Computação Gráfica em

python™

Luiz Eduardo Borges

Objetivo

Foto alterada com o filtro *Cubism* do GIMP.

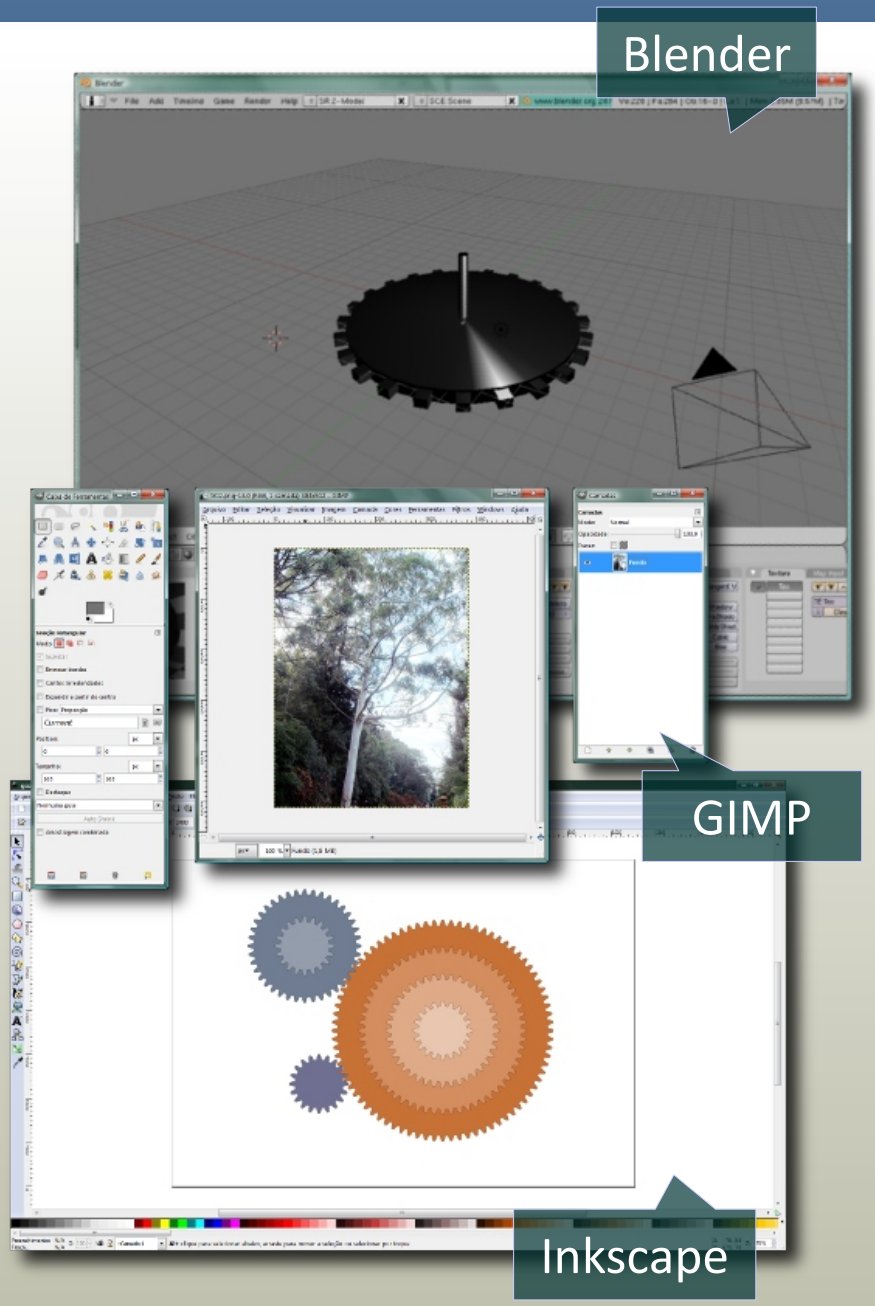
- Apresentar algumas soluções *Open Source* para Computação Gráfica com o uso de **Python**.
- Demonstrar as soluções em questão através de exemplos.

Cena 3D gerada no Blender.

Definição:

- Área da Ciência da Computação que estuda a geração e manipulação de conteúdo visual.

Python em Computação Gráfica



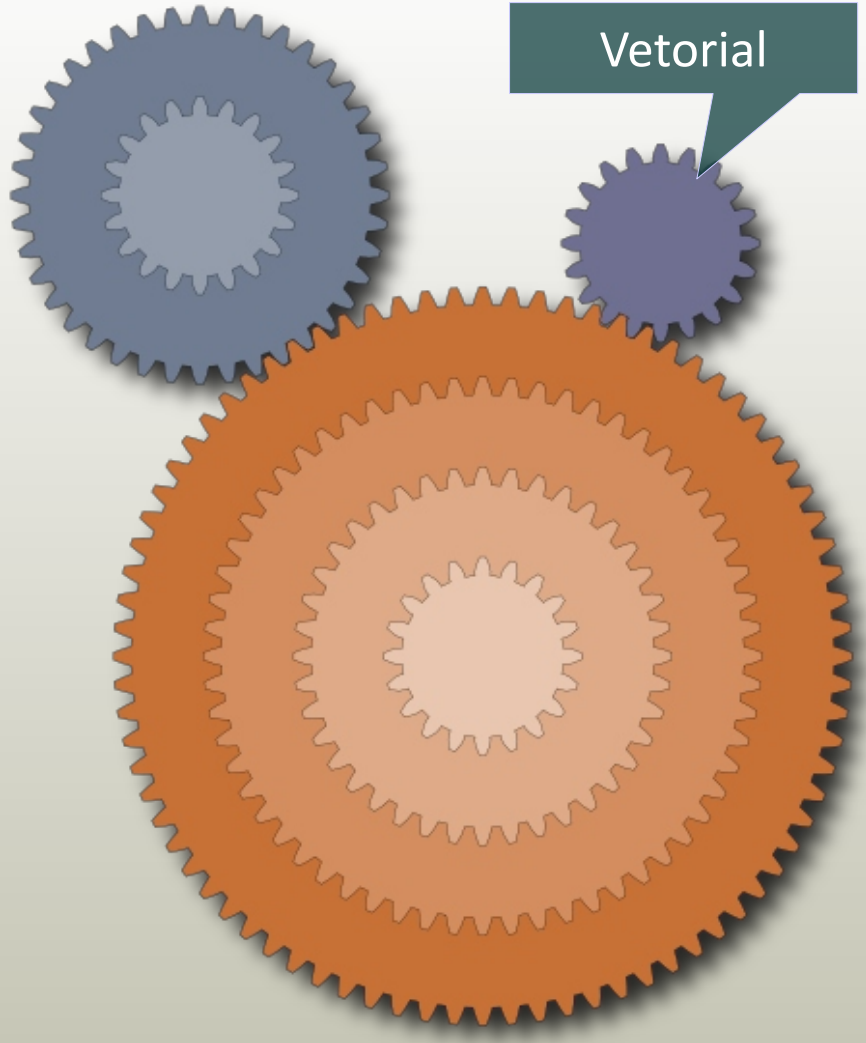
Por que **Python** ao invés de outras linguagens dinâmicas?

- Muitos *frameworks* e bibliotecas disponíveis.
- Várias aplicações, tais como **Blender**, **GIMP** e **Inkscape**, usam a linguagem para ampliar funcionalidades e/ou automatizar rotinas.

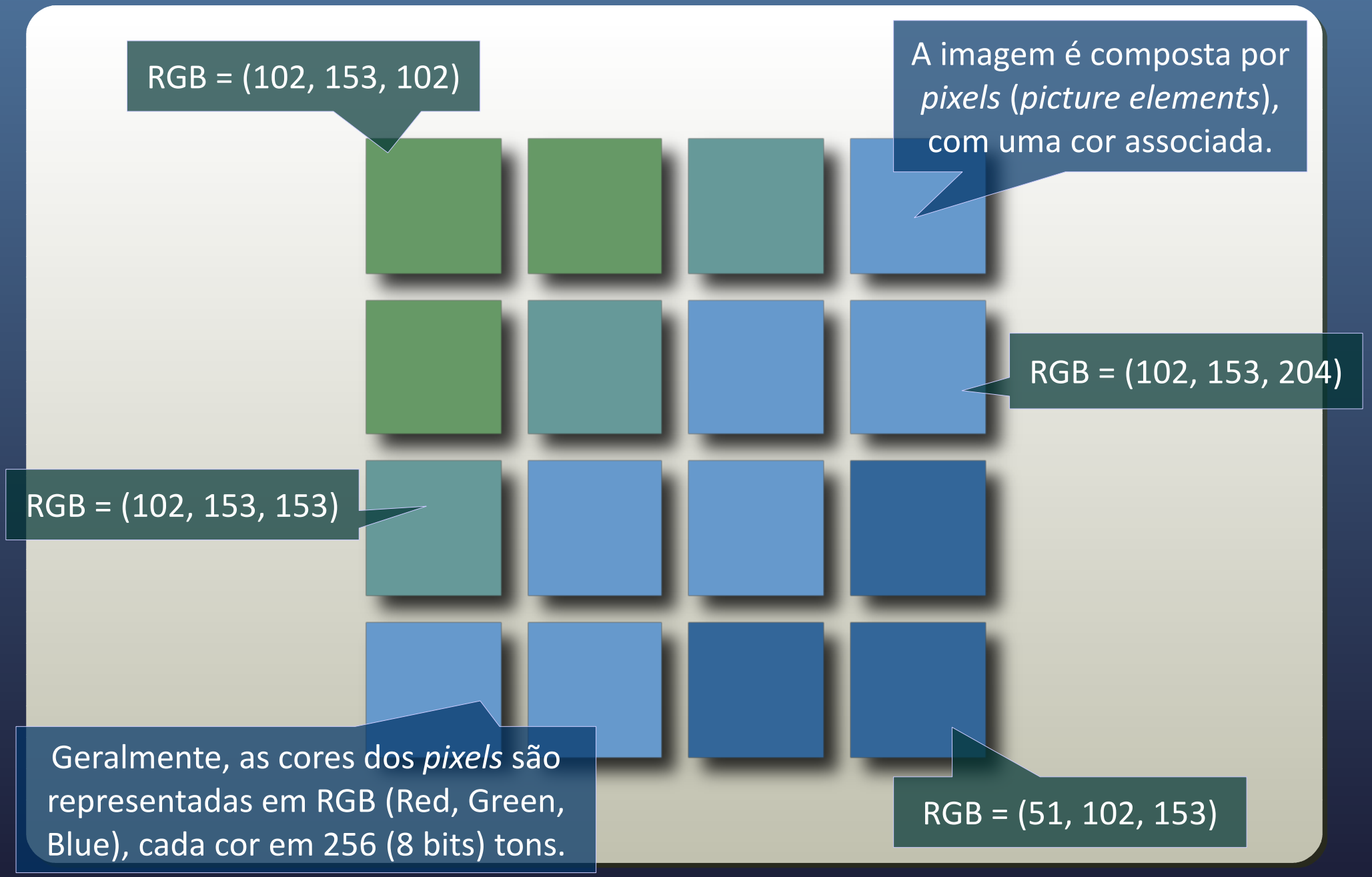
Formas de representação de imagem 2D

Mapa de *bits*

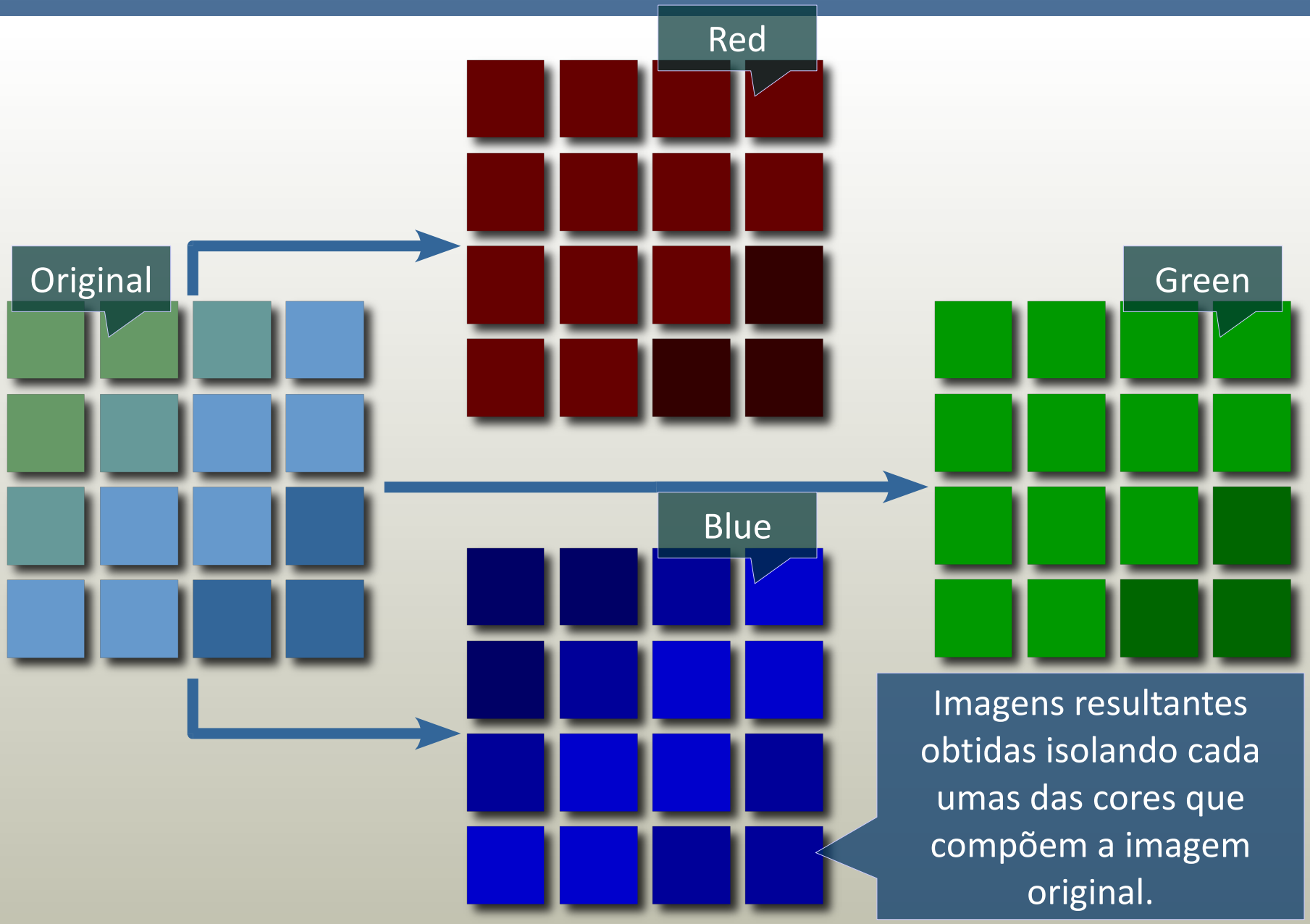
Vetorial



Mapas de *bits* (*Raster*)

