Computação Gráfica
Unidade 3

prof. Dalton S. dos Reis
dalton.reis@gmail.com

FURB - Universidade Regional de Blumenau
DSC - Departamento de Sistemas e Computação
Grupo de Pesquisa em Computação Gráfica, Processamento de Imagens e Entretenimento Digital
http://www.inf.furb.br/gcg/

Unidade 03

- Interface, Transformações 2D e Seleção
  - Interface, Transformações 2D e Seleção, Programação orientada a eventos
  - Elementos de interface, Eventos e atributos de elementos de interface
  - Funções callback (teclado e mouse)
  - Transformações de sistemas de coordenadas
  - Transformações geométricas 2D, Algoritmos de seleção, Boundaring Box

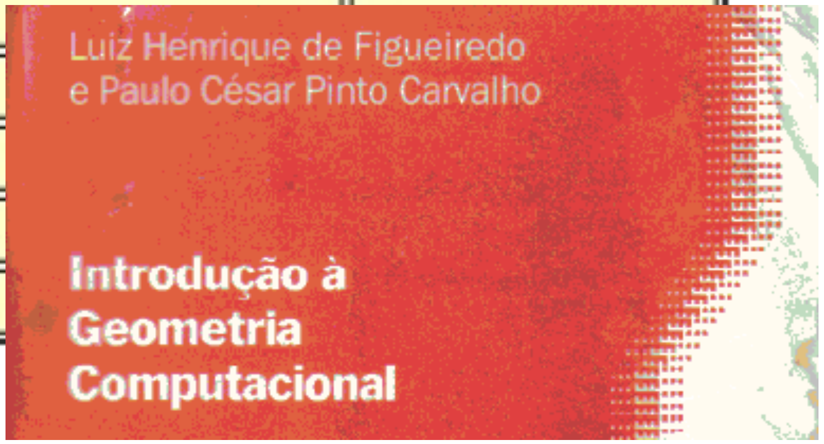
  - Objetivos Específicos
    - Demonstrar conhecimento no desenvolvimento de sistemas com interface gráfica com o usuário. Interpretar, especificar e desenvolver aplicações simples com transformações geométricas
  - Procedimentos Metodológicos
    - Aula expositiva dialogadaMaterial programado
    - Atividades em grupo (laboratório)
  - Instrumentos e Critérios de Avaliação
    - Trabalhos práticos (avaliação 3)

Algoritmo de seleção

Livro utilizado

Nº Obra Sistema	244342
Número de chamada	**516.00285, F475i, CG** **(Anote para localizar o material)**
Autor	Figueiredo, Luiz Henrique de
Título	Introdução a geometria computacional /Luiz Henrique de Figueiredo e Paulo César Pinto Carvalho. -Rio de Janeiro : IMPA, 1991. - 111p. :il.
Notas	Trabalho apresentado no 18. Colóquio Brasileiro de Matemática.
Assunto	Geometria Processamento de dados
Autor Secundário	César, Paulo,1952-
Autor Secundário	Instituto de Matemática Pura e Aplicada (Brasil).

Registro	Volume	Biblioteca	Situação	Usuário	Data Início	Data Fim
CG210707		Central	Disponível			
CG210708		Central	Empréstimo	133658	25/04/2011	02/05/2011
CG210709		Central	Disponível			
CG210710		Central	Disponível			
CG210711		Central	Disponível			
CG210712		Central	Disponível			
CG210713		Central	Disponível			



Algoritmo de seleção

O algoritmo a seguir testa se um ponto de seleção $p_{sel} = (x_{sel}, y_{sel})$ (ponto clicado pelo usuário) está dentro ou fora de um polígono não convexo P. O polígono é formado por uma lista de n vértices ou pontos $p_i = (x_i, y_i)$.

A idéia principal do algoritmo é a partir do ponto de seleção p_{sel} é disparado um "tiro". Se o tiro cruzar a fronteira do polígono uma quantidade ímpar de vezes, o ponto de seleção está dentro do polígono. Caso contrário está fora. A direção do tiro forma uma reta que chamamos de L. Para facilitar o cálculo de intersecção de L com cada lado $\overline{p_i p_{i+1}}$ do polígono, consideramos a reta L horizontal.

A figura 1 ilustra o tiro disparado sobre a reta L a partir do ponto de seleção p_{sel}. Os pontos p_i e p_{i+1} definem o lado de índice i (lado $\overline{p_i p_{i+1}}$) do polígono. Note que exitem 3 intersecções do tiro com a fronteira do polígono, indicando que p_{sel} está dentro.

