Computer Science in Architecture

António Menezes Leitão
antonio.menezes.leitao@ist.utl.pt

December, 2011

Preamble

ἕν οἶδα ὅτι οὐδὲν οἶδα

Σωκράτης

Preamble

ἕν οἶδα ὅτι οὐδὲν οἶδα

Σωκράτης

I know one thing, that I know nothing

Socrates

Architecture and Computer Science

Et ut litteratus sit, peritus graphidos, eruditus geometria, historias complures noverit, musicam scierit, medicinae non sit ignarus

Marcus Vitruvius Pollio

Architecture and Computer Science

> *Et ut litteratus sit, peritus graphidos, eruditus geometria, historias complures noverit, musicam scierit, medicinae non sit ignarus*
>
> *Marcus Vitruvius Pollio*

An architect must be:

- a good writer

Architecture and Computer Science

> *Et ut litteratus sit, peritus graphidos, eruditus geometria, historias complures noverit, musicam scierit, medicinae non sit ignarus*
>
> *Marcus Vitruvius Pollio*

An architect must be:

- a good writer
- a skilful draftsman

Architecture and Computer Science

> *Et ut litteratus sit, peritus graphidos, eruditus geometria, historias complures noverit, musicam scierit, medicinae non sit ignarus*
>
> *Marcus Vitruvius Pollio*

An architect must be:

- a good writer
- a skilful draftsman
- versed in geometry

Architecture and Computer Science

> *Et ut litteratus sit, peritus graphidos, eruditus geometria, historias complures noverit, musicam scierit, medicinae non sit ignarus*
>
> *Marcus Vitruvius Pollio*

An architect must be:

- a good writer
- a skilful draftsman
- versed in geometry
- acquainted with history

Architecture and Computer Science

> *Et ut litteratus sit, peritus graphidos, eruditus geometria, historias complures noverit, musicam scierit, medicinae non sit ignarus*
>
> *Marcus Vitruvius Pollio*

An architect must be:

- a good writer
- a skilful draftsman
- versed in geometry
- acquainted with history
- a musician

Architecture and Computer Science

> *Et ut litteratus sit, peritus graphidos, eruditus geometria, historias complures noverit, musicam scierit, medicinae non sit ignarus*
>
> *Marcus Vitruvius Pollio*

An architect must be:

- a good writer
- a skilful draftsman
- versed in geometry
- acquainted with history
- a musician
- not ignorant of the sciences

Architecture and Computer Science

> *Et ut litteratus sit, peritus graphidos, eruditus geometria, historias complures noverit, musicam scierit, medicinae non sit ignarus, eruditus computare*
>
> *Marcus Vitruvius Pollio*

An architect must be:

- a good writer
- a skilful draftsman
- versed in geometry
- acquainted with history
- a musician
- not ignorant of the sciences
- versed in computation

Architecture needs Computer Science

Architecture needs Computer Science

How do we teach Computer Science to Architects?

Architecture needs Computer Science

How do we teach Computer Science to Architects?

- Geometry

Architecture needs Computer Science

How do we teach Computer Science to Architects?

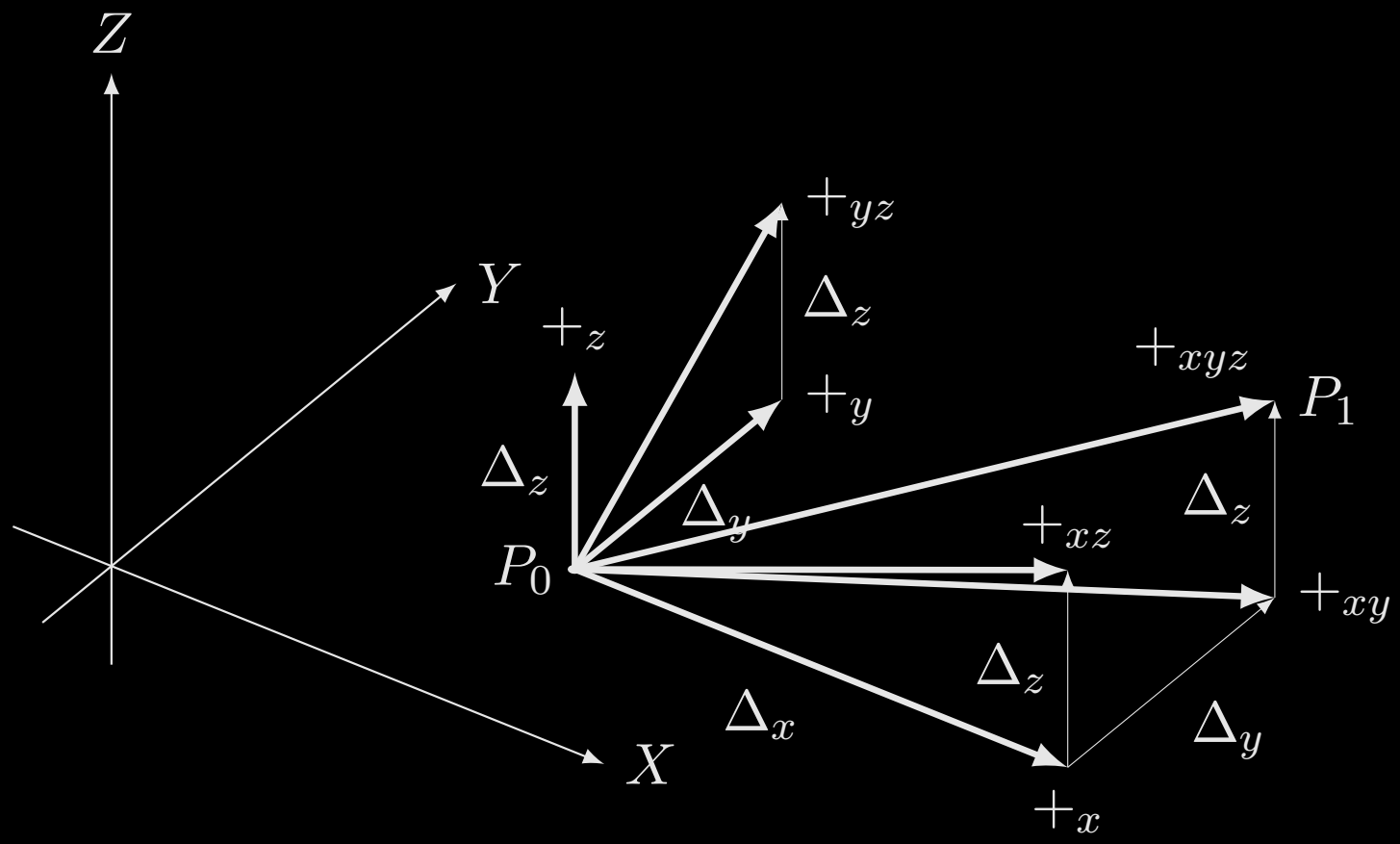
- Geometry
- Algorithms

Architecture needs Computer Science

How do we teach Computer Science to Architects?

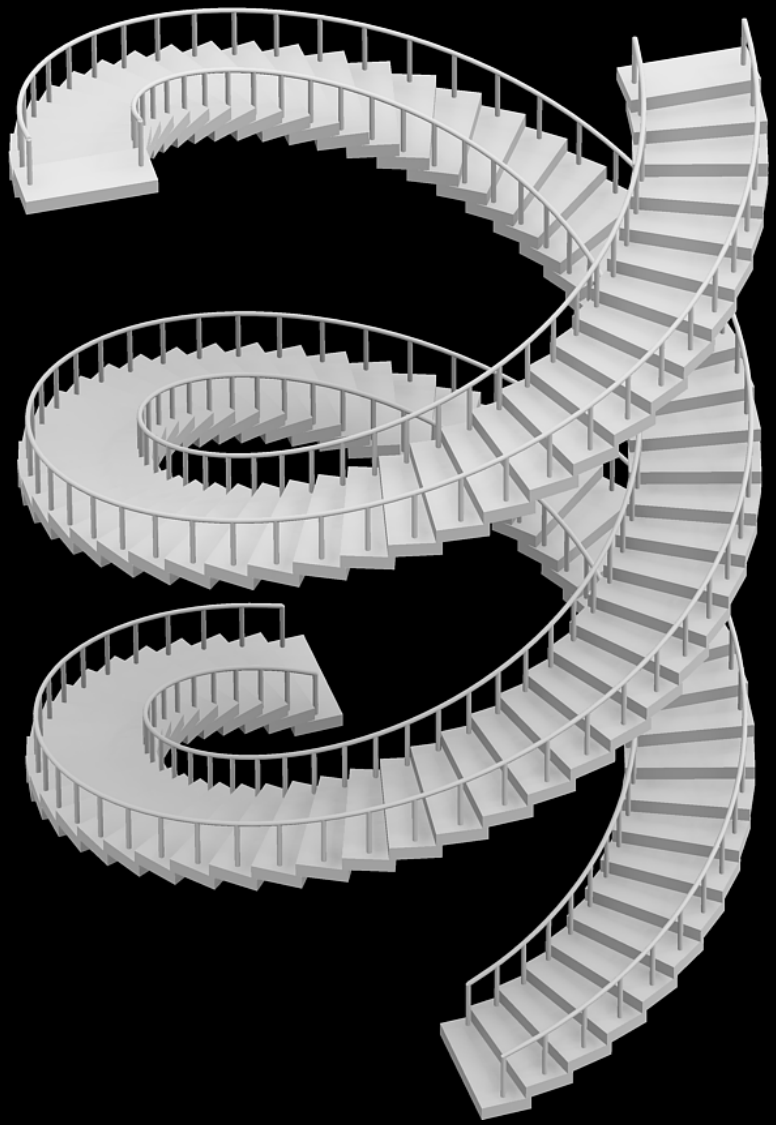
- Geometry
- Algorithms
- Programming Language

Geometry - Rectangular Coordinates

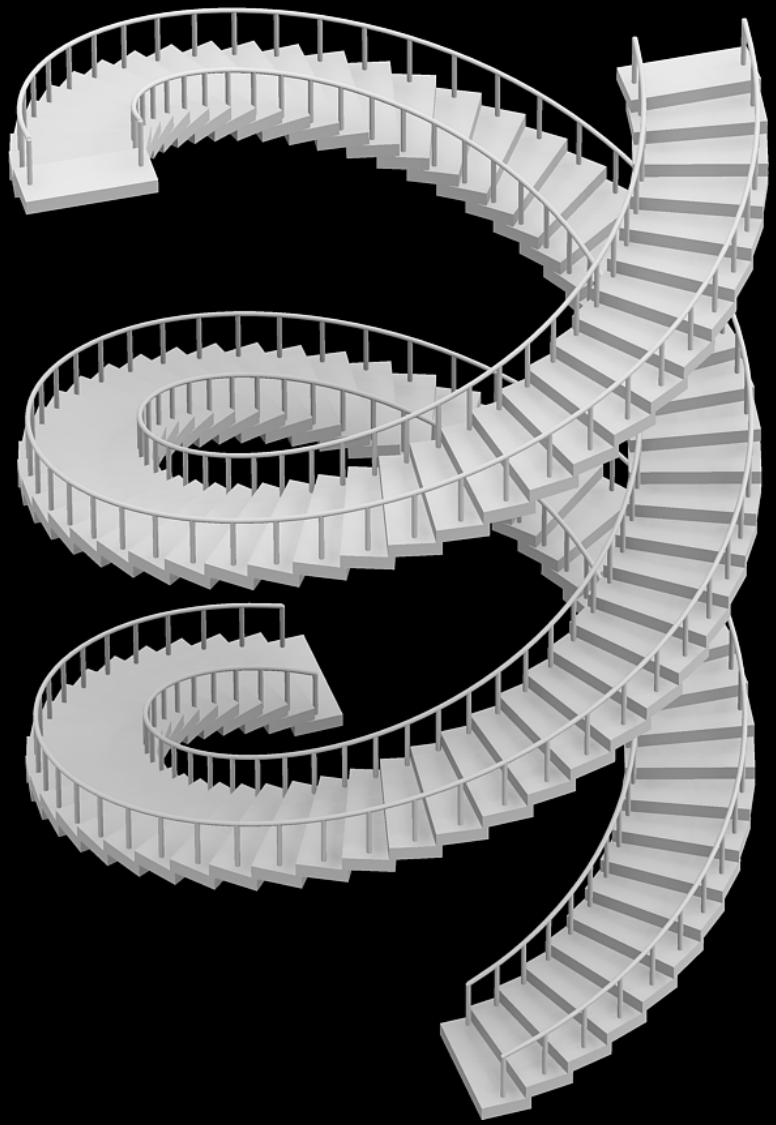


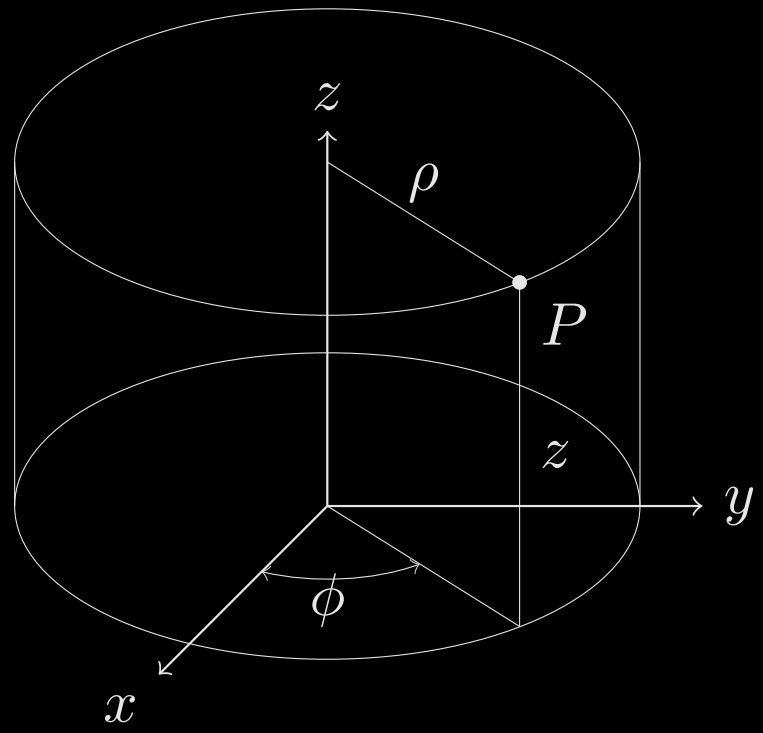
$$P_1 = +_{xyz}(P_0, \Delta_x, \Delta_y, \Delta_z)$$

Geometry - Cylindrical Coordinates



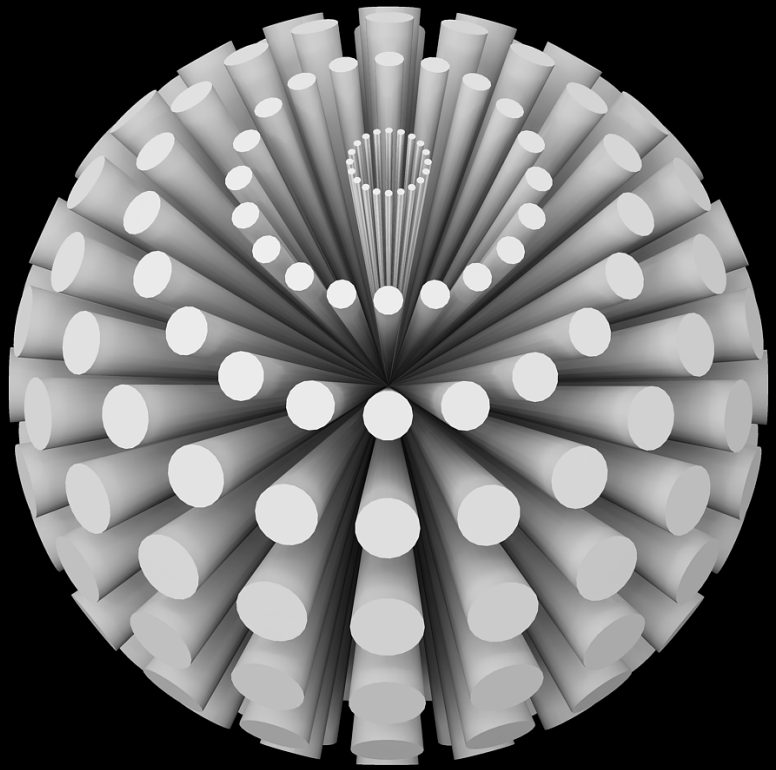
Geometry - Cylindrical Coordinates



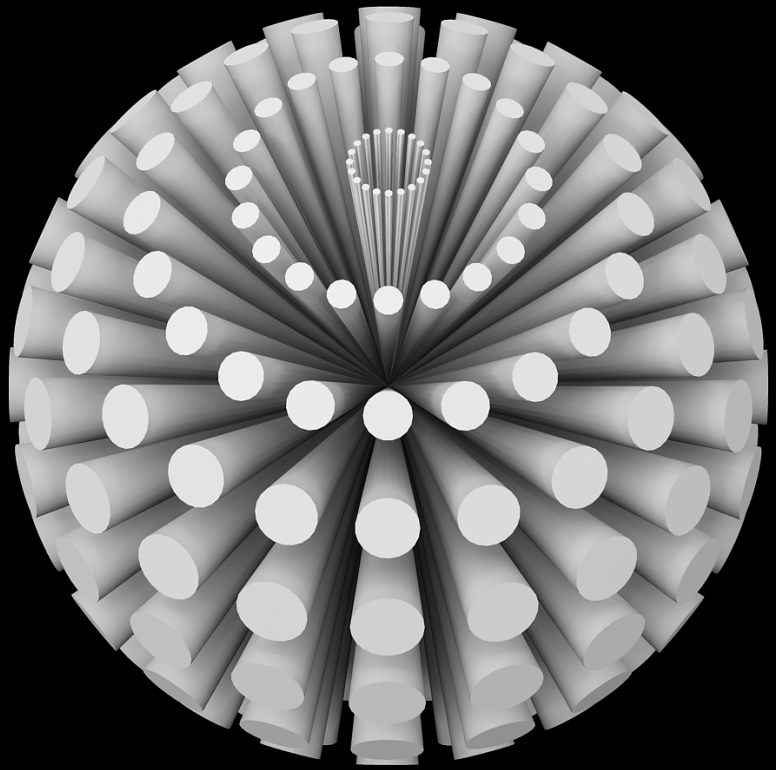


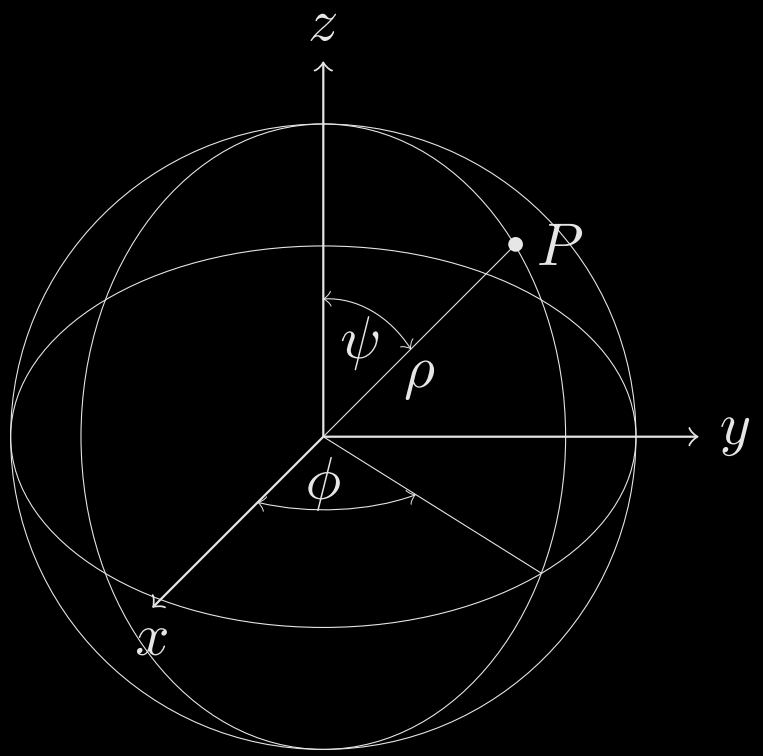
$$\forall_s \in [0, \ldots, n]$$
$$\mathsf{step}(\, \mathsf{cyl}(r_i, \Delta_\omega \cdot s, \Delta_h \cdot s),$$
$$\mathsf{cyl}(r_e, \Delta_\omega \cdot s, \Delta_h \cdot s))$$

Geometry - Spherical Coordinates



Geometry - Spherical Coordinates



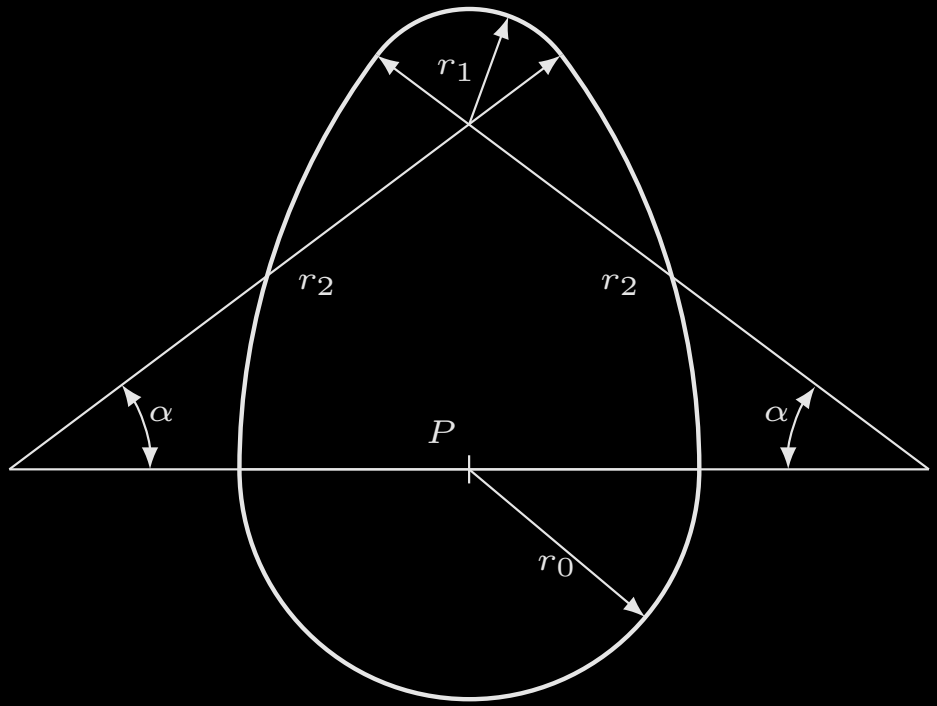


$$\forall_\psi \in \{0, \frac{\pi}{10}, \ldots, \pi\}$$

$$\forall_\phi \in \{\frac{\pi}{10}, 2\frac{\pi}{10}, \ldots, 2\pi\}$$

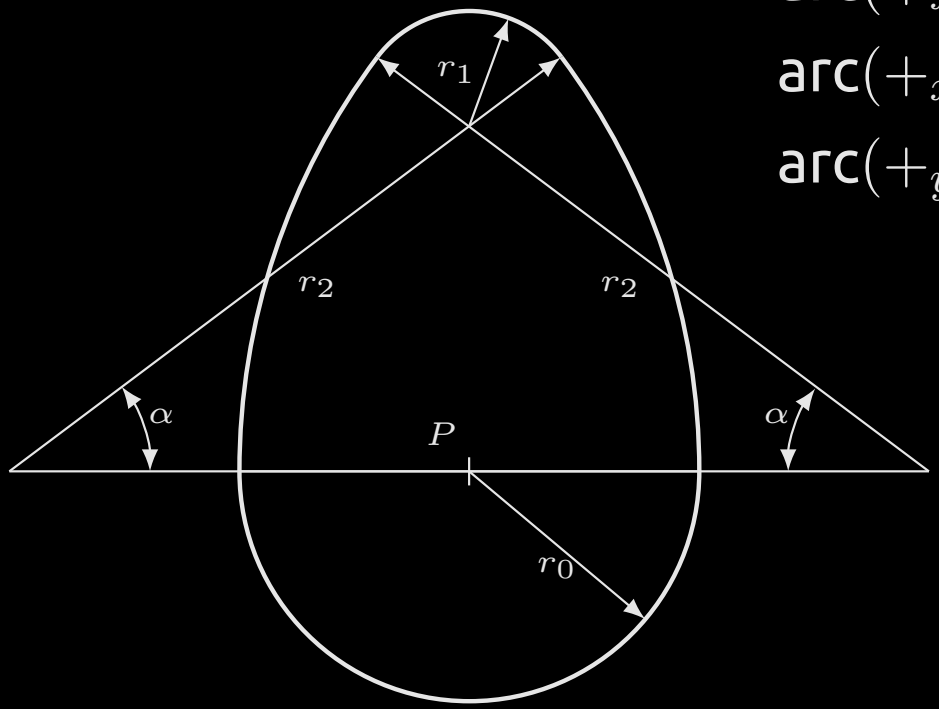
$$\mathsf{cone}(\mathsf{sph}(1, \phi, \psi), \frac{\sin\psi}{10}, \mathsf{xyz}(0, 0, 0))$$

Geometry - Primitives



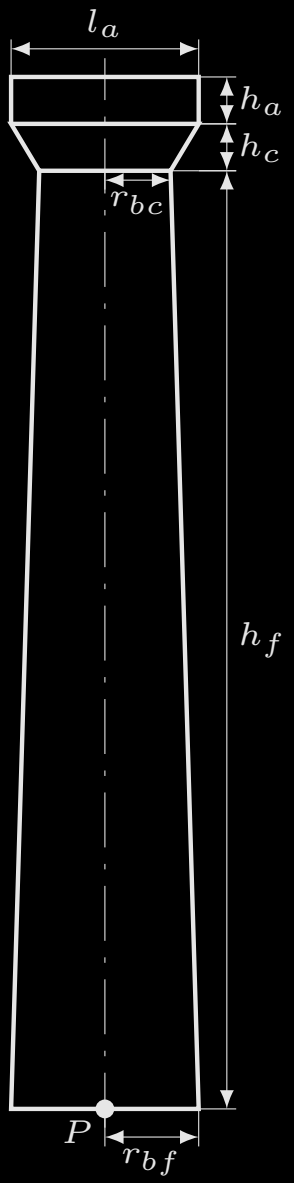
Geometry - Primitives

$$\mathsf{egg}(p, r_0, r_1, h) =$$
$$\mathsf{arc}(p, r_0, 0, -\pi) \cup$$
$$\mathsf{arc}(+_x(p, r_0 - r_2), r_2, 0, \alpha) \cup$$
$$\mathsf{arc}(+_x(p, r_2 - r_0), r_2, \alpha - \pi, \alpha) \cup$$
$$\mathsf{arc}(+_y(p, (r_2 - r_1)\,\mathsf{sin}\,\alpha), r_1, \alpha, \pi - 2\alpha)$$

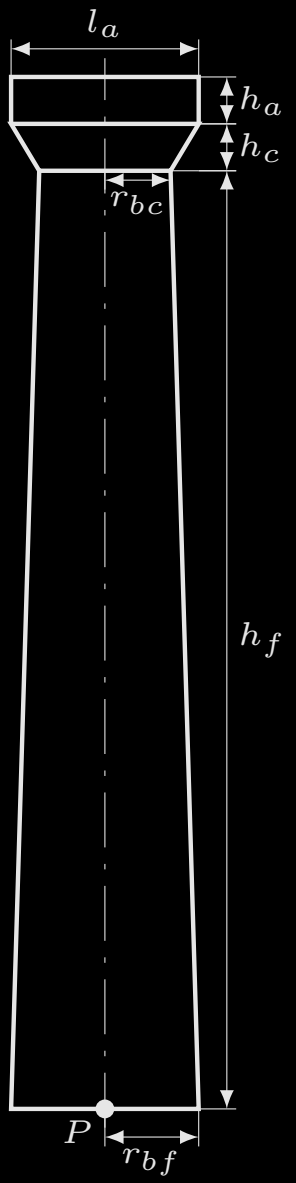


$$\alpha = 2\,\mathsf{tan}^{-1}\,\frac{r_0 - r_1}{h - r_0 - r_1}$$
$$r_2 = \frac{r_0 - r_1\,\mathsf{cos}\,\alpha}{1 - \mathsf{cos}\,\alpha}$$

Algorithms

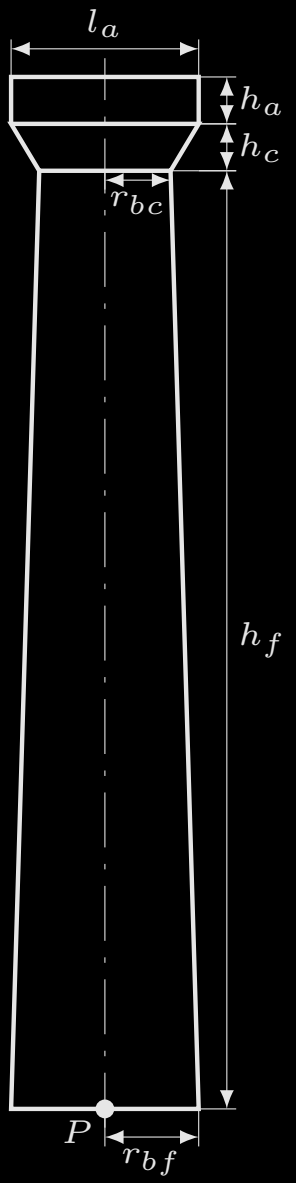


Algorithms



$$\mathsf{column}(p, h_f, r_{bf}, h_c, r_{bc}, h_a, l_a) =$$
$$\quad \mathsf{frustrum}(p, h_f, r_{bf}, r_{bc}) \cup$$
$$\quad \mathsf{frustrum}(+_z(p, h_f), h_c, r_{bc}, l_a/2) \cup$$
$$\quad \mathsf{box}(+_z(p, h_f + h_c), h_a, l_a)$$

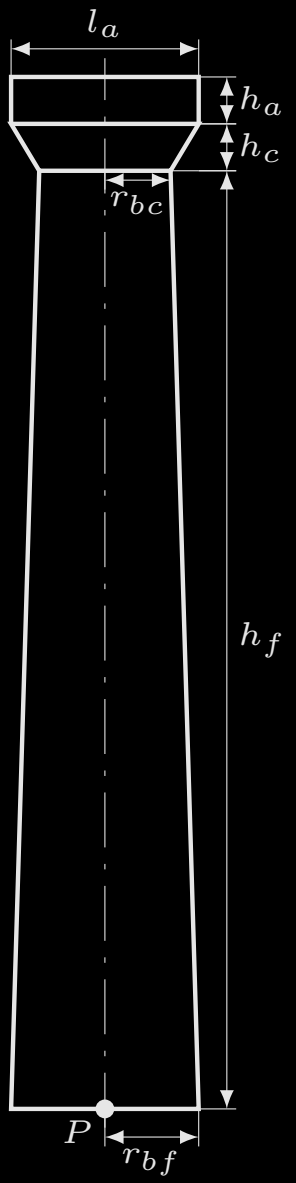
Algorithms



Crassitudo columnarum erit duorum modulorum, altitudo cum capitulo XIIII. Capituli crassitudo unius moduli, latitudo duorum et moduli sextae partis. Crassitudo capituli dividatur in partes tres, e quibus una plinthus cum cymatio fiat, altera echinus cum anulis, tertia hypotrachelion.

Vitruvius

Algorithms



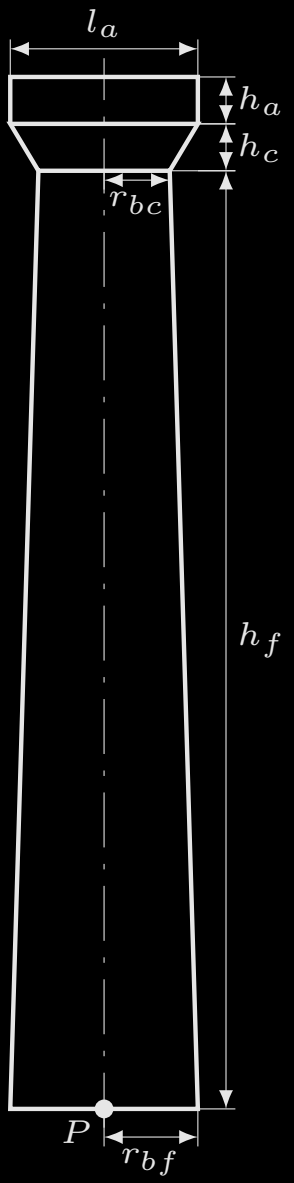
$$h_f = 13 r_{bf}$$

$$h_c = \frac{2}{3} r_{bf}$$

$$h_a = \frac{1}{3} r_{bf}$$

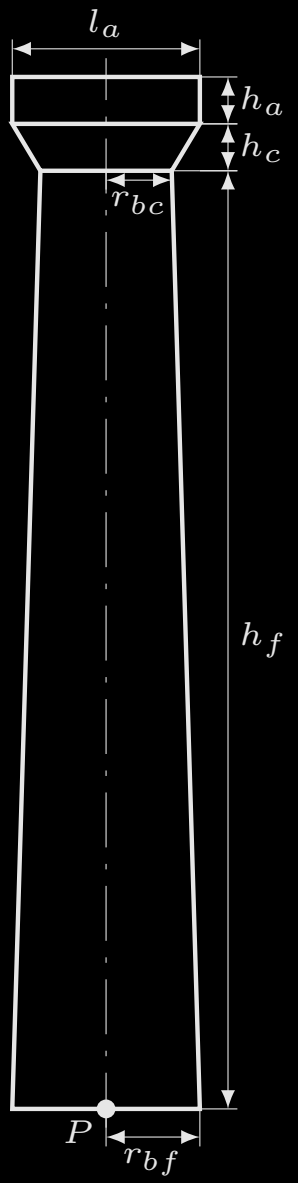
$$l_a = \frac{13}{6} r_{bf}$$

Algorithms



$$\mathsf{column}(p, h_f, r_{bf}, h_c, r_{bc}, h_a, l_a) =$$
$$\quad \mathsf{frustrum}(p, h_f, r_{bf}, r_{bc}) \cup$$
$$\quad \mathsf{frustrum}(+_z(p, h_f), h_c, r_{bc}, l_a/2) \cup$$
$$\quad \mathsf{box}(+_z(p, h_f + h_c), h_a, l_a)$$

Algorithms



$$\mathsf{column}(p, h_f, r_{bf}, h_c, r_{bc}, h_a, l_a) =$$
$$\mathsf{frustrum}(p, h_f, r_{bf}, r_{bc}) \cup$$
$$\mathsf{frustrum}(+_z(p, h_f), h_c, r_{bc}, l_a/2) \cup$$
$$\mathsf{box}(+_z(p, h_f + h_c), h_a, l_a)$$

$$\mathsf{doricColumn}(p, r_{bf}, r_{bc}) =$$
$$\mathsf{column}(p, 13r_{bf}, r_{bf}, \frac{2}{3}r_{bf}, r_{bc}, \frac{1}{3}r_{bf}, \frac{13}{6}r_{bf})$$

Programming Languages

Programming Languages

Programming Languages

Most languages have the same *computational power*

Programming Languages

Most languages have the same *computational power*

but they differ in

- clarity
- learning curve
- paradigms
- execution speed
- productivity
- …
- *expressive power*

Programming Languages - Clarity

Visual Basic

```
Function ConicSpiralPoints(Length,N)
  Dim points()
  ReDim points(N-1)
  Dim t, i
  For i=0 To N-1
    t=i*Length/N
    points(i)=Pt(t*Cos(5*t),t*Sin(5*t),t)
  Next
  ConicSpiralPoints=points
End Function
```

Programming Languages - Clarity

Haskell

```
conicSpiralPoints length n =
  [(t*(cos 5*t), t*(sin 5*t), t) |
     t <- [i*length/n | i <- [0..n]]]
```

Programming Languages - Choices

- It must be simple
- It must be expressive
- It must be usable with a CAD application

Programming Languages - Choices

- It must be simple
- It must be expressive
- It must be usable with a CAD application

- ArchiCad
  - GDL
- AutoCAD
  - Visual Basic
  - C++
  - AutoLisp
- Rhino
  - RhinoScript
  - Grasshopper
  - Python

Programming Languages - My Choice

Lisp isn't a language, it's a building material.