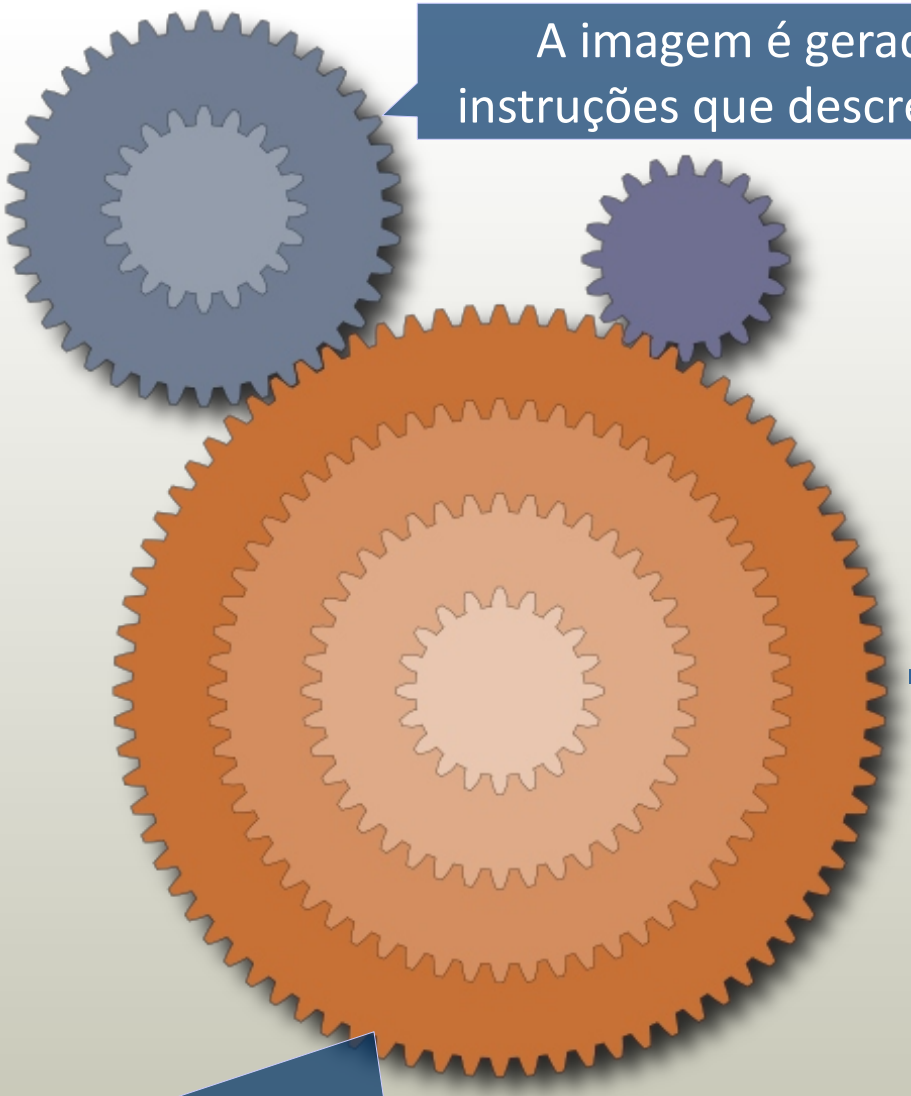


Canais


Red
Original
Green
Blue
Imagens resultantes obtidas isolando cada umas das cores que compõem a imagem original.

Imagens Vetoriais

A imagem é gerada a partir de instruções que descrevem os objetos.



gears.svg - SciTE

File Edit Search View Tools Options Language Buffers Help

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- Created with Inkscape (http://www.inkscape.org/) -->
<svg
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:cc="http://creativecommons.org/ns#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:svg="http://www.w3.org/2000/svg"
   xmlns="http://www.w3.org/2000/svg"
   xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-0.dtd"
   xmlns:inkscape="http://www.inkscape.org/namespaces/inkscape"
   width="1052.3622"
   height="744.09448"
   id="svg2"
   sodipodi:version="0.32"
   inkscape:version="0.46"
   sodipodi:docname="gears.svg"
   inkscape:output_extension="org.inkscape.output.svg.inkscape"
   version="1.0">
  <sodipodi:namedview
     id="base"
     pagecolor="#ffffff"
     bordercolor="#666666"
     borderopacity="1.0"
     inkscape:pageopacity="0.0"
     inkscape:pageshadow="2"
     inkscape:zoom="0.95283599"
     inkscape:cx="526.18109"
     inkscape:cy="372.04724"
     inkscape:document-units="px"
     inkscape:current-layer="layer1"
     showgrid="false"
     inkscape:window-width="1536"
     inkscape:window-height="921"
     inkscape:window-x="118"
     inkscape:window-y="43" />
```

line 1, column 1 (INS) (LF) - 0 chars selected

As instruções são primitivas geométricas, tais como linha, ponto, círculo e outros.

Arquivo SVG gerado no **Inkscape**.

Processamento de Imagem

Imagem original em um formato *raster* (mapa de *bits*).

Algoritmos

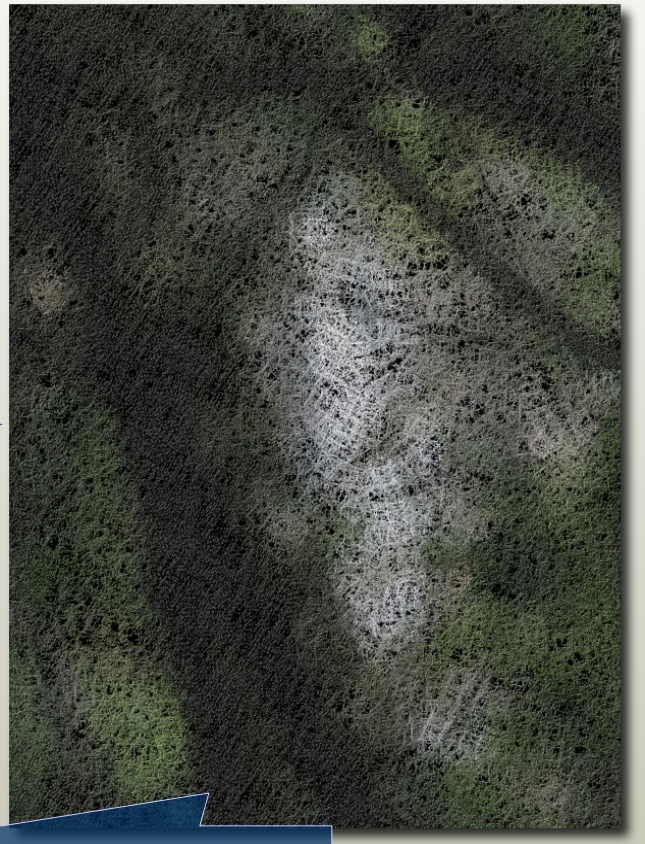


Imagem processada (filtro *GIMPressionist* / *Mossy*, do GIMP).

Processamento de Imagem (Exemplos)

Original

Contour

Emboss

Find edges

Alguns dos filtros disponíveis na Python Imaging Library (PIL).

PIL (Python Imaging Library)

Implementa:

- Ferramentas para cortar, redimensionar e mesclar imagens.
- Algoritmos de conversão.
- Filtros, tais como suavizar e detectar bordas.
- Ajustes, incluindo brilho e contraste.
- Operações com paletas de cores.
- Ferramentas básicas de desenho (linha, arco e outras).
- Rotinas para tratamento de imagens: equalizar, deformar, inverter e outras.

PIL (Visão Geral)

O módulo *Image* define uma classe básica que serve de abstração para a manipulação de imagens.

Objeto da classe *Image*

Um objeto da classe *Image* tem atributos como tamanho (*size*), formato (*format*) e modo (*mode*).

Os métodos fornecem rotinas de processamento de imagens, incluindo rotacionar (*rotate*) e redimensionar (*resize*).

Os outros módulos provêm recursos diversos, tais como filtros (*ImageFilter*), fontes (*ImageFonts*) e desenho (*ImageDraw*).

PIL (Exemplo I/II)

```
import Image
import ImageOps
import ImageFilter

img = Image.open('tr01.jpg')

img = ImageOps.autocontrast(img)
img = ImageOps.equalize(img)

img = img.filter(ImageFilter.SMOOTH)

img.thumbnail((512, 512),
   Image.ANTIALIAS)
img = ImageOps.expand(img,
   border=2, fill=1)

