

Programação II

Gráficos 2D (pygame)

Hugo Pacheco

DCC/FCUP

21/22

Gráficos

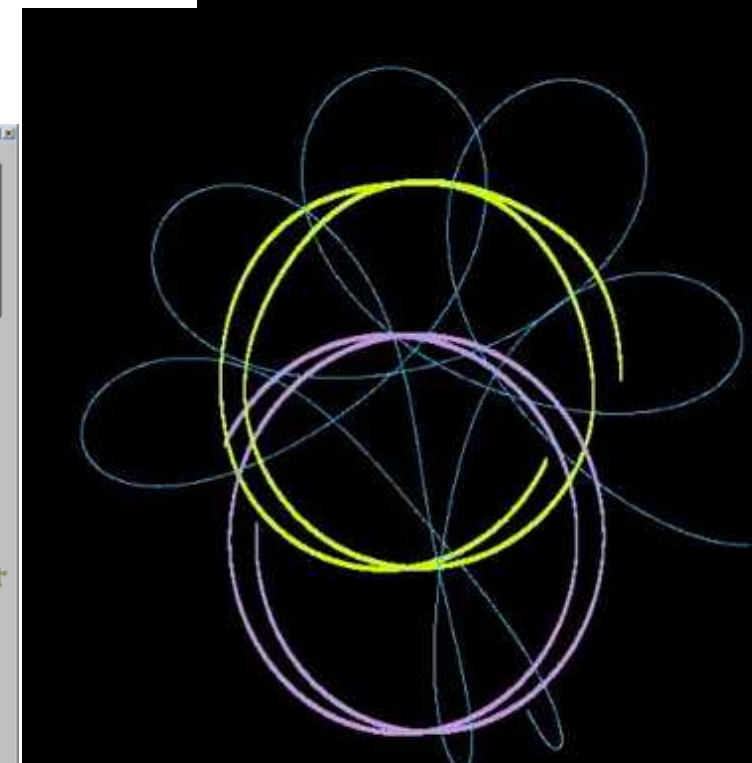
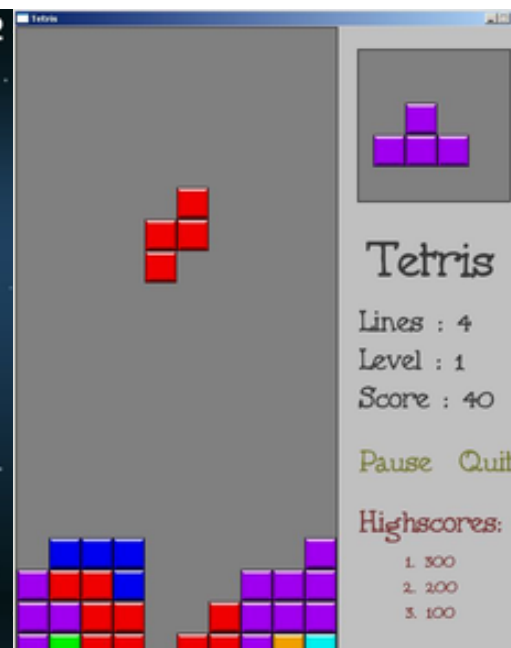
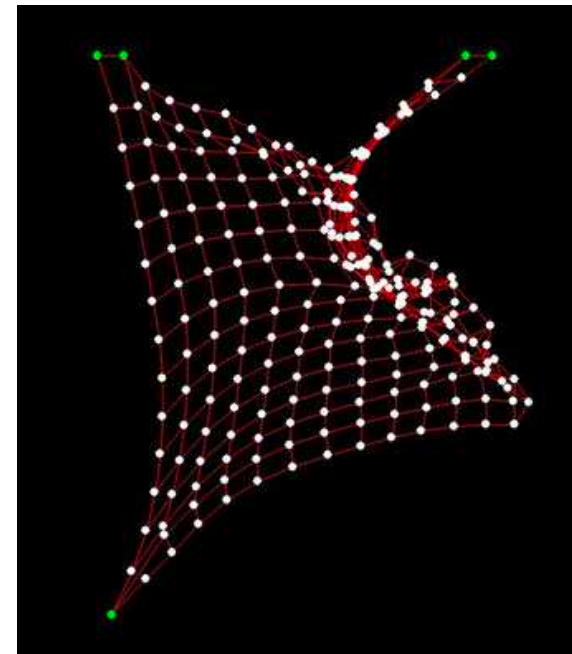
- Até agora:
 - Processamento e análise de dados
 - Visualizações na forma de gráficos e mapas
 - Gráficos interativos simples
- Próximas aulas (*PyGame*):
 - Gráficos 2D mais sofisticados
 - Animações e aplicações interativas mais flexíveis



