Exercícios de
Programação e Computação para Arquitectura

António Menezes Leitão

Conteúdo

Aula 1
Introdução ao Julia

1.1 Exercícios

Exercício 1.1.1 Converta as seguintes expressões da notação do Julia para a notação da aritmética:

1. `1/2*3`
2. `1/(2 - 3)`
3. `(1 + 2)/3`
4. `1/2/3`
5. `1/(2/3)`

Exercício 1.1.2 Calcule *mentalmente* o valor das seguintes expressões Julia e depois teste os seus palpites usando a **REPL** do Julia:

1. `1/2*3`
2. `1*(2 - 3)`
3. `(1 + 2)/3`
4. `1 - 2 - 3`

Exercício 1.1.3 Traduza as seguintes expressões matemáticas para Julia:

1. $\sqrt{\frac{1}{\log 2^{|(3-9\log 25)|}}}$

2. $\frac{\cos^4 \frac{2}{\sqrt{5}}}{\operatorname{atan} 3}$
3. $\frac{1}{2} + \sqrt{3} + \sin^{\frac{5}{2}} 2$

Exercício 1.1.4 Traduza as seguintes expressões Julia para a notação matemática:

1. `log(sin(2^4 + floor(atan(pi))/sqrt(5)))`
2. `cos(cos(cos(0.5)))^5`
3. `sin(cos(sin(pi/3)/3)/3)`

Exercício 1.1.5 Defina a função `radianos` que recebe uma quantidade angular em graus e calcula o valor correspondente em radianos. Note que 180 graus correspondem a π radianos. Teste a sua função na REPL do Julia, por exemplo, com o valor `180`.

Exercício 1.1.6 Defina a função `graus` que recebe uma quantidade angular em radianos e calcula o valor correspondente em graus. Teste a sua função na interacção do Julia, por exemplo, com o valor `pi`.

Exercício 1.1.7 A área A de um pentágono regular inscrito num círculo de raio r é dada pela fórmula

$$A = \frac{5}{8} r^2 \sqrt{10 + 2\sqrt{5}}$$

Defina uma função em Julia que calcule essa área. Teste-a na interacção do Julia para valores à sua escolha.

Exercício 1.1.8 Defina o predicado `impar` que, dado um número, testa se ele é ímpar, i.e., se o resto da divisão desse número por dois é um. Para calcular o resto da divisão de um número por outro, utilize a operação pré-definida `%`.

Exercício 1.1.9 Pretende-se criar um lanço de escada com n espelhos capaz de vencer uma determinada altura a em metros. Admitindo que cada degrau tem uma altura do espelho h e uma largura do cobertor d que verificam a proporção

$$2h + d = 0.64$$

defina uma função que, a partir da altura a vencer e do número de espelhos, calcula o comprimento do lanço de escada.

Exercício 1.1.10 Qual o valor das seguintes expressões?

1. `(2 > 3 || !(2 == 3)) && 2 < 3`
2. `! (1 == 2 || 2 == 3)`
3. `1 < 2 || 1 == 2 || 1 > 2`

Exercício 1.1.11 Sem usar a função predefinida `max`, defina a função `max2` que recebe dois números como argumento e calcula o maior entre eles.

Exercício 1.1.12 Sem usar a função predefinida `max`, defina a função `max3` que recebe três números como argumento e calcula o maior entre eles.

Exercício 1.1.13 Defina uma função `soma_maiores` que recebe três números como argumento e determina a soma dos dois maiores.

Exercício 1.1.14 Defina a função `segundo_maior` que recebe três números como argumento e devolve o segundo maior número, i.e., o que está entre o maior e o menor.

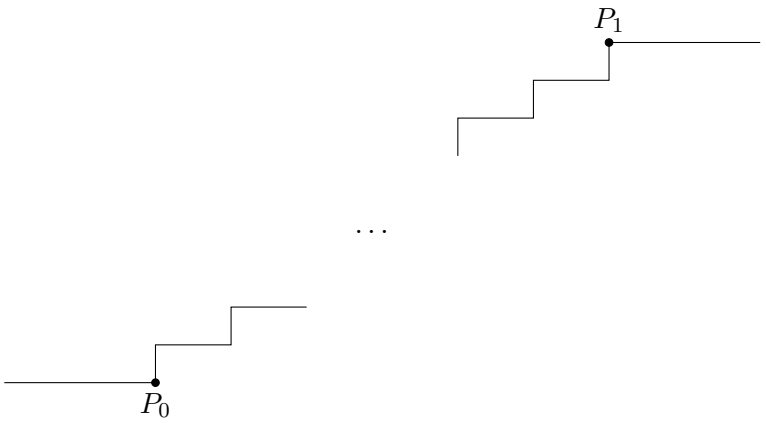
Aula 2
Operações Geométricas

2.1 Introdução

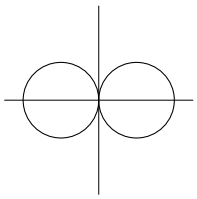
Considere as operações de manipulação de coordenadas `xyz`, `vxyz`, `xy`, `vxy`, `pol`, `vpol`, `cx`, `cy`, e `cz`.

2.2 Exercícios

Exercício 2.2.1 Sabendo que o espelho máximo admissível para um degrau é de 0.18m, defina uma função que calcula o número mínimo de espelhos que a escada representada no seguinte esquema necessita para fazer a ligação entre os pontos P_0 e P_1.

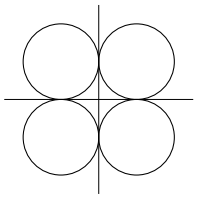


Exercício 2.2.2 Pretendemos colocar duas circunferências de raio unitário em torno da origem de modo a que fiquem encostadas uma à outra, tal como se ilustra no seguinte desenho:



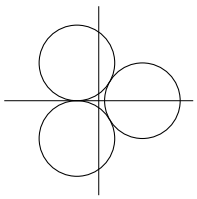
Escreva uma sequência de expressões que, quando avaliadas, produzem a figura anterior.

Exercício 2.2.3 Pretendemos colocar quatro circunferências de raio unitário em torno da origem de modo a que fiquem encostadas umas às outras, tal como se ilustra no seguinte desenho:



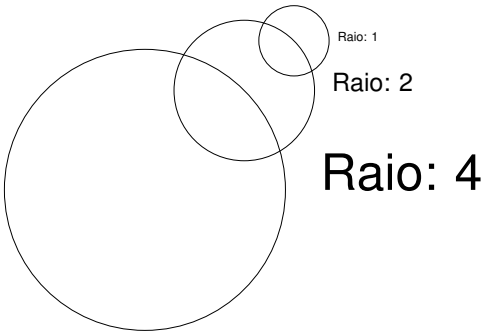
Escreva uma sequência de expressões que, quando avaliadas, produzem a figura anterior.

Exercício 2.2.4 Pretendemos colocar três circunferências de raio unitário em torno da origem de modo a que fiquem encostadas umas às outras, tal como se ilustra no seguinte desenho:



Escreva uma sequência de expressões que, quando avaliadas, produzem a figura anterior.

Exercício 2.2.5 Considere o seguinte desenho:



e as expressões que o produziram:

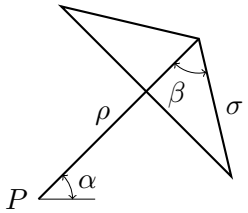
```
circle(pol(0, 0), 4)
text("Raio: 4", pol(0, 0) + vpol(5, 0), 1)
circle(pol(4, pi/4), 2)
text("Raio: 2", pol(4, pi/4) + vpol(2.5, 0), 0.5)
circle(pol(6, pi/4), 1)
text("Raio: 1", pol(6, pi/4) + vpol(1.25, 0), 0.25)
```

Repare que as expressões anteriores utilizam coordenadas polares. Refaça o desenho anterior mas utilizando apenas coordenadas rectangulares.

Exercício 2.2.6 Defina uma função denominada `circulo_e_raio` que, dadas as coordenadas do centro do círculo e o raio desse círculo, cria o círculo especificado e, à semelhança do que se vê no desenho anterior, coloca o texto a descrever o raio do círculo à direita desse círculo. O texto deverá ter um tamanho proporcional ao raio do círculo. **Sugestão:** Consulte a documentação da função `string`.

Exercício 2.2.7 Utilize a função `circulo_e_raio` definida na pergunta anterior para reconstituir o desenho anterior.

Exercício 2.2.8 Considere o desenho de uma seta com origem no ponto P, comprimento ρ, inclinação α, ângulo de abertura β e comprimento da "farpa" σ, tal como se representa em seguida:



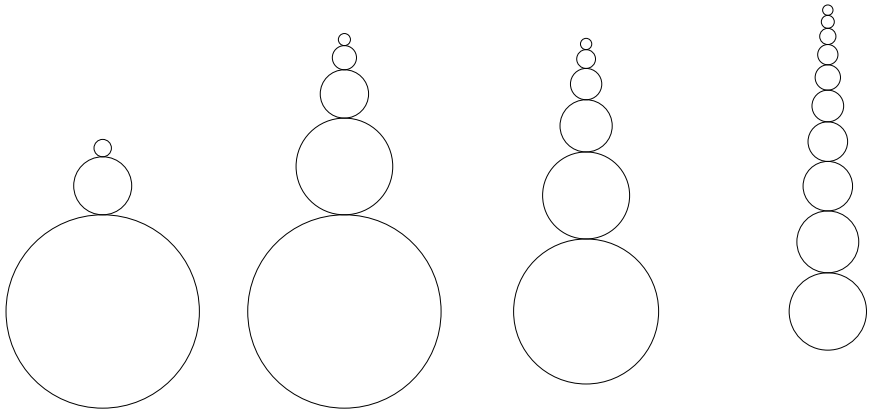
Defina uma função denominada `seta` que, a partir dos parâmetros P, ρ, α, σ e β, constrói a seta correspondente.

Aula 3

Recursão

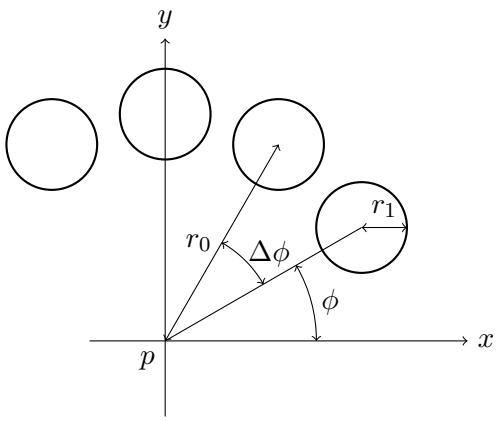
3.1 Exercícios

Exercício 3.1.1 Defina uma função `equilibrio_circulos` capaz de criar qualquer uma das figuras apresentadas em seguida:



Note que os círculos possuem raios que estão em progressão geométrica de razão f, com $0 < f < 1$. Assim, cada círculo (excepto o primeiro) tem um raio que é o produto de f pelo raio do círculo maior em que está apoiado. O círculo mais pequeno de todos tem raio maior ou igual a 1. A sua função deverá ter como parâmetros o centro e o raio do círculo maior e, ainda, o factor de redução f.

Exercício 3.1.2 Considere o desenho de círculos tal como apresentado na seguinte imagem:

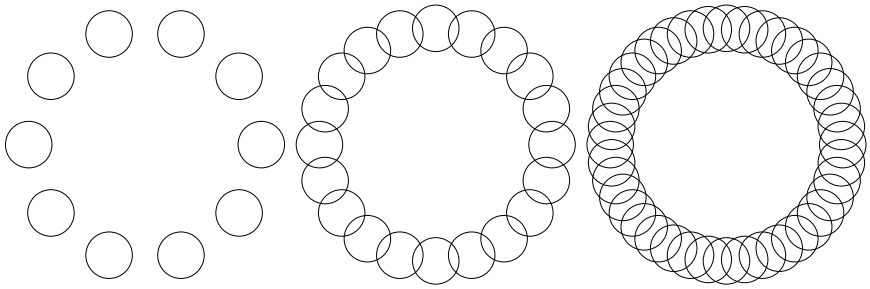


Escreva uma função denominada `circulos_radiais` que, a partir das coordenadas p do centro de rotação, do número de círculos n, do raio de translacção r_0, do raio de circunferência r_1, do ângulo inicial ϕ e do incremento de ângulo $\Delta\phi$, desenha os círculos tal como apresentados na figura anterior.

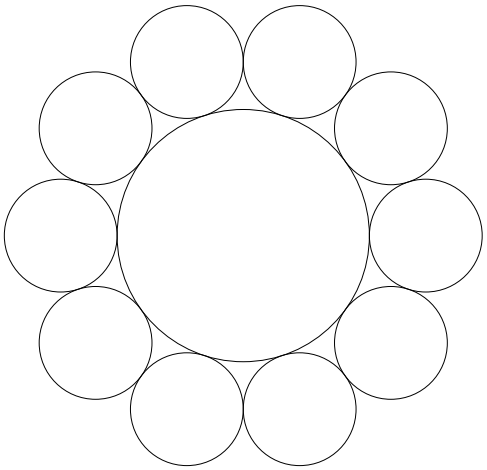
Teste a sua função com os seguintes expressões:

```
circulos_radiais(xy(0, 0), 10, 10, 2, 0, pi/5)
circulos_radiais(xy(25, 0), 20, 10, 2, 0, pi/10)
circulos_radiais(xy(50, 0), 40, 10, 2, 0, pi/20)
```

cuja avaliação deverá gerar a imagem seguinte:



Exercício 3.1.3 Considere o desenho de flores simbólicas compostas por um círculo interior em torno da qual estão dispostos círculos radiais correspondentes a pétalas. Estes círculos deverão ser tangentes uns aos outros e ao círculo interior, tal como se apresenta na seguinte imagem:

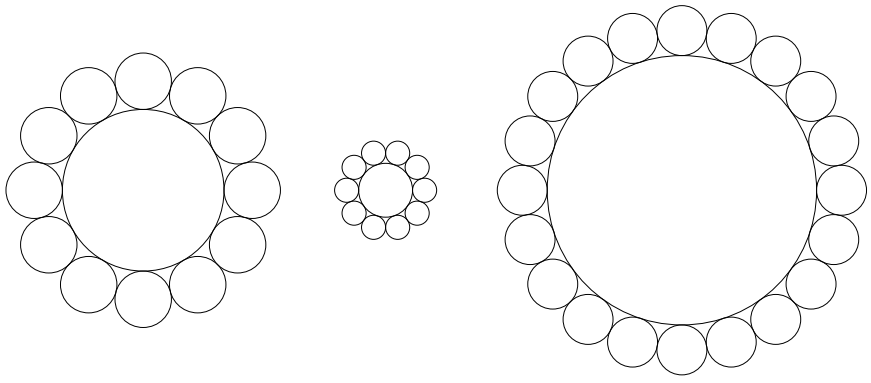


Defina a função `flor` que recebe apenas o ponto correspondente ao centro da flor, o raio do círculo interior e o número de pétalas.