img.save('tr02.png', 'PNG')
```

O módulo *Image* define a classe para tratar imagens com vários métodos para modificar suas características.

A função *open()* cria objetos da classe *Image* a partir de arquivos. Vários formatos são suportados.

O módulo *ImageOps* implementa várias rotinas comuns de processamento.

O método *filter()* aplica o filtro especificado na imagem.

O método *thumbnail()* redimensiona a imagem.

O método *save()* grava a imagem em arquivo no formato especificado.

PIL (Exemplo II/II)

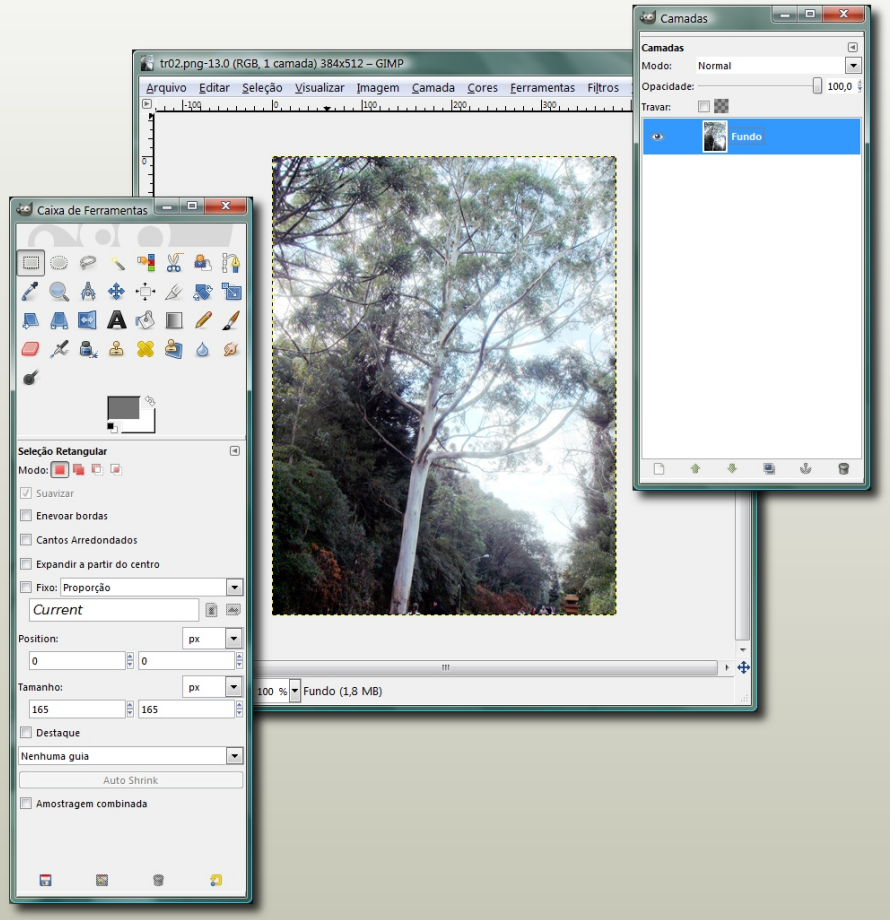
Original

Programa

Saída

Versão reduzida, suavizada, com borda e cores modificadas.

GIMP (GNU Image Manipulation Program)

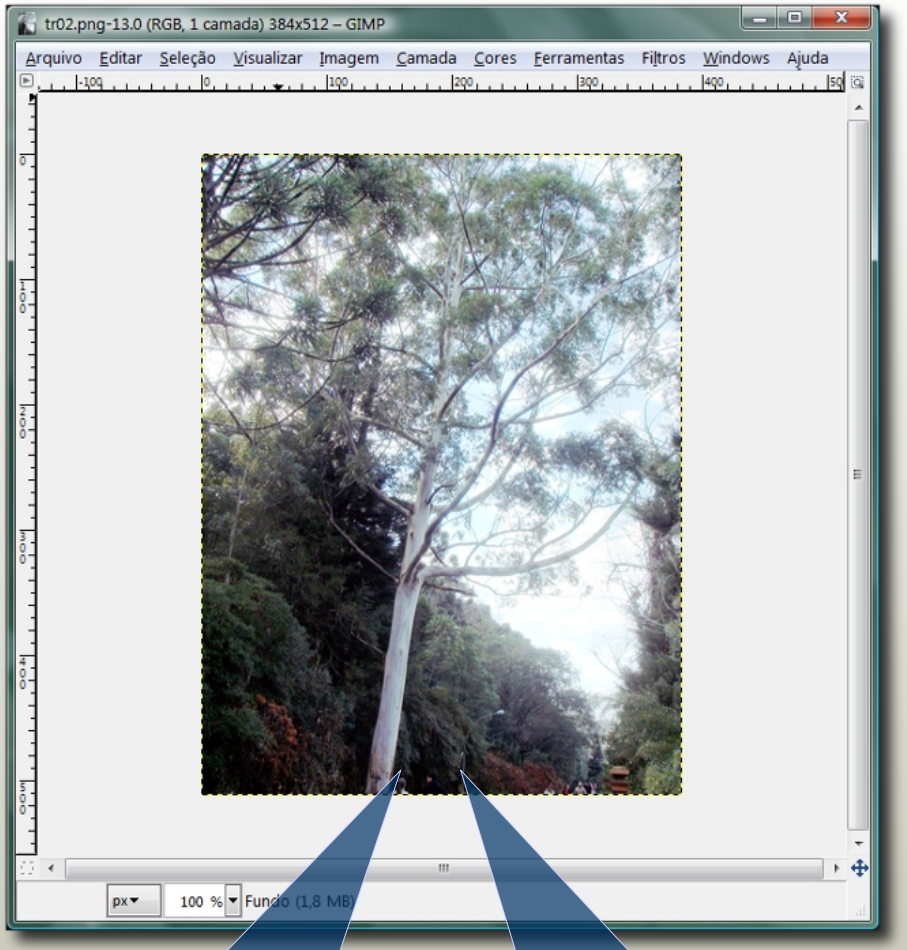


Implementa:

- Ferramentas para processamento e edição de imagens.
- Algoritmos de conversão.
- Manipulação de imagens compostas de múltiplas camadas.
- Uma API (**Python-Fu**) que permite a criação de *plug ins* usando **Python**.

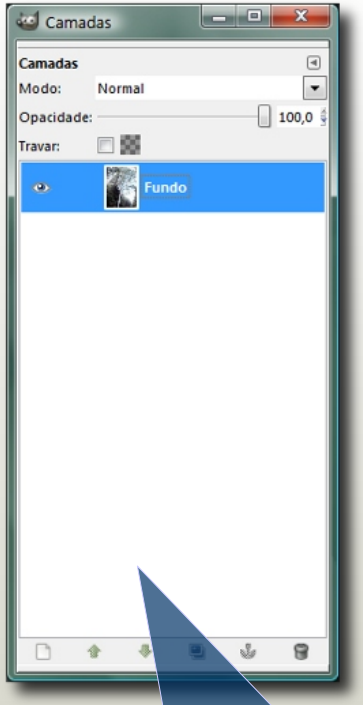
GIMP (Visão Geral)

Caixa de Ferramentas.



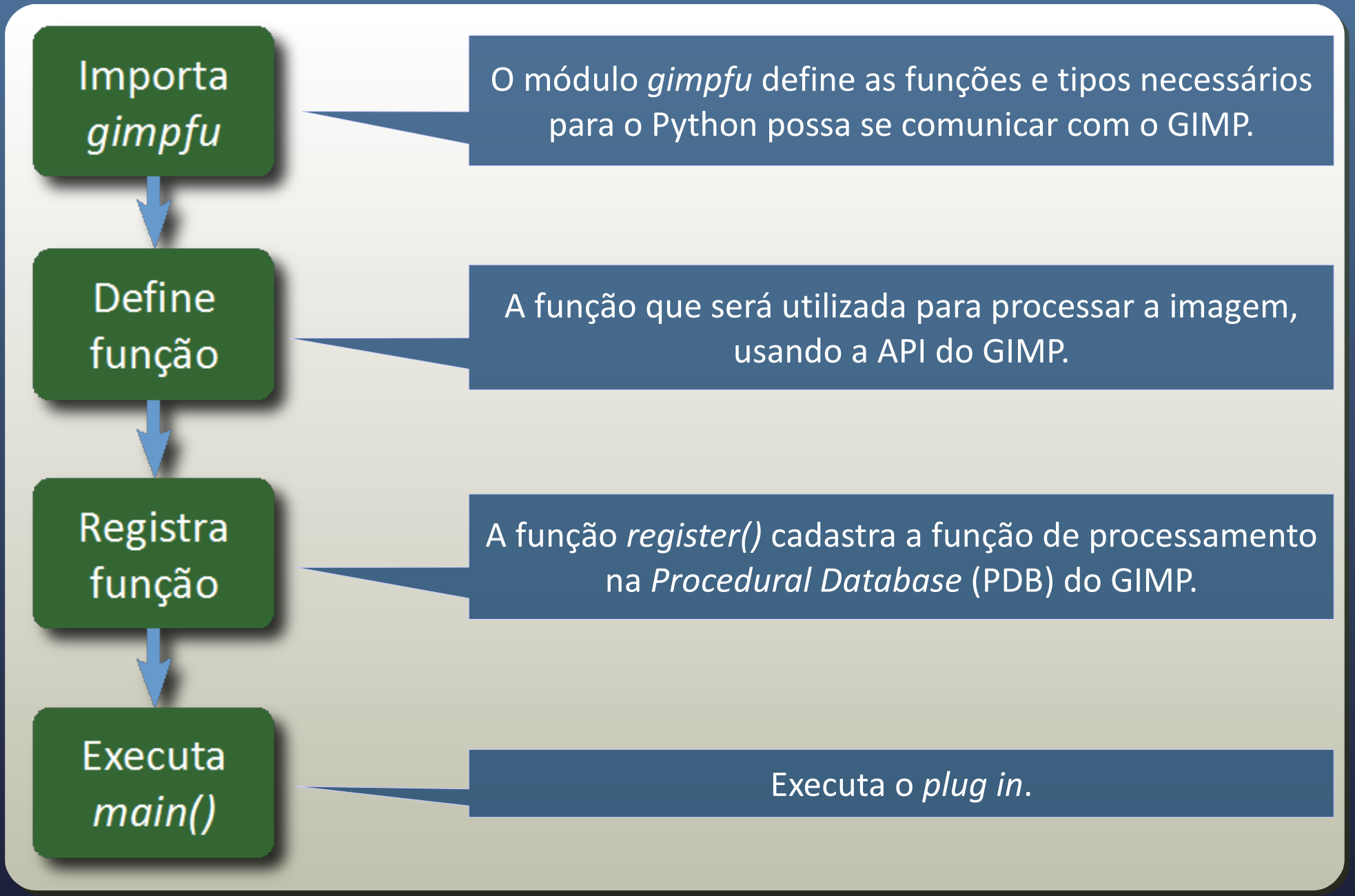
Cada camada pode ter um ou mais canais (*channels*).

Cada camada ou canal pode ser editado (*drawable*).

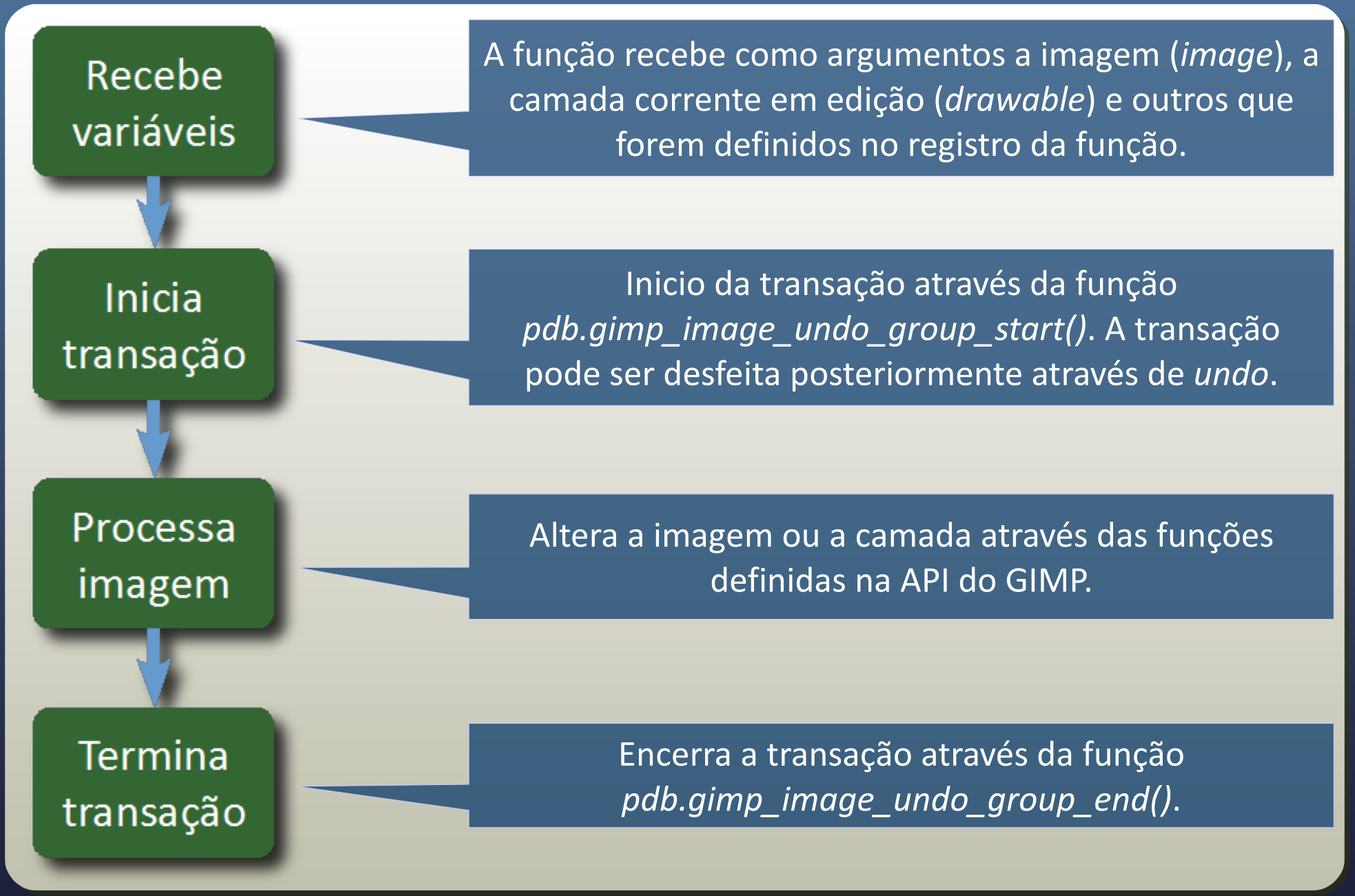


Cada imagem pode ter uma ou mais camadas (*layers*).

GIMP (*Plug in*)


Importa *gimpfu*
O módulo *gimpfu* define as funções e tipos necessários para o Python possa se comunicar com o GIMP.
Define função
A função que será utilizada para processar a imagem, usando a API do GIMP.
Registra função
A função *register()* cadastra a função de processamento na *Procedural Database* (PDB) do GIMP.
Executa *main()*
Executa o *plug in*.

GIMP (Função)


Recebe variáveis
A função recebe como argumentos a imagem (*image*), a camada corrente em edição (*drawable*) e outros que forem definidos no registro da função.
Inicia transação
Inicio da transação através da função *pdb.gimp_image_undo_group_start()*. A transação pode ser desfeita posteriormente através de *undo*.
Processa imagem
Altera a imagem ou a camada através das funções definidas na API do GIMP.
Termina transação
Encerra a transação através da função *pdb.gimp_image_undo_group_end()*.

GIMP (Exemplo I/IV)

```
from gimpfu import *

def stonify(img, drawable):
    pdb.gimp_image_undo_group_start(img)
    pdb.script_fu_lava(img, drawable, 10, 10, 7,
       'German flag smooth', 1, 1, 0)
    lava = img.layers[0]
    w, h = img.width, img.height
    rock = gimp.Layer(img, 'Rock', w, h, RGB_IMAGE,
       100, MULTIPLY_MODE)
    pdb.gimp_image_add_layer(img, rock, 0)
    pdb.plug_in_solid_noise(img, rock, 0, 0, 0, 1, 4, 4)
    pdb.plug_in_bump_map(img, rock, lava,
       135, 45, 15, 0, 0, 0, 0, 1, 0, 0)
    pdb.plug_in_bump_map(img, rock, drawable,
       135, 45, 30, 0, 0, 0, 0, 1, 0, 0)
    lava.visible = 0
```

Importa a interface com GIMP.

Inicia a transação para *undo*.

Cria uma camada com lava.

Cria uma camada de ruído.

Aplica revelo nas camadas.

Continua...

GIMP (Exemplo II/IV)