- Uma biblioteca para:
 - Desenhar gráficos 2D sofisticados (e.g. geometria)
 - Construir animações, i.e., gráficos ao longo do tempo
 - Lidar com multimédia (imagens, sons, etc)
 - Construir jogos e aplicações interativas
 - Lidar com eventos de input (rato, teclado, joystick, etc)
- Nota: mais flexível \Rightarrow mais baixo nível

A imaginação é o limite

- Muita documentação e galeria de exemplos em pygame.org
- Bibliografia complementar:
 - Albert Sweigart; [Making Games with Python & Pygame](#)
 - Paul Craven; [Program Arcade Games: With Python and Pygame](#)



Anatomia de um jogo

- Template para desenhar um gráfico 2D
- Conceitos: janela, desenho, eventos

```
import pygame

# inicializa o jogo e cria a janela
pygame.init()
screen = pygame.display.set_mode()

# desenha objetos
pygame.draw.* (screen, ...)

# atualiza janela
pygame.display.update()

# responde a eventos (fechar)
done = False
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
```



Similar ao
matplotlib

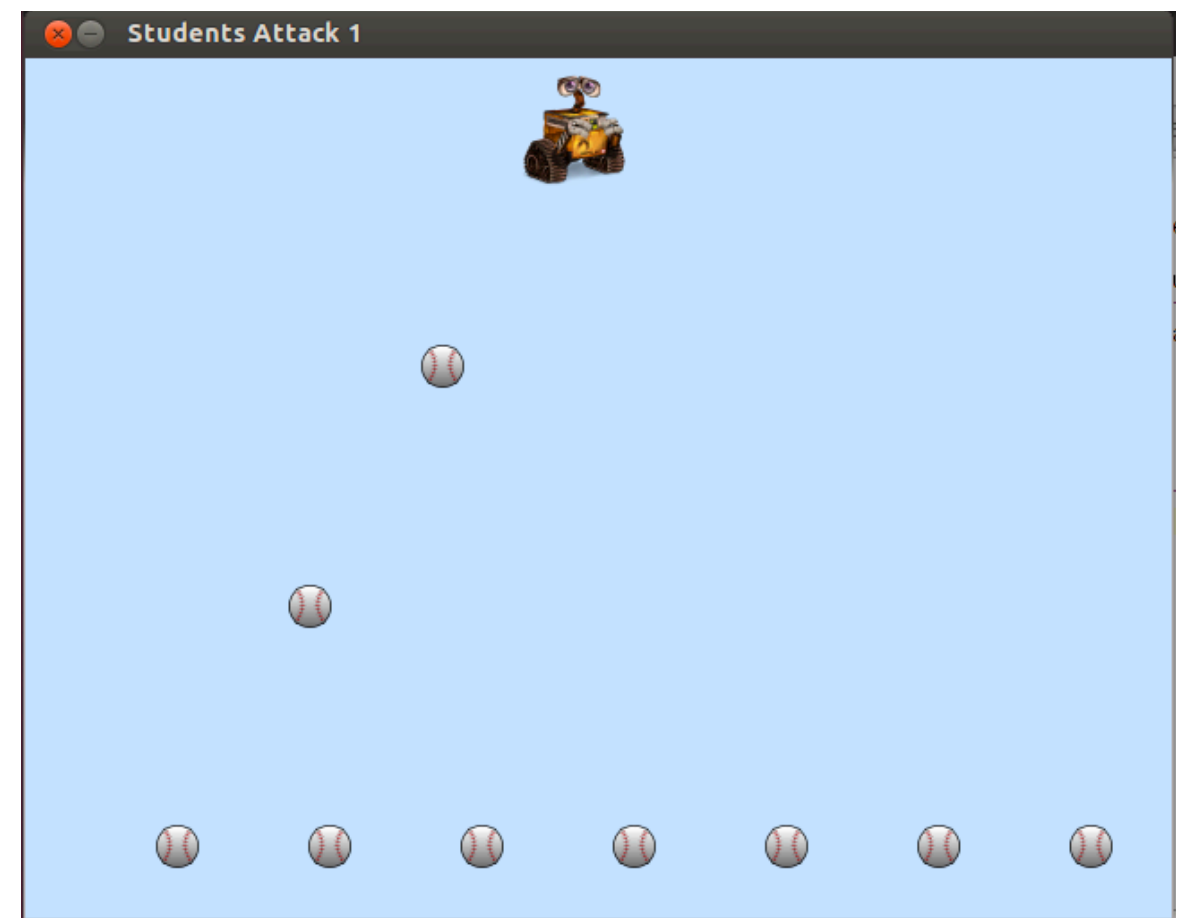
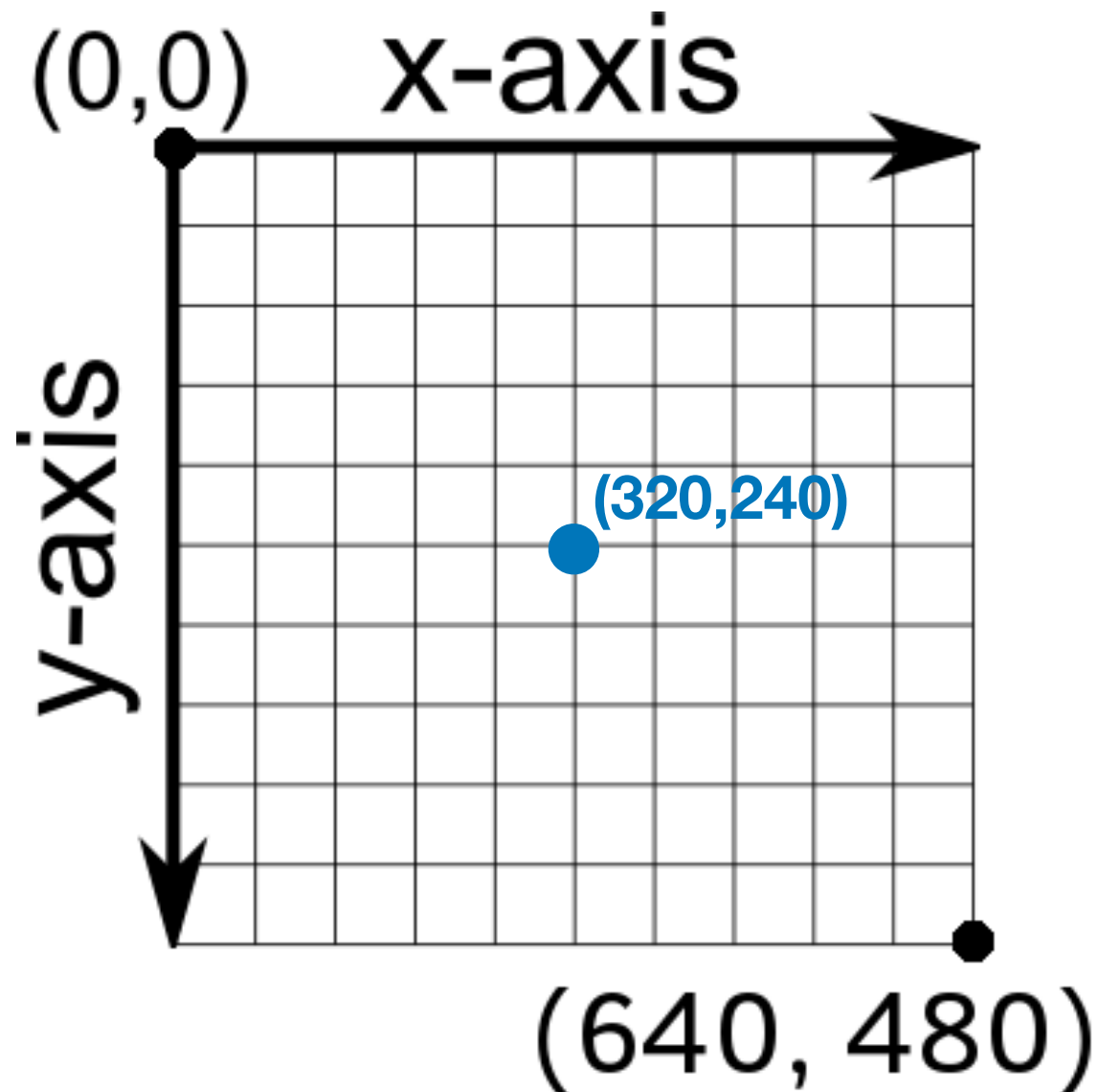
Janela

- O primeiro elemento de um jogo é a janela, criada no início
- Cores = tuplos (r,g,b) entre 0 e 255

```
# cria uma janela com o tamanho do ecrã
screen = pygame.display.set_mode()
# cria uma janela com um tamanho fixo
screen = pygame.display.set_mode((800, 600))

# uma cor RGB
blue=(0, 0, 255)
# altera a cor de fundo da janela
screen.fill(blue)
# altera o nome da janela
pygame.display.set_caption("window title")
# imprime tamanho da janela
print(screen.get_size())
```