Alan Kay

Programming Languages - My Choice

Lisp isn't a language, it's a building material.

Alan Kay

Lisp made me aware that software could be close to executable mathematics.

Peter Deutsch

Programming Languages - My Choice

Lisp isn't a language, it's a building material.

Alan Kay

Lisp made me aware that software could be close to executable mathematics.

Peter Deutsch

Lisp is worth learning for the profound enlightenment experience you will have when you finally get it; that experience will make you a better programmer for the rest of your days, even if you never actually use Lisp itself a lot

Eric S. Raymond

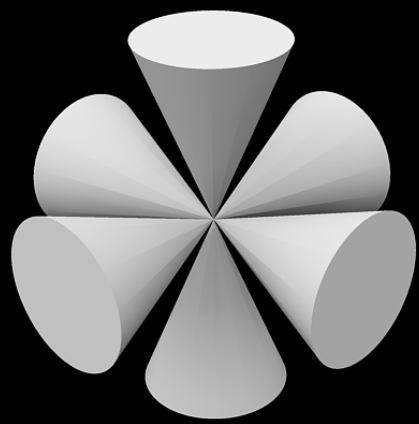
Course Outline

Course Outline

- Syntax & Semantics
- Recursion
- State & Randomness
- Lists
- Solid Geometry
- Higher Order Functions
- Parametrics

Syntax & Semantics

```
(defun cross (p rb rt c)
  (cone-frustrum
    p rb (+x p c) rt)
  (cone-frustrum
    p rb (+y p c) rt)
  (cone-frustrum
    p rb (+z p c) rt)
  (cone-frustrum
    p rb (+x p (- c)) rt)
  (cone-frustrum
    p rb (+y p (- c)) rt)
  (cone-frustrum
    p rb (+z p (- c)) rt))
```

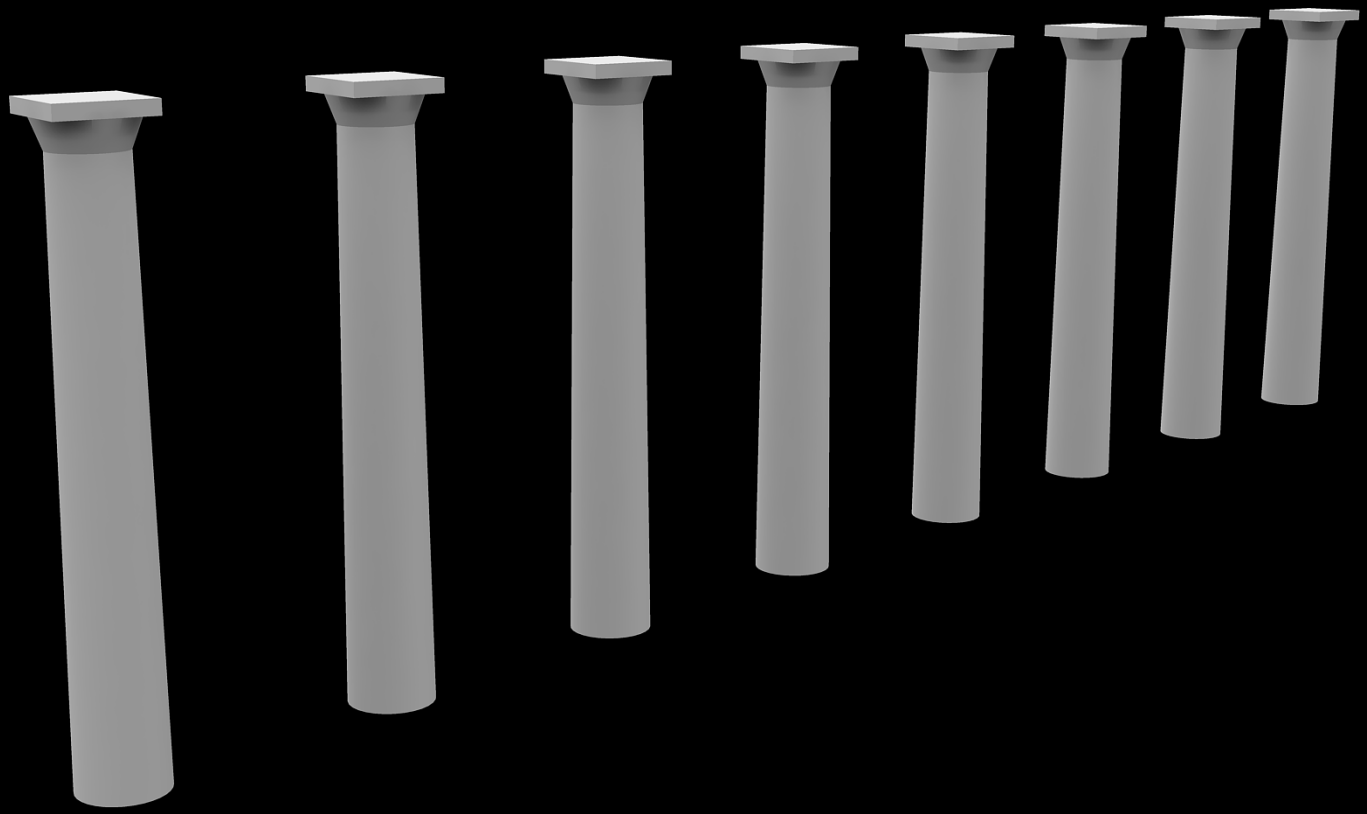


Recursion

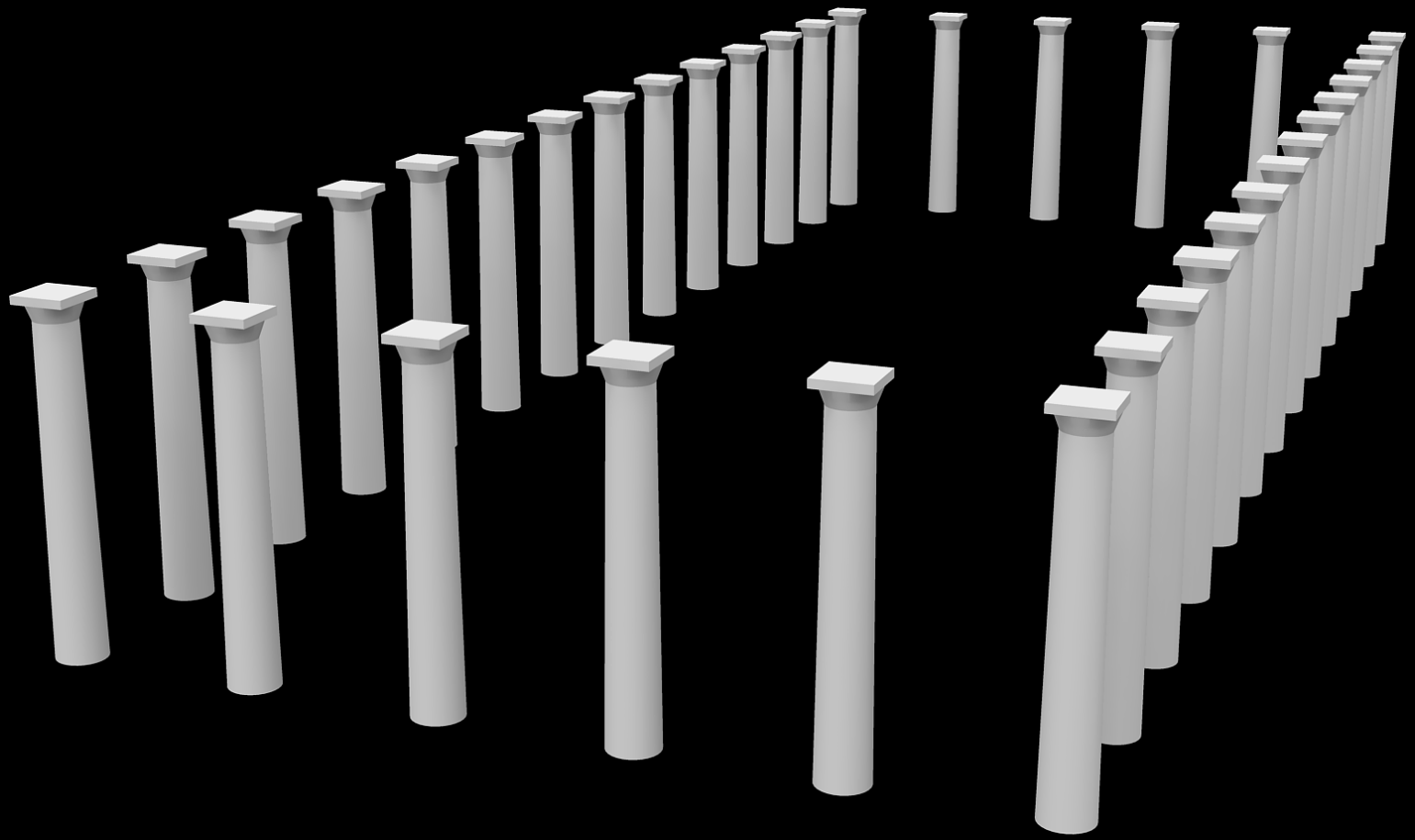
```
(defun colonnade (p h v n)
  (if (= n 0)
    nil
    (progn
      (column p h)
      (colonnade (+c p v) h v (- n 1)))))
```

Recursion

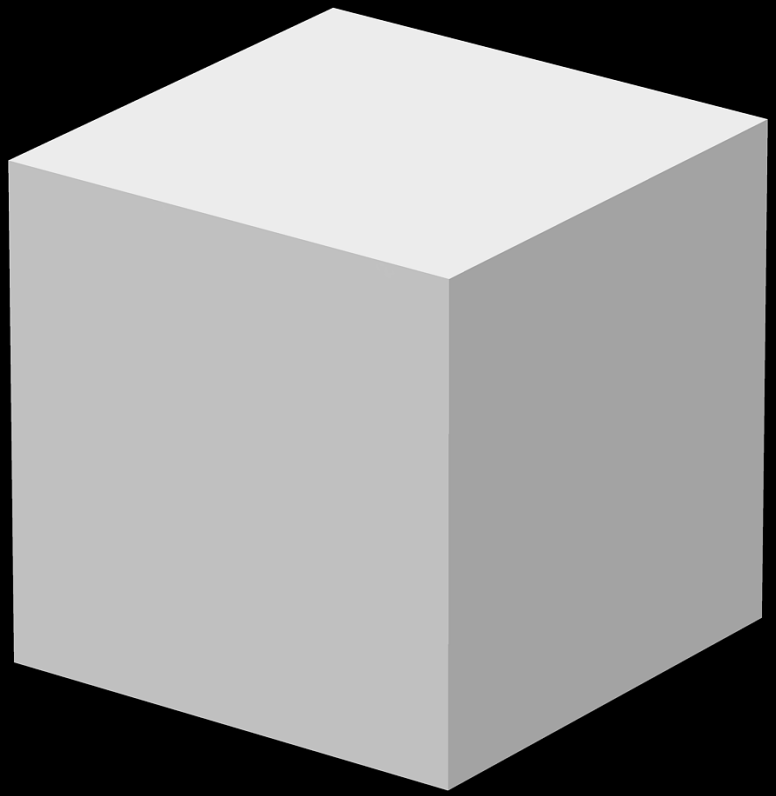
```
(defun colonnade (p h v n)
  (if (= n 0)
    nil
    (progn
      (column p h)
      (colonnade (+c p v) h v (- n 1)))))
```



Recursion



Recursion



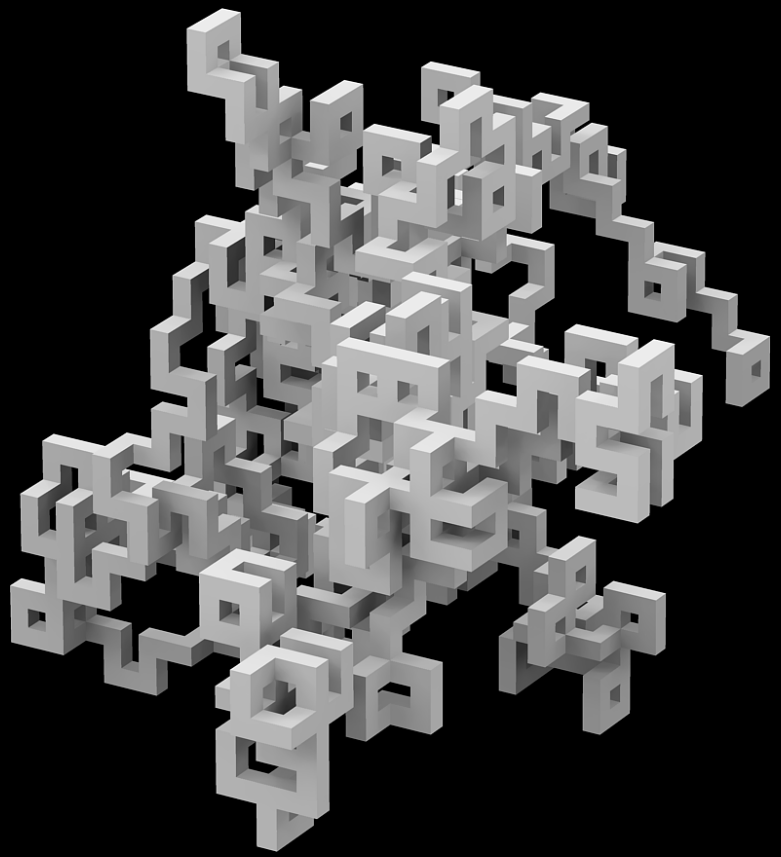
```
(defun menger (p l n)
  (if (= n 0)
    (cube p l)
    (progn
      (setq l (/ l 3.0))
      (foreach i (enum 0 2 1)
        (foreach j (enum 0 2 1)
          (foreach k (enum 0 2 1)
           (if (or (= i j 1) (= i k 1) (= j k 1))
              nil
              (menger
                (+xyz p (* i l) (* j l) (* k l))
                l
                (- n 1)))))))))
```

State & Randomness

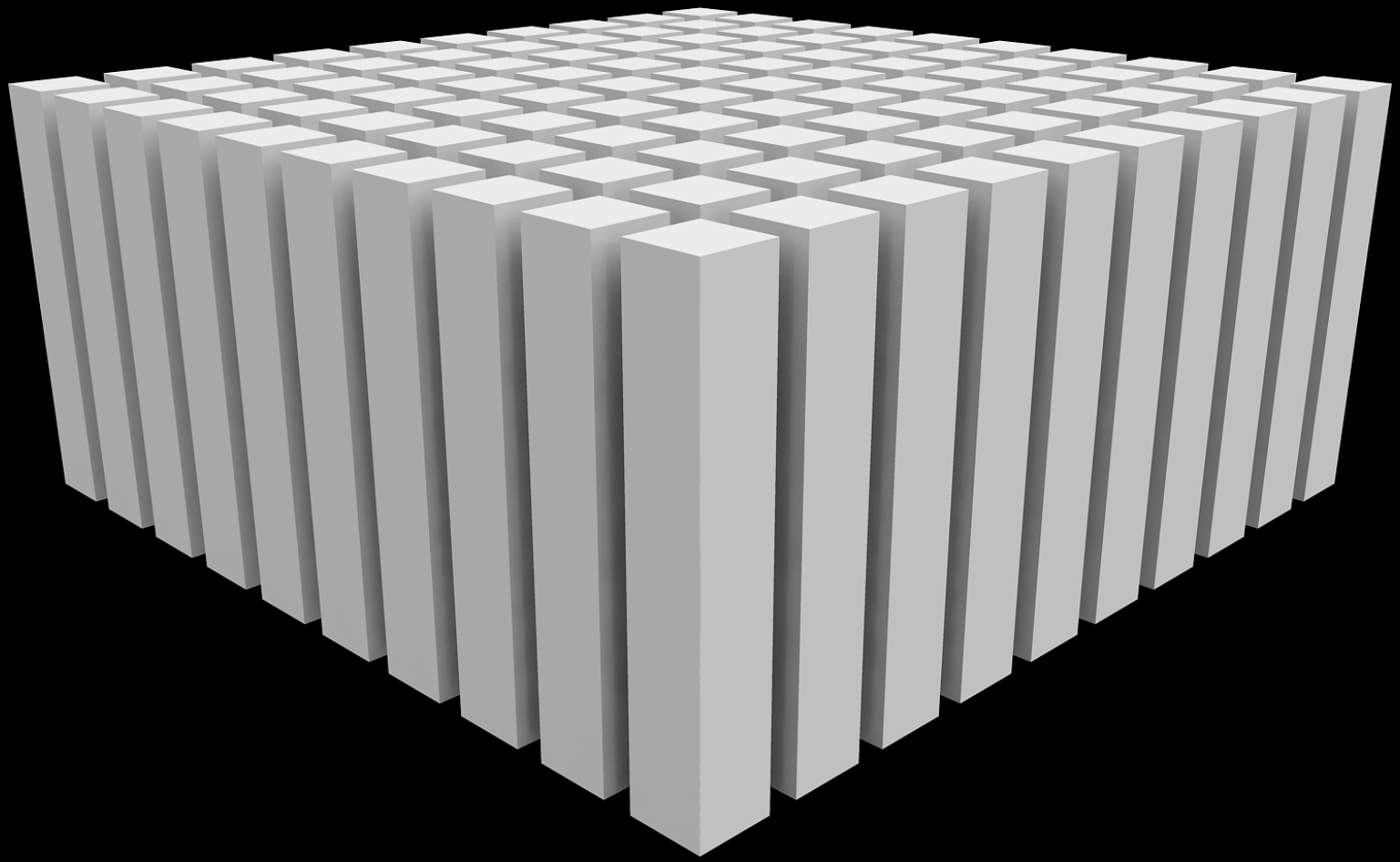
```
(defun random-direction ()
  (sph 1
       (* (random-[] 0 4) pi/2)
       (* (random-[] 0 4) pi/2)))
```

State & Randomness

```
(defun random-direction ()
  (sph 1
       (* (random-[] 0 4) pi/2)
       (* (random-[] 0 4) pi/2)))
```



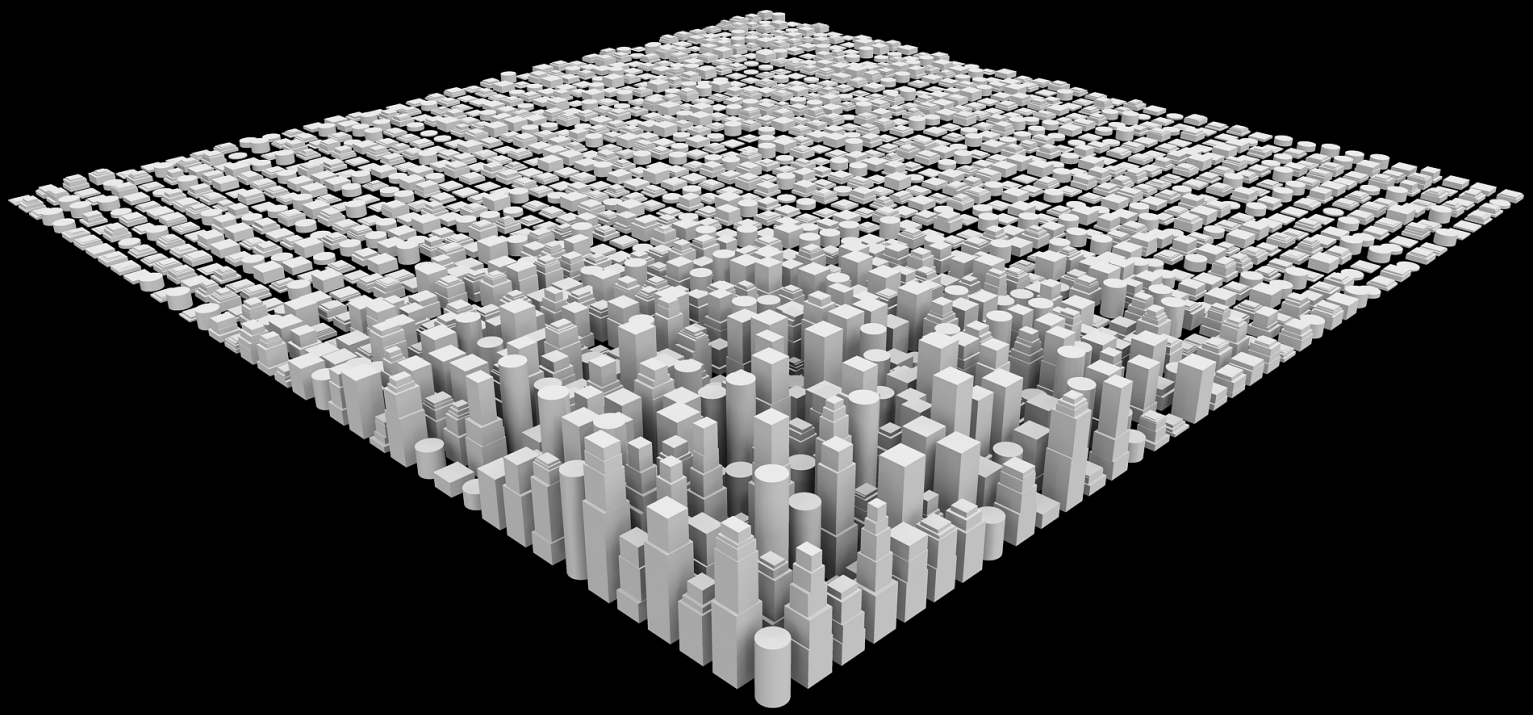
State & Randomness



State & Randomness

Gaussian Distribution

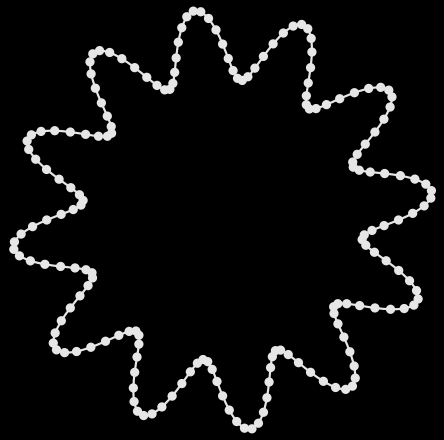
$$f(x,y) = e^{-\left((\frac{x-x_o}{\sigma_x})^2+(\frac{y-y_o}{\sigma_y})^2\right)}$$

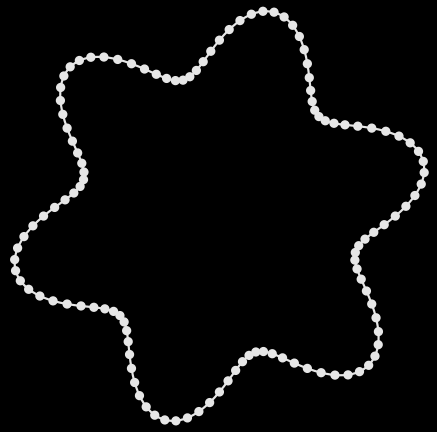


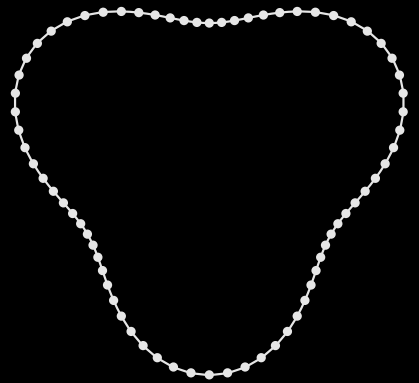
Lists

Lists

Curves from Points

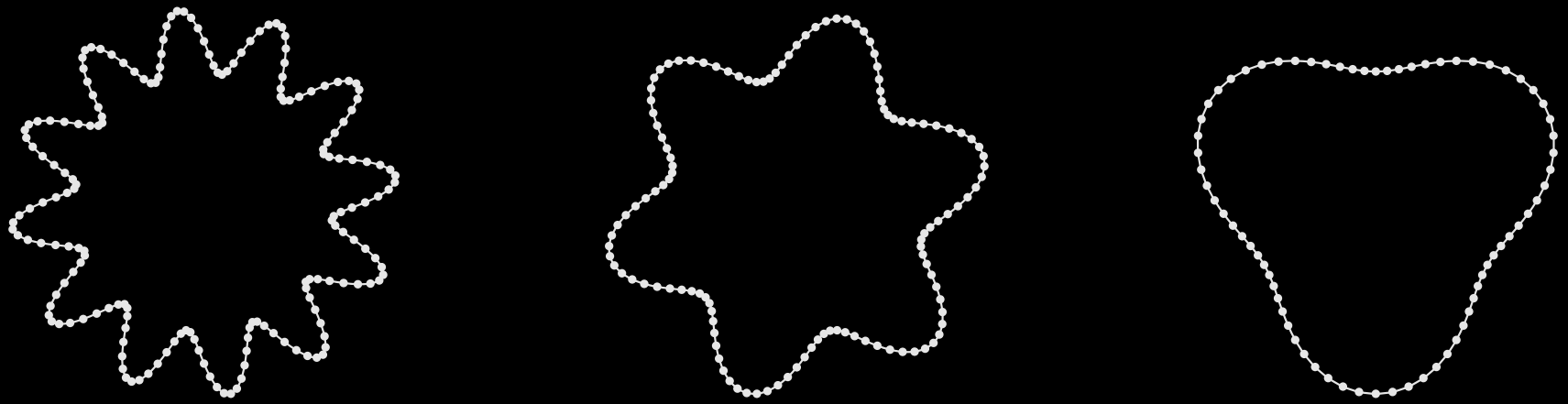






Lists

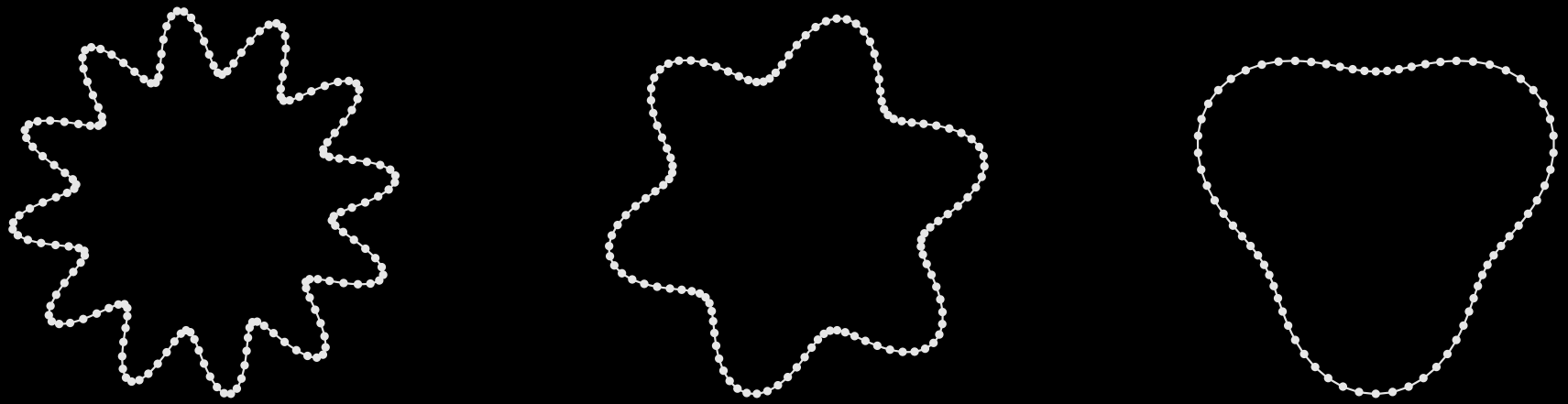
Curves from Points



```
(defun sinusoidal-arc-points (p r a c fi dfi n)
  (if (= n 0)
    (list)
    (cons (+pol p (+ r (* a (sin (* c fi)))) fi)
          (sinusoidal-arc-points p r a c (+ fi dfi) dfi (- n 1)))))
```

Lists

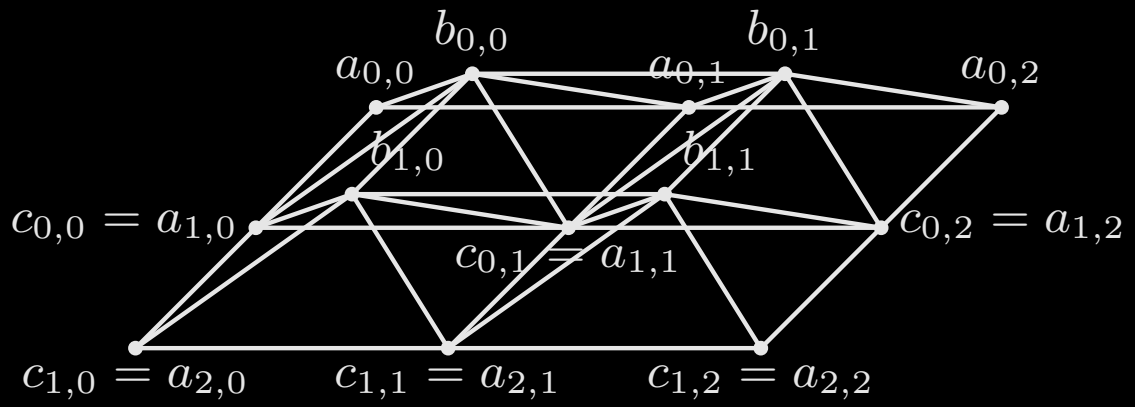
Curves from Points



```
(defun sinusoidal-arc-points (p r a c fi dfi n)
  (if (= n 0)
    (list)
    (cons (+pol p (+ r (* a (sin (* c fi)))) fi)
          (sinusoidal-arc-points p r a c (+ fi dfi) dfi (- n 1)))))

(defun sinusoide-circular-arc (p ri re c n)
  (sinusoidal-arc-points
    p (/ (+ ri re) 2.0) (/ (- re ri) 2.0)
    c 0 (/ 2*pi n) n))
```

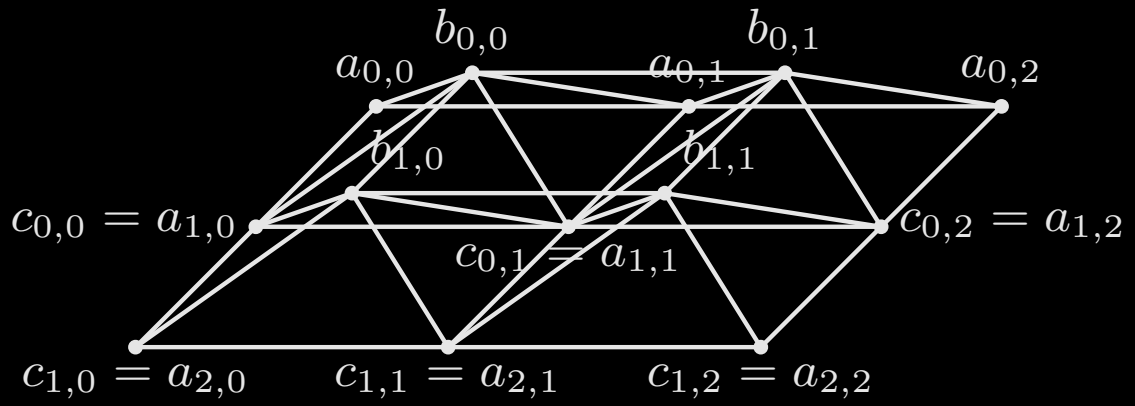
Lists

Trusses



Lists

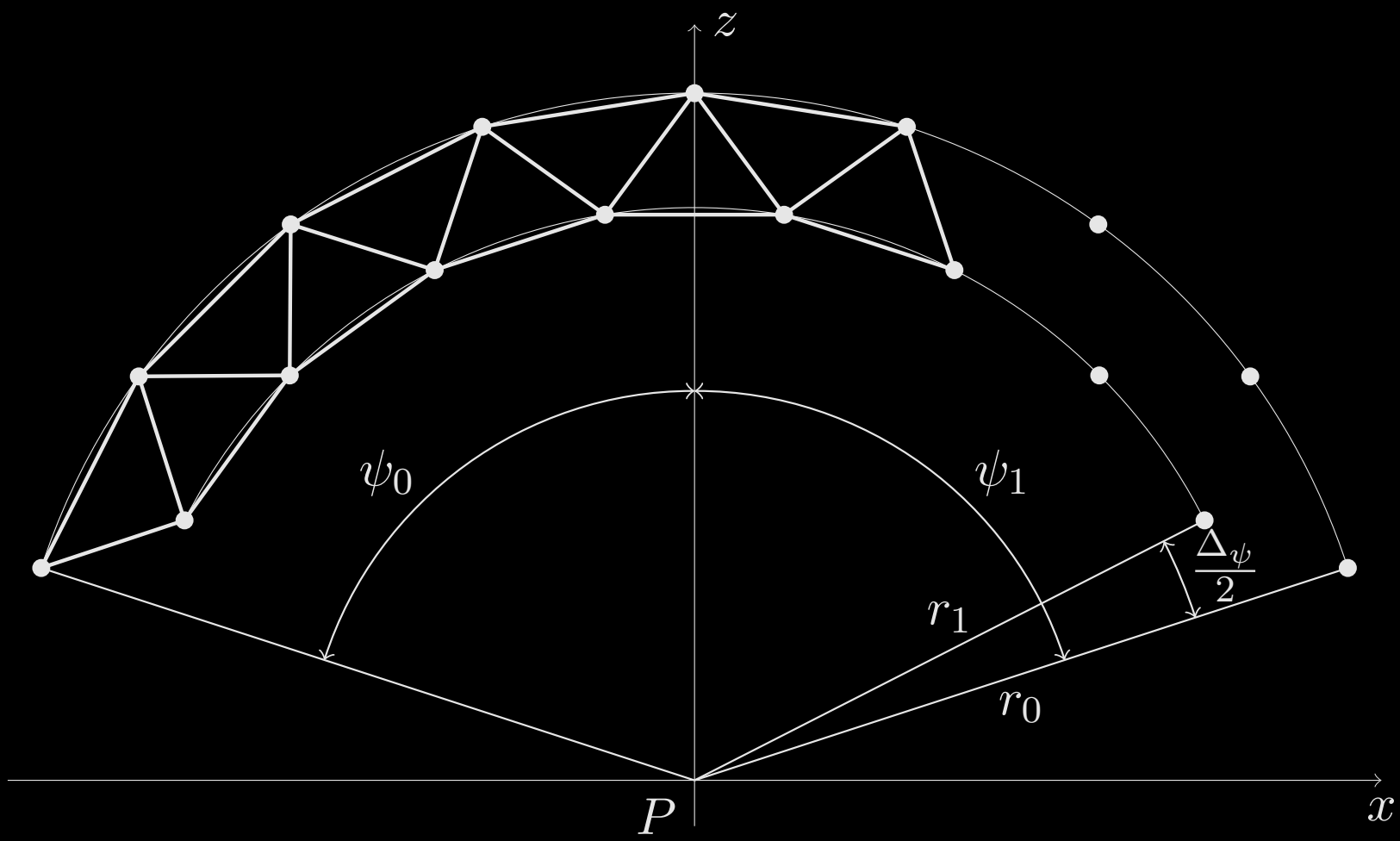
Trusses



$$
\begin{pmatrix}
\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & \dots & a_{0,n-1} & a_{0,n} \end{pmatrix} \\
\begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & \dots & b_{0,n-1} \end{pmatrix} \\
\begin{pmatrix} a_{1,0} & a_{1,1} & a_{1,2} & \dots & a_{1,n-1} & a_{1,n} \end{pmatrix} \\
\dots \\
\begin{pmatrix} c_{m,0} & c_{m,1} & c_{m,2} & \dots & c_{m,n-1} & c_{m,n} \end{pmatrix}
\end{pmatrix}
$$

