

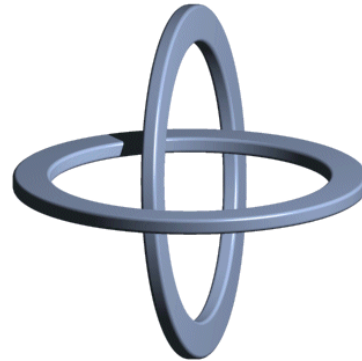
Ferramentas de Desenvolvimento do Tecgraf:

IUP, CD e IM

Antonio Scuri

scuri@tecgraf.puc-rio.br





Tecgraf
PUC-RIO

**Sistemas de
Ancoragem
e Flutuantes**

**Simulação,
Controle e
Automação**

**Sistemas de
Informação
Geográfica**

**Modelagem e
Visualização**

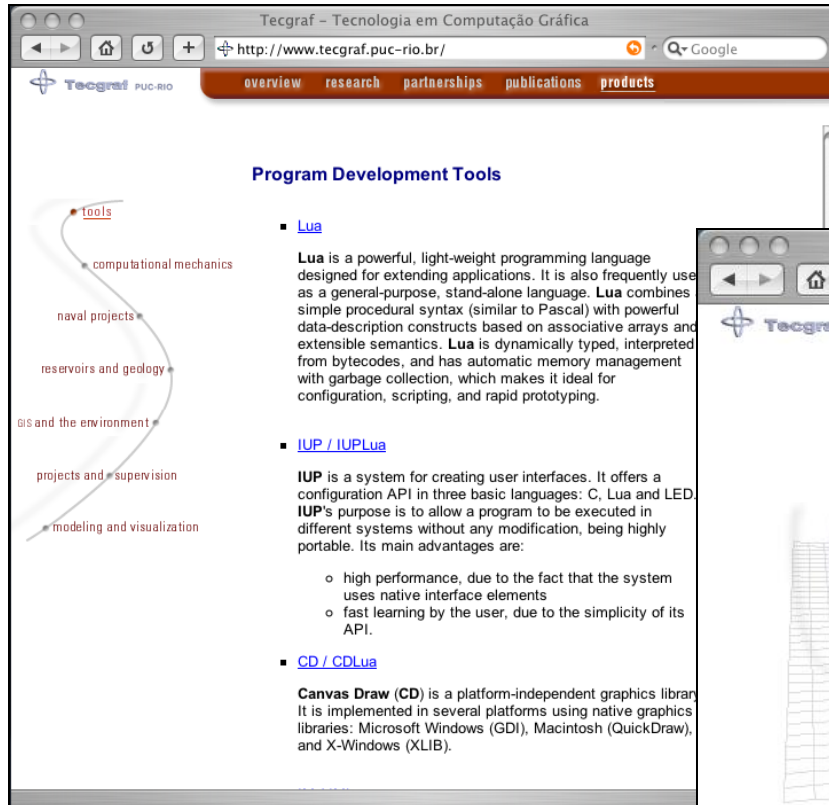
**Realidade
Aumentada**

**Jogos de
Treinamento**

Tecnologias de Desenvolvimento de Software

Para mais Informações

<http://www.tecgraf.puc-rio.br/>



Tecgraf - Tecnologia em Computação Gráfica

overview research partnerships publications **products**

Program Development Tools

- tools
- computational mechanics
- naval projects
- reservoirs and geology
- GIS and the environment
- projects and supervision
- modeling and visualization