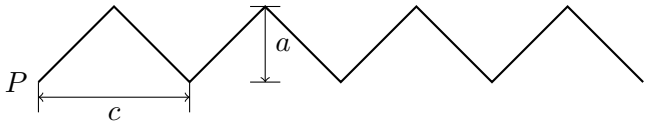
Teste a sua função com as expressões

```
flor(xy(0, 0), 5, 10)
flor(xy(18, 0), 2, 10)
flor(xy(40, 0), 10, 20)
```

que deverão gerar a imagem seguinte:



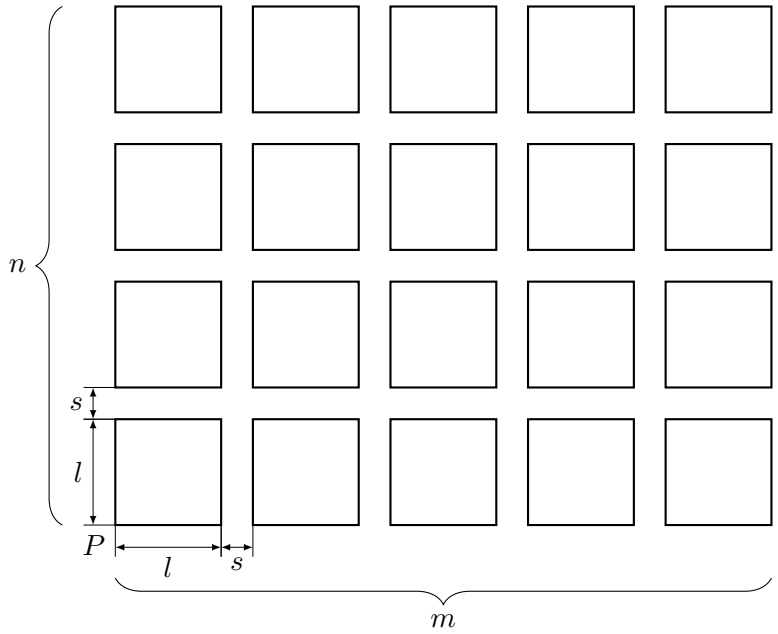
Exercício 3.1.4 Defina uma função denominada `serra` que, dado um ponto P, um número de dentes, o comprimento c de cada dente e a altura a de cada dente, desenha uma serra com o primeiro dente a começar a partir do ponto P, tal como se vê na imagem seguinte:


P
c
a

Aula 4
Aleatoriedade

4.1 Introdução

Considere uma cidade organizada em termos de quarteirões. Para simplificar, vamos assumir que cada quarteirão será de forma quadrada e irá conter um único edifício. Cada edifício terá uma largura determinada pela largura do quarteirão e uma altura máxima imposta. Os edifícios serão separados uns dos outros por ruas com uma largura fixa.



As duas seguintes funções modelam uma destas cidades:

```
malha_predios(p, n_ruas, m_predios, l, h, s) =
    if n_ruas == 0
        nothing
    else
        rua_predios(p, m_predios, l, h, s)
        malha_predios(p + vy(l + s), n_ruas - 1, m_predios, l, h, s)
    end

rua_predios(p, m_predios, l, h, s) =
    if m_predios == 0
        nothing
    else
        predio(p, l, h)
        rua_predios(p + vx(l + s), m_predios - 1, l, h, s)
    end
```

A função que modela cada edifício é a seguinte:

```
predio(p, l, h) =
    box(p, l, l, h)
```

A título de exemplo, considere a Figura 4.1 que foi produzida pela seguinte expressão:

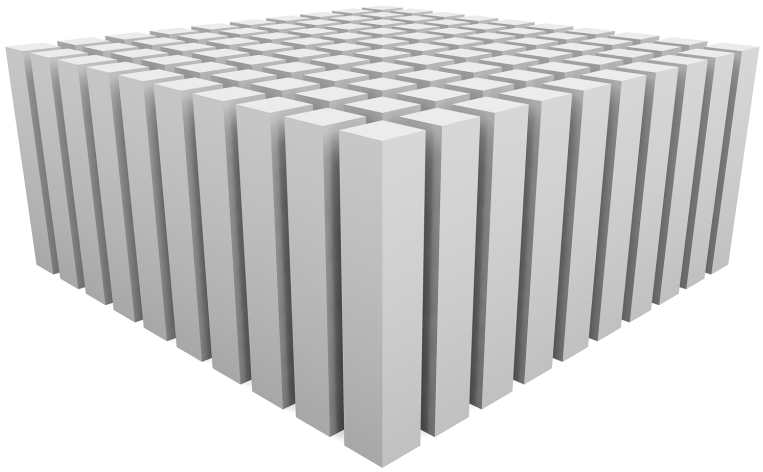


Figura 4.1: Uma urbanização em malha ortogonal com cem edifícios todos da mesma altura.

```
malha_predios(xyz(0, 0, 0), 10, 10, 100, 400, 40)
```

Para tornar as cidades produzidas mais realistas, considere a utilização das funções aleatórias `random` e `random_range`.

4.2 Exercícios

Exercício 4.2.1 Torne a cidade gerada pela função anterior mais realista através da incorporação da aleatoriedade na altura de cada edifício, tal como se apresenta na Figura 4.2. Dada a altura máxima de um edifício, a altura actual deverá ser um valor aleatório entre 10% da altura máxima e 100% da altura máxima.

Exercício 4.2.2 As urbanizações produzidas pelas funções anteriores não apresentam variabilidade suficiente pois os edifícios têm todos a mesma forma. Para melhorar a qualidade estética da urbanização pretende-se empregar diferentes funções para a construção de diferentes tipos de edifícios: a função `predio0` deverá construir um paralelipípedo de altura aleatória tal como anteriormente e a função `predio1` deverá construir uma torre cilíndrica de altura aleatória e contida no espaço de um quarteirão. Defina estas duas funções.

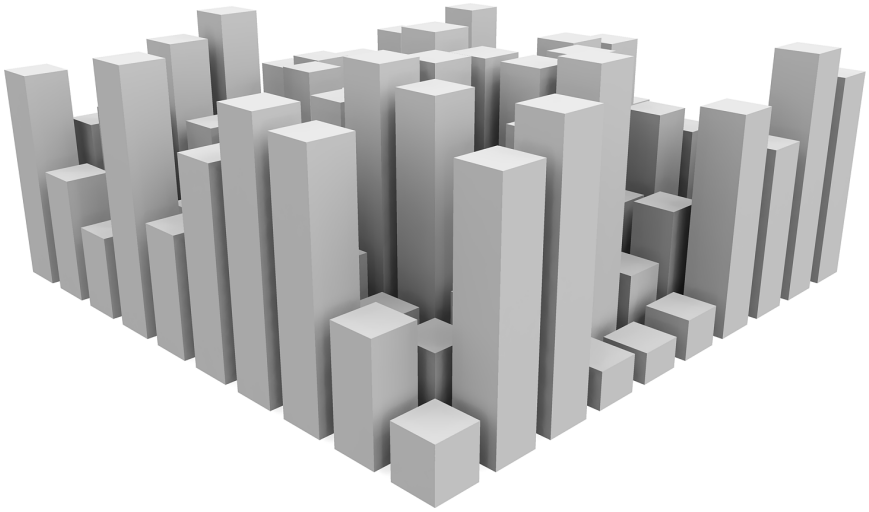
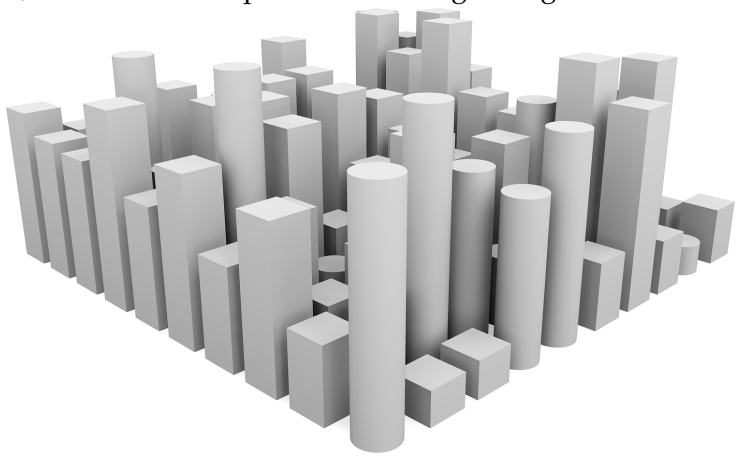


Figura 4.2: Uma urbanização em malha ortogonal com cem edifícios com alturas aleatórias.

Exercício 4.2.3 Pretende-se que use as duas funções `predio0` e `predio1` definidas no exercício anterior para a redefinição da função `predio` de modo a que esta, aleatoriamente, invoque uma ou outra de modo a construir prédios diferentes. A urbanização resultante deverá ser constituída aproximadamente por 20% de torres circulares e 80% de prédios rectangulares, tal como é exemplificado na imagem seguinte:

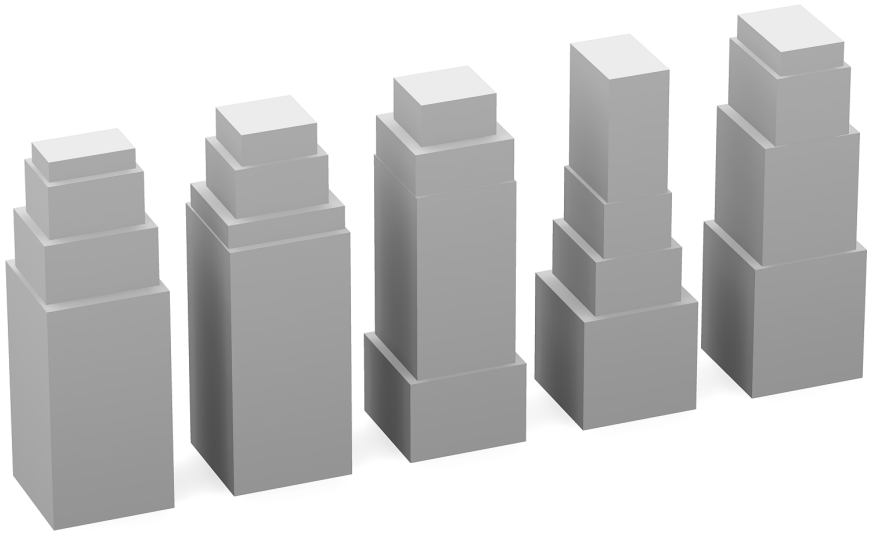


Exercício 4.2.4 As cidades criadas nos exercícios anteriores apenas permitem dois tipos de edifícios: torres paralelipipédicas ou torres cilíndricas.

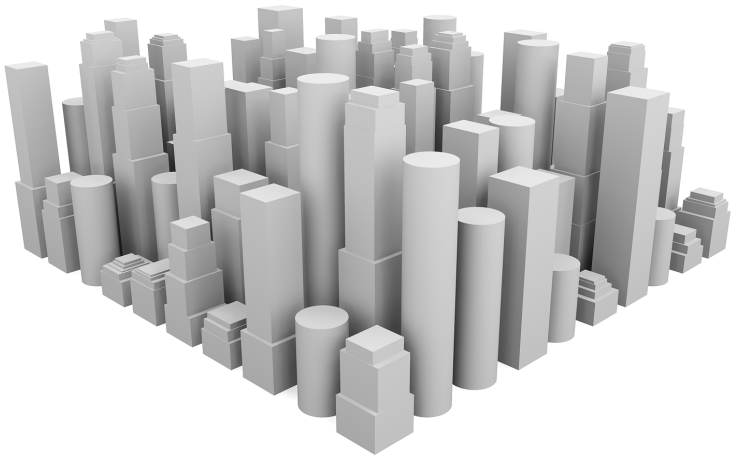
Contudo, quando observamos uma cidade real, verificamos que existem prédios com muitas outras formas pelo que, para aumentarmos o realismo das nossas simulações, teremos de implementar um vasto número de funções, cada uma capaz de construir um edifício diferente.

Felizmente, uma observação atenta mostra que, na verdade, muitos dos prédios seguem um padrão e podem ser modelados por paralelipípedos sobrepostos com dimensões aleatórias mas sempre sucessivamente mais pequenos em função da altura, algo que podemos facilmente implementar com uma única função.

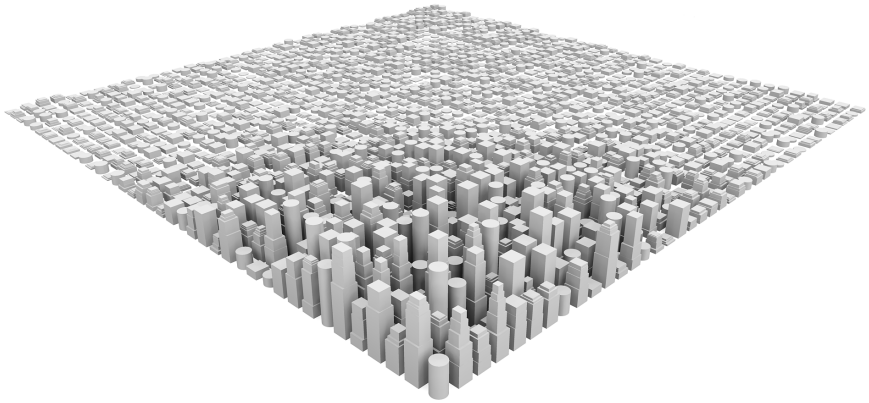
Considere que este modelo de edifício é parameterizado pelo número de "blocos" sobrepostos pretendidos, pelas coordenadas do primeiro bloco e pelo comprimento, largura e altura do edifício. O bloco de base tem exactamente o comprimento e largura especificados mas a sua altura deverá estar entre 20% e 80% da altura total do edifício. Os blocos seguintes estão centrados sobre o bloco imediatamente abaixo e possuem um comprimento e largura entre 70% e 100% dos parâmetros correspondentes desse bloco. A altura destes blocos deverá ser entre 20% e 80% da altura restante do edifício. A imagem seguinte mostra alguns exemplos deste modelo de edifícios:



Com base nesta especificação, defina a função `predio_blocos` e use-a para redefinir a função `predio0` de modo a que sejam gerados prédios com um número aleatório de blocos sobrepostos. Com esta redefinição, a função `malha_predios` deverá ser capaz de gerar urbanizações semelhantes à imagem seguinte, em que se empregou para cada prédio um número de blocos entre 1 e 5:



Exercício 4.2.5 Em geral, as cidades possuem um núcleo de edifícios relativamente altos no centro e, à medida que nos afastamos para a periferia, a altura tende a diminuir, sendo este fenómeno perfeitamente visível na figura seguinte:



A variação da altura dos edifícios pode ser modelada por diversas funções matemáticas mas, neste exercício, pretende-se que empregue uma distribuição Gaussiana bi-dimensional dada por

$$f(x,y) = e^{-\left((\frac{x-x_o}{\sigma_x})^2+(\frac{y-y_o}{\sigma_y})^2\right)}$$

em que f é o factor de ponderação da altura, (x_0, y_0) são as coordenadas do ponto mais alto da superfície Gaussiana e σ_x e σ_y são os factores que afectam o alargamento bi-dimensional dessa superfície. Para simplificar,

assuma que o centro da cidade fica nas coordenadas $(0, 0)$ e que $\sigma_x = \sigma_y = 25l$, sendo l a largura do edifício. Incorpore esta distribuição no processo de construção de cidades para produzir cidades mais realistas.

Aula 5
Treliças

5.1 Introdução

Uma treliça é uma estrutura composta por barras rígidas que se unem em nós, formando unidades triangulares, tal como se ilustra na Figura 5.1.

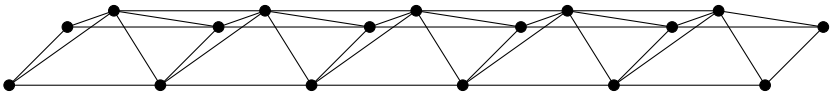


Figura 5.1: Treliça composta por elementos triangulares iguais.

Para o desenho de uma treliça, vamos considerar, como base de trabalho, três sequências arbitrárias de pontos em que cada ponto define um nó da treliça. A partir destas três sequências podemos criar as ligações necessárias entre cada par de nós. A Figura 5.2 apresenta o esquema de ligação a partir de três sequências de pontos (a_0, a_1, a_2), (b_0, b_1) e (c_0, c_1, c_2). Note-se que a sequência de topo estabelecida pelos pontos b_i da sequência intermédia tem sempre menos um elemento que as sequências a_i e c_i.

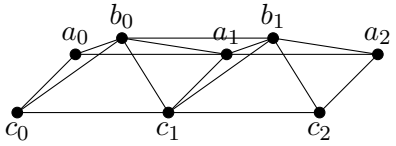


Figura 5.2: Esquema de ligação de barras de uma treliça em *space frame*.