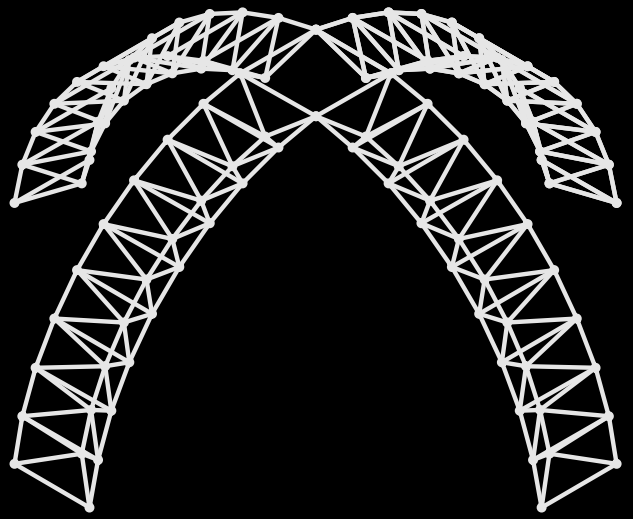
Lists

Trusses



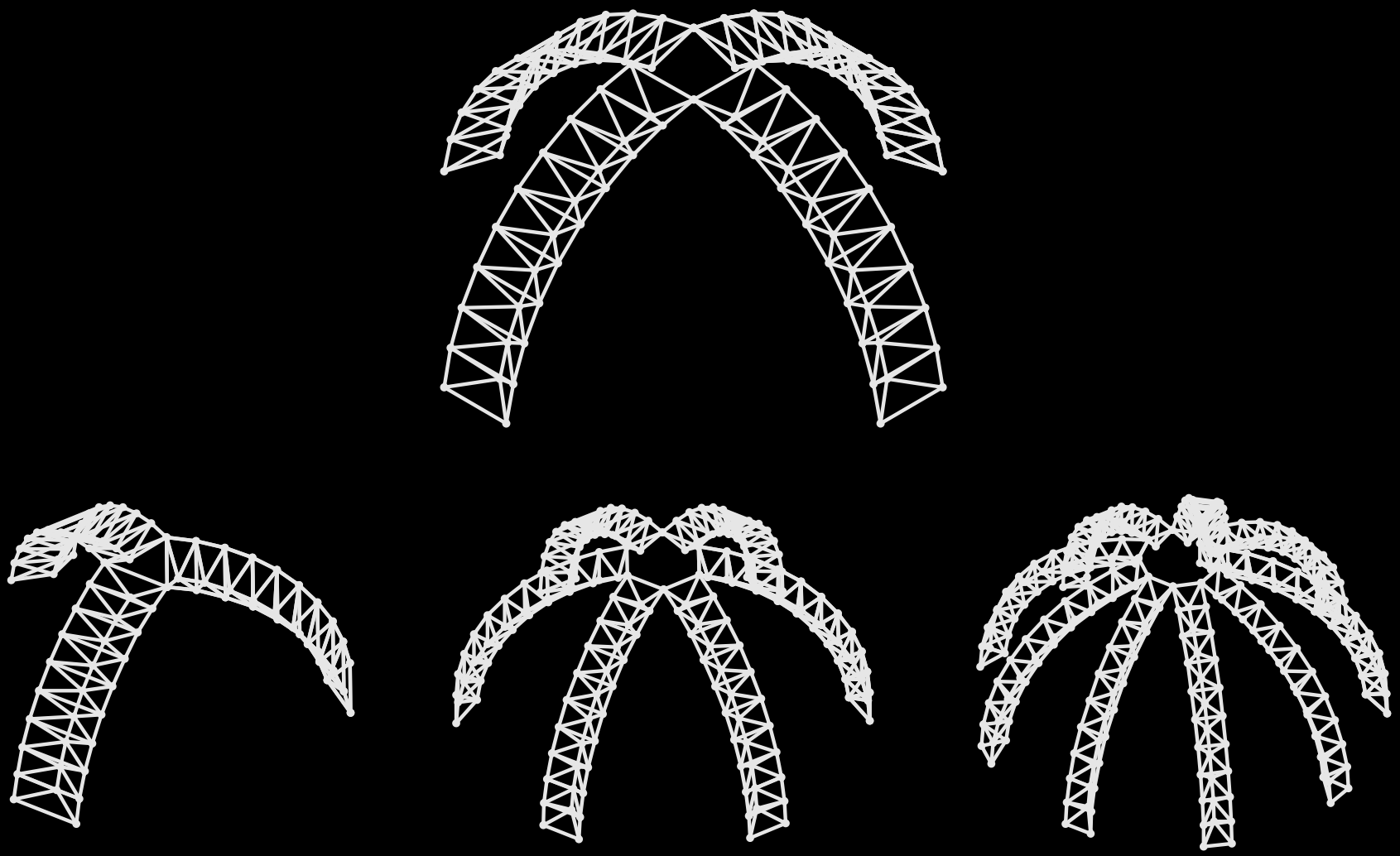
Lists

Trusses



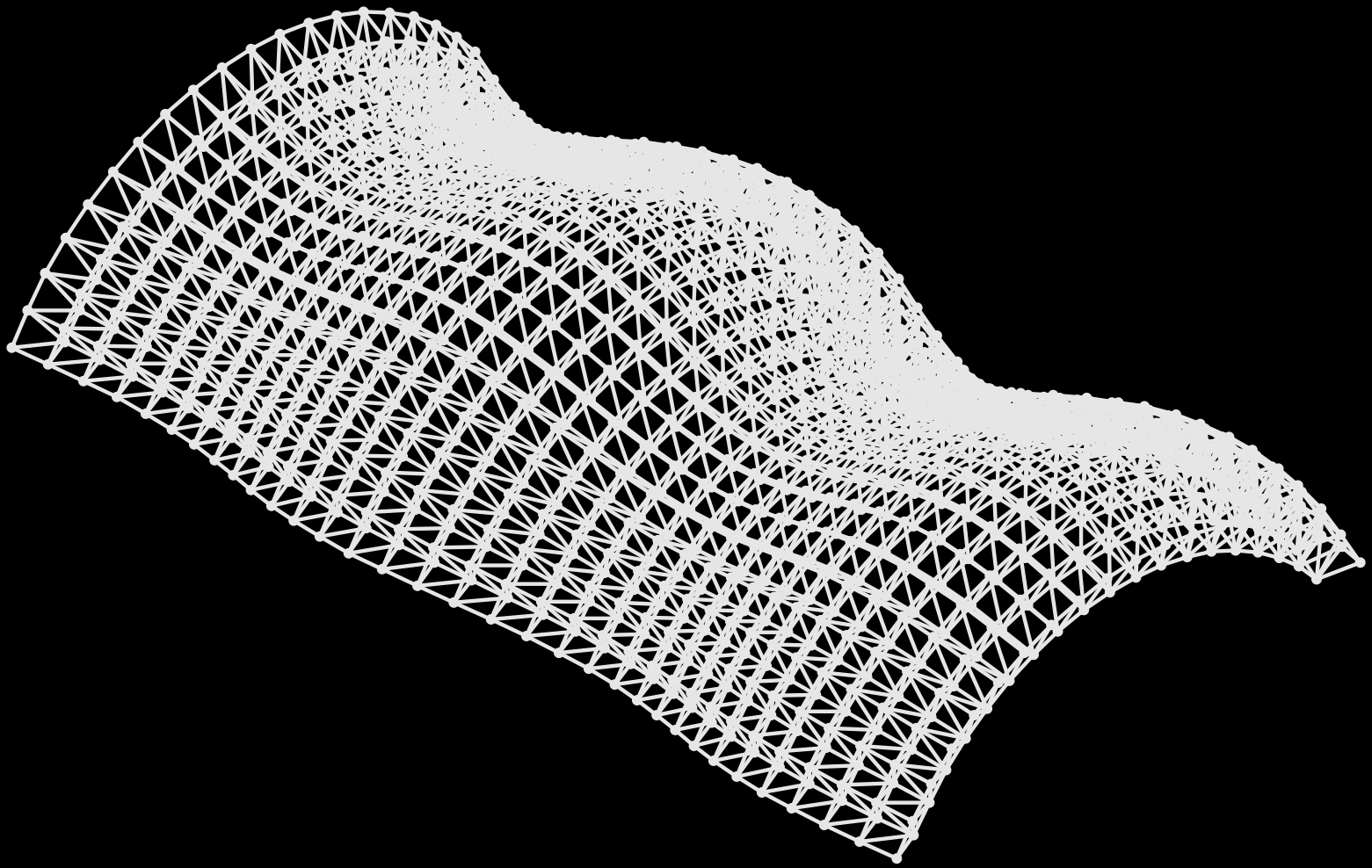
Lists

Trusses



Lists

Trusses



Solid Geometry

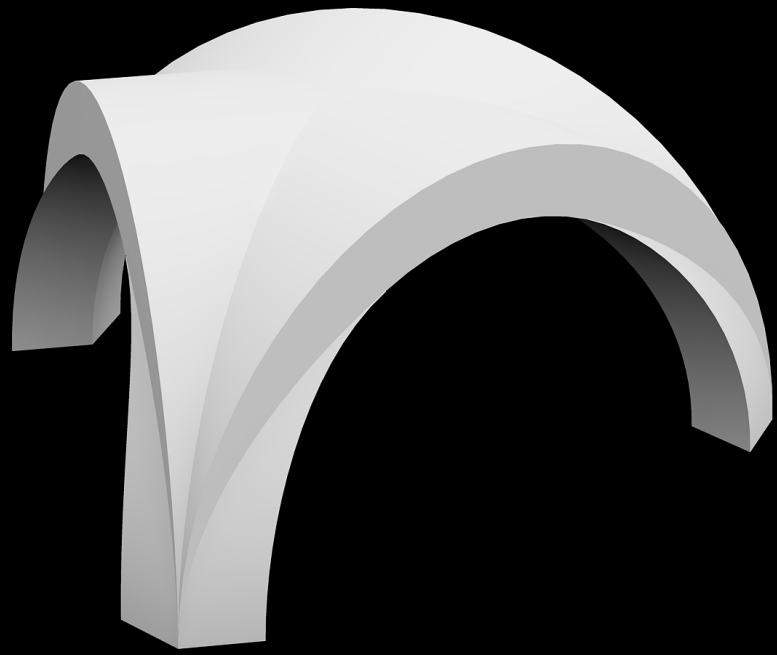
Region Algebra

$$R \cup \varnothing = \varnothing \cup R = R$$
$$R \cup U = U \cup R = U$$
$$R \cup R = R$$
$$R \cap \varnothing = \varnothing \cap R = \varnothing$$
$$R \cap U = U \cap R = R$$
$$R \cap R = R$$
$$R \setminus \varnothing = R$$

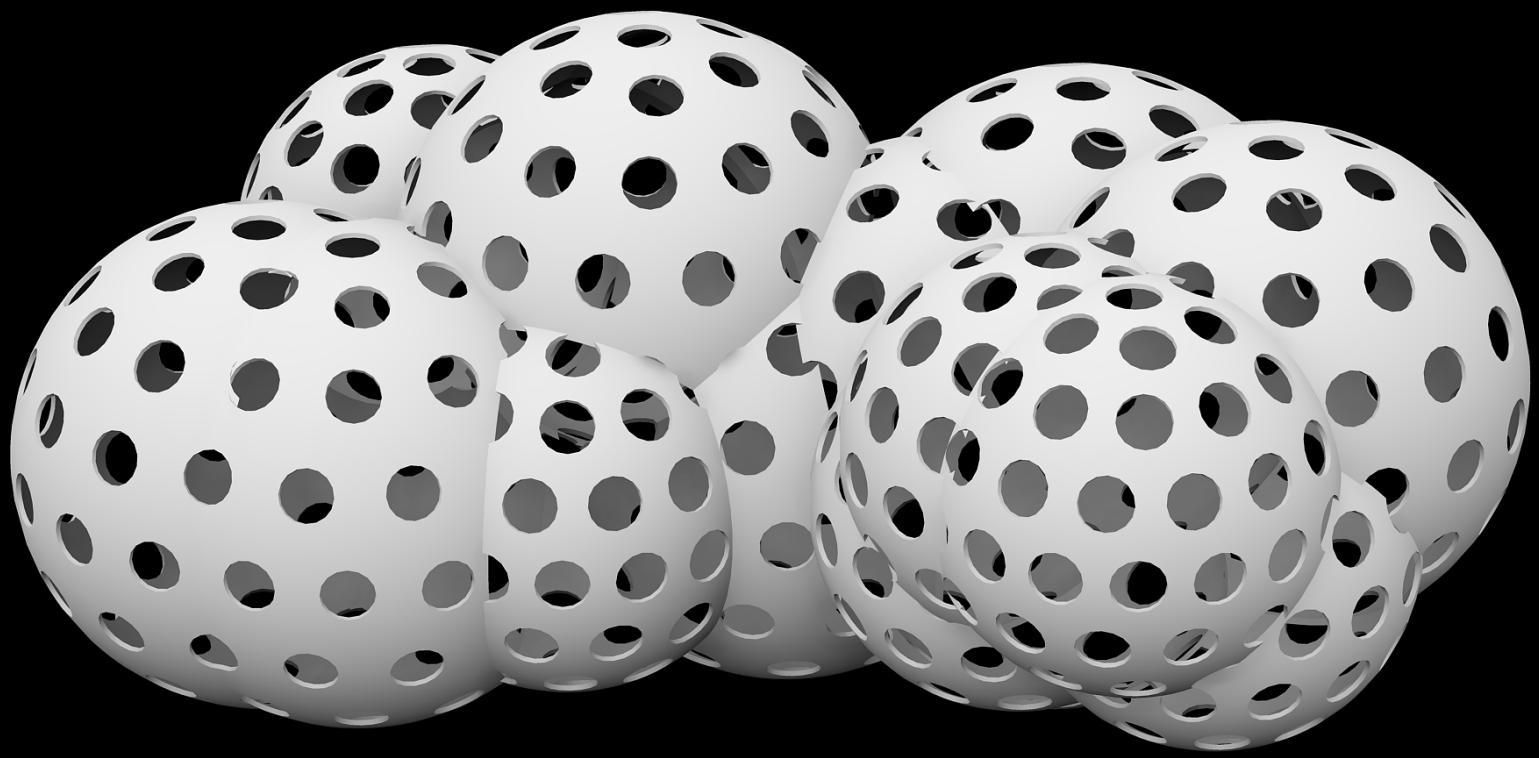
Solid Geometry

Region Algebra

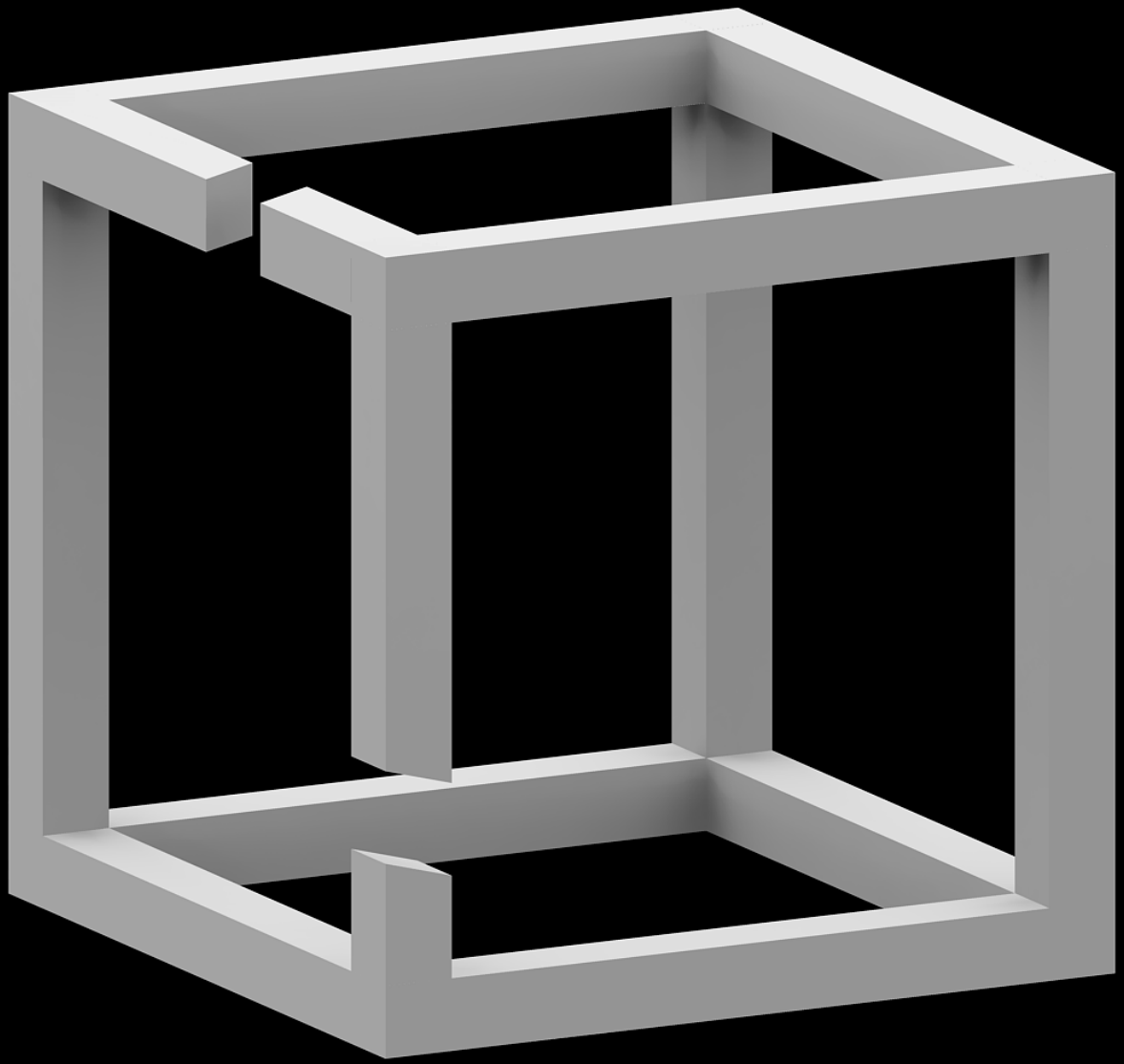
$$R \cup \varnothing = \varnothing \cup R = R$$
$$R \cup U = U \cup R = U$$
$$R \cup R = R$$
$$R \cap \varnothing = \varnothing \cap R = \varnothing$$
$$R \cap U = U \cap R = R$$
$$R \cap R = R$$
$$R \setminus \varnothing = R$$



Solid Geometry

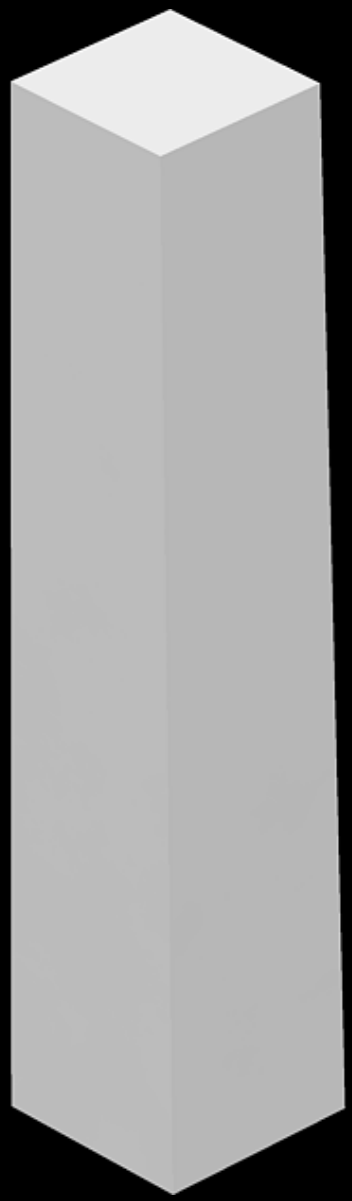


Solid Geometry



Solid Geometry

Solid Geometry



Operations

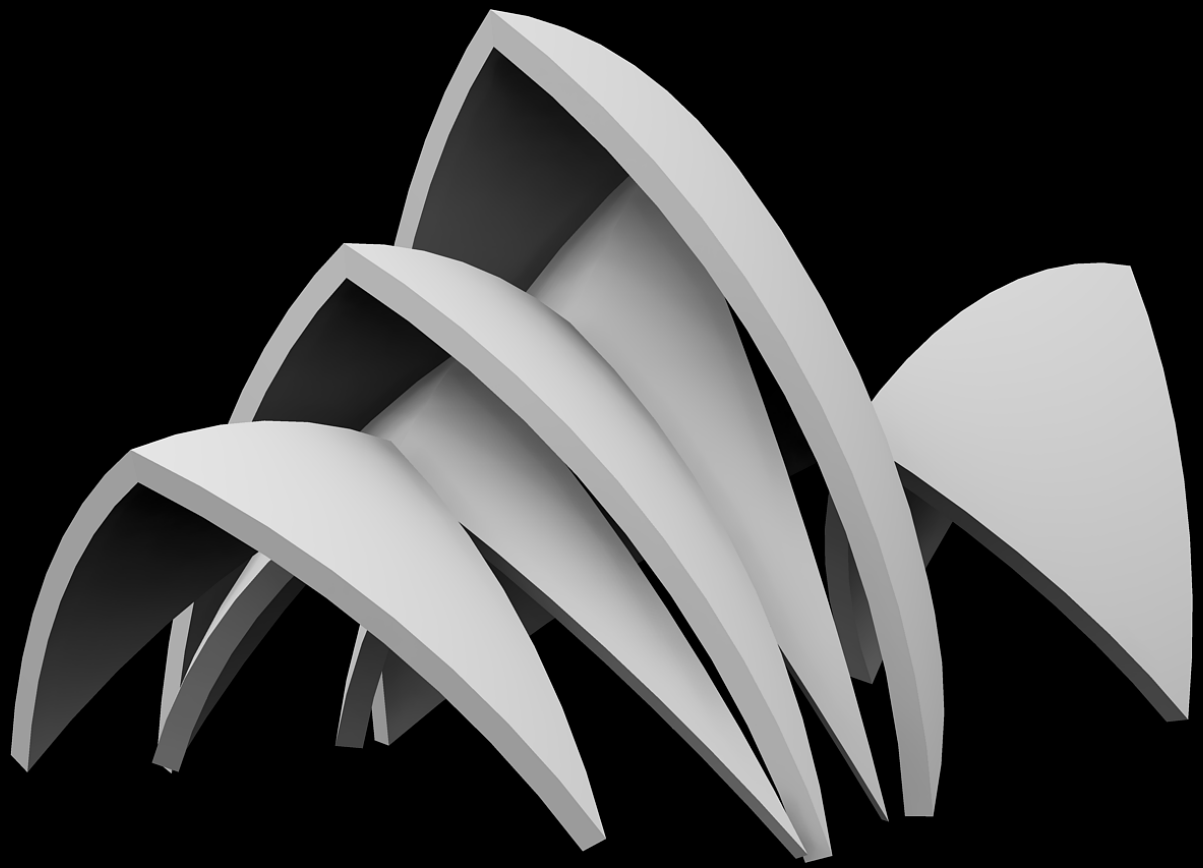
- Regular polygon
- Region
- Sweep
- Rotation
- Scale
- Intersection

Solid Geometry

Solid Geometry

Operations

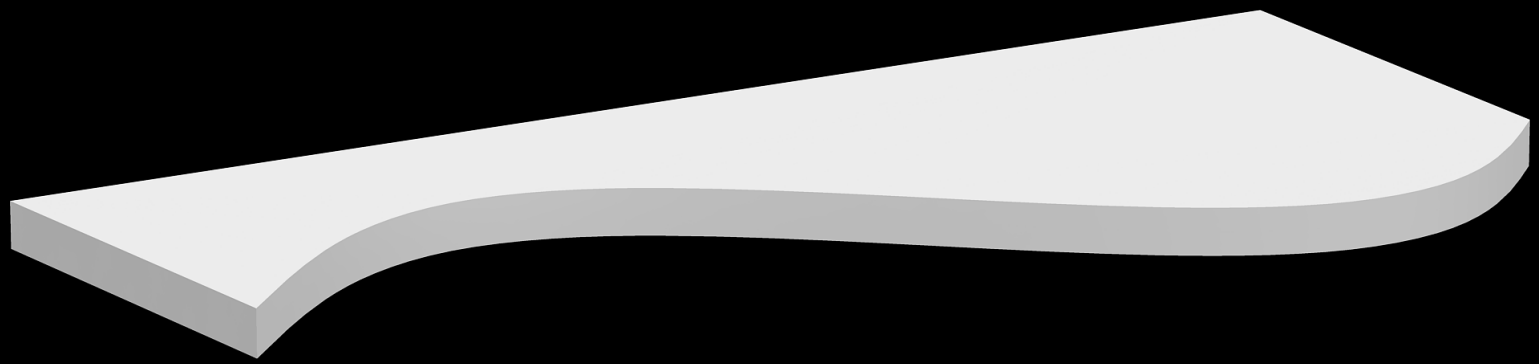
- Sphere
- Subtraction
- Section
- Translation
- Reflection
- Duplication
- Rotation
- Scale



Solid Geometry

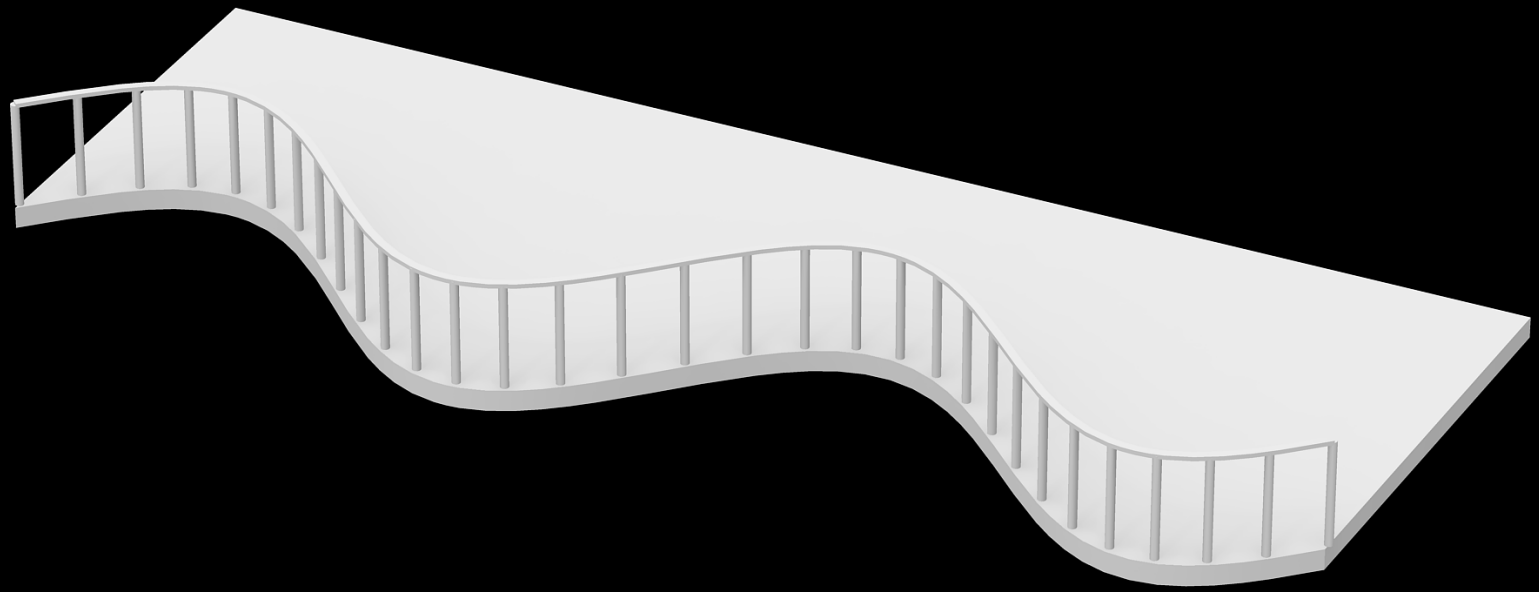
Mathematical Functions

$$y(x) = a\,\mathsf{sin}(\omega x + \phi)$$



Solid Geometry

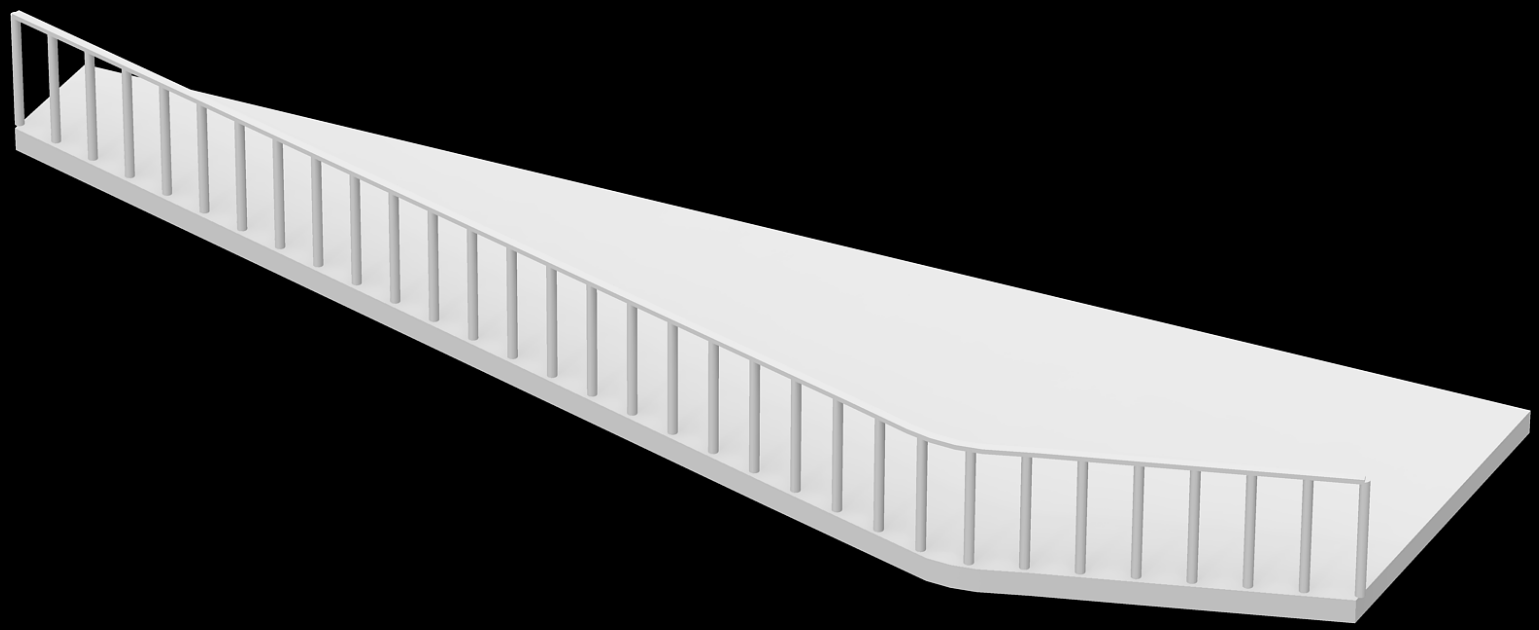
Possible Variations



```
(defun building (...)
  ...
  (balcony ...)
  ...)

(defun balcony (...)
  ... (* a (sin (+ (* omega x) phi))) ...)
```

