Instituto Superior Técnico

Programação e Computação para Arquitectura – 2016/2017

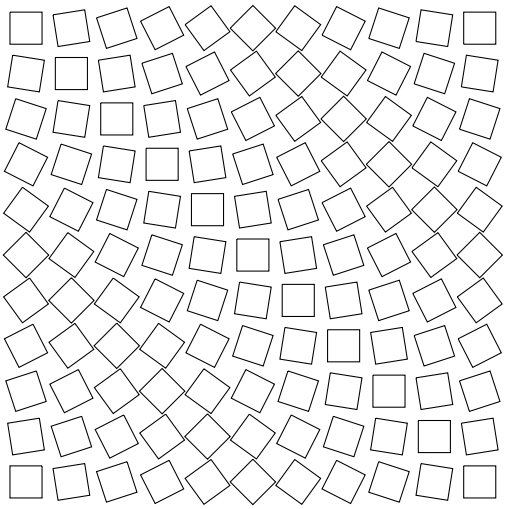
Segundo Teste – 18/01/2017

Número: _____

Nome: _____

Escreva o seu número em todas as folhas da prova. O tamanho das respostas deve ser limitado ao espaço fornecido para cada pergunta. Se tiver dúvidas de interpretação, faça suposições razoáveis e explicite-as na sua resposta. Pode usar os versos das folhas para rascunho. A prova tem **5** páginas e a duração é de **60 minutos**. A cotação de cada questão encontra-se indicada entre parêntesis. Boa sorte.

1. (1.0) Considere a seguinte ilustração, composta por uma distribuição regular de quadrados que vão sofrendo uma rotação proporcional à linha e coluna em que se encontram.

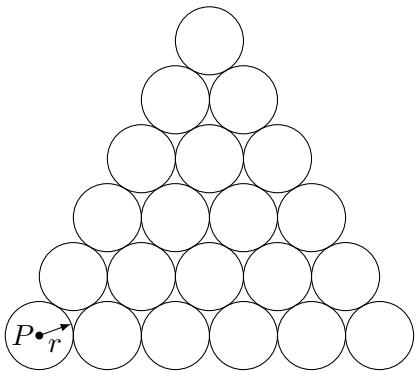


Defina a função `quadrados_rodados` que deverá receber, como parâmetros, o centro do quadrado no canto inferior esquerdo, o raio do círculo circunscrito ao quadrado (i.e., metade da diagonal do quadrado), o ângulo de rotação entre o primeiro e o último quadrado de uma fila horizontal ou vertical de quadrados e, finalmente, o número de quadrados, que será usado tanto na horizontal como na vertical.
Sugestão 1: utilize a função `regular_polygon` que cria um polígono regular a partir do número de lados, do centro, do raio, do ângulo de rotação de um vértice e, finalmente, de um boleano que indica se o raio é de um círculo inscrito ao polígono (ou seja, o raio é a distância dos lados ao centro) ou de um círculo circunscrito ao polígono (ou seja, o raio é a distância dos vértices ao centro).
Sugestão 2: utilize a função `map_division`.

```
def quadrados_rodado(p, r, a, n):
    da = a/n
    map_division(lambda i,j:
                    regular_polygon(4,
                                    p + vxy(2*r*i,2*r*j),
                                    r,
                                    pi/4 + da*(i+j),
                                    True),
                 0,n,n,
                 0,n,n)

quadrados_rodado(u0(), 1,pi/4, 10)
```

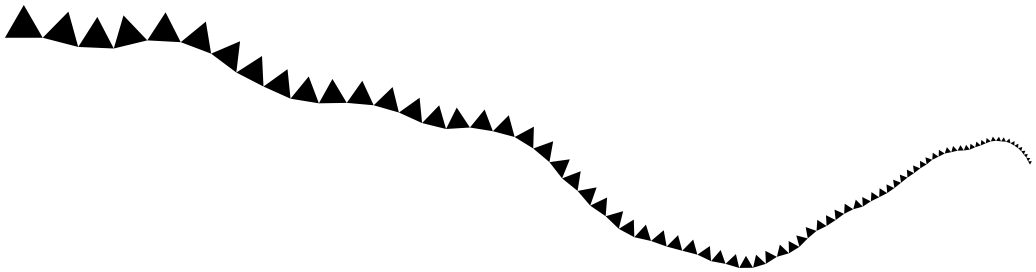
2. (1.0) Considere a seguinte pirâmide de círculos.