Janela (coordenadas)



Superfícies

- Uma folha de papel em branco
- Janelas e outros objetos 2D são superfícies

```
# uma janela é uma superfície  
screen = pygame.display.set_mode((width,height))
```

```
# cria uma nova superfície  
surface = Surface((width,height))
```

```
# pinta uma superfície  
surface.fill((r,g,b))
```

```
# dimensões de uma superfície  
surface.get_size()  
surface.get_width()  
surface.get_height()
```

```
# desenha uma superfície numa dada posição de outra  
surface.blit(surface2,coords)
```

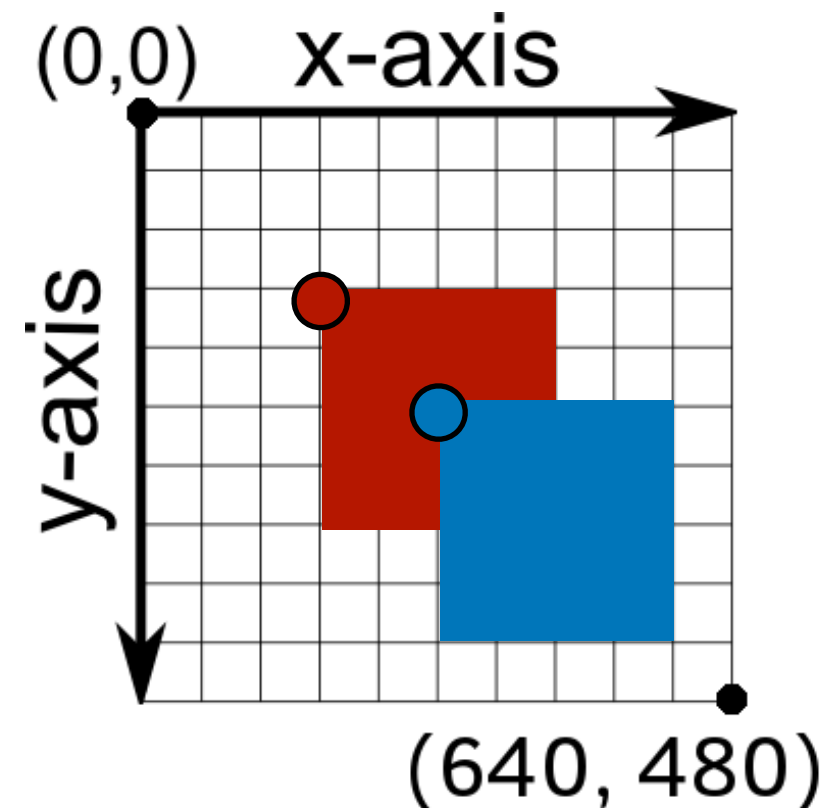

Imagens

- Imagens são superfícies

```
# carrega uma imagem de ficheiro
image = pygame.image.load('image.png')

# altera o ícone da janela
pygame.display.set_icon(image)

# desenha a imagem no centro da janela
pos = (screen.get_width()/2-
image.get_width()/2, screen.get_height()/
2-image.get_height()/2)
screen.blit(image, pos)
```



Texto

- Podemos criar superfícies com texto

```
red = (255,0,0); green = (0,255,0); blue = (0,0,255)

# durante a inicialização do programa
pygame.font.init()

# criar uma fonte
comicsans = pygame.font.SysFont('Comic Sans MS',30)

# criar superfície textual
text = comicsans.render('Hello pygame',False,red)

# desenha texto na janela
screen.blit(text,(screen.get_width()/2,screen.get_height()/2))
```

Desenhos

- Podemos desenhar numa superfície, e.g., formas geométricas. Por defeito, objeto preenchido (*width=0*)

```
red = (255, 0, 0); green = (0, 255, 0); blue = (0, 0, 255)
```

```
# desenha um círculo
```

```
# circle(Surface,color,pos,radius,width=n)
```

```
pygame.draw.circle(screen, red, (200, 200), 50)
```

```
pygame.draw.circle(screen, red, (200, 300), 50, width=1)
```

```
# desenha um polígono
```

```
# polygon(Surface,color,pointlist,width=n)
```

```
pygame.draw.polygon(screen, green, [(400, 400), (400, 300), (300, 400)])
```

```
# desenha uma linha
```

```
# line(Surface,color,start,end,width=n)
```

```
pygame.draw.line(screen, blue, (10, 50), (30, 100), width=10)
```

```
# copia a janela para o ícone
```

```
pygame.display.set_icon(screen)
```