Algoritmo de seleção

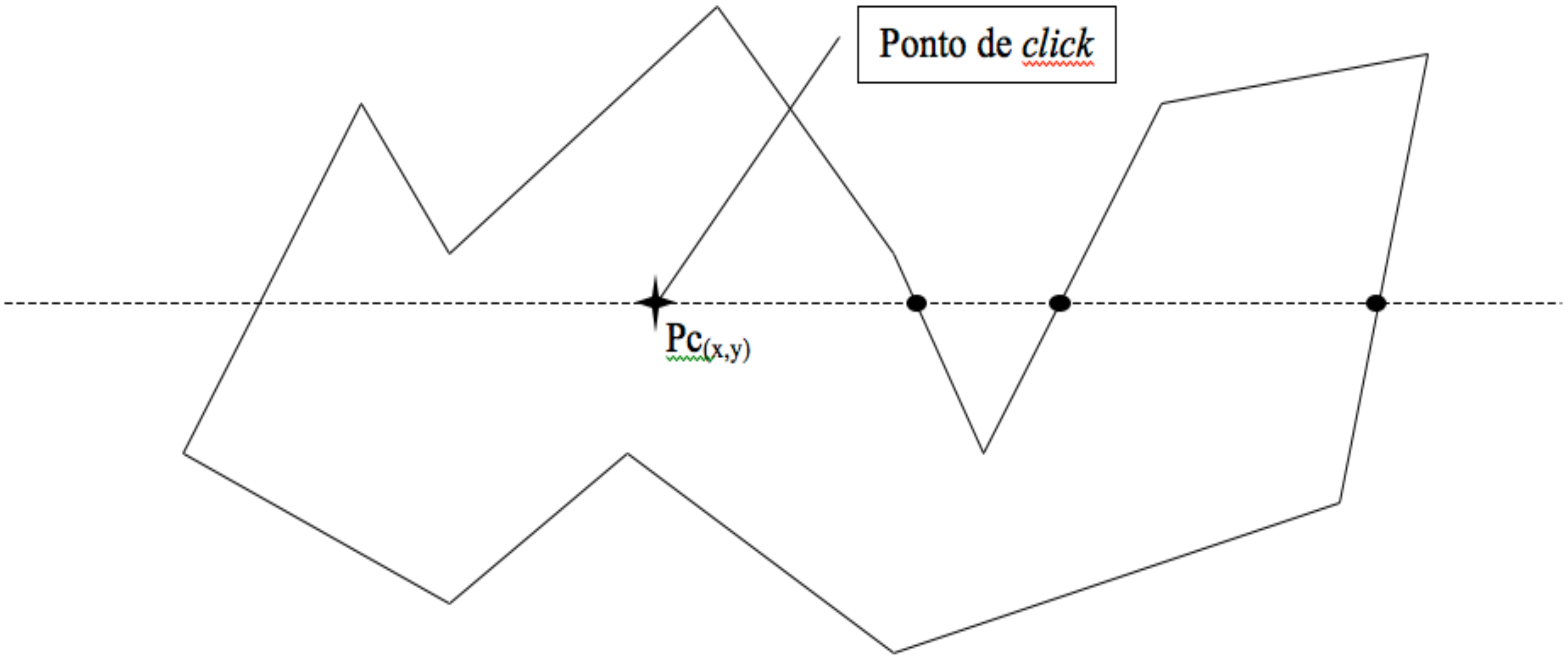


Figura 1: Ponto de seleção dentro do polígono

Agora vamos ao algoritmo. O laço principal, com contador i, refere-se a cada lado do polígono. Se o lado for horizontal, o teste de intersecção da reta L com o lado é um teste simples de comparação de cordenadas. Sempre que o ponto de seleção estiver sobre um dos lados, podemos parar o algoritmo considerando que o usuário selecionou o polígono.

Algoritmo: PontoEmPoligono(P, p_{sel})

$N_{int} \leftarrow 0$;
para i=*1* **até** n **faça**
  se $y_i \neq y_{i+1}$ **então**
    $(x_{int}, y_{int}) \leftarrow$ pto de intersecao do lado $\overline{p_i p_{i+1}}$ com a reta L;
    se $x_{int} == x_s$ **então**
      p_{sel} está sobre o lado $\overline{p_i p_{i+1}}$: PARE;
    senão
      se $x_{int} > x_{sel}$ **e** $y_{int} > min(y_i, y_{i+1})$ **e** $y_{int} \leq max(y_i, y_{i+1})$
      então
        $N_{int} \leftarrow N_{int} + 1$;
  senão
    se $y_{sel} == y_i$ **e** $x_{sel} \geq min(x_i, x_{i+1})$ **e** $x_{sel} \leq max(x_i, x_{i+1})$ **então**
      p_{sel} está sobre o lado horizontal $\overline{p_i p_{i+1}}$: PARE;
se N_{int} *é ímpar* **então**
  p_{sel} é interior a P;
senão
  p_{sel} é exterior a P;

Algoritmo 1: Algoritmo de ponto em polígono

Algoritmo: PontoEmPoligono(P, p_{sel})

$N_{int} \leftarrow 0$;
para i=*1* **até** n **faça**
  se $y_i \neq y_{i+1}$ **então**
    $(x_{int}, y_{int}) \leftarrow$ pto de intersecao do lado $\overline{p_i p_{i+1}}$ com a reta L;
    se $x_{int} == x_s$ **então**
      p_{sel} está sobre o lado $\overline{p_i p_{i+1}}$: PARE;
    senão
      se $x_{int} > x_{sel}$ **e** $y_{int} > min(y_i, y_{i+1})$ **e** $y_{int} \leq max(y_i, y_{i+1})$
      então
        $N_{int} \leftarrow N_{int} + 1$;
  senão
    se $y_{sel} == y_i$ **e** $x_{sel} \geq min(x_i, x_{i+1})$ **e** $x_{sel} \leq max(x_i, x_{i+1})$ **então**
      p_{sel} está sobre o lado horizontal $\overline{p_i p_{i+1}}$: PARE;
se N_{int} *é ímpar* **então**
  p_{sel} é interior a P;
senão
  p_{sel} é exterior a P;

A seguir, listamos os principais elementos do pseudo-código abaixo:

- i: contador do laço principal, refere-se a cada um dos n lados do polígono ($i \leftarrow 1$ até n);
- N_{int}: número de intersecções a serem contabilizadas;
- $\overline{p_i p_{i+1}}$: cada lado do polígono;
- P: o polígono (lista de pontos ou vértices);
- p_{sel}: ponto de seleção (ponto que o usuário clicou;
- p_{int}: ponto de intersecção da reta L com o lado $\overline{p_i p_{i+1}}$;

Seleção: intersecção

Cálculo da Equação da Reta - Inclinação e interseção

Para encontrar os parâmetros *a* e *b* da reta *y=ax+b* basta considerar que *a* representa a sua inclinação e *b* o valor da ordenada *y* da reta para o qual a abscissa *x* é nula.