Para a construção da treliça precisamos de encontrar um processo que, a partir das listas de pontos a_i, b_i e c_i, não só crie os nós correspondentes

aos vários pontos, como os interligue da forma correcta.

Podemos começar a idealizar a função que constrói a treliça completa a partir das listas de pontos `ais`, `bis` e `cis`:

```
trelica(ais, bis, cis) =
  begin
    nos_trelica(ais)
    nos_trelica(bis)
    nos_trelica(cis)
    ...
```

Comecemos por tratar da criação dos nós:

```
nos_trelica(ps) =
  for p in ps
    no_trelica(p)
  end
```

A função `no_trelica` (notemos o singular, por oposição ao plural empregue na função `nos_trelica`) recebe as coordenadas de um ponto e é responsável por criar o modelo tridimensional que representa o nó da treliça centrado nesse ponto.

De seguida, vamos tratar de estabelecer as barras entre os nós. Da análise da Figura 5.2 ficamos a saber que temos uma ligação entre cada a_i e cada c_i, outra entre a_i e b_i, outra entre c_i e b_i, outra entre b_i e a_{i+1}, outra entre b_i e c_{i+1}, outra entre a_i e a_{i+1}, outra entre b_i e b_{i+1} e, finalmente, outra entre c_i e c_{i+1}. Admitindo que a função `barra_trelica` cria o modelo tridimensional dessa barra (por exemplo, um cilindro, ou uma barra prismática), podemos começar por definir uma função denominada `barras_trelica` (notemos o plural) que, dadas duas listas de pontos `ps` e `qs`, cria barras de ligação ao longo dos sucessivos pares de pontos. Para criar uma barra, a função necessita de um elemento dos `ps` e outro dos `qs`, o que implica que a função deve terminar assim que uma destas listas estiver vazia. A definição fica então:

```
barras_trelica(ps, qs) =
  for (p, q) in zip(ps, qs)
    barra_trelica(p, q)
  end
```

Para interligar cada nó a_i ao correspondente nó c_i, apenas temos de avaliar `barras_trelica(ais, cis)`. O mesmo poderemos dizer para interligar cada nó b_i ao nó a_i correspondente e para interligar cada b_i a cada c_i. Assim, temos:

```
trelica(ais, bis, cis) =
  begin
    nos_trelica(ais)
    nos_trelica(bis)
    nos_trelica(cis)
    barras_trelica(ais, cis)
    barras_trelica(bis, ais)
    barras_trelica(bis, cis)
    ...
```

Para ligar os nós b_i aos nós a_{i+1} podemos simplesmente subtrair o primeiro nó da lista `ais` e estabelecer a ligação como anteriormente. O mesmo podemos fazer para ligar cada b_i a cada c_{i+1}. Finalmente, para ligar cada a_i a cada a_{i+1} podemos usar a mesma ideia mas aplicando-a apenas à lista `ais`. O mesmo podemos fazer para as listas `bis` e `cis`. A função completa fica, então:

```
trelica(ais, bis, cis) =
  begin
    nos_trelica(ais)
    nos_trelica(bis)
    nos_trelica(cis)
    barras_trelica(ais, cis)
    barras_trelica(bis, ais)
    barras_trelica(bis, cis)
    barras_trelica(bis, ais[2:end])
    barras_trelica(bis, cis[2:end])
    barras_trelica(ais[2:end], ais)
    barras_trelica(cis[2:end], cis)
    barras_trelica(bis[2:end], bis)
  end
```

As funções anteriores constroem treliças com base nas funções "elementares" `no_trelica` e `barra_trelica`. Embora o seu significado seja óbvio, ainda não definimos estas funções e existem várias possibilidades. Numa primeira abordagem, vamos considerar que cada nó da treliça será constituído por uma esfera onde se irão unir as barras, barras essas que serão definidas por cilindros. O raio das esferas e da base dos cilindros será determinado por uma variável global, para que possamos facilmente alterar o seu valor. Assim, temos:

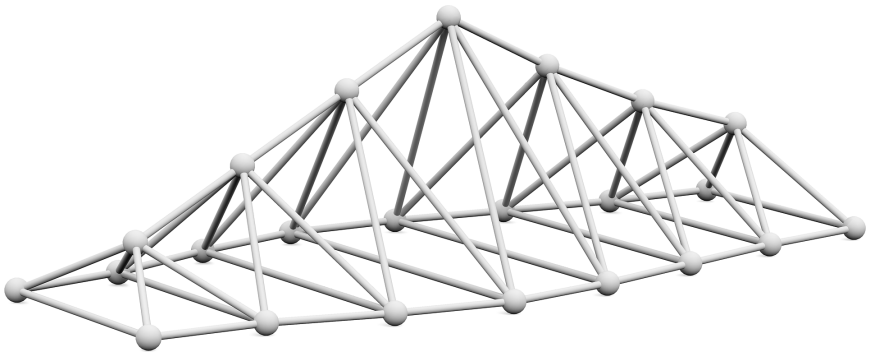


Figura 5.3: Treliça construída a partir de pontos especificados arbitrariamente.

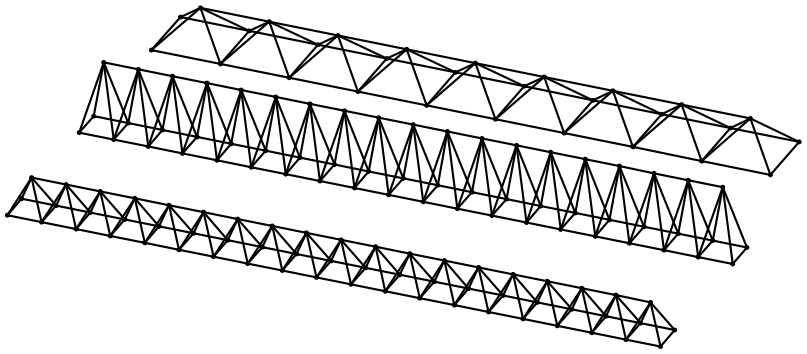
```
raio_no_trelica = 0.1

no_trelica(p) =
  sphere(p, raio_no_trelica)

raio_barra_trelica = 0.03

barra_trelica(p0, p1) =
  cylinder(p0, raio_barra_trelica, p1)
```

A Figura 5.3 mostra uma treliça desenhada a partir da expressão:

```
trelica([xyz(0, -1, 0), xyz(1, -1.1, 0), xyz(2, -1.4, 0),
         xyz(3, -1.6, 0), xyz(4, -1.5, 0), xyz(5, -1.3, 0),
         xyz(6, -1.1, 0), xyz(7, -1, 0)],
        [xyz(0.5, 0, 0.5), xyz(1.5, 0, 1), xyz(2.5, 0, 1.5),
         xyz(3.5, 0, 2), xyz(4.5, 0, 1.5), xyz(5.5, 0, 1.1),
         xyz(6.5, 0, 0.8)],
        [xyz(0, 1, 0), xyz(1, 1.1, 0), xyz(2, 1.4, 0),
         xyz(3, 1.6, 0), xyz(4, 1.5, 0), xyz(5, 1.3, 0),
         xyz(6, 1.1, 0), xyz(7, 1, 0)])
```

5.2 Exercícios

Exercício 5.2.1 Defina uma função denominada `trelica_recta` capaz de construir qualquer uma das treliças que se apresentam na imagem seguinte.

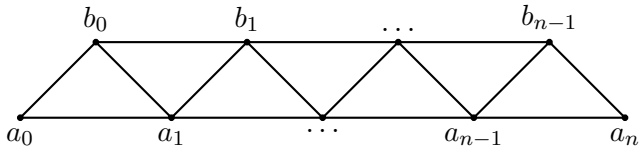


Para simplificar, considere que as treliças se desenvolvem segundo o eixo X. A função `trelica_recta` deverá receber o ponto inicial da treliça, a altura e largura da treliça e o número de nós das fileiras laterais. Com esses valores, a função deverá produzir três listas de coordenadas que passará como argumentos à função `trelica`. Como exemplo, considere que as três treliças apresentadas na imagem anterior foram o resultado da avaliação das expressões:

```
trelica_recta(xyz(0, 0, 0), 1.0, 1.0, 20)
trelica_recta(xyz(0, 5, 0), 2.0, 1.0, 20)
trelica_recta(xyz(0, 10, 0), 1.0, 2.0, 10)
```

Sugestão: comece por definir a função `coordenadas_x` que, dado um ponto inicial p, um afastamento l entre pontos e um número n de pontos, devolve uma lista com as coordenadas dos n pontos dispostos ao longo do eixo X.

Exercício 5.2.2 Considere o desenho de uma treliça *plana*, tal como se apresenta na seguinte figura:

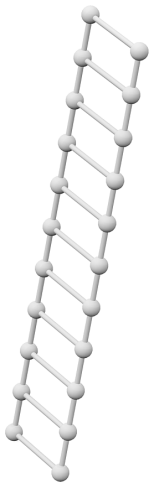


Defina uma função `trelica_plana` que recebe, como parâmetros, duas listas de pontos correspondentes aos pontos desde a_0 até a_n e desde b_0 até b_{n-1} e que cria os nós nesses pontos e as barras que os unem. Considere, como pré-definidas, as funções `nos_trelica` que recebe uma lista de posições como argumento e `barras_trelica` que recebe duas listas de posições como argumentos.

Teste a função que definiu com a seguinte expressão:

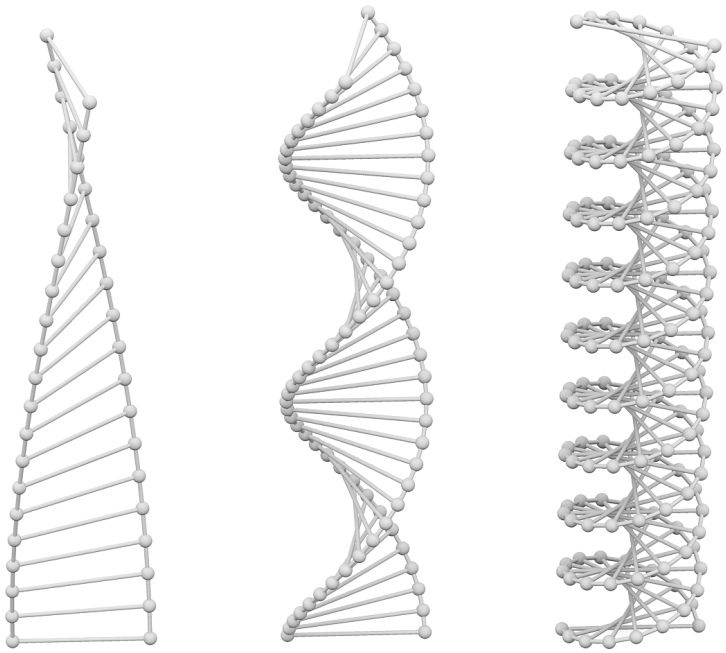
```
trelica_plana(coordenadas_x(xyz(0, 0, 0), 2.0, 20),
              coordenadas_x(xyz(1, 0, 1), 2.0, 19))
```

Exercício 5.2.3 Considere o desenho de uma escada de mão idêntica à que se apresenta na seguinte figura:



Repare que a escada de mão pode ser vista como uma versão (muito) simplificada de uma treliça composta por apenas duas sequências de nós. Defina a função `escada_de_mao` que recebe, como parâmetros, duas listas de pontos correspondentes aos centros das sequências de nós e que cria os nós nesses pontos e as barras que os unem. Considere, como pré-definidas, as funções `nos_trelica` que recebe uma lista de posições como argumento e `barras_trelica` que recebe duas listas de posições como argumentos.

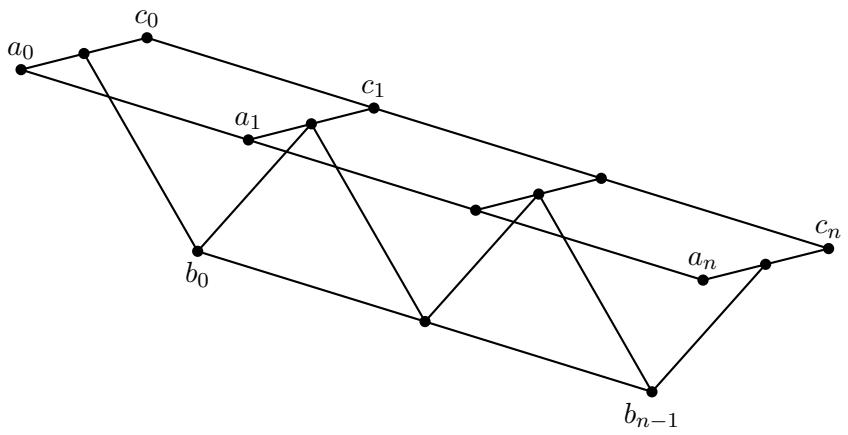
Exercício 5.2.4 Defina uma função capaz de representar a dupla hélice do genoma, tal como se pode ver na seguinte imagem:



A função deverá receber uma posição referente ao centro da base do genoma, o raio da hélice do genoma, a diferença angular entre os nós, a diferença em altura entre nós e, finalmente, o número de nós em cada hélice. Os genomas apresentados na imagem anterior foram gerados pelas seguintes expressões:

```
genoma(xyz(0, 0, 0), 1.0, pi/32, 0.5, 20)
genoma(xyz(4, 0, 0), 1.0, pi/16, 0.25, 40)
genoma(xyz(8, 0, 0), 1.0, pi/8, 0.125, 80)
```

Exercício 5.2.5 Considere o desenho da treliça especial apresentado na seguinte figura:



Defina uma função `trelica_especial` que recebe, como parâmetros, três listas de pontos correspondentes aos pontos desde a_0 até a_n, desde b_0 até b_{n-1} e desde c_0 até c_n e que cria os nós nesses pontos e as barras que os unem. Considere, como pré-definidas, as funções `nos_trelica` que recebe uma lista de posições como argumento e `barras_trelica` que recebe duas listas de posições como argumentos.

Exercício 5.2.6 Defina a função `trelica_especial_recta` capaz de construir a trelica especial descrita no exercício anterior admitindo que esta se desenvolve segundo o eixo X. A função deverá receber o ponto inicial da treliça, a altura e largura da treliça e o número de nós das fileiras laterais. Com esses valores, a função deverá produzir três listas de coordenadas que passará como argumentos à função `trelica_especial`.

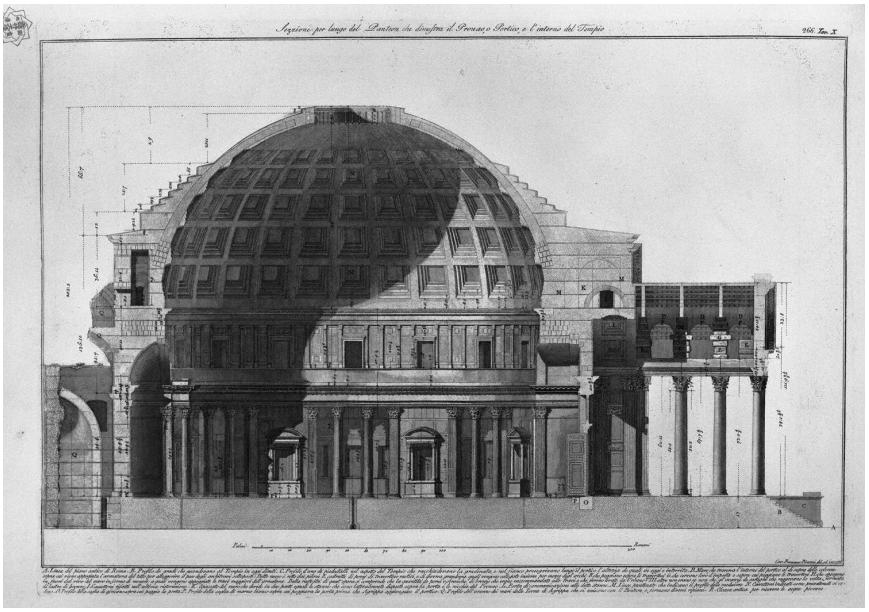
Aula 6

Geometria Construtiva de Sólidos

6.1 Introdução

O Panteão é um templo dedicado aos antigos deuses de Roma que foi construído pelo consul Marcus Agrippa no ano 80 e reconstruído pelo imperador Adriano no ano 126.

