O que se pede, em geral, ao Programador de aplicações gráficas?

- **Conhecimentos matemáticos sobre geometria 2D / 3D**
- **Construção de linguagens de interação e definição do diálogo**
- **Manipulação de modelos, incluindo os hierárquicos**
- **Conhecimentos de diversos algoritmos, de acordo com a aplicação**
- **Programas robustos, com facilidades de extensão, reutilização e manutenção (engenharia de software)**
- …

Modelo conceptual de base para o Programador

Utilizador
+
Hardware

Out

In

Software

Sistema Gráfico
+
Gestor de Janelas

Programa
de
Aplicação

Dispositivos físicos de Entrada de Dados

IN

- Teclado
- Rato
- *Joystick*
- *Trackball*
- Mesa digitalizadora *(Tablet)*
- Telecomando

- (...)

Dispositivos físicos para Visualização

OUT

- Monitor
- Impressora
- Óculos para Realidade Virtual (HMD)
- Traçador *(Plotter)*

- (…)

Paradigmas de usabilidade

O desenvolvimento de produtos e/ou a sua utilização servem-se, frequentemente, de sistemas conhecidos (com sucesso) como paradigmas (ie, como exemplos).

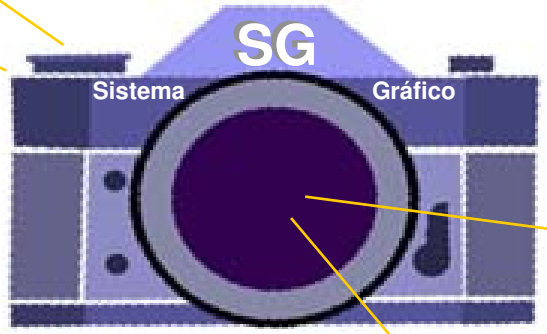
A metáfora (ie, transporte da significação de um conceito para outro por relação de semelhança) é um caso paradigmático que, se for adequada, pode acelerar substancialmente e de maneira simples a familiarização dos utilizadores com os sistemas em causa (aumentando a usabilidade através da interface).

Exemplos:

- O tampo de secretária (*desktop*) no ecrã do computador.
- A tartaruga na linguagem de programação Logo, produzindo os chamados *turtle graphics*.
- A máquina fotográfica, tanto na utilização como na programação de aplicações gráficas 3D.

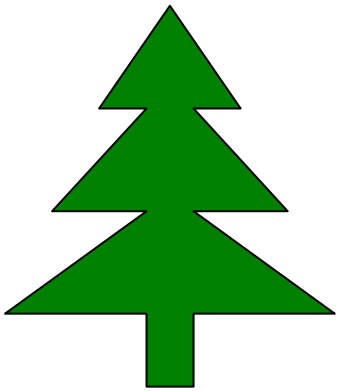
A metáfora da máquina fotográfica

Visualização (*Viewing*)



Modelação
(*Modeling*)

Programa de Aplicação



Modelo

M.Próspero

POSSÍVEIS CRITÉRIOS PARA CLASSIFICAÇÃO DAS APLICAÇÕES

- **Tipo de objeto e representação pictórica**

  2D + Linha, Cor acromática, Cor cromática

  3D + Linha c/ ou s/ efeitos, Cor e sombras c/ ou s/ efeitos

- **Tipo de interação**

  Grau de controlo do utilizador sobre as imagens

- **Objetivo da imagem**

  Fim em si ou meio para outra finalidade

- **Relação objeto – imagem: temporal e/ou lógica**

  Uma imagem relacionada com outras,
  ou no tempo (e.g. filme) ou por assemblagem (objeto composto)

O PROCESSO DE GERAÇÃO DA IMAGEM

- **Construção do Modelo da Aplicação e correspondente Representação**

  Coleção de dados, representando objetos físicos ou abstratos, e de relações, bem como de operações que ajudam a definir a sua estrutura e/ou comportamento. [Definição de âmbito mais alargado do que o de Computação Gráfica]

- **Descrição do Modelo ao Sistema Gráfico**

  - O quê ?