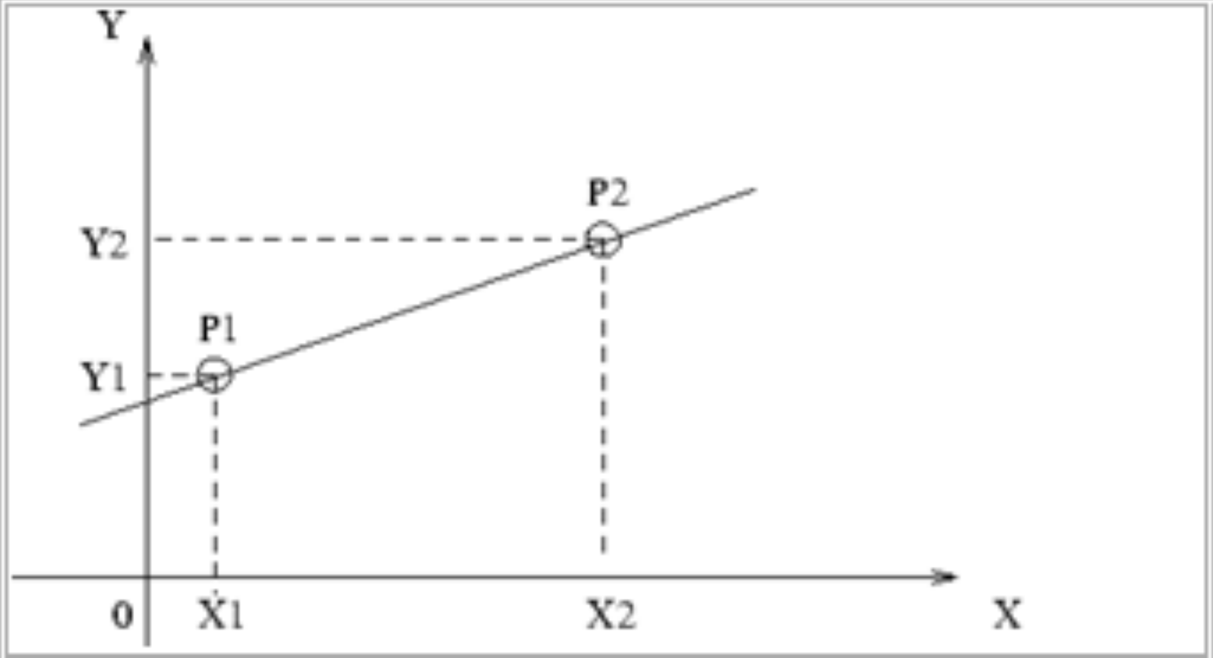


Figura: Uma reta passando por dois pontos P_1 e P_2.

Temos:

$$y_2 - y_1 = a(x_2 - x_1) \quad \Rightarrow \quad a = \frac{y_2 - y_1}{x_2 - x_1}$$

$$y_1 = \frac{y_2 - y_1}{x_2 - x_1} x_1 + b \quad \Rightarrow \quad b = y_1 - \frac{y_2 - y_1}{x_2 - x_1} x_1$$

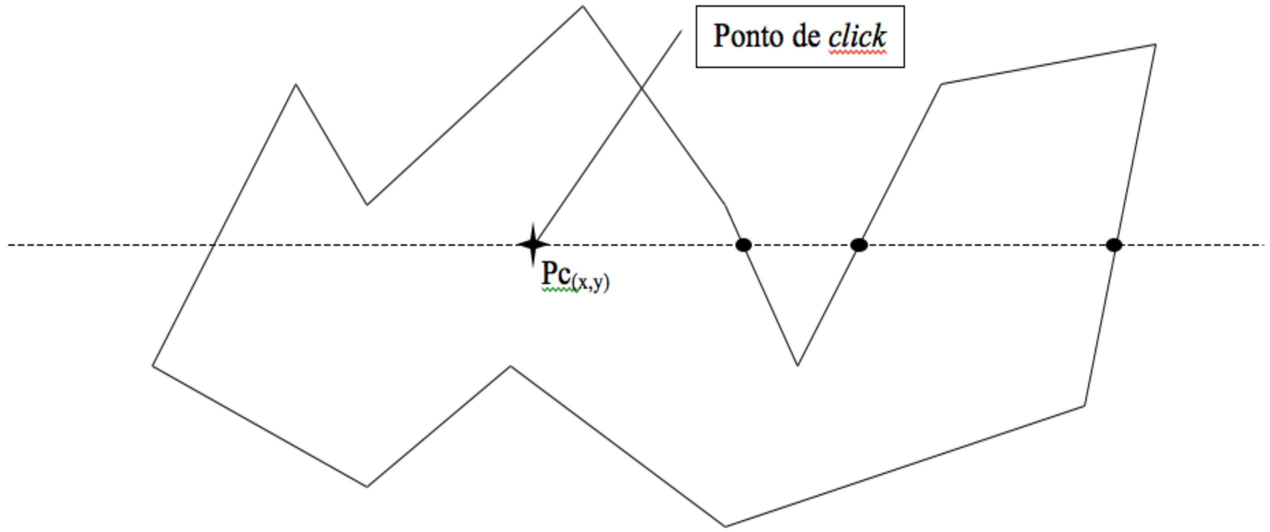
Seleção: intersecção

Equação Paramétrica da Reta

$$P = P_1 + (P_2 - P_1) \cdot t$$

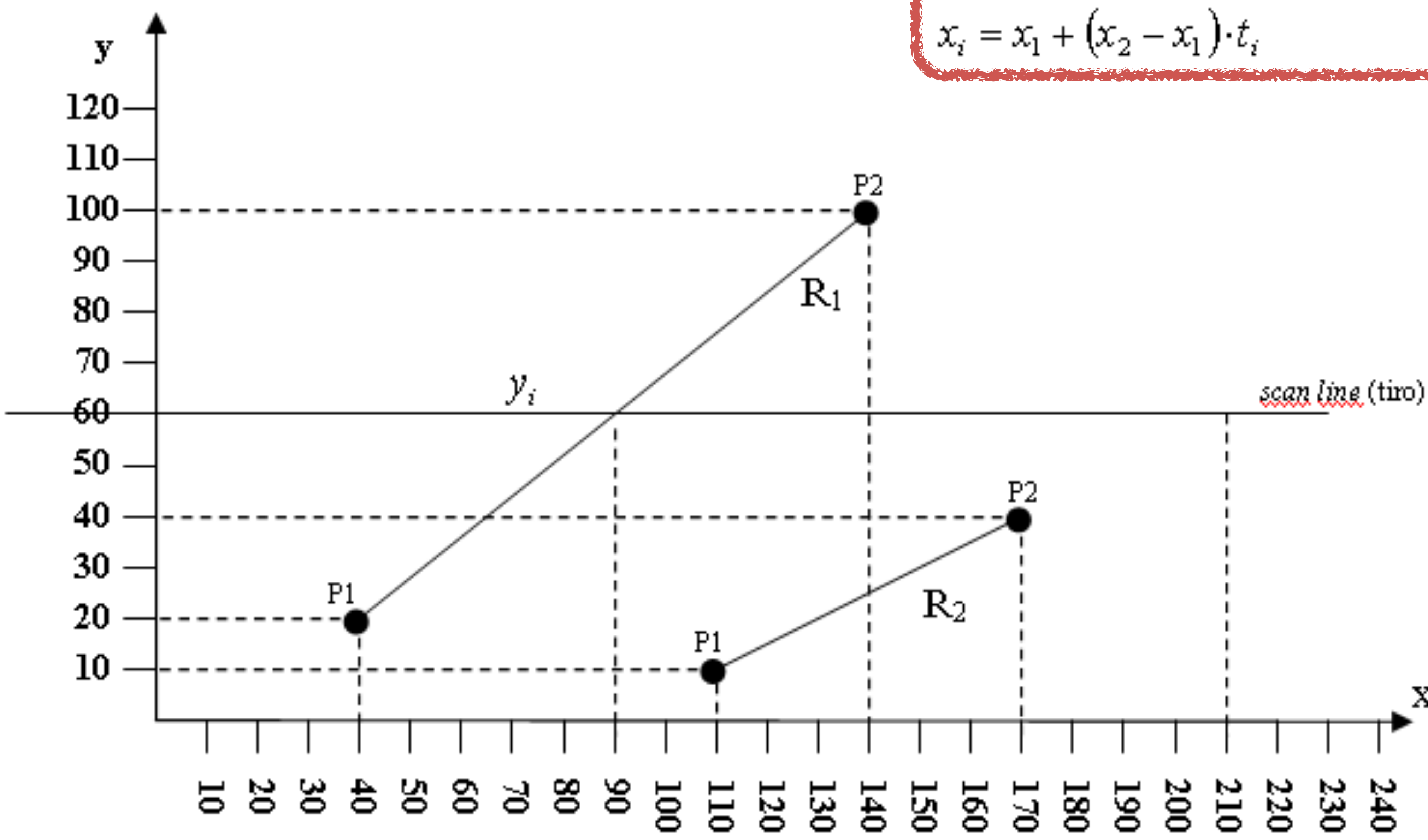
Se: $y_i = y_1 + (y_2 - y_1) \cdot t_i$ então: $\boxed{t_i = \dfrac{y_i - y_1}{y_2 - y_1}}$ $(t_i \in [0 \cdot\cdot 1]$ existe intersecção)