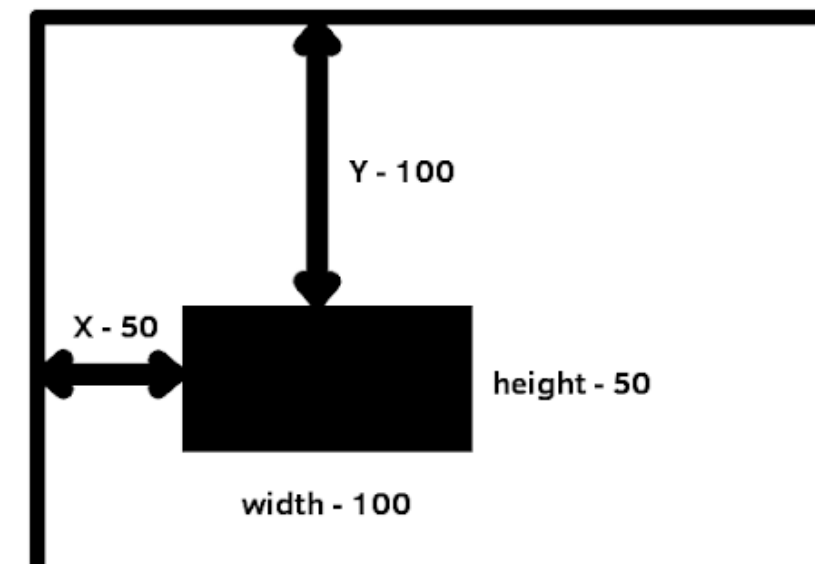
Retângulos

- Podemos criar e desenhar retângulos
- Retângulos como “bounding boxes” relativas à janela

```
# cria um retângulo
# Rect(left,top,width,height,width=n)
box = pygame.Rect(50,100,100,50)
# desenha um retângulo
# rect(Surface,color,Rect,width=n)
pygame.draw.rect(screen,blue,box)
# desenha uma ellipse dentro de um retângulo
# ellipse(Surface,color,Rect,width=n)
pygame.draw.ellipse(screen,green,box,width=3)
# desenha um arco
# arc(Surface,color,Rect,start,stop,width=n)

# bounding box de um círculo
bbox = pygame.draw.circle(screen,red,(200,200),50)
print(bbox)
# bounding box de uma superfície
print(screen.get_rect())
```

RECT = (50, 100, 100, 50)



Transformações

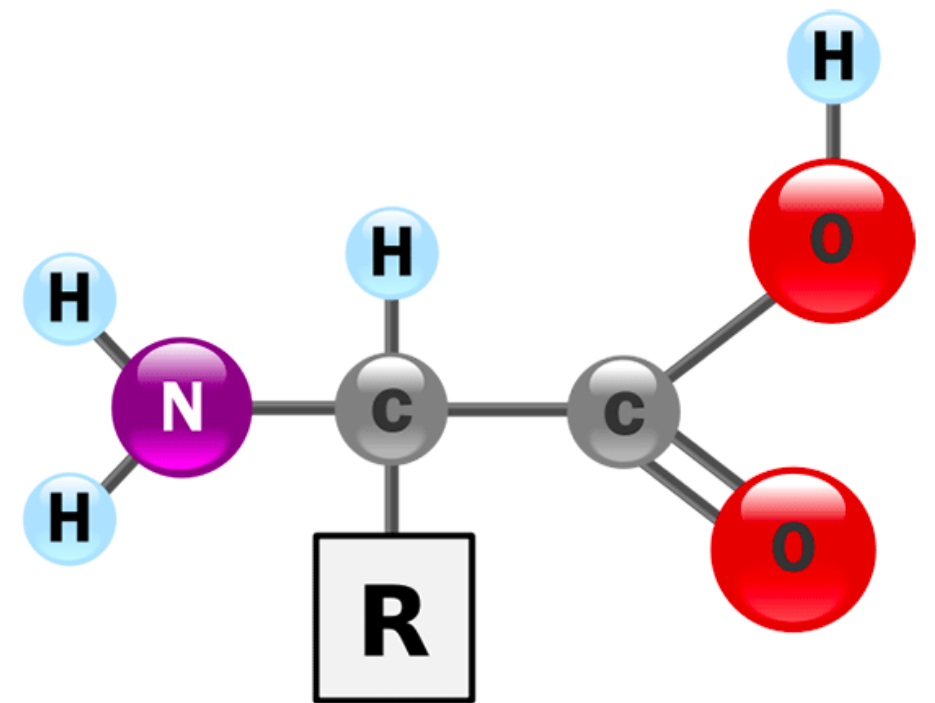
- Podemos aplicar transformações a uma superfície

```
# pygame.transform.*  
  
# inverte horizontalmente e/ou verticalmente  
newsurface = flip(surface, xbool, ybool)  
# redimensiona  
newsurface = scale(surface, (xscale, yscale))  
# redimensiona para o dobro do tamanho  
newsurface = scale2x(surface)  
# redimensiona com anti-aliasing  
newsurface = smoothscale(surface, (xscale, yscale))  
# roda com um ângulo em radianos  
newsurface = rotate(surface, angle)
```