```
    img.flatten()
    pdb.gimp_brightness_contrast (img.layers[0], 30, 10)
    pdb.gimp_image_undo_group_end(img)

register(
    'Stonify',
    'Carve image in stone',
    'Carve image in stone with colors',
    'Luiz Eduardo Borges',
    'Luiz Eduardo Borges',
    '2008',
    '<Image>/Python-Fu/Stonify',
    '*',
    [],
    [],
    stonify)
main()
```

Termina a transação.

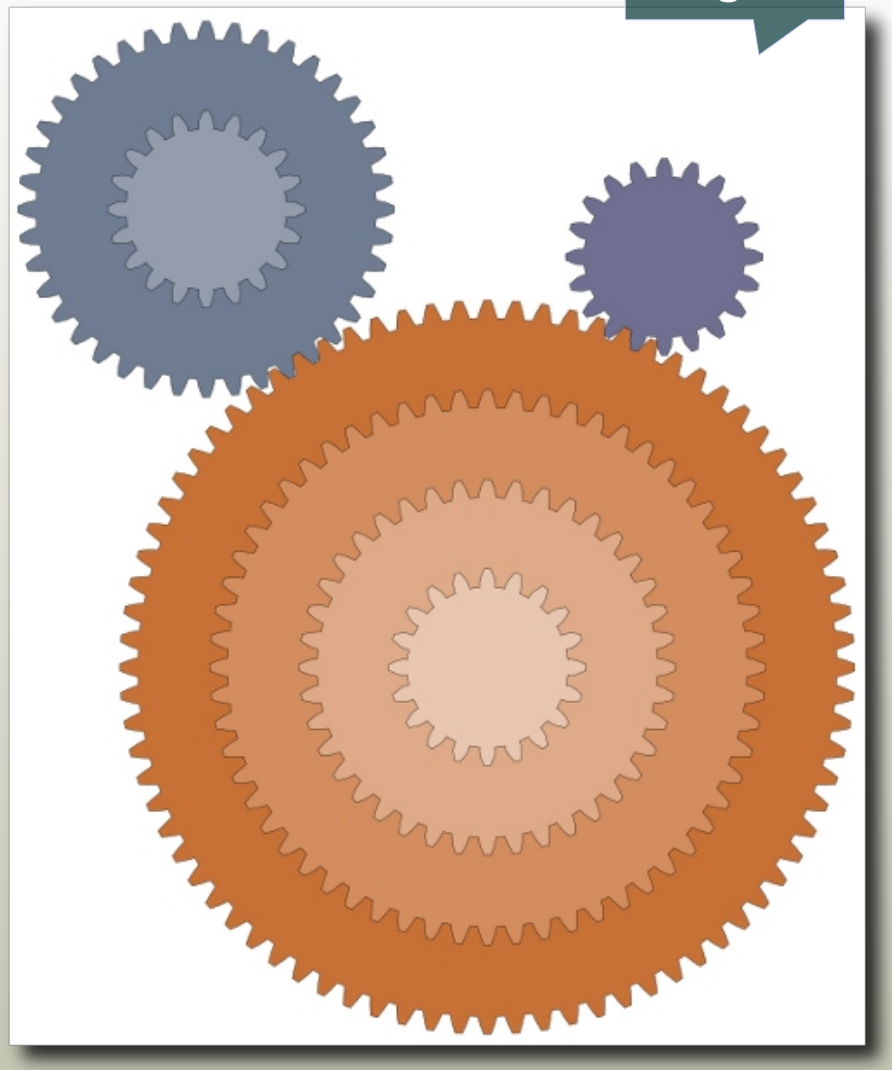
Registra a função na PDB.

Executa o *plug in*.

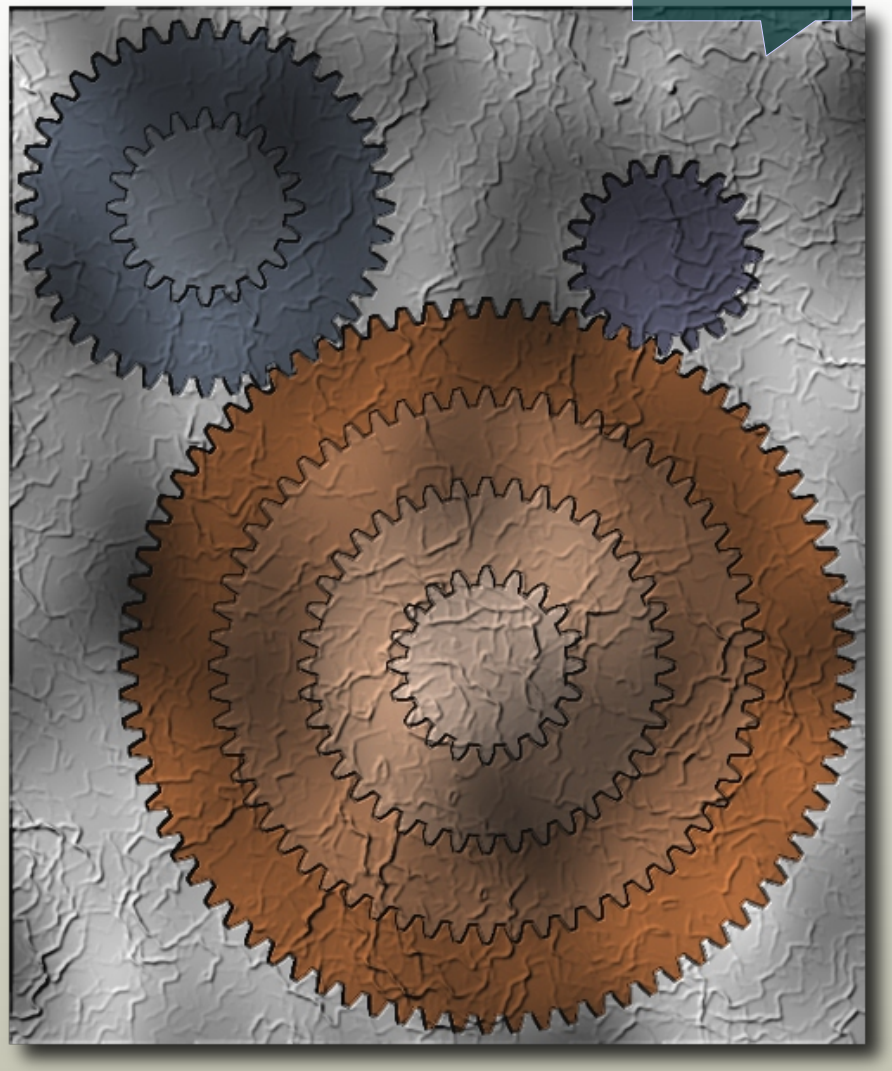
Continua...

GIMP (Exemplo III/IV)

Original

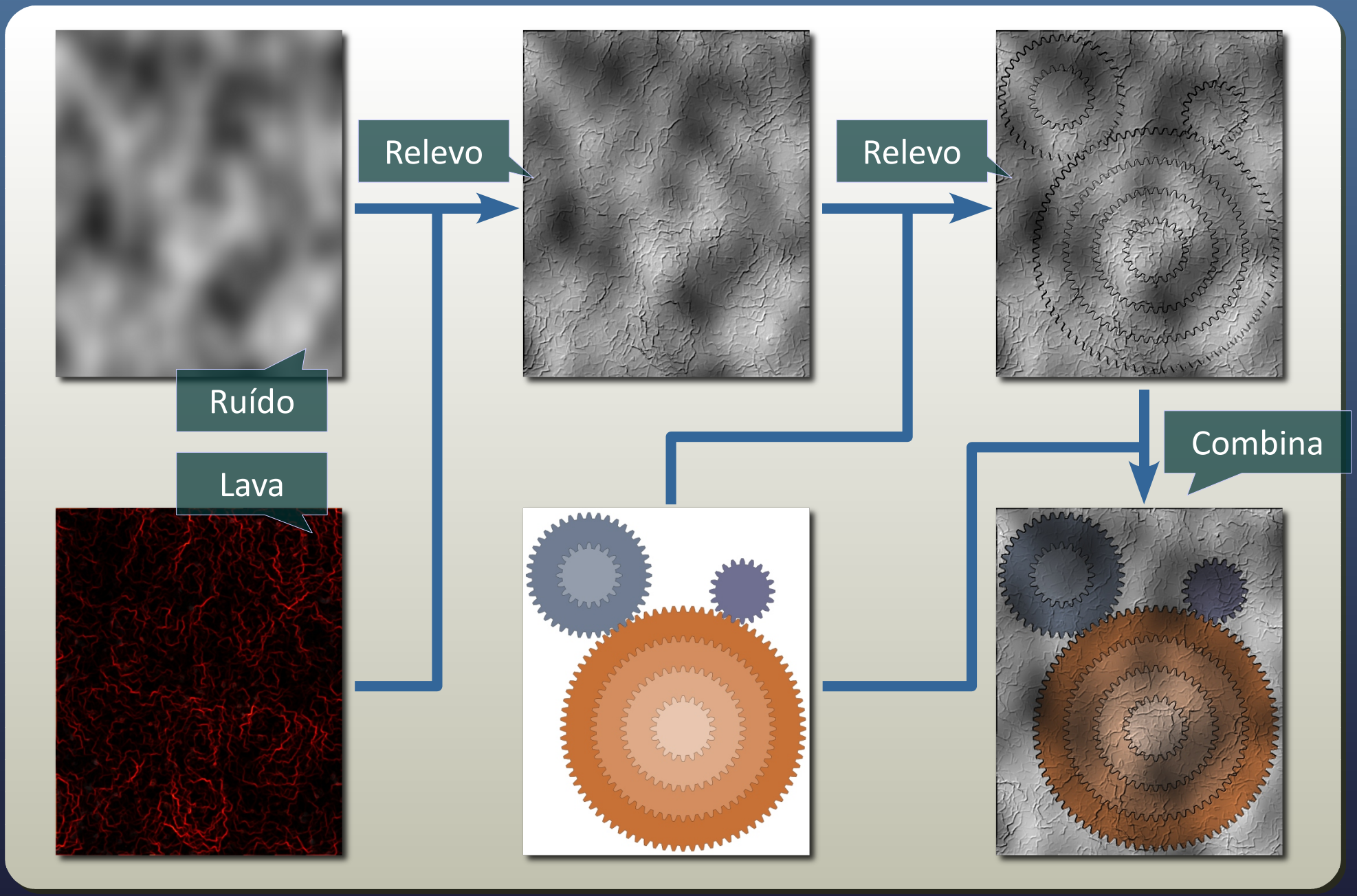


Saída



Continua...

GIMP (Exemplo IV/IV)

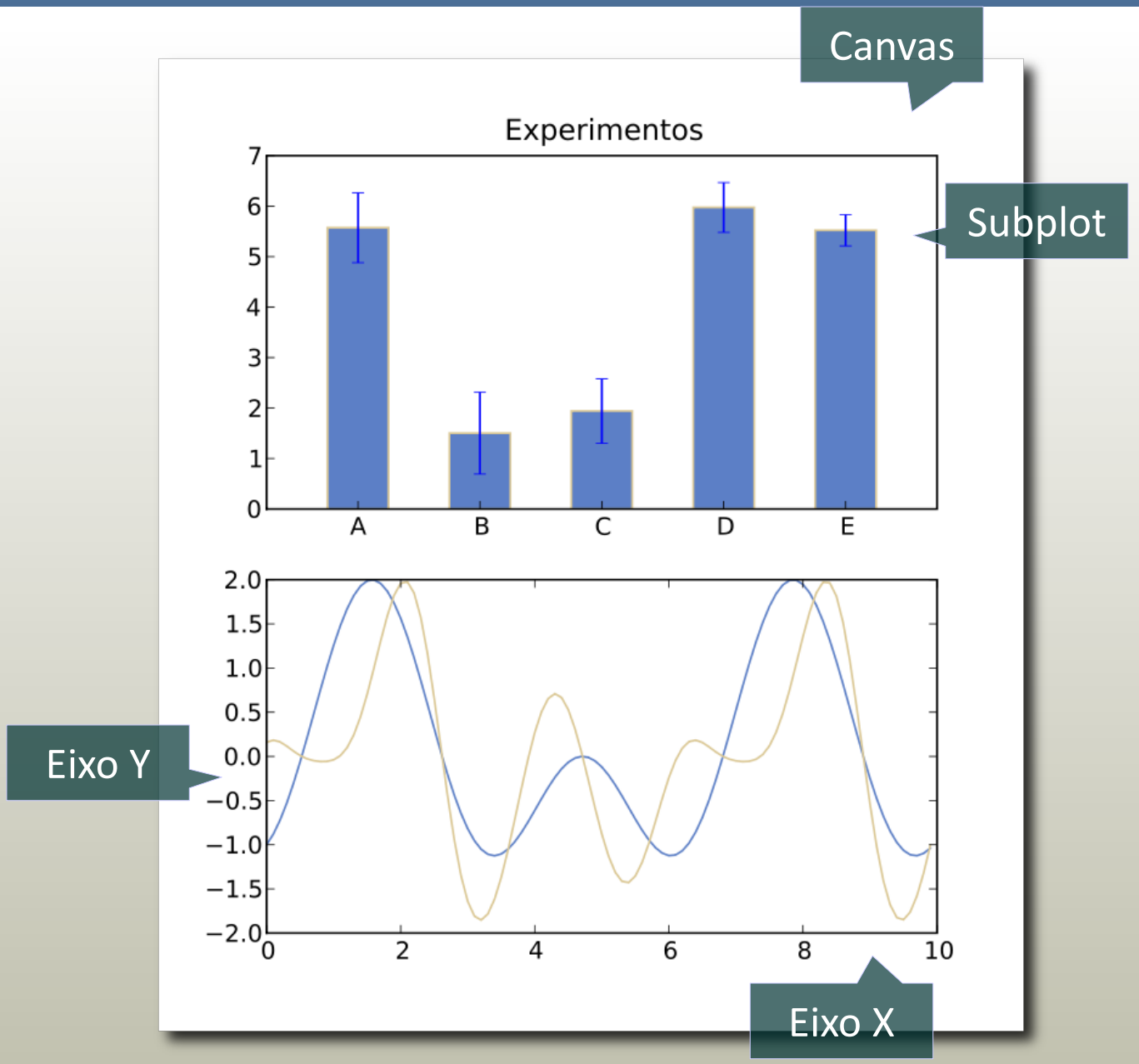


Pylab / Matplotlib

Implementa:

- **Pylab**: API destinada ao uso interativo.
- **Matplotib**: API orientada a objetos.
- Vários gráficos 2D mais usados: linhas, barras, pizza, dispersão e outros.
- Suporte a gráficos 3D (Beta).
- Vários *backends*, que permitem exportar para vários formatos populares.
- Suporte a extensões, como o **Basemap**.

Pylab / Matplotlib (Visão Geral)



Pylab / Matplotlib (Exemplo I/II)