A seguinte imagem, de Giovanni Piranesi, ilustra uma secção do Panteão:



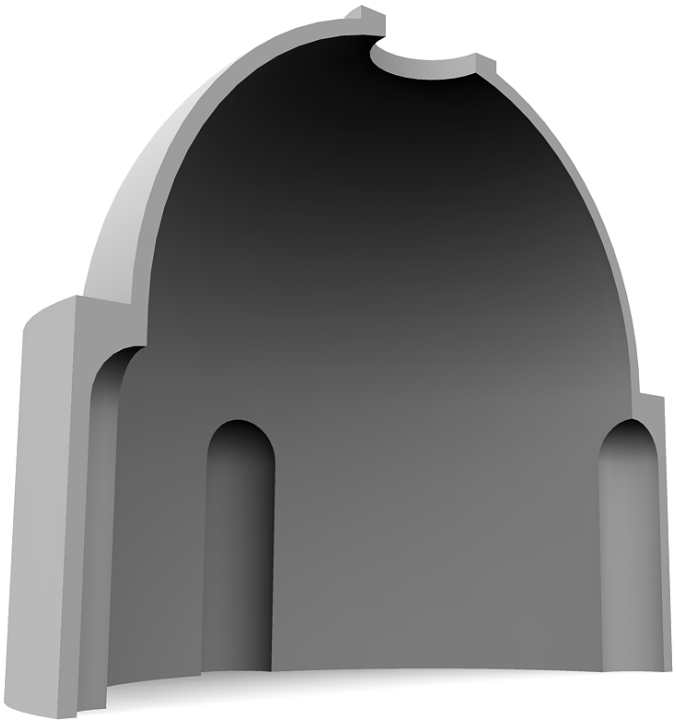
Pretende-se que crie um modelo do Panteão a partir das operações de geometria construtiva de sólidos.

6.2 Exercícios

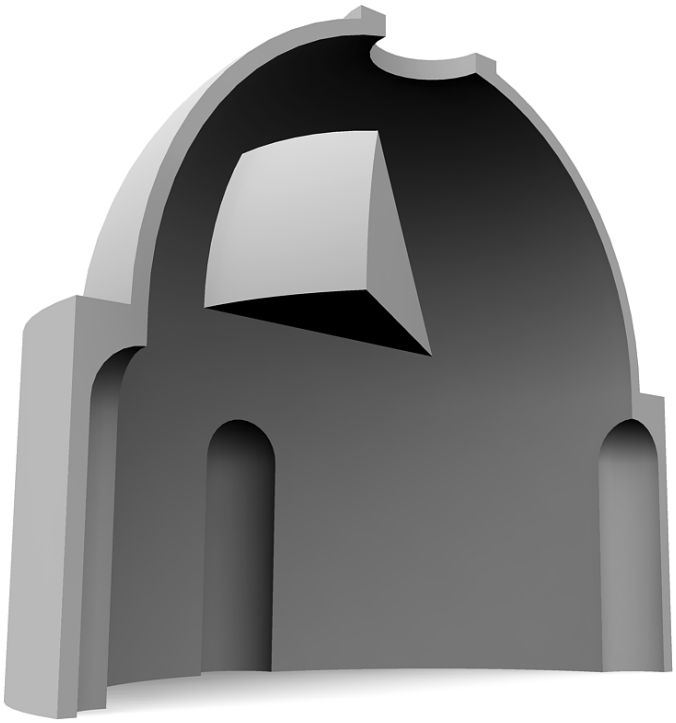
Exercício 6.2.1 Comece por criar um modelo paramétrico do corpo principal do edifício, composto por uma casca cilíndrica encimada por uma casca semi-esférica. No caso do Panteão, a altura do cilindro deverá ser igual ao raio da semi-esfera.

Exercício 6.2.2 Para modelar o óculo poderá empregar cilíndros, quer para perfurar a cúpula, quer para modelar o anel que suporta as cargas compressivas.

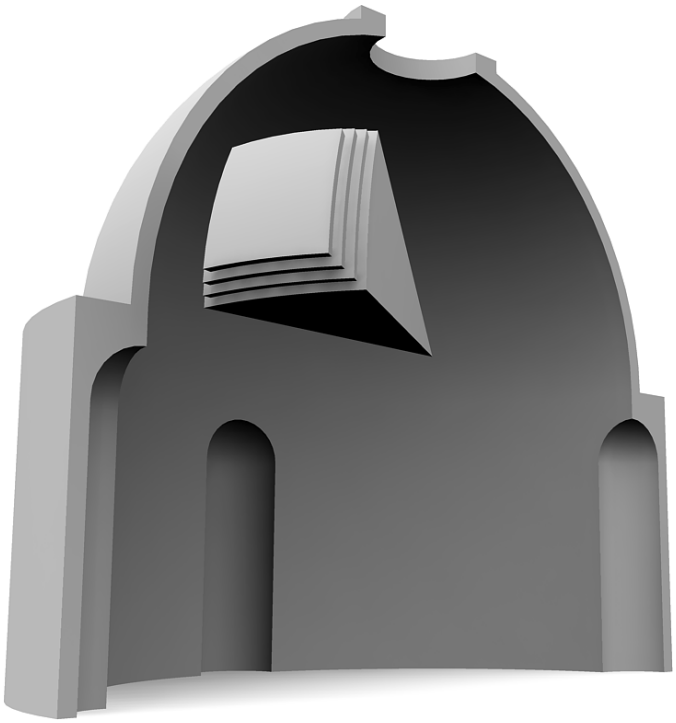
Exercício 6.2.3 Para modelar os nichos, assuma que estes têm a forma do próprio panteão, i.e., cilíndricos com uma semi-esfera no topo. Estes deverão ser distribuídos uniformemente ao longo da periferia.



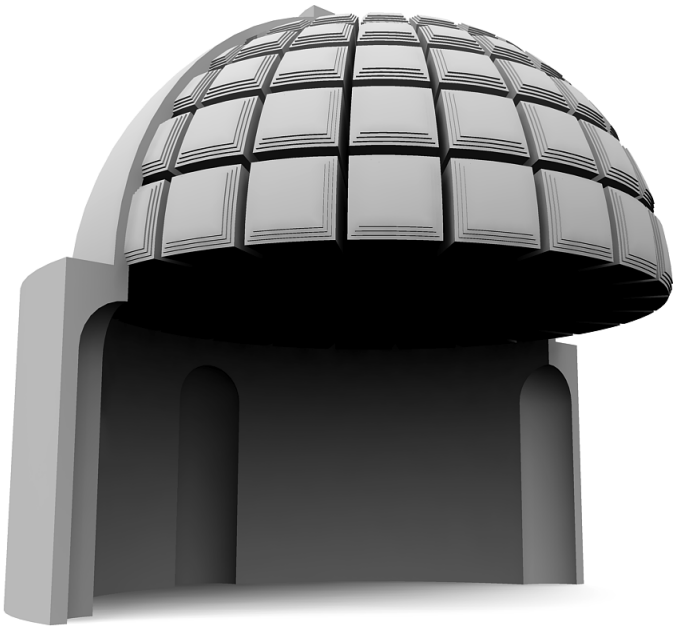
Exercício 6.2.4 Para modelar os caixotões da cúpula, é preferível começar por criar um molde que irá subtrair à cúpula, molde esse que deverá ser uma cunha esférica.

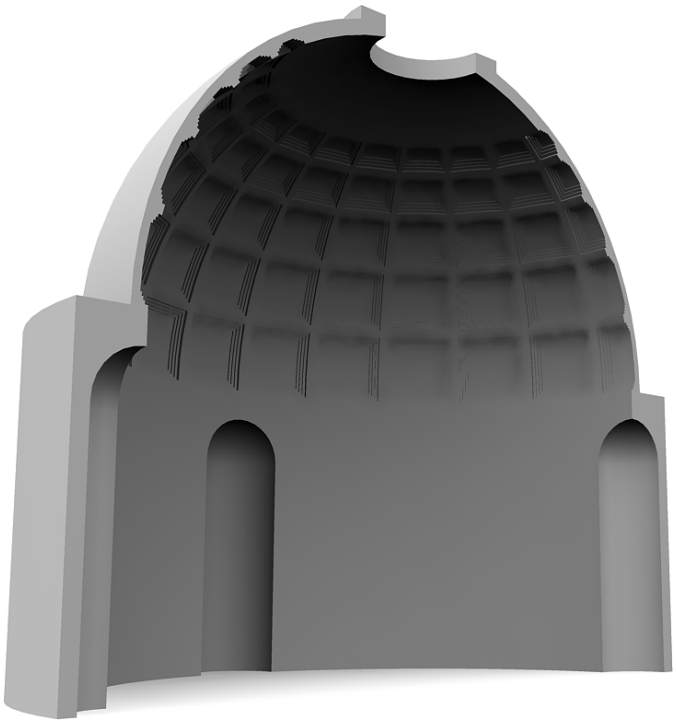


Exercício 6.2.5 Para modelar os degraus de um caixotão poderá empregar várias cunhas esféricas, com raios crescentes e dimensões decrescentes.



Exercício 6.2.6 Para modelar o conjunto dos caixotões deverá gerar caixotões ao longo dos meridianos e dos paralelos, subtraíndo o conjunto à cúpula.





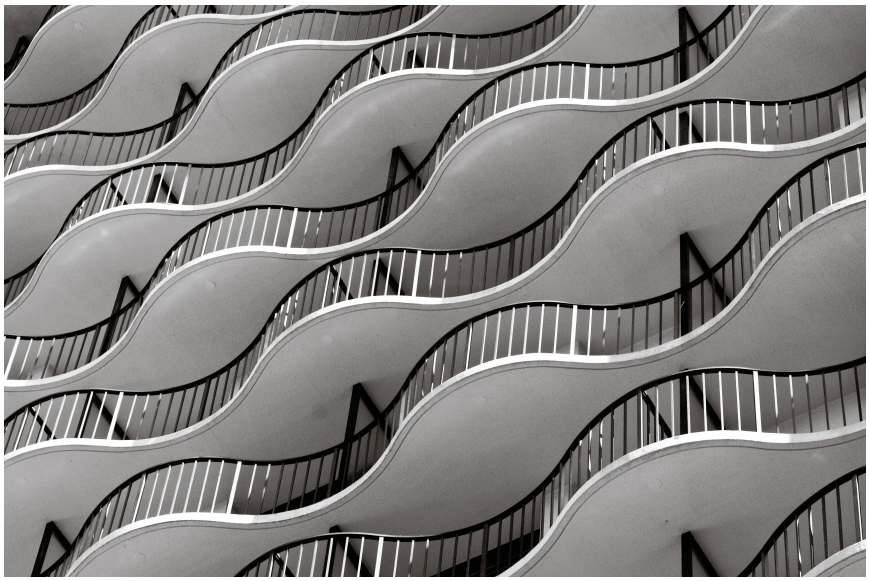
Aula 7

Extrusões

7.1 Introdução

A sinusóide é uma curva muito utilizada em arquitectura moderna. Por exemplo, o Kunst- und Ausstellungshalle, desenhado pelo arquitecto Vienense Gustav Peichl e destinado a ser um centro de exposições e de comunicações, é um edifício de forma quadrada mas em que um dos "cantos" deste quadrado é cortado por uma sinusóide, representada na imagem seguinte.

Na imagem seguinte apresentamos o Hotel Marriott em Anaheim como outro exemplo da utilização de sinusóides.



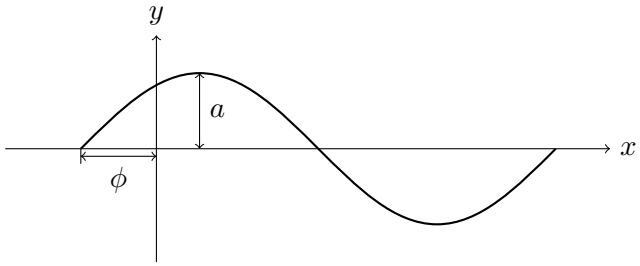
É fácil de ver que as varandas da fachada do hotel são sinusóides de igual amplitude e frequência mas que, entre cada dois pisos, apresentam fases opostas.

7.2 Funções Auxiliares

Para os exercícios que se seguem, considere as seguintes definições:

7.2.1 A Função Sinusóide

Consideremos a curva sinusóide tal como se apresenta no esquema seguinte, com amplitude a, frequência ω e fase ϕ:



A seguinte definição implementa a função sinusóide:

```
sinusoide(a, omega, fi, x) =
  a*sin(omega*x + fi)
```

7.2.2 Pontos de uma Sinusóide

Considere a seguinte função que constrói uma lista de pontos da função sinusóide num determinado intervalo $[x_0, x_1]$, com um determinado espaçamento entre coordenadas dx:

```
pontos_sinusoide(p, a, omega, fi, x0, x1, dx) =
  if x0 > x1
    []
  else
    [p + vy(sinusoide(a, omega, fi, x0)),
     pontos_sinusoide(p + vx(dx), a, omega, fi, x0 + dx, x1, dx)...]
  end
```

Note-se que os pontos gerados estão todos assentes no plano XY, com a sinusóide a evoluir ao longo do eixo X a uma determinada cota z determinada pelo ponto p.

7.3 Exercícios

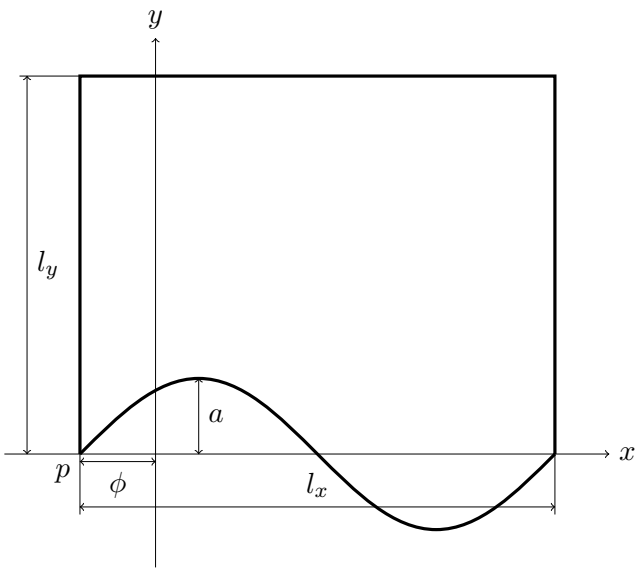
Pretende-se que crie um programa capaz de gerar a fachada do Hotel Marriott de Anaheim. O seu programa deverá ser suficientemente parameterizável para que se possam experimentar variações na fachada.

Exercício 7.3.1 Crie uma função denominada `laje` que, recebendo todos os parâmetros necessários para a geração de uma sinusóide, cria uma laje rectangular mas com uma das arestas de forma sinusoidal, tal como se apresenta na seguinte imagem:

A função deverá ter os seguinte parâmetros:

```
laje(p, a, omega, fi, lx, dx, ly, lz)
```

que estão explicados na seguinte figura:



O parâmetro `dx` representa o incremento Δ_x a considerar para a geração de pontos da sinusóide. O último parâmetro `lz` representa a espessura da laje.