Exemplo (Molécula)

- Moléculas químicas podem ser representadas por formas geométricas
- Formatos standard (e.g., *MOL2*) descrevem a posição 2D/3D e propriedades de cada átomo, bem como as suas ligações
- E.g., uma representação simplificada de uma molécula de um aminoácido

```
atoms = [(190,190,'H'), (190,325,'H'),
(260,260,'N'), (390,260,'C'), (390,160,'H'),
(390,390,'R'), (530,260,'C'), (630,350,'O'),
(630,160,'O'), (660,40,'H')]
ligands = {(0,2,'single'), (1,2,'single'),
(2,3,'single'), (3,4,'single'), (3,5,'single'),
(3,6,'single'), (6,7,'double'), (6,8,'single'),
(8,9,'single')}
weights = {'H':('circle',25), 'C':
('circle',35), 'N':('circle',45), 'O':
('circle',55), 'R':('square',55)}
colors =
{'H':lightblue, 'N':purple, 'C':darkgrey, 'O':red
, 'R':lightgrey}
```

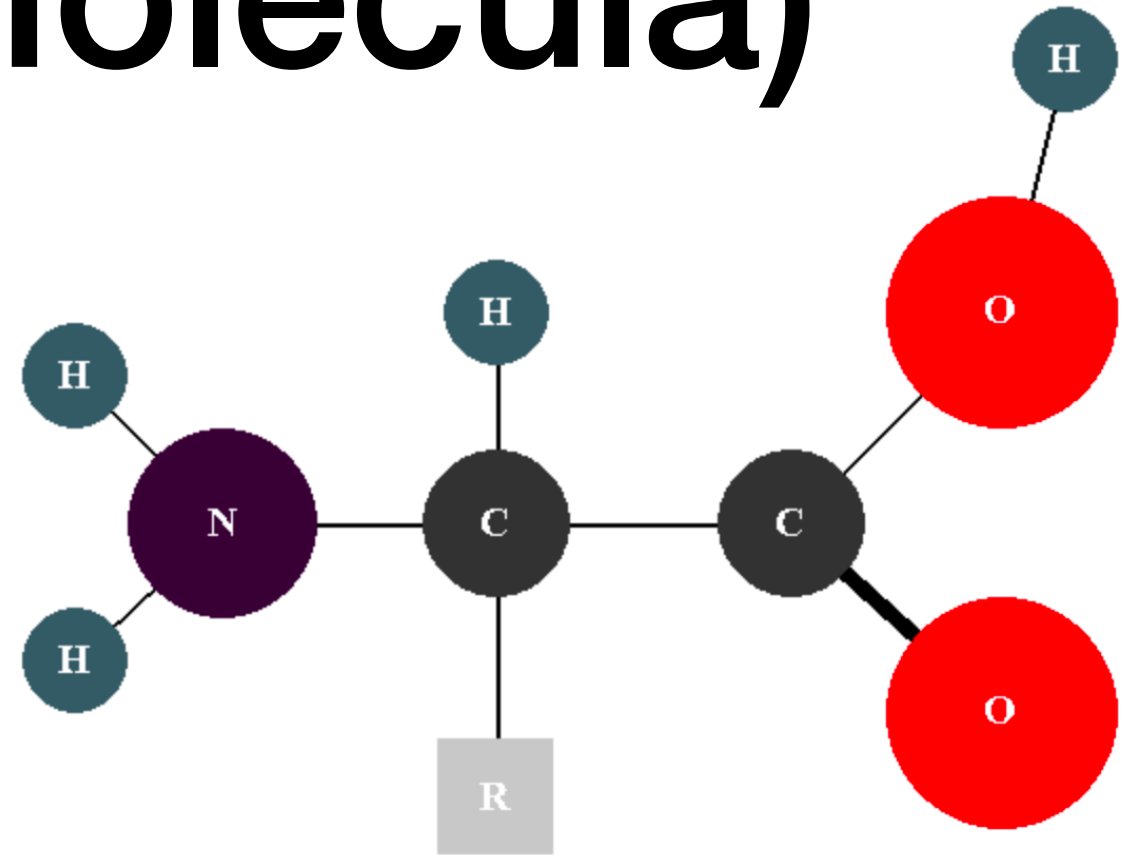


Exemplo (Molécula)