```
from pylab import *

x, y = ogrid[-1:1:.01, -1:1:.01]

z = 6*x**2 + 6*y**2 + cos(16*sqrt(x**2 + y**2) + 3*arctan2(x,y))

ext = [-1, 1, -1, 1]
imshow(z, origin='lower', extent=ext)
contour(z, origin='lower', extent=ext)

xlabel('x')
ylabel('y')
title('z = 6*x**2 + 6*y**2 + cos(16*sqrt(x**2 + y**2) + 3*arctan2(x,y))')
savefig('graf.png')
```

A função *ogrid()* cria um arranjo de arranjos com os pontos da grade.

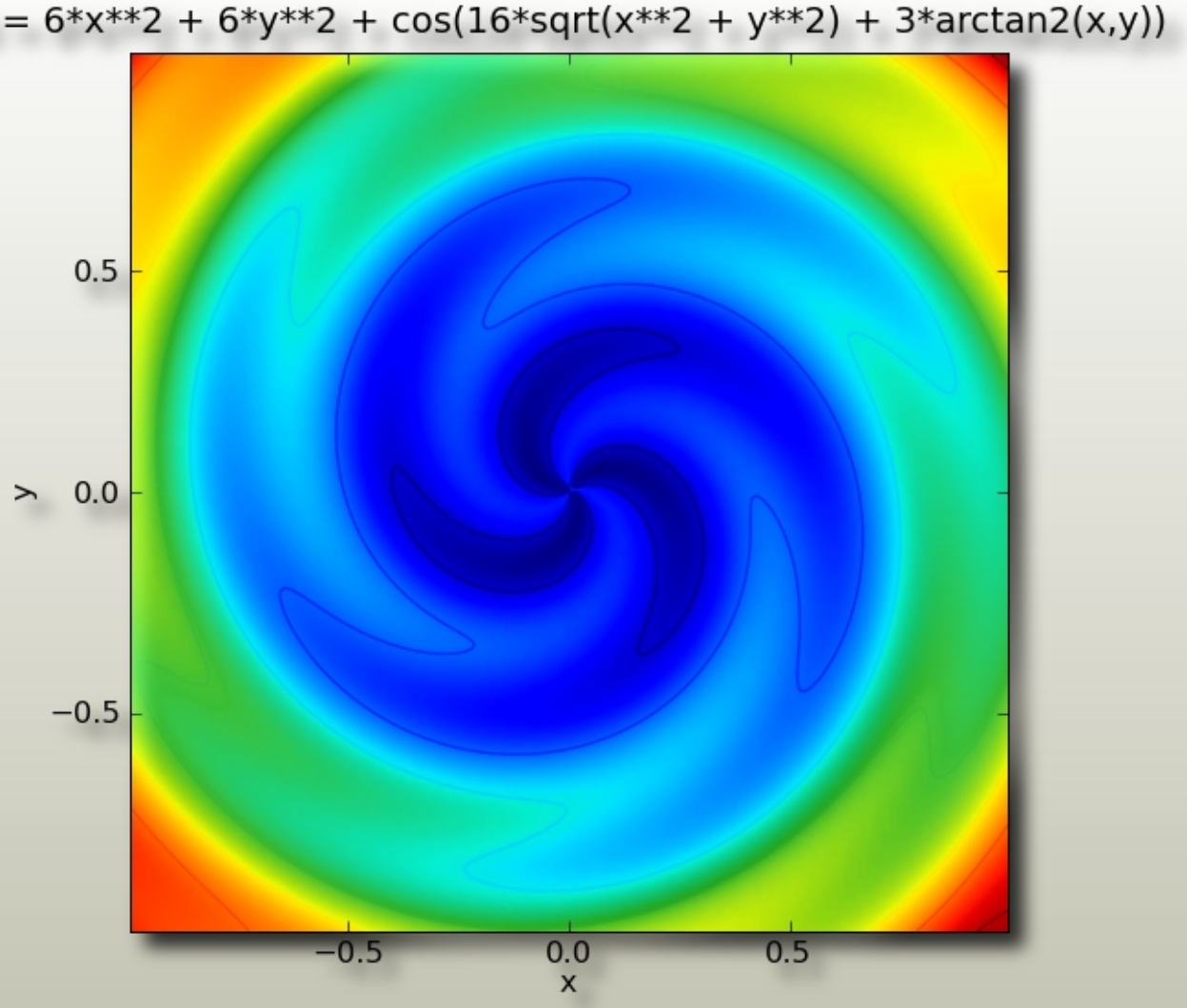
Uma função matemática.

A função *imshow()* plota gradientes de cores que variam com a saída função passada como parâmetro. Já *contour()* plota apenas os contornos das áreas de cor.

Salva o arquivo como PNG.

Pylab / Matplotlib (Exemplo II/II)

Saída



Pylab / Matplotlib / Basemap (Exemplo I/II)

```
from mpl_toolkits.basemap import Basemap
from matplotlib import pyplot
from numpy import arange