$\boxed{x_i = x_1 + (x_2 - x_1) \cdot t_i}$ então: $x_i = x_1 + (x_2 - x_1) \cdot \dfrac{(y_i - y_1)}{(y_2 - y_1)}$



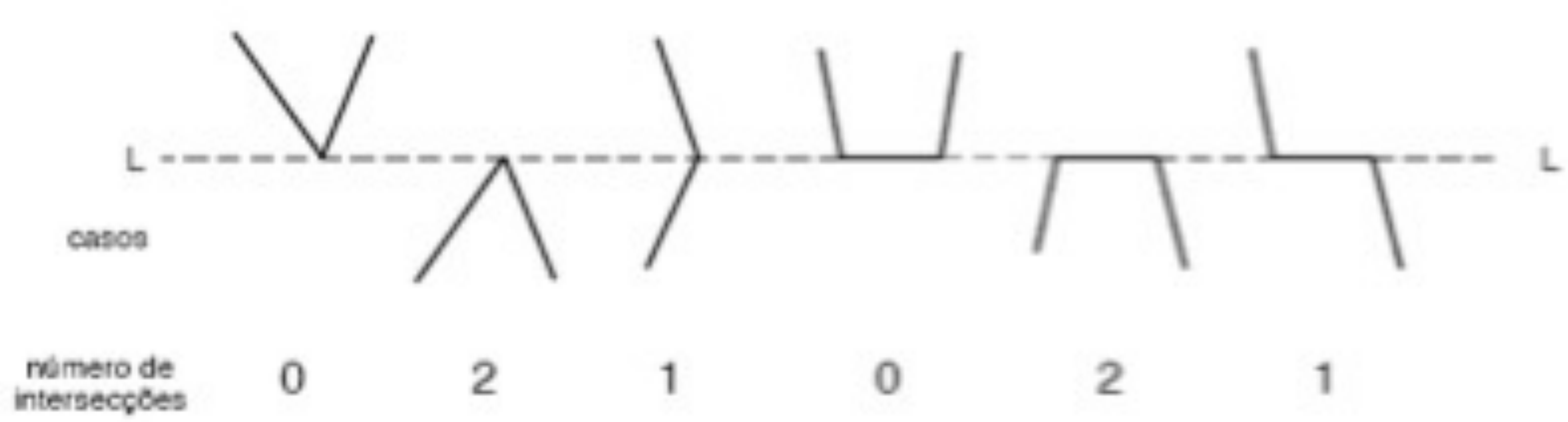
$$t_i = \frac{y_i - y_1}{y_2 - y_1} \qquad (t_i \in [0 \cdot\cdot 1] \text{ existe intersecção})$$

$$x_i = x_1 + (x_2 - x_1) \cdot t_i$$



Algoritmo de seleção

Excepcionalmente podem ocorrer casos especiais. Por exemplo, o ponto de seleção poderia estar exatamente na mesma ordenada que um dos vértices do polígono. A figura abaixo indica como esses casos são tratados. Os números indicam a quantidade de intersecções contabilizadas pelo algoritmo em cada caso.



Grafo de Cena

- **Cena**:
    - representação abstrata de todos os componentes que são apresentados em um mundo virtual;
    - todo componente que será representado no ambiente ou exercerá um comportamento que afeta outros componentes deverá ser anexado à um nó de cena.
- **Grafo de Cena**:
    - estrutura composta por arcos e nós, em forma de árvore, utilizada para especificar e documentar programas que façam uso do Java3D;
    - composto por um conjunto de símbolos que representam instâncias de objetos de classes específicas.

