

# Canvas-2D

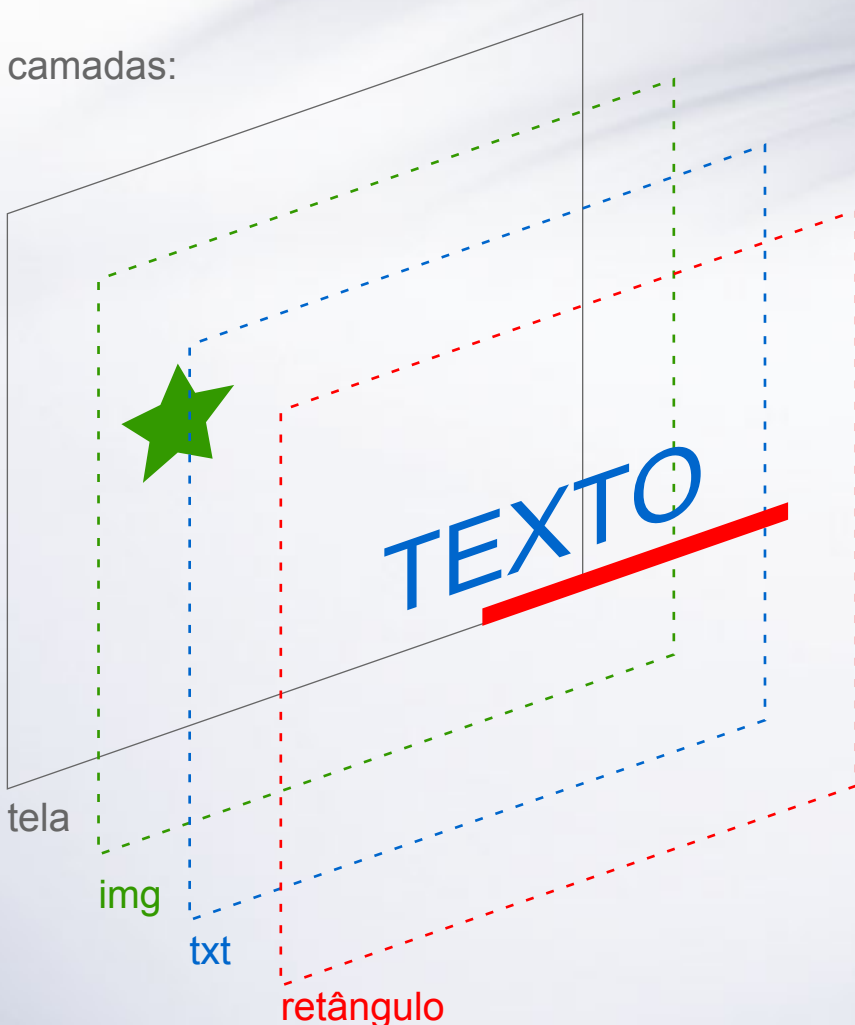
extremamente rápido usando EFL

PythonBrasil[7]  
São Paulo

Gustavo Svezut Barbieri

# o que é um canvas?

camadas:



tela  
img  
txt  
retângulo

etapas de pintura:

- 1
- 2
- 3
- 4

A **ordem** de composição das camadas é importante!

The diagram illustrates the layers of a canvas. It shows a white background with a green star, blue text 'TEXTO', and a red rectangle. The layers are labeled: 'tela' (screen), 'img' (image), 'txt' (text), and 'retângulo' (rectangle). The painting steps are shown in four stages: 1. Empty canvas. 2. Green star added. 3. Blue text 'TEXTO' added. 4. Red rectangle added, crossing over the text.

# otimizações comuns

The diagram is divided into two main parts. The left part shows a sequence of four panels (1a, 1b, 2a, 2b) illustrating the identification and marking of 'dirty areas' (áreas sujas) for cleaning. Panel 1a shows a black circle on a white background. Panel 1b shows the same circle with a dashed square around it labeled 'área suja'. Panel 2a shows a black circle and a black star on a white background. Panel 2b shows the same scene with dashed squares around both the circle and the star, indicating they are 'dirty areas'. Arrows indicate a clockwise sequence from 1a to 1b, 1b to 2a, and 2a to 2b.

The right part of the diagram illustrates occlusion. It shows two overlapping rectangles: a blue one in front of a yellow one. A 3D perspective view shows the blue rectangle is closer to the viewer. To the right, a 2D grid shows the intersection of the two rectangles. The text 'Oclusão a área não precisa ser pintada de azul' (Occlusion the area does not need to be painted blue) explains that the area behind the blue rectangle does not need to be painted blue.