mapa = Basemap(projection='ortho', lat_0=-20, lon_0=-50,
   resolution='l', area_thresh=1e3)
mapa.bluemarble()
mapa.drawcoastlines(color='#777799')
mapa.drawcountries(color='#ccccee')
mapa.drawmapboundary()
mapa.drawmeridians(arange(0,360,30), color='#ccccee')
mapa.drawparallels(arange(-90,90,30), color='#ccccee')
x,y = mapa([-43.1],[-22.6])
mapa.plot(x, y,'w^')
pyplot.text(x[0]+2e5, y[0]-6e5, 'Rio\nde\nJaneiro',
   color='#eeeecc')
pyplot.savefig('mapa.png')
```

Carrega o *basemap*.

Cria um mapa visto da latitude -20 e longitude -50.

Define a aparência do mapa.

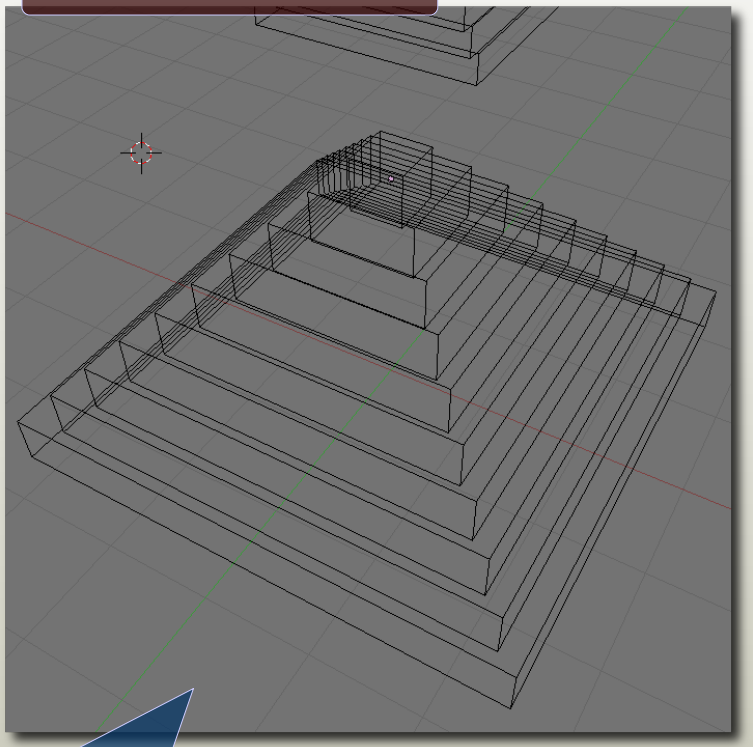
Marca um ponto no mapa.

Pylab / Matplotlib / Basemap (Exemplo II/II)

Saída

3D (I/III)

Wireframe



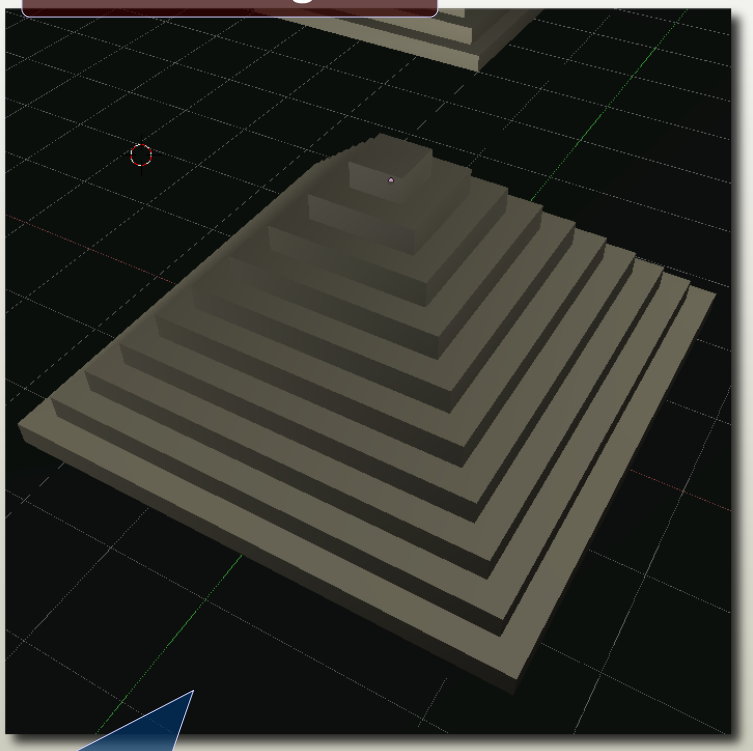
O *wireframe* representa a estrutura dos objetos que compõem a cena.

Materiais

Os materiais são aplicados aos objetos.

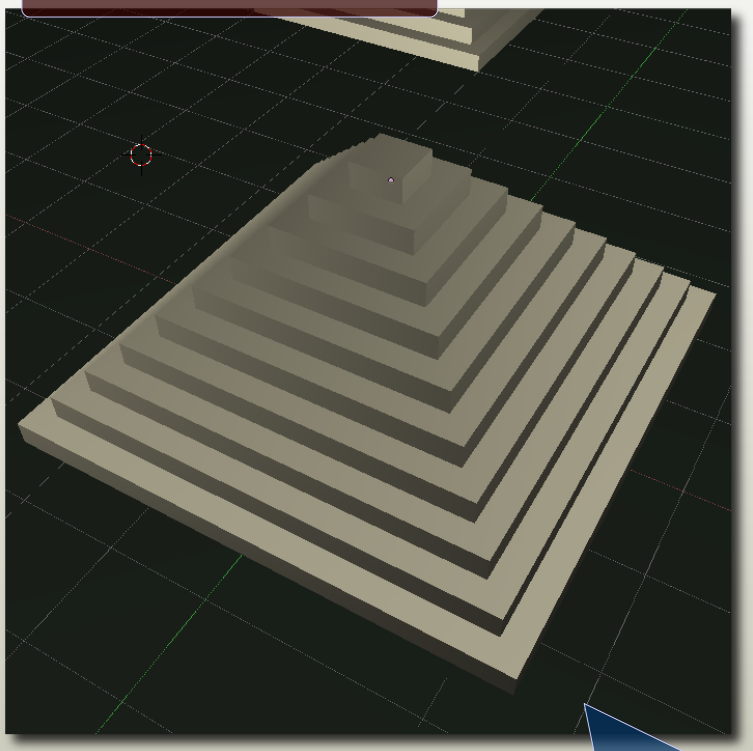
3D (II/III)

Shading



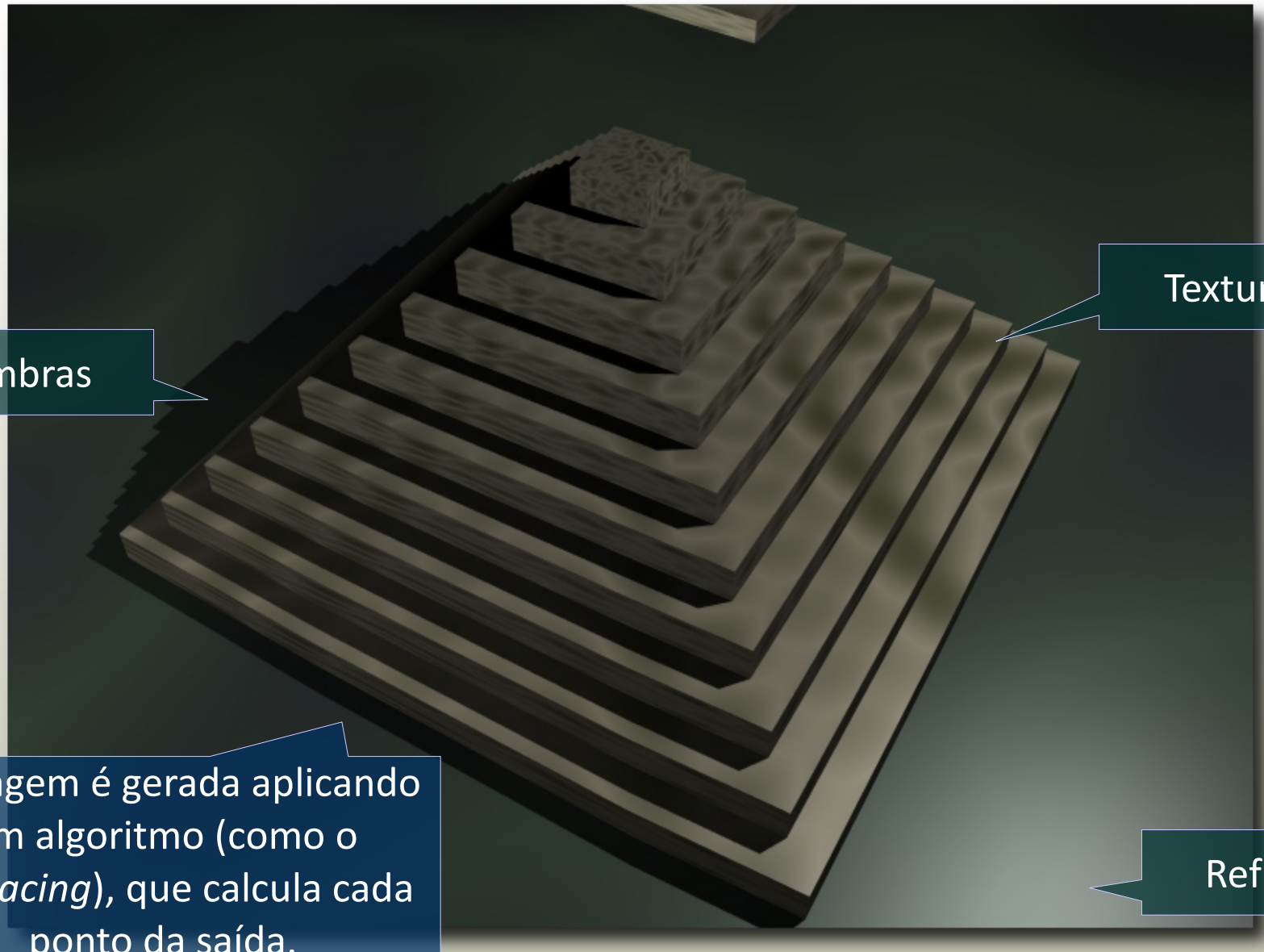
Sombreamento define como cada objeto da cena reage a luz.

Texturas



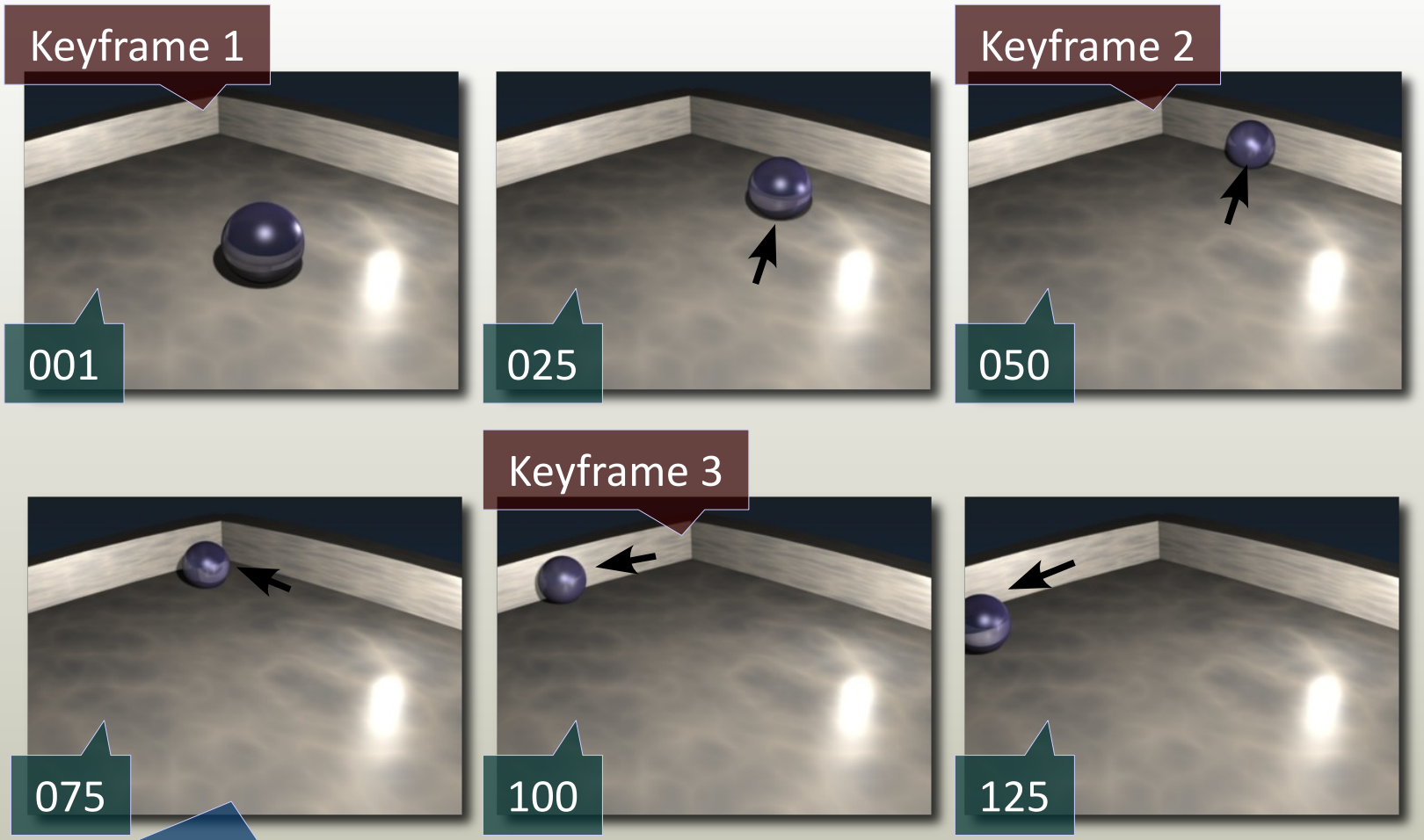
Texturas são imagens *raster* que são aplicadas aos materiais.

3D (III/III)



A imagem é gerada aplicando um algoritmo (como o *raytracing*), que calcula cada ponto da saída.

Animação 3D (Técnica *Keyframe*)



VPython

Implementa:

- Vários objetos 3D simples (cubo, esfera, cilindro, pirâmide, espiral e outros).
- Recursos para facilitar animações.
- Iluminação, *zoom* e *pan* automáticos.
- Controles interativos.

VPython (Exemplo I/II)

```
from visual import *

box(pos=(12, 0, 0) , length=1 , height=21 , width=29)
box(pos=(-12, 0, 0) , length=1 , height=21 , width=29)
box(pos=(0, 10, 0) , length=25 , height= 1 , width=29)
box(pos=(0, -10, 0) , length=25 , height=1 , width=29)

x, y, z = .1, .1, .1
b = sphere(pos=(3, 5, 7), radius=1.2)
s = cylinder(pos=(3, -10, 7), axis=(0, .5, 0), color=color.black)