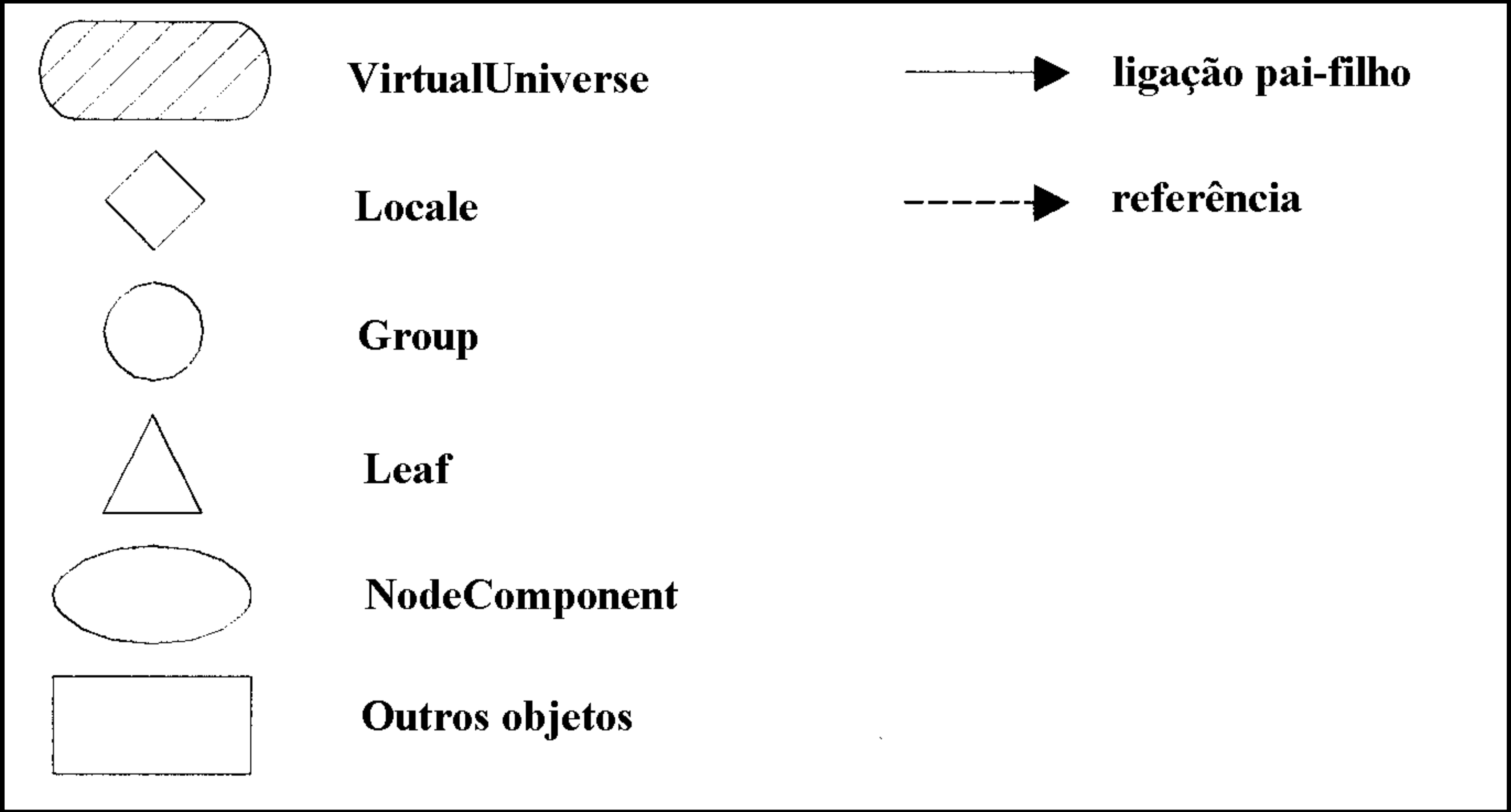
Grafo de Cena

São estruturas de dados, organizadas através de classes, onde por meio de hierarquia de objetos e atributos, pode-se mais facilmente especificar cenas complexas. Cada objeto ou atributo é representado por uma classe, que possui informações sobre sua aparência física, dentre outros fatores. Um grafo de cena trata problemas que geralmente surgem em composições ou gerencíamentos de cenas (POZZER, 2007).

Popularizado pelo SGI Open Inventor, o grafo de cena protege o desenvolvedor de se preocupar com os detalhes que compõe a renderização, fazendo com que ele foque nos objetos do qual deseja renderizar, ao invés de se preocupar com a lógica da renderização em si (WALSH, 2002).

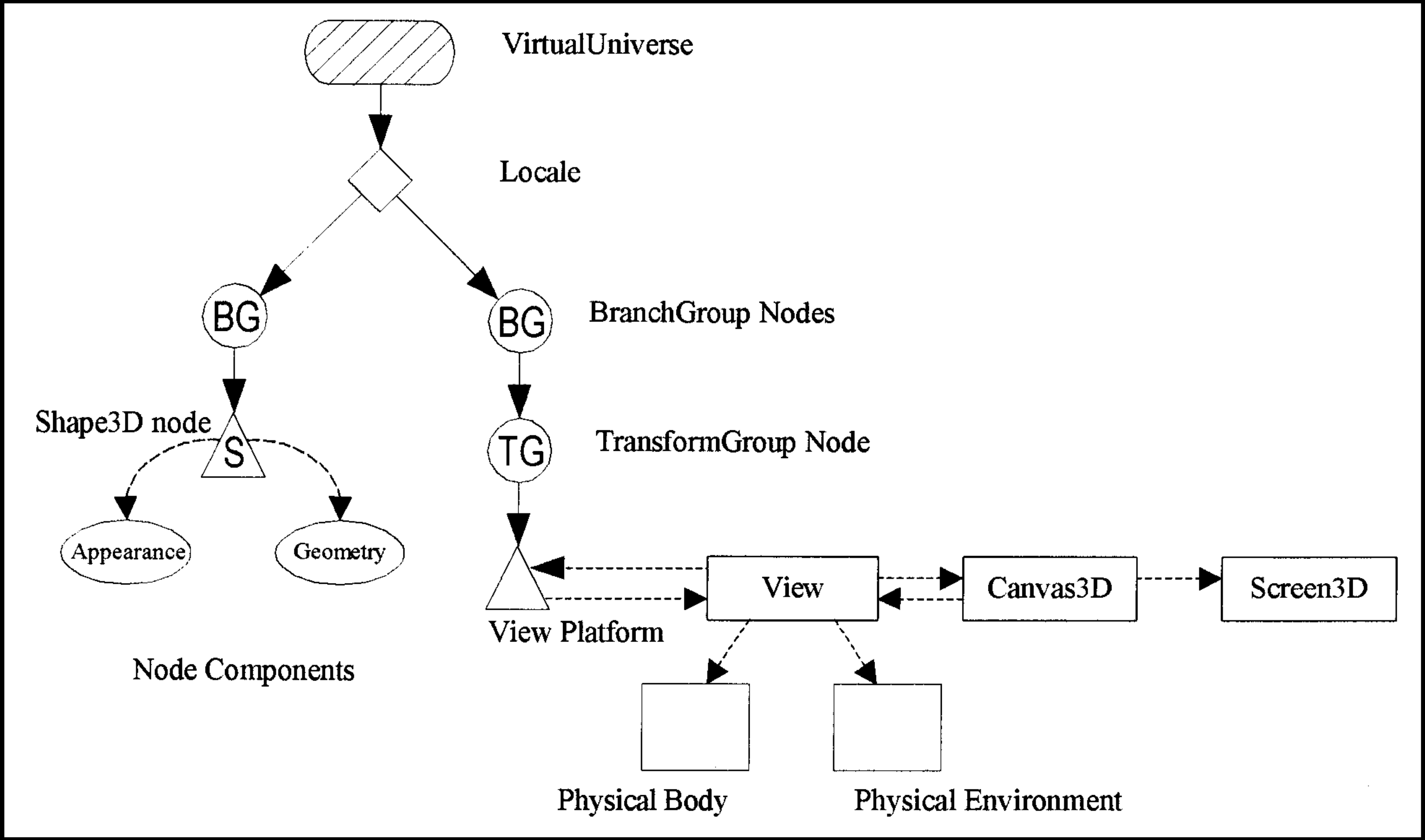
Grafo de Cena - Java3D

Simbologia

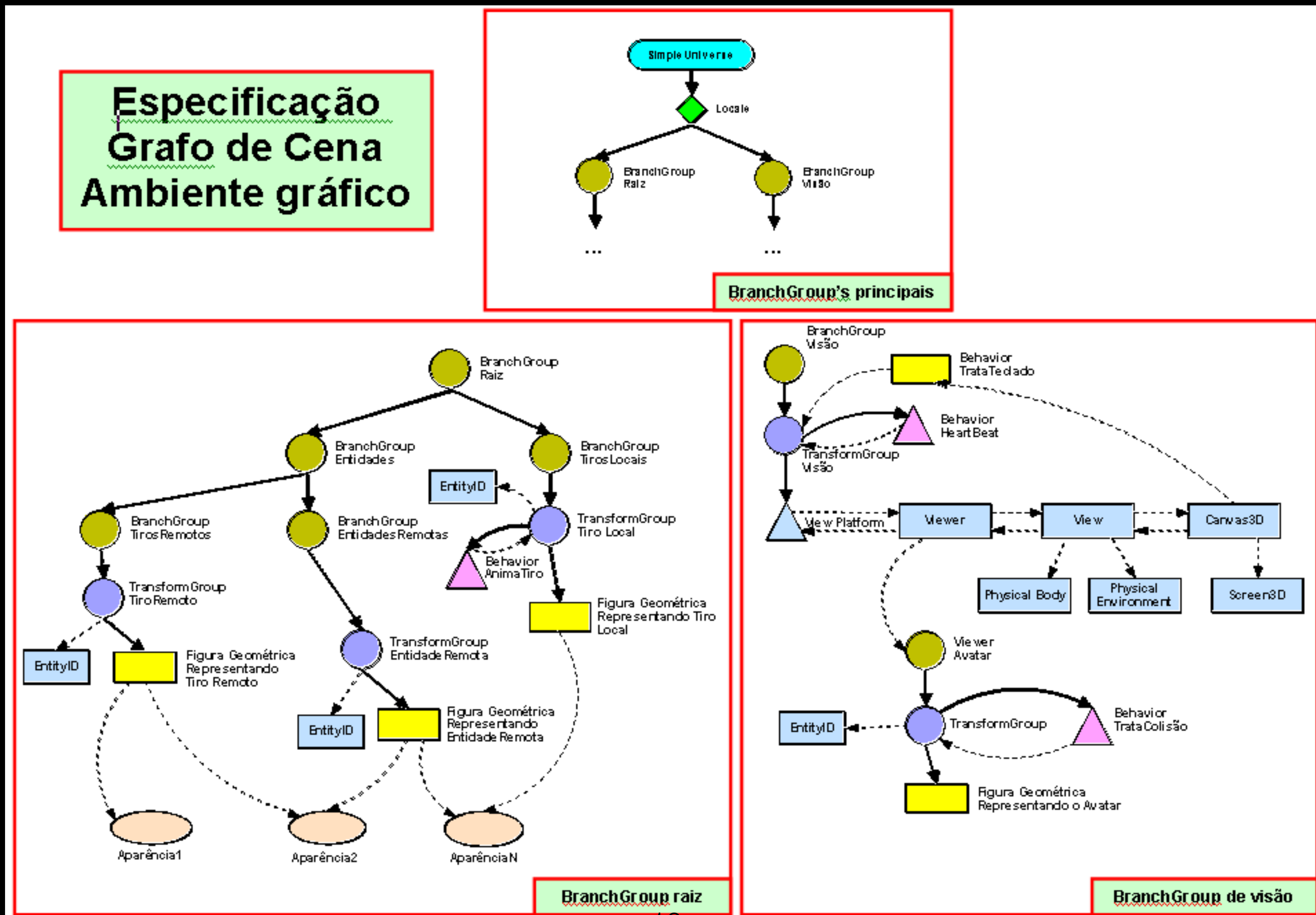


Grafo de Cena

Formalismo



Grafo de Cena



Grafo de cena e transformações geométricas

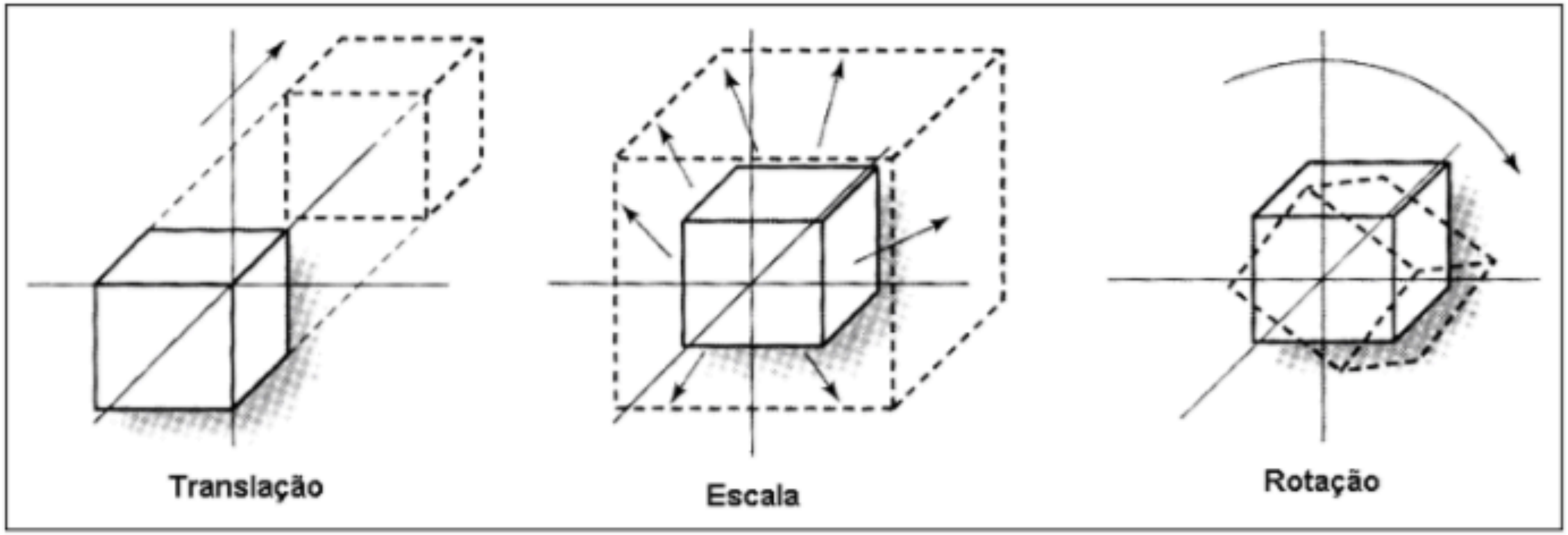
- OpenGL não implementa o grafo de cena
- Grafo de cena: estrutura a cena facilitando o processamento gráfico
- Estrutura básica:
  - Mundo (com lista de objetos gráficos)
  - Objeto gráfico
    - Forma: geometria e topologia
    - Aparência: cor, material, ...
    - Boundaring box
    - Transformações geométricas
    - Objetos "filhos": é um novo obj. gráfico (herda caract. do pai)

Grafo de cena e transformações geométricas

- OpenGL: uso dos "glPushMatrix()" e "glPopMatrix()"
- Objeto gráfico: os visíveis devem ter um método de desenho
- Objetos gráficos: devem recalcular as BBox quando for necessário

Transformações geométricas 2D e 3D

As Transformações Geométricas (translação, escala e rotação) do Ponto Médio é a prova de que qualquer segmento de reta pode ser transladado, escalado e rotacionado pela simples Transformação de seus pontos extremos.



Observação: as transformações são sempre em relação à origem.

Não usar na implementação

Transformações geométricas 2D

Translação:

$$P(x,y) \Rightarrow P'(x',y') \quad = \quad \begin{matrix} x + dx = x' \\ y + dy = y' \end{matrix} \qquad \begin{bmatrix} x & y \end{bmatrix} + \begin{bmatrix} dx & dy \end{bmatrix} = \begin{bmatrix} x' & y' \end{bmatrix}$$

A translação de um ponto *P(x,y)* no plano, ocorre pela adição ás coordenadas de *P*, dos valores de deslocamento *dx* e *dy*.

$$P + T = P'$$

Obs: *dx* e *dy* são deslocamentos, o elemento neutro é 0 (zero).

Transformações Homogêneas 2D 2D – em relação a origem

Translação

$$\begin{bmatrix} x & y & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{bmatrix} = \begin{bmatrix} x' & y' & 1 \end{bmatrix}$$

Translação Inversa

$$\begin{bmatrix} x & y & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -dx & -dy & 1 \end{bmatrix} * \begin{bmatrix} x' & y' & 1 \end{bmatrix}$$

Transformações geométricas 3D

Translação homogênea 3D – em relação a origem:

Translação

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix}$$

Translação Inversa

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \times \begin{bmatrix} 1\ 0\ 0 & -dx \\ 0\ 1\ 0 & -dy \\ 0\ 0\ 1 & -dz \\ 0\ 0\ 0\ 1 & \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

Não usar na implementação

Transformações geométricas 2D

Escala:

$$x \cdot Sx = x' \qquad y \cdot Sy = y'$$

$$\begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} = \begin{bmatrix} x' & y' \end{bmatrix} \qquad P \cdot S = P'$$

Obs: *Sx* e *Sy* são fatores de escala, caso sejam iguais à escala é proporcional em ambas as dimensões (elemento neutro é 1).

Transformações Homogêneas 2D 2D – em relação a origem

Escala

$$\begin{bmatrix} x & y & 1 \end{bmatrix} * \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x' & y' & 1 \end{bmatrix}$$

Escala Inversa

$$\begin{bmatrix} x & y & 1 \end{bmatrix} * \begin{bmatrix} \frac{1}{Sx} & 0 & 0 \\ 0 & \frac{1}{Sy} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x' & y' & 1 \end{bmatrix}$$

Transformações geométricas 3D

Escala homogênea 3D – em relação a origem:

Escala

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \times \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix}$$

Escala Inversa

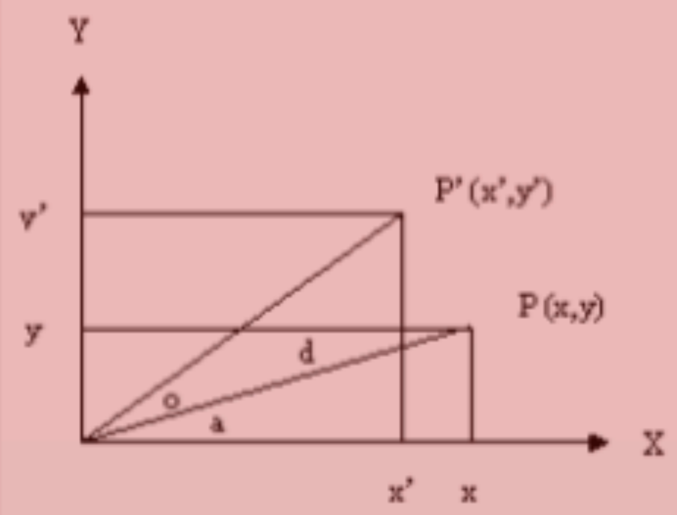
$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \times \begin{bmatrix} \frac{1}{Sx} & 0 & 0 & 0 \\ 0 & \frac{1}{Sy} & 0 & 0 \\ 0 & 0 & \frac{1}{Sz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

Não usar na implementação

Transformações geométricas 2D - Apoio

Graus	Seno	Cosseno
0°	0	1
30°	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$
45°	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$
60°	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$
90°	1	0