  - Como ?

    Em termos de primitivas gráficas universais (pontos, linhas, polígonos, …) em vez de estruturas de dados normalizadas dependentes da aplicação (mais restritivas e ineficientes em muitos casos).

  - Modo de interação (Pessoa-Computador) para exploração visual ou edição ?

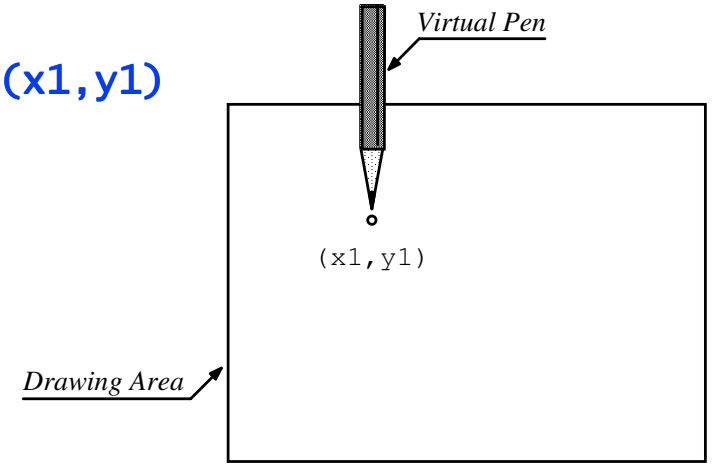
- **Criação da Imagem**

  Representação gráfica (realizada pelo SG) para ser vista pelo utilizador.

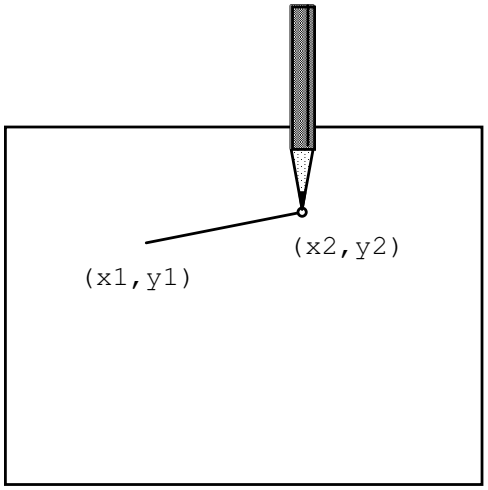
Programação com Caneta Virtual (metáfora)

Desenho de segmentos de reta só c/ Coordenadas Absolutas

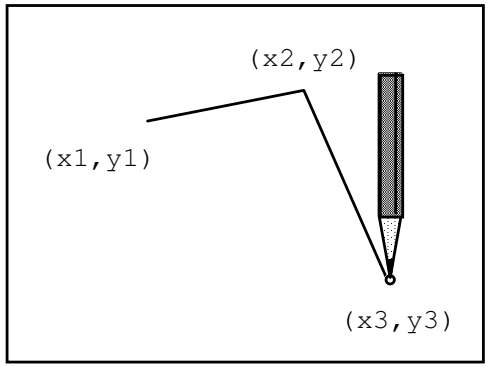
MOVE (x1,y1)



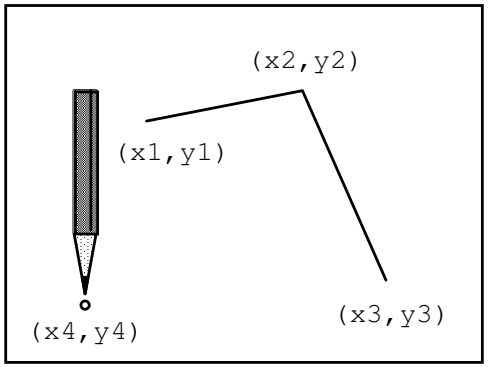
LINE (x2,y2)



LINE (x3,y3)



MOVE (x4,y4)



