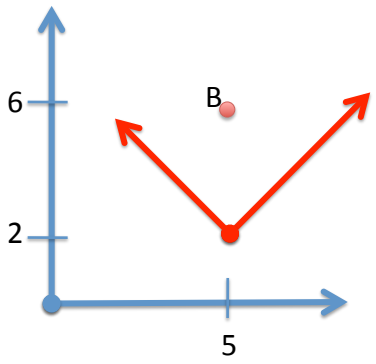
$$K = \frac{1}{\sqrt{2}} = \frac{\sqrt{2}}{2}$$

$$B' = \left(\begin{bmatrix} 5 & 6 \end{bmatrix} - \begin{bmatrix} 5 & 2 \end{bmatrix}\right) * \begin{bmatrix} K & -K \\ K & K \end{bmatrix}$$

$$B' = \begin{bmatrix} 0 & 4 \end{bmatrix} * \begin{bmatrix} K & -K \\ K & K \end{bmatrix}$$

$$B' = \begin{bmatrix} 0*K + 4*K & 0*-K + 4*K \end{bmatrix}$$

$$B' = \begin{bmatrix} 4K & 4K \end{bmatrix} \Rightarrow \begin{bmatrix} 2\sqrt{2} & 2\sqrt{2} \end{bmatrix}$$



Retas

- Equação paramétrica da reta
  - Ponto Inicial + Vetor colocado no ponto
  - Utiliza a multiplicação do um vetor por um escalar

$$R(t) = A + (B - A) * t$$

$$R(t) = A + B * t - A * t$$

$$R(t) = A * (1 - t) + B * t$$

$$t \in [0..1]$$

Retas

- Interseção entre retas

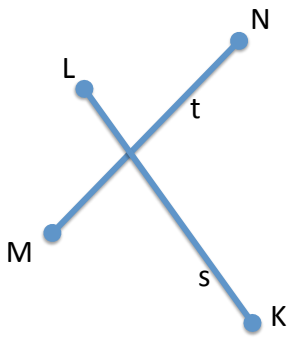
```
int intersec2d (Ponto k, Ponto l, Ponto m, Ponto n,
                double &s, double &t)
{
    double det = (n.x - m.x) * (l.y - k.y)  -  (n.y - m.y) * (l.x - k.x);
    if (det == 0.0)  return 0 ; // não há intersecção
    s = ((n.x - m.x) * (m.y - k.y) - (n.y - m.y) * (m.x - k.x))/ det ;
    t = ((l.x - k.x) * (m.y - k.y) - (l.y - k.y) * (m.x - k.x))/ det ;
    return 1;  // há intersecção
}
```

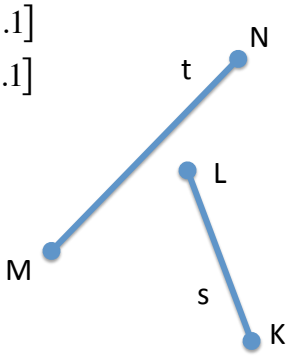

Retas

- Interseção entre retas

Condição para que haja interseção

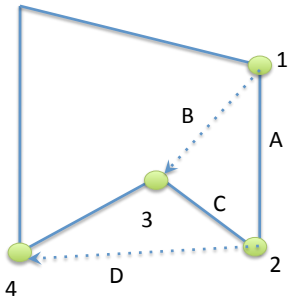
$$s \in [0..1]$$
$$t \in [0..1]$$





Polígonos

- Sequência de vértices
- Determinação de Concavidade

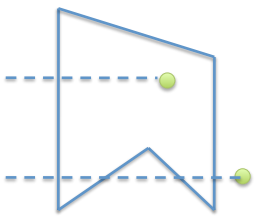


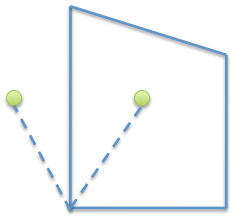
$$A \times B = (0 \quad 0 \quad Z1)$$
$$C \times D = (0 \quad 0 \quad Z2)$$

$$Z1 > 0$$
$$Z2 < 0$$

Polígonos

- Determinação de Inclusão





Côncavos
Conta-se o número de Intersecções

Convexos
Verifica-se de que lado está o ponto

FIM