   Note que os círculos estão tangentes uns aos outros e que cada fila de círculos, excepto a mais em baixo, tem menos um círculo do que a fila imediatamente abaixo.
   Defina uma função denominada `piramide_circulos` que, a partir do ponto P, do número n de círculos da fila da base, e do raio r dos círculos, constrói a pirâmide de círculos.




```
def triangulo_pintas(p,r,n):
    if n==0:
        pass
    else:
        for i in range(n):
            circle(p+vx(i*2*r),r)
        triangulo_pintas(p+vxy(r,r*sqrt(3)),r,n-1)

triangulo_pintas(u0(),1,6)
```

3. (1.0) Considere a seguinte *fita de triângulos*.



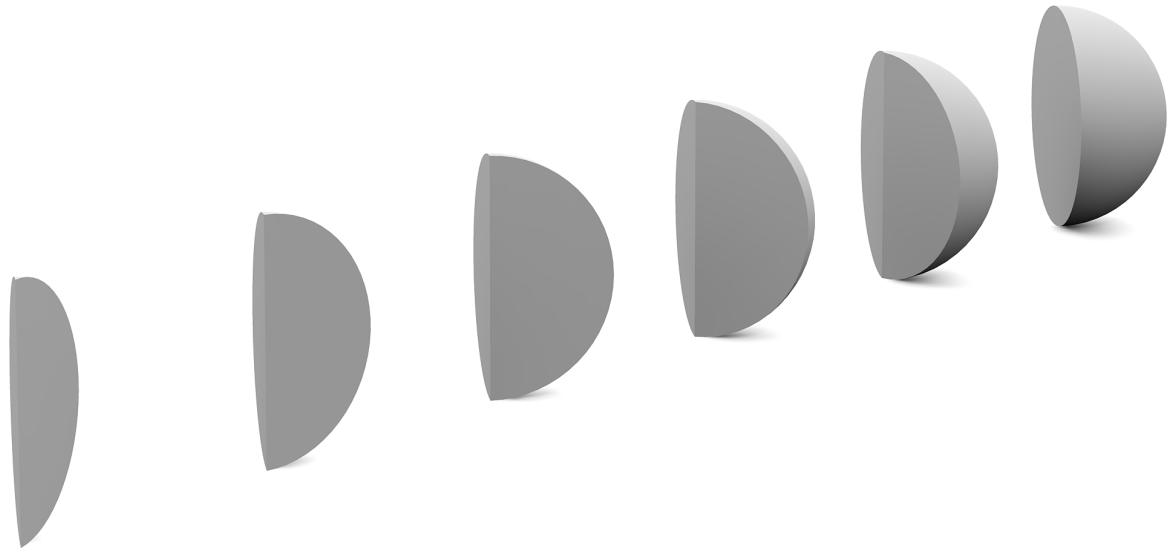
   Note que os triângulos são equiláteros, cada par de triângulos partilha um vértice e não há triângulos cujas arestas tenham um comprimento inferior a 1. Note ainda que os triangulos vão rodando de forma semi-aleatória.

   Defina a função `fita_triangulos` que recebe o ponto inicial da fita (correspondente ao vértice esquerdo do primeiro triângulo), o comprimento da aresta do triângulo equilátero, o ângulo da aresta inferior do triângulo, a variação máxima do ângulo entre um triângulo e o seguinte e, finalmente, um factor de redução a aplicar aos triângulos.

   Sugestão: utilize a função `random_range(`a, b`)` para calcular um valor aleatório entre a e b.

```
def fita_triangulos(p1,l,a,da,f):
  if l <= 1:
    pass
  else:
    p2=p1+vpol(l, a)
    p3=p1+vpol(l, a+pi/3)
    surface_polygon(p1, p2, p3)
    fita_triangulos(p2, l*f, a+random_range(-1.0,1.0)*da, da, f)

fita_triangulos(x(0), 10, 0, pi/10, 0.97)
```

4. (1.0) Considere a seguinte imagem que representa diferentes *gomos* de esfera.