A título de exemplo, considere a laje representada anteriormente como tendo sido gerada pela invocação

```
laje(xy(0, 0), 1.0, 1.0, 0, 6.5, 0.1, 2, 0.2)
```

Exercício 7.3.2 Crie uma função denominada `corrimao` que, recebendo todos os parâmetros necessários para a geração de uma sinusóide, cria um corrimão de secção rectangular mas ao longo de uma curva sinusoidal, tal como se apresenta na seguinte imagem:

A função deverá ter os seguinte parâmetros:

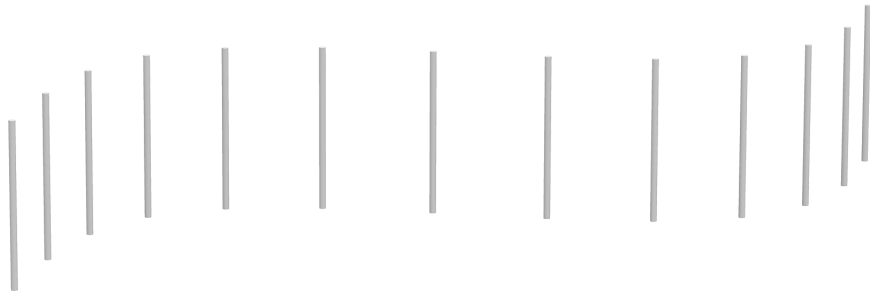
```
corrimao(p, a, omega, fi, lx, dx, l_corrimao, a_corrimao) =
  ...
```

Os primeiros parâmetros são os necessários para especificar completamente a curva sinusóide. Os dois últimos `l_corrimao` e `a_corrimao` correspondem à largura e altura da secção quadrada do corrimão.

Por exemplo, a imagem anterior poderia ser gerada pela invocação

```
corrimao(xy(0, 0), 1.0, 1.0, 0, 6.5, 0.1, 0.06, 0.02)
```

Exercício 7.3.3 Crie uma função denominada `prumos` que, recebendo todos os parâmetros necessários para a geração de uma sinusóide, cria os prumos de apoio a um corrimão de curva sinusoidal, tal como se apresenta na seguinte imagem:



Note que os prumos são de secção circular e, consequentemente, correspondem a cilindros com uma determinada altura e um determinado raio. Note também que os prumos possuem um determinado afastamento horizontal `dx`.

A função deverá ter os seguinte parâmetros:

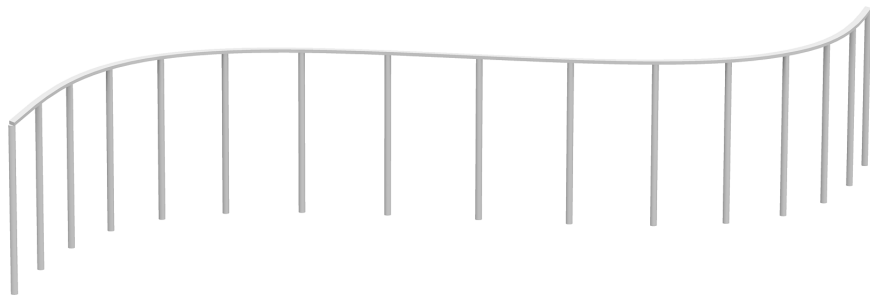
```
prumos(p, a, omega, fi, lx, dx, altura, raio) =
  ...
```

Os primeiros parâmetros são os necessários para especificar completamente a curva sinusóide. Os dois últimos `altura` e `raio` correspondem à altura e raio de cada cilindro.

Por exemplo, a imagem anterior poderia ser gerada pela invocação

```
prumos(xy(0, 0), 1.0, 1.0, 0, 6.5, 0.5, 1, 0.02)
```

Exercício 7.3.4 Crie uma função denominada `guarda` que, recebendo todos os parâmetros necessários para a criação do corrimão e dos prumos, cria uma guarda, tal como se apresenta na imagem seguinte:



Para simplificar, considere que os prumos deverão ter como diâmetro exactamente a mesma largura que o corrimão.

A função deverá ter os seguinte parâmetros:

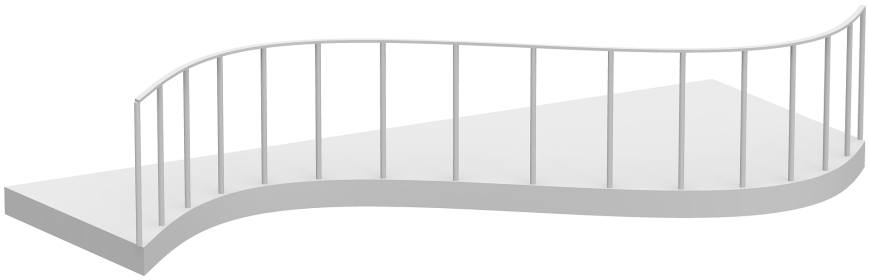
```
guarda(p, a, omega, fi, lx, dx,
       a_guarda, l_corrimao, a_corrimao, d_prumos) =
  ...
```

Os primeiros parâmetros são os necessários para especificar completamente a curva sinusóide. Os parâmetros `a_guarda`, `l_corrimao`, `a_corrimao` e `d_prumos` são, respectivamente, a altura da guarda, a largura e altura da secção quadrada do corrimão e a separação horizontal entre os prumos.

Por exemplo, a imagem anterior poderia ser gerada pela invocação

```
guarda(xy(0, 0), 1.0, 1.0, 0, 6.5, 0.5, 1, 0.06, 0.02, 0.4)
```

Exercício 7.3.5 Crie uma função denominada `piso` que, recebendo todos os parâmetros necessários para a criação da laje e da guarda, cria um piso, tal como se apresenta na imagem seguinte:



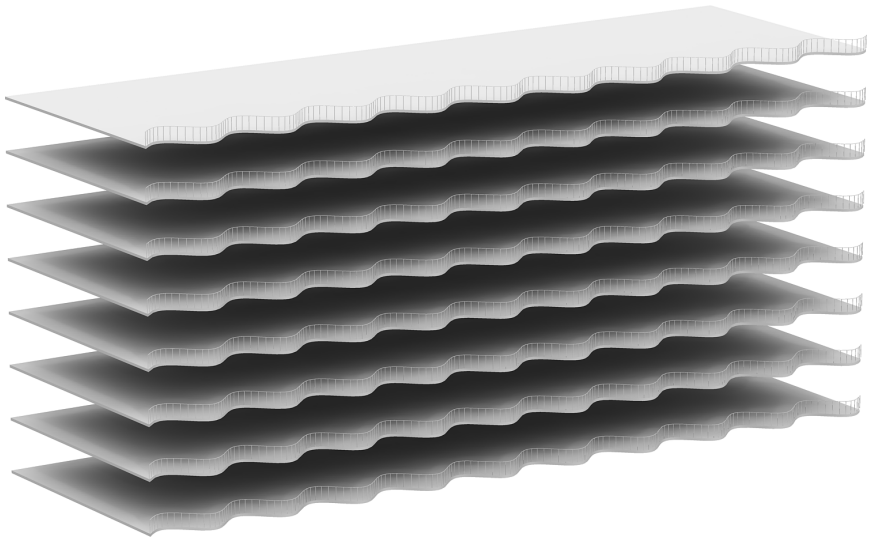
Para simplificar, considere que a guarda fica assente na extremidade da laje. A função `piso` deverá ter os seguinte parâmetros:

```
piso(p, a, omega, fi, lx, dx, ly,
     a_laje, a_guarda, l_corrimao, a_corrimao, d_prumos) =
  ...
```

Por exemplo, a imagem anterior poderia ser gerada pela invocação:

```
piso(xy(0, 0), 1.0, 1.0, 0, 6.5, 0.1, 2, 0.2, 1, 0.06, 0.02, 0.4)
```

Exercício 7.3.6 Crie uma função denominada `predio` que recebe todos os parâmetros necessários para a criação de um andar, incluindo a altura de cada andar `a_andar`, e o número de andares `n_andares`, e cria um prédio com esses andares, tal como se apresenta na imagem seguinte:



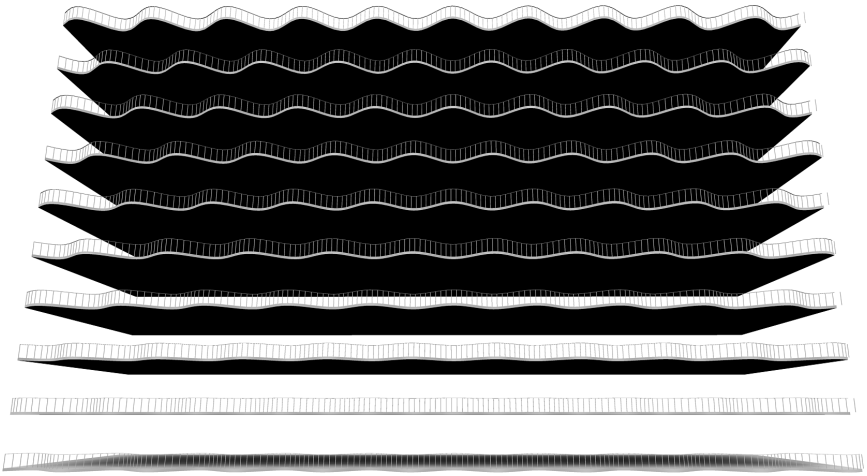
A função `predio` deverá ter os seguinte parâmetros:

```
predio(p, a, omega, fi, lx, dx, ly,
       a_laje, a_guarda,
       l_corrimao, a_corrimao,
       d_prumos, a_andar, n_andares) =
  ...
```

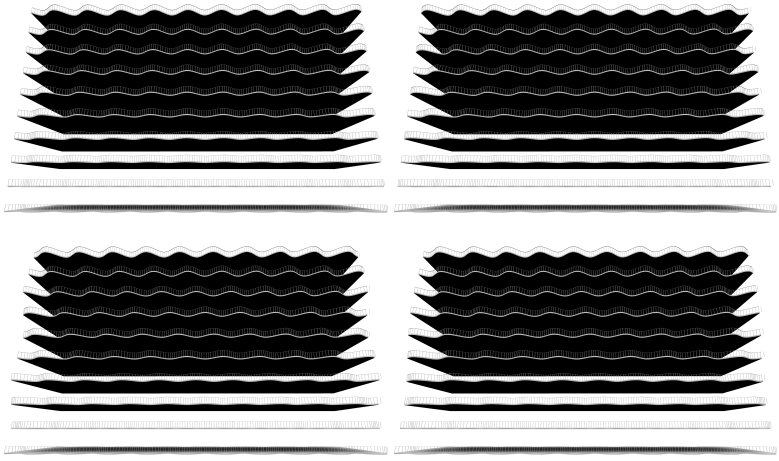
Por exemplo, a imagem anterior poderia ser gerada pela invocação:

```
predio(xy(0, 0), 1.0, 1.0, 0, 60, 0.5, 20, 0.2, 1, 0.06, 0.02, 0.4, 4, 10)
```

Exercício 7.3.7 Modifique a função `predio` de modo a receber um parâmetro adicional que representa um incremento de fase a considerar em cada andar. Através desse parâmetro, é possível gerar edifícios com fachadas mais interessantes, em que cada piso tem uma forma sinusoidal diferente. Por exemplo, compare o seguinte edifício com o anterior:



Exercício 7.3.8 Experimente variações nos parâmetros da sua função `predio` de modo a gerar diferentes fachadas. As imagens seguintes apresentam a visualização de algumas variações em torno do incremento de fase. Tente identificar qual foi o incremento empregue em cada imagem.



Aula 8

Superfícies e Sólidos de Revolução

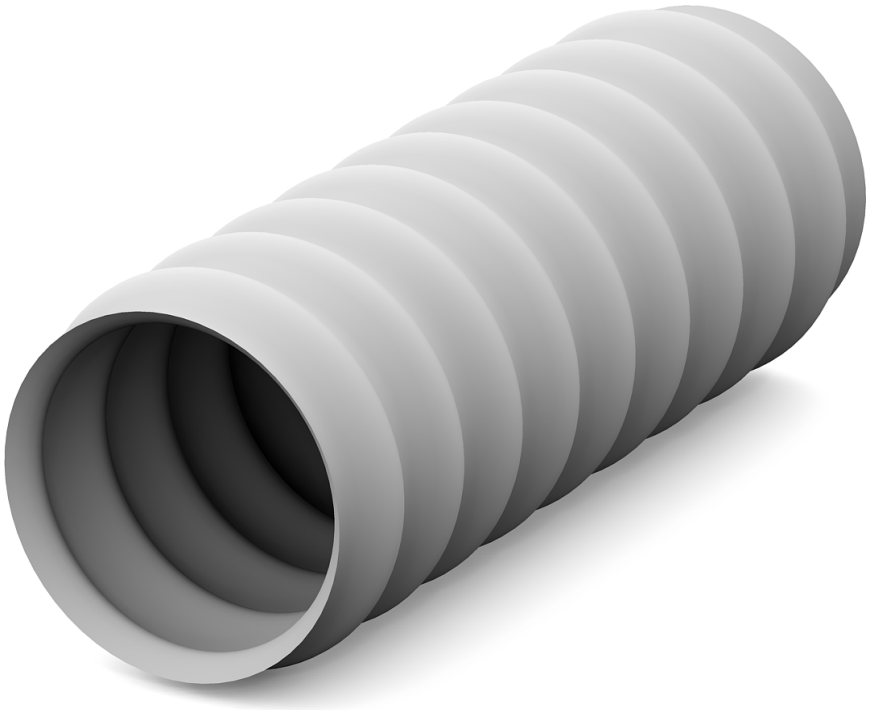
8.1 Introdução

Uma superfície de revolução é uma superfície gerada pela rotação de uma curva bidimensional em torno de um eixo. Um sólido de revolução é um sólido gerado pela rotação de uma região bidimensional em torno de um eixo.

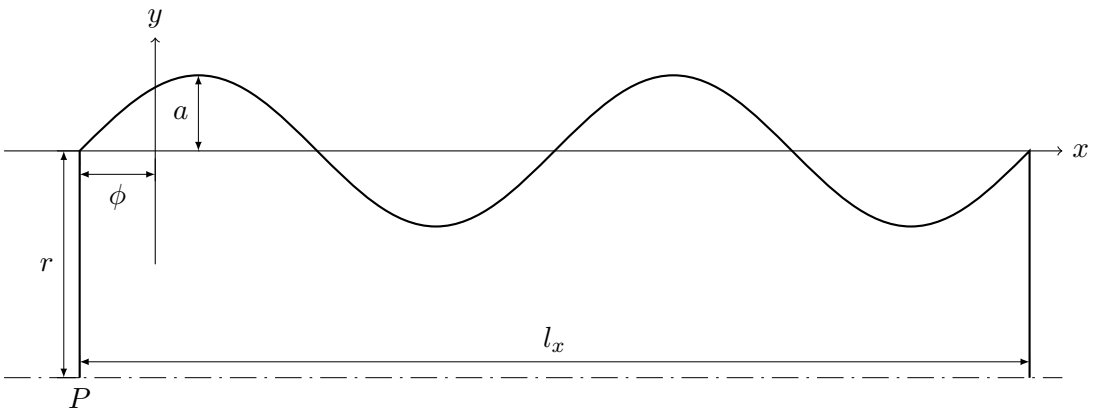
Sendo a rotação um processo muito simples de criação de superfícies e sólidos, é natural que exista uma operação para o fazer. A função `revolve` destina-se precisamente a esse fim. Esta função recebe, como argumentos, a região a "revolver" e, opcionalmente, um primeiro ponto no eixo de rotação (por omissão, a origem), um vector que descreve a direcção do eixo de rotação (por omissão, um vector paralelo ao eixo Z), o ângulo inicial de revolução (por omissão, zero) e a amplitude da revolução (por omissão, 2π). Naturalmente, se se omitir a amplitude ou este for de 2π radianos, obtém-se uma revolução completa.

8.2 Exercícios

Exercício 8.2.1 Considere o tubo de perfil sinusoidal apresentado na imagem seguinte.