Solid Geometry

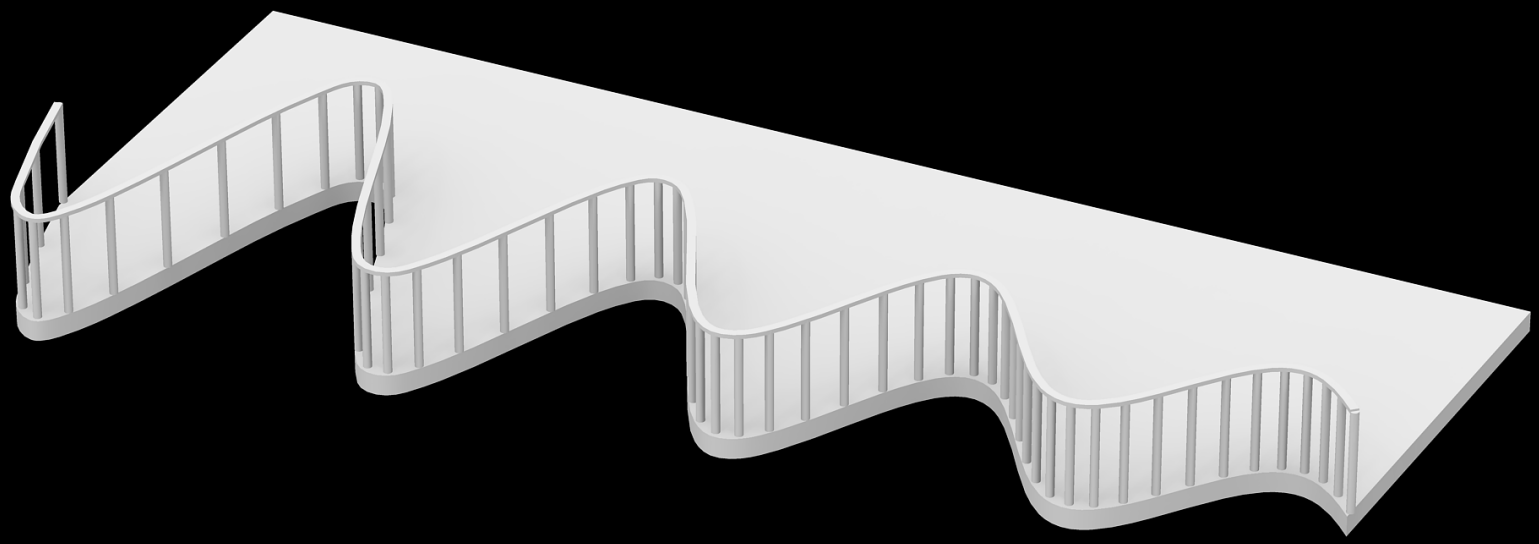
Impossible Variations



$$y(x) = |a + b(x - c)|$$

Solid Geometry

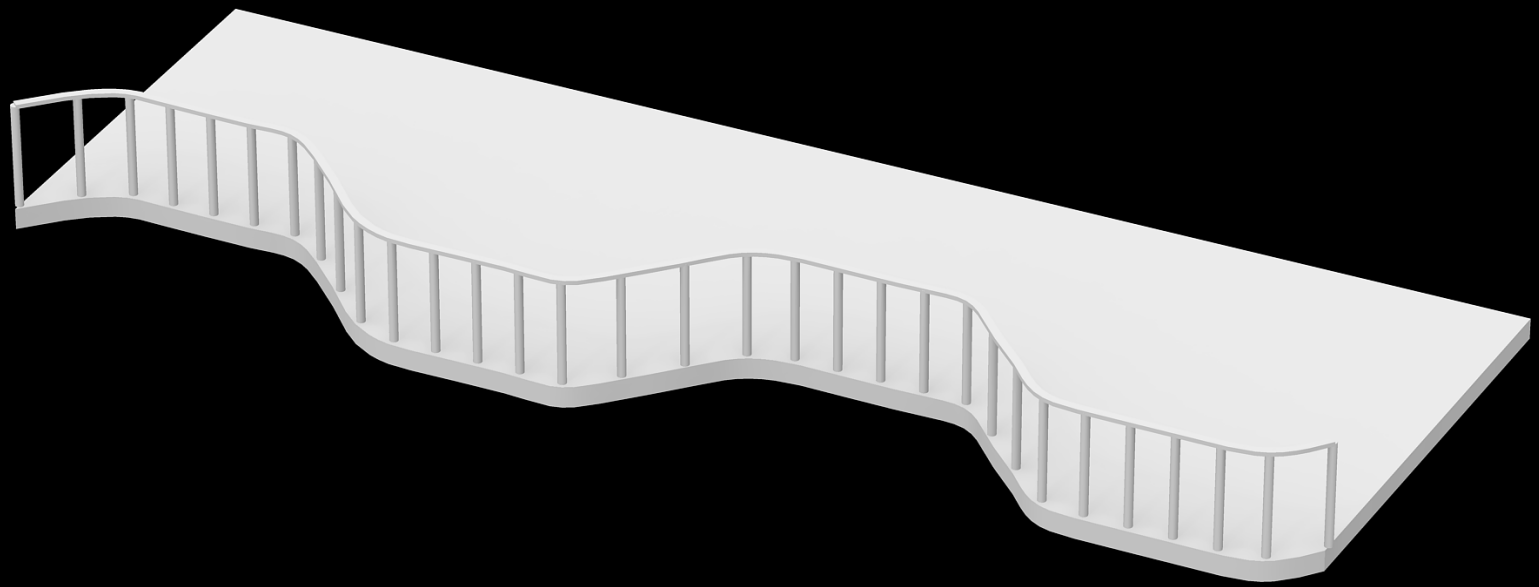
Impossible Variations



$$y(x) = ae^{-bx}\,\mathsf{sin}(cx)$$

Solid Geometry

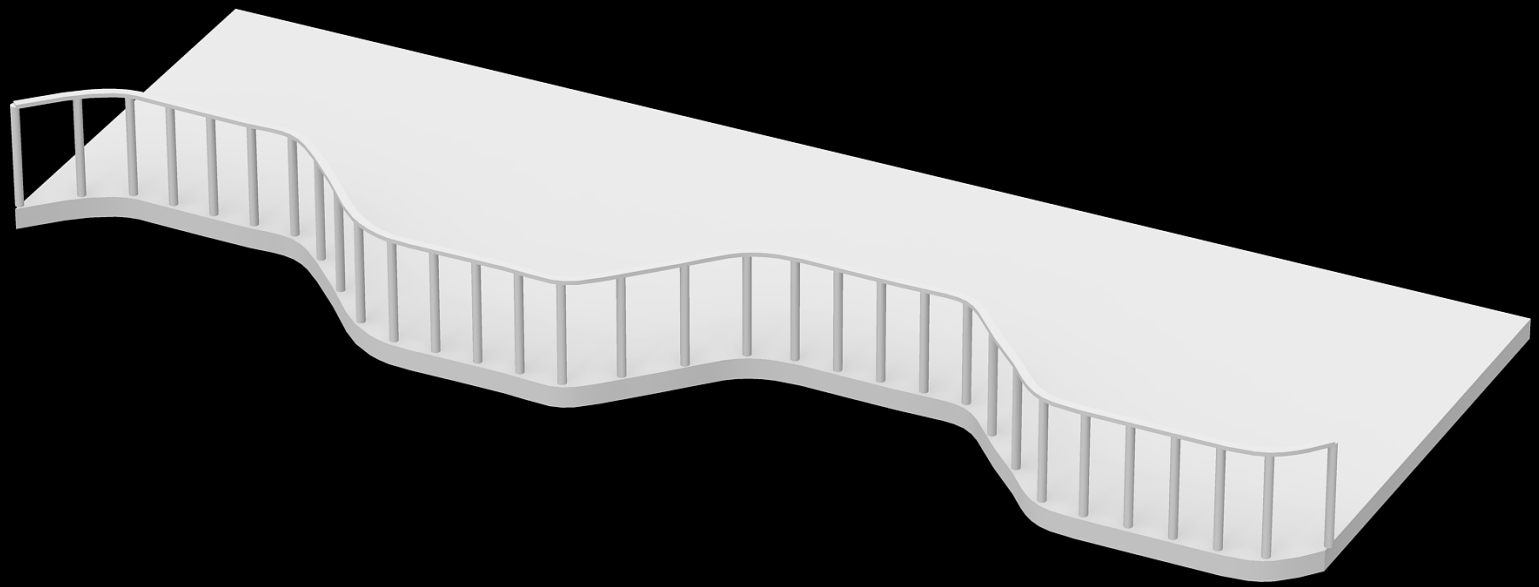
Impossible Variations



$$y(x) = \mathsf{max}(y_0, \mathsf{min}(y_1, a\,\mathsf{sin}(\omega x + \phi)))$$

Solid Geometry

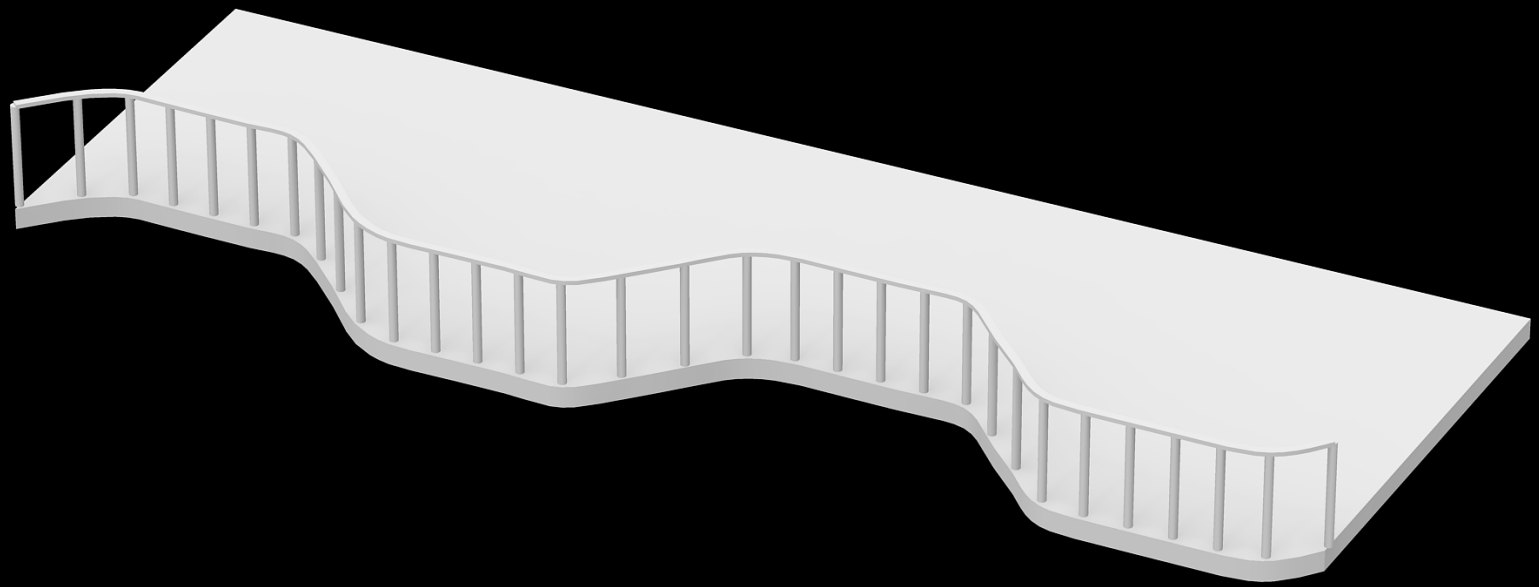
Impossible Variations



```
(defun building (...)
  ...
  (balcony ...)
  ...)
```

Solid Geometry

Impossible Variations



```
(defun building (... balcony ...)
  ...
  (balcony ...)
  ...)
```

Higher Order Functions

Definition

A function that receives or returns functions

Higher Order Functions

Definition

A function that receives or returns functions

Example

$$\sum_{i=m}^{n} f(i) = \begin{cases} 0, & \text{if } m > n \\ x_m + \sum_{i=m+1}^{n} f(i), & \text{otherwise.} \end{cases}$$

Higher Order Functions

Definition

A function that receives or returns functions

Example

$$\sum_{i=m}^{n} f(i) = \begin{cases} 0, & \text{if } m > n \\ x_m + \sum_{i=m+1}^{n} f(i), & \text{otherwise.} \end{cases}$$

$$\frac{d}{dx} f(x) = f'(x) = D_x f(x) = \lim_{\Delta_x \to 0} \frac{f(x + \Delta_x) - f(x)}{\Delta_x}$$

Higher Order Functions

Definition

A function that receives or returns functions

Example

$$\sum_{i=m}^{n} f(i) = \begin{cases} 0, & \text{if } m > n \\ x_m + \sum_{i=m+1}^{n} f(i), & \text{otherwise.} \end{cases}$$

$$\frac{d}{dx} f(x) = f'(x) = D_x f(x) = \lim_{\Delta_x \to 0} \frac{f(x + \Delta_x) - f(x)}{\Delta_x}$$

$$f \circ g = \lambda(x) f(g(x))$$

Parametric Curves

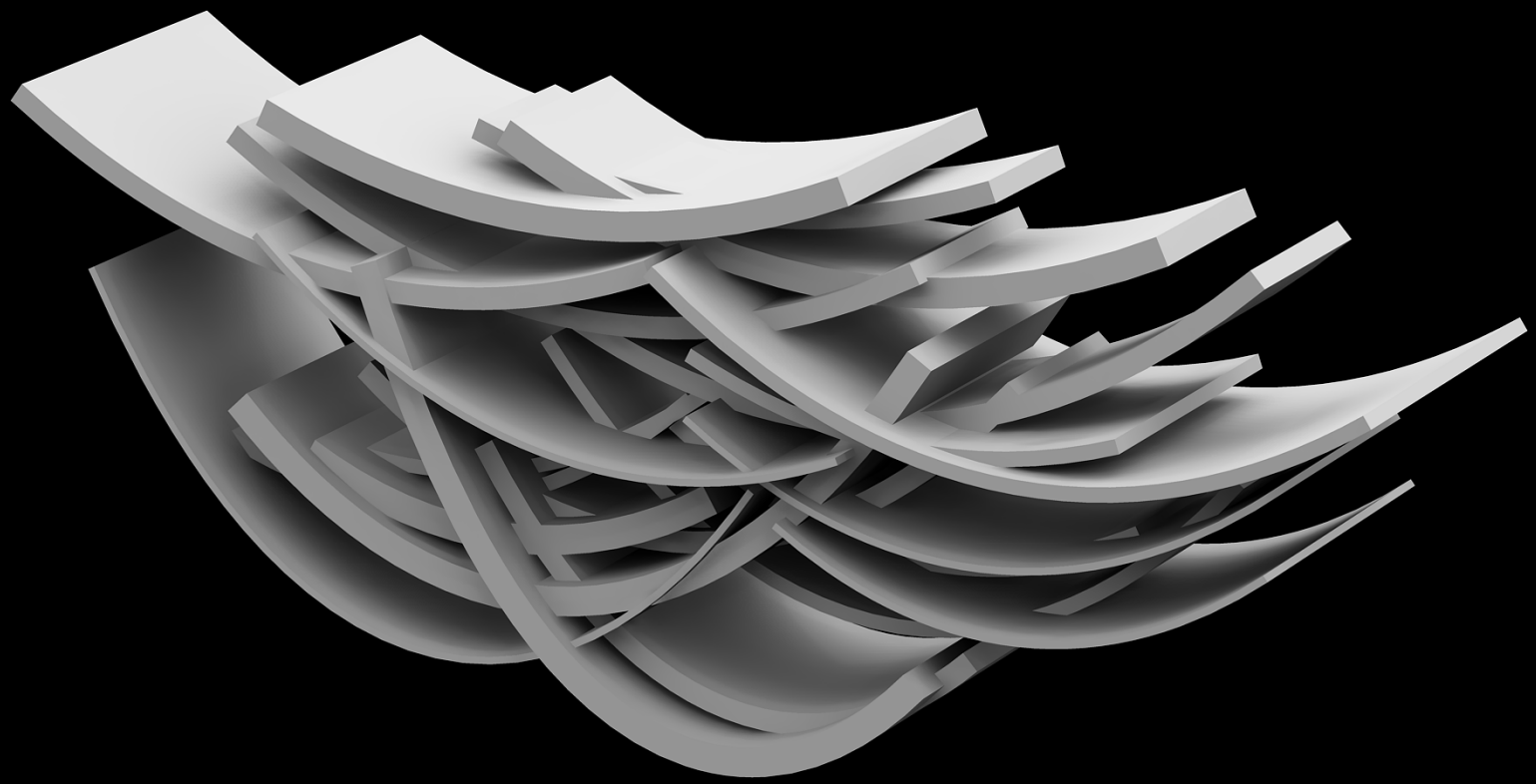
Catenary

$$y = a \cosh \frac{x}{a}$$

Parametric Curves

Catenary

$$y = a \cosh \frac{x}{a}$$



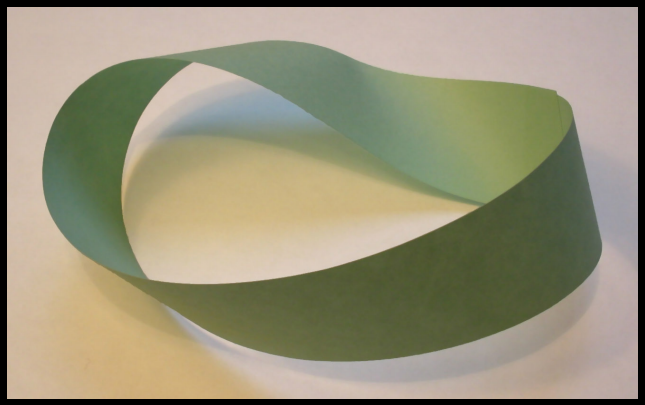
Parametric Curves

Catenary

$$y = a \cosh \frac{x}{a}$$

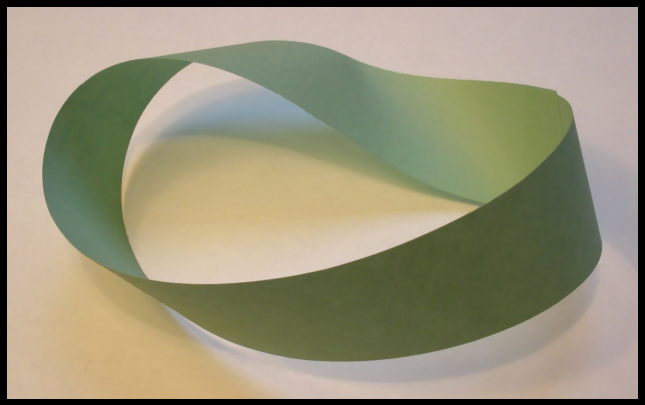
Parametric Surfaces

Möbius Strip



Parametric Surfaces

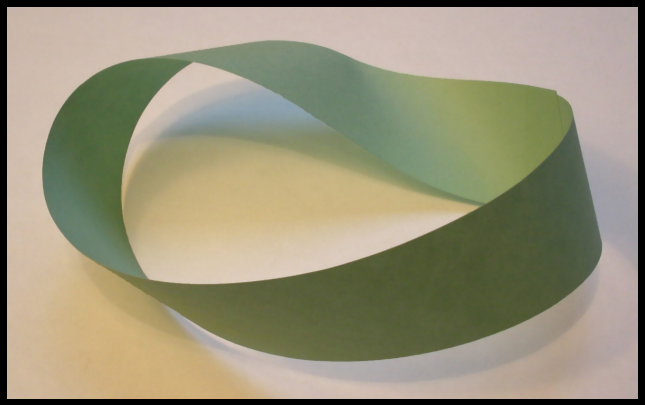
Möbius Strip



$$\begin{cases} \rho(u,v) = 1 + v \cos \dfrac{u}{2} \\ \theta(u,v) = u \\ z(u,v) = v \sin \dfrac{u}{2} \end{cases}$$

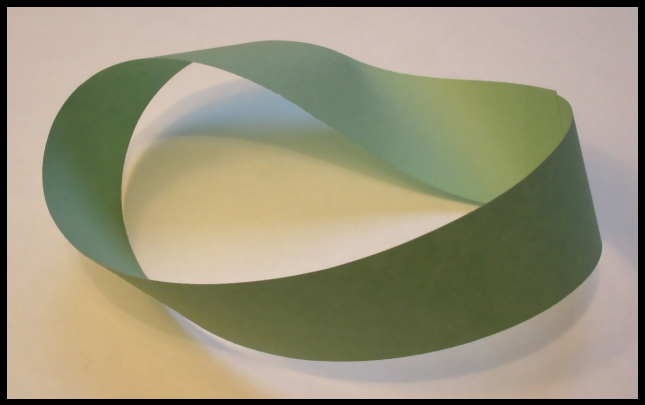
Parametric Surfaces

Möbius Strip



Parametric Surfaces

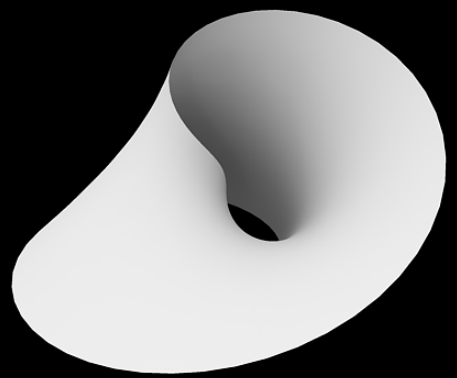
Möbius Strip



```
(defun moebius-strip (u0 u1 m v0 v1 n)
  (parametric-surface
    (lambda (u v)
      (cyl (+ 1 (* v (cos (/ u 2.0))))
           u
           (* v (sin (/ u 2.0)))))
    u0 u1 m v0 v1 n))
```

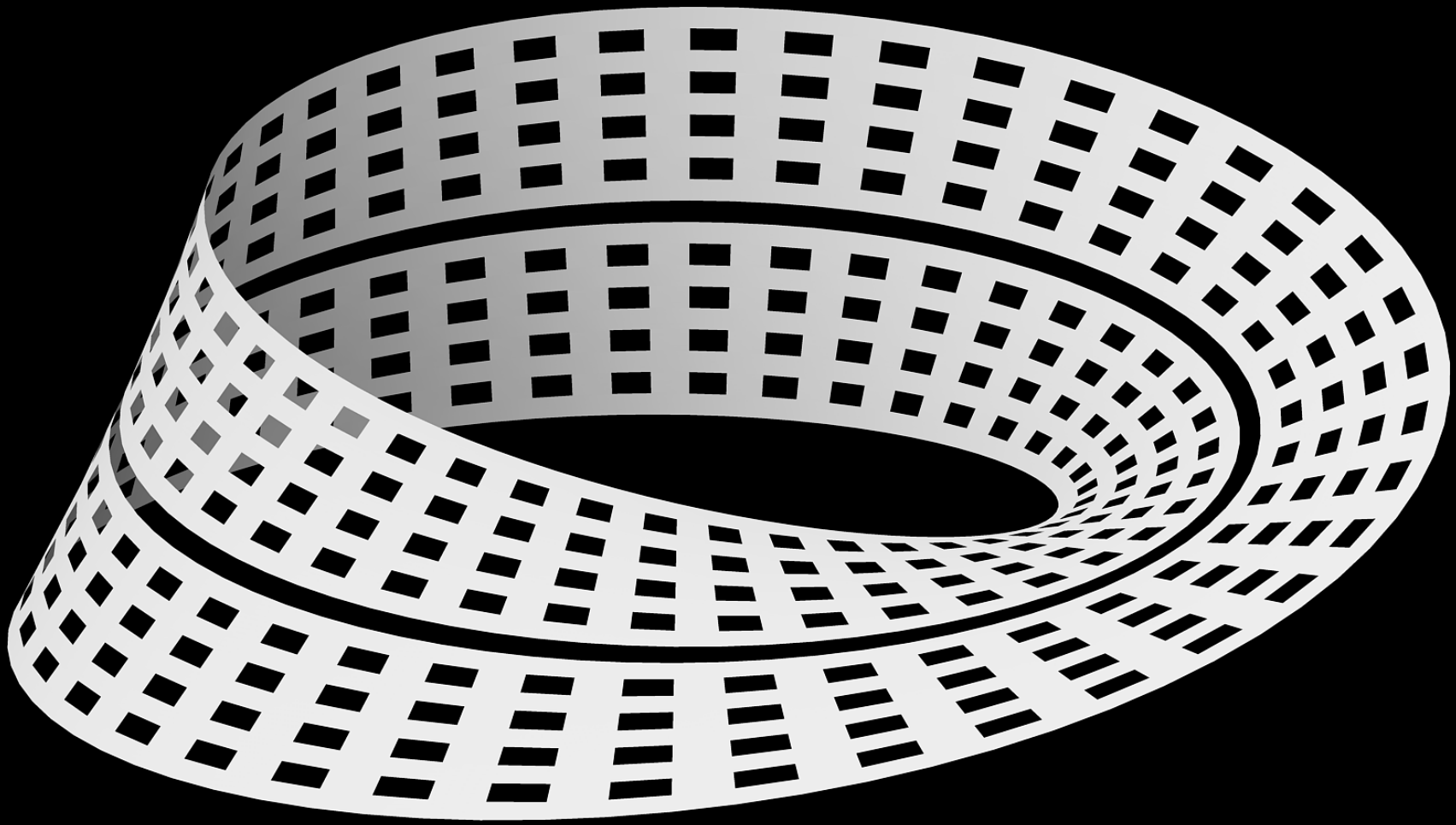
Parametric Surfaces

Möbius Strip



Parametric Surfaces

Möbius Strip

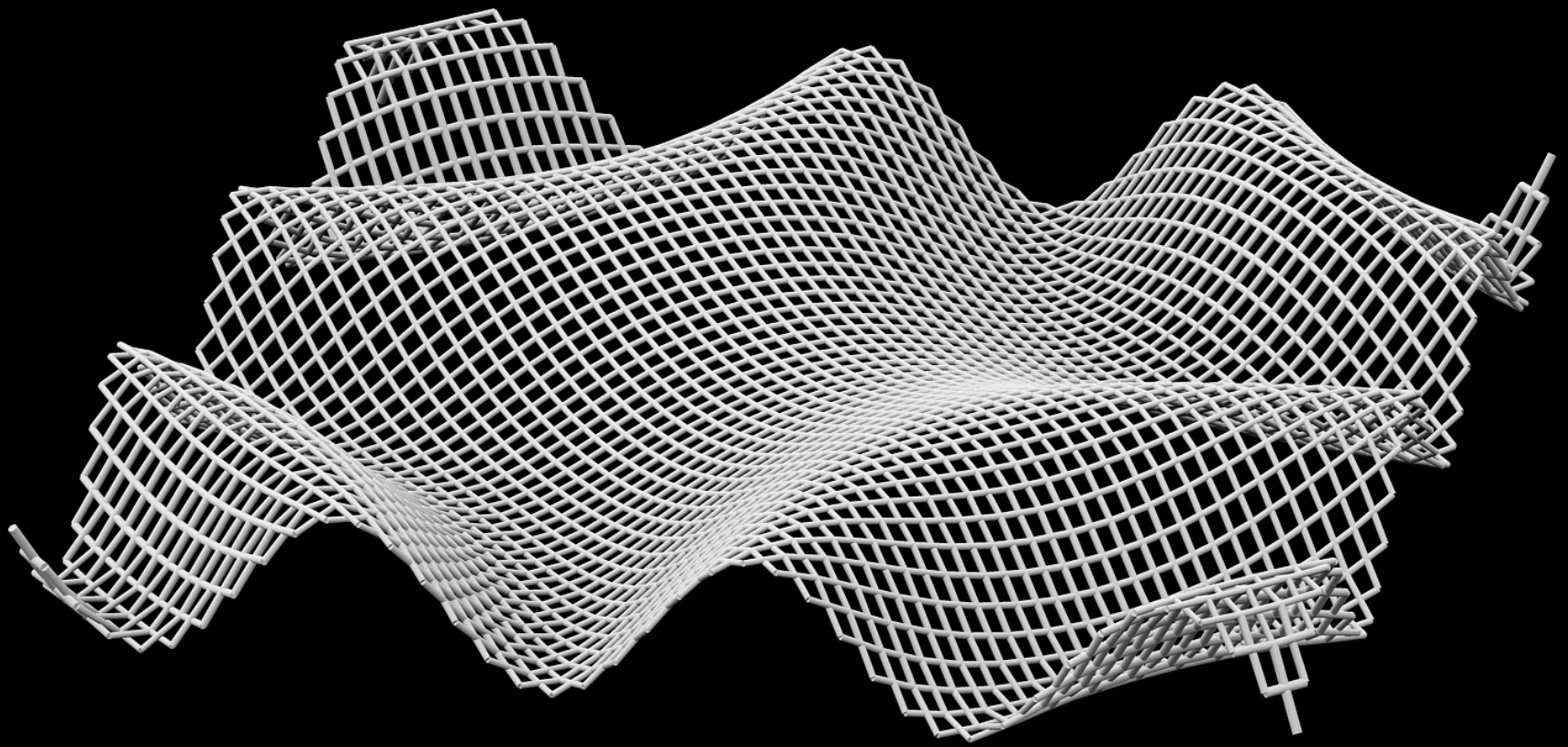


Parametric Surfaces

- Tesselations

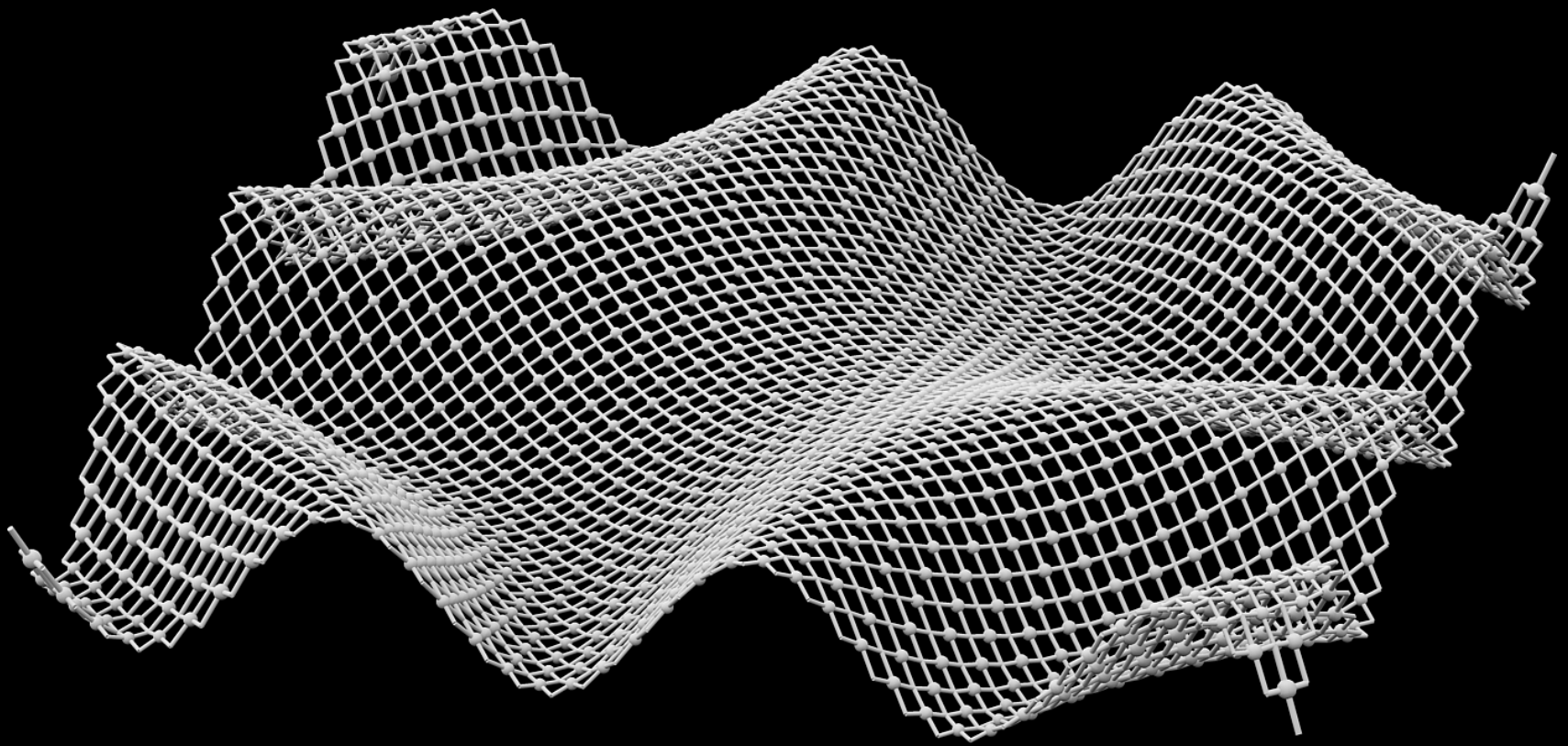
Parametric Surfaces

Tesselations



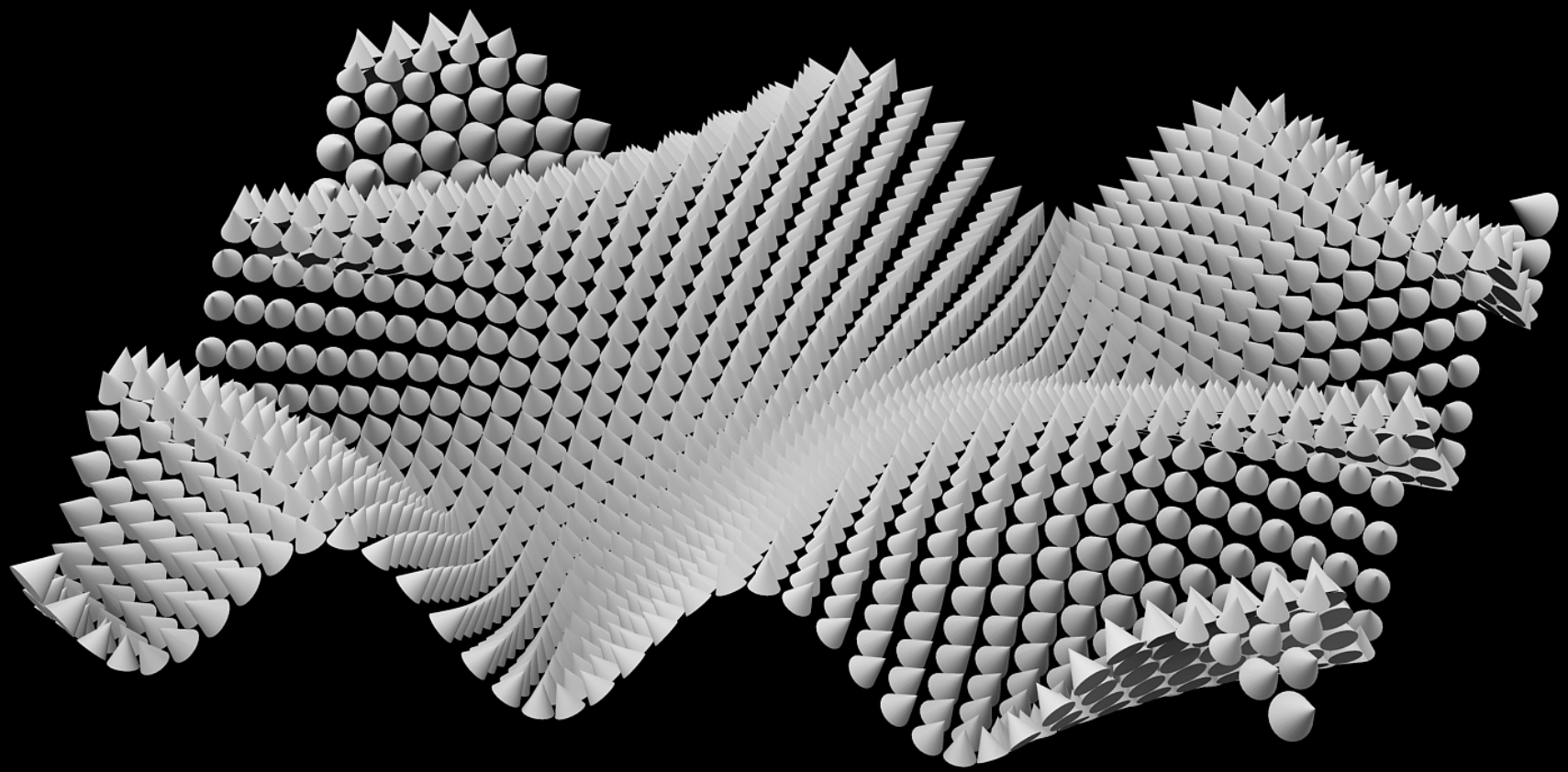
Parametric Surfaces

Tesselations



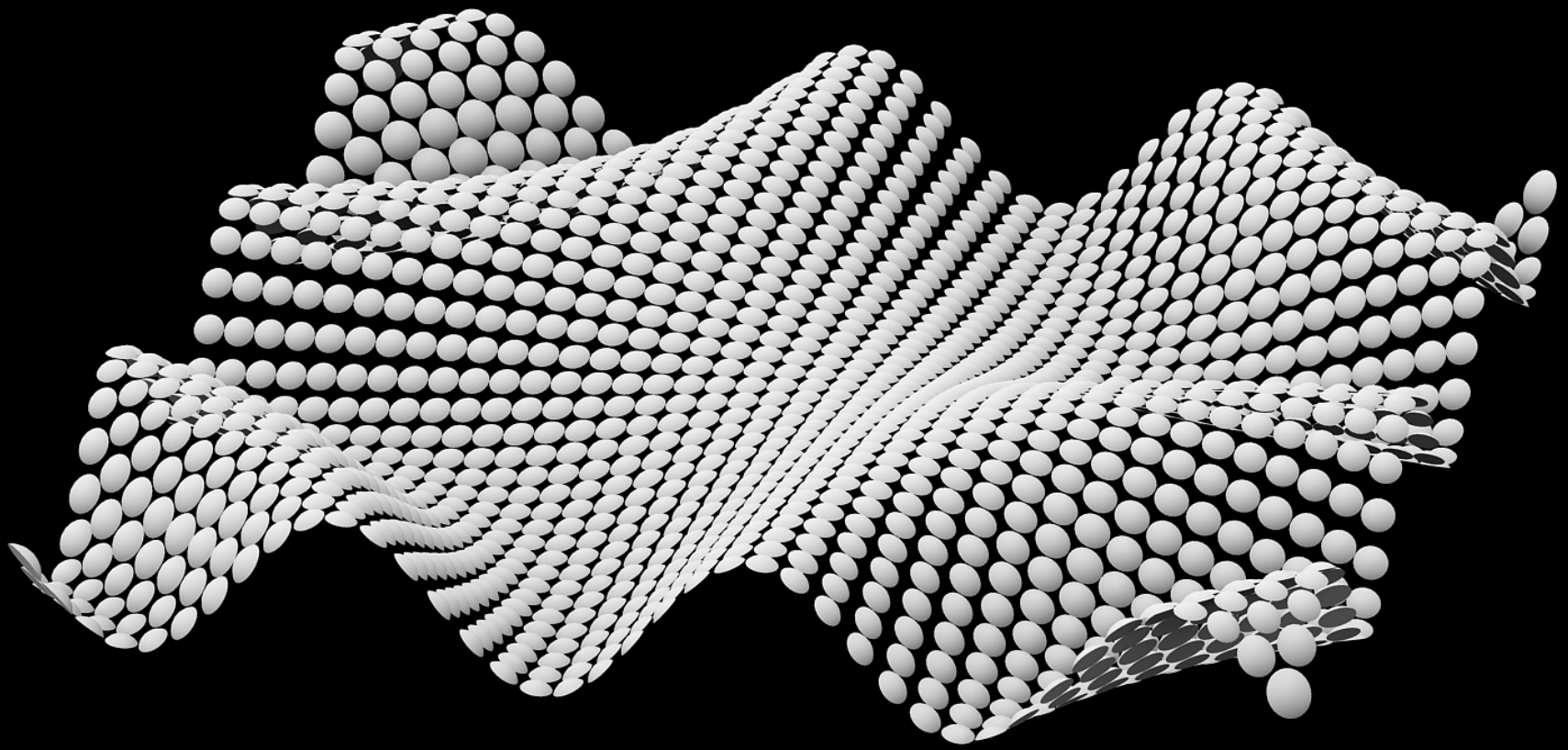
Parametric Surfaces

Tesselations



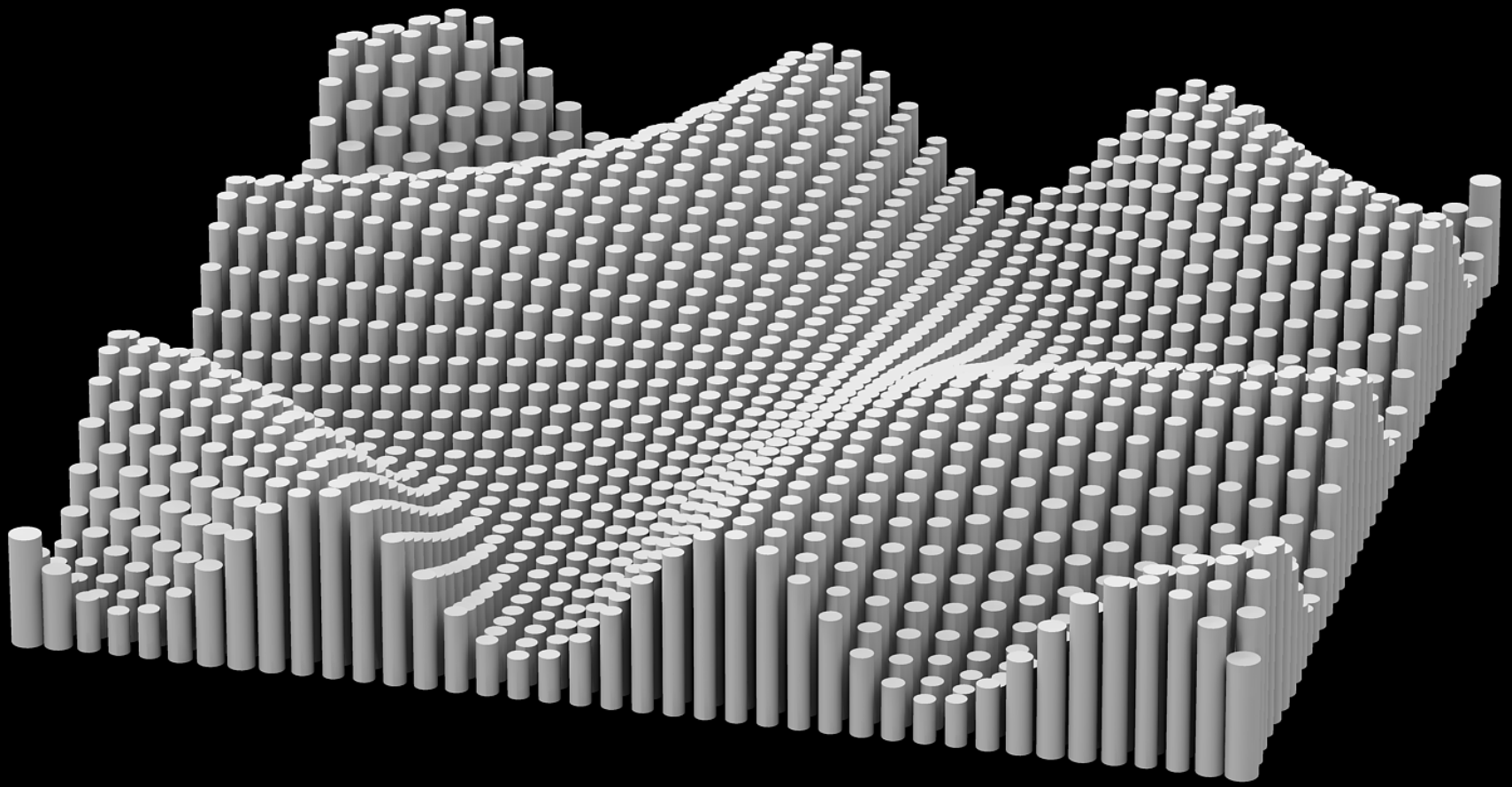
Parametric Surfaces

Tesselations



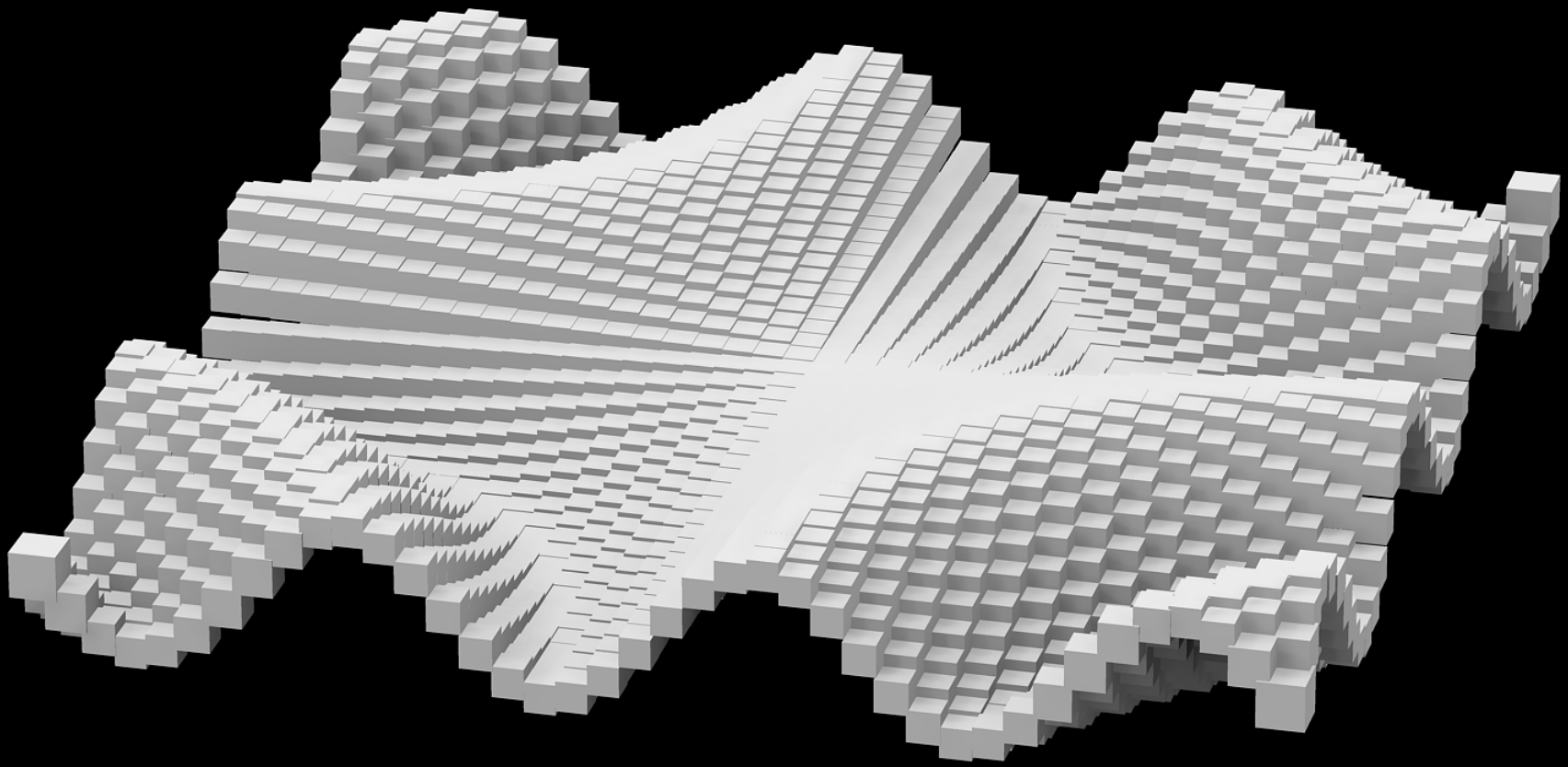
Parametric Surfaces

Tesselations



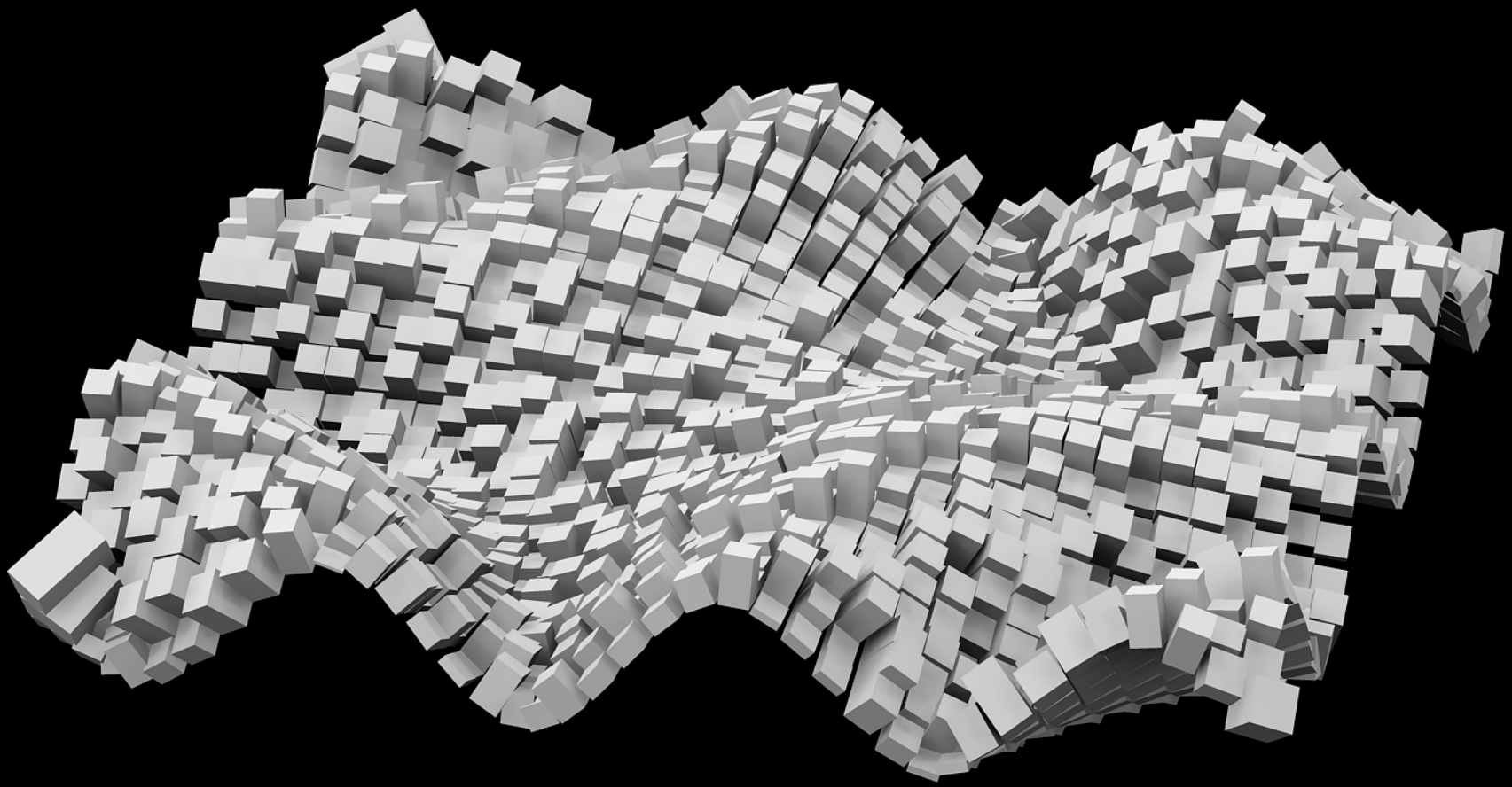
Parametric Surfaces

Tesselations



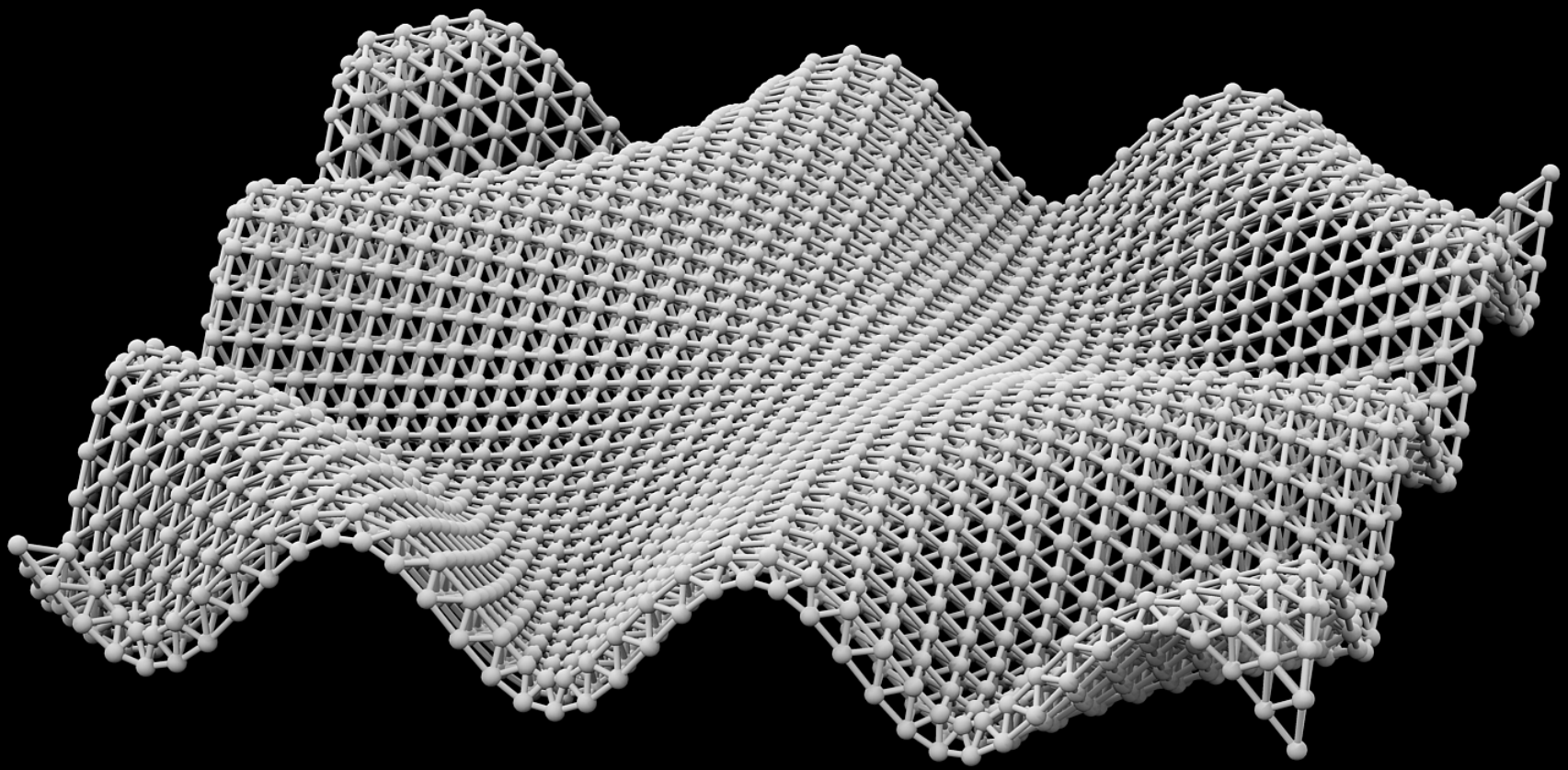
Parametric Surfaces

Tesselations



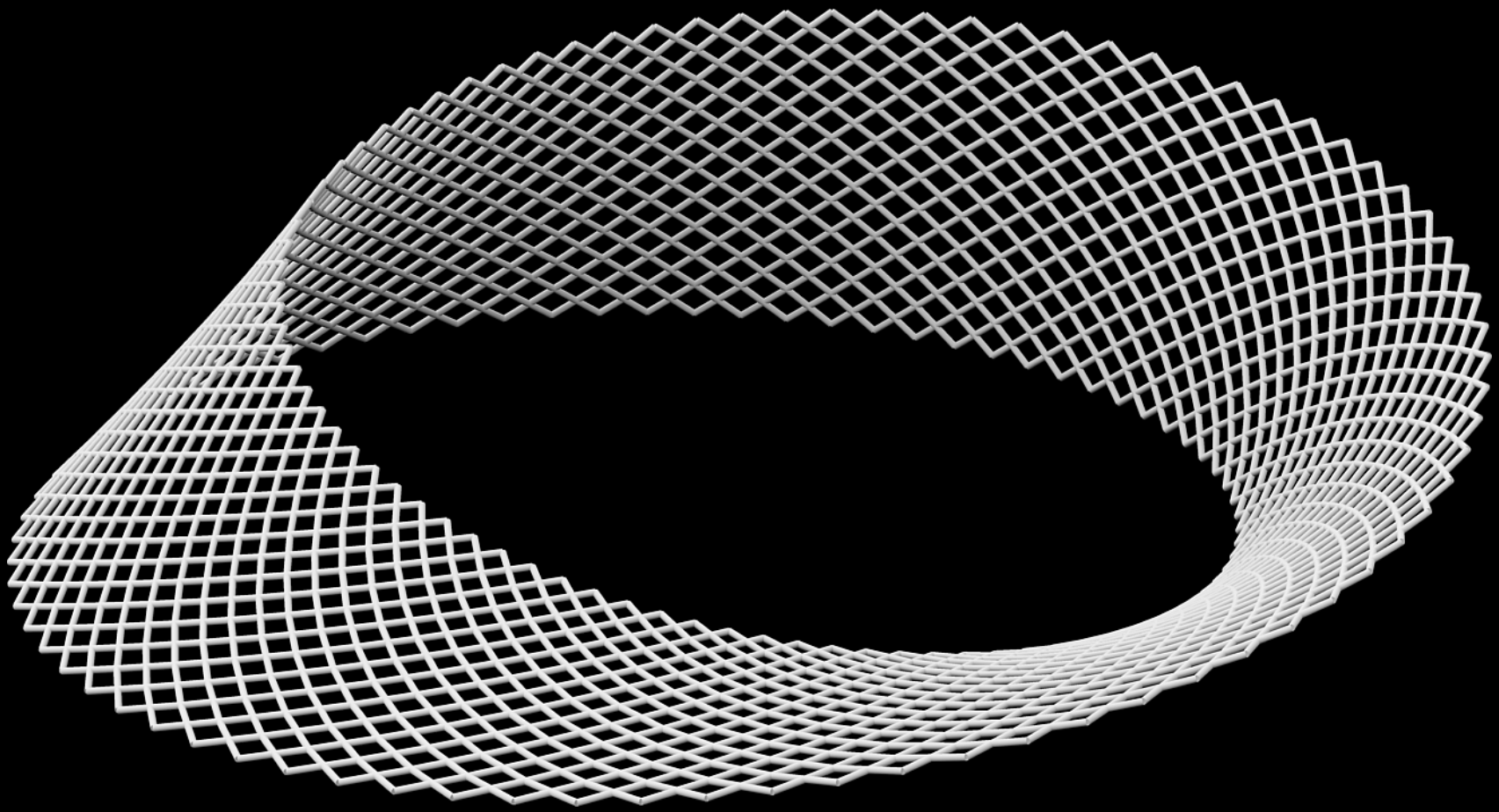
Parametric Surfaces

Tesselations



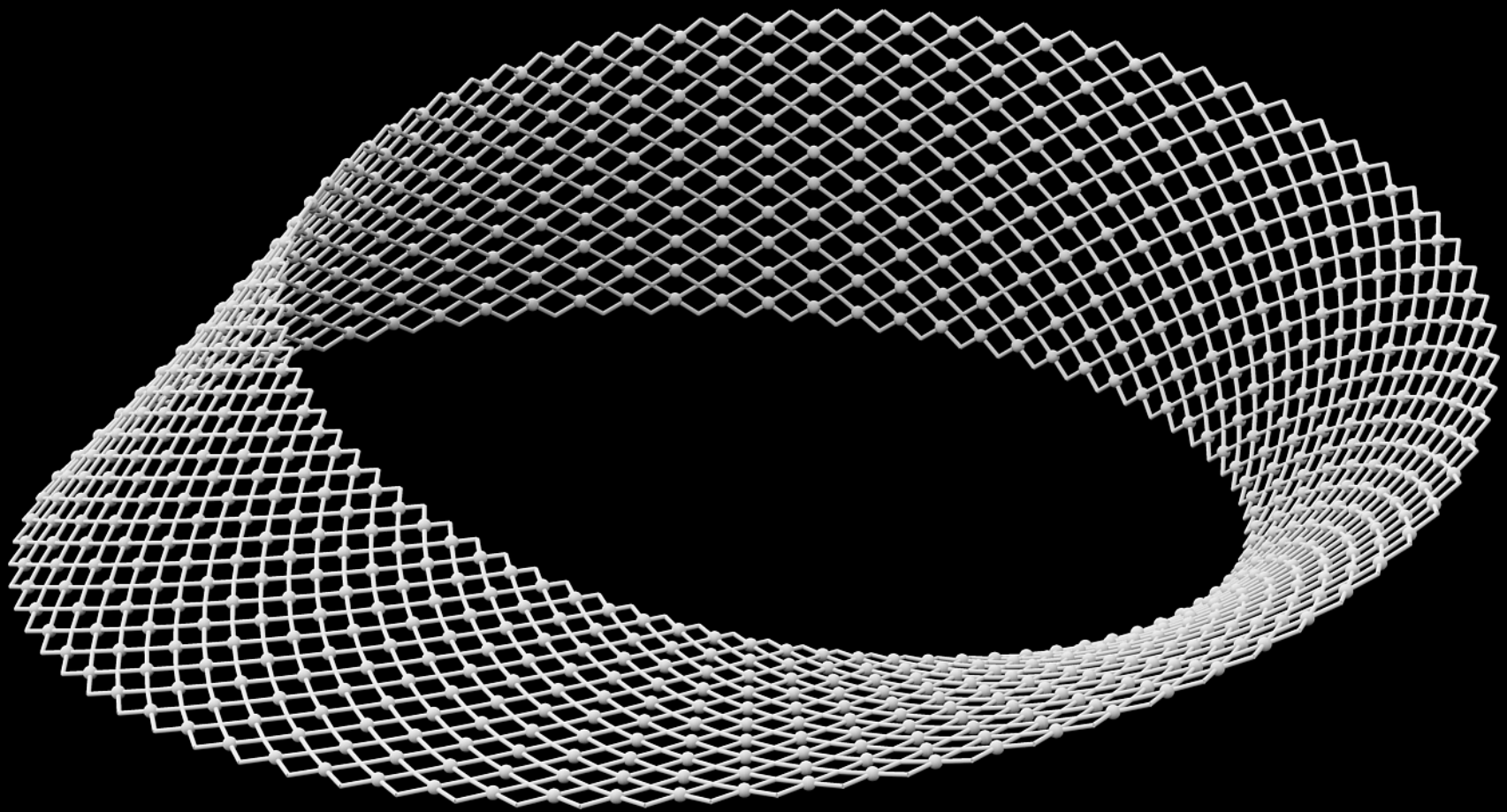
Parametric Surfaces

Tesselations



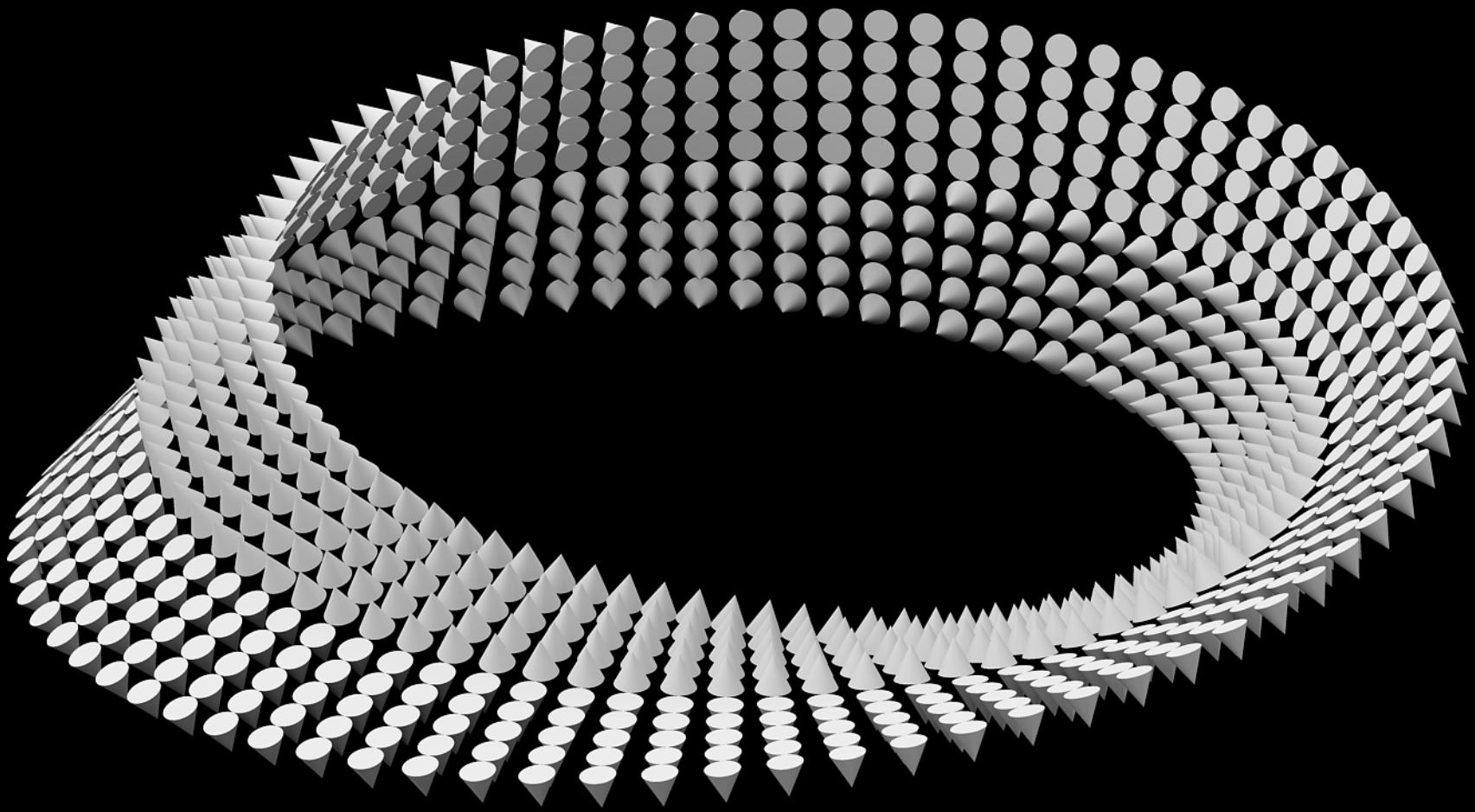
Parametric Surfaces

Tesselations



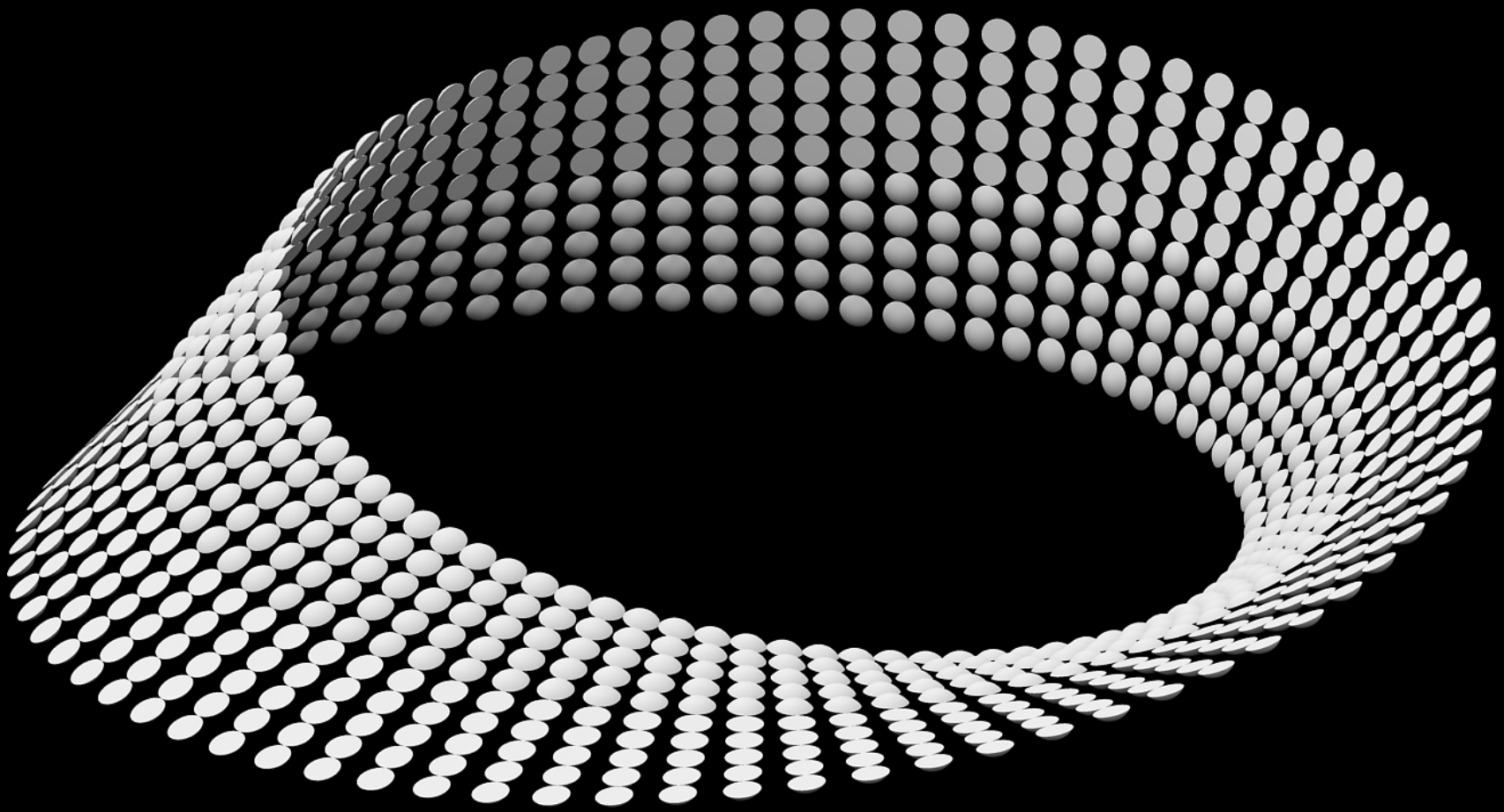
Parametric Surfaces

Tesselations



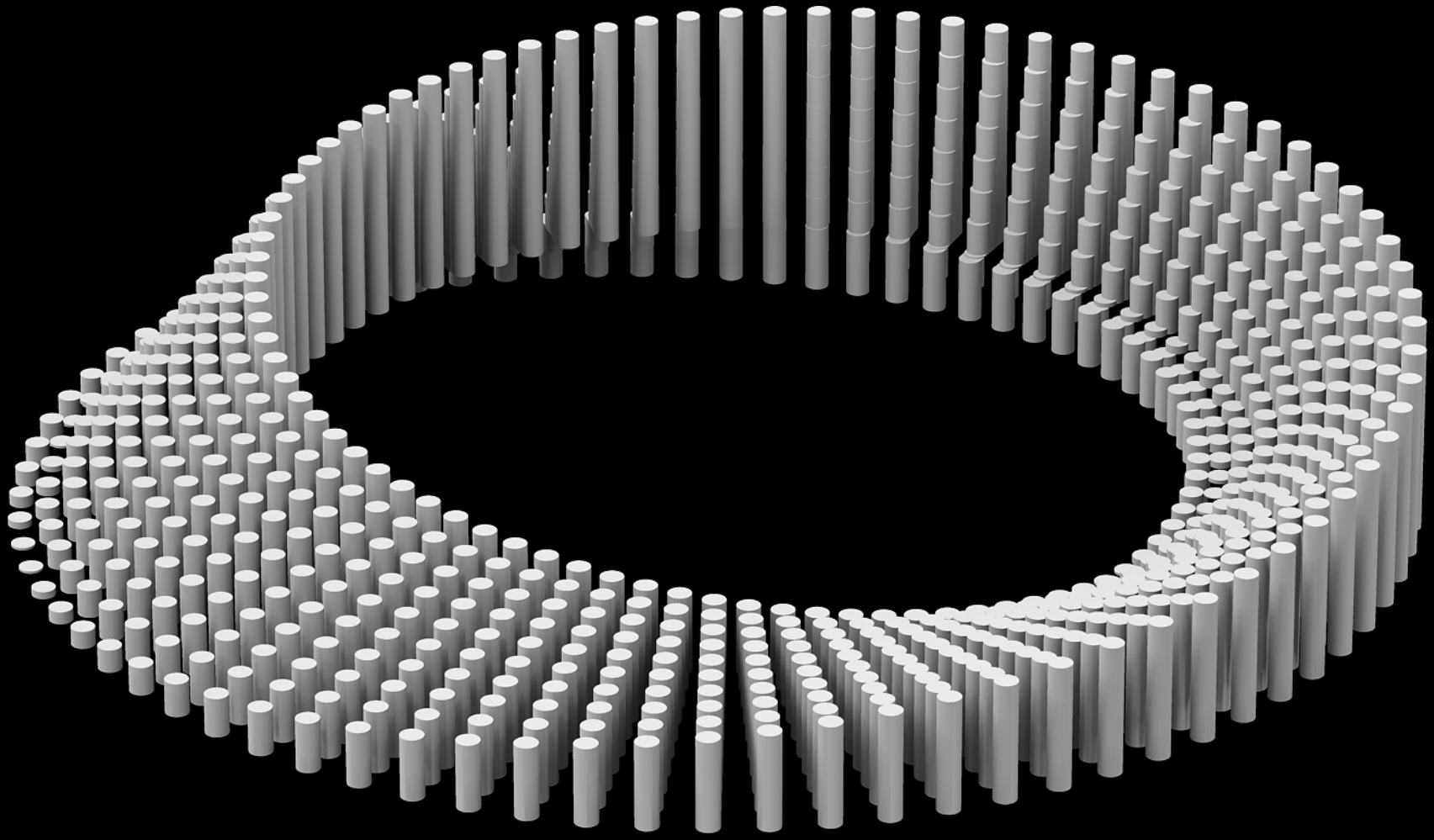
Parametric Surfaces

Tesselations



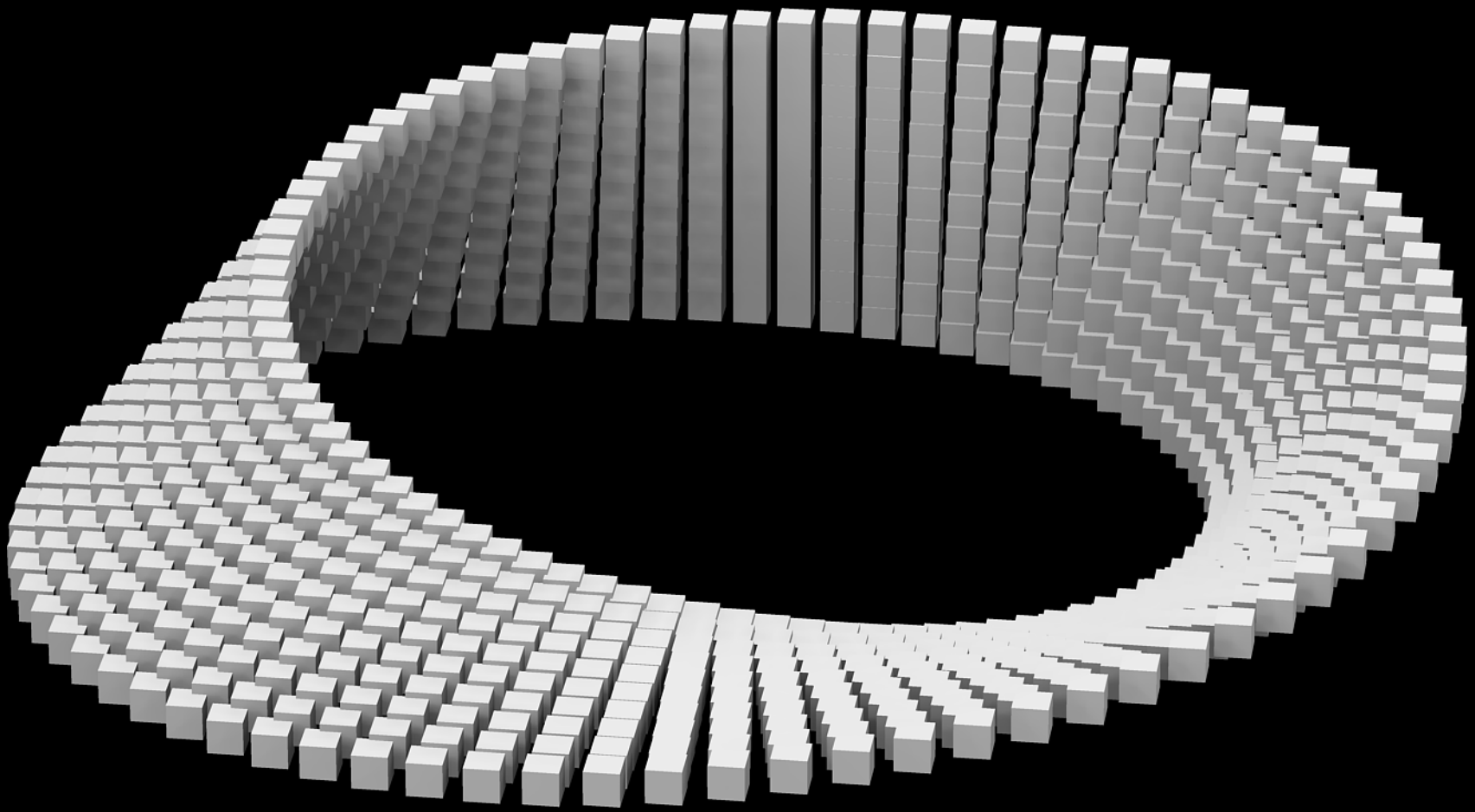
Parametric Surfaces

Tesselations



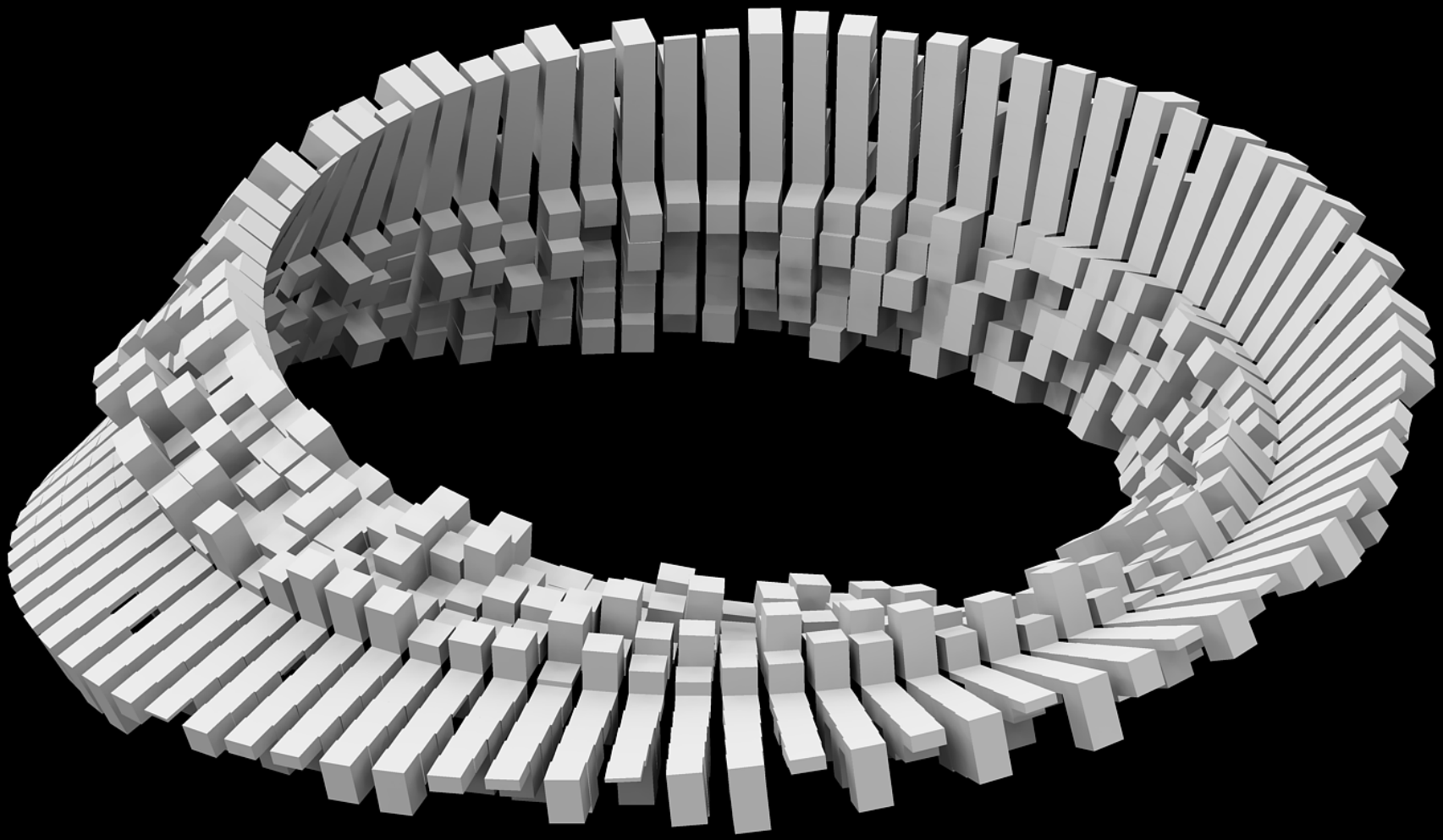
Parametric Surfaces

Tesselations



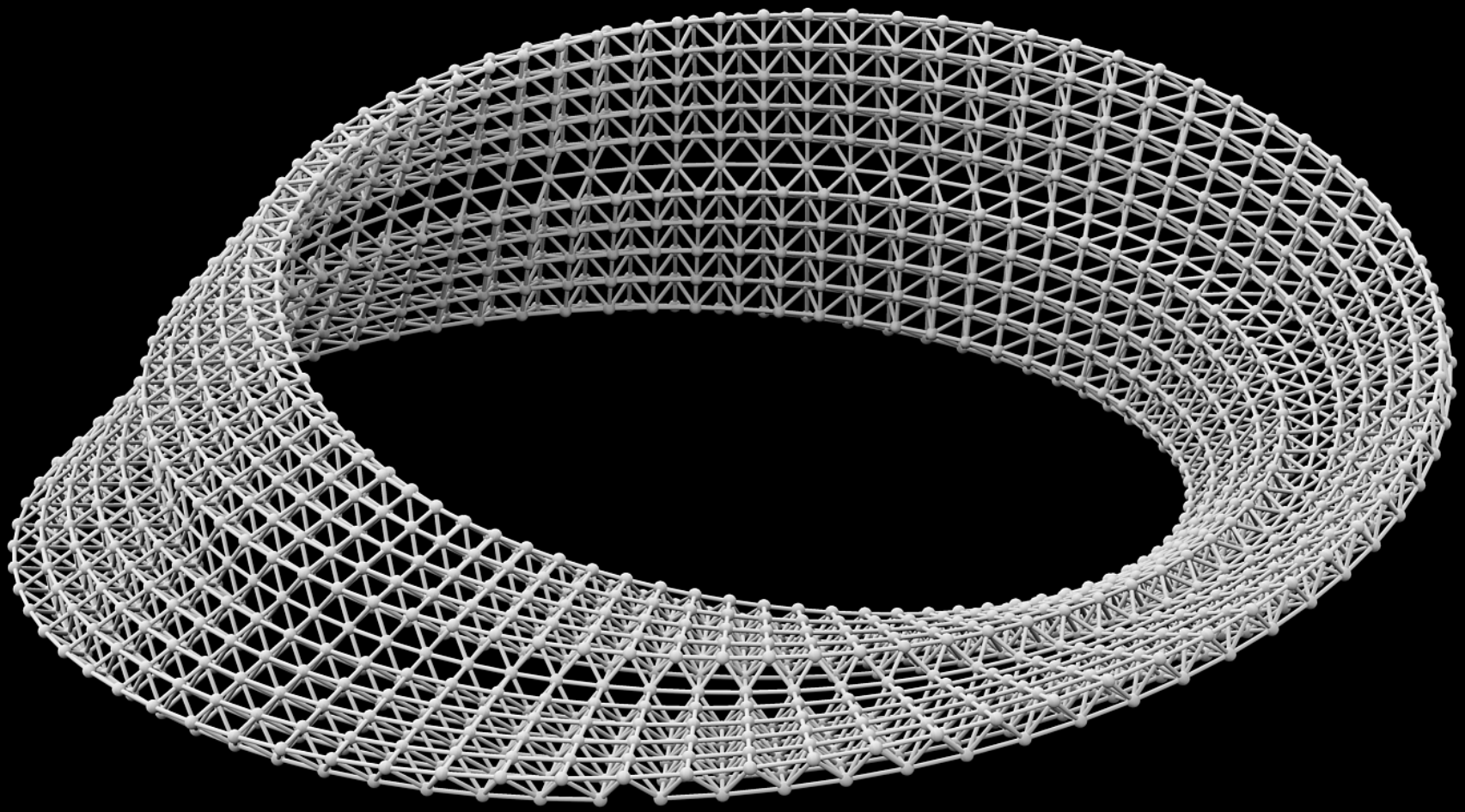
Parametric Surfaces

Tesselations



Parametric Surfaces

Tesselations



Teaching

Evaluation

- Project
- Exam
- Mini tests

Teaching - Exam

INSTITUTO SUPERIOR TÉCNICO

Instituto Superior Técnico

Programação e Computação para Arquitectura – 2010/2011

Segundo Exame – 31/02/2011

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 **8** páginas e a duração é de **120 minutos**. A cotação de cada questão encontra-se indicada entre parêntesis. Boa sorte.

Em todos os exercícios, pode usar todas as funções descritas na sebenta.

1. (2.0) Considere a seguinte definição Auto Lisp para a função de Ackermann:

```
(defun ack (m n)
  (if (= m 0)
    (+ n 1)
    (if (= n 0)
      (ack (- m 1) 1)
      (ack (- m 1) (ack m (- n 1))))))
```

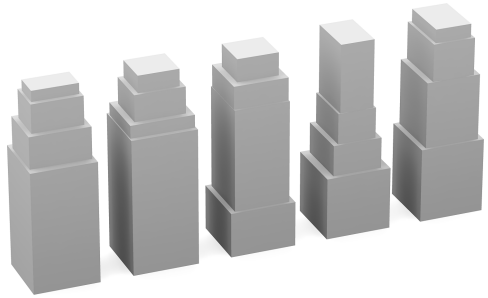
(a) (0.5) A função apresentada é recursiva? Porquê?

(b) (0.5) A função apresentada é de ordem superior? Porquê?

(c) (0.5) Calcule o resultado de `(ack 1 2)`.

(d) (0.5) Traduza a definição da função de Ackermann para a notação tradicional da matemática.

2. (2.0) Considere um modelo simplificado de um edifício baseado numa sequência de paralelipípedos empilhados, cuja dimensão é aleatória mas sempre sucessivamente mais pequena, tal como é ilustrado na seguinte imagem:

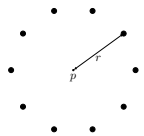


Considere que este modelo de edifício é parameterizado pelo número de paralelipípedos empilhados, pelas coordenadas do primeiro paralelipípedo e pelo comprimento, largura e altura do edifício. O bloco de base tem exactamente o comprimento e largura especificados mas a sua altura deverá estar entre 20% e 80% da altura total do edifício. Os blocos seguintes estão centrados sobre o bloco imediatamente abaixo e possuem um comprimento e largura que estão entre 70% e 100% dos parâmetros correspondentes no bloco imediatamente abaixo. A altura destes blocos deverá ser entre 20% e 80% da altura restante do edifício. Defina uma função capaz de produzir este modelo de edifícios.

Teaching - Exam

Número: _____ 3

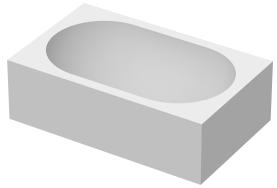
3. (1.0) Defina a função `pontos-circunferencia` que, a partir das coordenadas de um ponto p, de um raio r e de um número de pontos n, devolve **uma lista** com as coordenadas de n pontos uniformemente distribuídos ao longo da circunferência de raio r, tal como se ilustra na imagem seguinte para $n = 10$:



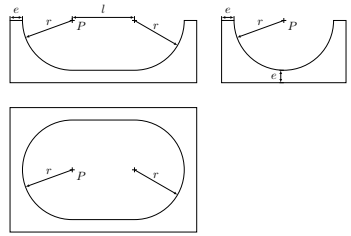
4. (2.0) Defina a função `pontos-sinusoide-circular`, de parâmetros p, r_i, r_e, c e n que computa **uma lista** com n pontos de uma curva fechada com a forma de uma sinusoide com c ciclos que se desenvolve num anel circular centrado no ponto p e delimitado pelos raios interior r_i e exterior r_e, tal como se pode ver nos vários exemplos apresentados na seguinte figura onde, da esquerda para a direita, o número de ciclos c é 12, 6, e 3.

Número: _____ 4

5. (1.5) Pretende-se modelar uma banheira de pedra idêntica à que se apresenta na imagem seguinte:



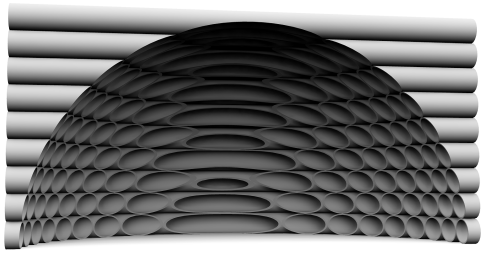
Os parâmetros relevantes para a banheira encontram-se representados no alçado frontal, planta e alçado lateral apresentados na imagem seguinte:



Defina uma função denominada `banheira` que constroi uma banheira idêntica à da figura anterior.

Teaching - Exam

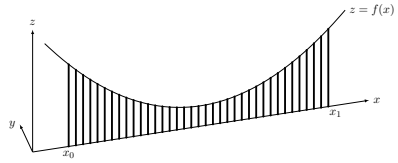
6. (2.5) Considere a imagem seguinte:



A imagem representa um abrigo de forma paralelipipédica construído com tubos cilíndricos cortados de modo a que o espaço interior tenha a forma de um quarto de esfera. Note que os tubos cilíndricos possuem uma espessura que é 10% do raio. Note ainda que o raio do quarto de esfera é igual à altura e largura do abrigo menos o raio de um cilindro.

Defina uma função que constrói o abrigo a partir do centro da esfera, da altura do paralelipípedo e do número de tubos a colocar ao longo da altura.

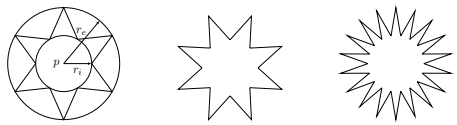
7. (2.0) Considere o seguinte esquema para os cabos verticais de uma ponte suspensa:



Os cabos verticais são modelados por cilindros de raio r (representados, no esquema, pelas linhas mais grossas) cujas extremidades são definidas por duas sequências com n pontos cada uma, uma ao longo do eixo x, a começar na coordenada $(x_0, 0, 0)$ e a acabar em $(x_1, 0, 0)$ e outra ao longo da curva $z = f(x)$, a começar em $(x_0, 0, f(x_0))$ e a acabar em $(x_1, 0, f(x_1))$.

Defina a função `cilindros-verticais` que recebe a função f, as abcissas x_0 e x_1, o raio r e o número de cilindros n e constrói o modelo apresentado.

8. (2.0) Defina uma função denominada `estrela` que, a partir de um ponto p, de um raio interior r_i, de um raio exterior r_e e de um número n de vértices no raio exterior, desenha estrelas como as apresentadas em seguida:

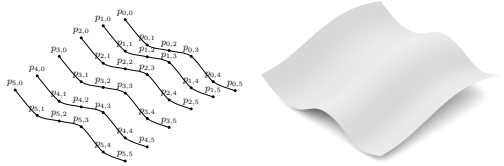


Teaching - Exam

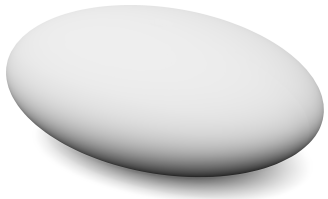
9. (1.5) Considere uma superfície representada por um conjunto de coordenadas tri-dimensionais. Admita que essas coordenadas estão armazenadas numa lista de listas da forma:

```
( (p0,0  p0,1  ...  p0,5)
  (p1,0  p1,1  ...  p1,5)
  ...
  (p5,0  p5,1  ...  p5,5) )
```

Defina a função `superficie-interpolacao` que recebe uma lista com a forma anterior, começa por criar uma lista de *splines* em que cada *spline* S_i passa pelos pontos $p_{i,0}, p_{i,1}, \ldots, p_{i,5}$ tal como se apresenta na imagem seguinte à esquerda e acaba por usar essa lista de splines $S_0, S_1, \ldots, S_5$ para fazer uma interpolação suave de secções, tal como se apresenta na imagem seguinte à direita:



10. (2.0) Considere o seguinte elipsóide:

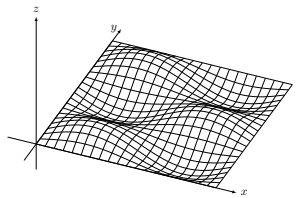


Um elipsoide é caracterizado pelas dimensões dos seus três raios ortogonais a, b, e c. A sua equação paramétrica é:

$$x = a \sin\psi \cos\phi$$
$$y = b \sin\psi \sin\phi$$
$$z = c \cos\psi$$
$$-\tfrac{\pi}{2} \leq \phi \leq +\tfrac{\pi}{2}; \quad -\pi \leq \psi \leq +\pi;$$

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.

11. (1.5) 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:

```
(parametrica-m-n
  (lambda (u v)
    (xyz u
         v

         _____))
  0 2*pi 20
  0 2*pi 20)
```

Teaching - Project

Projecto de
Programação e Computação para Arquitectura
Turning Torso

António Menezes Leitão

14 de Fevereiro de 2008

1 Introdução

O *Turning Torso* é um edifício desenhado pelo arquitecto espanhol Santiago Calatrava. A Figura 1 mostra uma imagem deste edifício. Com uma altura de 190 metros divididos em 54 pisos, o *Turning Torso* é o mais alto edifício da Suécia.

Figura 1: Turning Torso. Fotografia de Jerzy Kociatkiewicz.

O edifício é composto por nove "cubos" com cinco pisos cada. Cada cubo funciona como um edifício independente, sendo o espaço intermédio entre os cubos usado como espaço técnico para manutenção das fachadas.