**Áreas sujas** são partes que precisam ser pintadas (com o fundo) para limpar a imagem. Lembre-se que existe uma ORDEM CERTA!

# desenho imediato x retido

---

## - imediato:

- mais simples de implementar
- otimizações a cargo do usuário
- sujeito a retrabalho de pintura
- múltiplas requisições = múltiplas pinturas

## - retido:

- mais complexo para implementar
- pode otimizar pintura
- baseado em estado final - inicial
- múltiplas requisições != múltiplas pinturas

- canvas-2D, retido e muito otimizado
- software e hardware
- também trata foco e eventos (mouse, teclado...)
- apenas primitivas básicas!
- ...geralmente só se usa imagem, retângulo e texto
- conceito de grupos com "smart objects"
- ...delega ações para o usuário
- faz parte das EFL

# o que são EFL?

---

- base para o gerenciador de janelas Enlightenment
- desenvolvidas desde 1997 (atual desde 2001)
- foco em desempenho (cpu + memória)
- desenvolvidas para Linux, portadas para Win e Mac
- ...primeiro framework portado para PS3 nativo!

# componentes chave das EFL

---

- estrutura de dados: eina
- persistência de dados: eet
- abstração de sistema básico: ecore
- comunicação e redes: ecore\_con, ecore\_con\_url, azy
- canvas: evas
- temas: edje
- widgets: elementary
- áudio e vídeo: emotion
- html5/js/css: webkit-efl
- outros: e\_dbus, ethumb, efreet, epdf, eps...

# EFL e python

---

- porte criado em 2007 pelo INdT
- ainda mantido pela ProFUSION e comunidade
- desenvolvido "sob demanda"
- cobre grande parte da API
- utilizado em projetos embarcados:
  - Maemo: Canola2, BlueMaemo
  - OpenMoko: padrão desde 2008
- dupla-face:
  - API pythônica, com properties e decorators
  - API C-like para reuso de documentação EFL



```
import elementary, evas

win = elementary.Window("test", elementary.ELM_WIN_BASIC)
win.title_set("Hello World")
win.autodel_set(True)

bg = elementary.Background(win)
win.resize_object_add(bg)
bg.size_hint_weight_set(evas.EVAS_HINT_EXPAND, evas.EVAS_HINT_EXPAND)
bg.show()

def on_click(obj):
    print "clicked:", obj
```

```
btn = elementary.Button(win)
win.resize_object_add(btn)
btn.text = "Hello"
btn.callback_clicked_add(on_click)
btn.show()

win.resize(320, 240)
win.show()

elementary.policy_set(
    elementary.ELM_POLICY_QUIT,
    elementary.ELM_POLICY_QUIT_LAST_WINDOW_CLOSED)
elementary.run()
```

# principais tecnologias

---

- evas
- ecore
- edge
- elementary

- focado em retângulo, imagens e texto
- imagens com diversas propriedades:
  - borda, load-cache, scale-cache, smooth-scale...
- texto (linha simples e bloco):
  - sombra suave/dura, contorno, "glow"
- transformações 2D/3D:
  - rotação 360, perspectiva, iluminação
- suporte simples a linhas e polígonos
  - pode ser usado para implementar curvas/caminhos
- independente de outras EFL