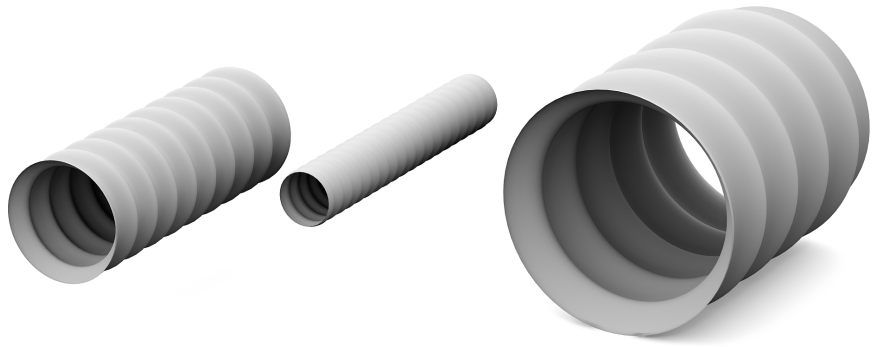
O tubo foi produzido tendo em conta os parâmetros geométricos descritos no perfil apresentado em seguida:



Defina a função `tubo_sinusoidal` que, a partir dos parâmetros anteriores P, r, a, $omega$, fi, lx e, finalmente, da separação entre pontos de interpolação Δ_x, gera o tubo sinusoidal pretendido. Por exemplo, os tubos apresentados na imagem seguinte foram gerados pela avaliação das

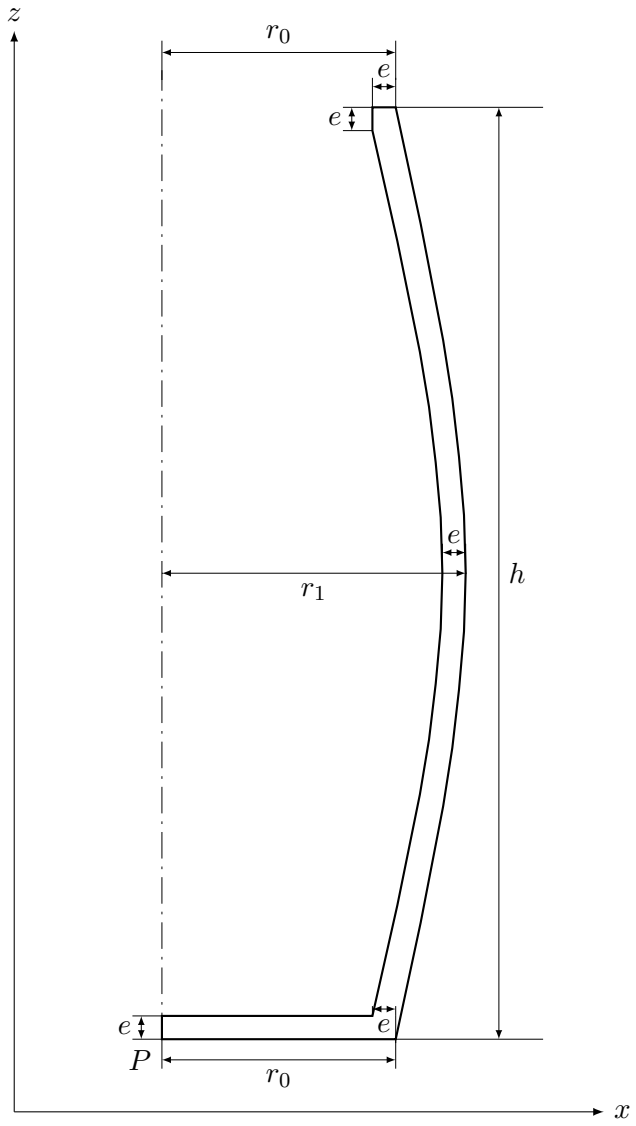
seguintes expressões:

```
tubo_sinusoidal(xyz(-20, 80, 0), 20, 2.0, 0.5, 0,    60, 0.2)
tubo_sinusoidal(xyz(0, 30, 0),    5, 0.2, 2.0, 0,    60, 0.1)
tubo_sinusoidal(xyz(30, 0, 0),   10, 1.0, 1.0, pi/2, 60, 0.2)
```



Exercício 8.2.2 Pretende-se criar um programa capaz de gerar um barril com a base assente no plano XY. Considere que a base do barril está fechada mas o topo está aberto.

Crie uma função denominada `perfil_barril` que recebe o ponto P, os raios r_0 e r_1, a altura h e a espessura e e que devolve a região que define o perfil do barril, tal como se apresenta na imagem seguinte:

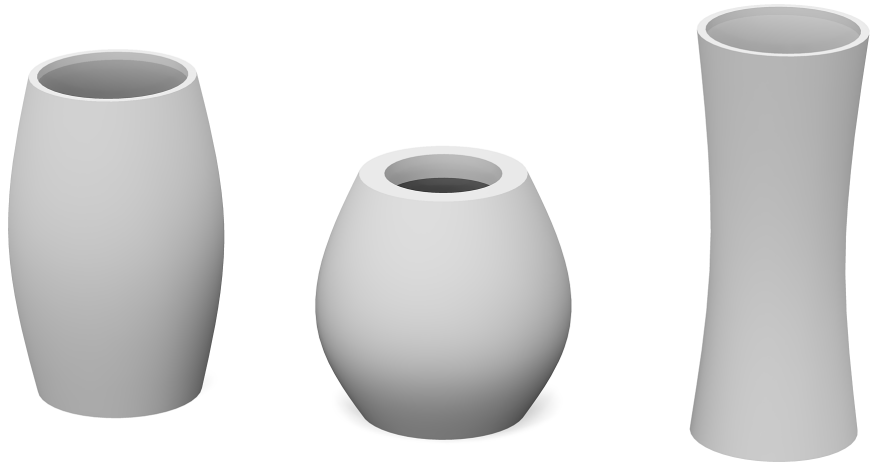


Exercício 8.2.3 Usando a função `perfil_barril`, defina a função `barril` que, tendo os mesmos parâmetros da anterior, cria o barril tridimensional. A função deverá ter a seguinte forma:

```
barril(p, r0, r1, h, e)
```

A título de exemplo, considere que a imagem seguinte foi gerada pela avaliação das expressões:

```
barril(xyz(0, 0, 0), 1, 1.3, 4, 0.1)
barril(xyz(4, 0, 0), 1, 1.5, 3, 0.3)
barril(xyz(8, 0, 0), 1, 0.8, 5, 0.1)
```

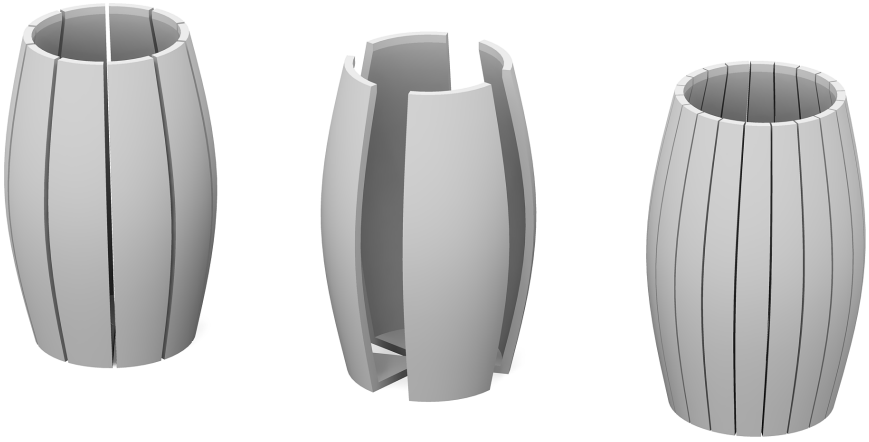


Exercício 8.2.4 O barril criado no exercício anterior é excessivamente "moderno," não representando os barris tradicionais de madeira que são constituídos por múltiplas tábuas encostadas umas às outras. A imagem seguinte mostra alguns exemplos destas tábuas, com dimensões e posicionamentos angulares diferentes:

Defina a função `tabua_barril` que, para além dos parâmetros que definem um barril, dado o ângulo de rotação inicial α a que se inicia a tábua e dado o incremento de ângulo Δ_α que corresponde à âmplitude angular da tábua, constrói a secção tridimensional do barril correspondente à tábua em questão.

Exercício 8.2.5 Defina uma função denominada `tabuas_barril` que recebe como parâmetros o ponto P, os raios r_0 e r_1, a altura h, a espessura e, o número de tábuas n e a "folga" angular entre tábuas s e que cria um barril tridimensional com esse número de tábuas. A imagem seguinte mostra alguns exemplos destes barris, criados pela avaliação das seguintes expressões:

```
tabuas_barril(xyz(0, 0, 0), 1, 1.3, 4, 0.1, 10, 0.05)
tabuas_barril(xyz(4, 0, 0), 1, 1.3, 4, 0.1, 4, 0.5)
tabuas_barril(xyz(8, 0, 0), 1, 1.3, 4, 0.1, 20, 0.01)
```



Aula 9

Curvas Paramétricas

9.1 A Curva de Lamé

A curva de Lamé é uma curva cujos pontos satisfazem a equação

$$\left|\frac{x}{a}\right|^n + \left|\frac{y}{b}\right|^n = 1$$

em que $n > 0$ e a e b são os raios da curva.

A curva foi estudada no século dezanove pelo matemático Francês Gabriel Lamé como uma generalização óbvia da elipse. Quando n é um número racional, a curva de Lamé diz-se uma *superelipse*. Quando $n > 2$, obtemos uma hiperelipse, tanto mais próxima de um rectângulo quanto maior for n. Quando $n < 2$ obtemos uma hipoelipse, tanto mais próxima de uma cruz quanto menor for n. Para $n = 2$, obtemos uma elipse e, para $n = 1$, obtemos um losango. Estas variações são visiveis na Figura 9.1.

A curva de Lamé tornou-se famosa quando foi proposta pelo cientista e poeta Dinamarquês Piet Hein como um compromisso estético e funcional entre formas baseadas em padrões rectangulares e formas baseadas em padrões curvilineos. No seu estudo para uma intersecção de ruas no bairro Sergels Torg, em Estocolmo, em que se estava a hesitar entre uma rotunda tradicional ou um arranjo rectangular de vias, Piet Hein sugeriu a superelipse como forma intermédia entre as duas alternativas, produzindo o resultado que está visível na Figura 9.2. De todas as superelipses, as mais esteticamente perfeitas eram, na opinião de Piet Hein, as parametrizadas por $n = \frac{5}{2}$ e $\frac{a}{b} = \frac{6}{5}$.

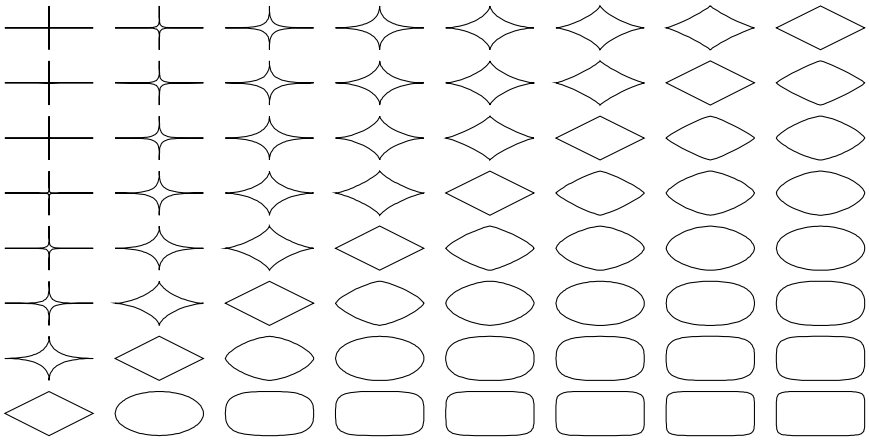


Figura 9.1: A curva de Lamé para $a = 2$, $b = 1$, $n = p/q$, $p, q \in \{1..8\}$. A variável p varia ao longo do eixo das abcissas, enquanto q varia ao longo do eixo das ordenadas.

9.2 Exercícios

Pretende-se que implemente um conjunto de funções capazes de construir a praça de Sergels Torg, em Estocolmo (excluíndo a estátua). Deverá ainda parametrizar convenientemente as suas funções de modo a permitir experimentar variações na forma da praça.

Exercício 9.2.1 Considere a seguinte formulação paramétrica para uma superelipse centrada na origem:

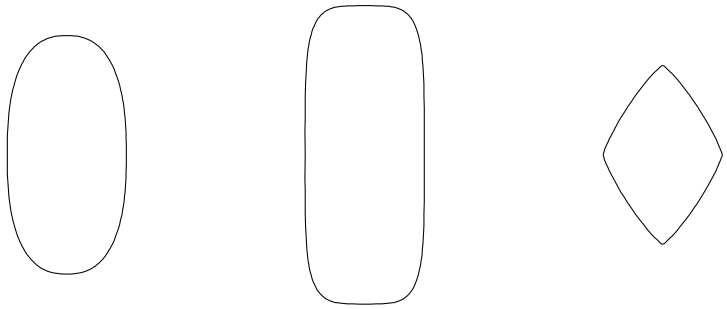
$$\begin{cases} x(t) = a\left(\cos^2 t\right)^{\frac{1}{n}} \cdot \operatorname{sgn}\cos t \\ y(t) = b\left(\sin^2 t\right)^{\frac{1}{n}} \cdot \operatorname{sgn}\sin t \end{cases} \qquad -\pi \leq t < \pi$$

Defina uma função denominada `superelipse` que, dado um ponto p correspondente ao centro da superelipse, dados os parâmetros a, b e n da superelipse e dado o parâmetro t, computa o ponto da curva da superelipse correspondente a t.

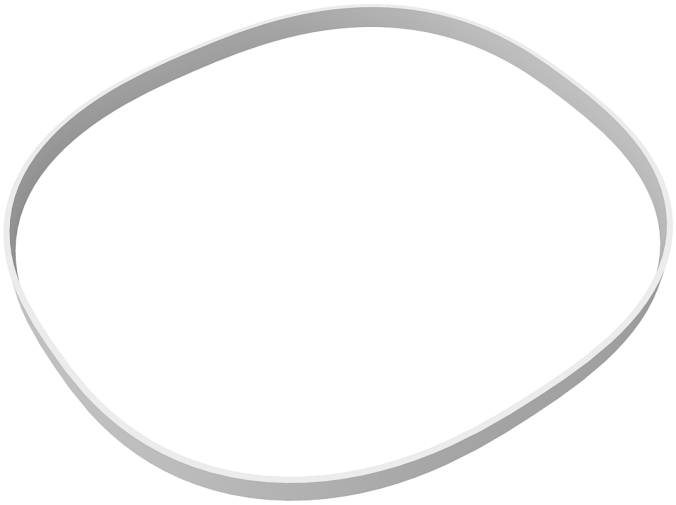
Exercício 9.2.2 Defina a função `curva_superelipse` que, dado um ponto p correspondente ao centro da superelipse, dados os parâmetros a, b e n da superelipse e dado um número de pontos a computar, constrói uma curva *spline* com a forma da superelipse.

Figura 9.2: A superelipse proposta por Piet Hein para a praça Sergels, em Estocolmo. Fotografia de Nozzman.