Cada cubo sofre uma torção à medida que ganha altura sendo que o piso mais alto do edifício está rodado de noventa graus (no sentido dos ponteiros do relógio) relativamente ao piso mais baixo.

No exterior do edifício encontra-se um *exosqueleto* metálico, representado na Figura 2. Esta treliça é constituída por um tubo com braços horizontais e inclinados amarrados ao edifício. A treliça está torcida para acompanhar a

Teaching - Project

rotação do edifício. Este exosqueleto faz o travamento horizontal do edifício e destina-se a aumentar a resistência do edifício aos ventos e a diminuir a sua vibração.

Figura 2: Turning Torso. Fotografia de Umberto Luparelli.

A estrutura de suporte do edifício é constituída por um núcleo cilíndrico de betão reforçado a aço que alberga os elevadores e escadas de serviço bem como os sistemas eléctricos e hidráulicos. O núcleo tem um diâmetro de 10.6 metros, constante deste a base até ao topo, embora a sua espessura interior varie desde 2.5 m na base até 0.40 m no topo. Este cilindro é visível na Figura 3.

A fachada do *Turning Torso* é revestida com 2800 painéis de aluminio e 2250 janelas. Para acompanhar a torção, os painéis são inclinados para dentro no lado oeste do edifício e para fora no lado leste, variando a inclinação entre 0º e 7º. A inclinação lateral dos paineis é de 6º.

Figura 3: Turning Torso. Fotografia de Lars Tufvesson.

Cada piso consiste de uma laje de secção rectangular arredondada acoplada a uma secção triangular. As lajes envolvem e apoiam-se no núcleo central de suporte cujo eixo coincide com o eixo de rotação das lajes, tal como se apresenta na Figura 4. Nas extremidades das lajes encontram-se colunas de betão armado inclinadas de modo a acompanhar a torção do edifício.

O edifício é de utilização mista, com os dois primeiros cubos reservados para escritórios e os restantes para habitação, com um total de 147 apartamentos distribuídos por trinta e três diferentes tipologias, com áreas desde 45 a 190 metros quadrados.

Para mais informações sobre o edifício sugere-se a consulta dos seguintes sites:

- http://www.turningtorso.com/
- http://www.designbuild-network.com/projects/turning-torso/
- http://www.arcspace.com/architects/calatrava/torso2/torso2.html
- http://www.revistaau.com.br/arquitetura-urbanismo/154/artigo39463-1.asp
- http://www.arcspace.com/architects/calatrava/turning_torso_index.html
- http://www.construlink.com/Homepage/verDestaqueArquitectura.php?id=3

Teaching - Project

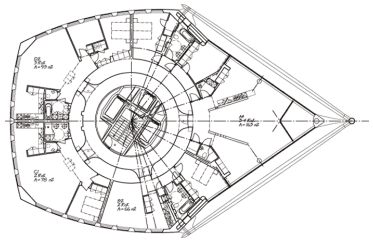


Figura 4: Turning Torso. Planta de um piso.

2 Trabalho a Desenvolver

O projecto é para ser realizado em grupos de dois alunos ou, excepcionalmente, de um só aluno.

O projecto consiste na escrita de um programa Auto Lisp capaz de gerar o *Turning Torso*. O seu programa deverá ser suficientemente parameterizado para que se possam explorar variações em torno das ideias fundamentais da obra de Calatrava. Em particular, deverá ser possível experimentar variações nos seguintes parâmetros:

- número de pisos por bloco,
- número de blocos no edifício,
- ângulo de rotação de cada piso,
- altura de cada piso,
- dimensões do núcleo,
- dimensões das paredes exteriores,
- curvatura das paredes exteriores,
- dimensões das barras da treliça

Naturalmente, importa que todos os parâmetros definidores da geometria do edifício estejam bem identificados para que seja possível alterá-los facilmente.

Para permitir uma avaliação intercalar do projecto, este será decomposto em duas fases com as seguintes datas de entrega:

Figura 5: Turning Torso. Fotografia de Wanshan.

1. Primeira fase: a entregar até às 24:00 do dia 9 de Dezembro de 2007.
2. Segunda fase: a entregar até às 24:00 do dia 21 de Dezembro de 2007.

O trabalho a entregar em cada fase é detalhado em seguida.

2.1 Primeira Fase

Nesta fase pretende-se que implemente um programa Auto Lisp capaz de gerar os piso do edifício, sem necessidade de incluir quaisquer elementos internos excepto o núcleo cilíndrico de suporte. Este núcleo deverá obedecer à geometria real do edifício. Os pisos deverão ser suficientemente parameterizáveis para permitir experimentar variações na sua forma. Não é necessário mas é recomendável que a geração dos pisos possa afectá-los de um determinado ângulo de rotação.

Até ao final do prazo de entrega desta fase deverá aceder à página da cadeira no Fénix para submeter o seu trabalho por via electrónica. O trabalho a submeter deverá ser composto por um arquivo `ZIP` ou `tar.gz` contendo:

Teaching - Project

Figura 6: Turning Torso. Fotografia de Lisa Liu.

- um ou mais ficheiros Auto Lisp com o código desenvolvido para o projecto.
- uma ou mais imagens (`PS`, `EPS`, `PDF`, `PNG`, `JPG`, etc.) com vistas do edifício gerado.

O código desenvolvido deverá estar escrito num estilo que facilite a leitura e que dispense, na medida do possível, excessivos comentários. Estes deverão ser incluídos, não para dizerem o que o código já diz claramente, mas para documentar os módulos e funções principais e, eventualmente, algumas partes menos claras dos programas.

Esta entrega será avaliada mas não será atribuída qualquer nota, servindo apenas para informar os alunos do andamento dos seus trabalhos e para esclarecer quaisquer dúvidas que tenham quando ao sucesso ou insucesso das abordagens escolhidas.

Figura 7: Turning Torso. Fotografia de John Liu.

2.2 Segunda Fase

Na segunda fase dever-se-á ir tão longe quanto possível na modelação realista do *Turning Torso*. Em particular deverá ser possível modelar as paredes do edifício com os paineis de alumínio e suas janelas e deverá incluir-se a treliça de travamento.

Até ao final do prazo de entrega desta fase deverá aceder à página da cadeira no Fénix para submeter o seu trabalho por via electrónica. O trabalho a submeter deverá ser composto por um arquivo `ZIP` ou `tar.gz` contendo:

- um ou mais ficheiros Auto Lisp com o código desenvolvido para o projecto.
- um documento PDF com um relatório do projecto.

2.2.1 Código

O código desenvolvido deverá estar escrito no melhor estilo que for possível, permitindo a sua fácil leitura e dispensando excessivos comentários. É sempre preferível ter código mais claro com poucos comentários do que ter código obscuro com muitos comentários.

O código deverá ser modular, divido em funções com responsabilidades específicas e reduzidas. Cada módulo deverá ter um curto comentário a descrever o seu objectivo.

O código será testado pelo corpo docente pelo que deverá incluir informação sobre qual a função principal e qual o significado dos seus parâmetros.

Teaching - Project

Figura 8: Turning Torso. Fotografia de Kerstin Andreasson.

2.2.2 Relatório

O relatório do projecto consiste de um documento PDF com uma descrição dos trabalhos desenvolvidos, explicando as decisões de programação tomadas e ilustrado com imagens geradas pela execução do programa, em particular, imagens que demonstrem a versatilidade do mesmo para a geração de geometrias alternativas para o edifício.

3 Avaliação

Os critérios de avaliação incluem:

- A qualidade das soluções desenvolvidas.
- A clareza dos programas desenvolvidos.
- A capacidade de geração de geometrias alternativas.

- A qualidade do relatório.

Em caso de dúvidas, o corpo docente poderá exigir explicações sobre o funcionamento do projecto desenvolvido, incluíndo eventuais demonstrações.

4 Plágio

Considera-se plágio o uso de quaisquer fragmentos de programas que não tenham sido fornecidos pelos docentes da disciplina. Não se considera plágio o uso de ideias cedidas por terceiros.

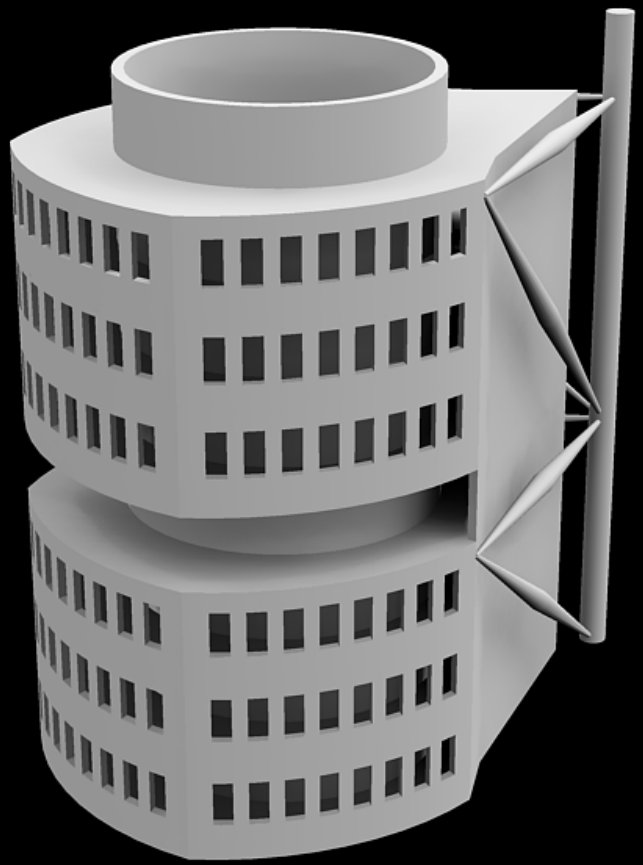
Esta disciplina segue normas muito rígidas relativamente ao plágio. Quaisquer projectos que sejam considerados plagiados serão anulados, independentemente de quem plagiou e de quem tiver sido plagiado, independentemente de o plágio ter sido autorizado, ou não, pela parte plagiada.

Isto não deverá ser impedimento para a troca salutar de ideias e para a normal camaradagem e entreajuda que deve existir entre colegas. Contudo, sugere-se que nunca cedam fragmentos de programas sob pena de quem os recebe não os entender e se limitar a plagiá-los com maior ou menos esforço de "camuflagem."

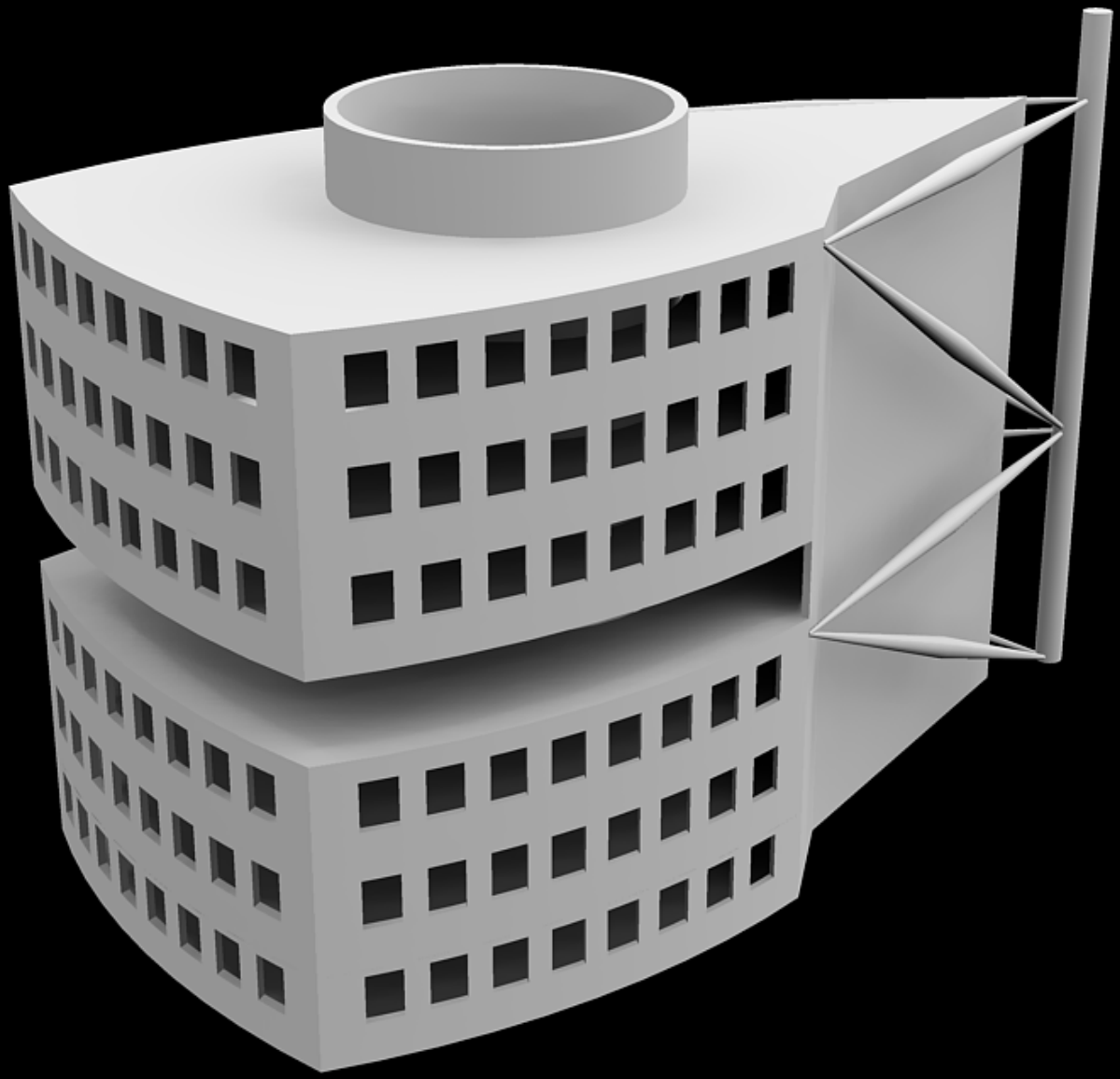
5 Notas Finais

Não se esqueça da Lei de Murphy.

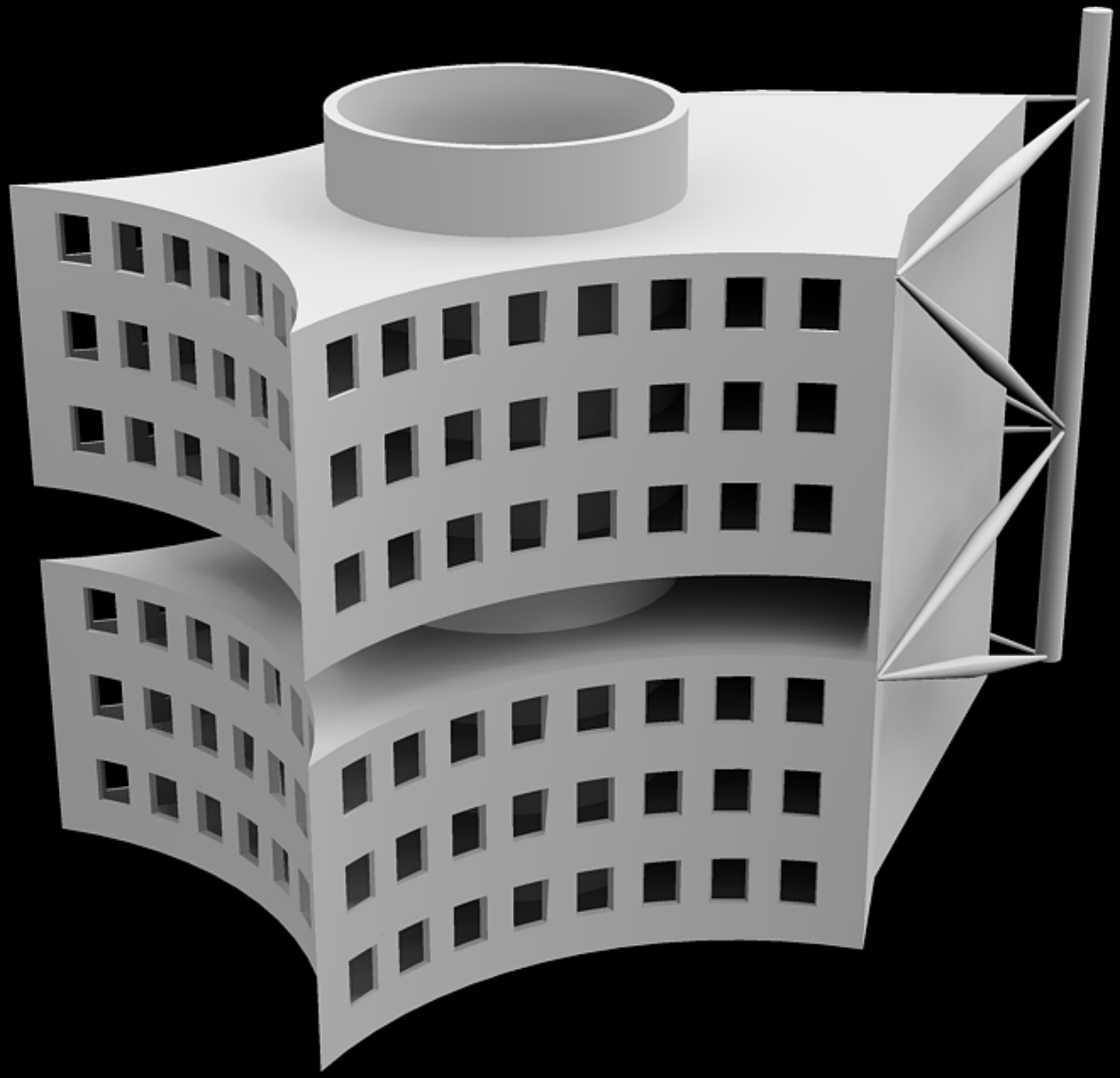
Results



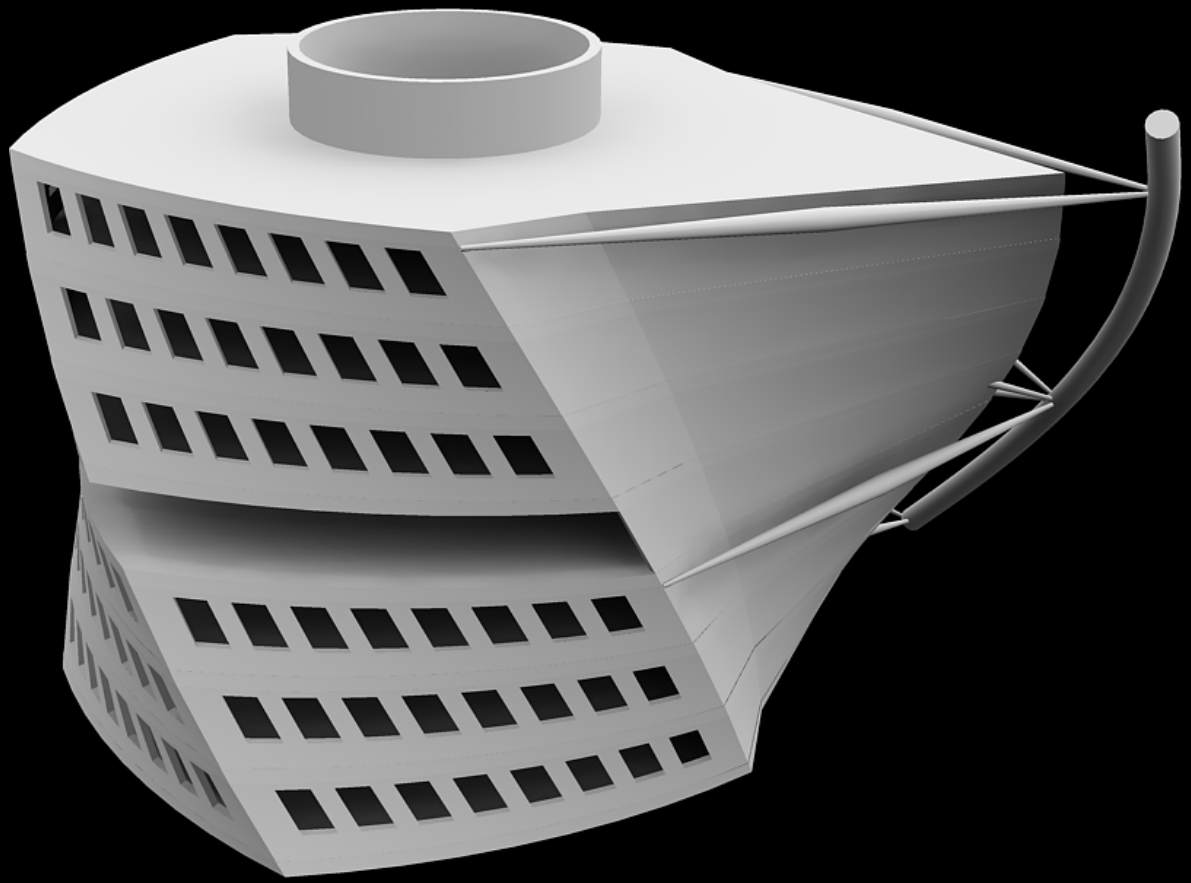
Results



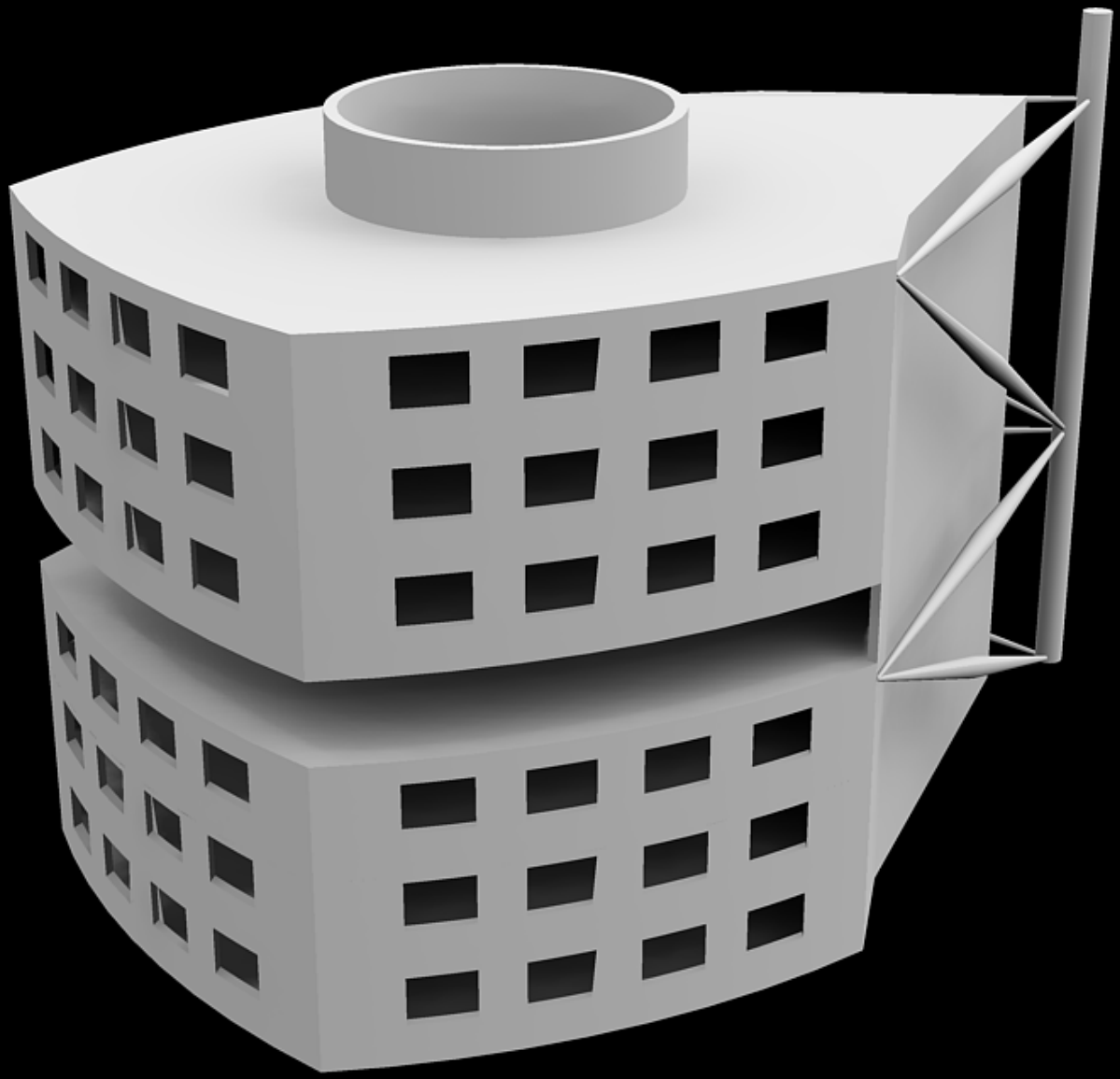
Results



Results



Results



Results

Practice

Different Goals

- Automation
- Generative Design
- Research

Common Needs

- CAD Application
- Programming Language

Practice

Alternatives

- ArchiCad
  - GDL
- AutoCAD
  - Visual Basic
  - C++
  - AutoLisp
- Rhino
  - RhinoScript
  - Grasshopper
  - Python
- Microstation
  - VBA
  - C++
  - Java

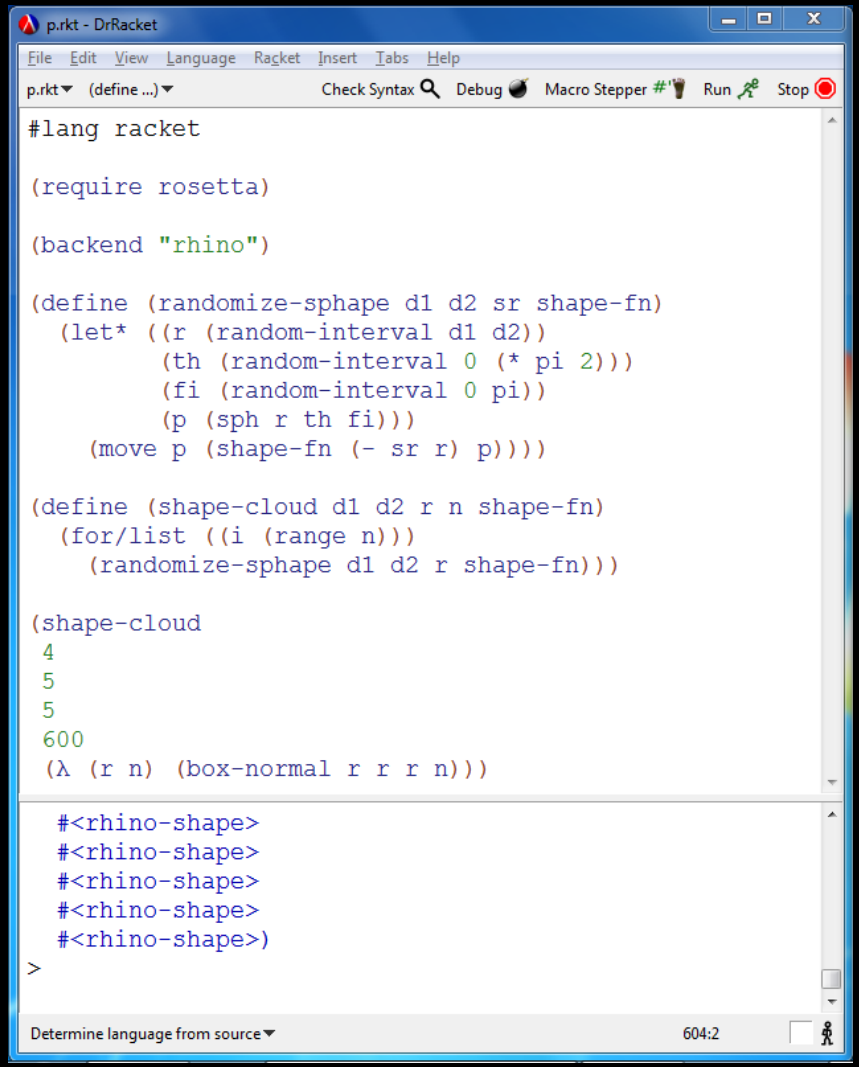
Practice

Alternatives

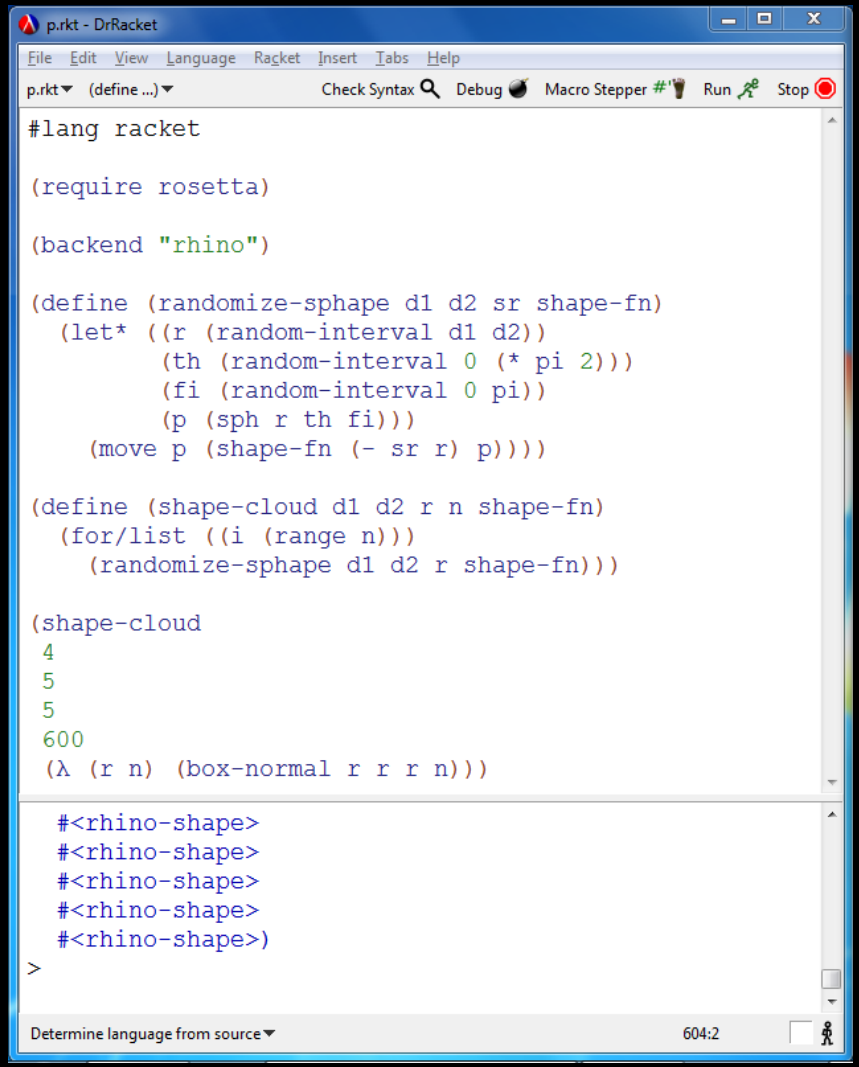
- ArchiCad
  - GDL
- AutoCAD
  - Visual Basic
  - C++
  - AutoLisp
- Rhino
  - RhinoScript
  - Grasshopper
  - Python
- Microstation
  - VBA
  - C++
  - Java

Incompatible

Rosetta


p.rkt - DrRacket
File Edit View Language Racket Insert Tabs Help
p.rkt (define ...) Check Syntax Debug Macro Stepper Run Stop
#lang racket
(require rosetta)
(backend "rhino")
(define (randomize-sphape d1 d2 sr shape-fn)
(let* ((r (random-interval d1 d2))
(th (random-interval 0 (* pi 2)))
(fi (random-interval 0 pi))
(p (sph r th fi)))
(move p (shape-fn (- sr r) p))))
(define (shape-cloud d1 d2 r n shape-fn)
(for/list ((i (range n)))
(randomize-sphape d1 d2 r shape-fn)))
(shape-cloud
4
5
5
600
(λ (r n) (box-normal r r r n)))
#<rhino-shape>
#<rhino-shape>
#<rhino-shape>
#<rhino-shape>
#<rhino-shape>)
>
Determine language from source
604:2

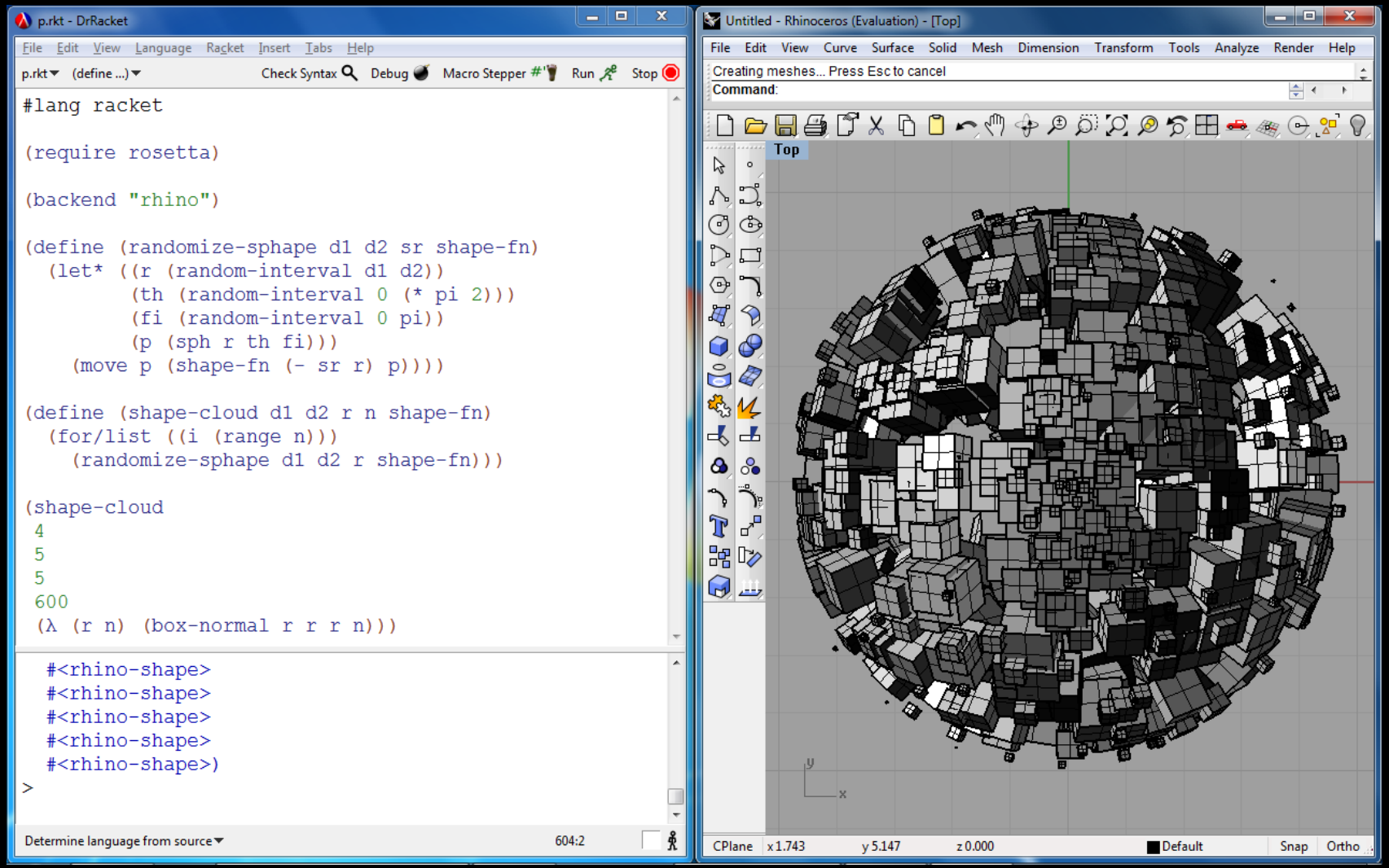
Rosetta



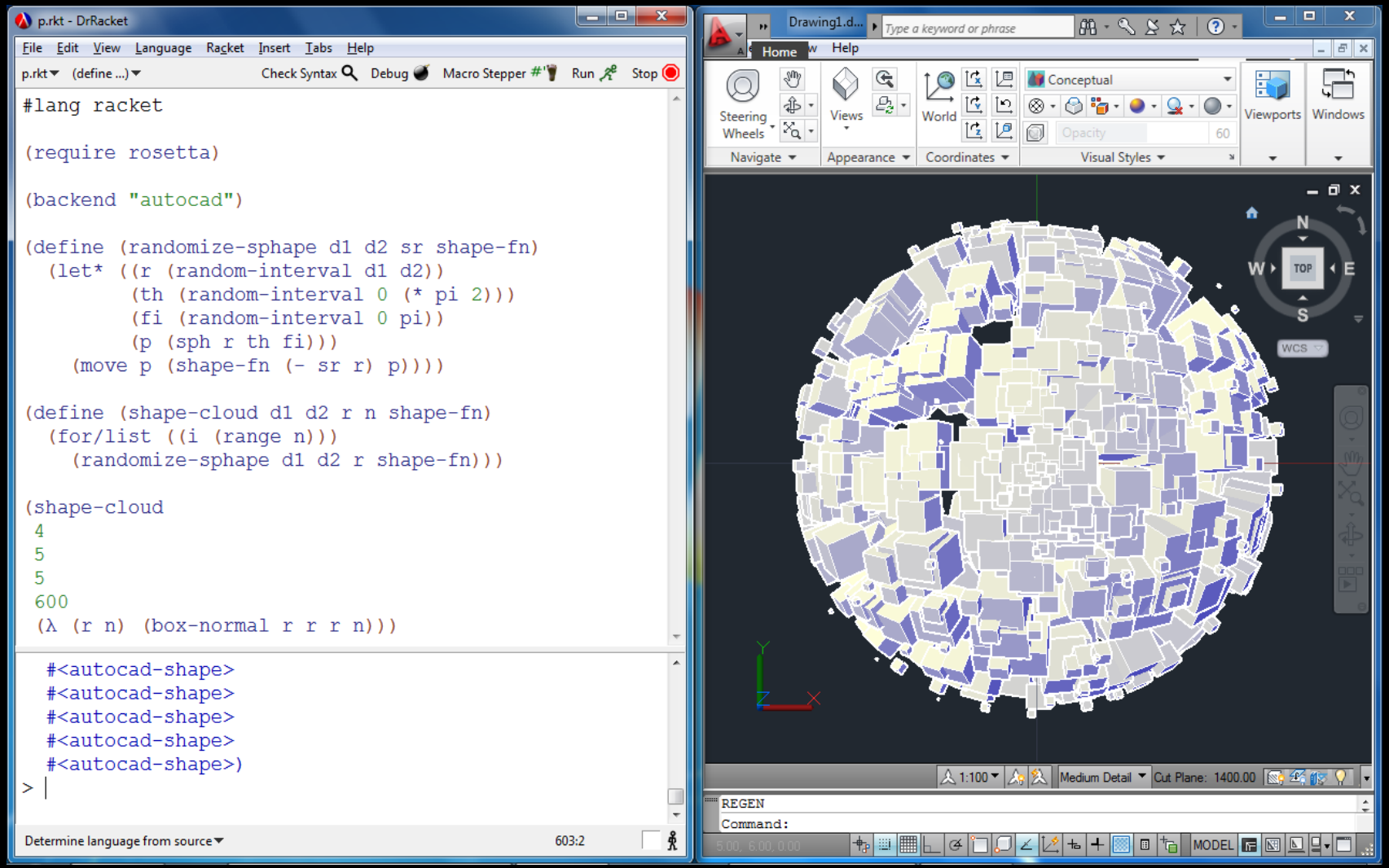
Programming Environment

- Editor
- Debugger
- Listener

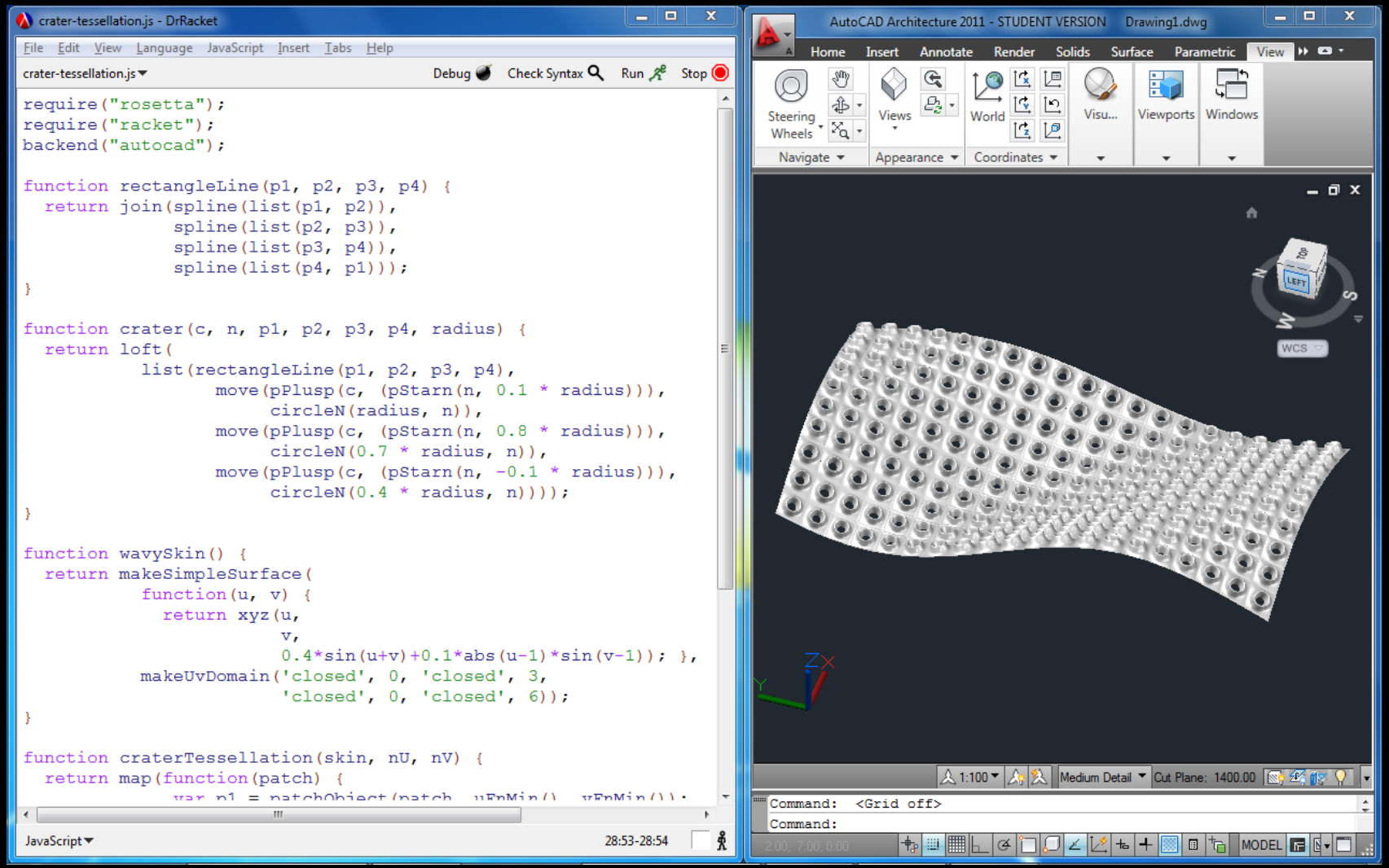
Rosetta - Multiple Back-Ends



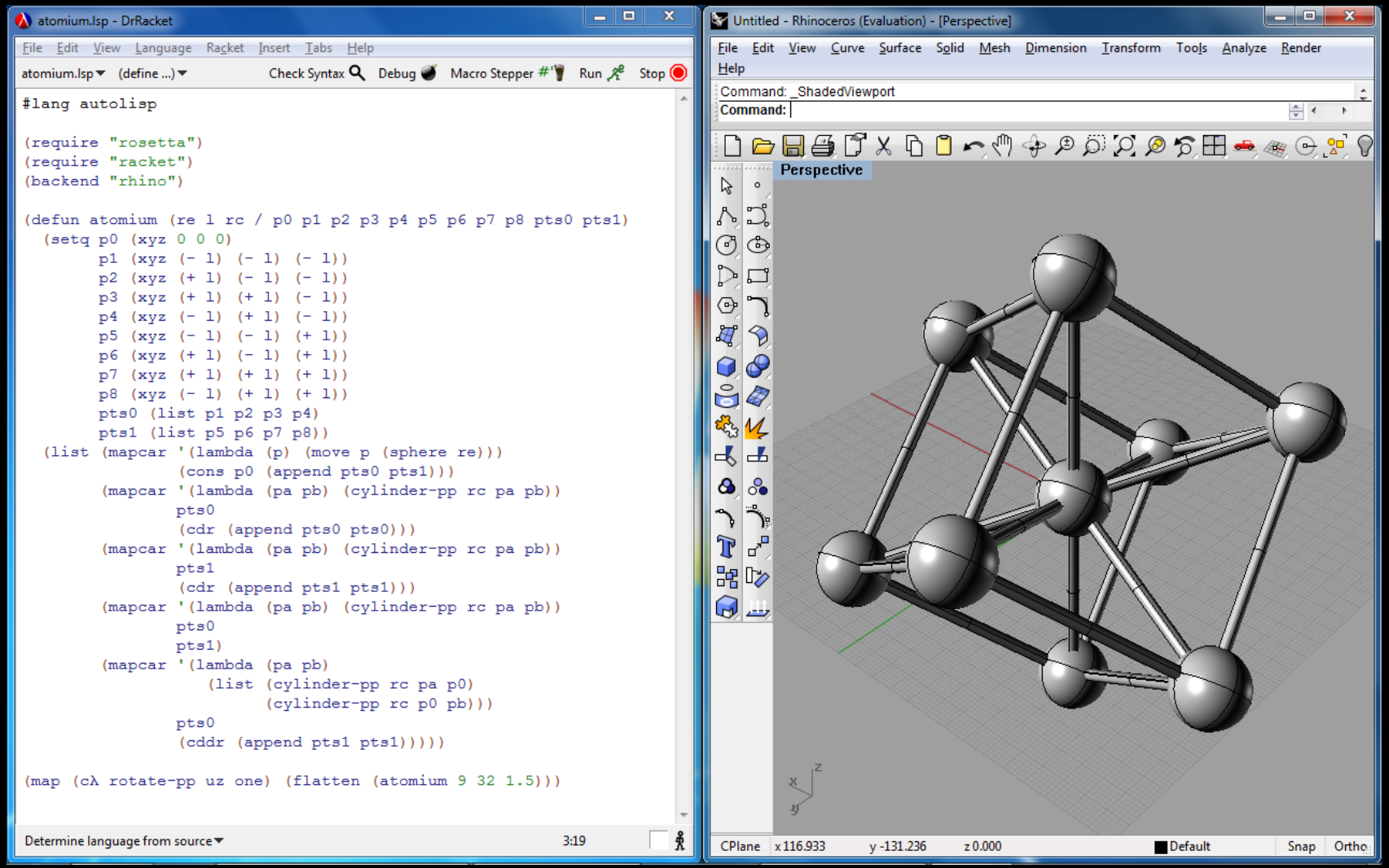
Rosetta - Multiple Back-Ends



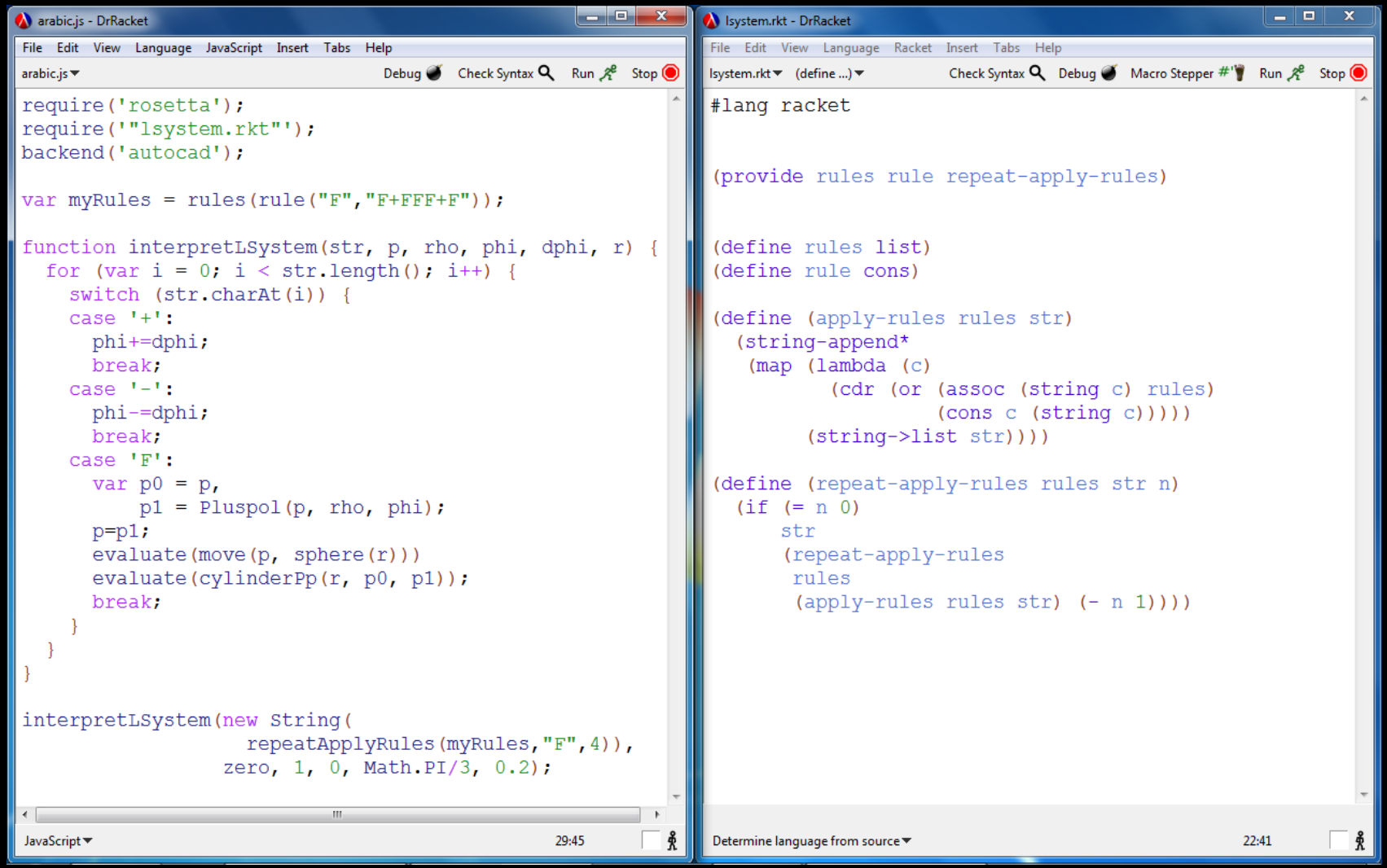
Rosetta - Multiple Front-Ends



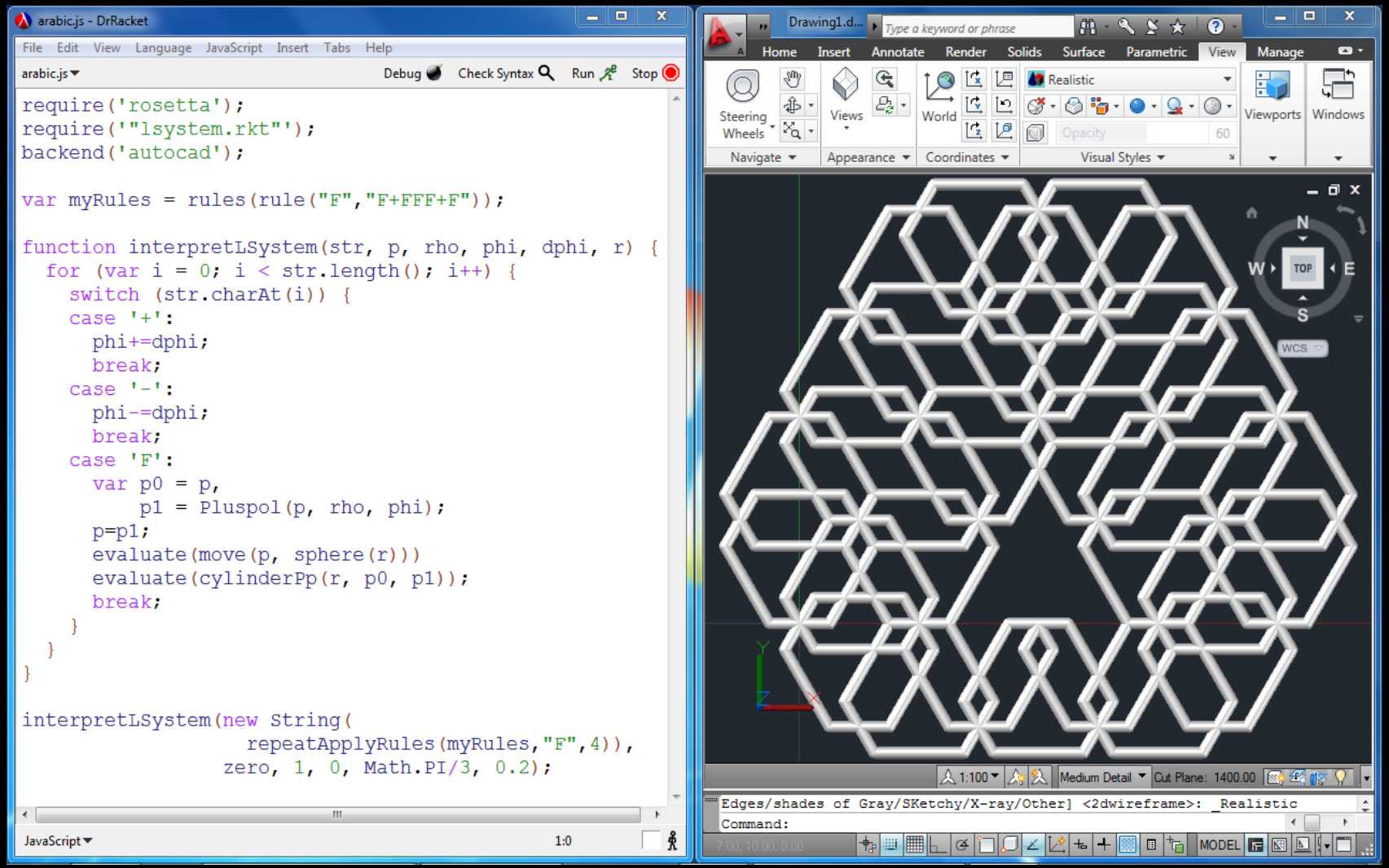
Rosetta - Multiple Front-Ends



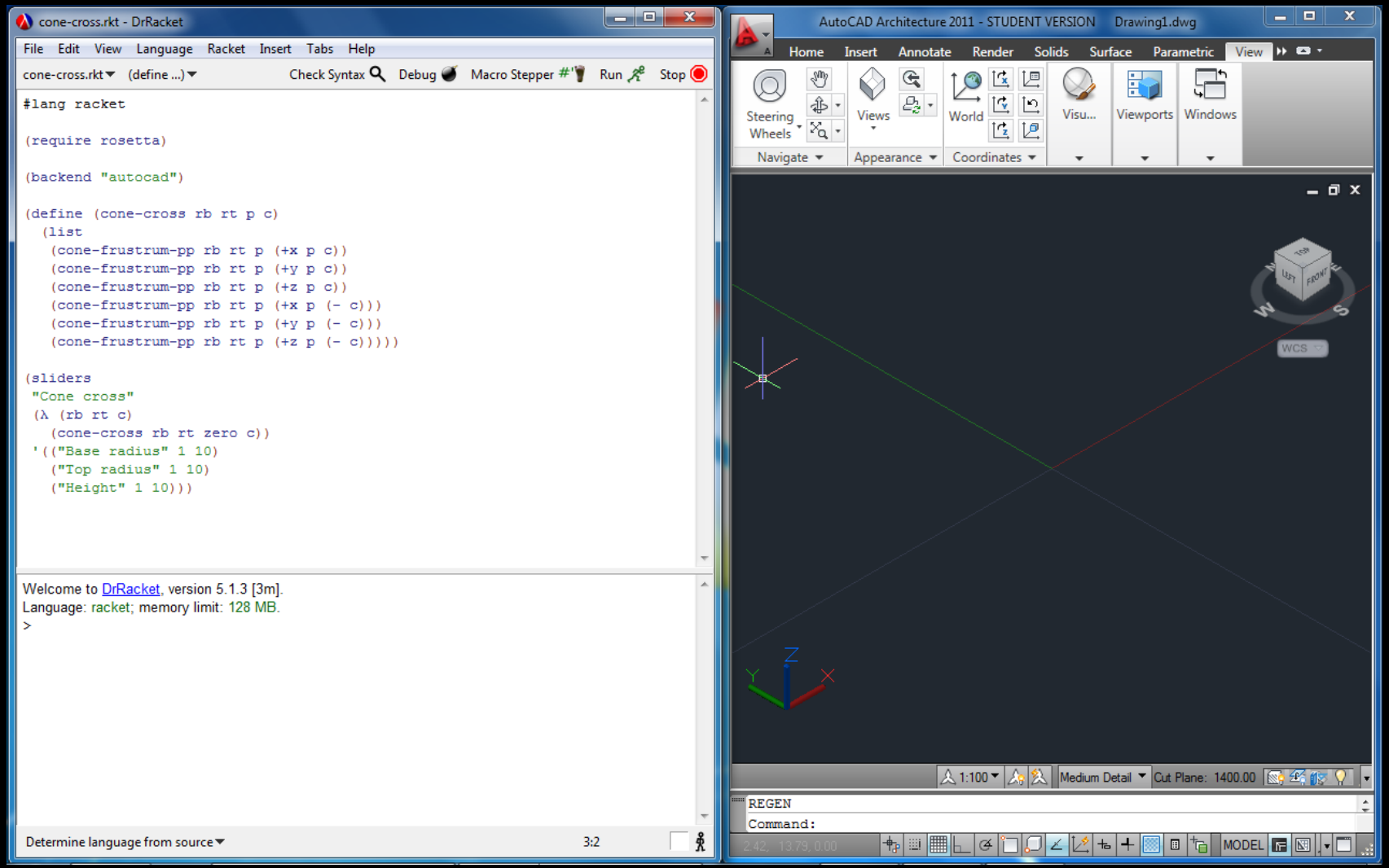
Rosetta - Portability



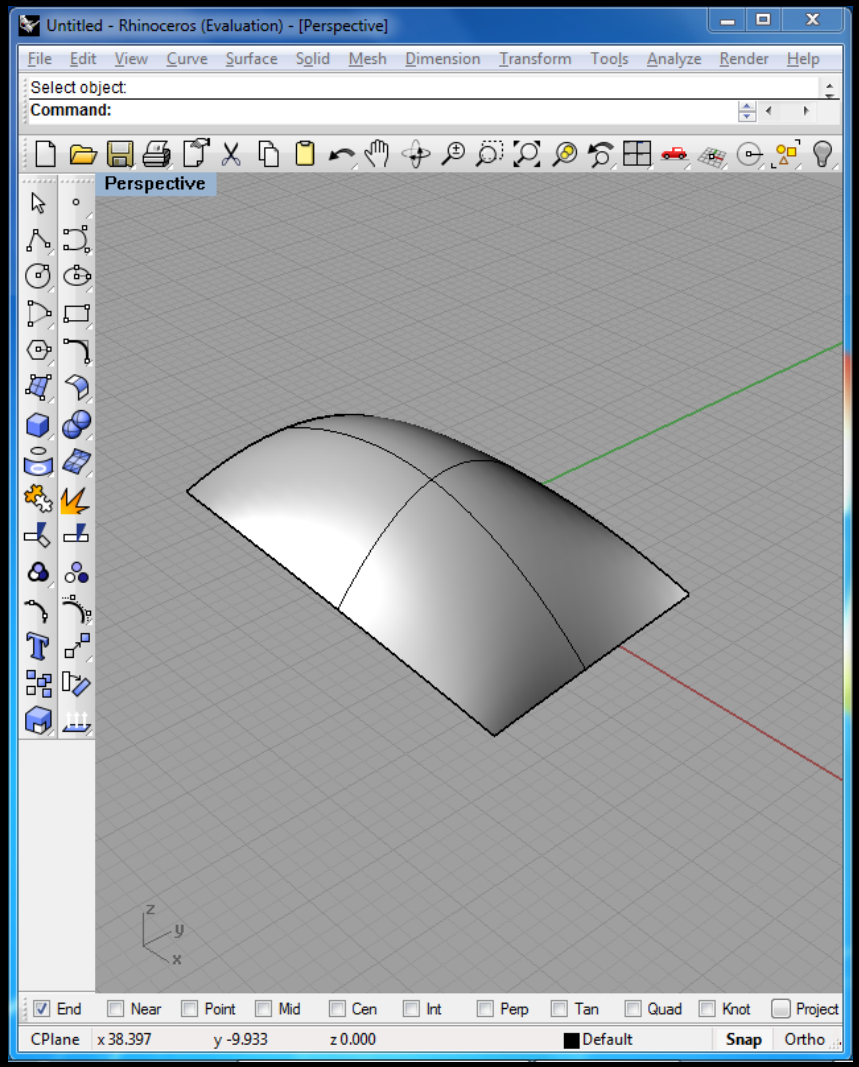
Rosetta - Portability



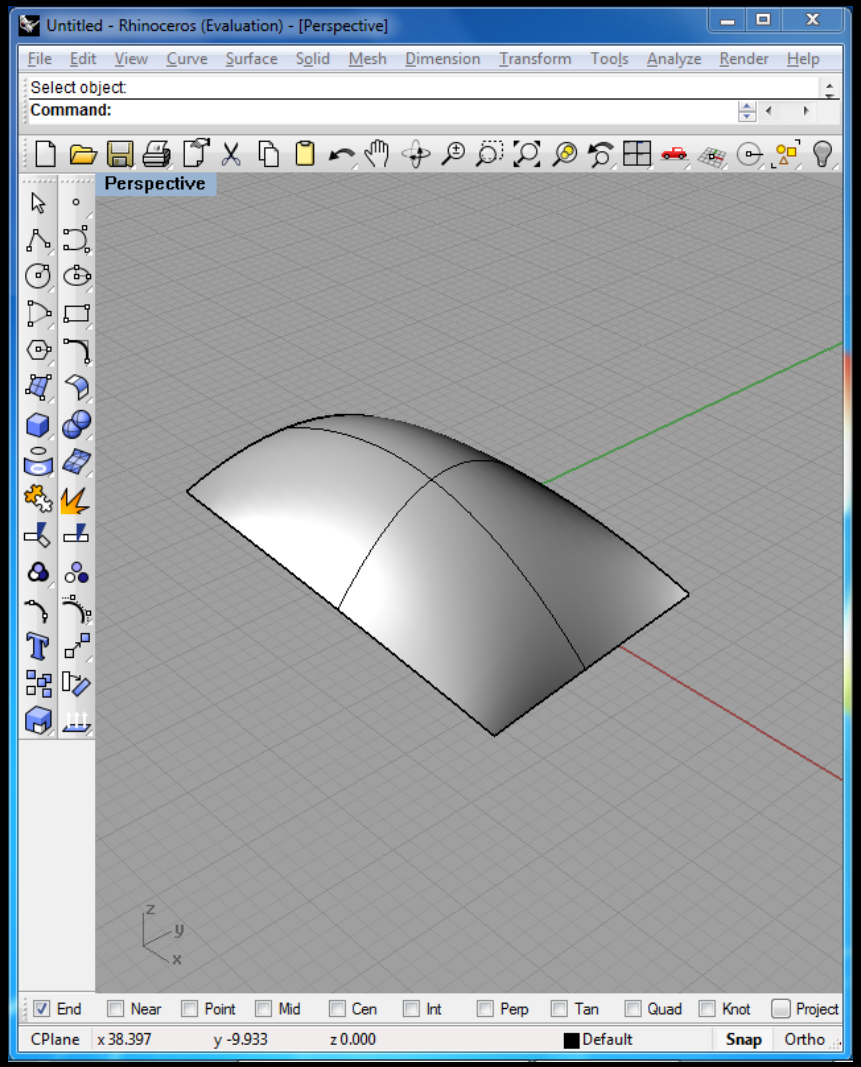
Rosetta - Interactiveness



Rosetta - Cross-CAD Applications

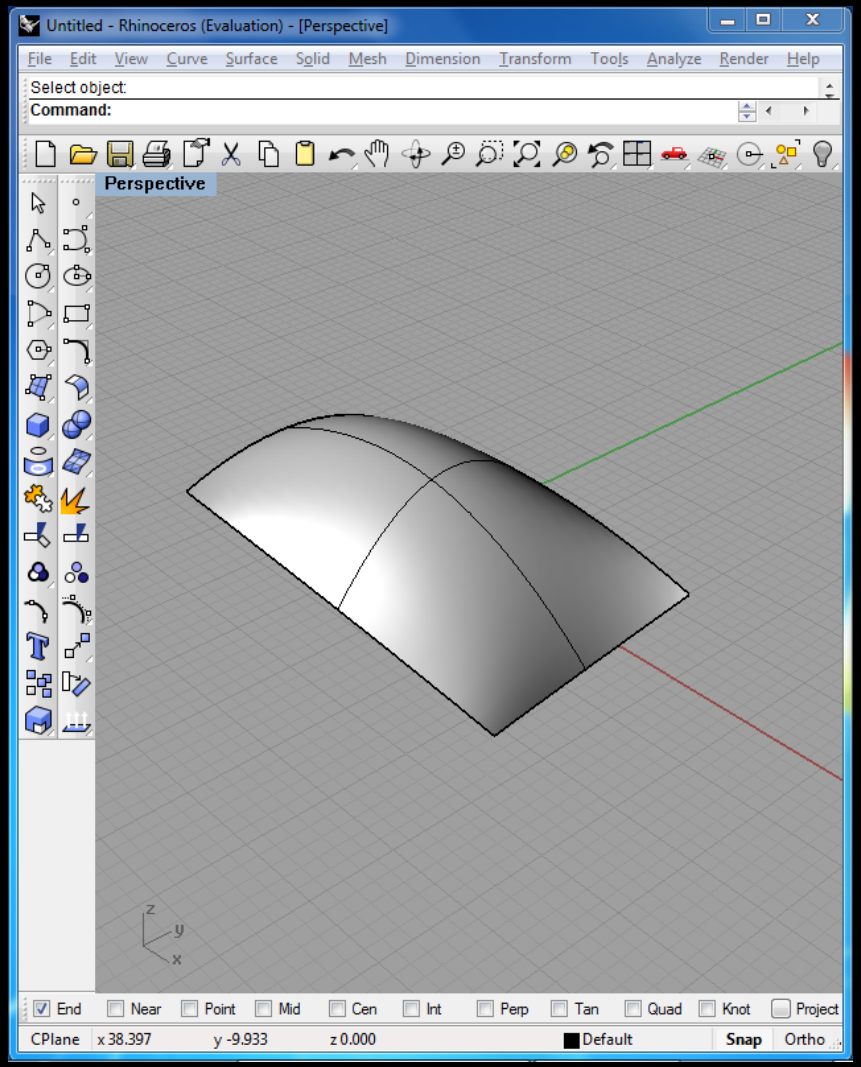


Rosetta - Cross-CAD Applications



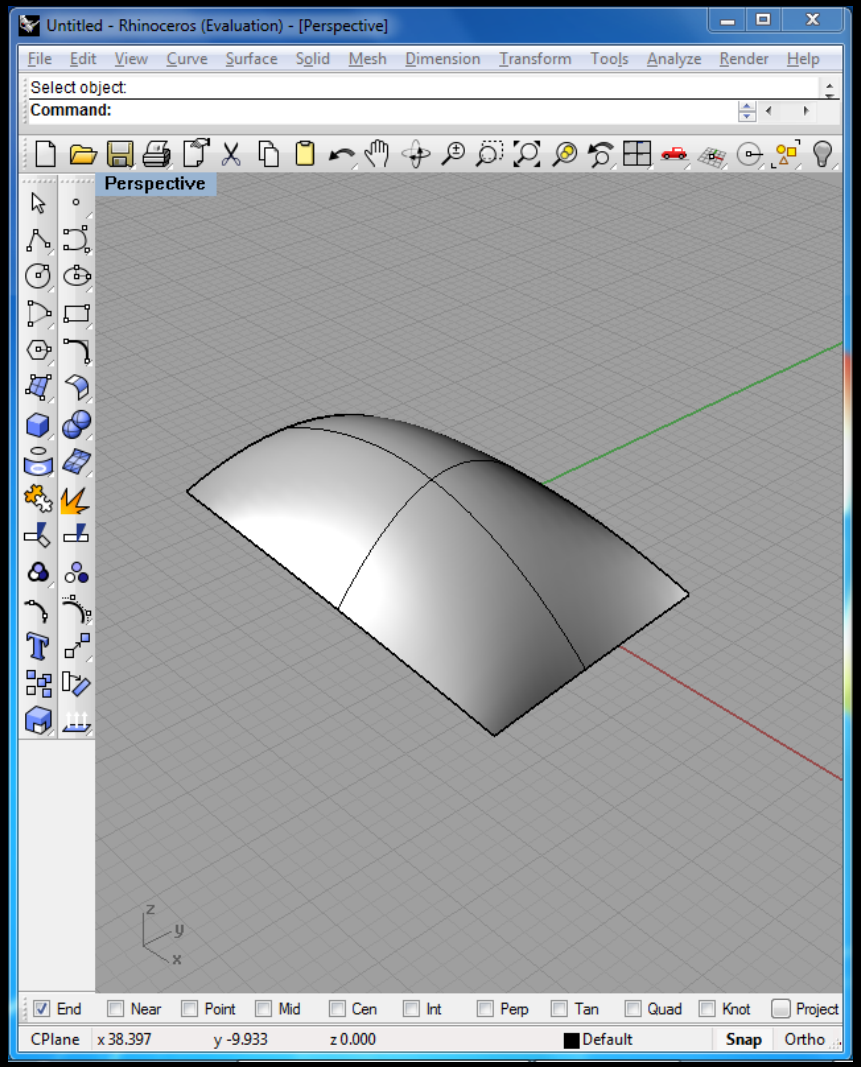

```
(backend "rhino")
```

Rosetta - Cross-CAD Applications



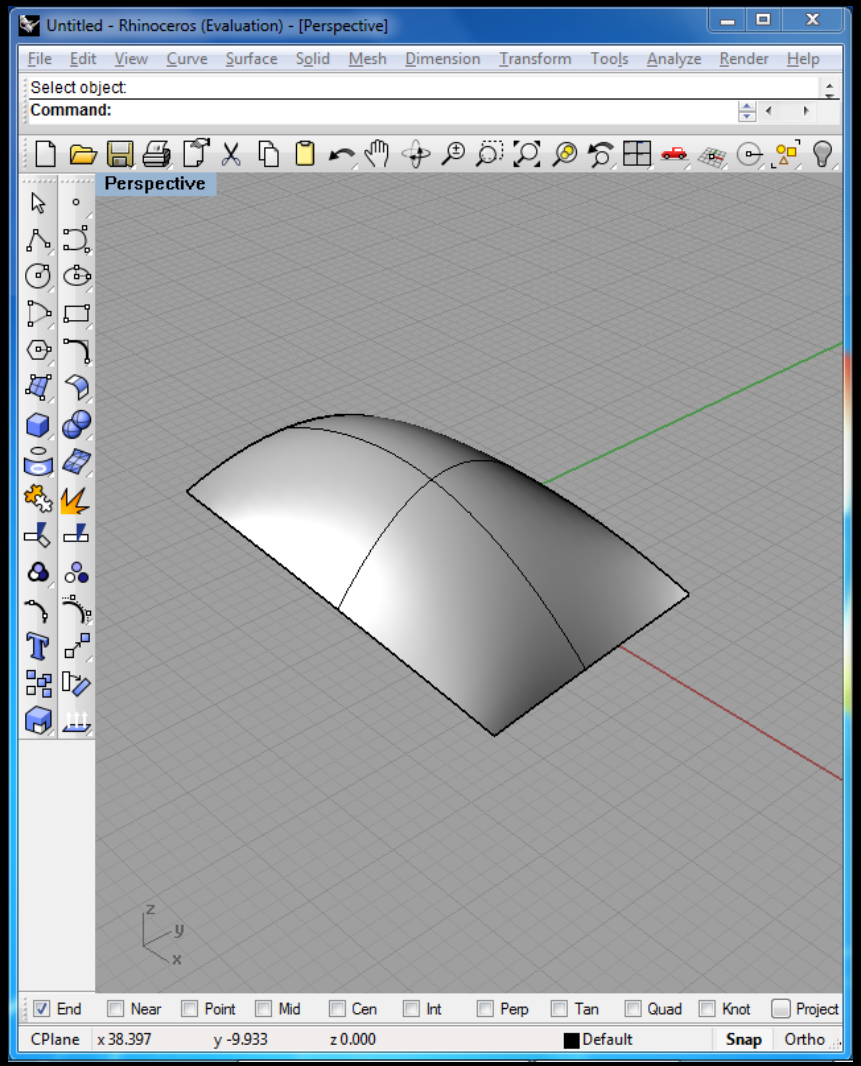

```
(backend "rhino")

(define surf
  (get-object "Select surface"))
```

Rosetta - Cross-CAD Applications



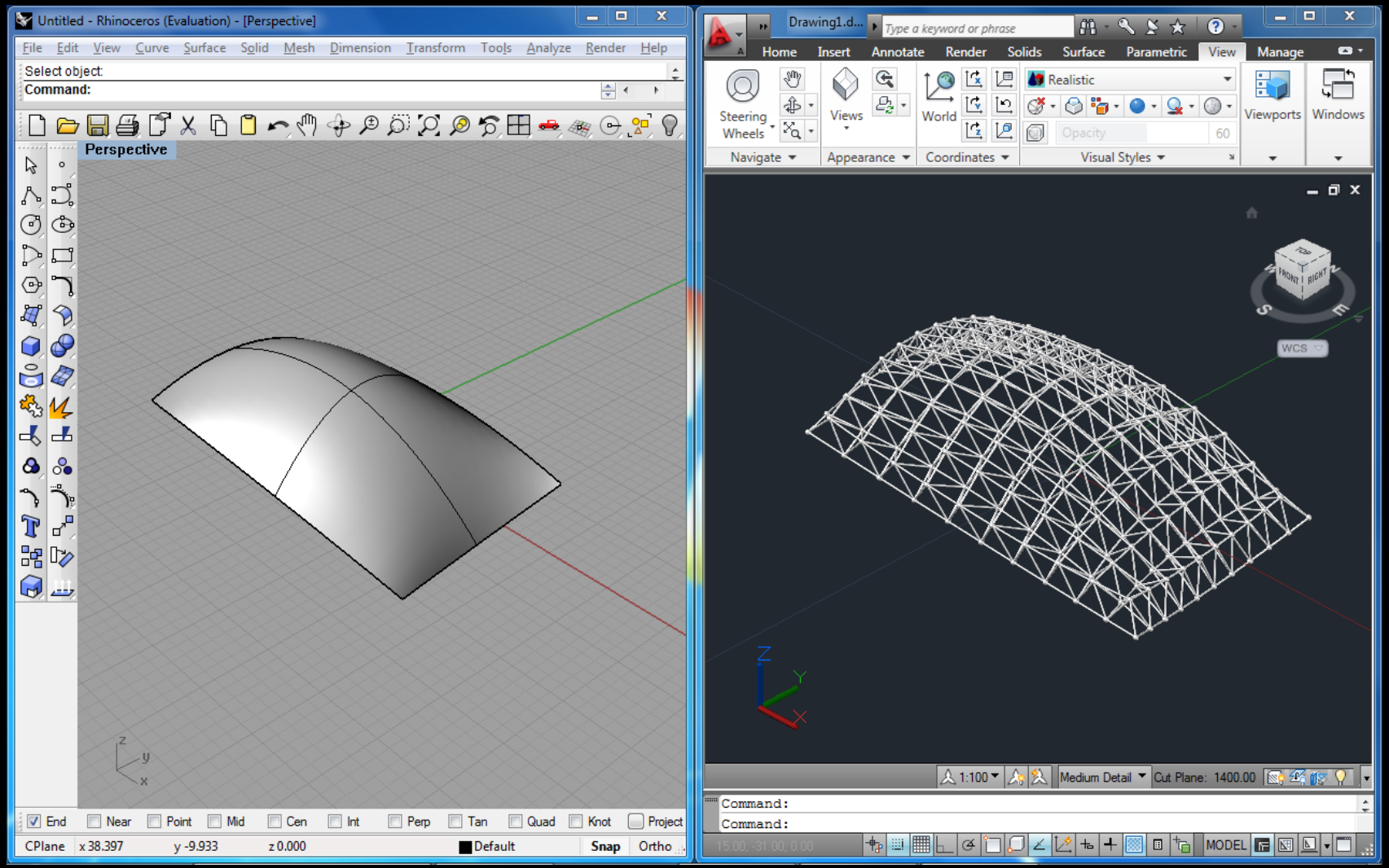

```
(backend "rhino")

(define surf
  (get-object "Select surface"))

(backend "autocad")
```

Rosetta - Cross-CAD Applications

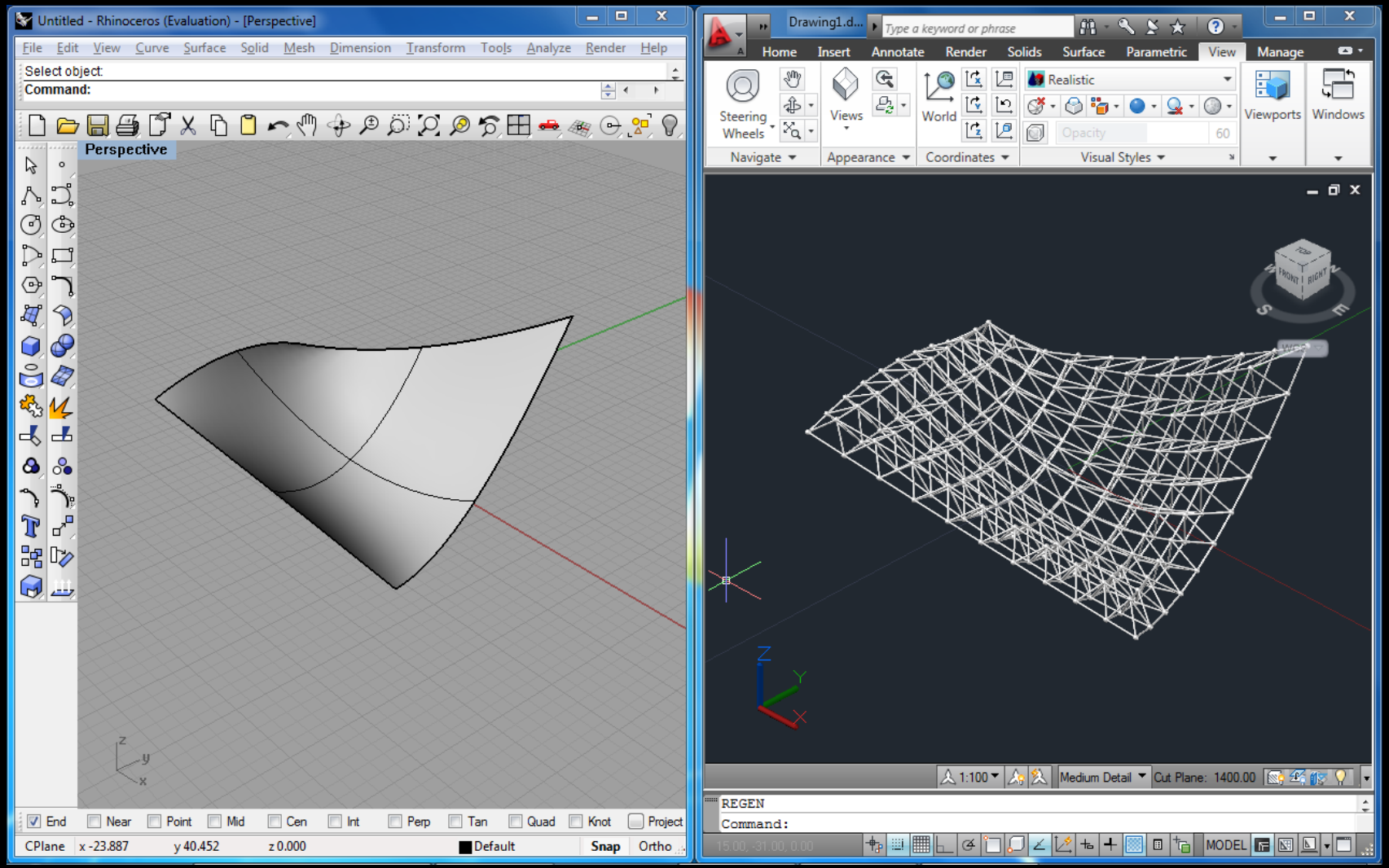



```
(backend "rhino")

(define surf
  (get-object "Select surface"))

(backend "autocad")

(truss-from-surface surf)
```

Rosetta - Cross-CAD Applications



Rosetta - Cross-CAD Applications



Conclusion

The architect must be a prophet… a prophet in the true sense of the term… if he can't see at least ten years ahead don't call him an architect

Frank Lloyd Wright

Conclusion

> *The architect must be a prophet… a prophet in the true sense of the term… if he can't see at least ten years ahead don't call him an architect*
>
> *Frank Lloyd Wright*

Programming will be a fundamental tool for Architecture

Conclusion

The architect must be a prophet… a prophet in the true sense of the term… if he can't see at least ten years ahead don't call him an architect

Frank Lloyd Wright

Programming will be a fundamental tool for Architecture

Thank You

Conclusion

The architect must be a prophet… a prophet in the true sense of the term… if he can't see at least ten years ahead don't call him an architect

Frank Lloyd Wright

Programming will be a fundamental tool for Architecture

Thank You

Questions?