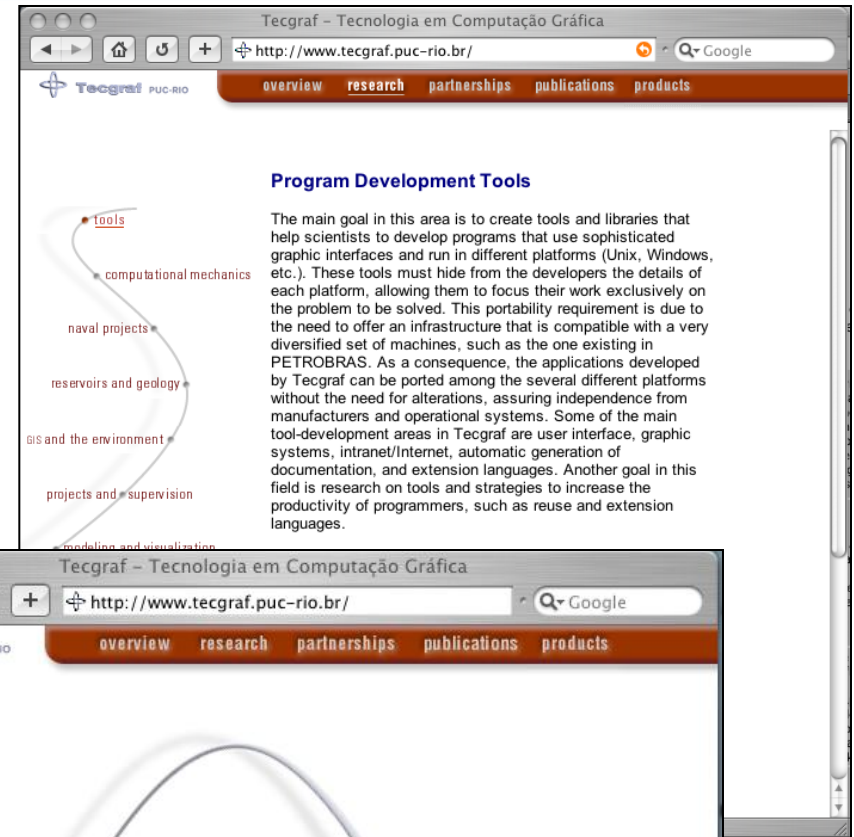
- [Lua](#)

Lua is a powerful, light-weight programming language designed for extending applications. It is also frequently used as a general-purpose, stand-alone language. **Lua** combines simple procedural syntax (similar to Pascal) with powerful data-description constructs based on associative arrays and extensible semantics. **Lua** is dynamically typed, interpreted from bytecodes, and has automatic memory management with garbage collection, which makes it ideal for configuration, scripting, and rapid prototyping.
- [IUP / IUPLua](#)

IUP is a system for creating user interfaces. It offers a configuration API in three basic languages: C, Lua and LED. **IUP's** purpose is to allow a program to be executed in different systems without any modification, being highly portable. Its main advantages are:

 - o high performance, due to the fact that the system uses native interface elements
 - o fast learning by the user, due to the simplicity of its API.
- [CD / CDLua](#)

Canvas Draw (CD) is a platform-independent graphics library. It is implemented in several platforms using native graphics libraries: Microsoft Windows (GDI), Macintosh (QuickDraw), and X-Windows (XLIB).



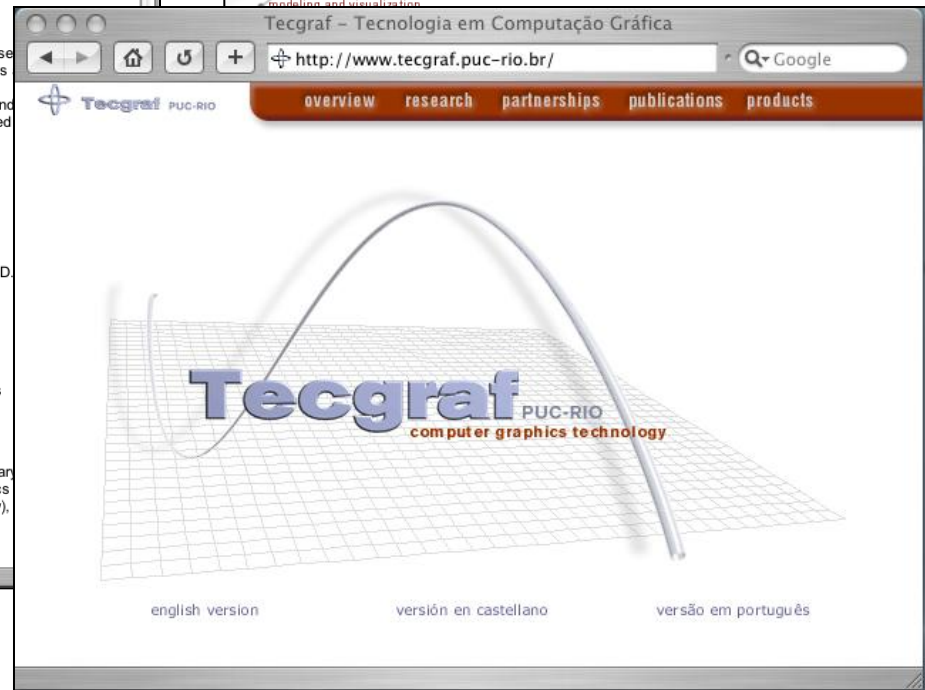
Tecgraf - Tecnologia em Computação Gráfica