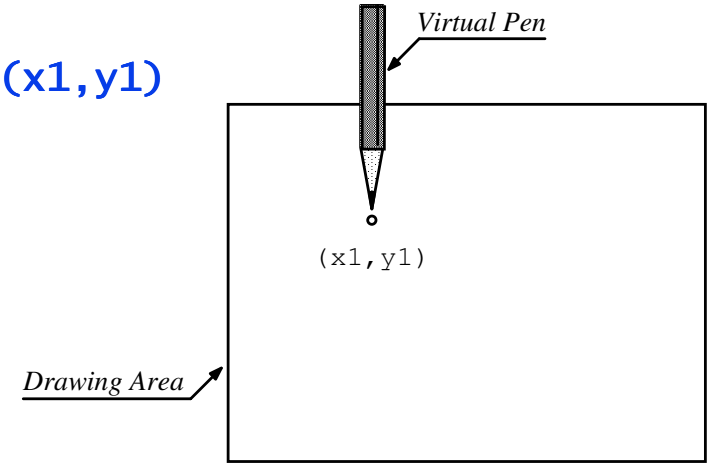
As instruções apresentadas não são de qualquer API do mercado, mas meros exemplos paradigmáticos.

Conceito de Posição Corrente: Posição, nesse instante, da caneta virtual

Programação com Caneta Virtual (metáfora)

Desenho de segmentos de reta c/ Coordenadas Relativas

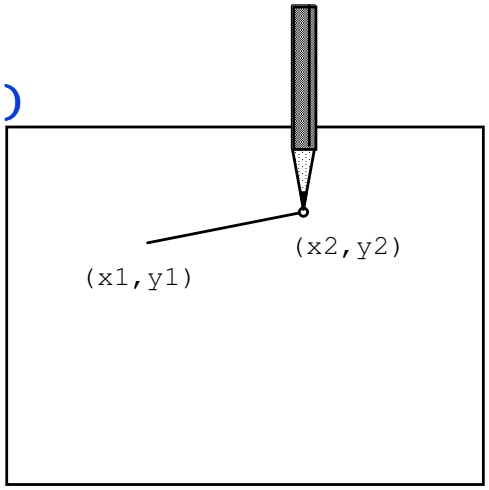
MOVE (x1,y1)



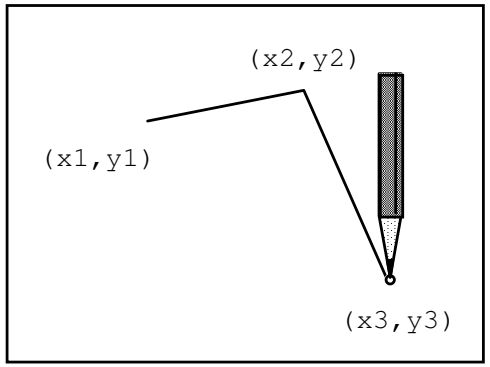
LINE_R (Δx2,Δy2)

Δx2=x2-x1

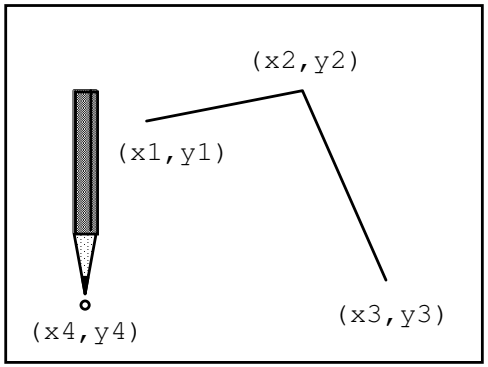
Δy2=y2-y1



LINE_R (Δx3,Δy3)

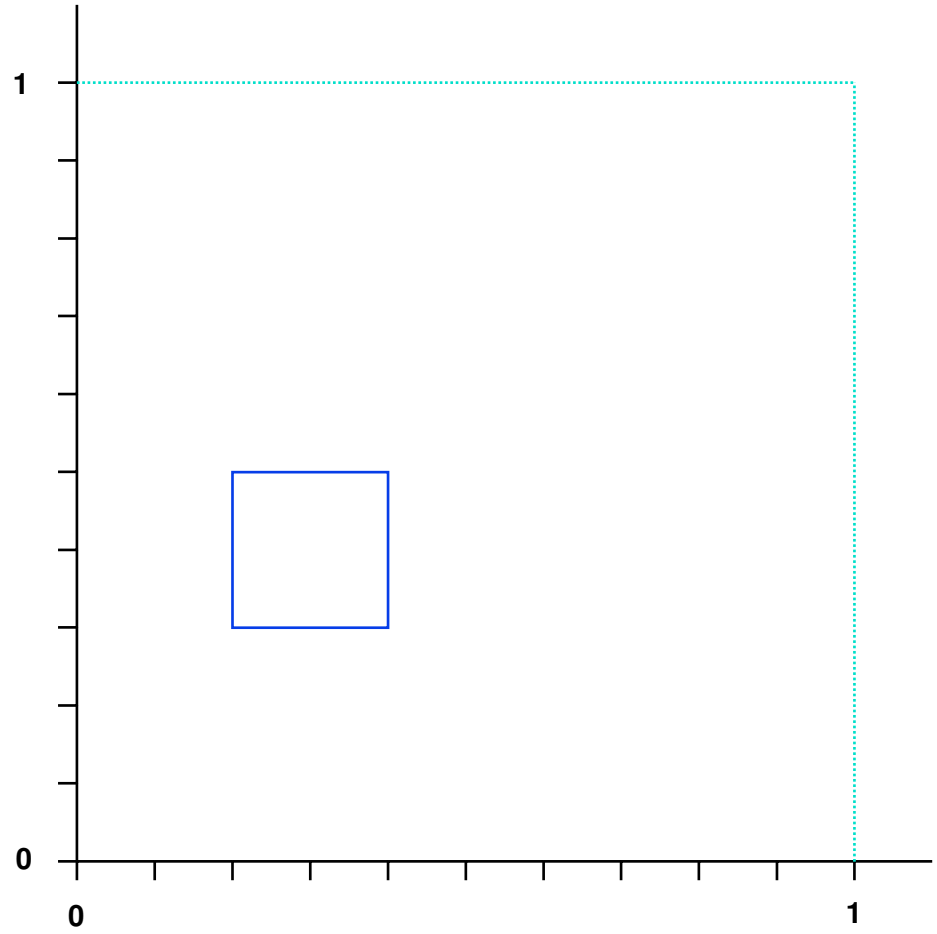


MOVE_R (Δx4,Δy4)



Conceito de Posição Corrente

Exemplo de aplicação no desenho de um quadrado



Dispositivo de Visualização Normalizado:

Pontos especificados em
NDC (Normalized Device Coordinates):
as coordenadas variam entre 0 e 1

C/ Posição Corrente:

```
MOVE (0.2, 0.3)
LINE (0.4, 0.3)
LINE (0.4, 0.5)
LINE (0.2, 0.5)
LINE (0.2, 0.3)
```

S/ Posição Corrente:

```
LINE (0.2, 0.3, 0.4, 0.3)
LINE (0.4, 0.3, 0.4, 0.5)
LINE (0.4, 0.5, 0.2, 0.5)
LINE (0.2, 0.5, 0.2, 0.3)
```

Características do uso da Posição Corrente:

+ Reduz o número de argumentos requeridos para cada primitiva de saída gráfica.

- Aumenta o número de primitivas de saída gráfica.

- Não tem independência da ordem de execução.

- Implica valores arbitrários em certas transições de estado do sistema gráfico (indefinição da posição corrente).

Normas gráficas ISO/ANSI (a nível API) que **não usam** o conceito de Posição Corrente:

GKS
Graphical Kernel System

PHIGS
Programmer's Hierarchical Interactive Graphics System

5.3 Output functions

POLYLINE **WSAC,SGOP** **L0a**

Parameters:

In	number of points	(2..n)	I
In	points	WC	n × P

Effect: A sequence of connected straight lines is generated, starting from the first point and ending at the last point. The current values of the polyline attributes, as given by the GKS state list (see 6.4), are bound to the primitive. The polyline attributes are listed in 4.4.2.