while True:
    b.pos += (x, y, z)
    s.x, s.z = b.x, b.z
    if abs(b.x) > 10: x = -x
    if abs(b.y) > 8: y = -y
    if abs(b.z) > 12: z = -z
    Rate(100)
```

Desenha quatro caixas.

Desenha a esfera e sombra.

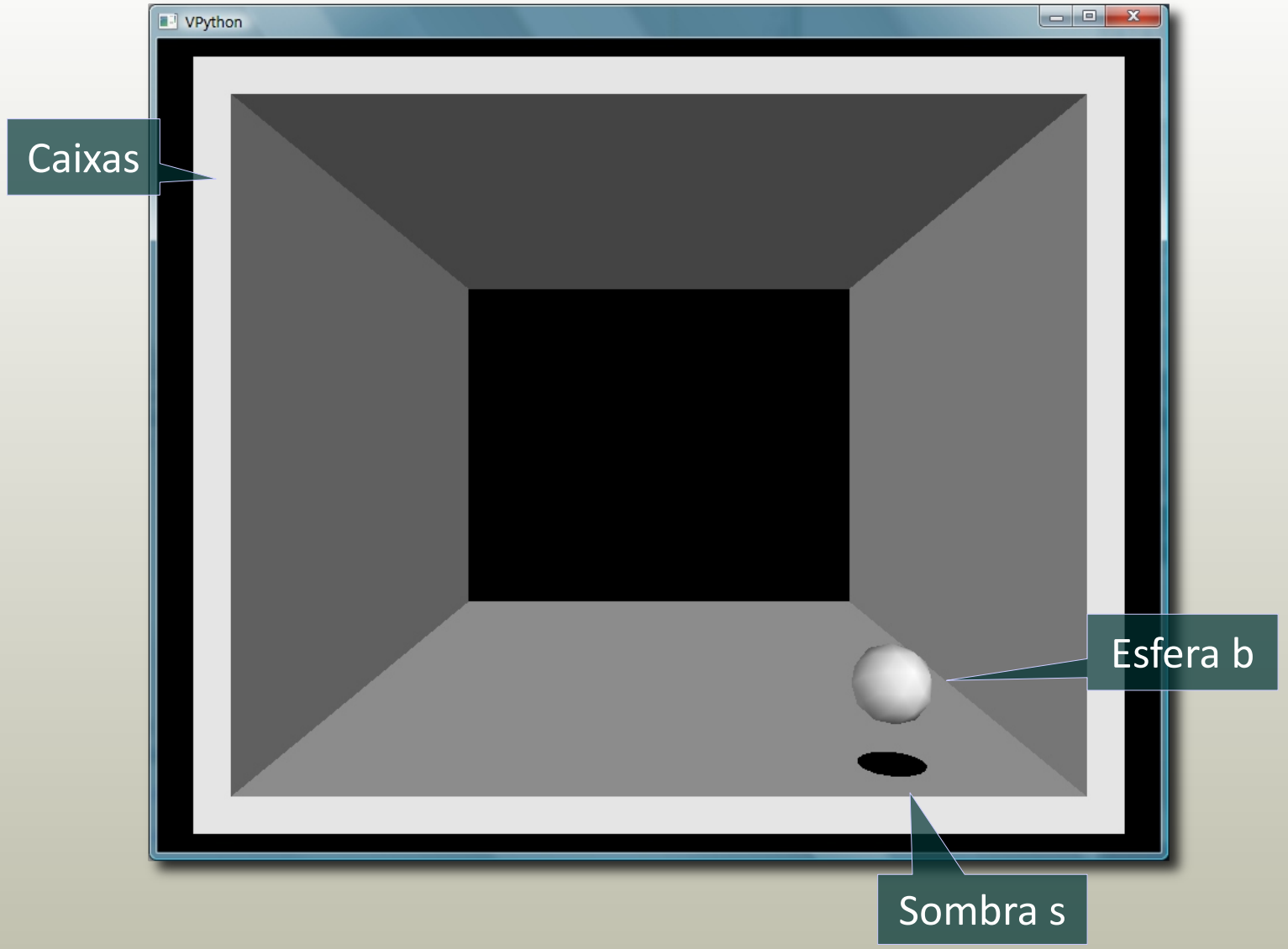
Move a esfera e a sombra.

Muda a direção da esfera e da sombra.

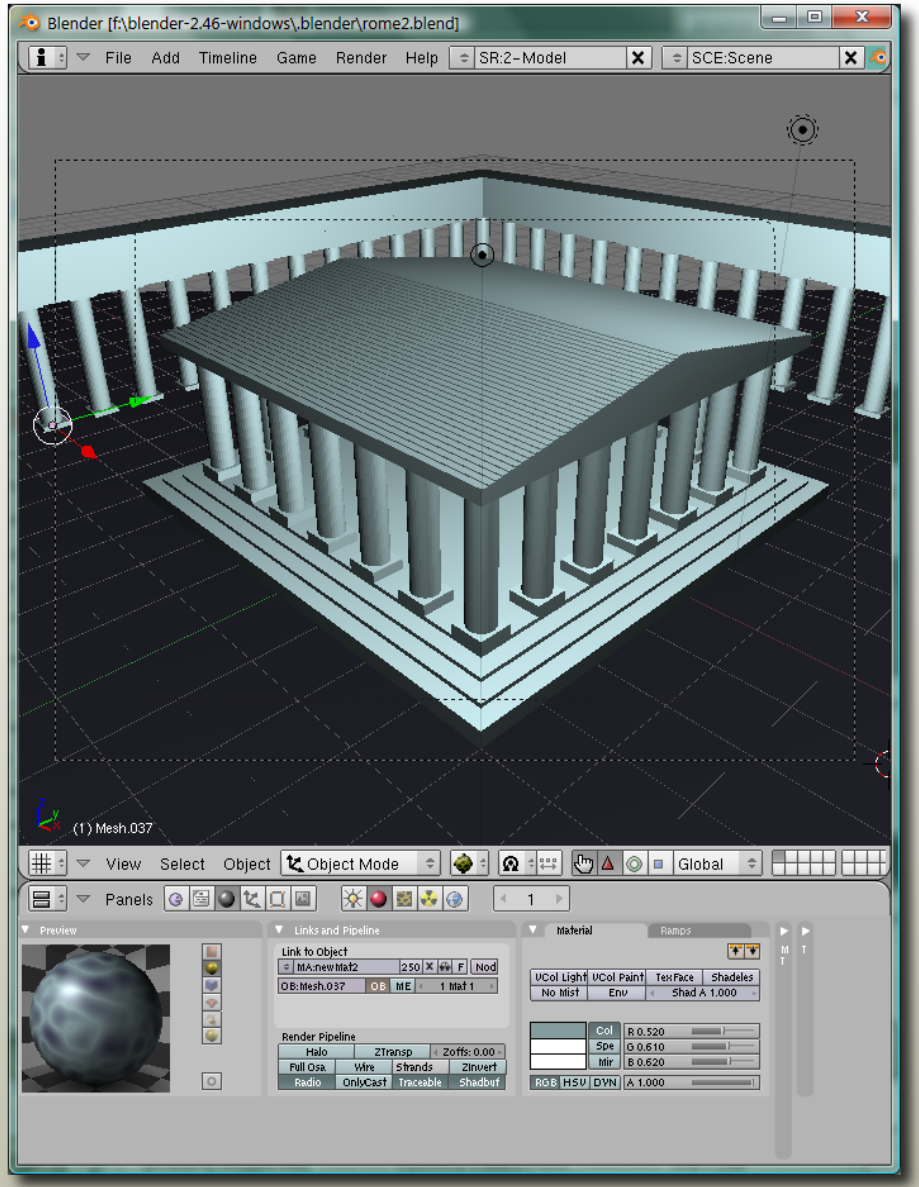
Controla a velocidade da animação.

VPython (Exemplo II/II)

Janela de saída



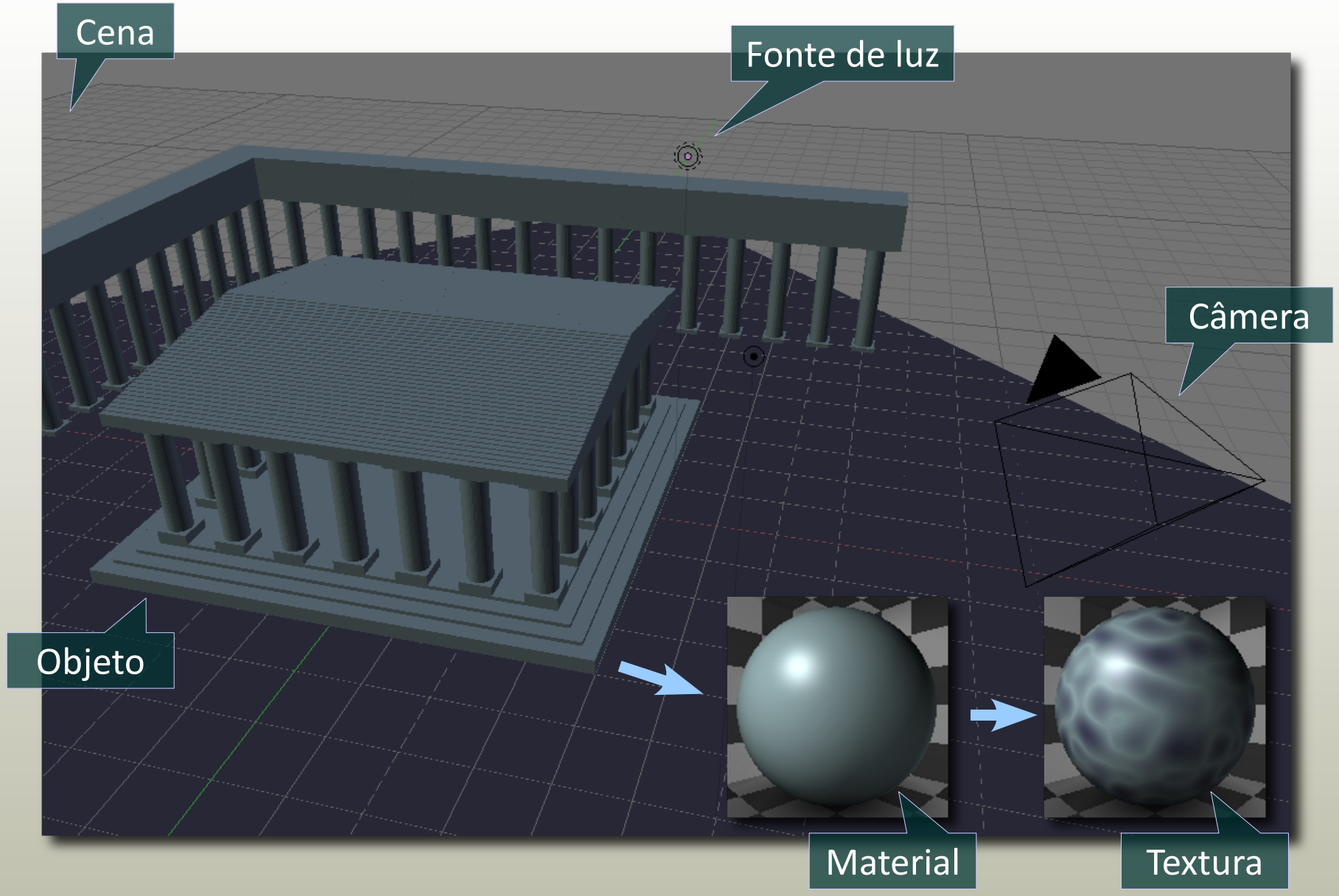
Blender



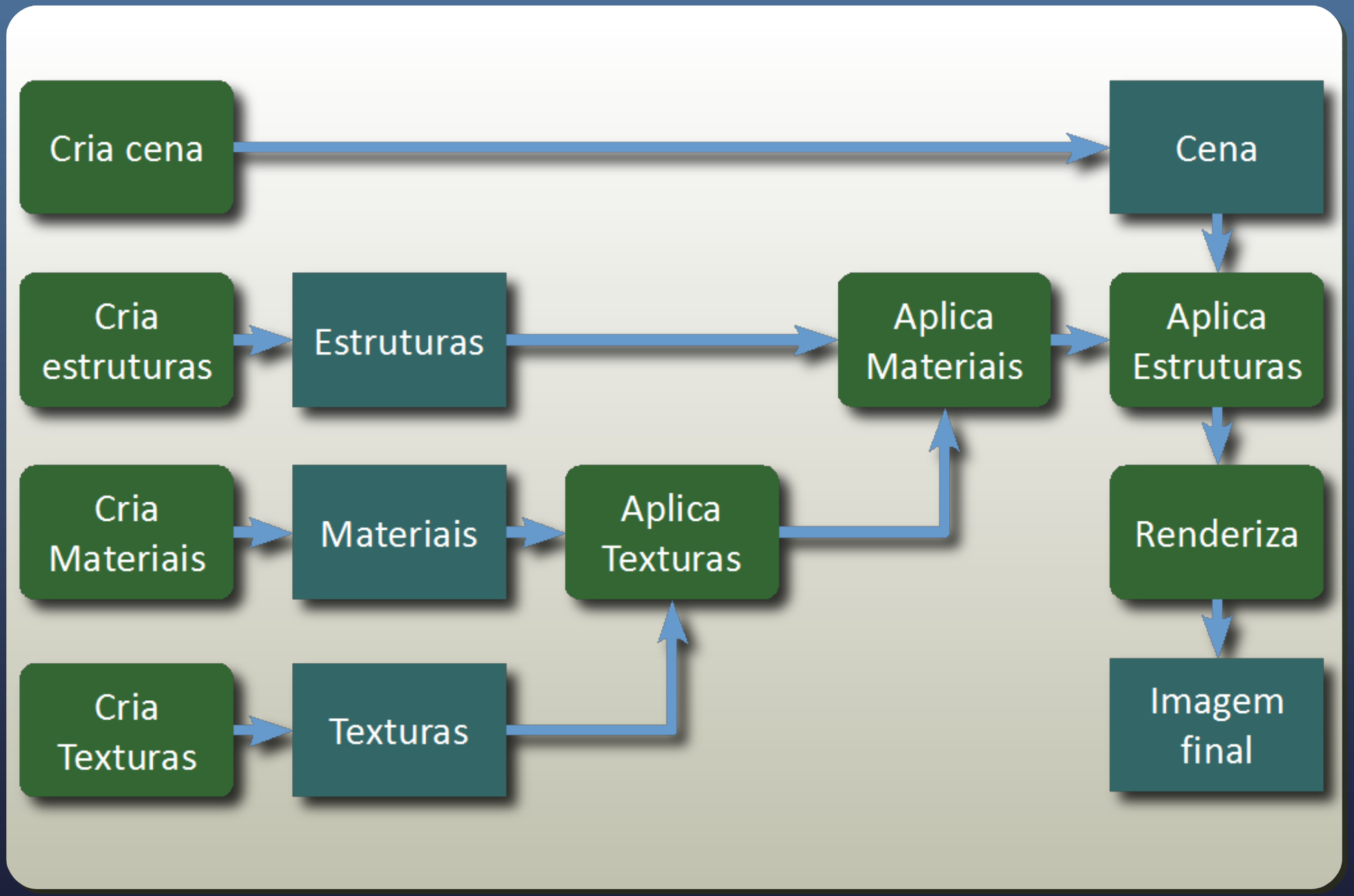
Implementa:

- Modelagem 3D avançada, com materiais, texturas, iluminação, extrusão, entre outros.
- Recursos de animação sofisticados.
- Um *game engine* poderoso.
- Uma API que permite uso do **Python**.

Blender (Visão Geral)

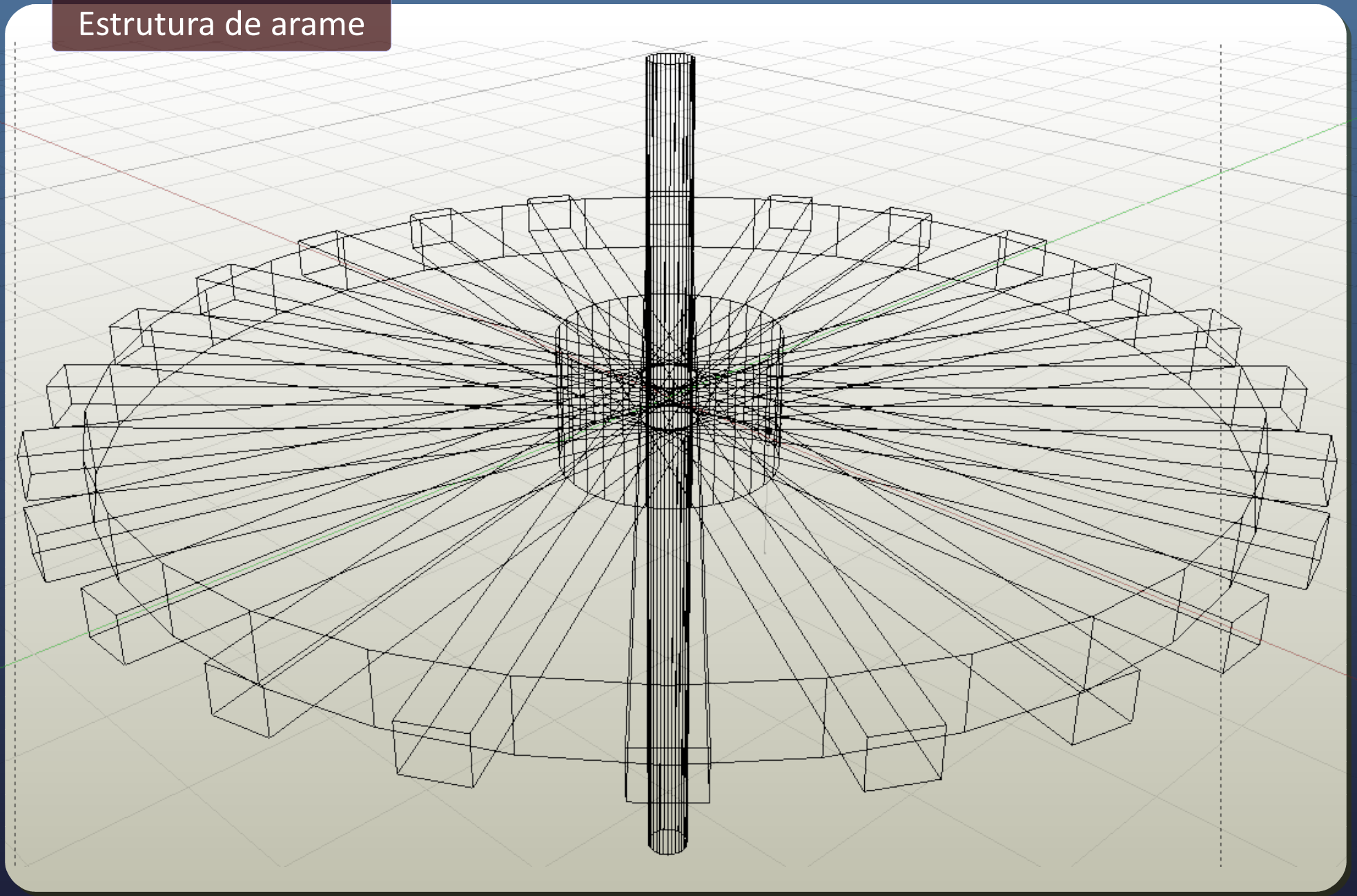


Blender (Construindo uma Cena)



Blender (Exemplo I/VI)

Estrutura de arame



Blender (Exemplo II/VI)

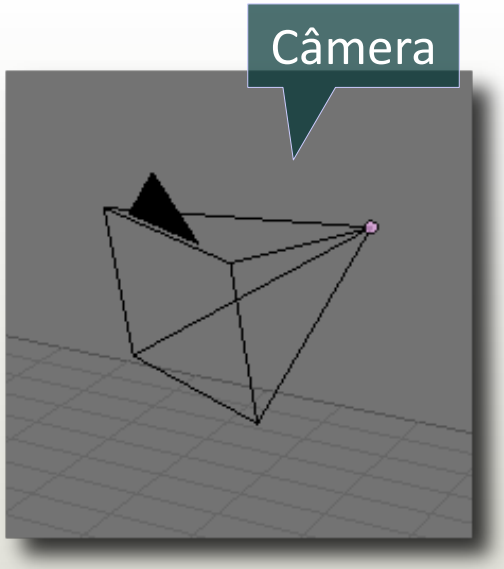
```
import math
import Blender