Rotação: passos para obtenção da matriz de rotação



$$\cos\theta = \frac{ca}{h} \qquad \cos(\alpha \pm \theta) = \cos\alpha \cdot \cos\theta \mp \operatorname{sen}\alpha \cdot \operatorname{sen}\theta$$

$$\operatorname{sen}\theta = \frac{co}{h} \qquad \operatorname{sen}(\alpha \pm \theta) = \operatorname{sen}\alpha \cdot \cos\theta \pm \cos\alpha \cdot \operatorname{sen}\theta$$

$$\begin{matrix} x = \ell \cos\alpha \\ y = \ell \operatorname{sen}\alpha \end{matrix} \Rightarrow \begin{matrix} x' = \ell(\cos\alpha + \theta) \\ y' = \ell(\operatorname{sen}\alpha + \theta) \end{matrix} \qquad Se\ x, y, \theta \rightarrow x', y'$$

$$x' = \ell(\cos\alpha \cdot \cos\theta - \operatorname{sen}\alpha \cdot \operatorname{sen}\theta) \qquad y' = \ell(\operatorname{sen}\alpha \cdot \cos\theta + \operatorname{sen}\theta \cdot \cos\alpha)$$

$$x' = \underbrace{\ell \cdot \cos\alpha}_{x} \cdot \cos\theta - \underbrace{\ell \cdot \operatorname{sen}\alpha}_{y} \cdot \operatorname{sen}\theta \qquad y' = \underbrace{\ell \cdot \operatorname{sen}\alpha}_{y} \cdot \cos\theta + \underbrace{\ell \cdot \cos\alpha}_{x} \cdot \operatorname{sen}\theta$$

$$x' = x \cdot \cos\theta - y \cdot \operatorname{sen}\theta \qquad y' = y \cdot \cos\theta + x \cdot \operatorname{sen}\theta$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\operatorname{sen}\theta \\ \operatorname{sen}\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Não usar na implementação

Transformações geométricas 2D

Rotação:

$$[x \quad y] \cdot \begin{bmatrix} \cos\theta & -sen\theta \\ sen\theta & \cos\theta \end{bmatrix} = [x' \quad y']$$

Obs: o teta (minúsculo) é o ângulo de rotação, sua faixa é de 0 a 360 e seu elemento neutro é 0 (ou 360).

Transformações Homogêneas 2D 2D – em relação a origem

Rotação

$$[x \quad y \quad 1] * \begin{bmatrix} \cos\alpha & -sen\alpha & 0 \\ sen\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} = [x' \quad y' \quad 1]$$

Rotação Inversa

$$[x \quad y \quad 1] * \begin{bmatrix} \cos\alpha & sen\alpha & 0 \\ -sen\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} = [x' \quad y' \quad 1]$$

Transformações geométricas 3D

Rotação X homogênea 3D – em relação a origem:

Rotação: sentido horário

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix}$$

Rotação: sentido anti-horário

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \times \begin{bmatrix} 1\ 0\ 0\ 0 \\ 0\ cos\alpha\ sin\alpha\ 0 \\ 0\ -sin\alpha\ cos\alpha\ 0 \\ 0\ 0\ 0\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

Transformações geométricas 3D

Rotação Y homogênea 3D – em relação a origem:

Rotação: sentido horário

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \times \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix}$$

Rotação: sentido anti-horário

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \times \begin{bmatrix} 0\ cos\alpha\ sin\alpha\ 0 \\ 0\ 1\ 0\ 0 \\ 0\ \ -sin\alpha\ cos\alpha\ 0 \\ 0\ 0\ 0\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

Transformações geométricas 3D

Rotação Z homogênea 3D – em relação a origem:

Rotação: sentido horário

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \times \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix}$$

Rotação: sentido anti-horário

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \times \begin{bmatrix} 0\ cos\alpha\ sin\alpha\ 0 \\ 0\ \ -sin\alpha\ cos\alpha\ 0 \\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

Transformações geométricas 3D

Composição de transformações geométricas:

Ordem das transformações: rotação seguida de uma translação

Ordem das transformações: translação seguida de uma rotação

Observe que o resultado final é diferente, ou seja, a ordem das transformações interfere no resultado após uma seqüência de transformações. Caso "clássico" é a transformação de escala ou rotação a um ponto fixo, onde se usa a seqüência:

- translação para origem em relação ao ponto que se quer fixar;
- a transformação, no caso, escala ou rotação;
- translação inversa da origem para o ponto fixado.

Transformações geométricas 3D

<u>Composição de transformações geométricas:</u>

A multiplicação de diferentes matrizes de transformação, entre si, geram a concatenação de todas as modificações em uma única estrutura, que é chamada de matriz de modelação-visualização. Ela é responsável por determinar dentro de um contexto, as posições e modificações dos objetos 3D de uma cena.

Transformações geométricas 3D

Composição de transformações geométricas:

```
#pragma mark Draw and Display
void mainDisplay(void) {                    // maindisplay
    lastTick = VART::Time::NOW(); // calculate FPS

        glMatrixMode (GL_PROJECTION);
        glLoadIdentity ();
        gluOrtho2D(ortho2D_minX, ortho2D_maxX, ortho2D_minY, ortho2D_maxY);
        glMatrixMode (GL_MODELVIEW);
        glLoadIdentity ();
        glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    SRU();
    drawObject();

    calculateFPS();

    char text[40]; sprintf(text,"FPS [%3.2f] - Mouse[%i,%i]",fps,mouseXPos,
mouseYPos);
    drawText(10, 10, text);

        glutSwapBuffers();
}
```

Transformações geométricas 3D

“empilhar/desempilhar” transformações:

```
gl.glPushMatrix();
    ...
gl.glPopMatrix();
```

exemplos: **Não usar na implementação**

gl.glTranslatef(dx, dy, dz);	gl.glTranslatef(-0.5f, 1.5f, 0.0f);
gl.glRotatef(angle, Rx, Ry, Rz);	gl.glRotatef(angle, 0, 0, 1);
gl.glScalef(Sx, Sy, Sz);	gl.glScalef(0.5f, 0.5f, 1.0f);

Obs.: no caso do 2D deve-se:
- translação: assumir 0 (zero) para coordenada Z;
- escala: assumir 1 (um) para coordenada Z;
- rotação: assumir como eixo de rotação o Z (ângulo, 0,0,1);
- para Transformações Globais usar as transformações na ordem inversa.

No OpenGL deve ser Translação inversa, Rotação/Escala e Translação
ver exemplos: 06_Rotate-BBox-glMultMatrixd

Atenção

Computação Gráfica
Unidade 03

prof. Dalton S. dos Reis
dalton.reis@gmail.com

FURB - Universidade Regional de Blumenau
DSC - Departamento de Sistemas e Computação
Grupo de Pesquisa em Computação Gráfica, Processamento de Imagens e Entretenimento Digital
http://www.inf.furb.br/gcg/