If, after the workstation transformation, all points coincide, no error is generated and whether anything is drawn is workstation dependent.

References:

4.4.1
4.4.2
4.4.3
4.5.3

Errors:

5 GKS not in proper state: GKS shall be either in the state WSAC or in the state SGOP
100 Number of points is invalid

(...)

5.3 Output primitive functions

POLYLINE 3 **(PHOP,*,STOP,*)**

Parameters:

In	point list	MC	L(P3)

Effect: This function fully specifies the three dimensional form of the polyline primitive. Depending upon the 'edit mode', a "polyline 3" element is inserted into the open structure after the 'element pointer' or replaces the element pointed at by the 'element pointer'. The 'element pointer' is then updated to point to this "polyline 3" structure element. The value specified by the parameter is associated with the element.

When an element of this type is interpreted a connected sequence of straight lines is generated starting from the element's first point and ending at the element's last point. The current values of the polyline attributes, as defined in the PHIGS traversal state list (see 6.4), are bound to the primitive. The polyline attributes are described in 4.5.3.

A "polyline 3" element with less than two points will be placed in the open structure. When a "polyline 3" element with less than two points is interpreted it will have no visual effect.

References: 4.5.1 4.5.2 4.5.3 4.5.12 4.5.13 4.5.14 4.5.15 4.5.16 4.5.17 4.6.3

Errors:

005 *Ignoring function, function requires state (PHOP,*,STOP,*)*

(...)

Programa exemplo:

```
/* Access the PHIGS C binding. */
/* The next line is implementation dependent. */
#include <phigs.h>

#define WS 1
#define NUM_PTS 5
#define BOX 1

Ppoint3 box_pts[] =                          /* Define the data
                                                for the box. */
   {{0.2,0.3,0.0}, {0.4,0.3,0.0},
    {0.4,0.5,0.0}, {0.2,0.5,0.0},
    {0.2,0.3,0.0}};

main()
   {
   popenphigs(0, 0);                         /* Open PHIGS. */
   popenws(WS, 0, 0);                        /* Open a workstation. */
   popenstruct(BOX);                         /* Define the box structure. */
   ppolyline3(NUM_PTS, box_pts);             /* Insert a polyline 3 element
                                                to draw the box. */
   pclosestruct();                           /* Close the structure. */
   ppoststruct(WS, BOX, 1.0);                /* Post the structure. */
   sleep(60);                                /* Sleep for 60 seconds. */
   pclosews(WS);                             /* Close the workstation. */
   pclosephigs();                            /* Close PHIGS. */
}
```

Existem *bindings* (ie, regras sintáticas normalizadas) para outras linguagens

Exemplo de utilização de primitivas gráficas (nível API)

M.Próspero

```
<HTML>
<BODY BGCOLOR="#FFCC99">

<applet code="Casa.class" width=400 height=280></applet>

</BODY></HTML>
```

```
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Polygon;
/**
 * @author M. Próspero
 */
public class Casa extends java.applet.Applet {

   public void init() {
      setBackground(Color.WHITE);
   }

   public void paint(Graphics screen) {

      screen.setColor(Color.BLACK);  // paredes:
      screen.drawLine(120, 220, 120, 140);
      screen.drawLine(280, 220, 280, 140);

      screen.setColor(Color.BLUE);  // porta:
      int[] xPorta={140, 140, 180, 180};
      int[] yPorta={220, 160, 160, 220};
      int pontos=4;
      screen.drawPolyline(xPorta,yPorta,pontos);

      screen.setColor(Color.MAGENTA);  // janela:
      screen.drawRect(200, 160, 60, 40);

      screen.setColor(Color.RED);  // telhado:
      int[] xTelhado={100, 200, 300};
      int[] yTelhado={140, 60, 140};
      Polygon telhado=new Polygon(xTelhado, yTelhado, 3);
      screen.fillPolygon(telhado);
   }
}
```