# A cena atual
cena = Blender.Scene.GetCurrent()

# Muda o lugar da "camera"
camera = Blender.Object.Get()[0]
camera.setLocation(9.5, -9.5, 6)
print dir(camera)

# Remove da cena o objeto "default"
cubo = Blender.Object.Get()[1]
cena.objects.unlink(cubo)

# Muda o lugar da fonte de luz
lamp = Blender.Object.Get()[-1]
lamp.setLocation(2.5, -1.5, 1)
```



Continua...

Blender (Exemplo III/VI)

```
def obj(mat, tam=(1, 1, 1), pos=None,
    prim=Blender.Mesh.Primitives.Cube):
    mesh = prim()
    obj = cena.objects.new(mesh, 'Mesh')
    obj.size = tam
    obj.colbits = 1
    obj.setMaterials(mat)
    if pos:
        obj.setLocation(*pos)
    return obj
```

Desenha um objeto na cena.

```
mater = Blender.Material.New('newMat1')
mater.rgbCol = (.14, .14, .14)
mater.setAlpha(1)
mater.setMode('RayMirr')
mater.setRayMirr(.3)
mater.setFresnelMirr(2)
mat = [mater]
```

Cria um novo material e define seus atributos.

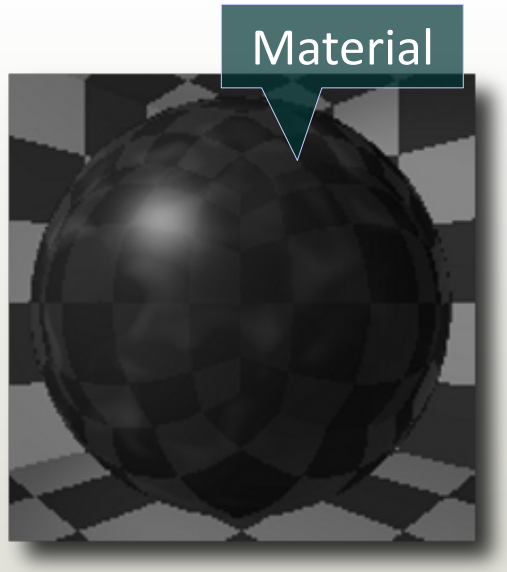
Continua...

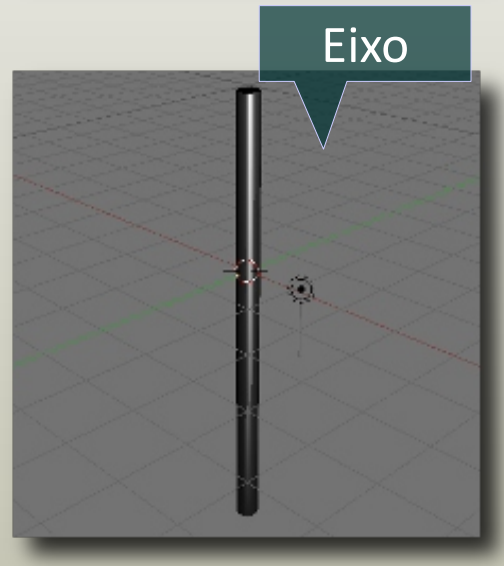
Blender (Exemplo IV/VI)

```
# Cria uma textura
textur = Blender.Texture.Get()[0]
textur.setType('DistortedNoise')

# Coloca no material
mater.setTexture(0, textur)
mtex = mater.getTextures()[0]
mtex.col = (.28, .28, .28)

# Cria eixo
prim = Blender.Mesh.Primitives.Cylinder
obj(mat, (.2, .2, 4), pos=(0, 0, -1), prim=prim)
```





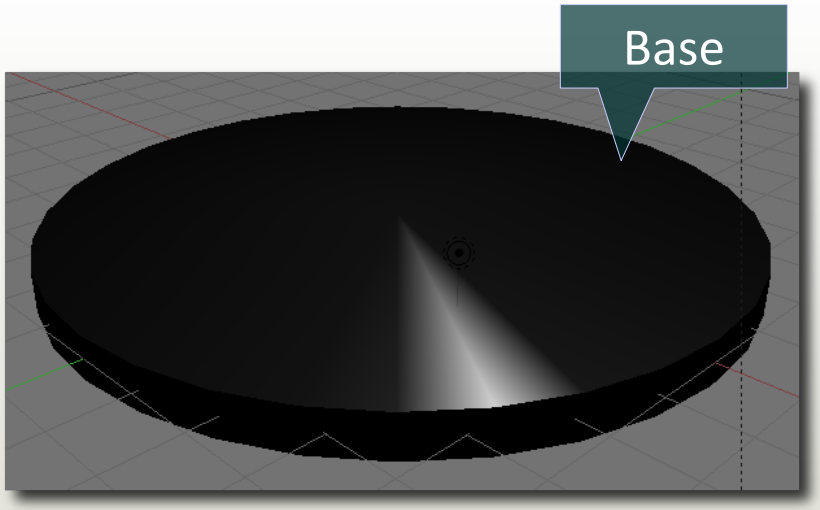
Continua...

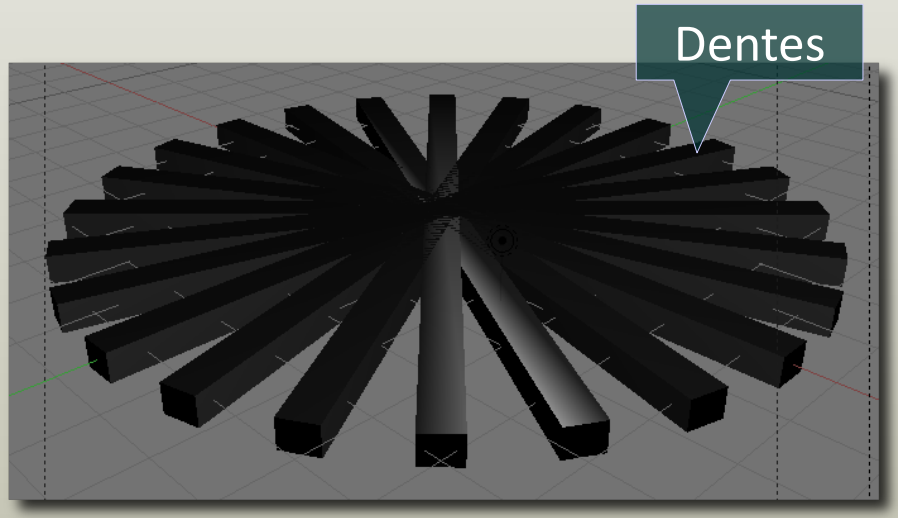
Blender (Exemplo V/VI)

```
# Cria base
pos = (0, 0, 0)
obj(mat, (1, 1, .6), pos=pos, prim=prim)
base = obj(mat, (5, 5, .3), pos=pos,
    prim=prim)

# Cria dentes
for ang in xrange(0, 180, 15):
    dentes = obj(mat, (5.5, .25, .2),
        pos=pos)
    dentes.setEuler(0, 0,
        math.radians(ang))

# Atualiza a cena
cena.update()
```

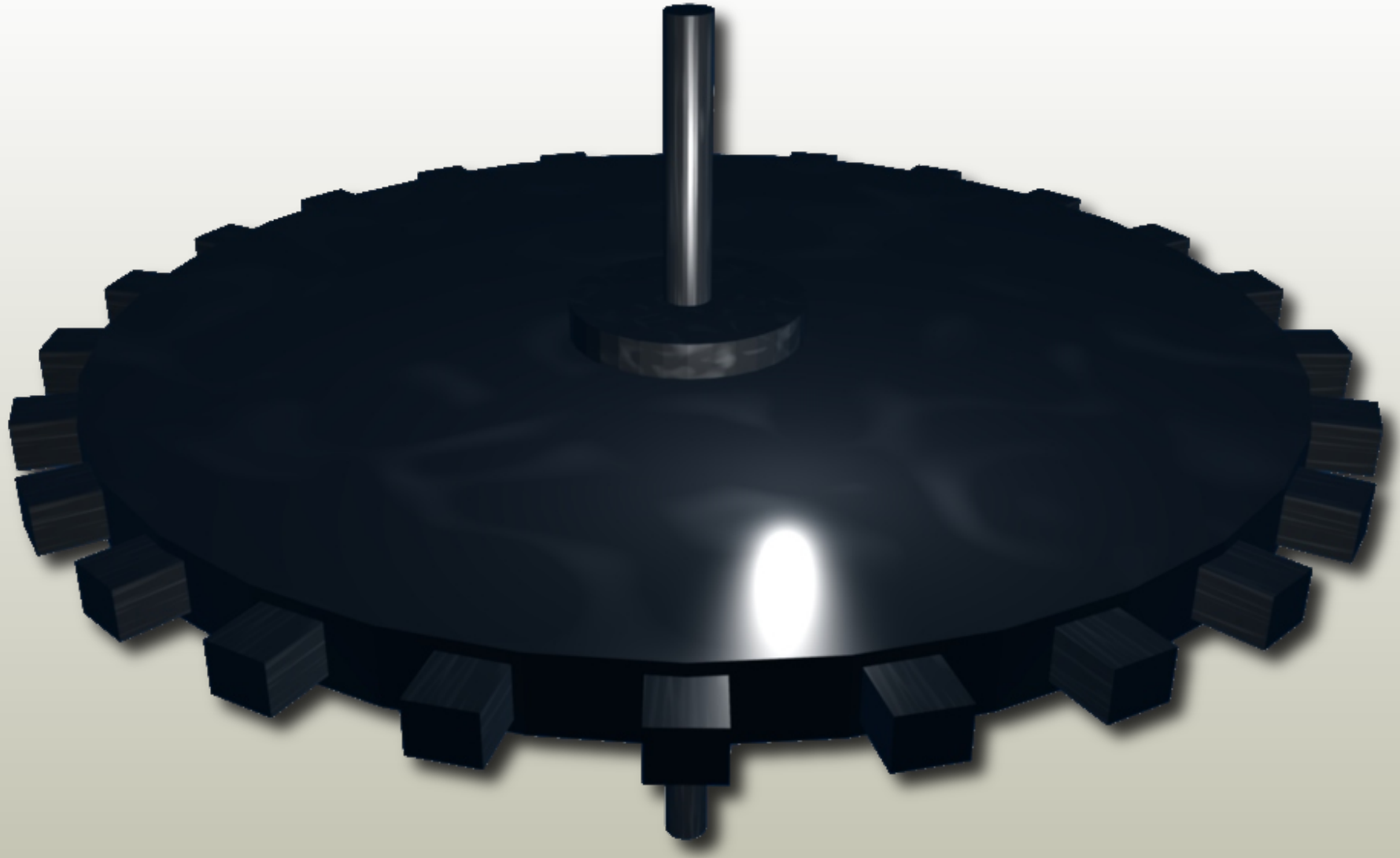




Continua...

Blender (Exemplo VI/VI)

Objeto renderizado



Referências

PIL:

- http://www.pythonware.com/products/pil/

GIMP:

- http://www.gimp.org/

Matplotlib:

- http://matplotlib.sourceforge.net/

VPython:

- http://vpython.org/

Blender:

- http://www.blender.org/

Duvidas?

Blog: http://ark4n.wordpress.com/