# evas - imagens

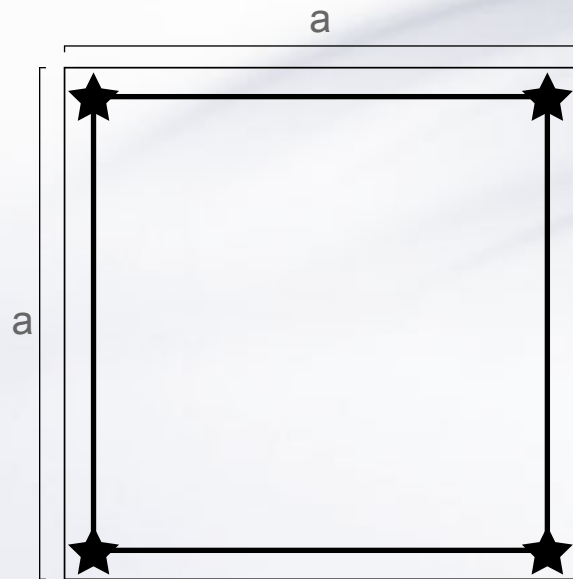
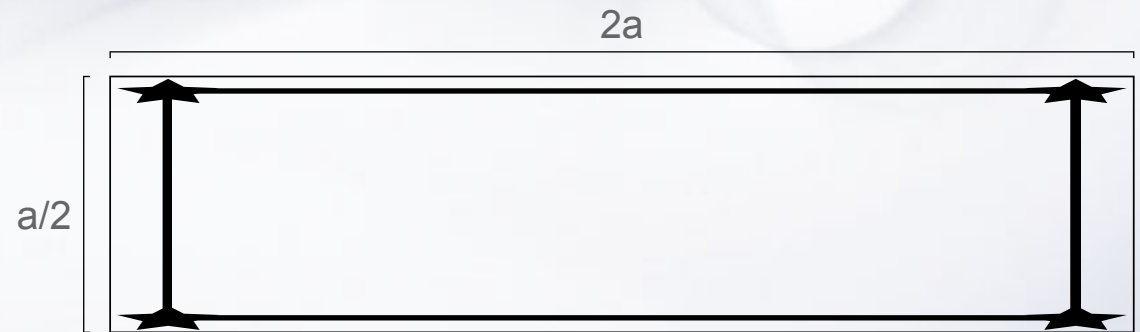
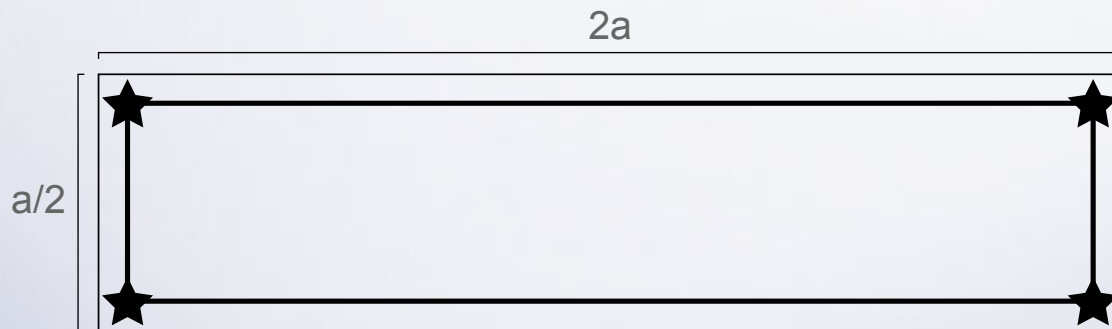


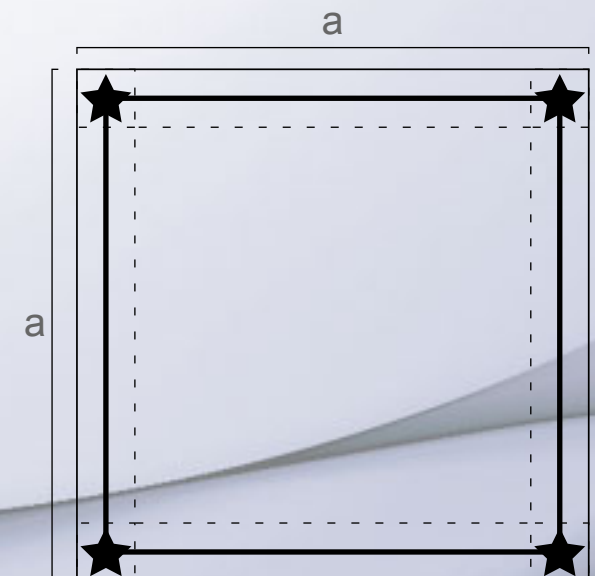
imagem original



redimensionar, sem que as bordas sejam especificadas



redimensionar, com as bordas especificadas



- laço principal (main loop):
  - temporizadores (timers)
  - arquivos e rede (file descriptor watchers/handlers)
  - idler, idler enterer, idler exiter
  - outros: eventos, jobs, poller e threads
- abstração do sistema operacional
- acesso a vários dispositivos de entrada/saída
  - X11, DirectFB, FBdev, Win32, WinCE, Quartz...
  - tslib, multi-touch...
- integração com Evas (ecore-evras)

- máquina de estados de objetos Evas (partes)
- transições animadas
- programável em "embryo" (subconjunto de C)
- posicionamento absoluto (pixel) ou relativo
- comunicação via sinais e mensagens
- descrição textual simples (porém verbosa)
- compilado em formato otimizado
- ... embute fontes e imagens!