overview research partnerships publications **products**

Program Development Tools

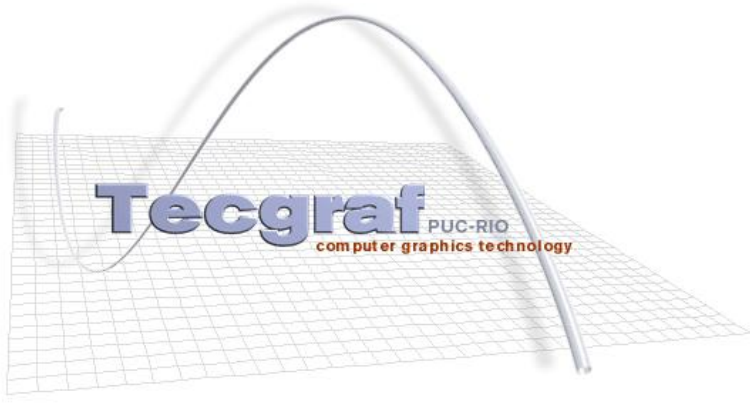
The main goal in this area is to create tools and libraries that help scientists to develop programs that use sophisticated graphic interfaces and run in different platforms (Unix, Windows, etc.). These tools must hide from the developers the details of each platform, allowing them to focus their work exclusively on the problem to be solved. This portability requirement is due to the need to offer an infrastructure that is compatible with a very diversified set of machines, such as the one existing in PETROBRAS. As a consequence, the applications developed by Tecgraf can be ported among the several different platforms without the need for alterations, assuring independence from manufacturers and operational systems. Some of the main tool-development areas in Tecgraf are user interface, graphic systems, intranet/Internet, automatic generation of documentation, and extension languages. Another goal in this field is research on tools and strategies to increase the productivity of programmers, such as reuse and extension languages.

- tools
- computational mechanics
- naval projects
- reservoirs and geology
- GIS and the environment
- projects and supervision
- modeling and visualization



Tecgraf - Tecnologia em Computação Gráfica

overview research partnerships publications **products**



Tecgraf PUC-RIO
computer graphics technology

english version versión en castellano versão em português

<http://www.tecgraf.puc-rio.br/cd/>

<http://www.tecgraf.puc-rio.br/iup/>

IUP - Version 3.0

IUP

Portable User Interface

Version 3.0

IUP is a portable toolkit for building graphical user interfaces. It offers a configuration API in three basic languages: C, Lua and LED. **IUP's** purpose is to allow a program to be executed in different systems without any modification, therefore it is highly portable. Its main advantages are:

- high performance, due to the fact that it uses native interface elements.
- fast learning by the user, due to the simplicity of its API.

This work was developed at Tecgraf/PUC-Rio by means of the partnership with

CD - Version 5.2

CD

Canvas Draw, A 2D Graphics Library

Version 5.2

CD (Canvas Draw) is a platform-independent graphics library. It is implemented in several platforms using native graphics libraries: Microsoft Windows (GDI) and X-Windows (XLIB).

The library contains functions to support both vector and image applications, and the visualization surface can be either a window or a more abstract surface, such as Image, Clipboard, Metafile, PS, and so on.

This work was developed at Tecgraf/PUC-Rio by means of the partnership with PETROBRAS/CENPES.

IM - Version 3.5

IM

Image Representation, Storage, Capture and Processing

Version 3.5

IM is a toolkit for Digital Imaging. IM is based on 4 concepts: Image Representation, Storage, Processing and Capture. The main goal of the library is to provide a simple API and abstraction of images for scientific applications.

The most popular file formats are supported: TIFF, BMP, PNG, JPEG, GIF and AVI. Image representation includes scientific data types. About a hundred Image Processing operations are available.

This work was developed at Tecgraf/PUC-Rio by means of the partnership with PETROBRAS/CENPES.

Project Management:

Antonio Escaño Scuri

Tecgraf - Computer Graphics Technology Group, PUC-Rio, Brazil
<http://www.tecgraf.puc-rio.br/im>