- Desenhar a molécula

```
for i,j,tp in ligands:
    # draw ligands
    xi,yi,_ = atoms[i]
    xj,yj,_ = atoms[j]
    if tp=='single': pygame.draw.line(screen,black,(xi,yi),(xj,yj),width=2)
    else: pygame.draw.line(screen,black,(xi,yi),(xj,yj),width=10)

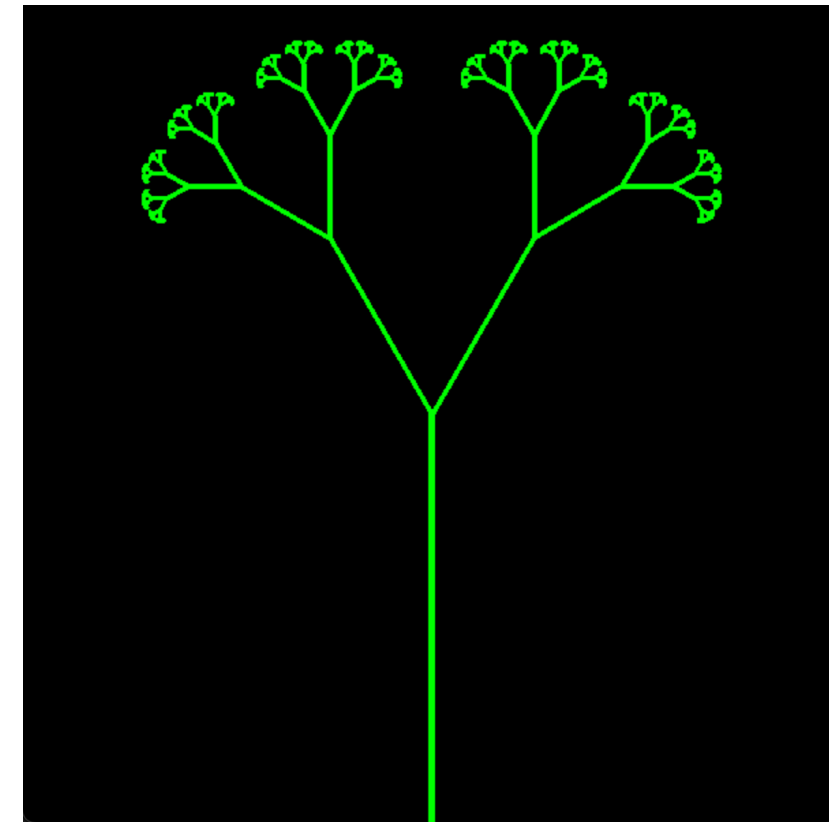
for x,y,name in atoms:
    # draw shapes
    tp,sz = weights[name]
    color = colors[name]
    if tp=='circle': pygame.draw.circle(screen,color,(x,y),sz)
    else: pygame.draw.rect(screen,color,pygame.Rect(x-sz/2,y-sz/2,sz,sz))
    # draw names
    text = font.render(name,False,white)
    screen.blit(text,(x-text.get_width()/2,y-text.get_height()/2))
```



Exemplo (Fractal)

- Podemos visualizar progressões geométricas
- Fractais = progressões geométricas com padrões repetitivos de tamanho decrescente
 - São muitas vezes utilizados para descrever padrões naturais (plantas, relâmpagos, etc)
- E.g., um fractal de uma árvore (utilizando recursividade)

```
def tree(surface, n, pos, angle, width):  
    if n > 0:  
        x, y = pos  
        x1 = x + width * math.cos(angle)  
        y1 = y - width * math.sin(angle)  
        pos1 = x1, y1  
        pygame.draw.line(surface, green, pos, pos1, width=n//4)  
        tree(surface, n-1, pos1, angle+math.pi/6, width/2)  
        tree(surface, n-1, pos1, angle-math.pi/6, width/2)  
  
start = (screen.get_width()/2, screen.get_height())  
tree(screen, 20, start, math.pi/2, 300)
```