- estado: descreve propriedades
- parte: um objeto Evas com conjunto de estados
- programa: transição de estados, ação ou script
- grupo: conjunto de partes e programas
- edc - fonte (texto), conjunto de grupos
- edj - binário compilado à partir do edc
- outros: listagem de fontes, imagens, estilos...



```
collections {
  group { name: "main";

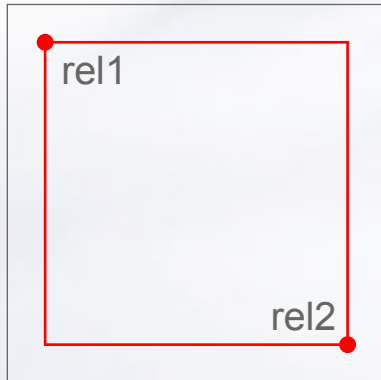
  parts {
    part { name: "fundo";
      type: RECT;
      mouse_events: 0;
      description { state: "default" 0.0;
        color: 255 0 0 255;
      }
    }
  }
}
```

```
part { name: "texto";  
    type: TEXT;  
    effect: SOFT_SHADOW;  
    description { state: "default" 0.0;  
        color: 255 255 255 255;  
        color3: 0 0 0 255;  
        rel1.relative: 0.1 0.1;  
        rel2.relative: 0.9 0.9;  
        text { font: "Sans:style=Bold";  
            size: 32;  
            text: "Olá Mundo!";  
        }  
    }  
}
```

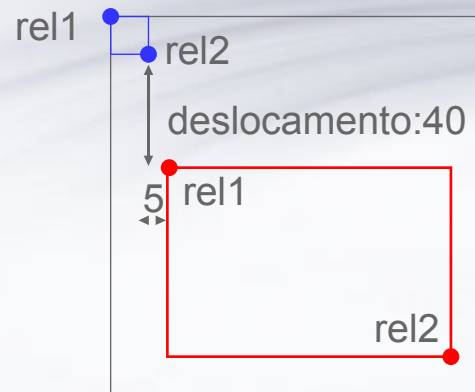
```
part { name: "rect";  
    type: RECT;  
    mouse_events: 0;  
    description { state: "default" 0.0;  
        color: 0 0 255 128;  
        rel1 {  
            relative: 0.0 0.5;  
            to: "texto";  
        }  
    }  
    description {  
        state: "escondido" 0.0;  
        color: 0 0 255 0;  
        rel1.relative: 0.0 0.0;  
    }  
}
```

```
programs {
    program { signal: "mouse,in";
        source: "texto";
        action: STATE_SET "escondido" 0.0;
        target: "rect";
        transition: LINEAR 0.5;
    }
    program { signal: "mouse,out";
        source: "texto";
        action: STATE_SET "default" 0.0;
        target: "rect";
        transition: LINEAR 0.5;
    }
}
}
```

# edje - rel1, rel2



```
parte:  
  rel1.relative: 0.1 0.1;  
  rel2.relative: 0.9 0.9;  
resultado:  
  x1 = largura * 0.1  
  y1 = altura * 0.1  
  
  x2 = largura * 0.9  
  y2 = altura * 0.9
```



```
parte1:  
  rel2 {  
    relative: 0.0 0.0;  
    offset: 10 10;  
  }  
parte2:  
  rel1 {  
    to: "parte1";  
    relative: 1.0 1.0;  
    offset: 5 40;  
  }  
rel2.relative: 0.9 0.9;
```

```
padrão:  
  rel1 {  
    relative: 0.0 0.0;  
    offset: 0 0;  
  }  
  rel2 {  
    relative: 1.0 1.0;  
    offset: -1 -1;  
  }
```

- conjunto de elementos gráficos (widgets)
- usa principalmente Edje e Evas
- facilita casos comuns:
  - janela
  - entradas de texto
  - organização (layout, box, table)
  - rolagem e visualização (viewport)
  - botão, lista, grade, ícones...

- PyGame é melhor suportado em diversas plataformas
- porém EFL ganha em geral:
  - mais otimizado do que sprites.Render\*
  - mais primitivas gráficas
  - main-loop, timers, animators, idlers, ...
  - edje para temas em geral
  - widgets com temas avançados
  - suporte a vídeo bem integrado
  - webkit "di grátis"

# problemas das EFL

- comunidade pequena (e um tanto rude ;-P)
- falta de pacotes
- poucos desenvolvedores para Win e Mac
- foco em 2D, porém suporta transformações/perspectiva
- poucas primitivas 2D (sem paths e afins)
- curva de aprendizado



# considerações finais

---

- pontos positivos compensam os negativos
- edje faz muita diferença! acredite!
- elementary é uma mão na roda
- webkit-efl abre novas portas

# demos!

---

- expedite: stress-test e benchmark
- emotion: playback de vídeo com transparência
- elementary\_test: widgets
- eve: webkit-efl

Obrigado!

Gustavo Sverzut Barbieri

<barbieri@profusion.mobi>