<http://www.tecgraf.puc-rio.br/im/>

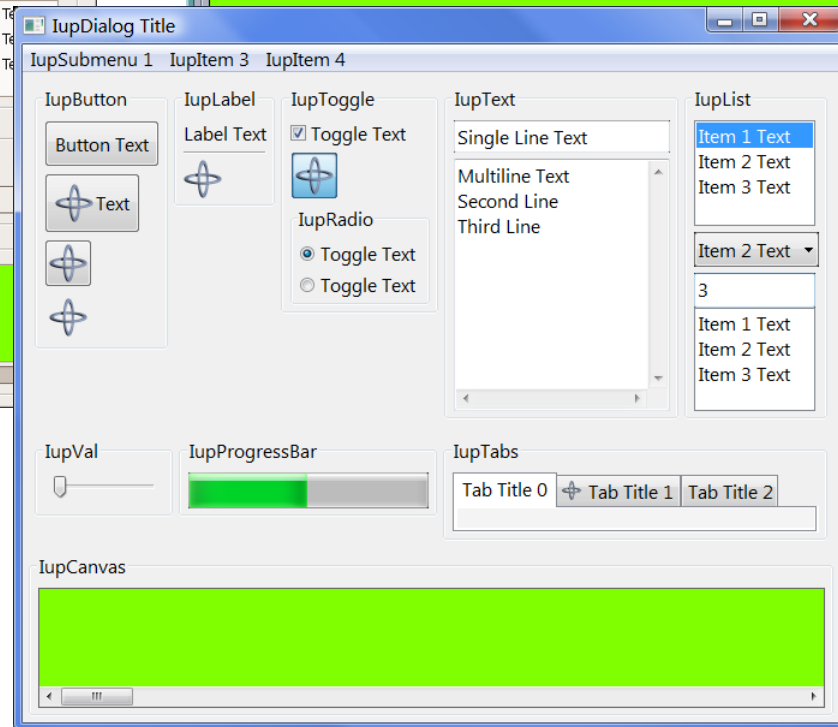
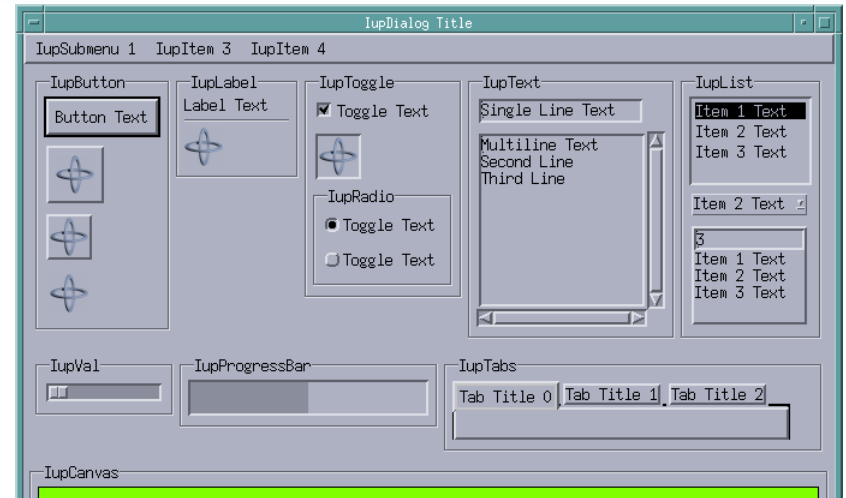
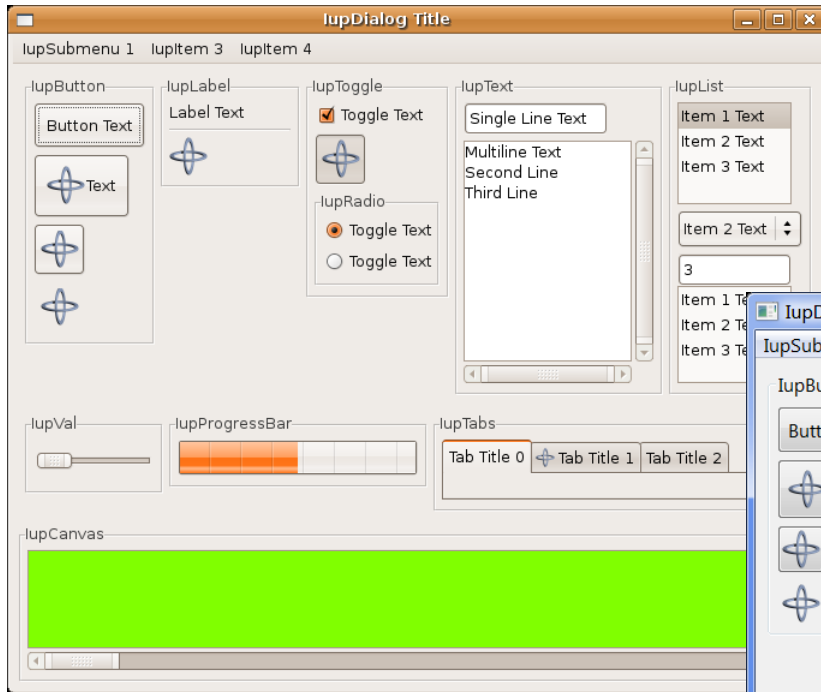
Webbook
Tecmake
LuaBinaries