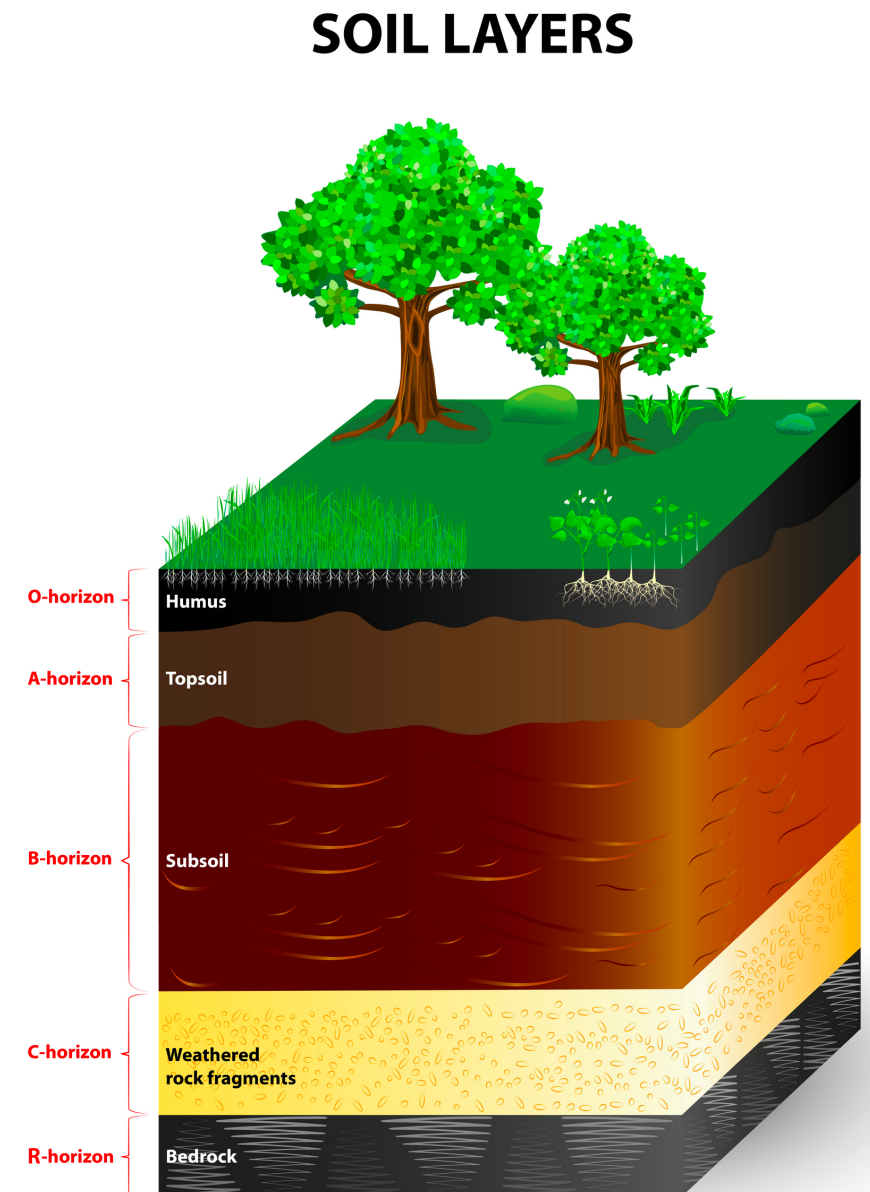
Exemplo (Solo)

- Podemos desenhar projeções 2D de cenas 3D simples (muita matemática)
- E.g., camadas de solo terrestre

```
angle = math.pi/6; width = 100; height = 50
w = width * math.cos(angle); h = height *
math.sin(angle)
```

```
layers = [ ('O', 5, (0, 0, 0)), ('A', 25, (78, 44, 20)),
            ('B', 76, (90, 0, 0)), ('C', 122, (255, 230, 115)),
            ('R', 60, (50, 50, 50)) ]
green=(0, 135, 46)
```

```
offsets = sum([sz for x,sz,c in layers])
terrain = pygame.Surface((width*2,width+offsets))
```



Exemplo (Solo)

- Desenhar e ajustar à janela

```
offset = width
for n,t,c in layers:
    ps = [(0,offset), (width,offset), (width+w,offset-h),
          (width+w,offset-h+t), (width,offset+t),
          (0,offset+t)]
    pygame.draw.polygon(terrain,c,ps)
    darker_c = tuple([max(0,rgb-50) for rgb in c])
    pygame.draw.line(terrain,darker_c,
                     (width,offset+t), (width,offset),width=1)
    offset +=t
ps = [(0,width), (width,width), (width+w,width-h),
      (0+w,width-h)]
pygame.draw.polygon(terrain,green,ps)

factor = terrain.get_width()/terrain.get_height()
terrain = pygame.transform.scale(terrain,
                                  (int(screen.get_height()*factor),screen.get_height()))
screen.blit(terrain, ((screen.get_width()-
terrain.get_width())/2,0))
```