Exercício 9.2.3 Experimente a função anterior para gerar superelipses que se aproximem das seguintes curvas:

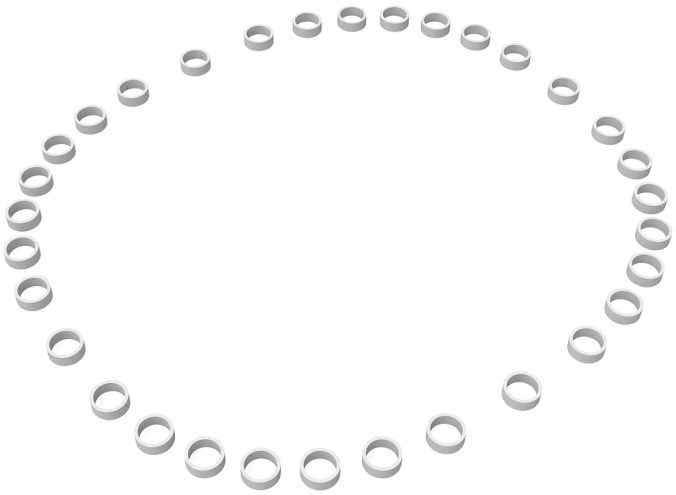


Exercício 9.2.4 Defina a função `tanque_superelipse` que constrói um tanque de forma superelíptica idêntico ao da praça de Sergels Torg, tal como se apresenta em seguida:



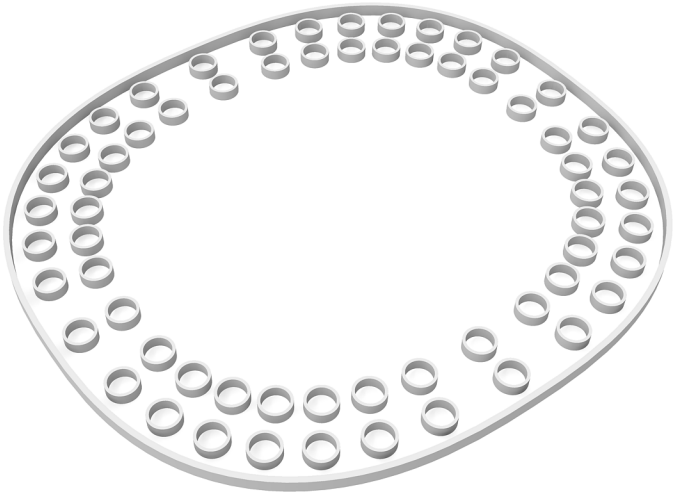
Sugestão: defina previamente uma função denominada `parede_curva` que constrói uma parede ao longo de uma determinada curva. A função deverá receber a espessura e altura da parede a construir e uma entidade representando a curva (por exemplo, um círculo, uma *spline*, etc).

Exercício 9.2.5 Defina a função `tanques_circulares` que constrói uma sucessão de tanques circulares cujos centros estão localizados ao longo de uma superelipse, tal como se apresenta na imagem seguinte:

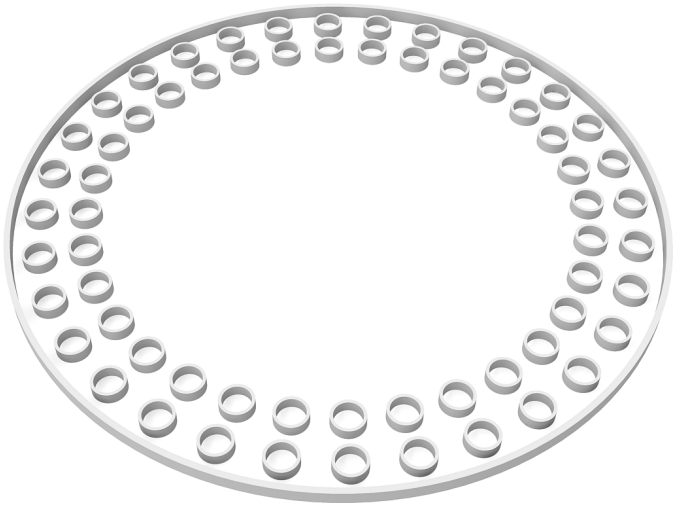


A função deverá receber os parâmetros da superelipse, os parâmetros de um tanque circular e o número de tanques a criar.

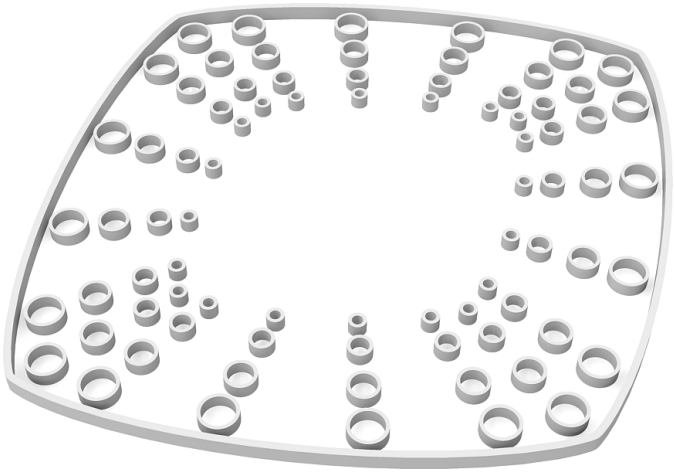
Exercício 9.2.6 Usando as funções anteriores (i.e., `tanque_superelipse` e `tanques_circulares`), defina uma função que cria uma praça tão semelhante quanto possível à praça de Sergels Torg, tal como se apresenta na imagem seguinte:



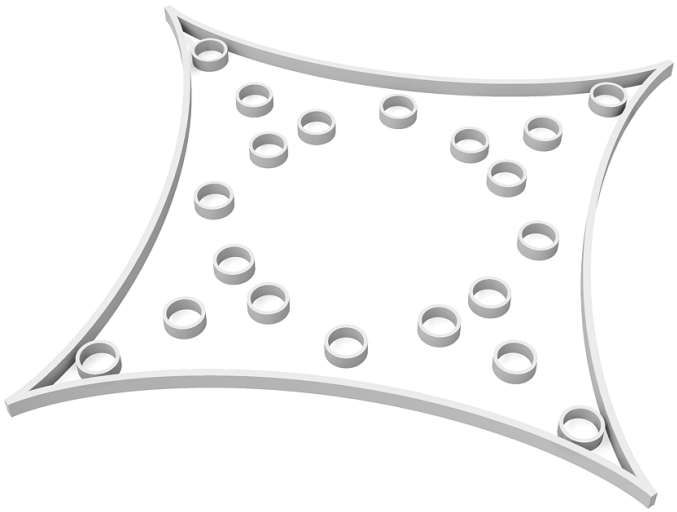
Exercício 9.2.7 Modifique a função anterior para produzir uma variante perfeitamente circular:



Exercício 9.2.8 Modifique a função anterior para produzir a seguinte variante:



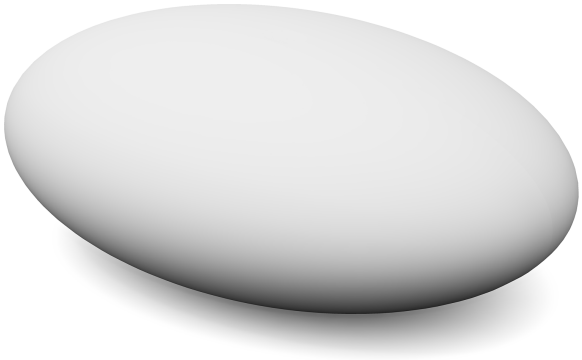
Exercício 9.2.9 Modifique as funções anteriores para produzir a seguinte variante:



Aula 10
Superfícies Paramétricas

10.1 Exercícios

Exercício 10.1.1 Considere o seguinte elipsóide:



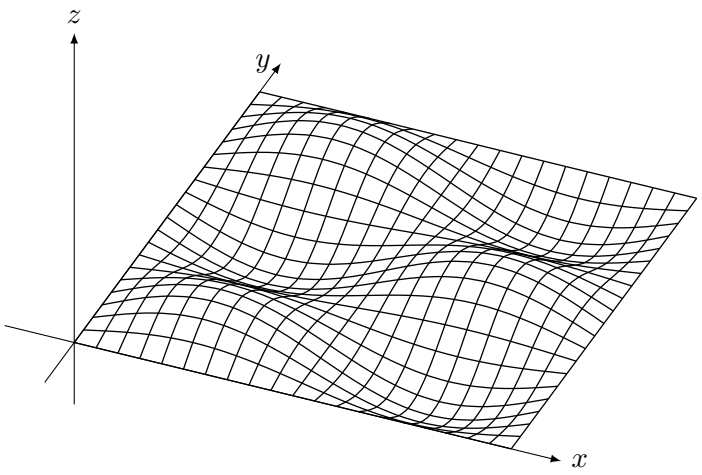
Um elipsóide é caracterizado pelas dimensões dos seus três raios ortogonais a, b, e c. A sua equação paramétrica é:

$$\begin{aligned} x &= a \sin\psi \cos\phi \\ y &= b \sin\psi \sin\phi \\ z &= c \cos\psi \end{aligned}$$

$$-\tfrac{\pi}{2} \leq \phi \leq +\tfrac{\pi}{2}; \quad -\pi \leq \psi \leq +\pi;$$

Defina a função `elipsoide` que produz o elipsóide a partir dos três raios a, b, c e ainda do número n de valores a usar ao longo de ϕ e de ψ.

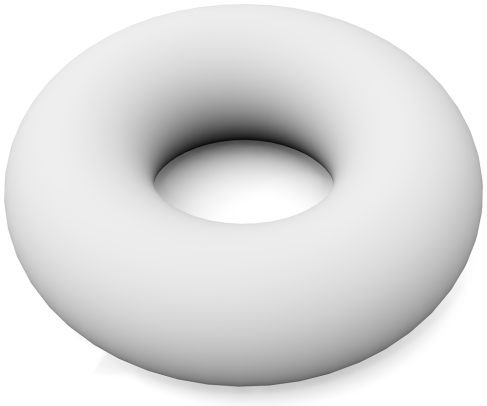
Exercício 10.1.2 Considere a seguinte superfície gerada por duas sinusóides que se desenvolvem em direcções ortogonais:



Sabendo que a superfície oscila entre a altura máxima de 0.5 e a mínima de -0.5 e que x e y ambos variam no intervalo $[0, 2\pi]$, complete a seguinte expressão de modo a criar as coordenadas da superfície anterior:

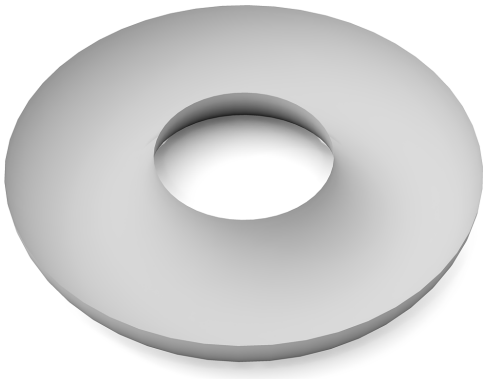
```
map_division((u, v) -> xyz(u, v, ?),
             0, 2*pi, 20,
             0, 2*pi, 20)
```

Exercício 10.1.3 A seguinte imagem descreve um toro:



Deduza a equação paramétrica do toro e defina a função Julia `toro` que cria um toro idêntico ao da imagem anterior a partir do centro do toro P, dos raios maior r_0 e menor r_1 e do número de intervalos m e n a considerar em cada dimensão.

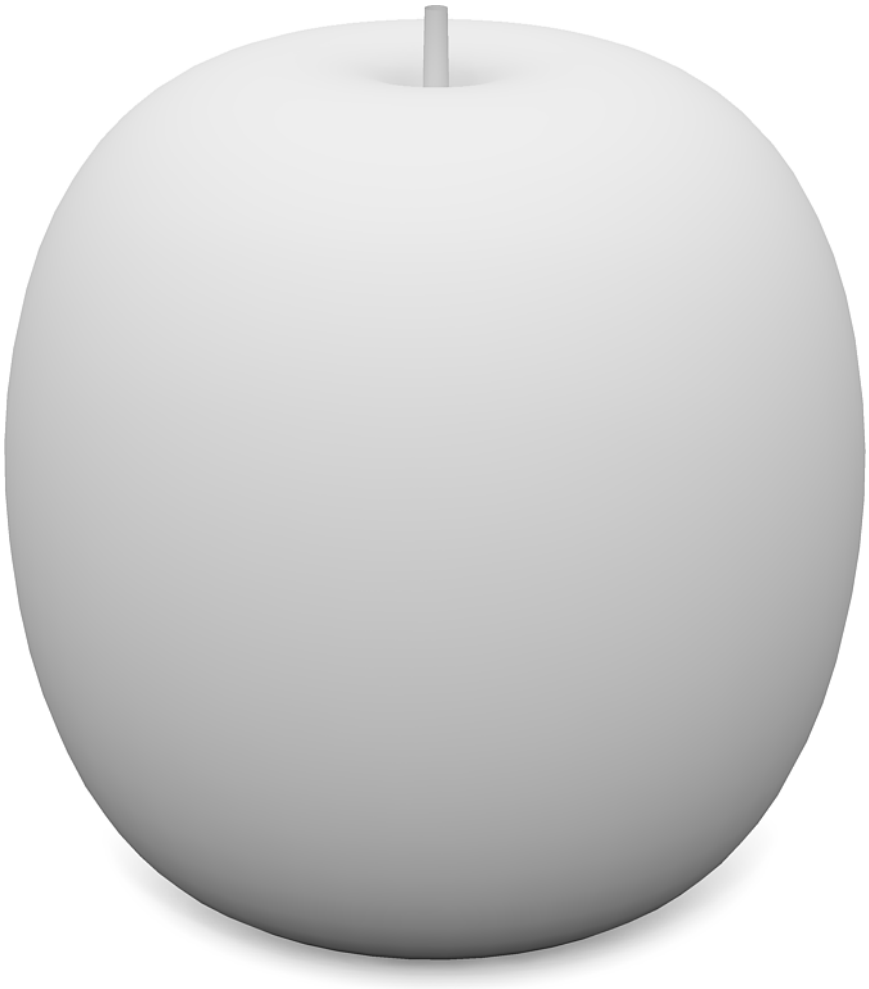
Exercício 10.1.4 Defina a função `meio_toro` que, a partir dos mesmos parâmetros da função `toro` cria uma metade de um toro idêntica à da imagem seguinte:



Exercício 10.1.5 A maçã apresentada abaixo pode ser descrita pela equação paramétrica

$$
\begin{aligned}
x &= \cos u \cdot (4 + 3.8 \cos v) \\
y &= \sin u \cdot (4 + 3.8 \cos v) \\
z &= (\cos v + \sin v - 1) \cdot (1 + \sin v) \cdot \log(1 - \pi \cdot \frac{v}{10}) + 7.5 \sin v
\end{aligned}
$$

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



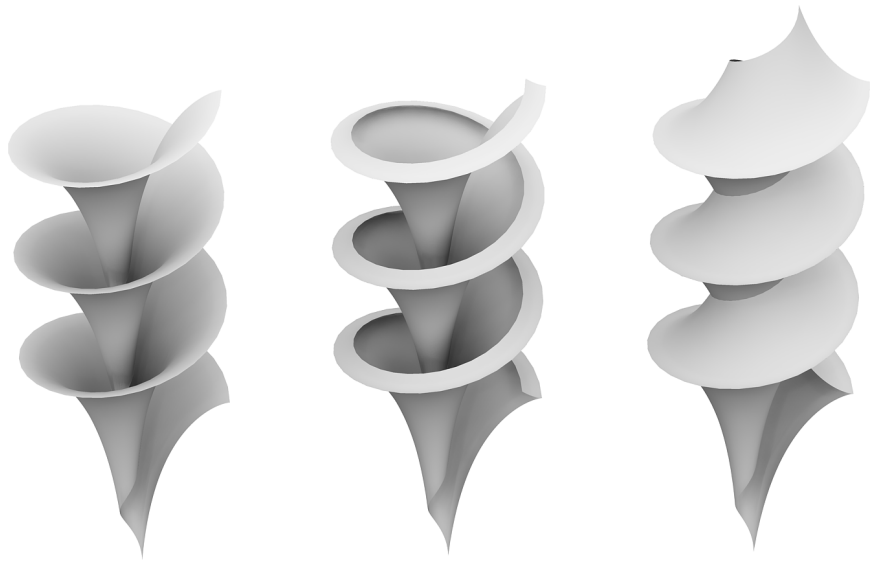
Escreva uma expressão Julia que reproduz a maçã anterior.