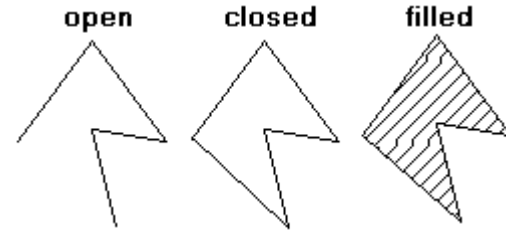
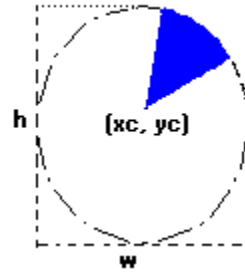
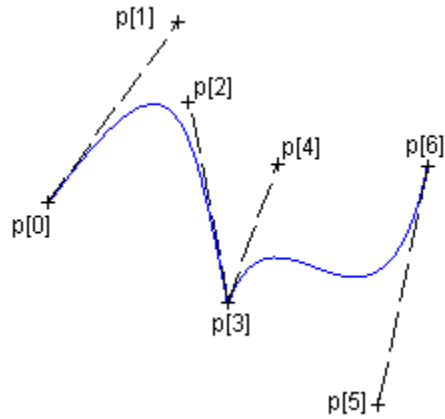
- Modelo de Abstração
- Portabilidade (Desktop)
- Desempenho (recursos nativos do sistema)
- Simplicidade com Recursos
- Livre e Código Aberto (mesma licença de Lua)
- Binários de várias plataformas prontos para Download
- API em C e em Lua
- Extensa Documentação com Exemplos
- Política de Compatibilidade entre Versões
- Melhor Curva de Aprendizado (*)

Motif em MWM

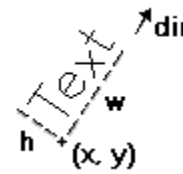
GTK em Gnome

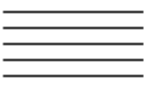







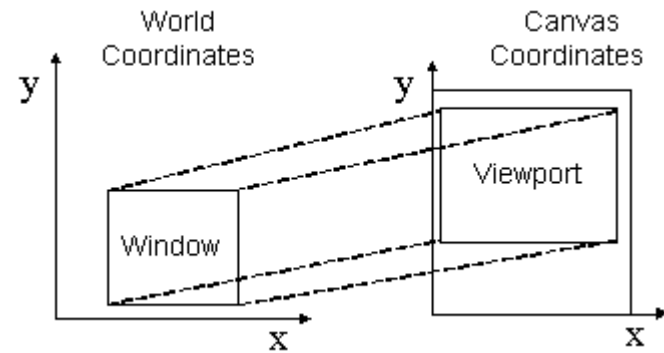
Windows Vista

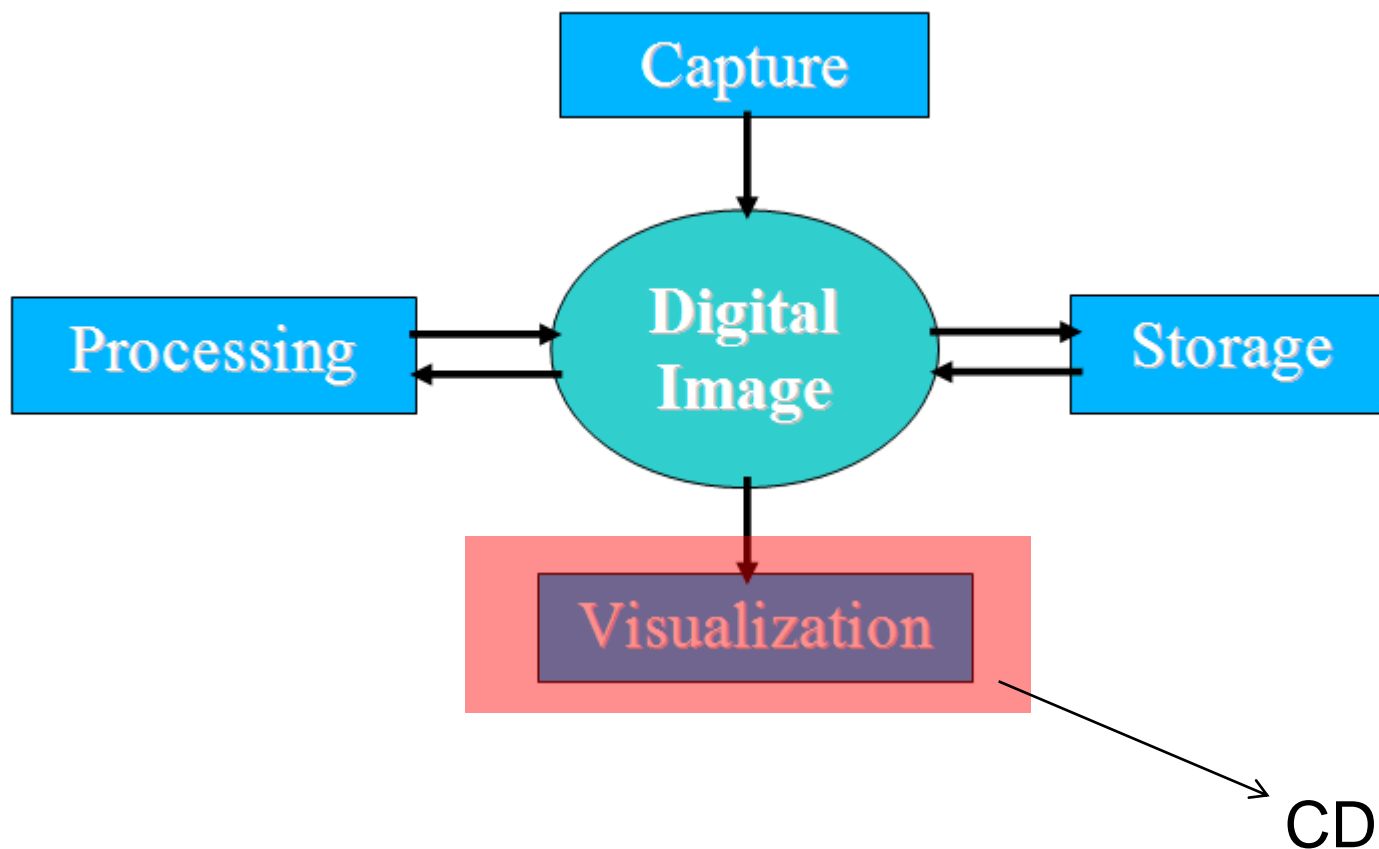


- CONTINUOUS
- - - - DASHED
- DOTTED
- . - . DASH_DOT
- . . . DASH_DOT_DOT



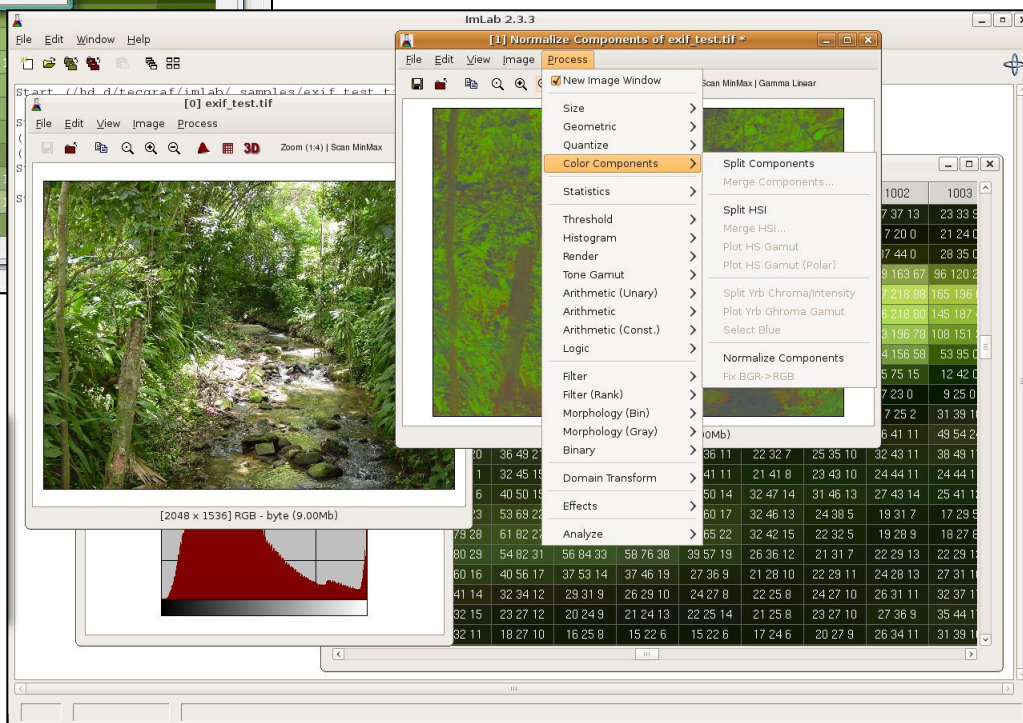
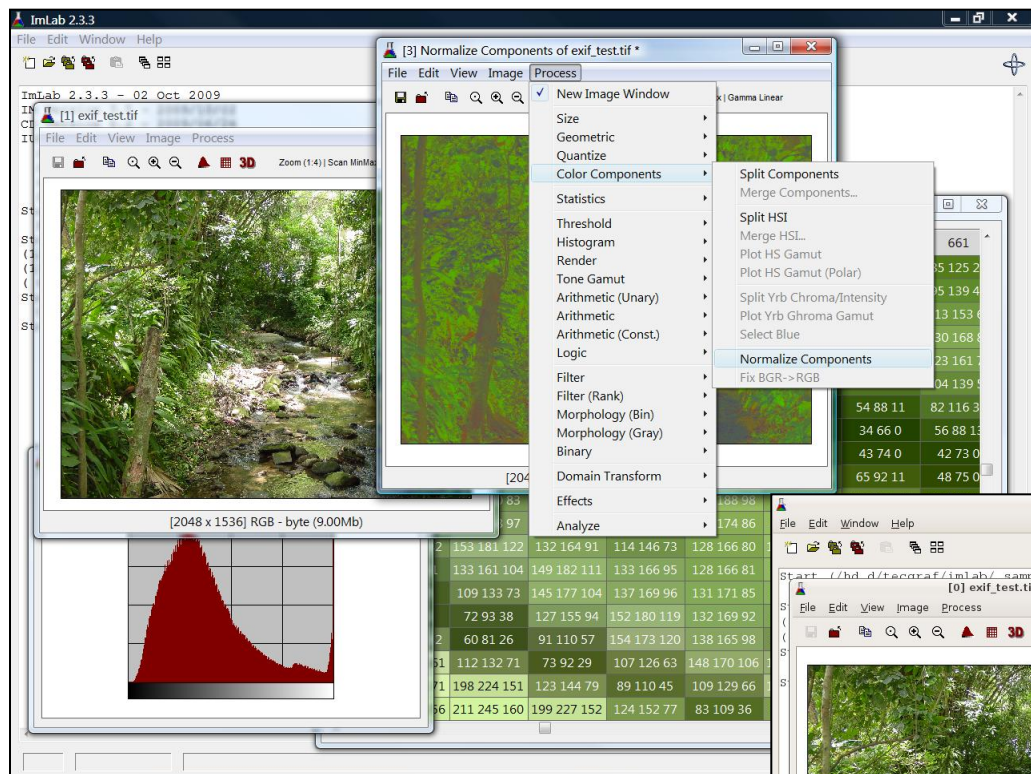
-  HORIZONTAL
-  VERTICAL
-  FDIAGONAL
-  BDIAGONAL
-  CROSS
-  DIAGCROSS





Exemplo de uma Aplicação Completa

Windows Vista



GTK em Gnome



- **IUP**: layout abstrato. Construção do diálogo sem o posicionamento manual dos controles.
- **CD**: mesma API para diversos tipos de Canvas (Janela, Impressora, Clipboard, Imagem, Postscript)
- **IM**: foco em aplicações científicas mas flexibilidade para atender quaisquer aplicações.



- Inicialização

```
require"imlua"
```

(LUA_CPATH)

```
require"cdlua"
```

```
require"iuplua"
```

- Regras gerais

- Funções: `imXXX` => `im.XXX`

- Definições: `IM_XXX` => `im.XXX`

- Métodos: `imFileXXX(ifile,...)` => `ifile:XXX(...)`

- Construtores especiais para IUP



- **Metatable:**

- `IupSetAttribute(label, "TITLE", "test")` =>
 `label.title = "test"`
- `title = IupGetAttribute(label, "TITLE")` =>
 `title = label.title`
- `IupSetCallback(button, "ACTION", button_action_cb);` =>
 `function button:action() ... end`

- **Construtores:**

- `IupButton("test")` =>
 `iup.button{title = "test", alignment="acenter"}`
- `IupHbox(bt1, bt2, NULL)` =>
 `iup.hbox{bt1, bt2, margin="10x10"}`

Exemplos IM - Indexação

```
require"imlua"  
  
local filename = "lena.jpg"  
local image = im.FileImageLoad(filename)  
  
local r = image[0]  
local g = image[1]  
local b = image[2]  
  
for row = 0, image:Height() - 1, 10 do  
    for column = 0, image:Width() - 1, 10 do  
        r[row][column] = 0  
        g[row][column] = 0  
        b[row][column] = 0  
    end  
end  
  
image:Save("lena_indexing.bmp", "BMP")
```



```
image, err = im.FileImageLoad(file_name);
```

```
if (err and err ~= im.ERR_NONE) then  
    error(err_msg[err+1])  
end
```

```
local new_image = im.ProcessCropNew(image, x1, x2, y1, y2)  
image:Destroy()
```

```
new_image:Save(file_name, "JPEG");  
new_image:Destroy()
```



```
local image = im.FileImageLoad(filename)
```

```
local gray = im.ImageCreate(image:Width(), image:Height(), im.GRAY,  
    image:DataType())
```

```
im.ConvertColorSpace(image, gray)
```

```
gray:Save("lena_gray.jpg", "JPEG")
```

```
local binary = im.ImageCreate(image:Width(), image:Height(), im.BINARY,  
    image:DataType())
```

```
im.ProcessSliceThreshold(gray, binary, 0, 128)
```

```
binary:Save("lena_binary.jpg", "JPEG")
```

```
local region = im.ImageCreate(image:Width(), image:Height(), im.GRAY,  
    im.USHORT)
```

```
local count = im.AnalyzeFindRegions(binary, region, 4, 1)
```

```
print("regions: ", count)
```



```
local canvas = cd.CreateCanvas(cd.PS, "cdtest.ps")

canvas:MarkSize(40)
canvas:Mark(x, y)

canvas:Font("Courier", cd.PLAIN, 12)
canvas:TextAlignment(cd.CENTER)
canvas:Text(x, y, text)

local xmin, xmax, ymin, ymax = canvas:GetTextBox(x, y, text)
canvas:Rect(xmin, xmax, ymin, ymax)

canvas:Kill()
```



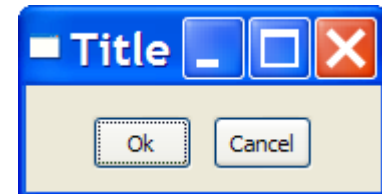

```
require"imlua"  
require"cdlua"  
require"cdluaim"  
  
local image = im.ImageCreate(500, 500, im.RGB, im.BYTE)  
local canvas = image:cdCreateCanvas() -- Creates a CD_IMAGERGB canvas  
  
canvas:Activate()  
canvas:Background(cd.EncodeColor(255, 255, 255))  
canvas:Clear()  
fgcolor = cd.EncodeColor(255, 0, 0) -- red  
fgcolor = cd.EncodeAlpha(fgcolor, 50) -- semi transparent  
canvas:Foreground(fgcolor)  
canvas:Font("Times", cd.BOLD, 24)  
canvas:Text(100, 100, "Test")  
canvas:Line(0,0,100,100)  
canvas:Kill()  
  
image:Save("new.bmp", "BMP")
```

Exemplos IUP – Layout Abstrato



```
dlg = iup.dialog
{
  iup.hbox
  {
    iup.fill{},
    iup.button{title="Ok",size="40"},
    iup.button{title="Cancel",size="40"},
    iup.fill{}
    ;margin="10x10", gap="10"
  }
  ;title="Test"
}
```

```
dlg:show()
```





```
btok = dlg[1][2]
btcancel = dlg[1][3]

btok.bgcolor = "255 0 0"

function btok:action()
    local aux = self.fgcolor
    self.fgcolor = self.bgcolor
    self.bgcolor = aux
End

function btcancel:action()
    return iup.CLOSE
end
```

Exemplo com IUP, CD e IM

```
image = im.FileImageLoad("flower.jpg")
cnv = iup.canvas{rastersize = image:Width().."x"..image:Height(), border
    = "NO"}
cnv.image = image -- store the new image in the IUP canvas as an
    attribute

function cnv:map_cb()
    -- the CD canvas can only be created when the IUP canvas is mapped
    self.canvas = cd.CreateCanvas(cd.IUP, self)
end

function cnv:action()
    -- called everytime the IUP canvas needs to be repainted
    self.canvas:Activate()
    self.canvas:Clear()
    self.image:cdCanvasPutImageRect(self.canvas, 0, 0, 0, 0, 0, 0, 0, 0)
end

dlg = iup.dialog{cnv}
dlg:show()
```

Antonio Scuri
scuri@tecgraf.puc-rio.br