Defina a função `gomo_esfera` que, dado o centro e raio de uma esfera e ainda dois ângulos medidos no plano XY, produz a parte da esfera que se encontra delimitada por esses dois ângulos.

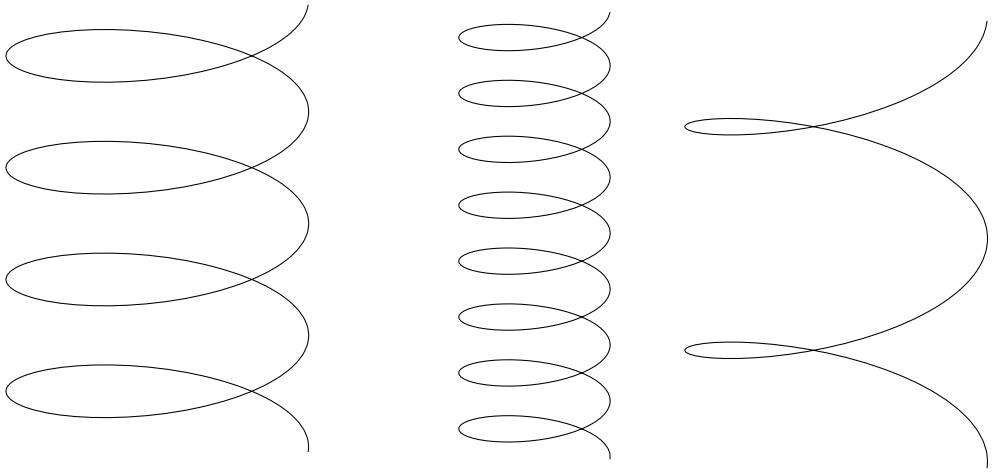
```
def gomo_esfera(p, r, fi0, fi1):
    return slice(slice(sphere(p, r),
                       p,
                       vcyl(-1, fi0+pi/2, 0)),
                 p,
                 vcyl(1, fi1+pi/2, 0))

gomo_esfera(xy(0,0), 1, 0, pi/4)
```

5. (1.0) Considere a seguinte representação paramétrica:

$$\begin{cases} \rho(t) = \alpha \\ \phi(t) = \beta t \\ z(t) = \gamma t \end{cases}$$

em que $t \in [0, 2\pi]$

Desenhe, à mão livre, três exemplos da curva descrita pela representação paramétrica anterior.



6. (1.0) Relativamente à curva descrita na pergunta anterior, defina uma função Python que recebe como parâmetros α, β e γ e ainda um número inteiro n, e que devolve com resultado uma lista contendo n pontos pertencentes à curva.

```
def pontos_helice(a, b, g, n):
    return map_division(lambda ti: cyl(a, b*ti, g*ti),
                        0, 2*pi, n)
```

7. (1.0) Defina a função de ordem superior `sucessao` que recebe os limites a e b de um intervalo $[a, b[$ e uma função f e que gera uma lista com todos os elementos x_i inferiores a b tais que $x_0 = a$ e $x_{i+i} = f(x_i)$. Por exemplo:

```
>>> sucessao(2, 600, lambda x: x*2)
[2, 4, 8, 16, 32, 64, 128, 256, 512]
```

```
def sucessao(a, b, f):
    if a >= b:
        return []
    else:
        return [a] + sucessao(f(a), b, f)
```

8. (1.0) Um *toro* de raios maior r_0 e menor r_1 é definido parametricamente pela equação:

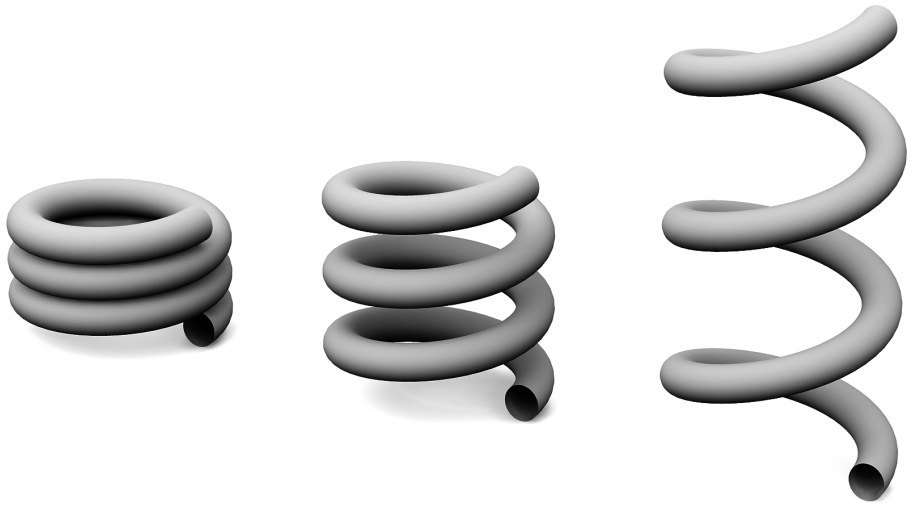
$$\begin{aligned} \rho(u,v) &= r_0 + r_1 \cos u \\ \phi(u,v) &= v \\ z(u,v) &= r_1 \sin u \end{aligned}$$

$$0 \leq u \leq 2\pi; \quad 0 \leq v \leq 2\pi;$$

Defina uma função Python que, a partir dos parâmetros de um toro e de um número de valores a considerar para o domínio de u e outro para o domínio de v, cria uma superfície com a forma de um toro.

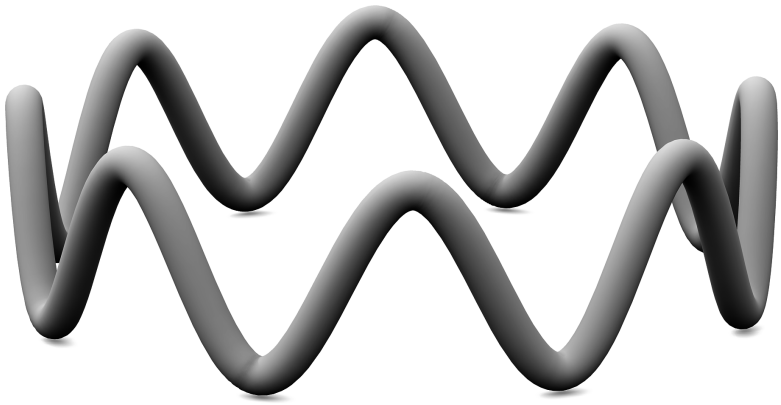
```
def toro(r0, r1, m, n):
    return surface_grid(
                map_division(lambda u, v: cyl(r0 + r1*cos(v),
                                              u,
                                              r1*sin(v)),
                             0, 2*pi, m, False,
                             0, 2*pi, n, False),
                True,
                True)
```

9. (1.0) Baseando-se na equação paramétrica do toro, escreva a equação paramétrica de uma mola, tal como se apresenta nas imagens seguintes:



$$\begin{cases} \rho(u,v) = r_0 + r_1 \cos u \\ \theta(u,v) = \alpha v \\ z(u,v) = \beta v + r_1 \sin u \end{cases}$$

10. (1.0) Considere o anel sinusoidal representado na imagem seguinte:



O anel é constituído por um arame de raio r_i cujo eixo é uma sinusóide com uma determinada amplitude a e frequência ω que se desenvolve ao longo de uma circunferência de raio r_e centrada num ponto p. Considere que a curva da sinusóide pode ser aproximada por uma *spline* que passa por n pontos pertencentes à sinusóide.

Defina a função `anel_sinusoidal` que, a partir dos parâmetros p, r_e, r_i, a, ω e n, constrói o anel pretendido.

```
def anel_sinusoidal(p, re, ri, a, omega, n):
  return sweep(
           closed_spline(
             map_division(lambda t: p+vcyl(re, t, a*sin(omega*t)),
                          0, 2*pi, n, False)),
           surface_circle(x(0), ri))
```