Exercício 10.1.6 As equações paramétricas que a definem a superfície de Dini são as seguintes:

$$\begin{cases} x(u,v) = \cos(u)\sin(v) \\ y(u,v) = \sin(u)\sin(v) \\ z(u,v) = \cos(v) + \log(\tan(\frac{v}{2})) + a * u \end{cases}$$

em que $0 \leq u$ e $0 < v$.

Diferentes valores dos parâmetros permitem gerar as seguintes superfícies:



Defina a función `dini` que, convenientemente parametrizada, permite gerar as imagens anteriores.

Aula 11

Processamento de Superfícies

11.1 Introdução

Por convenção, representamos as coordenadas de uma superfície como um *array* de *array* de posições. Se o objectivo não é gerar uma superfície mas sim processar os vários quadrângulos que constituem a superfície, então temos de operar sobre essas posições em grupos de quatro. A seguinte função de ordem superior realiza essa tarefa:

```
itera_quadrangulos(f, ptss) =
  [[f(p0, p1, p2, p3)
    for (p0, p1, p2, p3)
    in zip(pts0[1:end-1], pts1[1:end-1], pts1[2:end], pts0[2:end])]
   for (pts0, pts1)
   in zip(ptss[1:end-1], ptss[2:end])]
```

Para processar cada um dos quadrângulos pode ser útil determinar o seu centro e a sua normal. As funções `quad_center` e `quad_normal` fazem precisamente isso.

11.2 Exercícios

Exercício 11.2.1 Considere a superfície definida por:

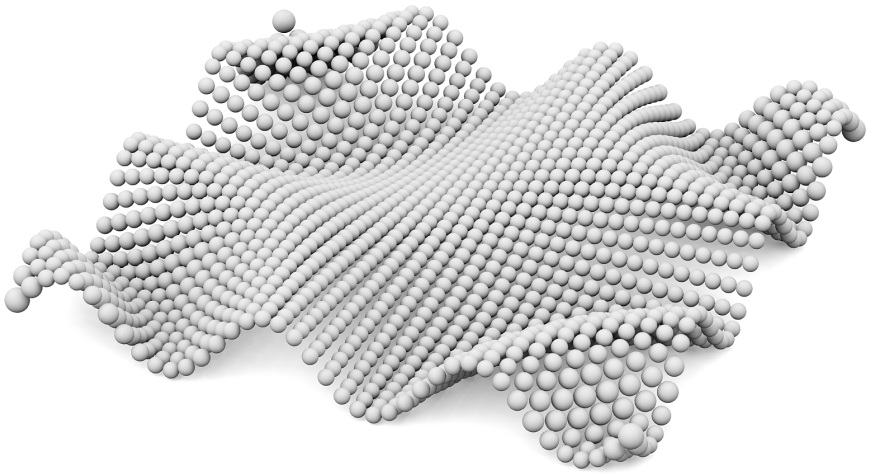
$$\begin{cases} x(u,v) = u \\ y(u,v) = v \\ z(u,v) = \dfrac{4}{10}\sin(u \cdot v) \end{cases}$$
$$-\pi \leq u \leq \pi$$

$$-\pi \leq v \leq \pi$$

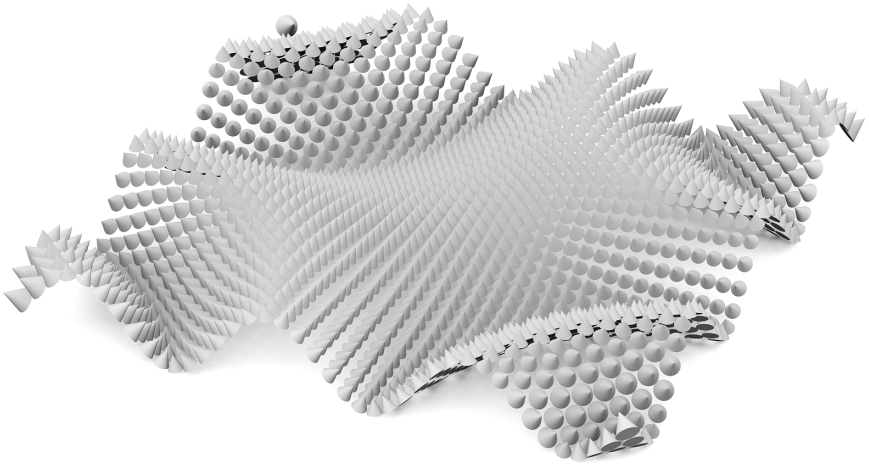
Escreva uma função `sin_u_mul_v` que, dado o número de intervalos n a considerar para cada dimensão, devolve as posições da superfície anterior. Visualize essa superfície.

Exercício 11.2.2 Usando a função `itera_quadrangulos`, defina a função `superficie_esferas` que, dada uma superfície representada por um *array* de *array*s de posições, gera uma união de esferas, cada uma posicionada no centro de cada quadrângulo e cujo raio é metade da menor distância entre os vértices do quadrângulo.

Use a função `superficie_esferas` para gerar a seguinte imagem:



Exercício 11.2.3 Usando a função `itera_quadrangulos`, defina a função `superficie_cones` que, dada uma superfície representada por um *array* de *array*s de posições, gera uma união de cones, semelhante à que se apresenta na seguinte imagem:

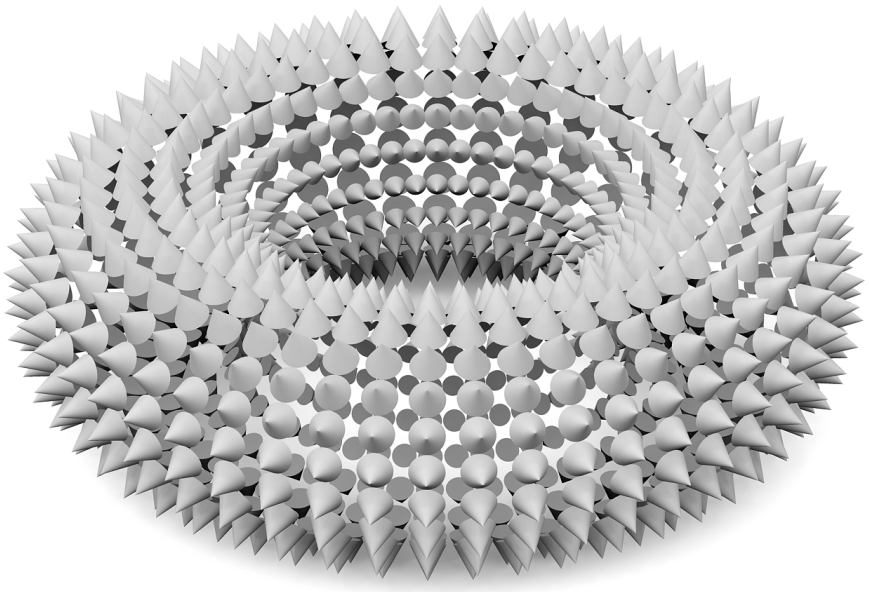


Exercício 11.2.4 O toro é definido parametricamente por:

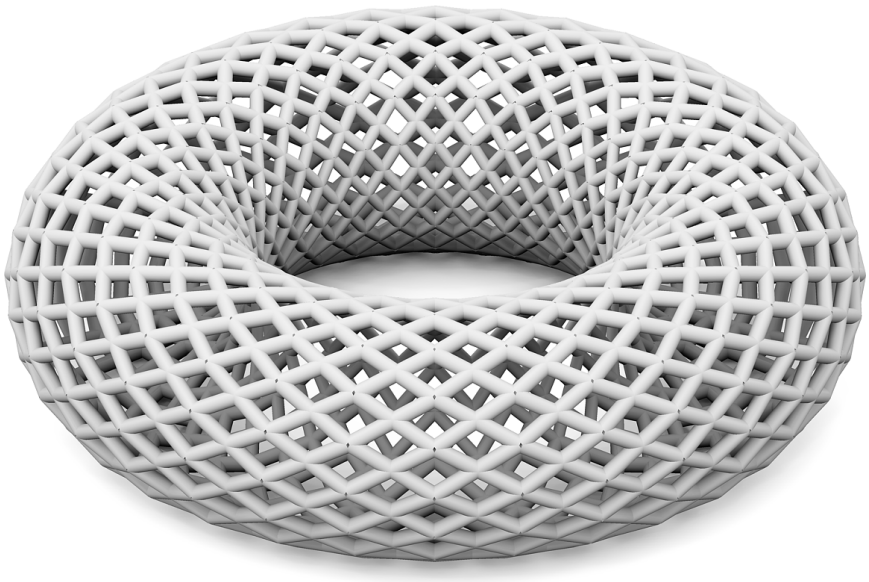
$$\begin{aligned} x &= \cos u(R + r\cos v) \\ y &= \sin u(R + r\cos v) \\ z &= r\sin v \end{aligned}$$

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

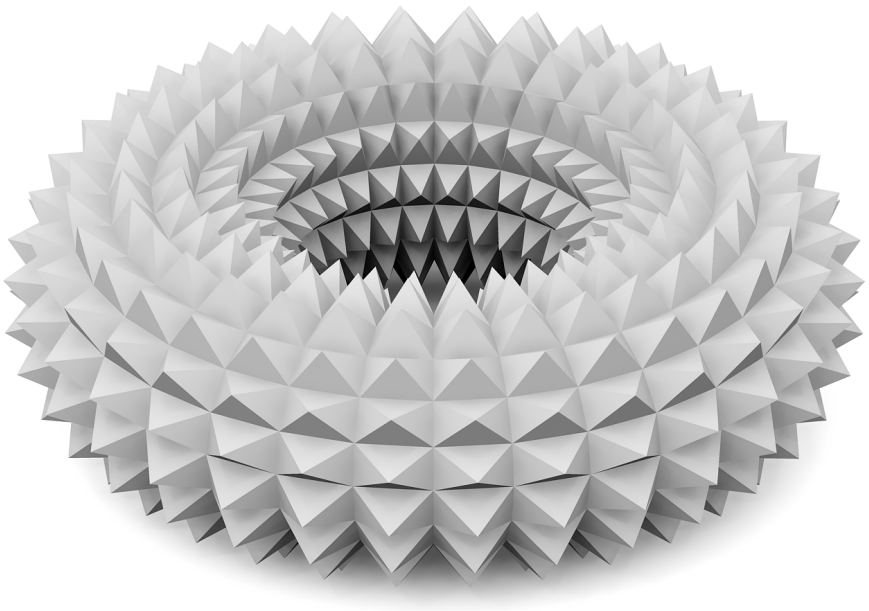
Crie um toro de cones semelhante ao que se apresenta em seguida:



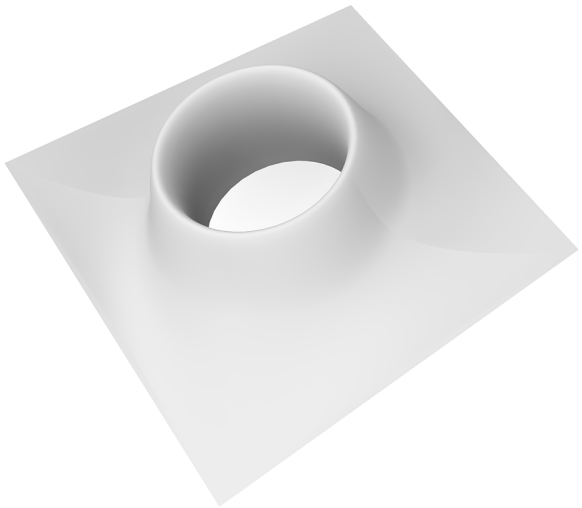
Exercício 11.2.5 Defina a função `malha` que, dada uma superfície representada por um *array* de *array*s de posições, gera uma forma semelhante à que se apresenta na seguinte imagem:



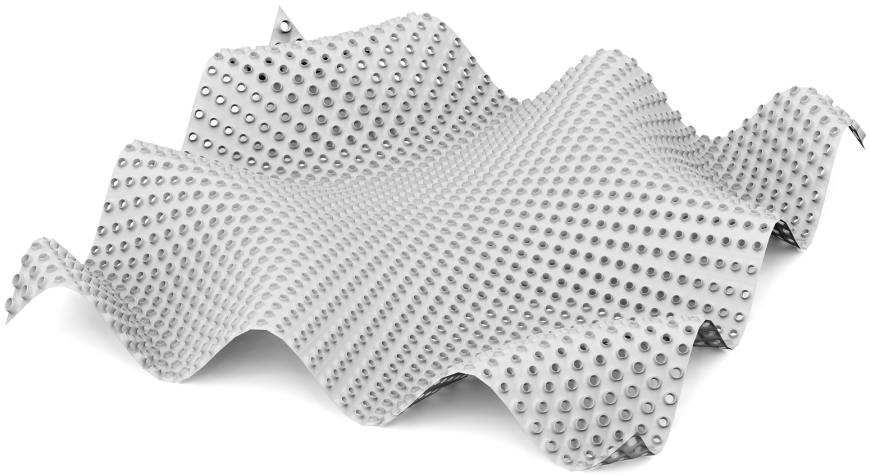
Exercício 11.2.6 Defina a função `superficie_piramides` que, dada uma superfície representada por um *array* de *array*s de posições, gera uma união de pirâmides, semelhante à que se apresenta na seguinte imagem:



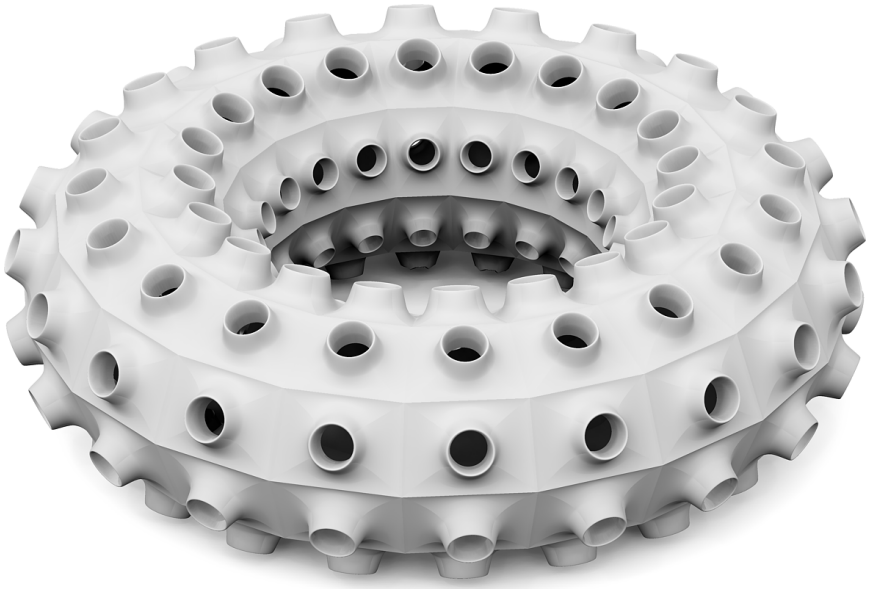
Exercício 11.2.7 Defina a função `oculo` que, dados os quatro vértices de um quadrângulo (em sentido anti-horário), cria uma forma semelhante à que se apresenta em seguida:



Exercício 11.2.8 Usando a função `oculo`, crie a seguinte forma:



Exercício 11.2.9 Usando a função `oculo`, crie a seguinte forma:



Exercício 11.2.10 Defina a função `superficie_cilindros` que, dada uma superfície representada por um *array* de *array*s de posições, gera uma união de cilindros verticais, cada um com o topo no centro de cada quadrângulo e com base na cota mínima da superfície, e cujo raio é metade da menor distância entre os pontos do quadrângulo. A seguinte imagem ilustra um exemplo: