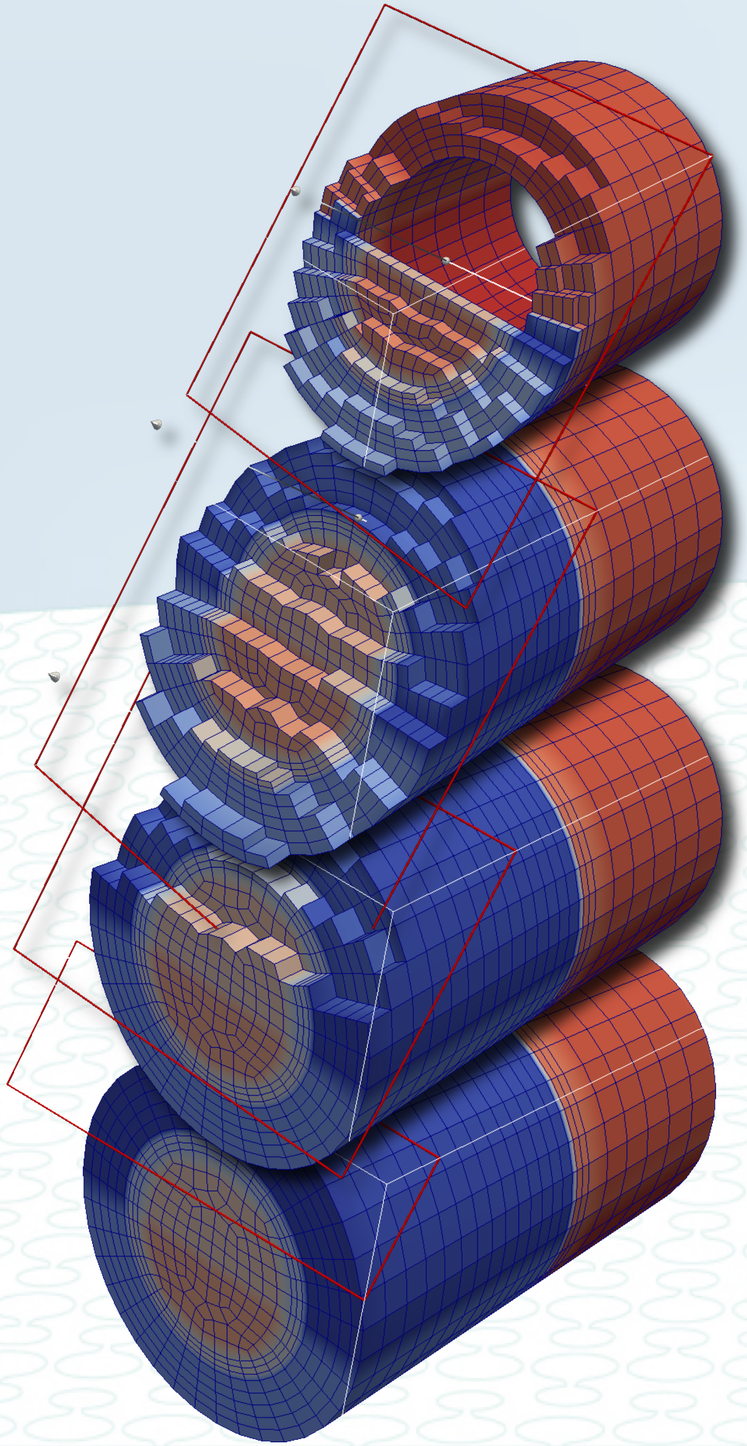


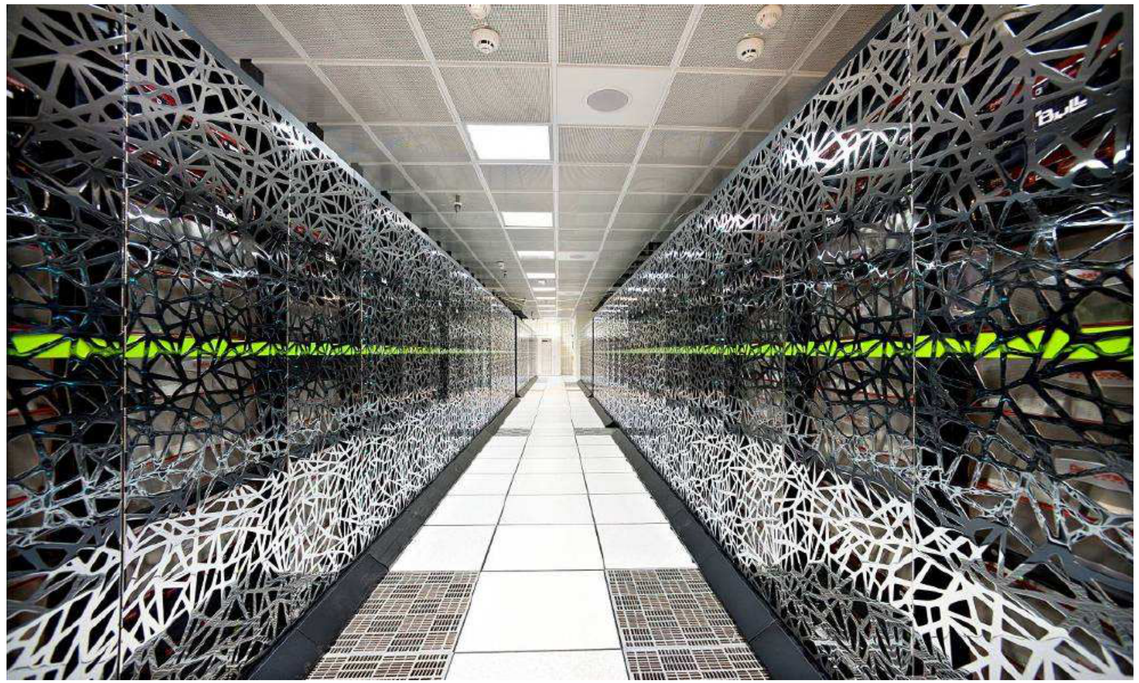
Domain Representations

M2S

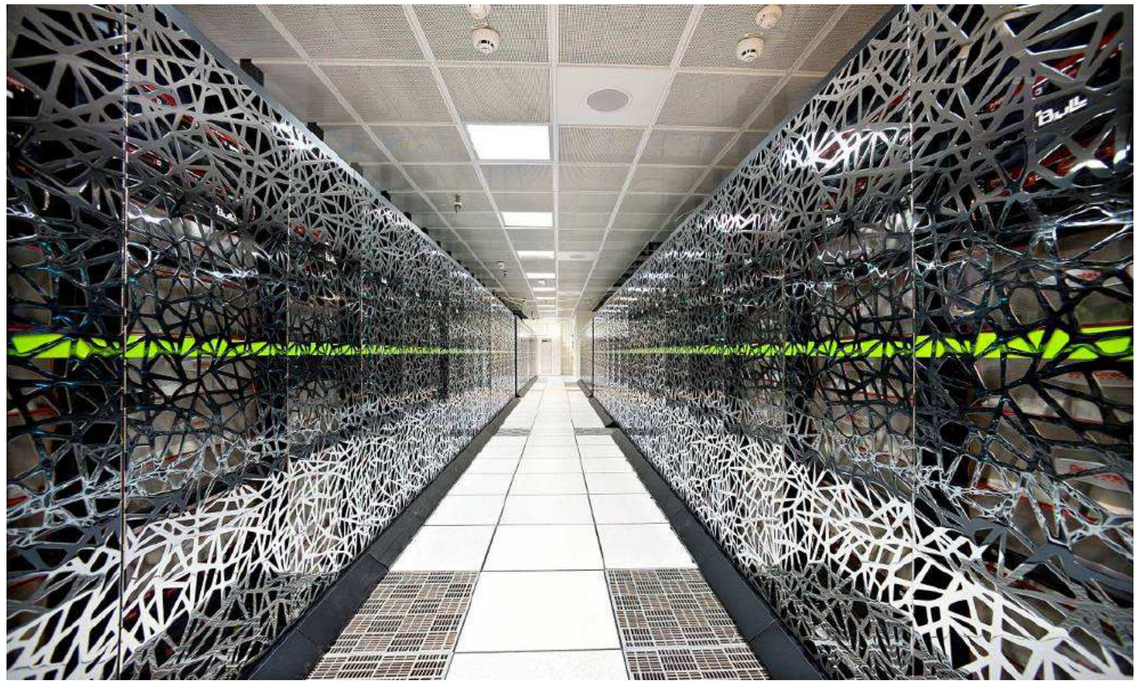
J. Tierny

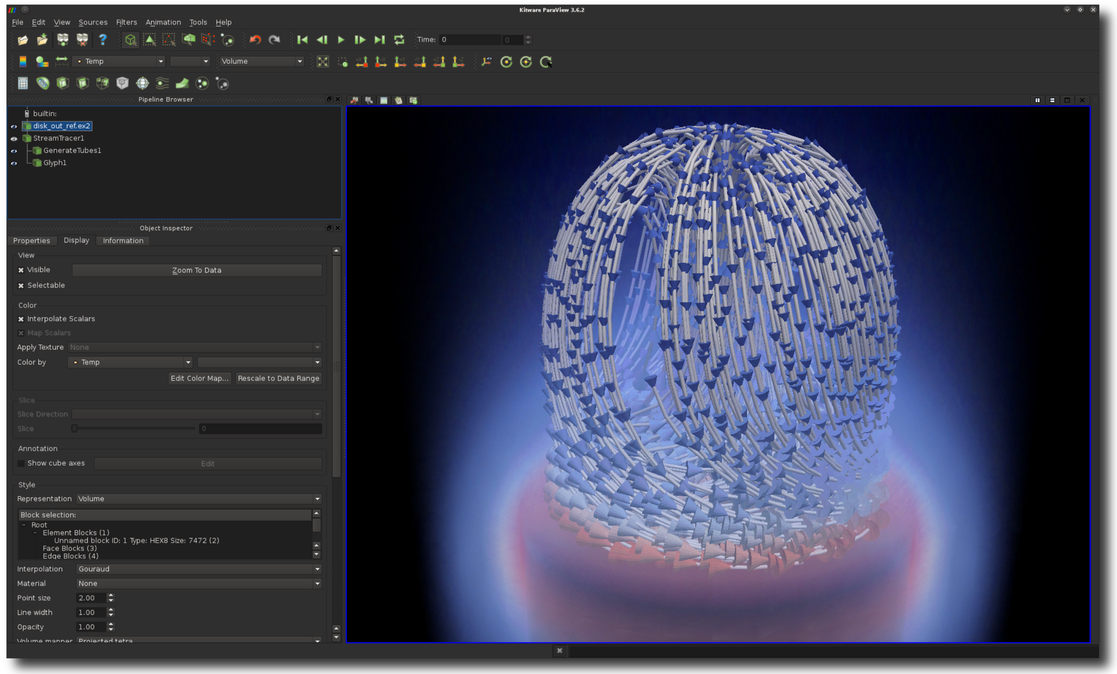
What?

What?

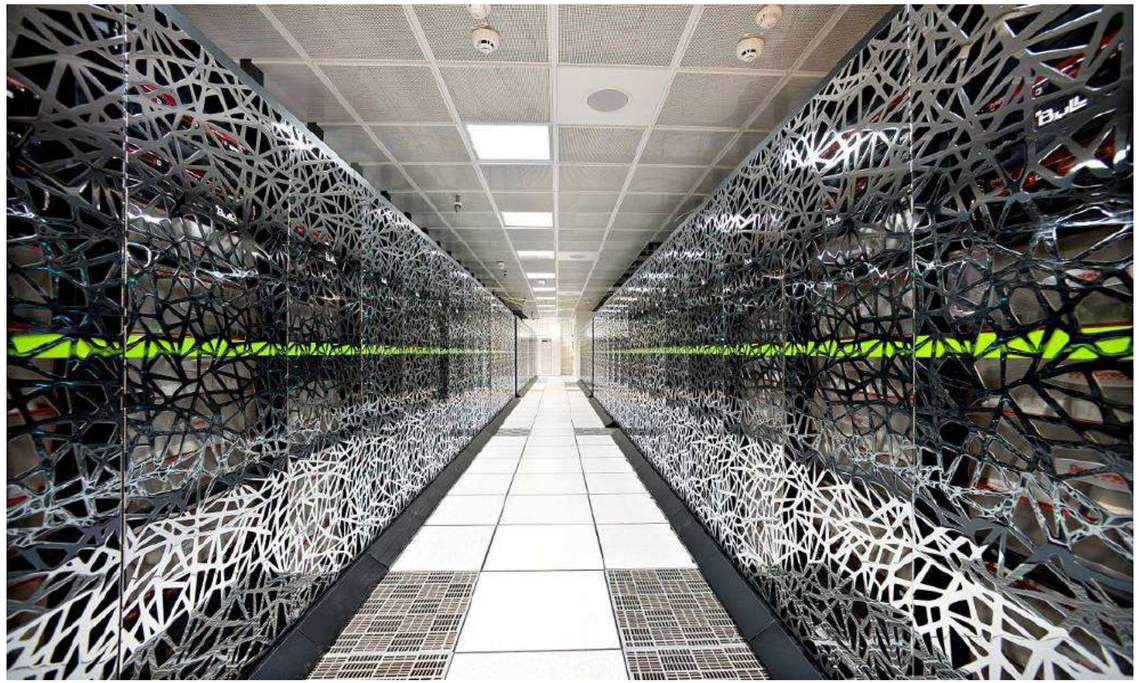


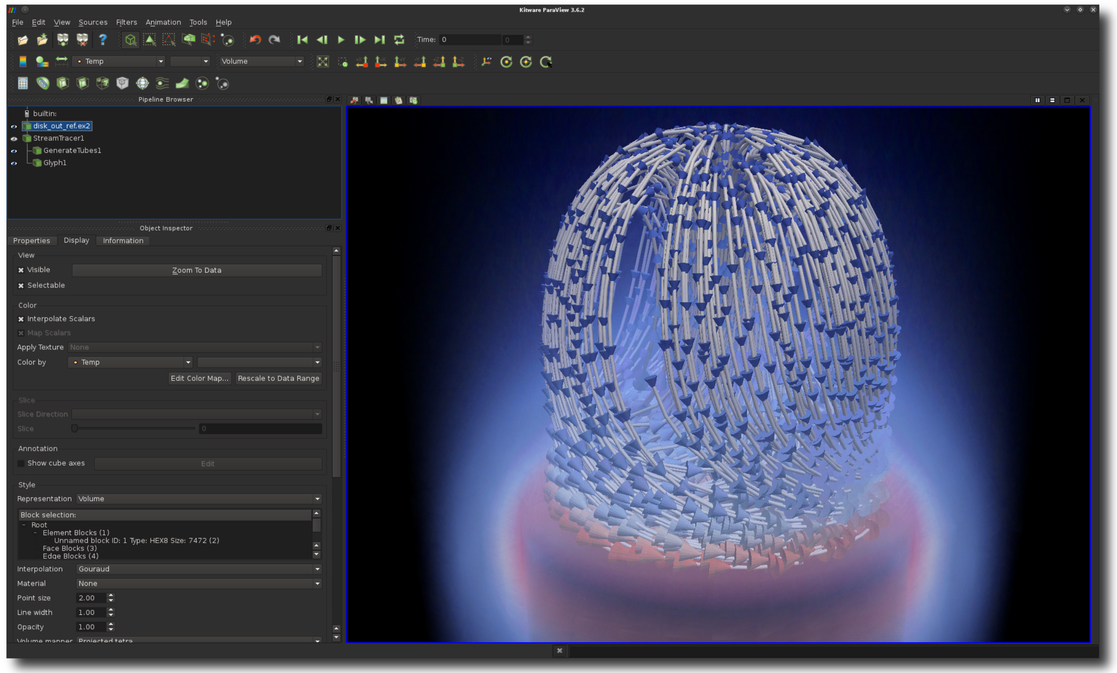
What?




File Edit View Sources Filters Animation Tools Help
Pipeline Browser
builtin:
disk_out_ref.ex2
StreamTracer1
GenerateTubes1
Glyph1
Object Inspector
Properties Display Information
View
Visible
Zoom To Data
Selectable
Color
Interpolate Scalars
Map Scalars
Apply Texture None
Color by Temp
Edit Color Map...
Rescale to Data Range
Slice
Slice Direction
Slice
Annotation
Show cube axes
Edit
Style
Representation Volume
Block selection:
Root
Element Blocks (1)
Unnamed block ID: 1 Type: HEX8 Size: 7472 (2)
Face Blocks (3)
Edge Blocks (4)
Interpolation Gouraud
Material None
Point size 2.00
Line width 1.00
Opacity 1.00

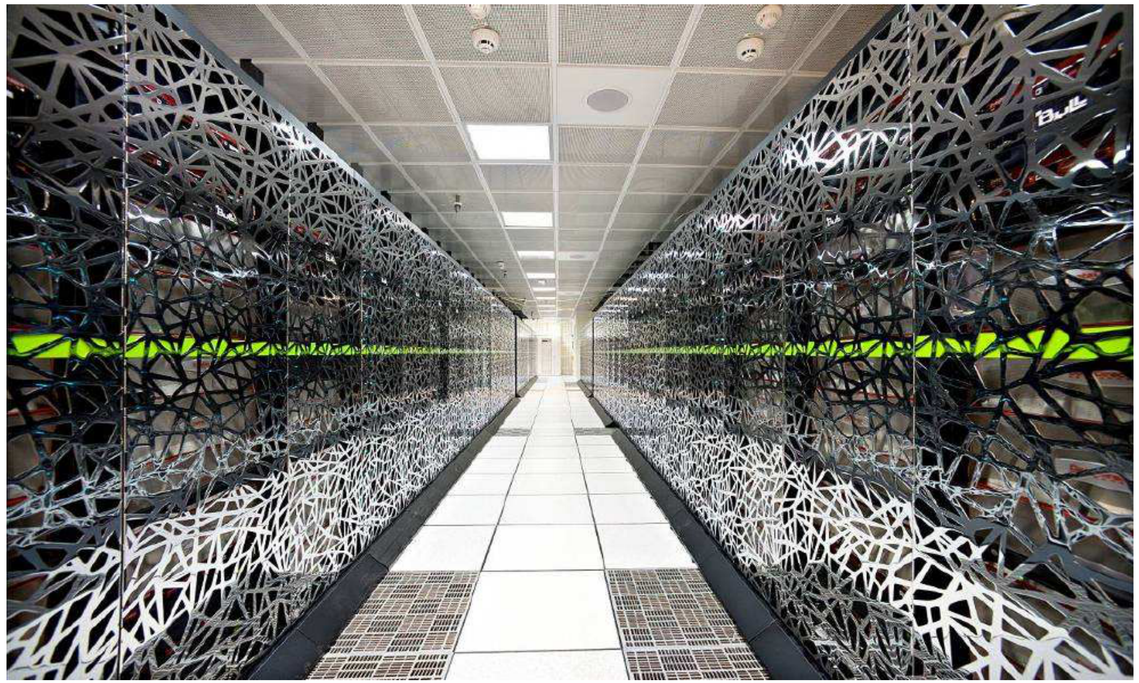
What?

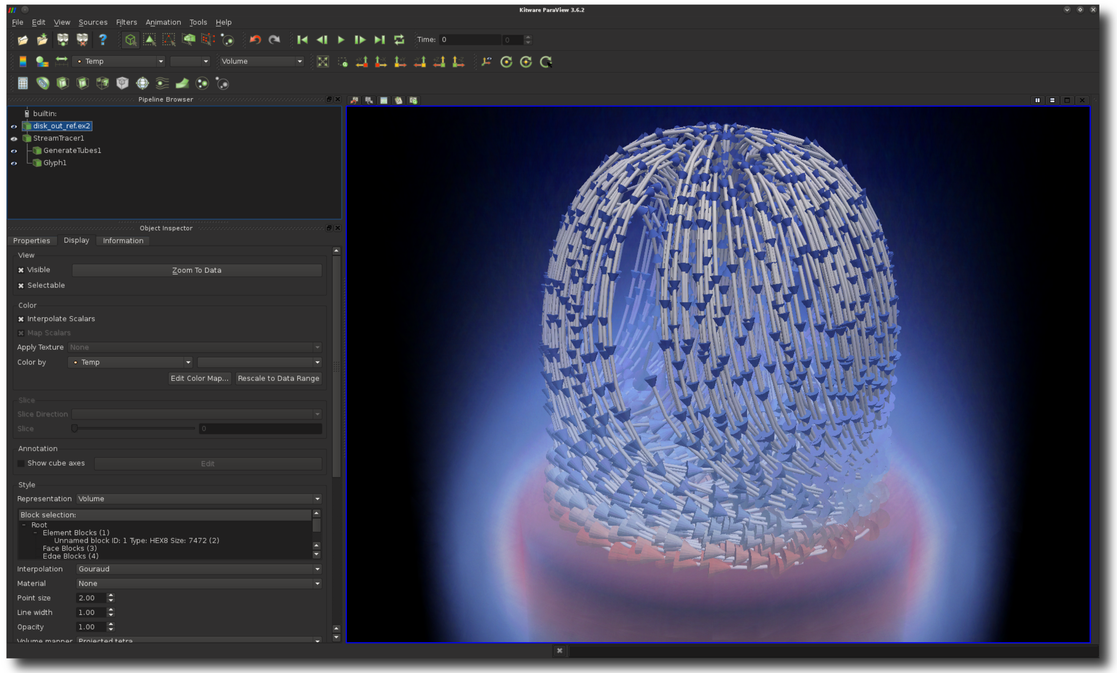




- What have we computed?

What?

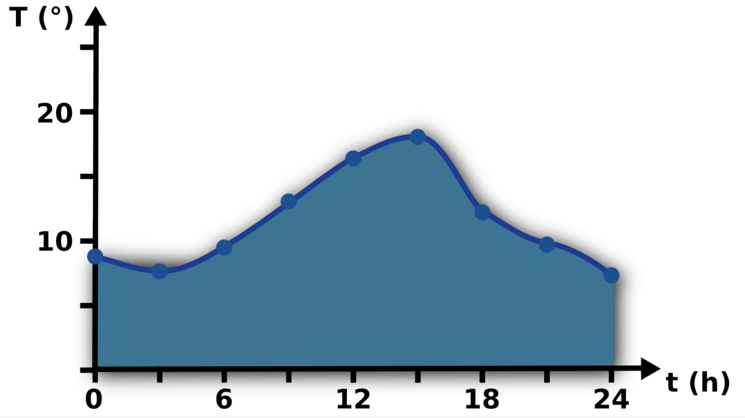




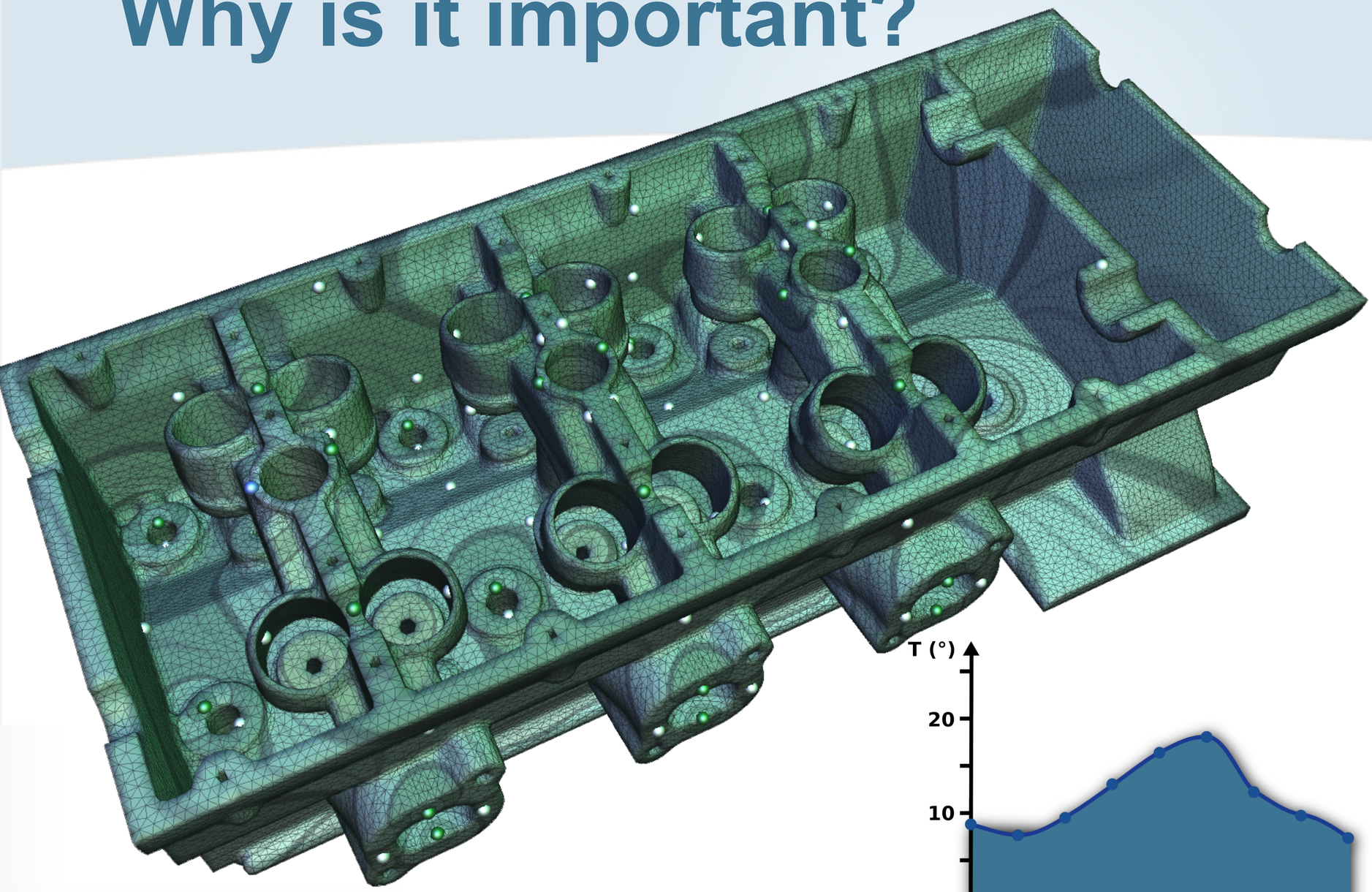
- What have we computed?
- On what kind of *discretized-space*?

Why is it important?

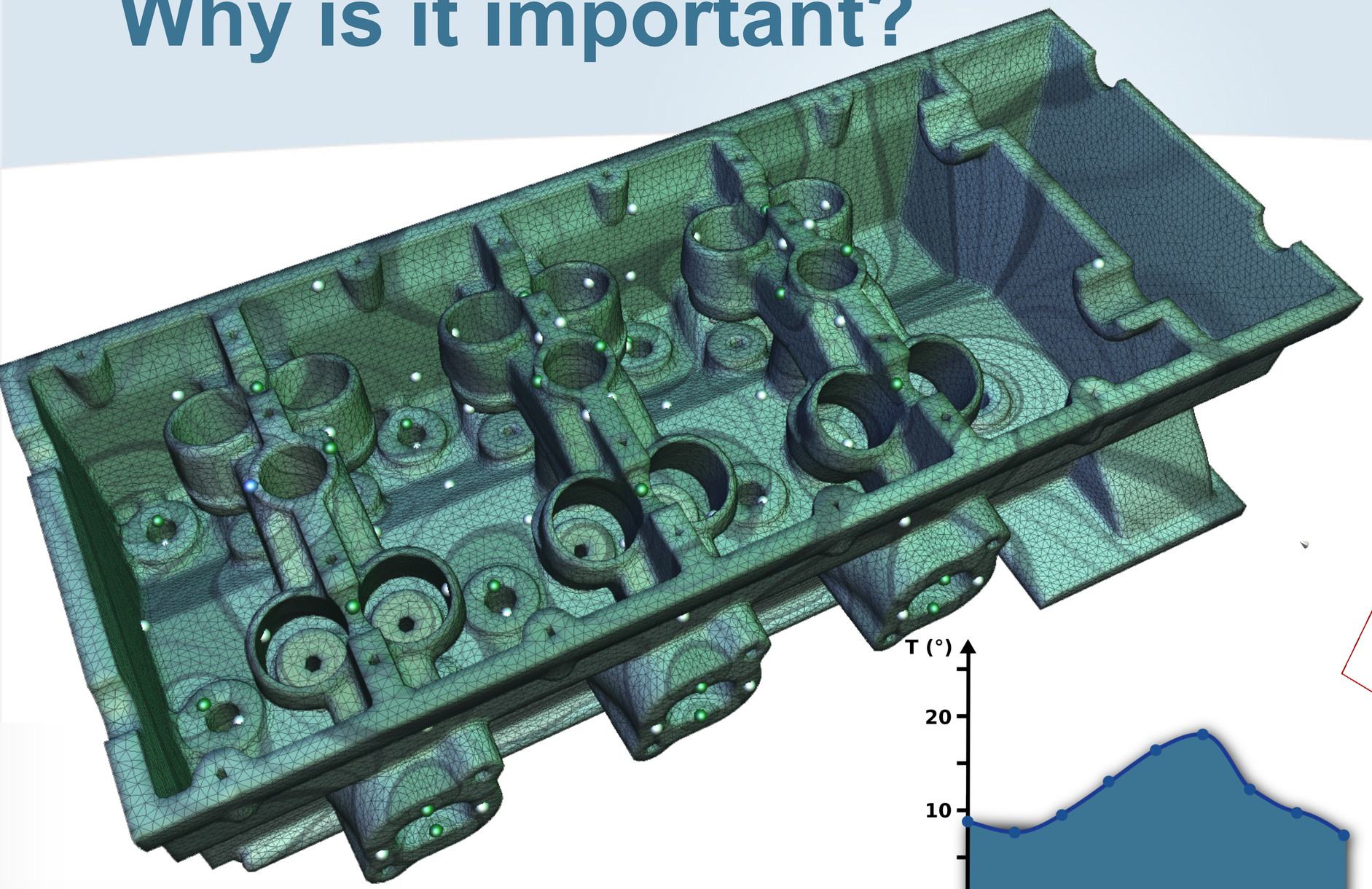
Why is it important?

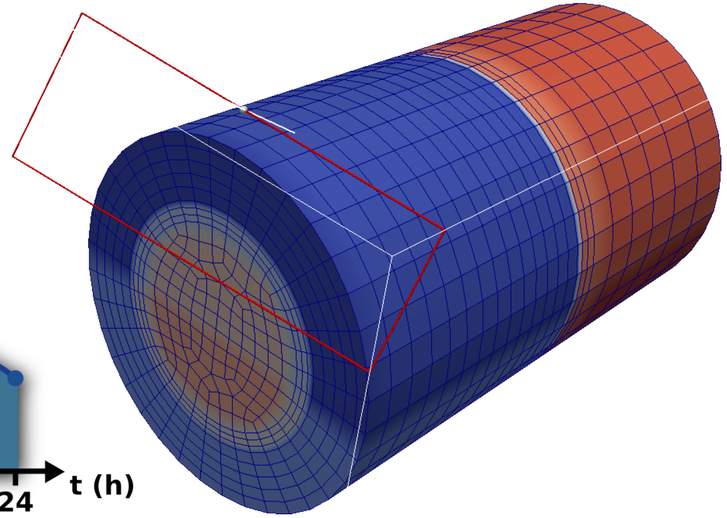


Why is it important?

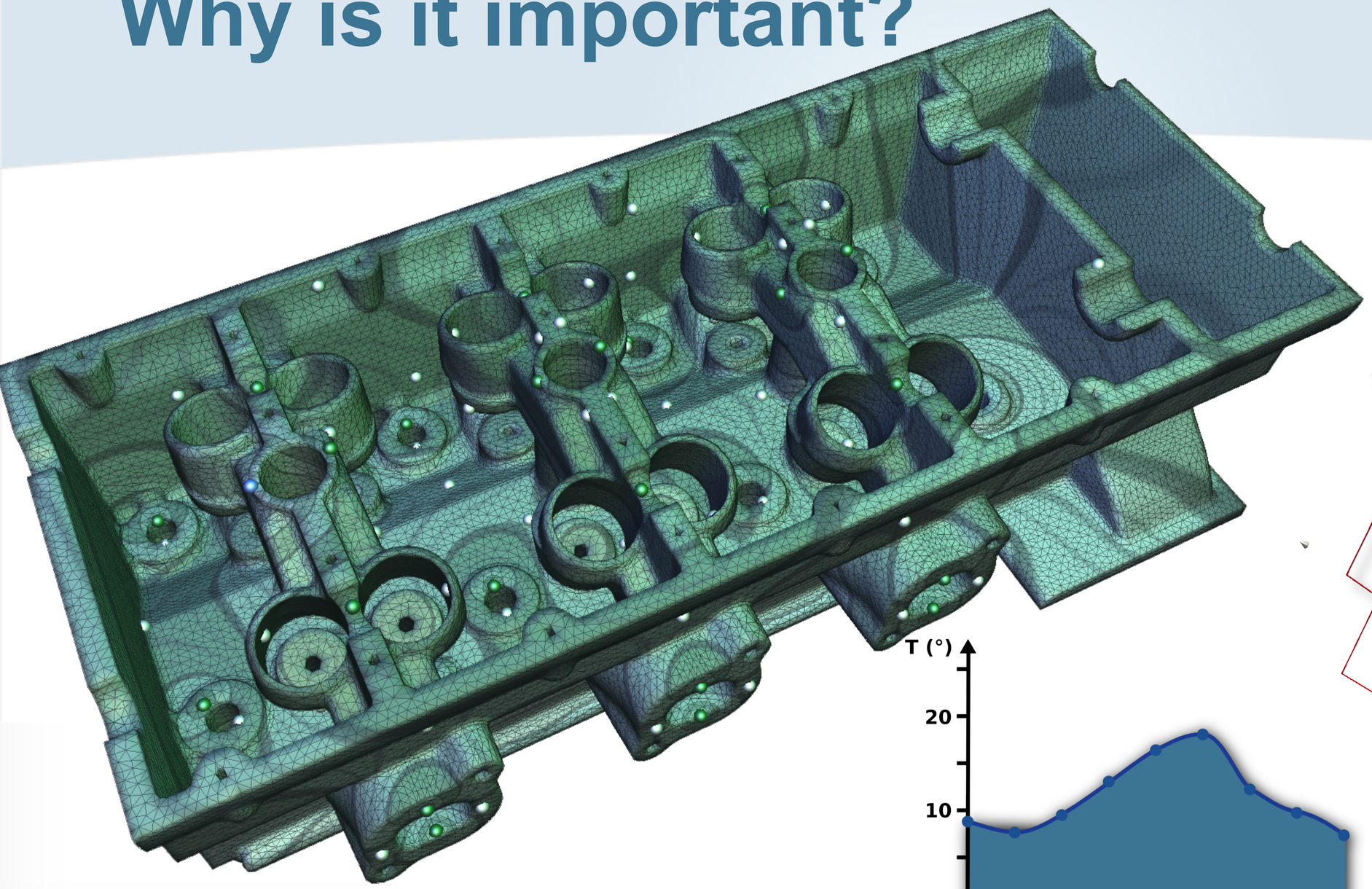


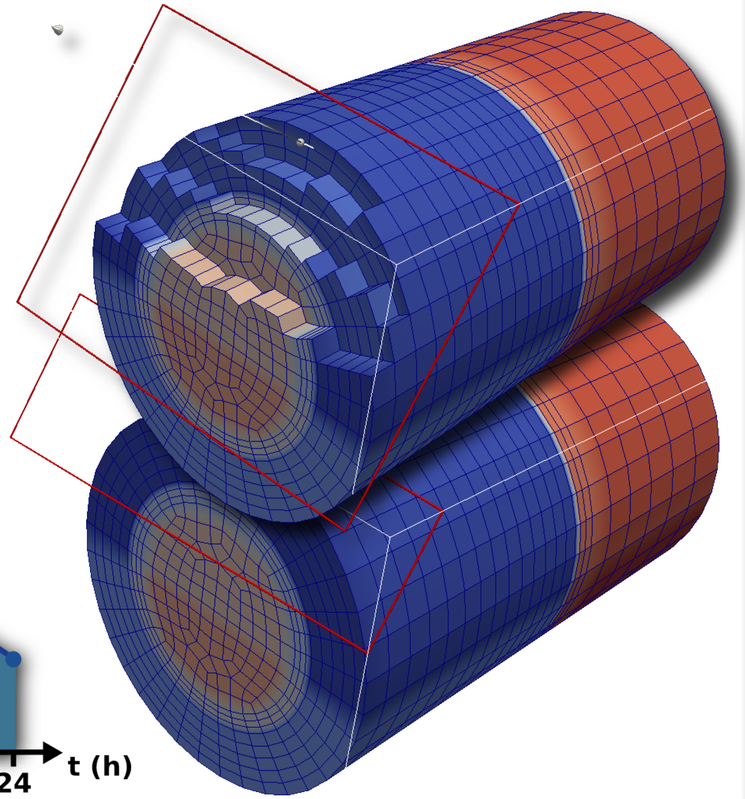
Why is it important?



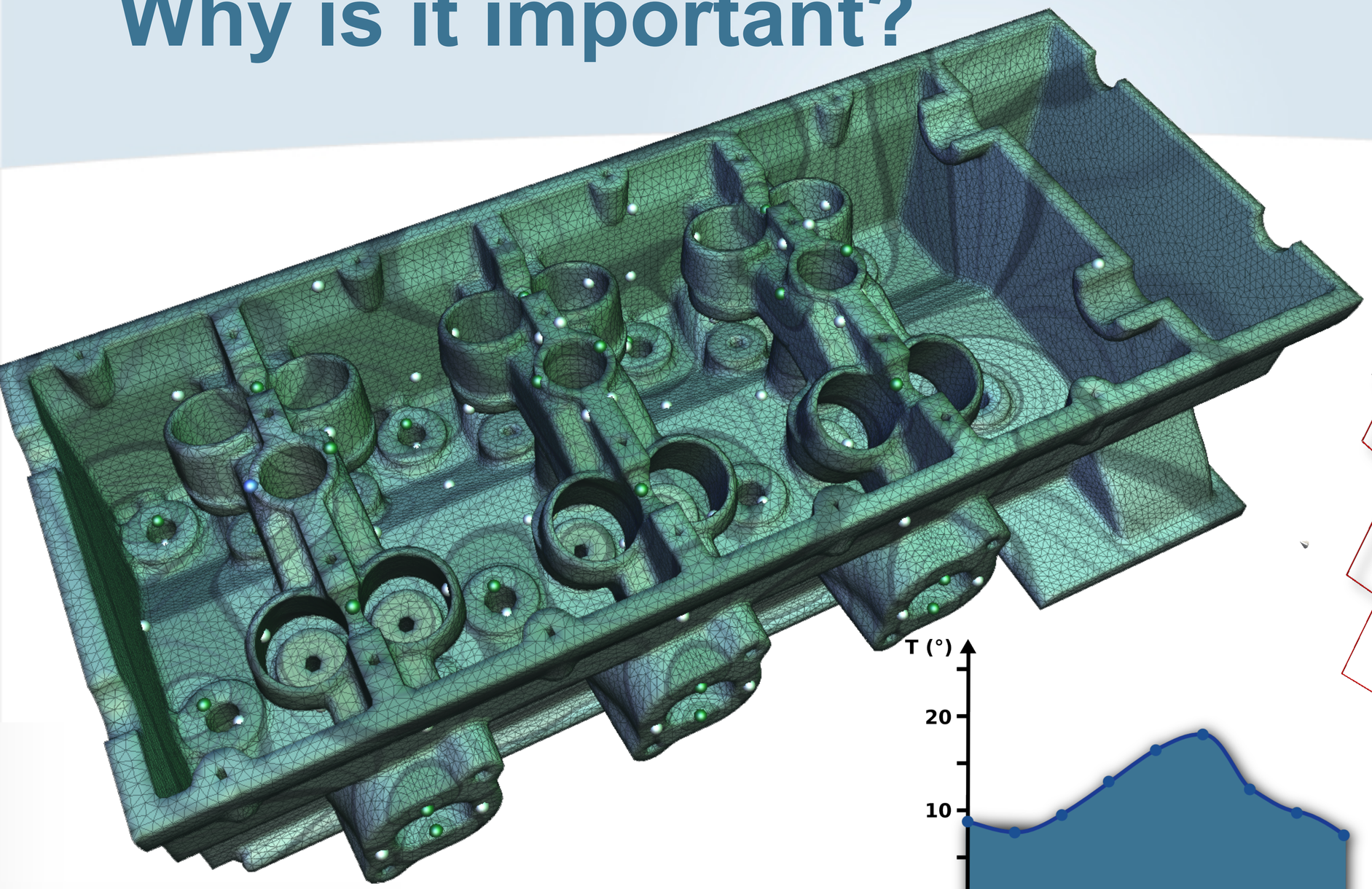


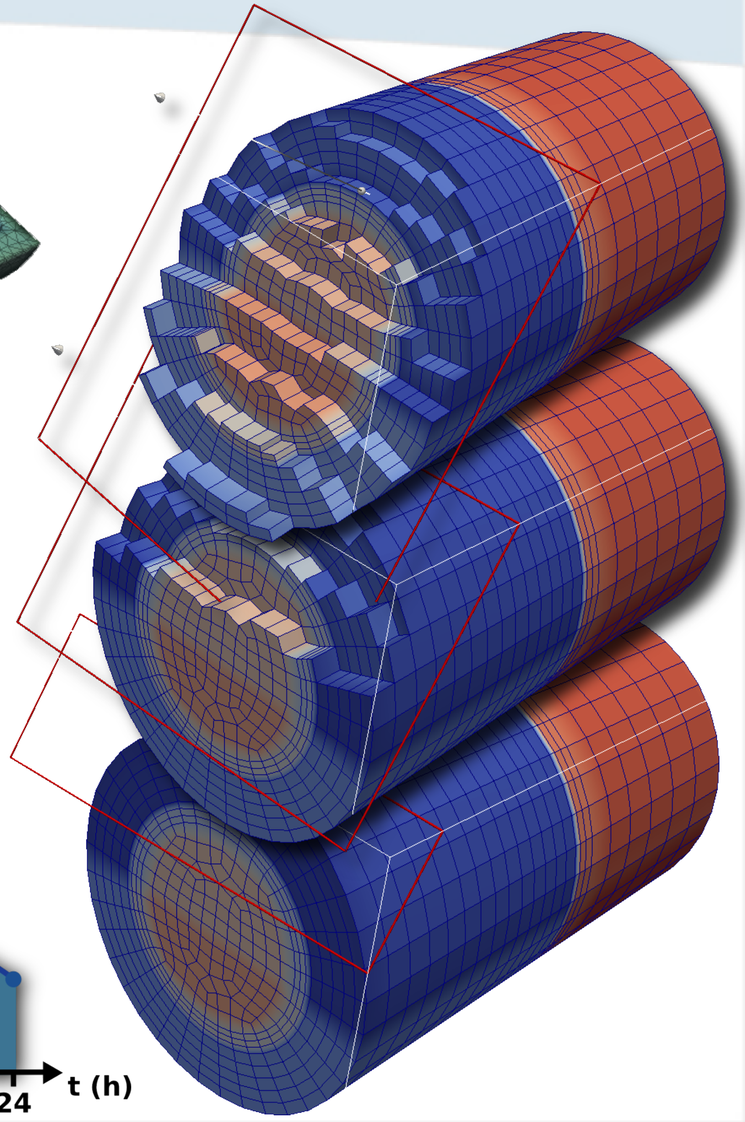
Why is it important?



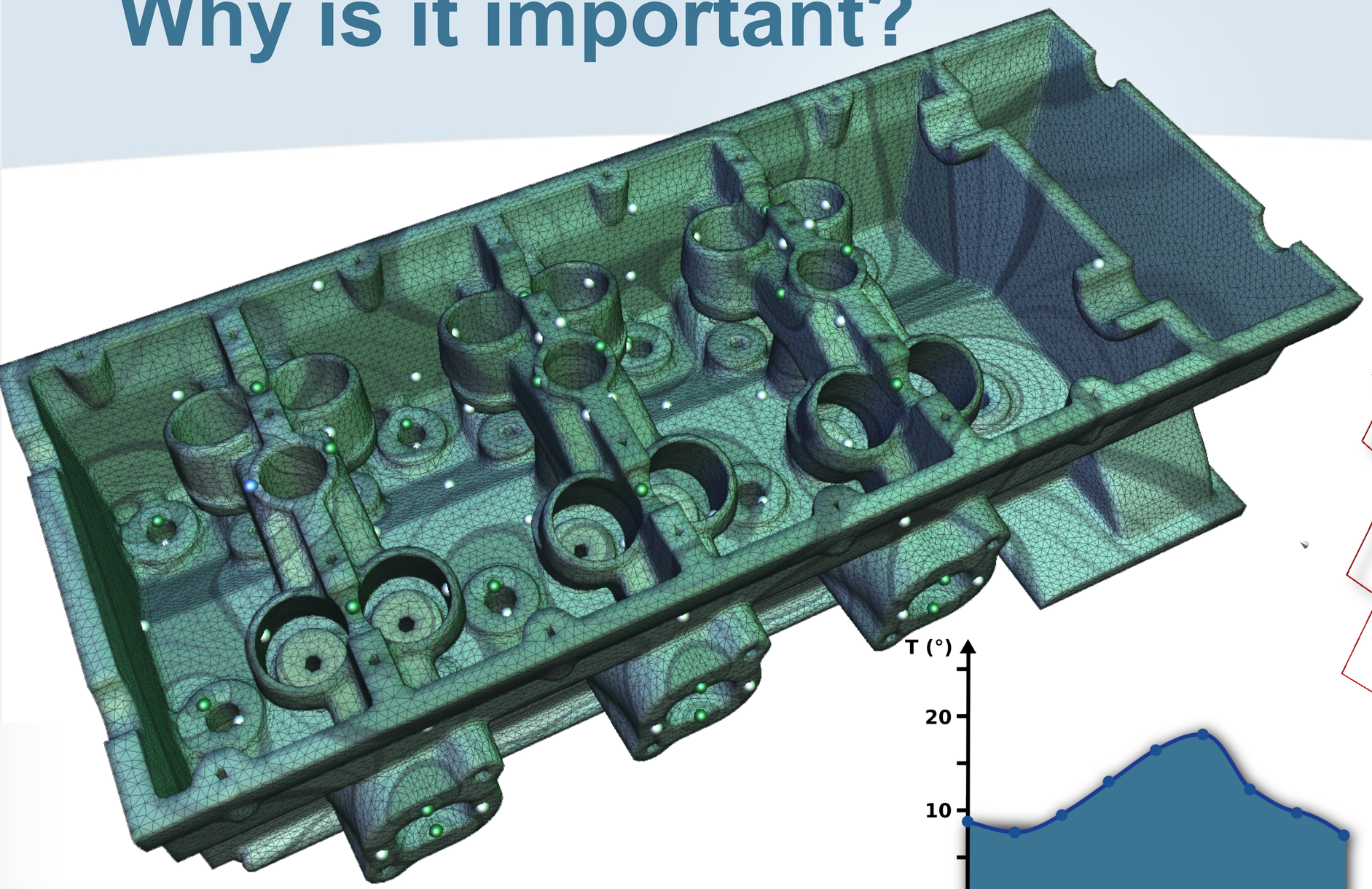


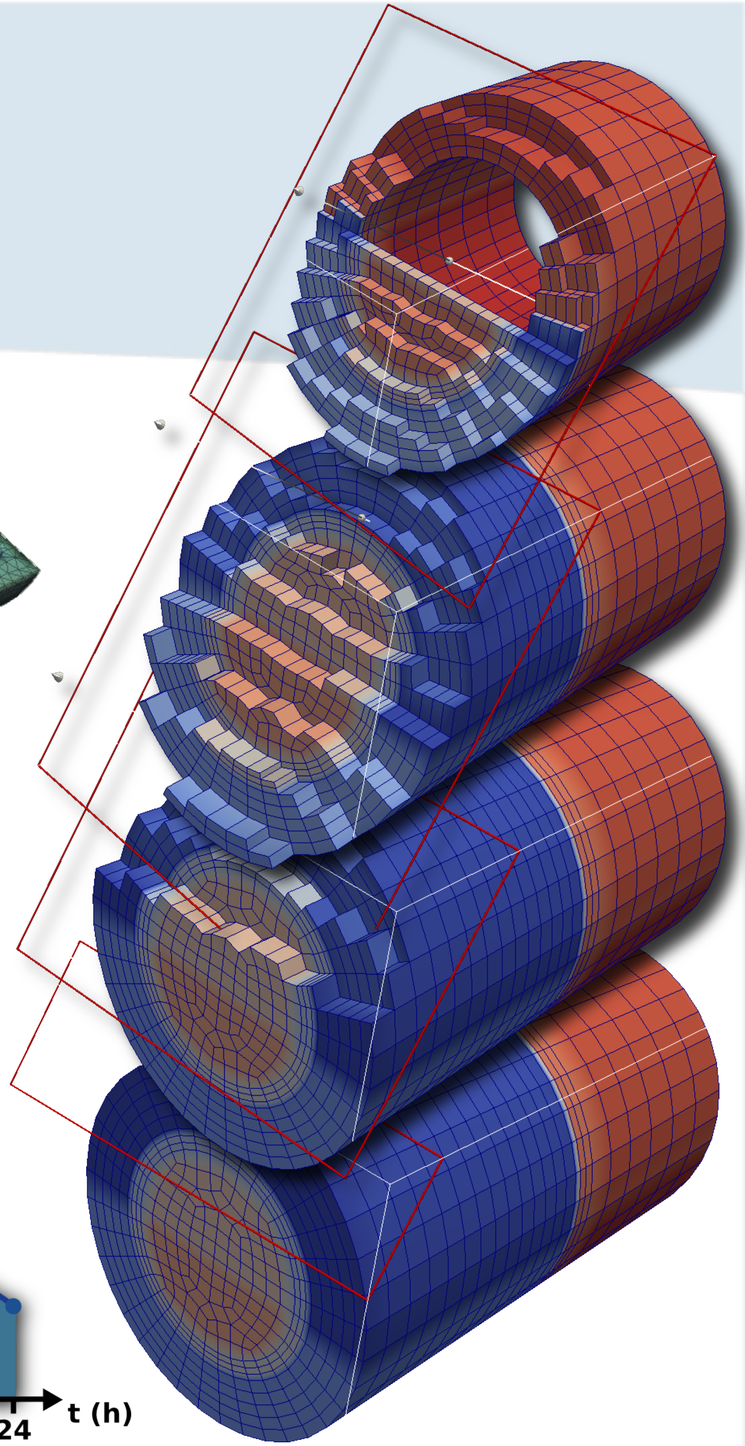
Why is it important?





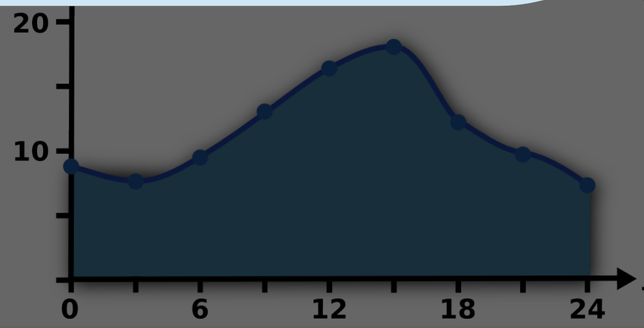
Why is it important?

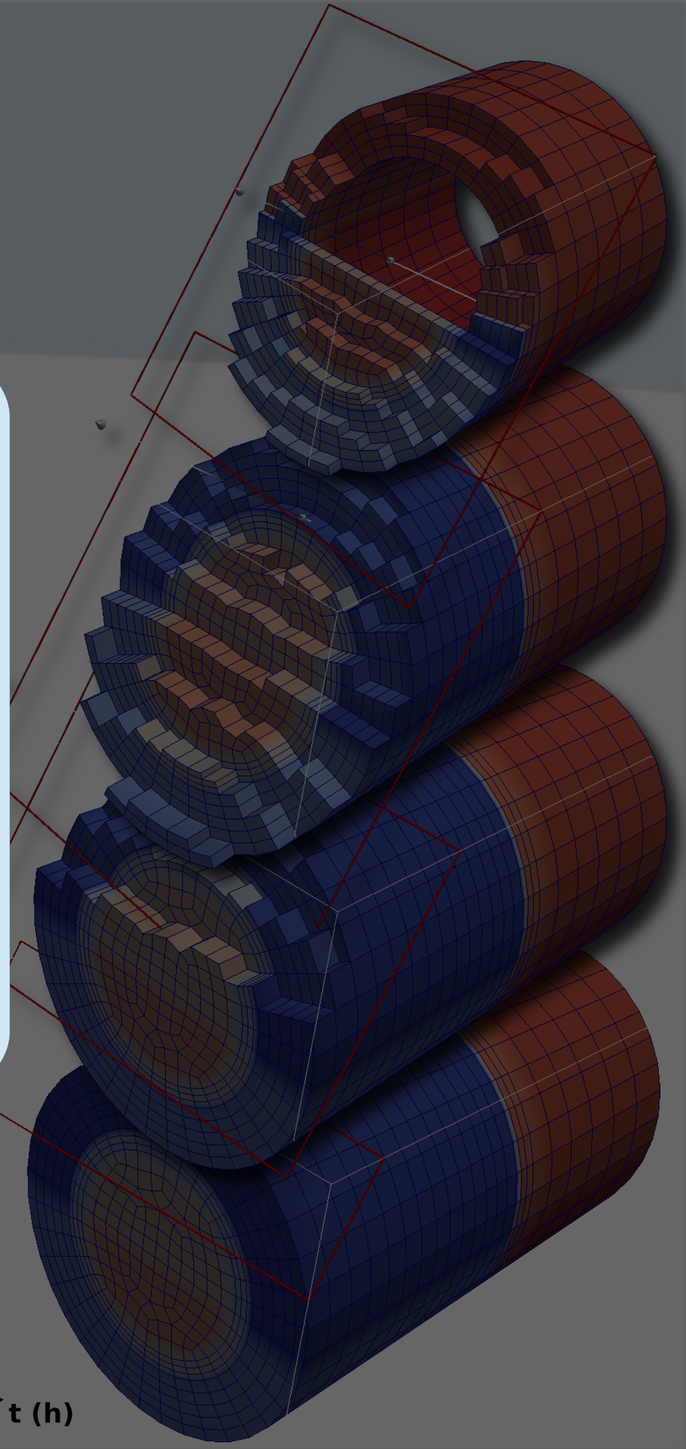




Why is it important?

- Dimension of the domain
- Topology of the domain
- Local topology of the discretization
- …

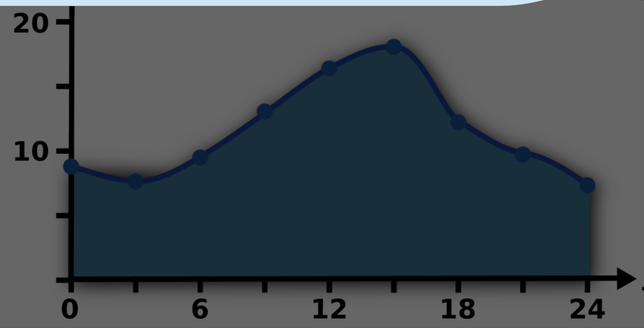


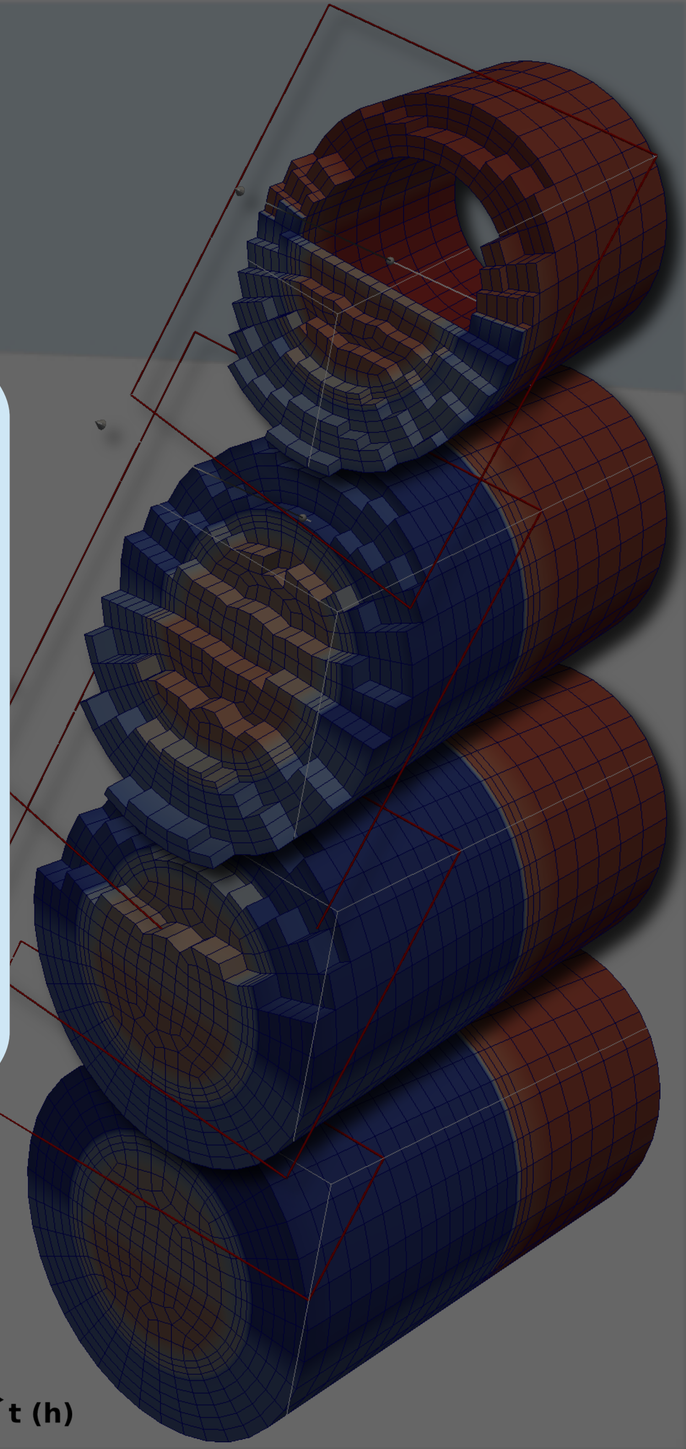


Why is it important?

- Dimension of the domain
- Topology of the domain
- Local topology of the discretization
- …

→ **Adapted algorithms**





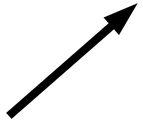
What do we mean by domain?

What do we mean by domain?

$$f : \mathcal{D} \rightarrow \mathcal{A}$$

What do we mean by domain?

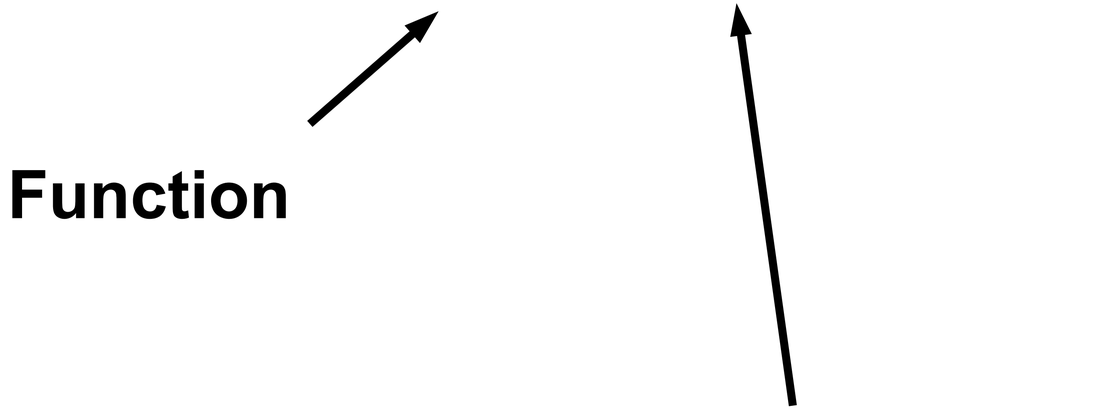
$$f : \mathcal{D} \rightarrow \mathcal{A}$$



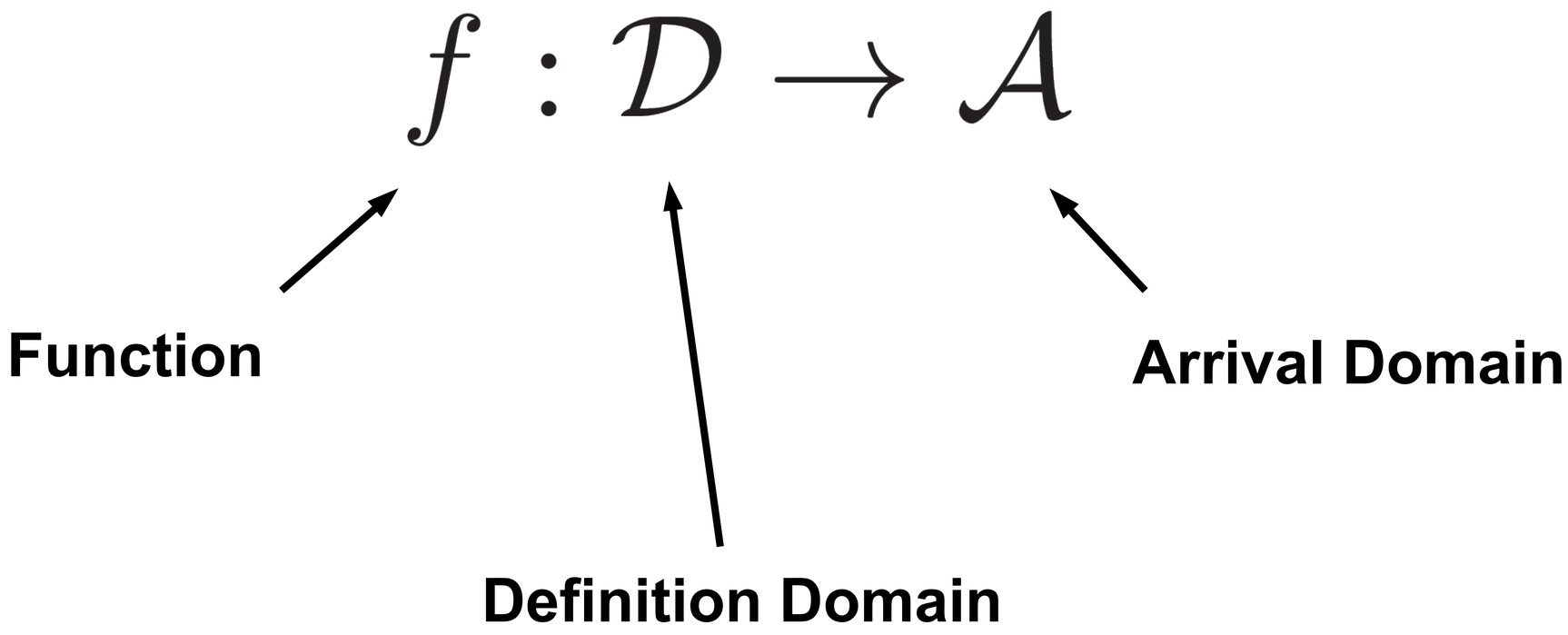
Function

What do we mean by domain?

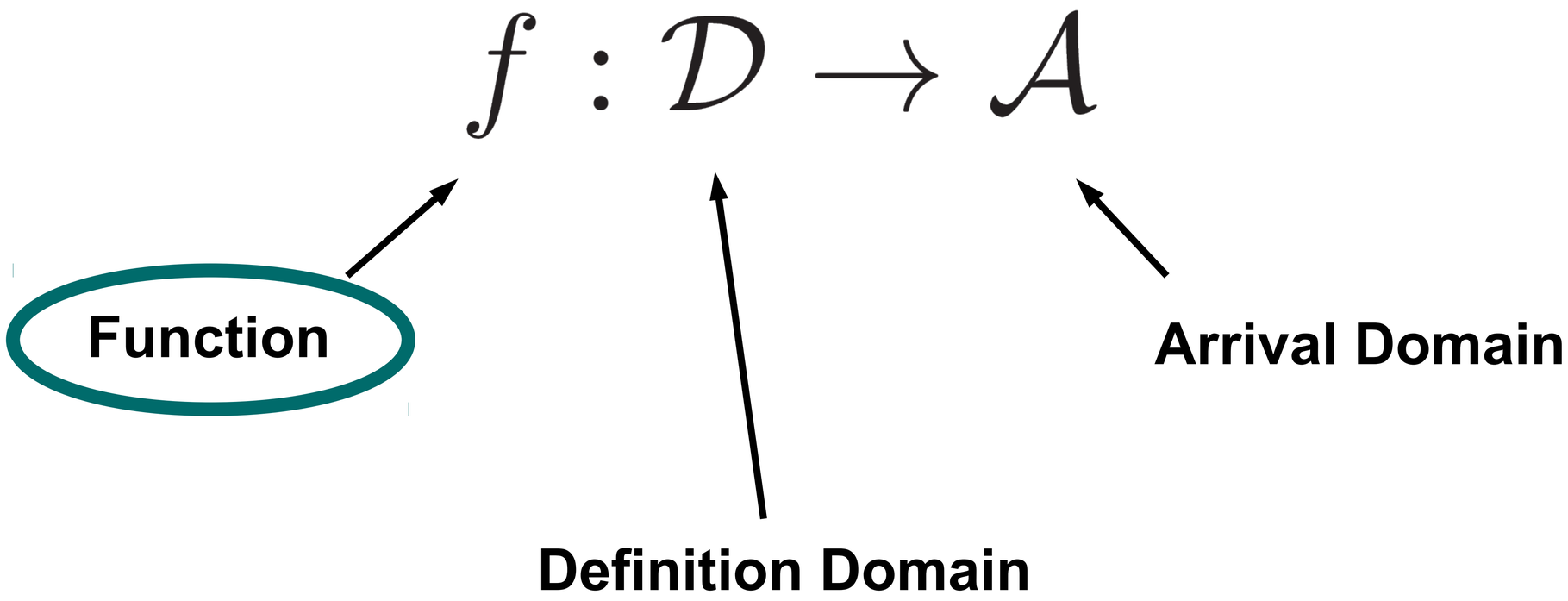
$$f : \mathcal{D} \rightarrow \mathcal{A}$$



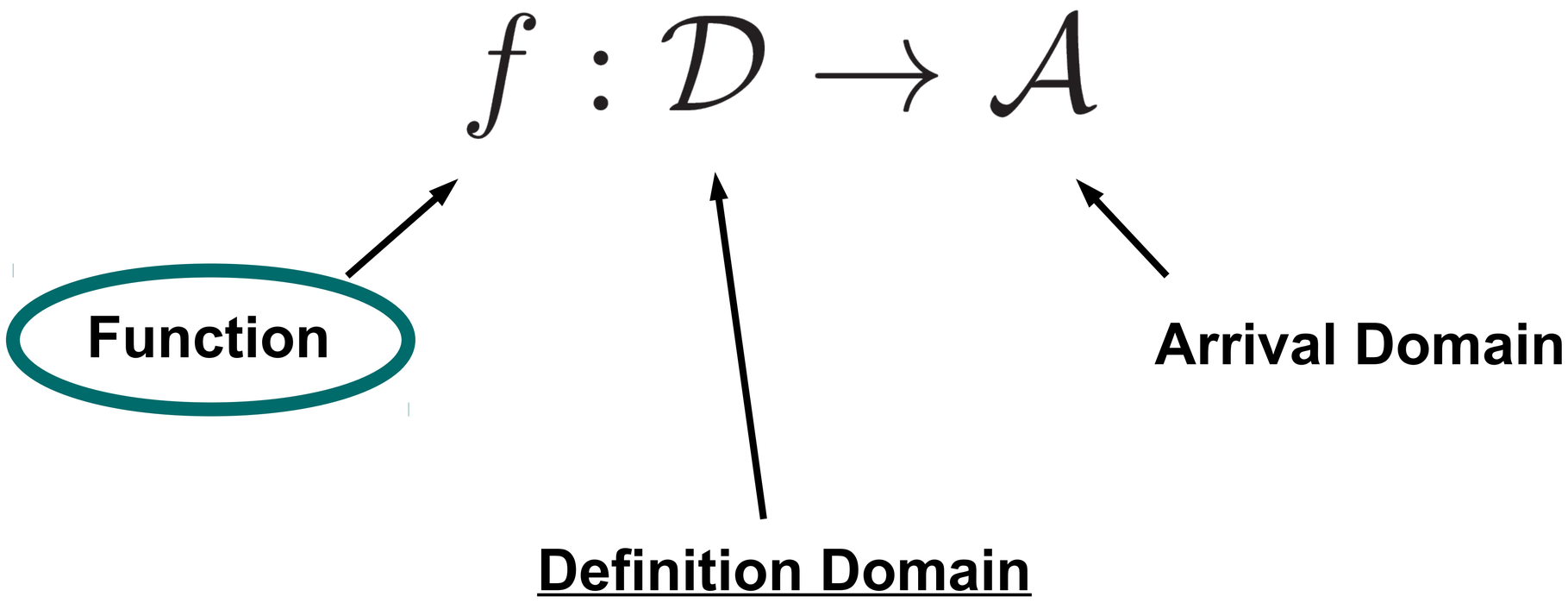
What do we mean by domain?



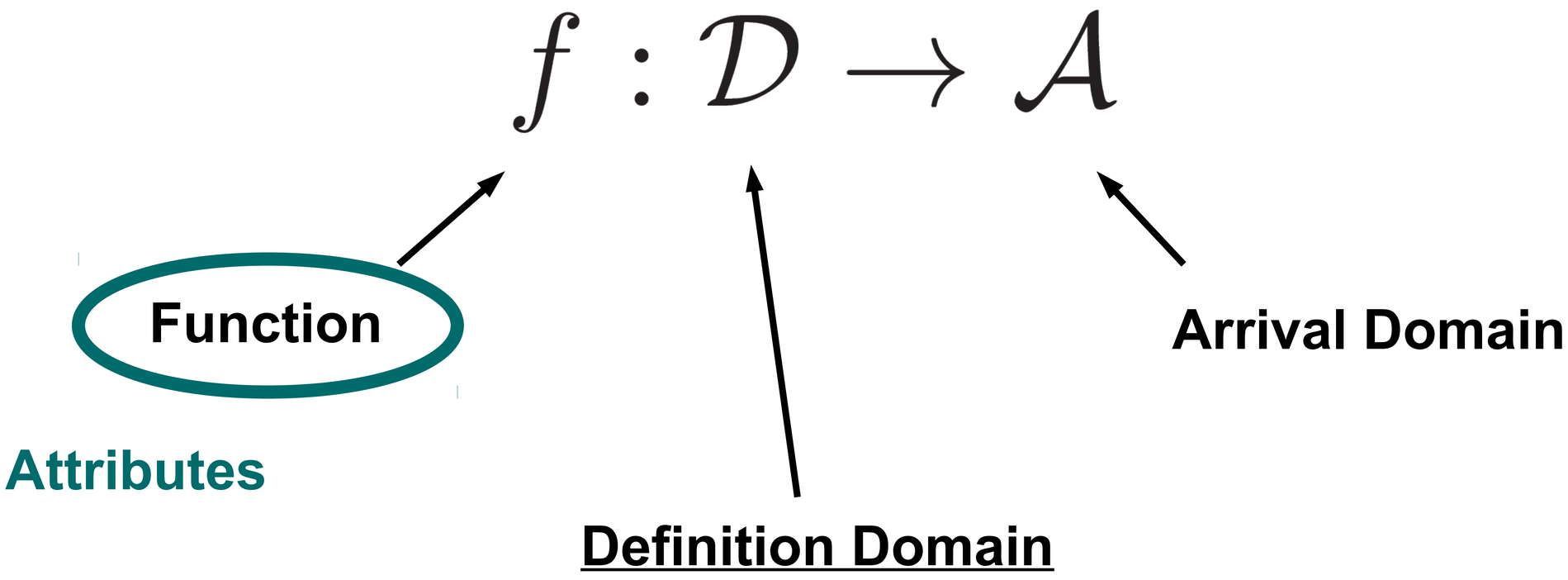
What do we mean by domain?



What do we mean by domain?



What do we mean by domain?



What do we mean by domain?

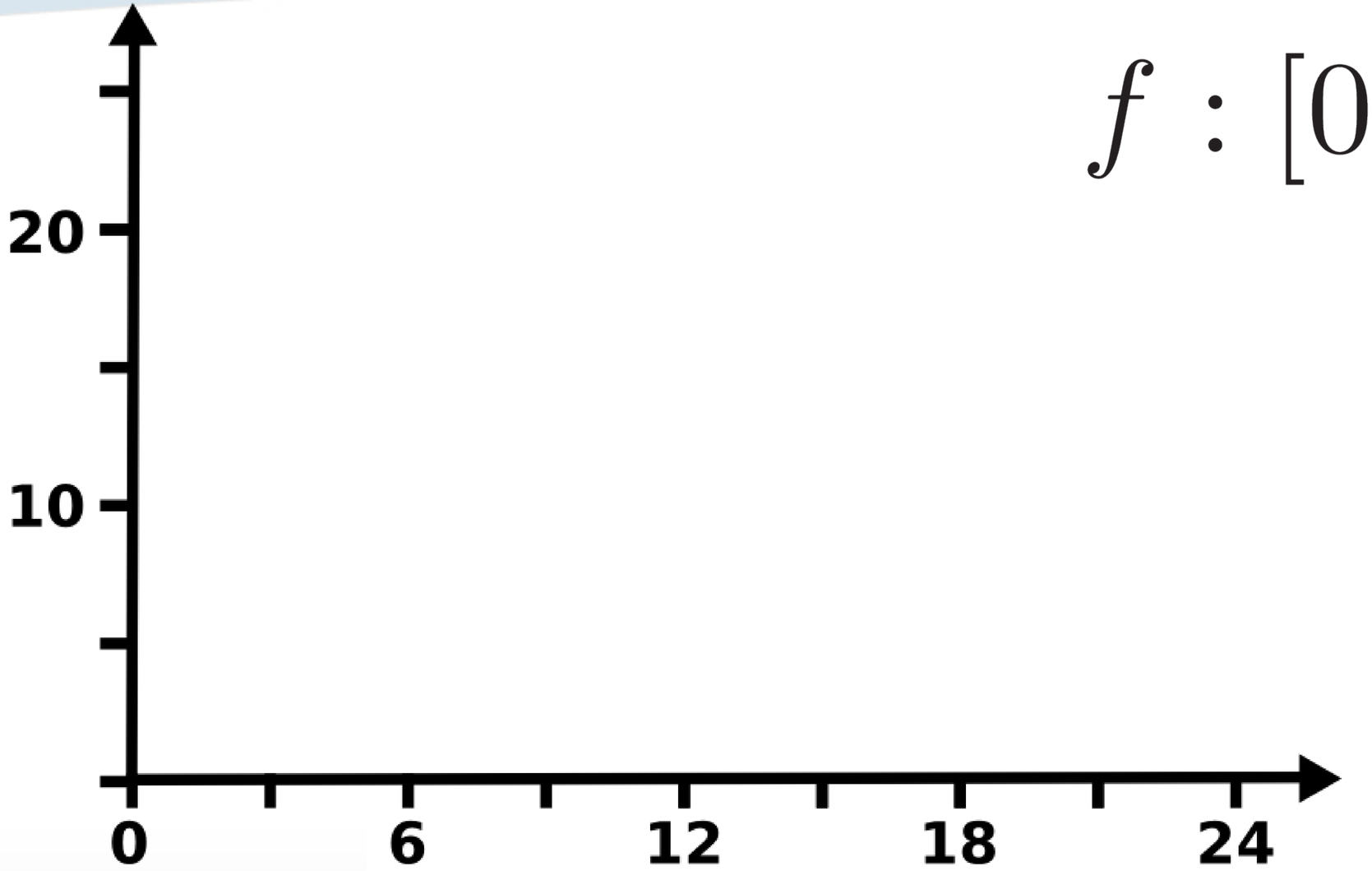
$$f : [0, 24] \to \mathbb{R}$$

What do we mean by domain?

$$f : [0, 24] \to \mathbb{R}$$

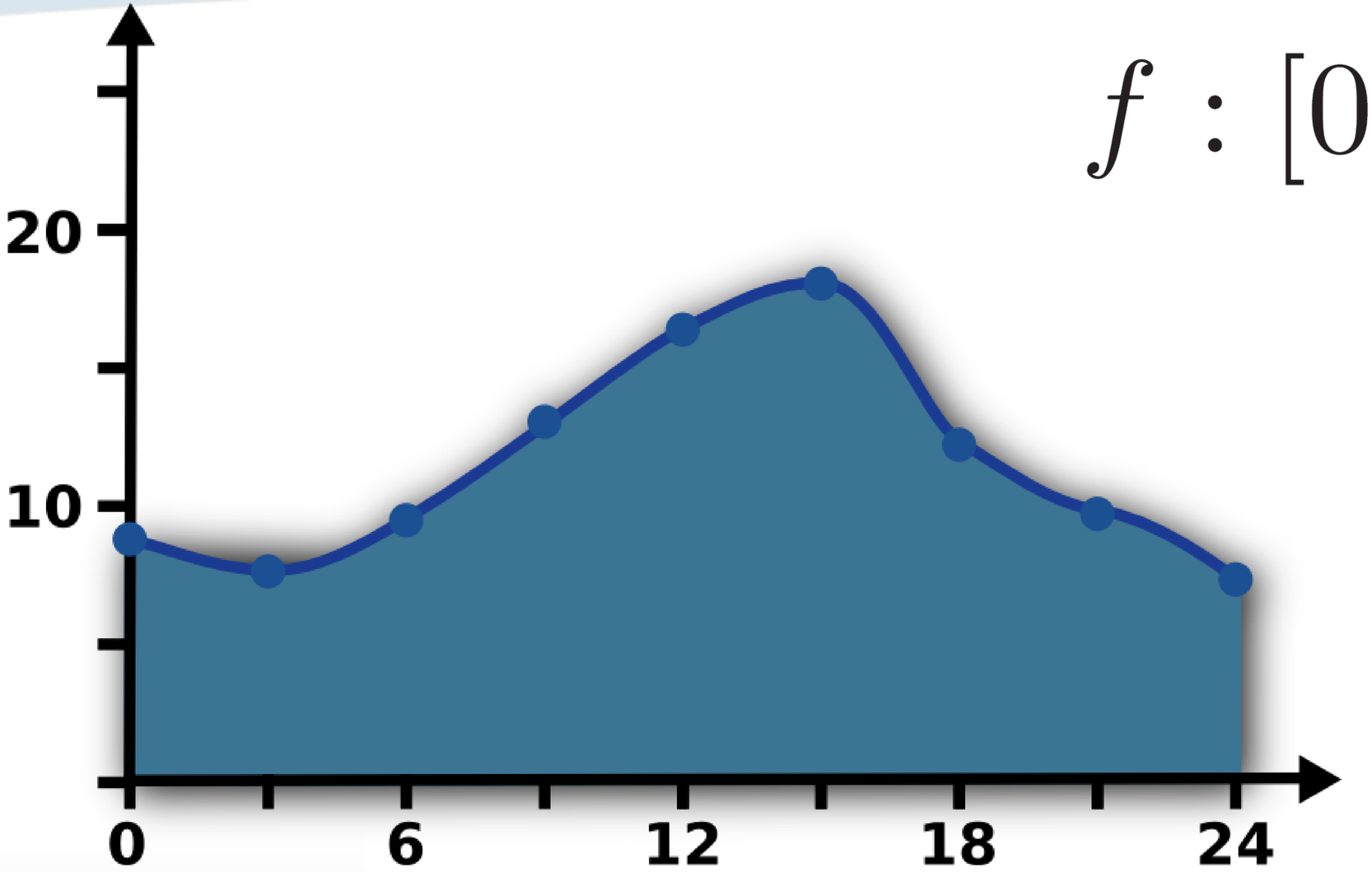
What do we mean by domain?

$$f : [0, 24] \to \mathbb{R}$$

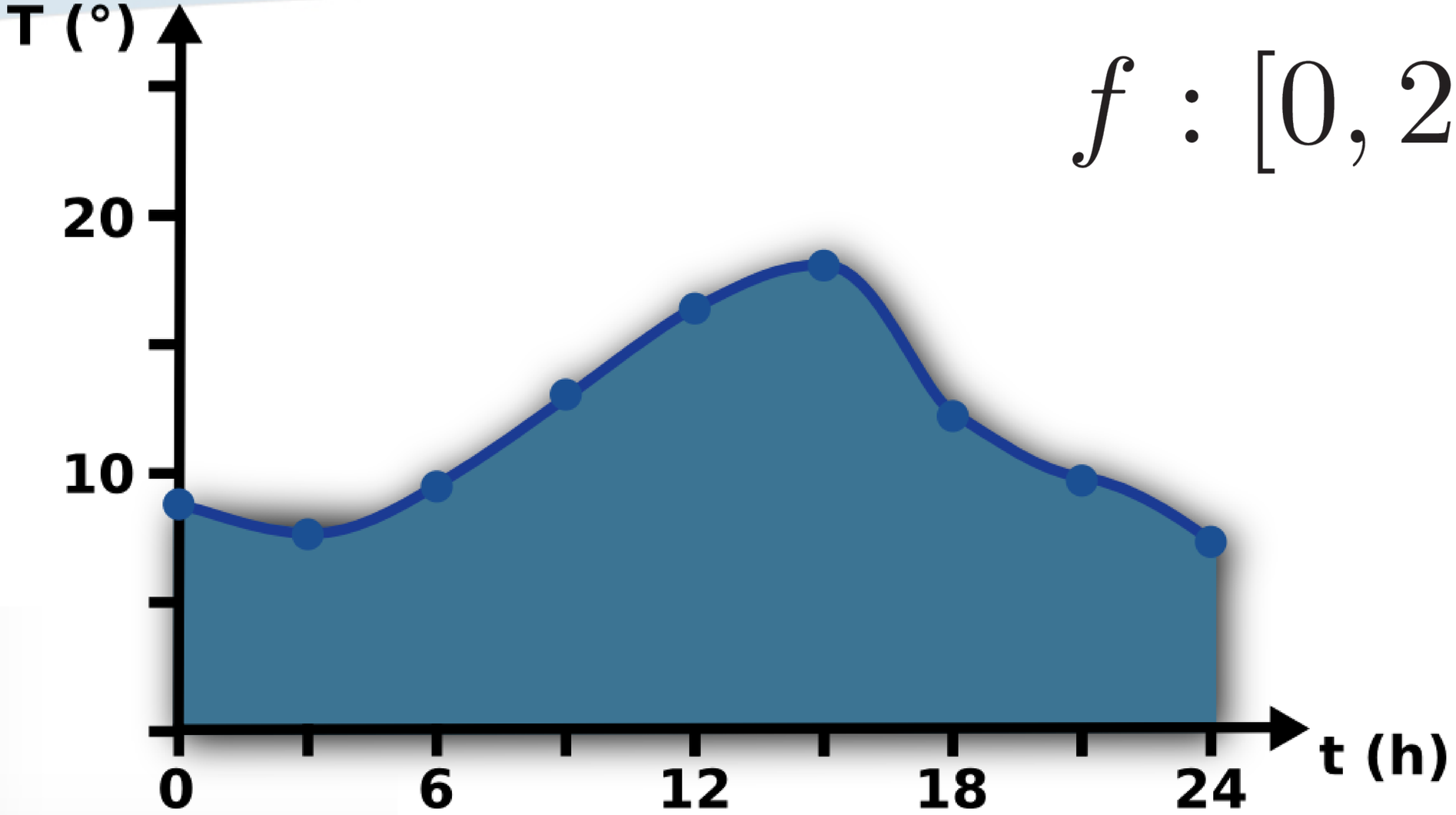


What do we mean by domain?

$$f : [0, 24] \to \mathbb{R}$$



What do we mean by domain?



$$f : [0, 24] \to \mathbb{R}$$

What do we mean by domain?

What do we mean by domain?

- Continuous discrete representation of a space $\mathbb{X}$

What do we mean by domain?

- *Continuous discrete* representation of a space $\mathbb{X}$

What do we mean by domain?

- *Continuous discrete* representation of a space $\mathbb{X}$
  - Finite set of samples

What do we mean by domain?

- *Continuous discrete* representation of a space $\mathbb{X}$
  - Finite set of samples
    - Positional information in an embedding space $\mathbb{E}$

What do we mean by domain?

- *Continuous discrete* representation of a space $\mathbb{X}$
  - Finite set of samples
    - Positional information in an embedding space $\mathbb{E}$
    - Attributes

What do we mean by domain?

- *Continuous discrete* representation of a space $\mathbb{X}$
  - Finite set of samples
    - Positional information in an embedding space $\mathbb{E}$
    - Attributes

What do we mean by domain?

- *Continuous discrete* representation of a space $\mathbb{X}$
  - Finite set of samples
    - Positional information in an embedding space $\mathbb{E}$
    - Attributes
  - Connectivity of the samples

What do we mean by domain?

- *Continuous discrete* representation of a space $\mathbb{X}$
  - Finite set of samples
    - Positional information in an embedding space $\mathbb{E}$
    - Attributes
  - Connectivity of the samples
    - Continuity of the domain

What do we mean by domain?

- *Continuous discrete* representation of a space $\mathbb{X}$
  - Finite set of samples
    - Positional information in an embedding space $\mathbb{E}$
    - Attributes
  - Connectivity of the samples
    - Continuity of the domain
    - Cellular elements (dimension of $\mathbb{X}$)

What do we mean by domain?

- *Continuous discrete* representation of a space $\mathbb{X}$
  - Finite set of samples
    - Positional information in an embedding space $\mathbb{E}$
    - Attributes
  - Connectivity of the samples
    - Continuity of the domain
    - Cellular elements (dimension of $\mathbb{X}$)
  - Cell interpolation scheme

What do we mean by domain?

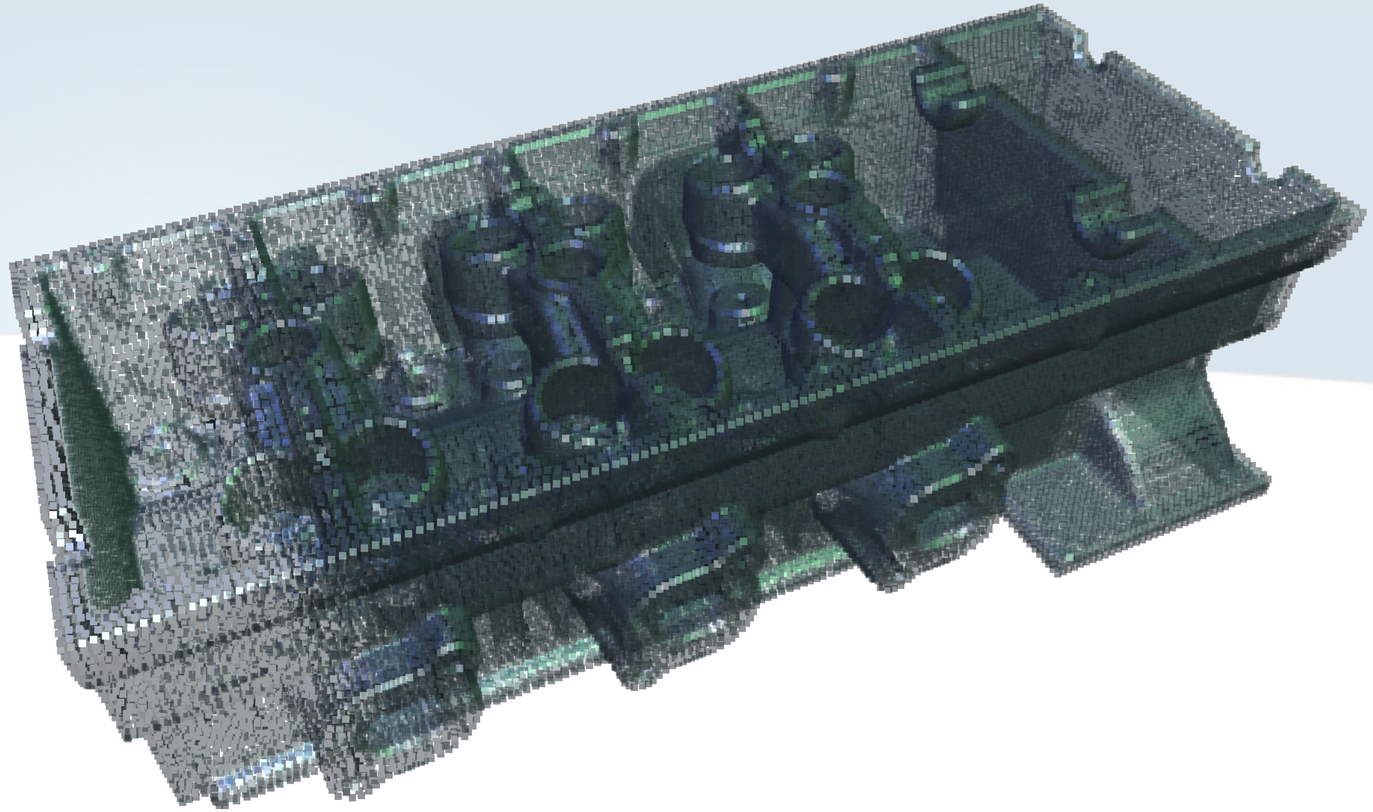
- *Continuous discrete* representation of a space $\mathbb{X}$
  - Finite set of samples
    - Positional information in an embedding space $\mathbb{E}$
    - Attributes
  - Connectivity of the samples
    - Continuity of the domain
    - Cellular elements (dimension of $\mathbb{X}$)
  - Cell interpolation scheme
    - Continuity of the attributes

What do we mean by domain?

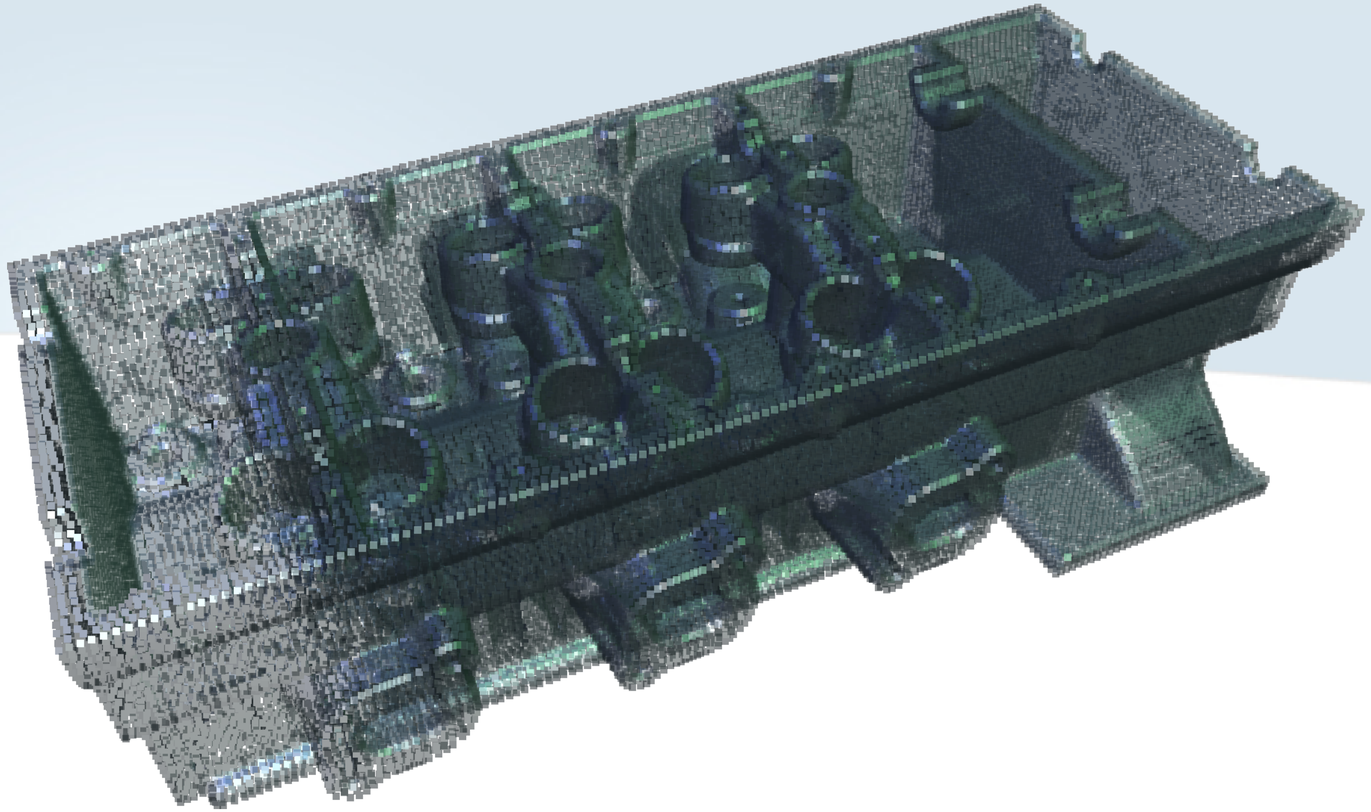
- *Continuous discrete* representation of a space $\mathbb{X}$
  - **Finite set of samples**
    - Positional information in an embedding space $\mathbb{E}$
    - Attributes
  - **Connectivity of the samples**
    - Continuity of the domain
    - Cellular elements (dimension of $\mathbb{X}$)
  - **Cell interpolation scheme**
    - Continuity of the attributes

Example

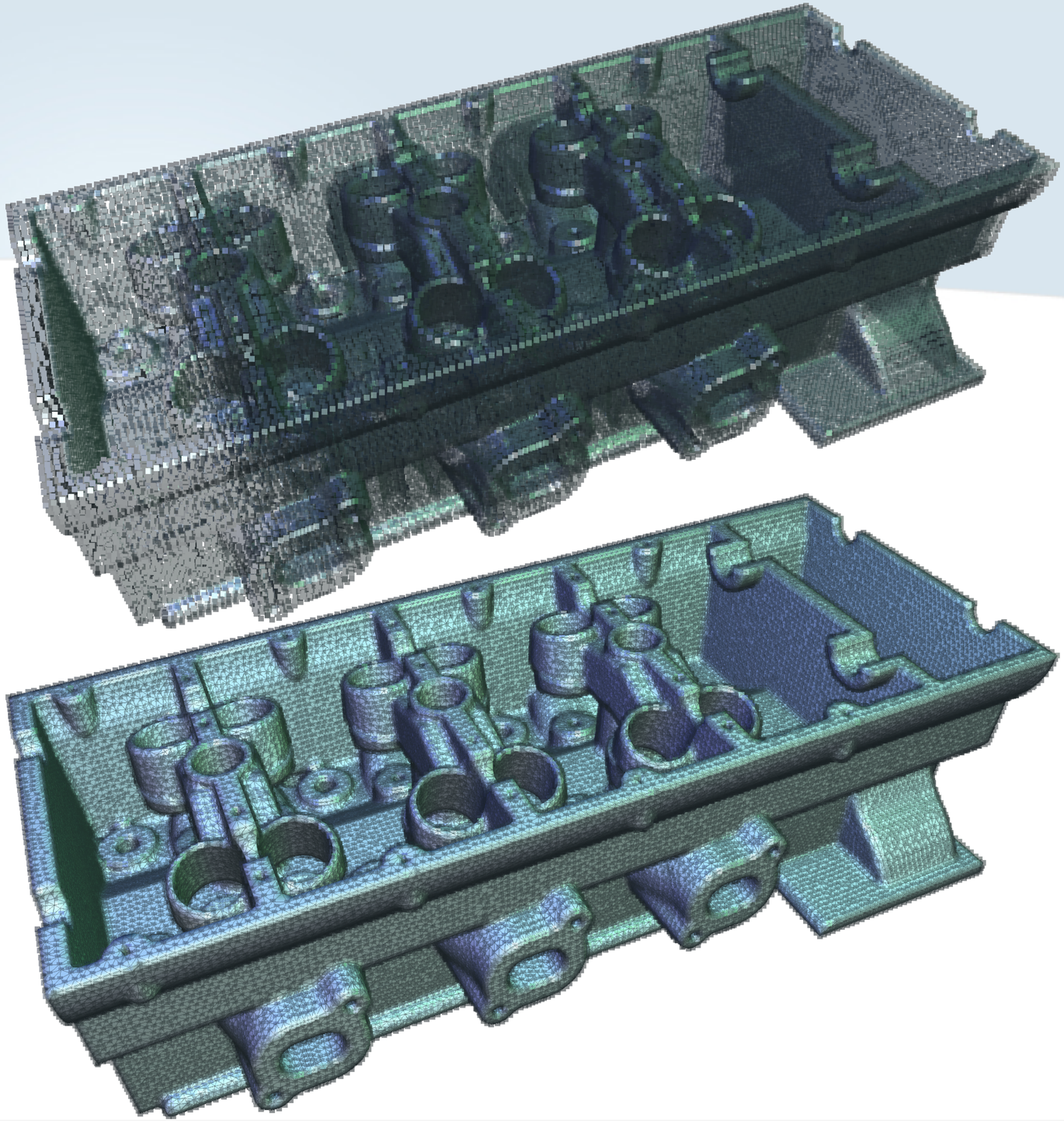
Example



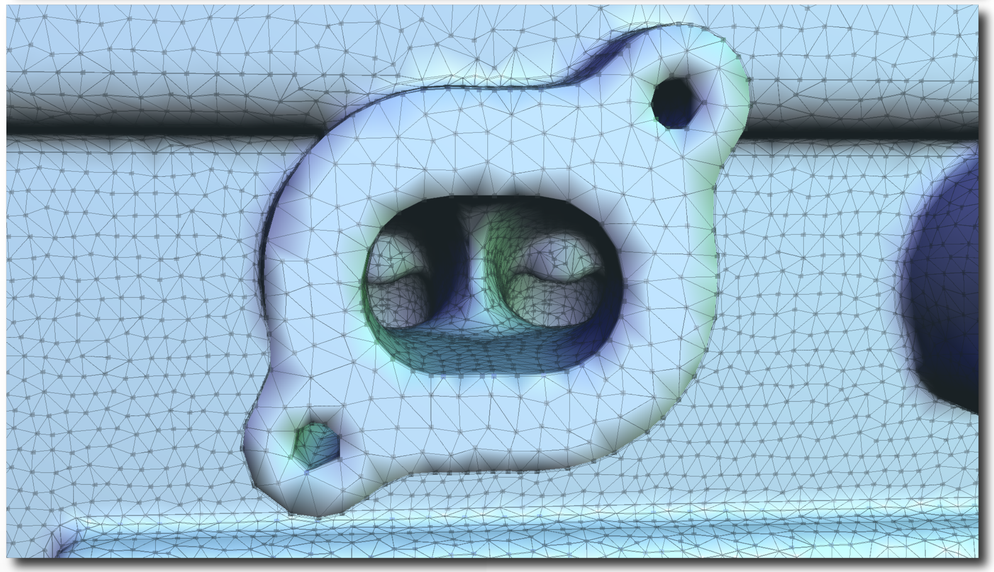
Example

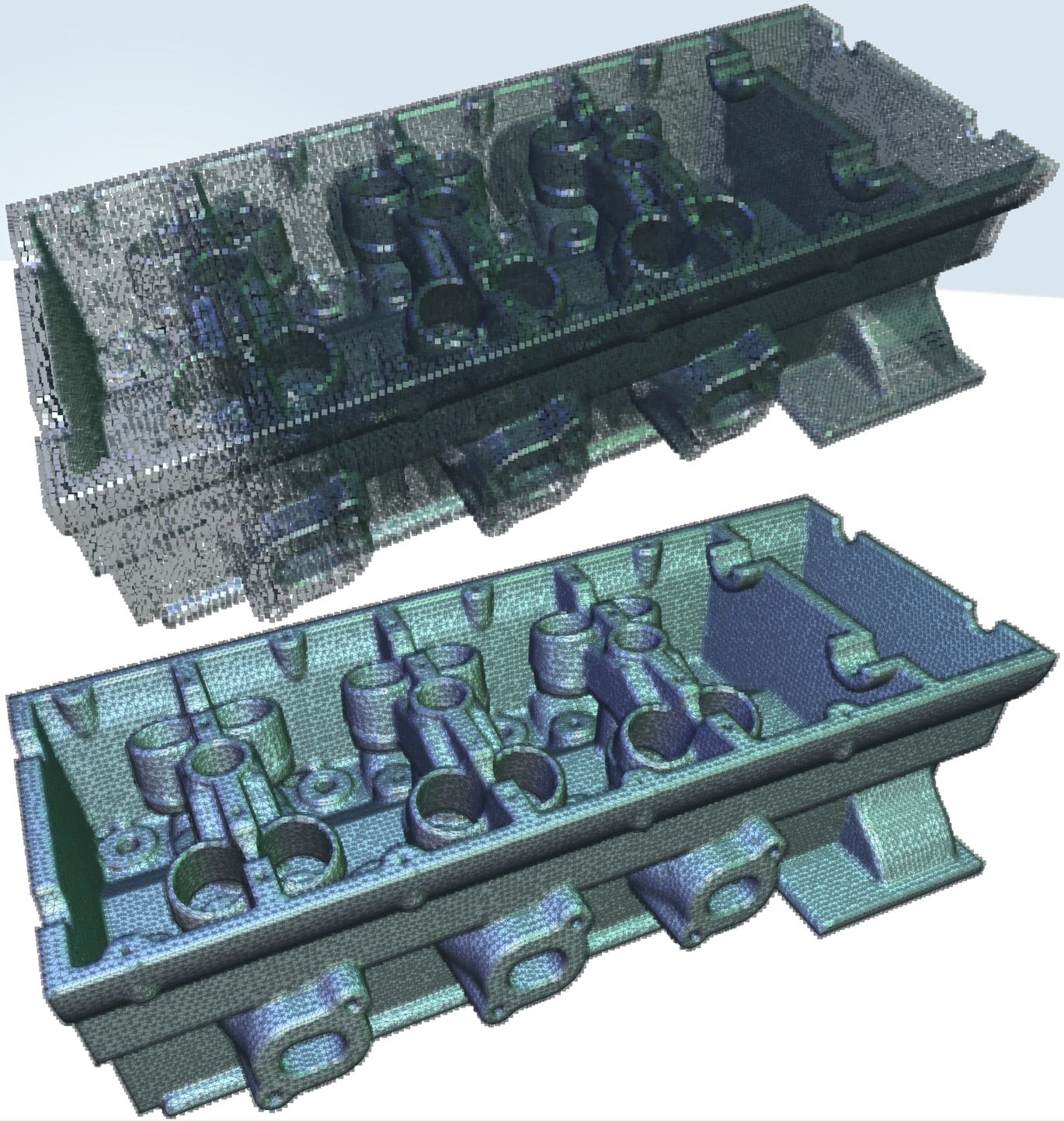


Example

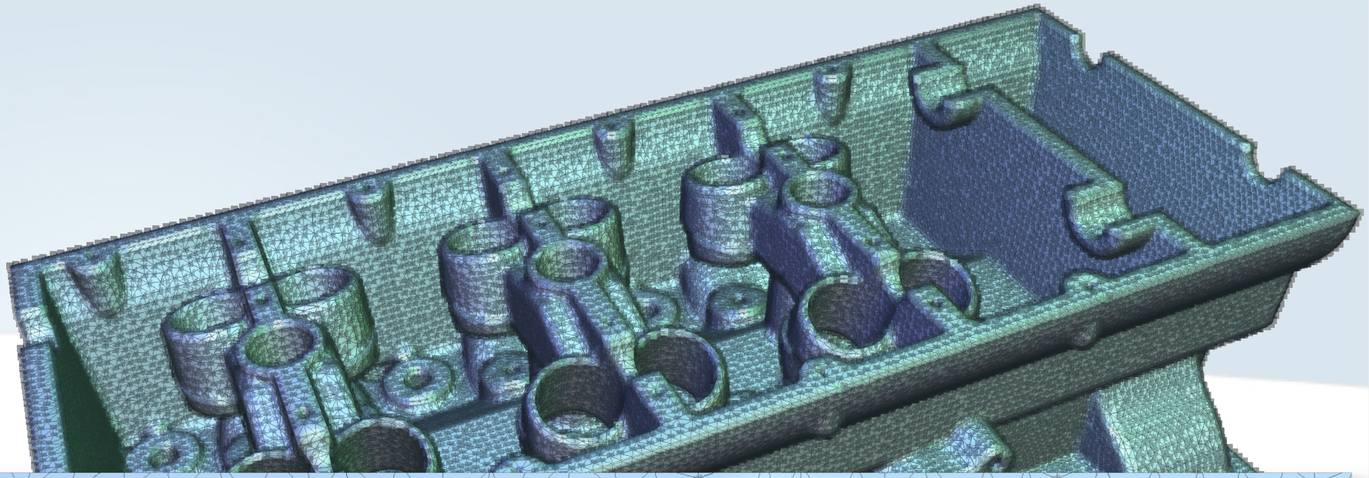


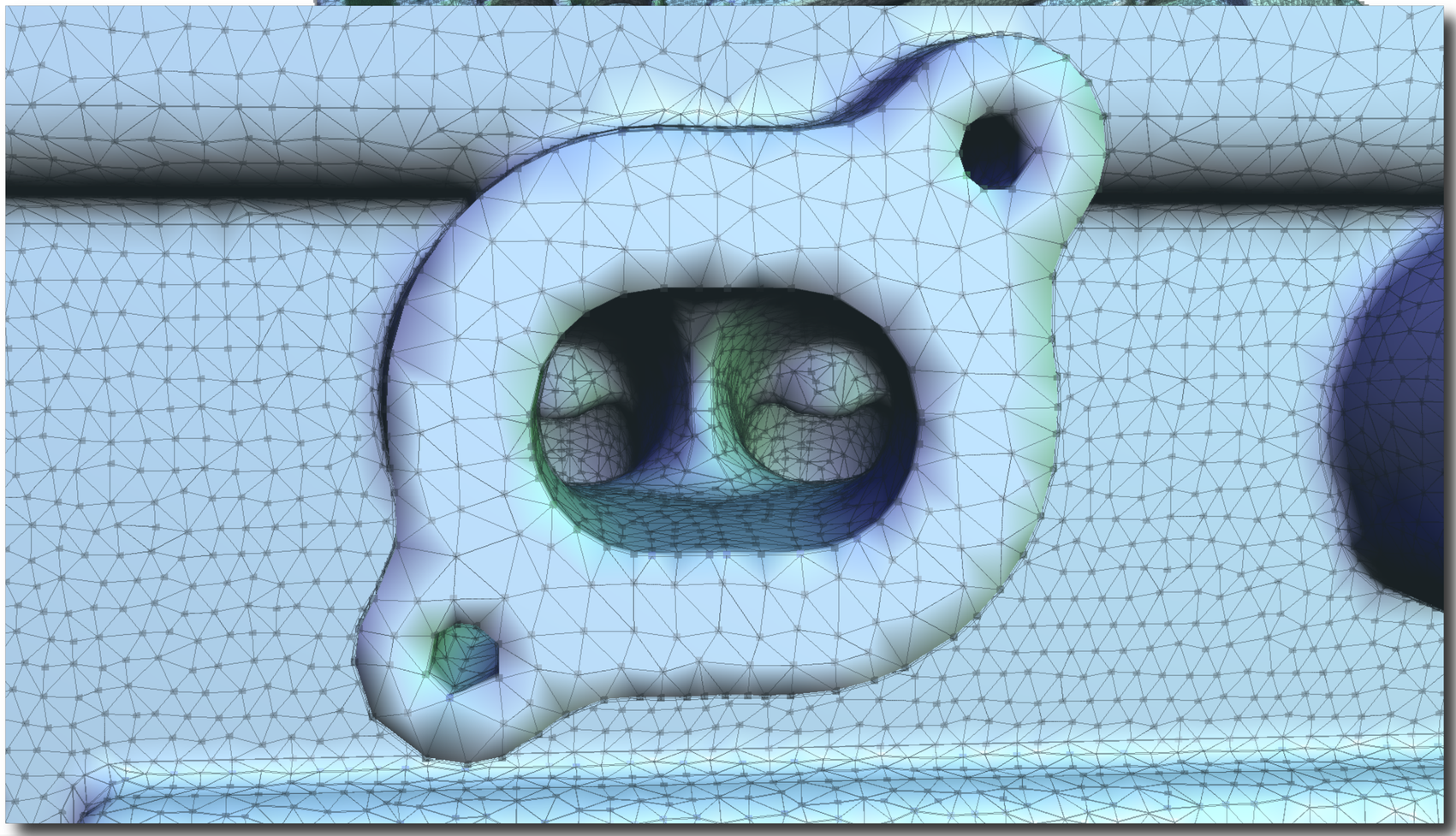
Example



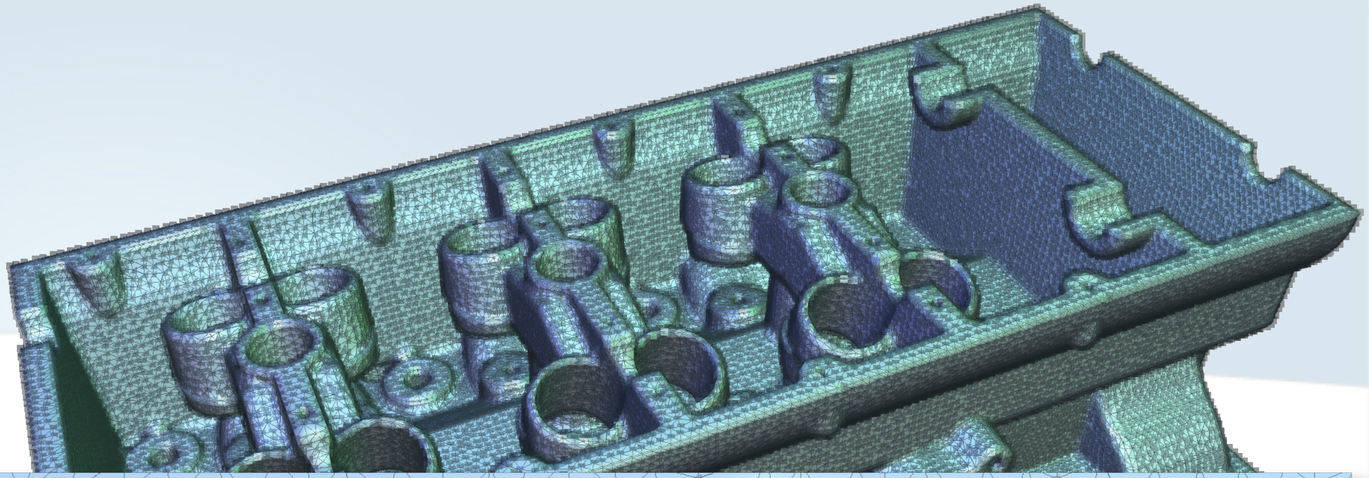


Example

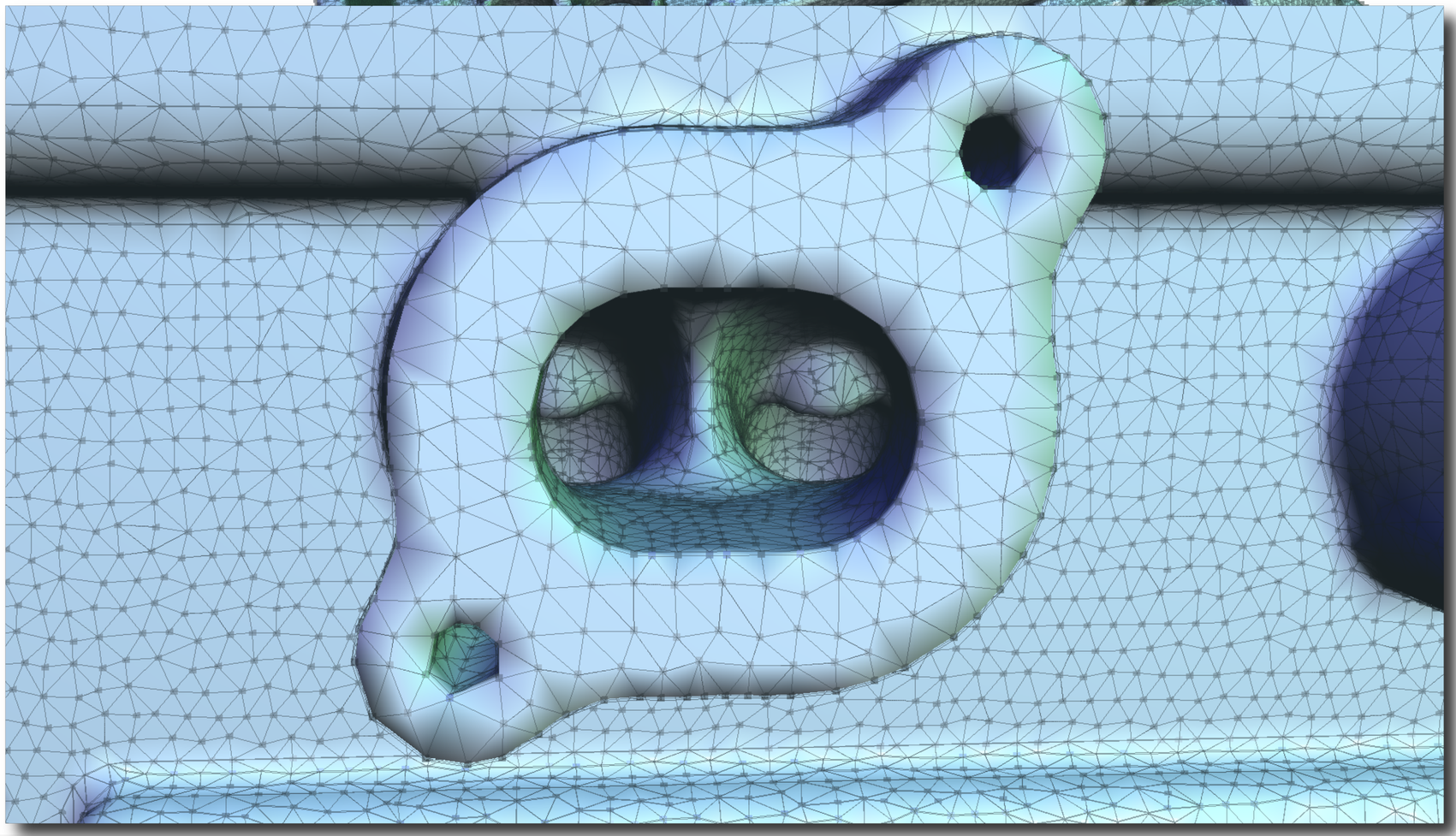




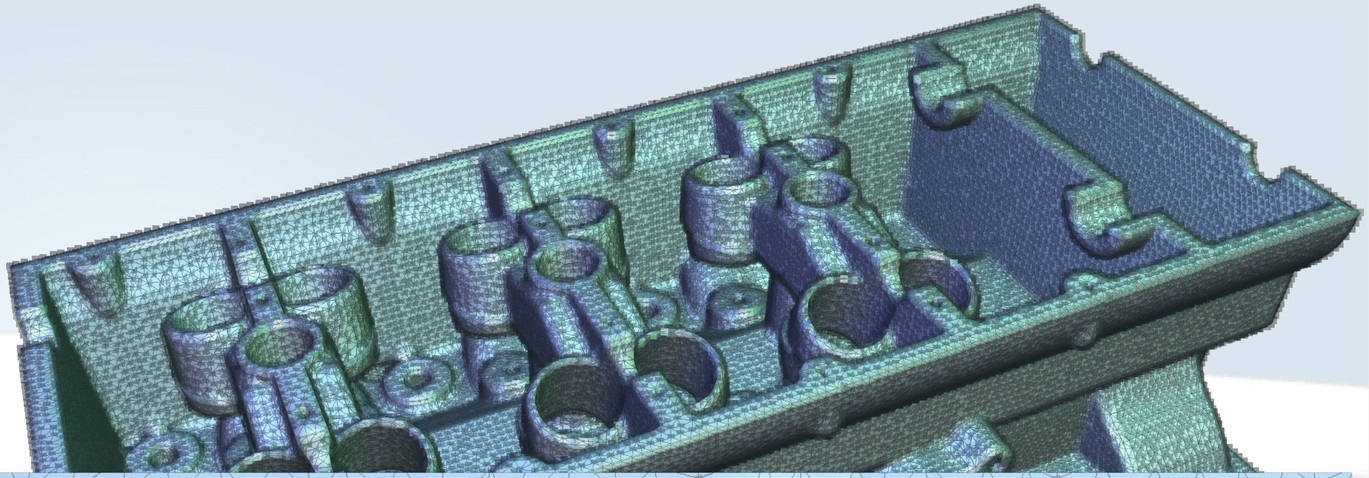
Example



- Domain
- Embedding space
- Cellular elements

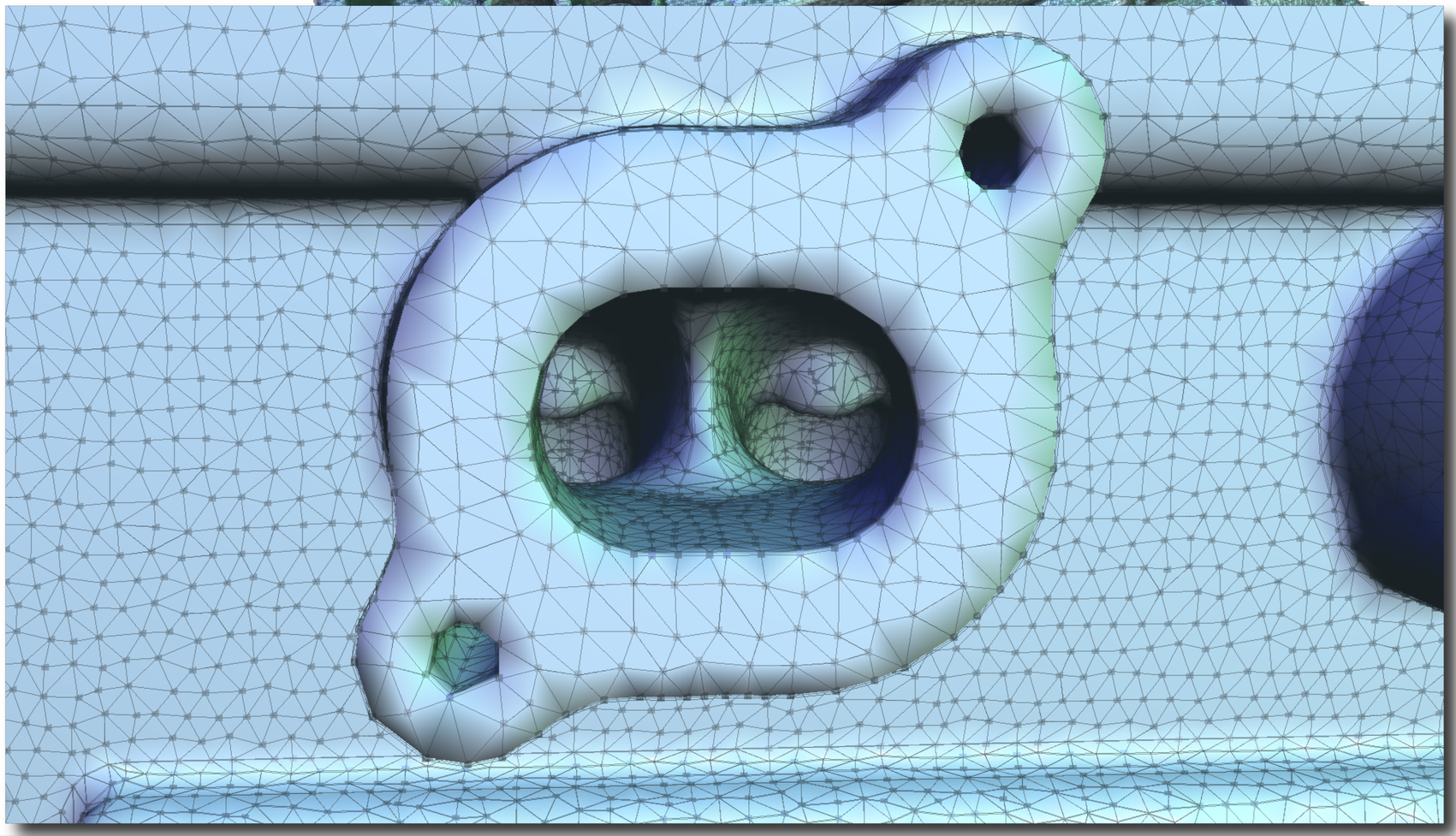


Example

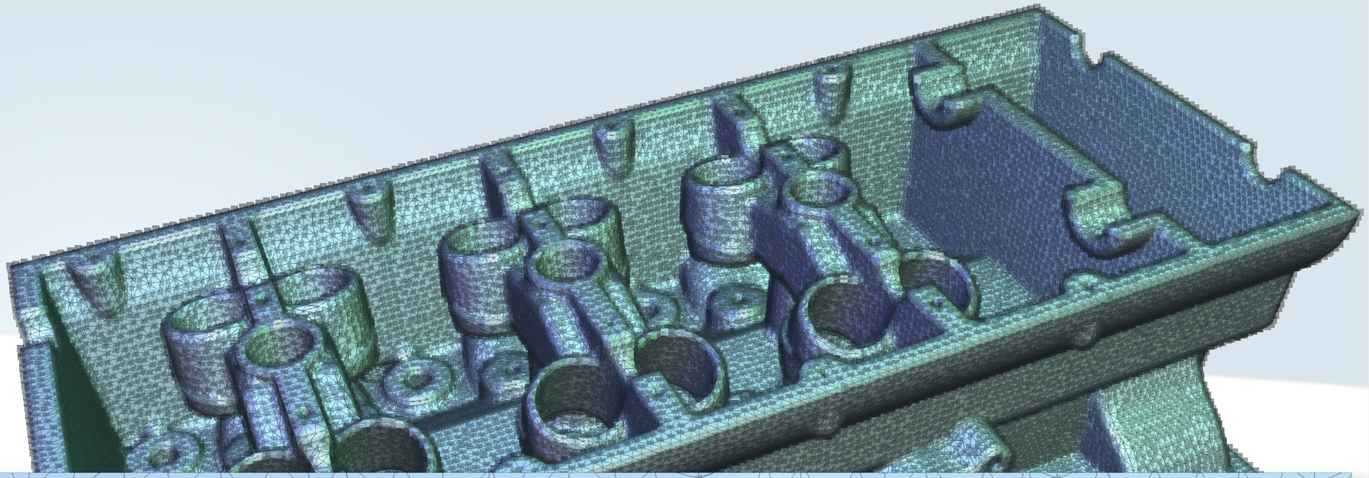


- Domain
  - 2-manifold $\mathcal{S}$
- Embedding space

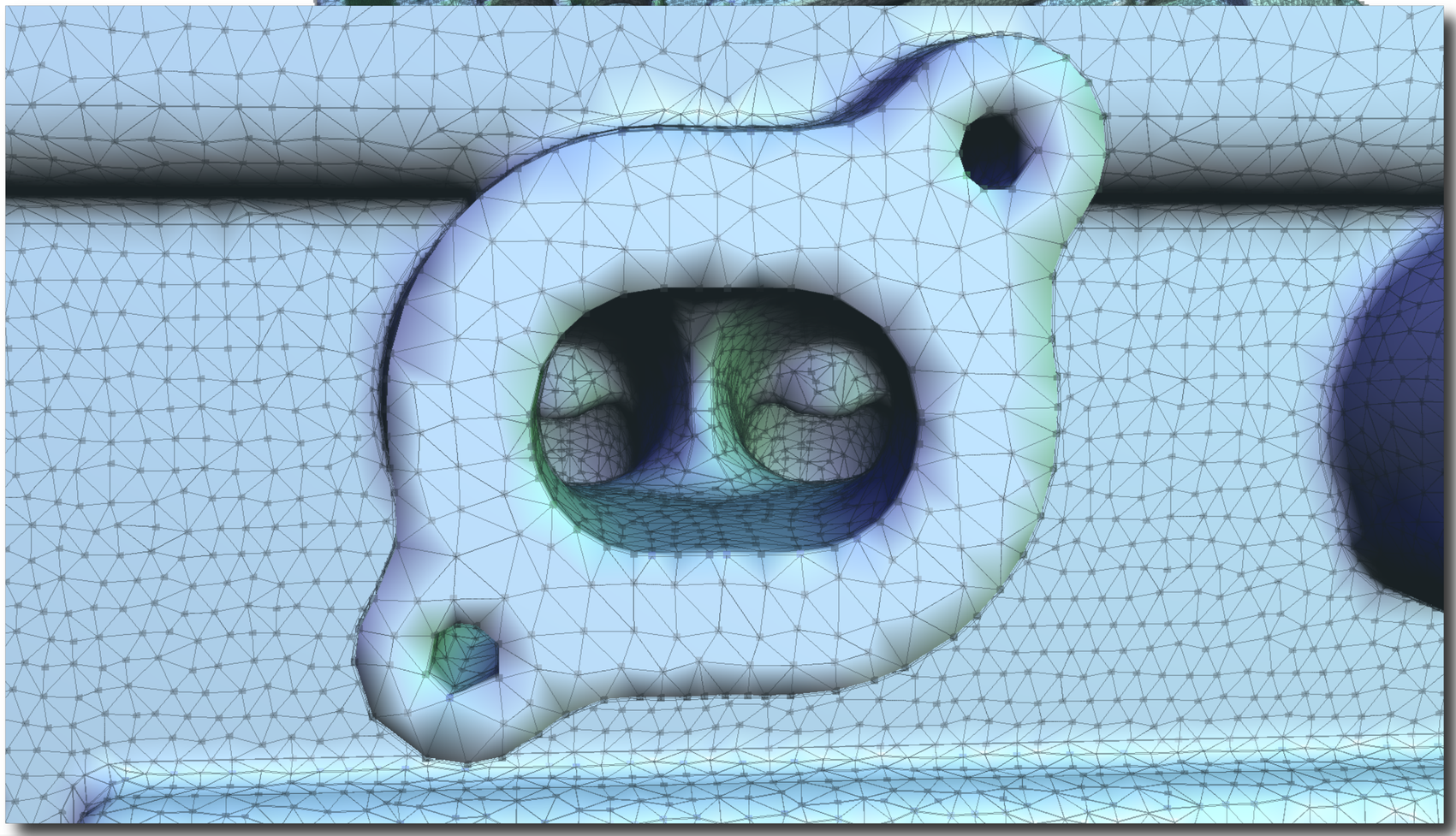
- Cellular elements



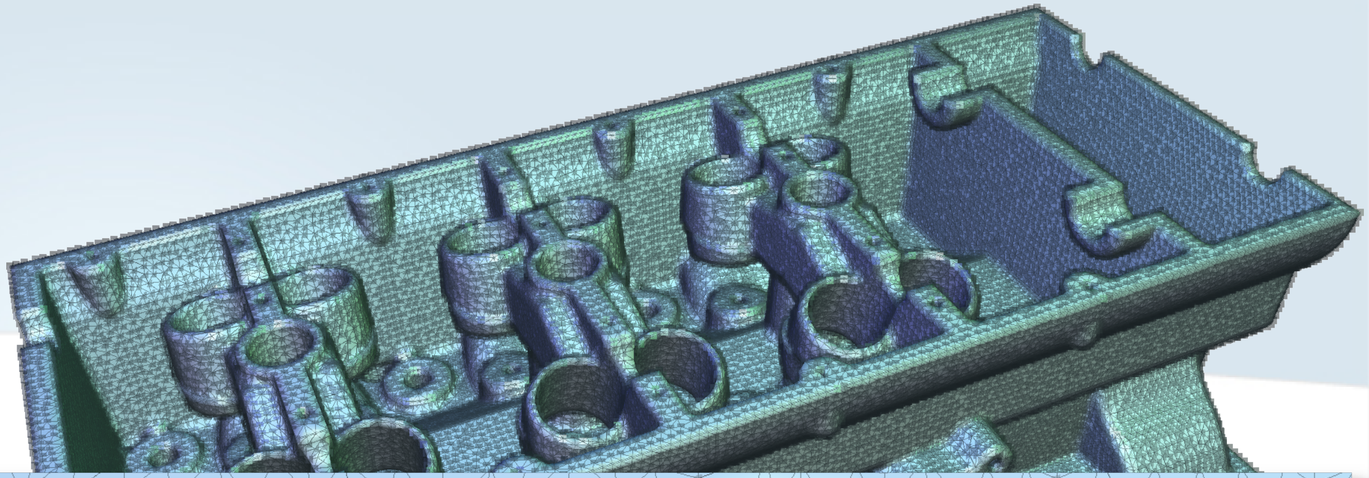
Example



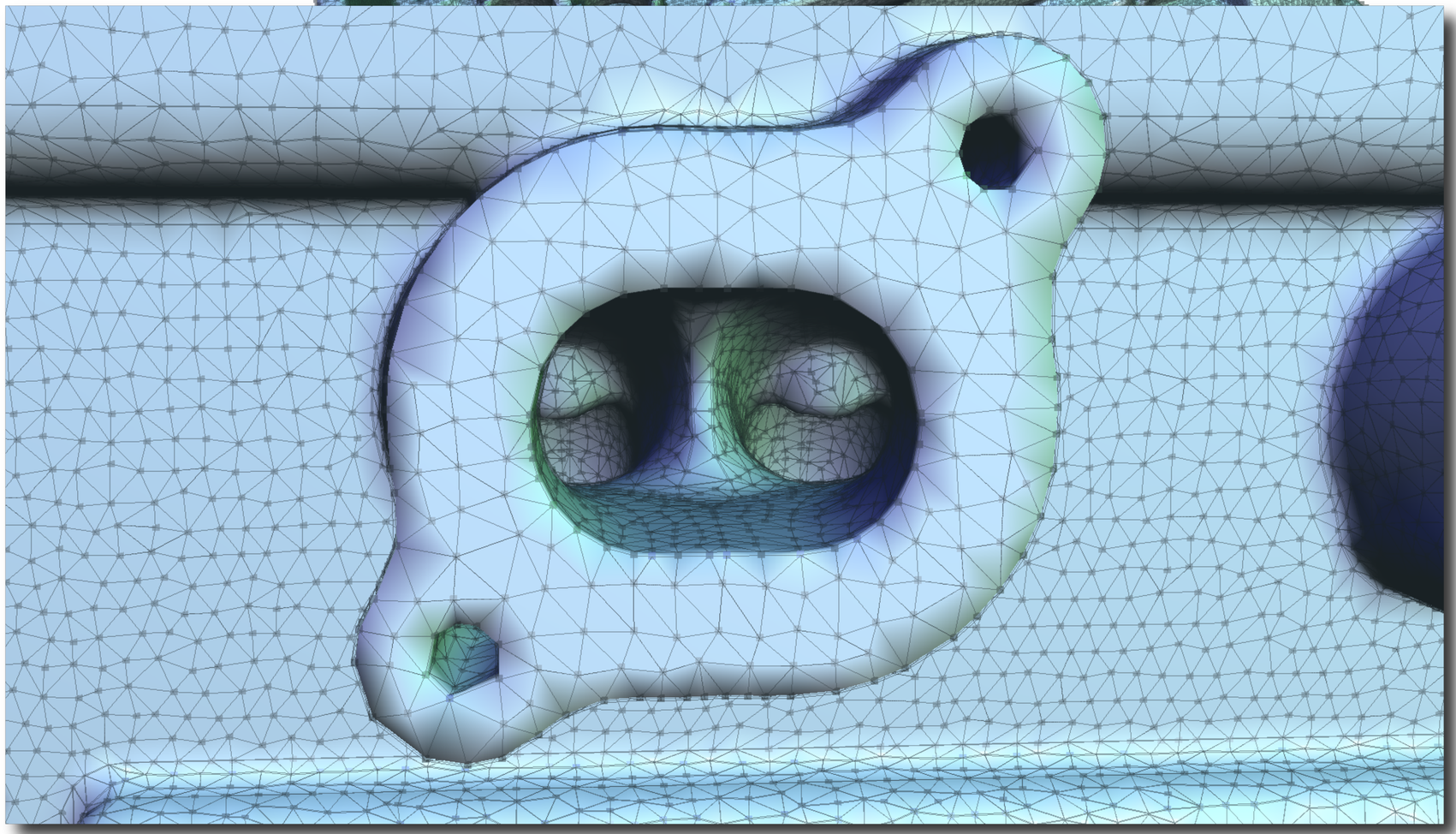
- Domain
  - 2-manifold $\mathcal{S}$
- Embedding space
  - $\mathbb{R}^3$
- Cellular elements



Example



- Domain
  - 2-manifold $\mathcal{S}$
- Embedding space
  - $\mathbb{R}^3$
- Cellular elements
  - Triangles



Domains of interest: Manifolds

Domains of interest: Manifolds

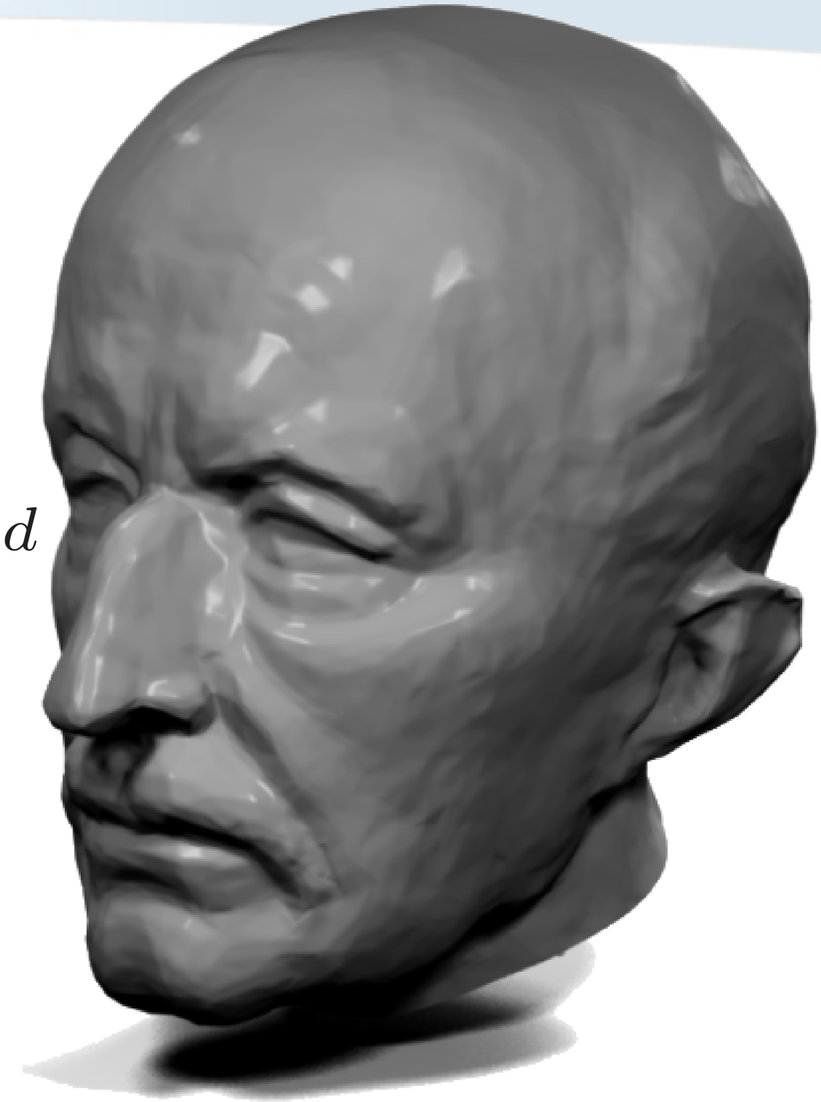
- d-manifold

Domains of interest: Manifolds

- d-manifold
  - Topological space such that:

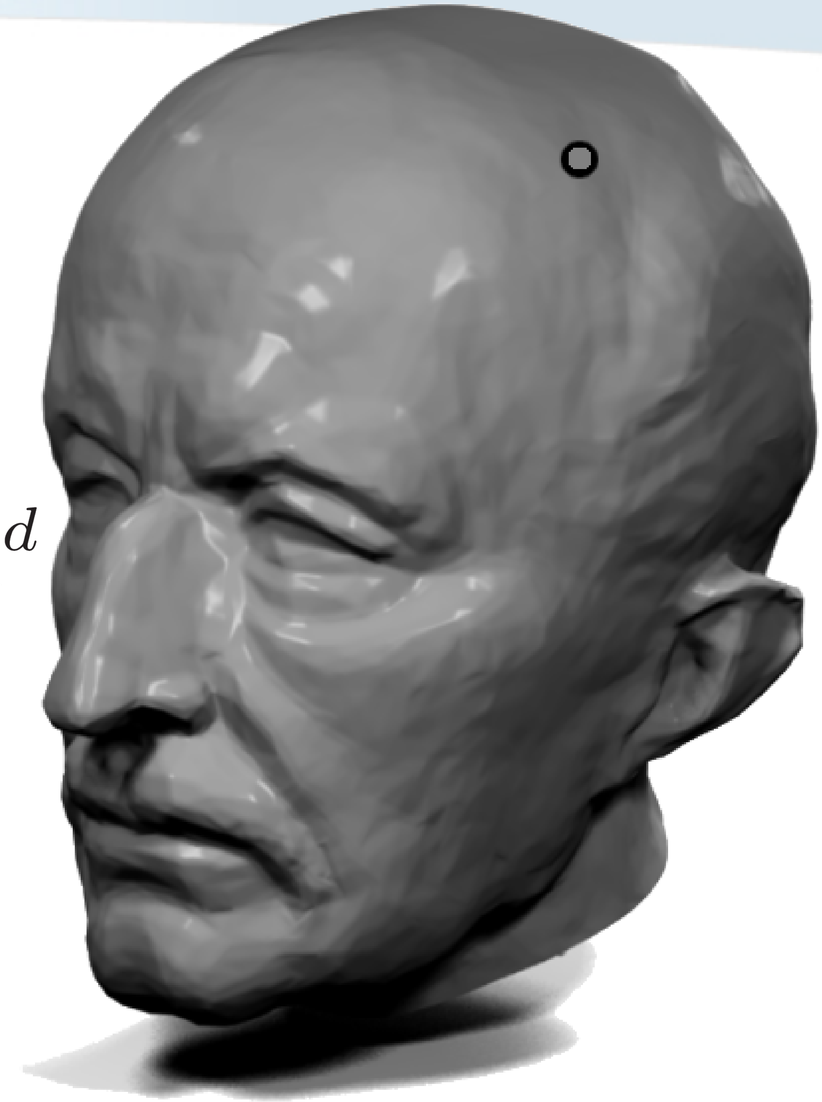
Domains of interest: Manifolds

- d-manifold
  - Topological space such that:
    - Any point has a neighborhood homeomorphic to a neighborhood of $\mathbb{R}^d$



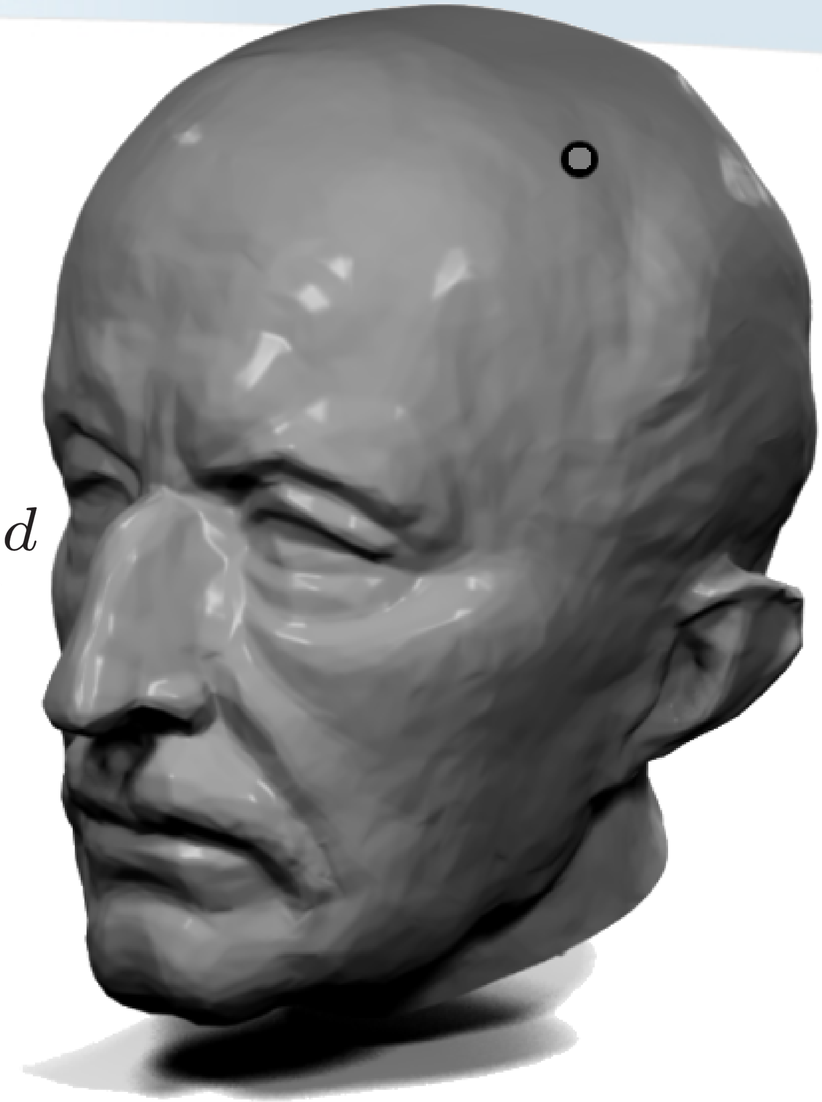
Domains of interest: Manifolds

- d-manifold
  - Topological space such that:
    - Any point has a neighborhood homeomorphic to a neighborhood of $\mathbb{R}^d$



Domains of interest: Manifolds

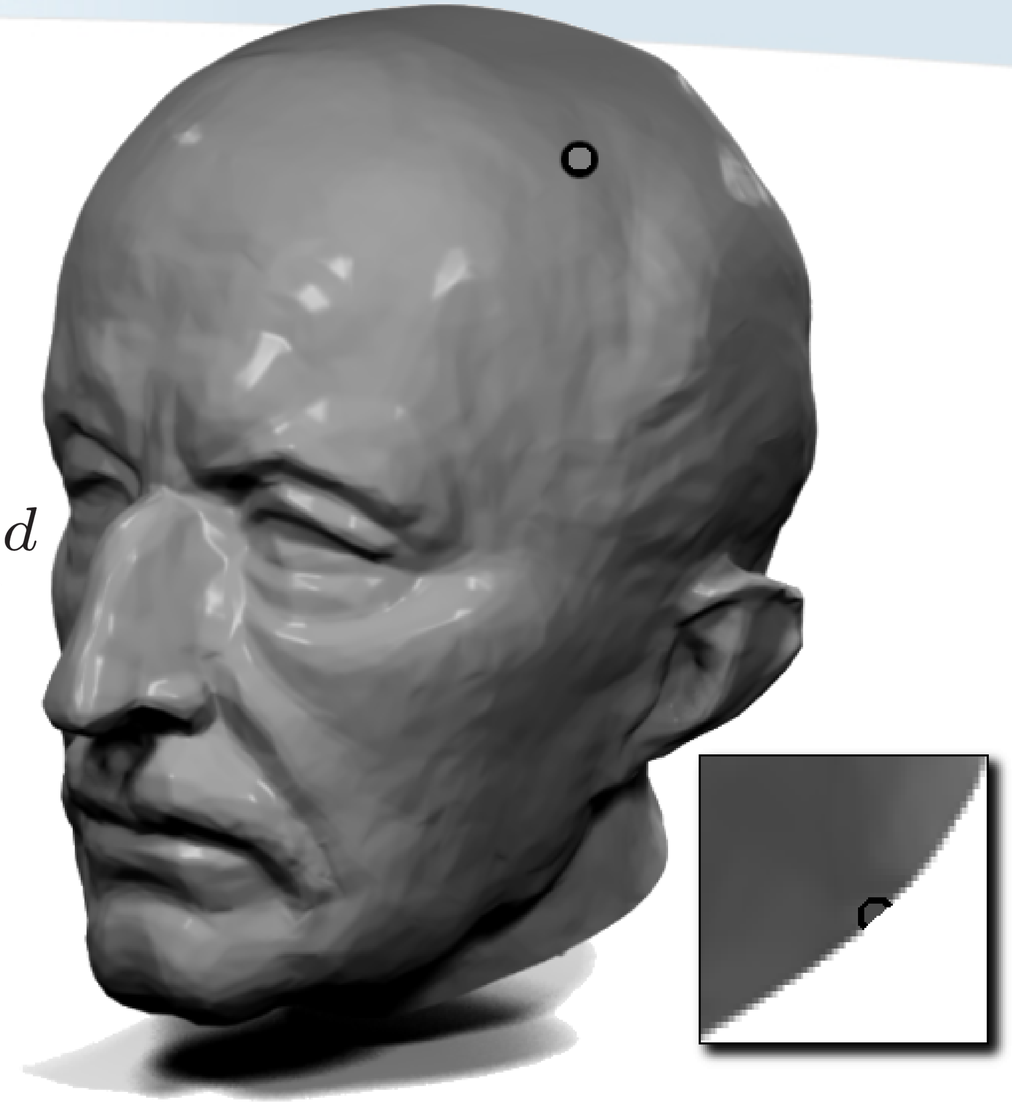
- d-manifold with boundary
  - Topological space such that:
    - Any point has a neighborhood homeomorphic to a neighborhood of $\mathbb{R}^d$ *or* its half space



Domains of interest: Manifolds

- d-manifold with boundary
  - Topological space such that:
    - Any point has a neighborhood homeomorphic to a neighborhood of $\mathbb{R}^d$

      or its half space

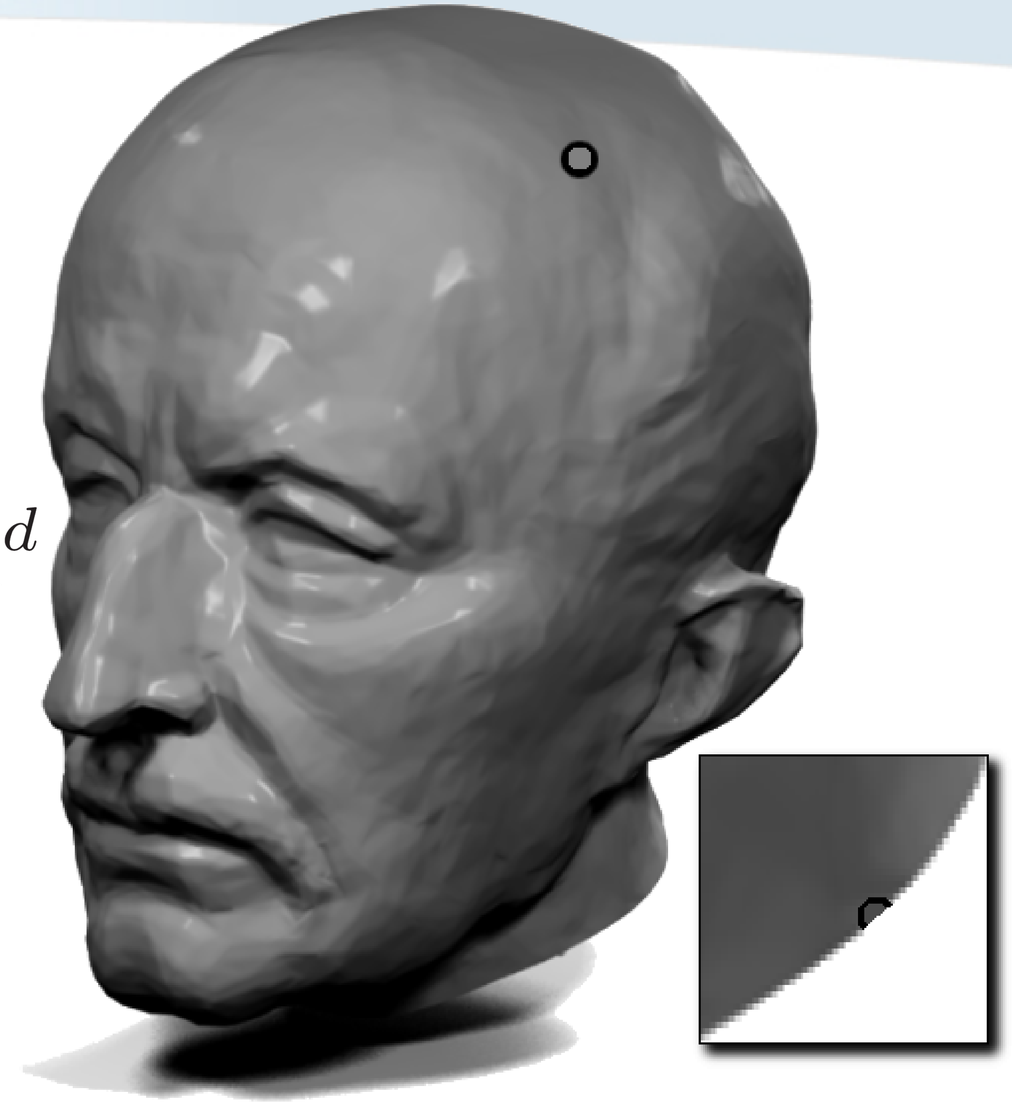


Domains of interest: Manifolds

- d-manifold with boundary
  - Topological space such that:
    - Any point has a neighborhood homeomorphic to a neighborhood of $\mathbb{R}^d$

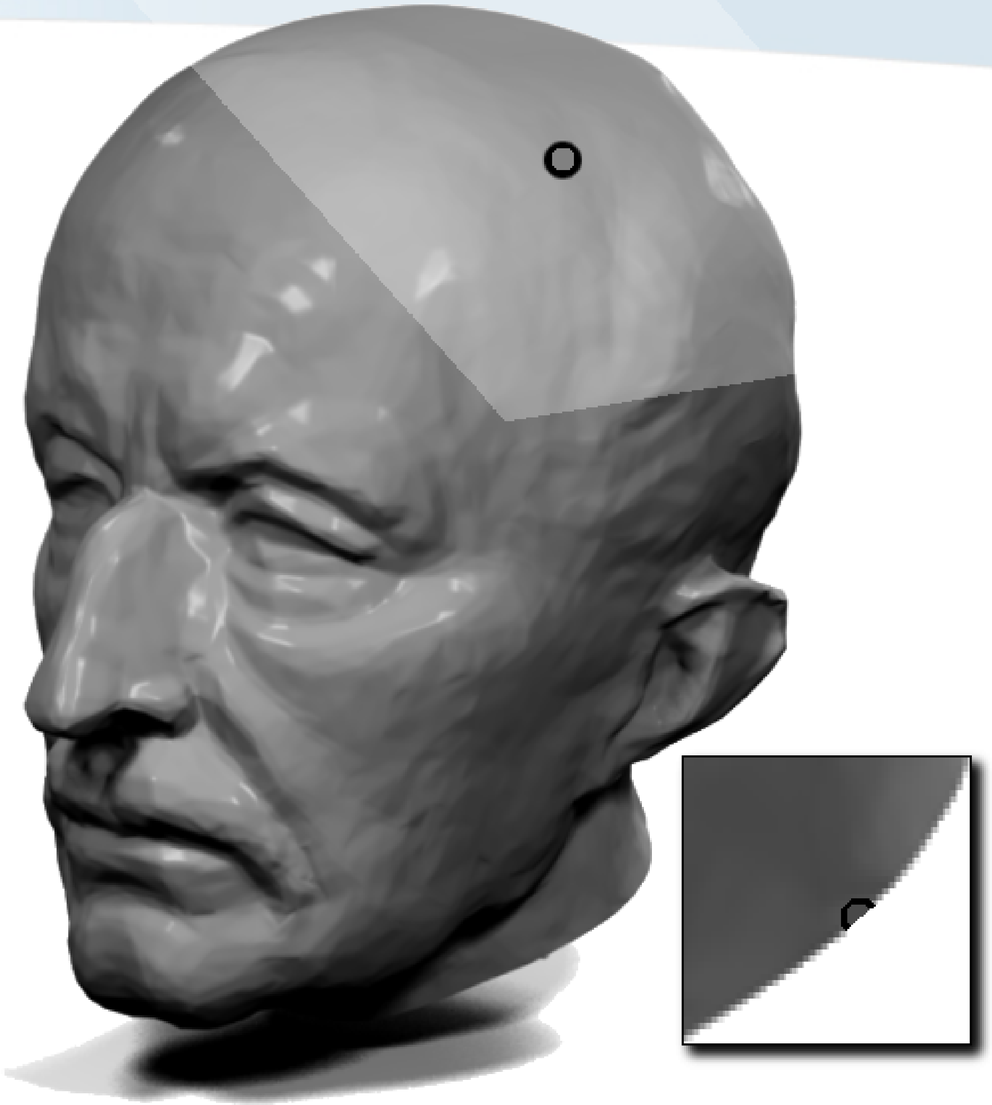
      or its half space

- Boundary: closed (d-1)-manifold



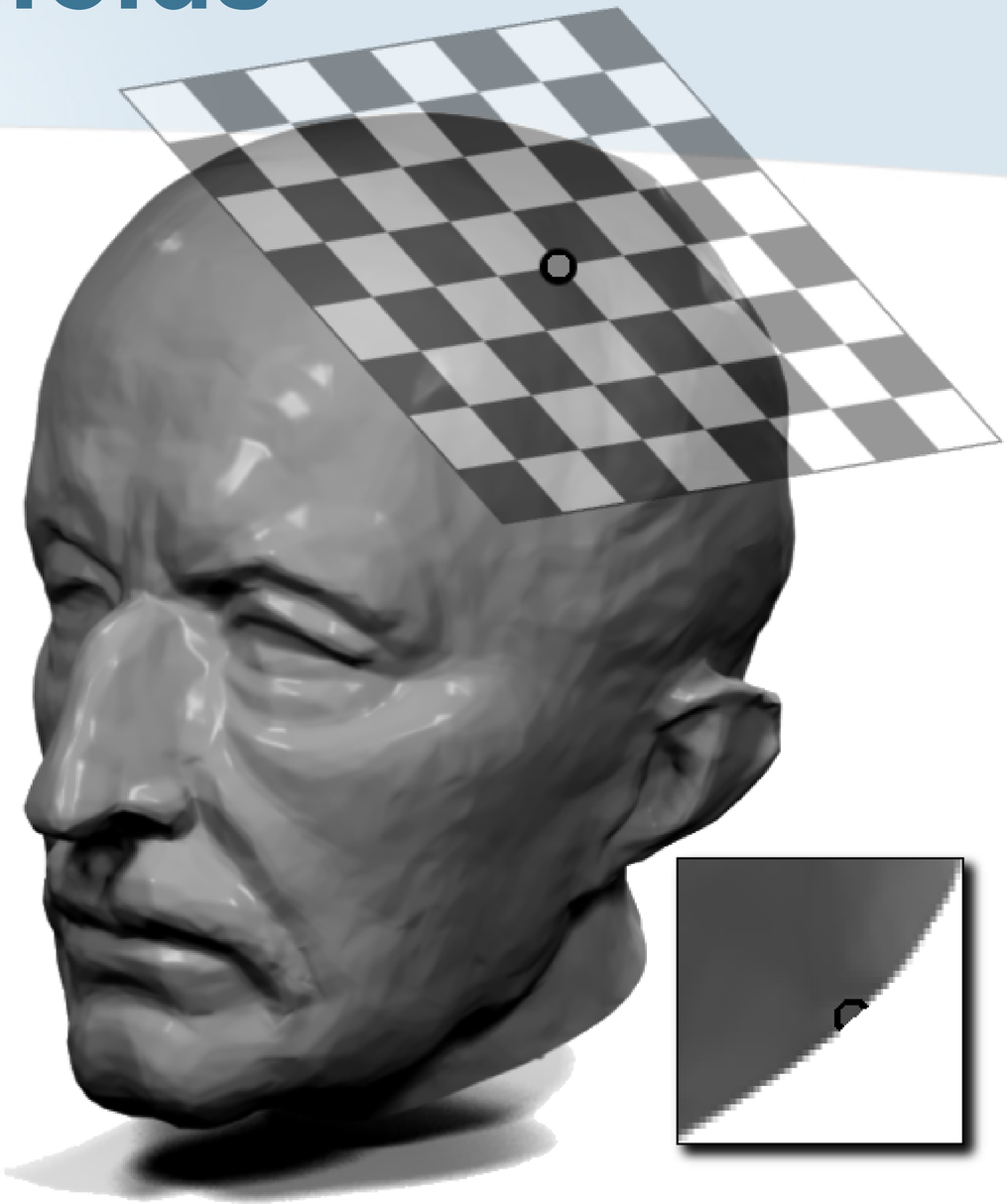
Domains of interest: Manifolds

- Tangent space
  - Localized Euclidean approximation



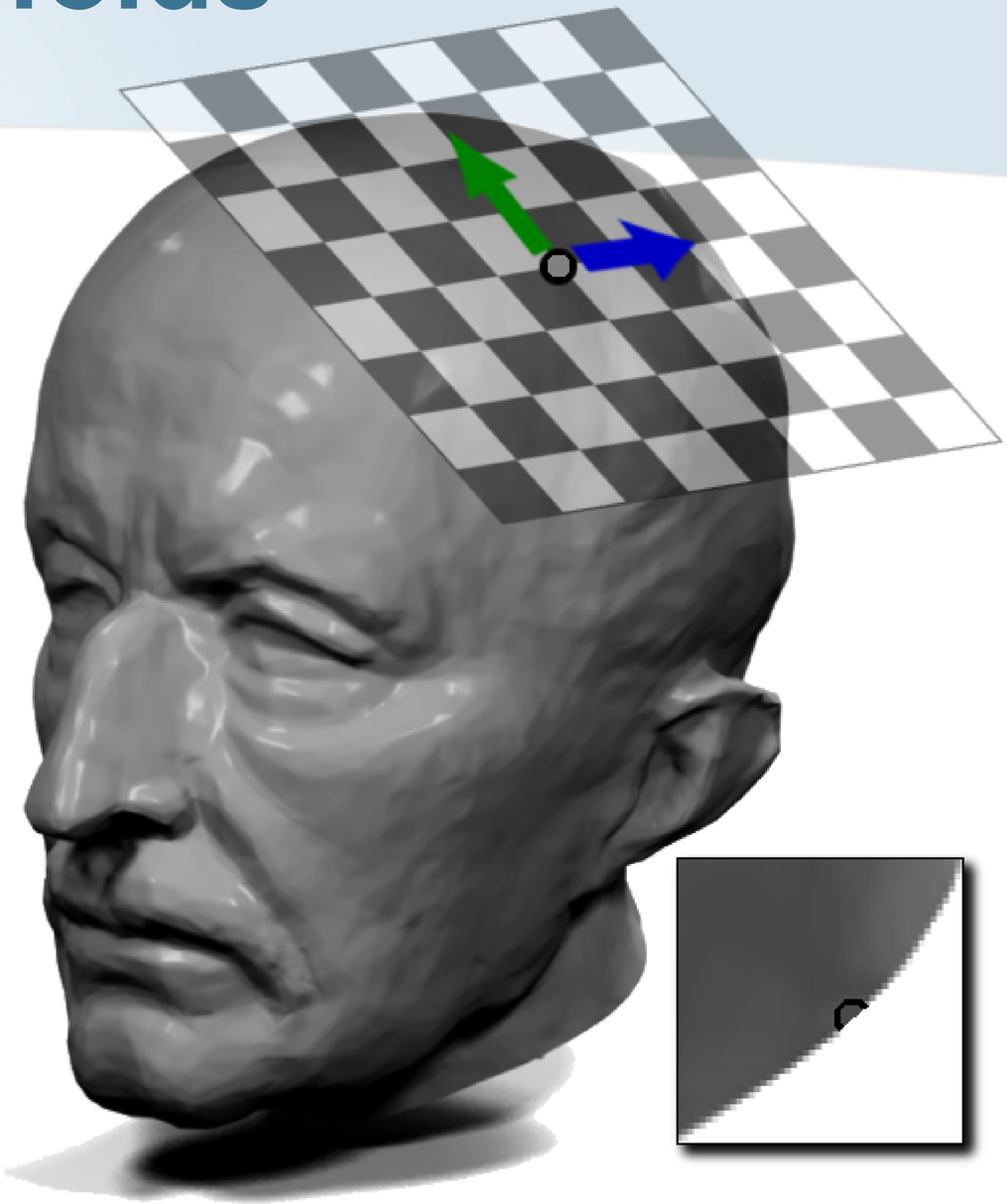
Domains of interest: Manifolds

- Tangent space
  - Localized Euclidean approximation
    - Localized Euclidean geometry



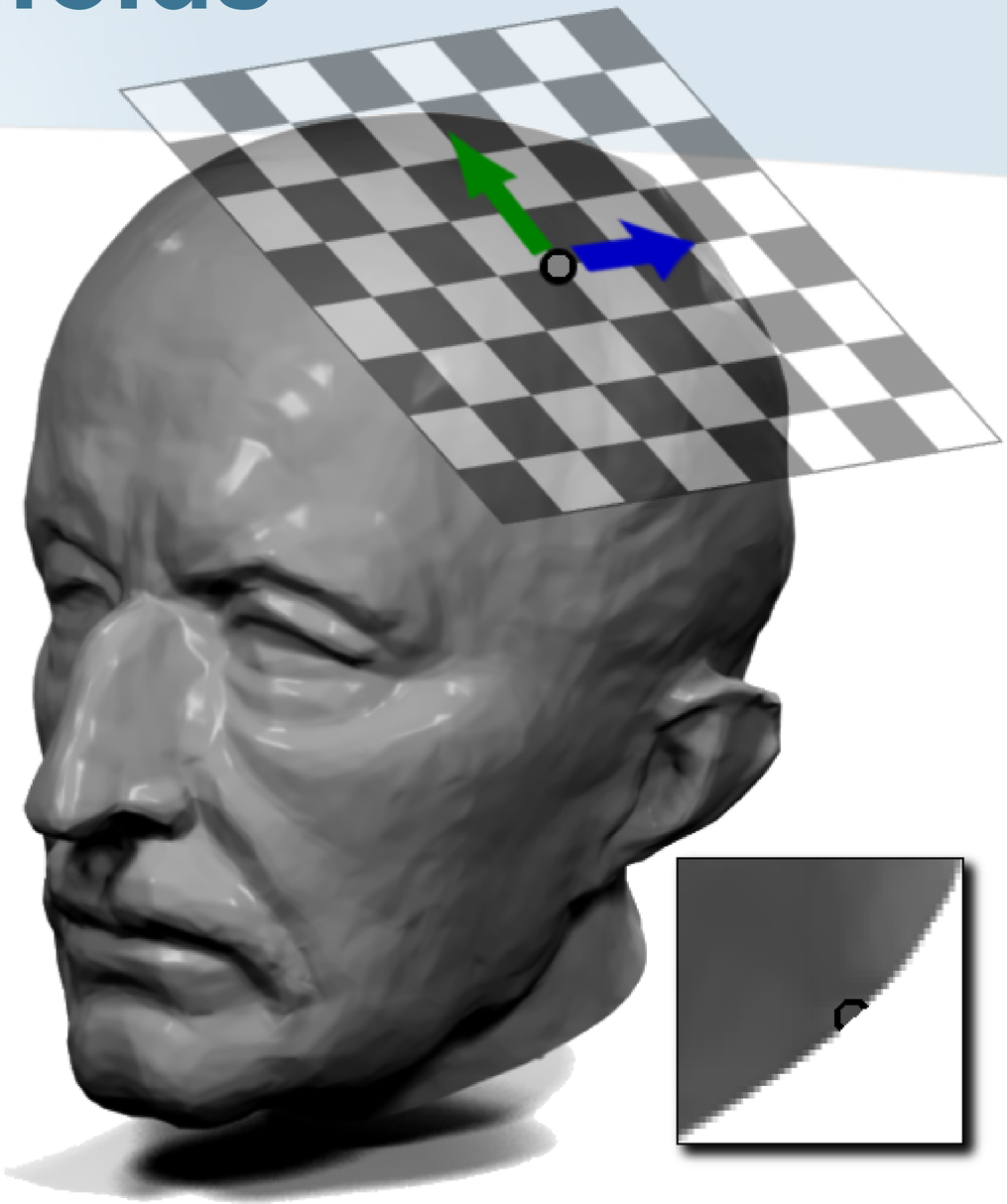
Domains of interest: Manifolds

- Tangent space
  - Localized Euclidean approximation
    - Localized Euclidean geometry
      - Inner products



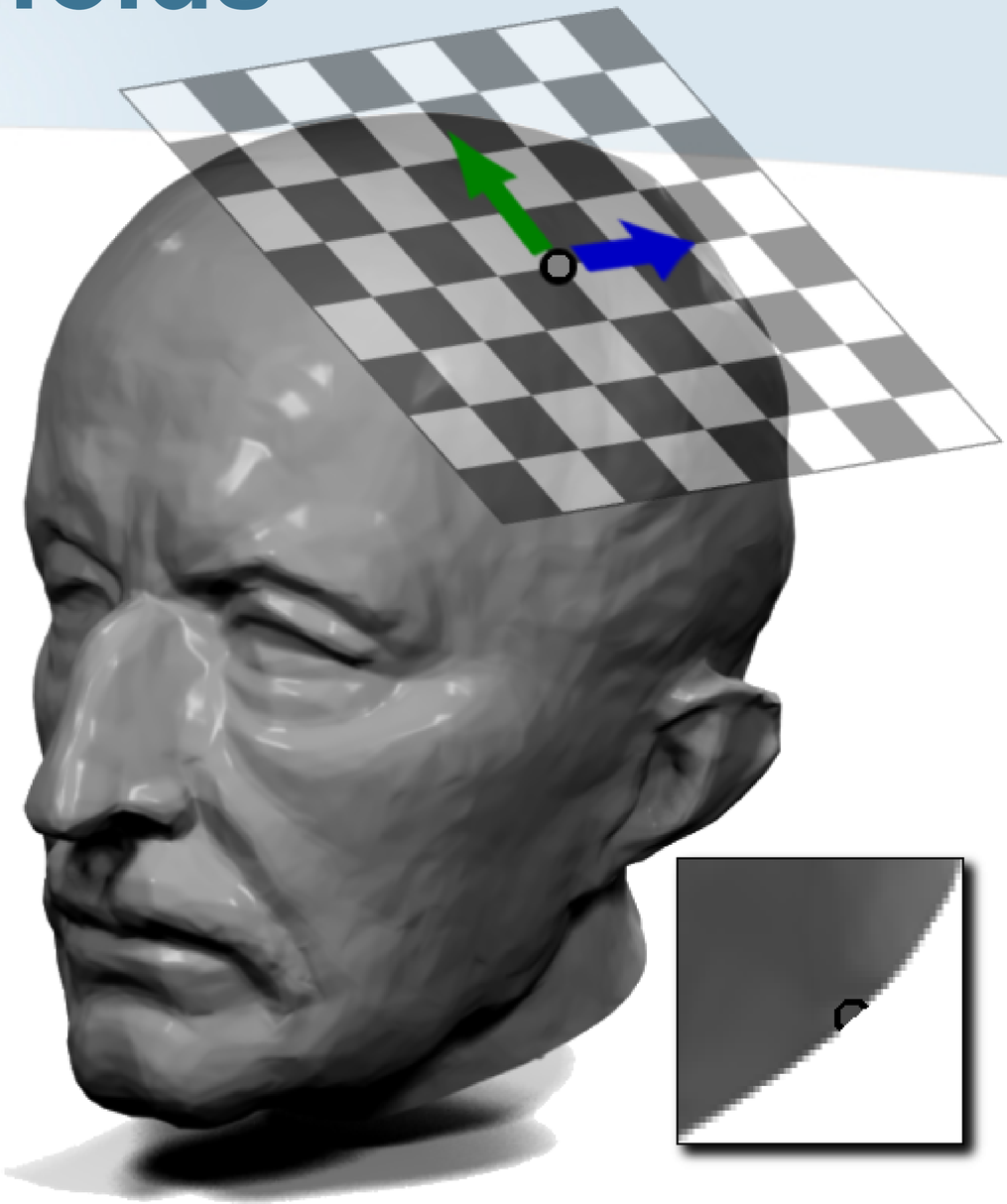
Domains of interest: Manifolds

- Tangent space
  - Localized Euclidean approximation
    - Localized Euclidean geometry
      - Inner products
      - Distances, angles, areas, …



Domains of interest: Manifolds

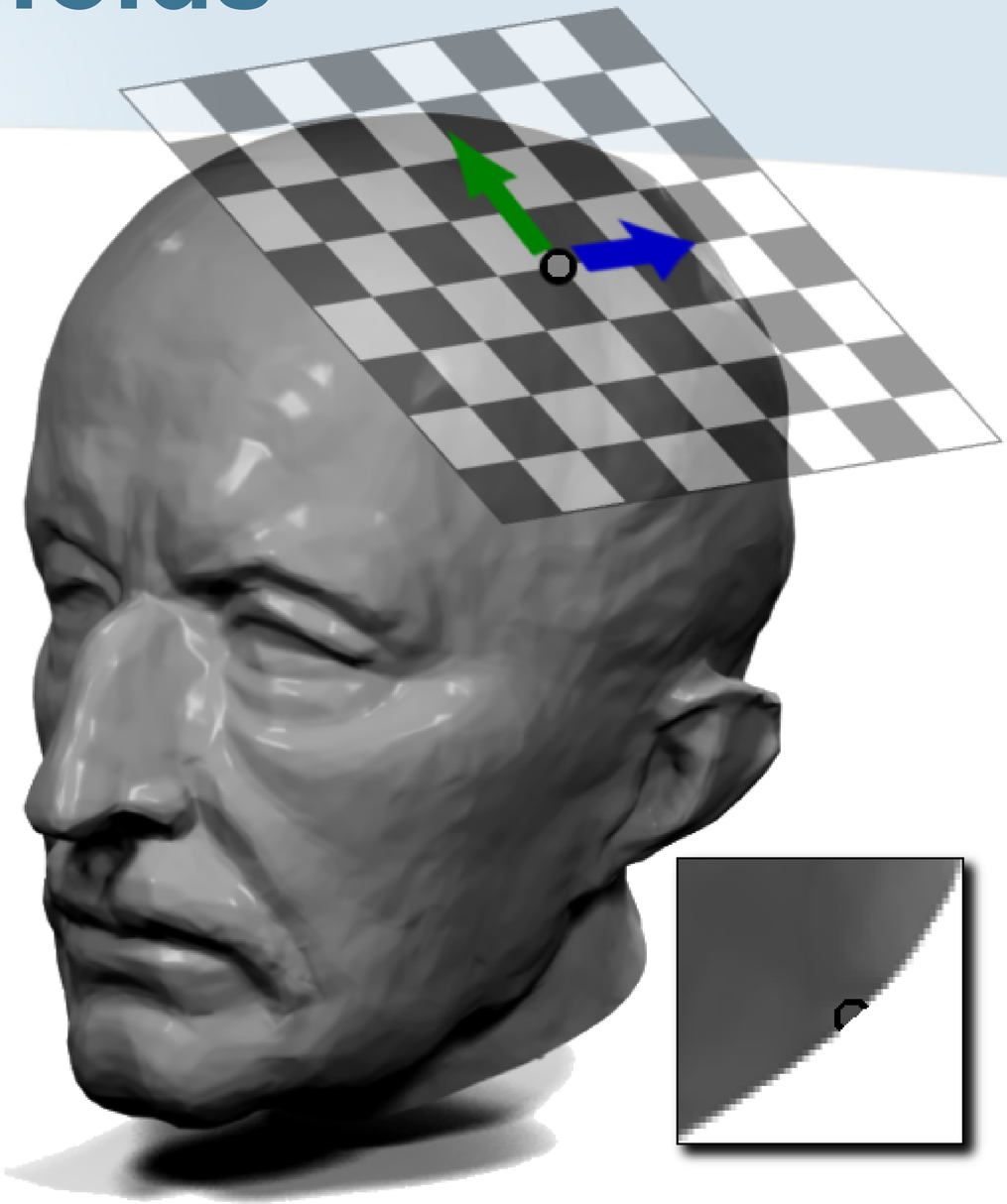
- Tangent space
  - Localized Euclidean approximation
    - Localized Euclidean geometry
      - Inner products
      - Distances, angles, areas, …
      - Gradient, Laplace operator, etc.



Domains of interest: Manifolds

- Tangent space
  - Localized Euclidean approximation
    - Localized Euclidean geometry
      - Inner products
      - Distances, angles, areas, …
      - Gradient, Laplace operator, etc.

- Discrete surface in $\mathbb{R}^3$
  - Triangular (planar) cells



Manifold examples

Manifold examples

- Dimension?
- Embedding space?
- Boundary?

Manifold examples

$\mathbb{R}$

- Dimension?
- Embedding space?
- Boundary?

Manifold examples

$\mathbb{R}$

$[0, 1]$

- Dimension?
- Embedding space?
- Boundary?

Manifold examples

- Dimension?
- Embedding space?
- Boundary?

$$\mathbb{R}$$

$$[0, 1]$$

$$\{p \in \mathbb{R}^2 \ / \ \sqrt{p_x^2 + p_y^2} = r\}$$

Manifold examples

$$\mathbb{R}^2$$

$$[0, 1]$$

$$\{p \in \mathbb{R}^2 \ / \ \sqrt{p_x^2 + p_y^2} = r\}$$

- Dimension?
- Embedding space?
- Boundary?

Manifold examples

$$\mathbb{R}^2$$

$$[0,1] \times [0,1]$$

$$\{p \in \mathbb{R}^2 \ / \ \sqrt{p_x^2 + p_y^2} = r\}$$

- Dimension?
- Embedding space?
- Boundary?

Manifold examples

$$\mathbb{R}^2$$

$$[0,1] \times [0,1]$$

$$\{p \in \mathbb{R}^2 \ / \ \sqrt{p_x^2 + p_y^2} \leq r\}$$

- Dimension?
- Embedding space?
- Boundary?

Manifold examples

$$\mathbb{R}^3$$

$$[0,1] \times [0,1]$$

$$\{p \in \mathbb{R}^2 \ / \ \sqrt{p_x^2 + p_y^2} \leq r\}$$

- Dimension?
- Embedding space?
- Boundary?

Manifold examples

- Dimension?
- Embedding space?
- Boundary?

$$\mathbb{R}^3$$

$$[0,1] \times [0,1] \times [0,1]$$

$$\{p \in \mathbb{R}^2 \ / \ \sqrt{p_x^2 + p_y^2} \leq r\}$$

Manifold examples

$$\mathbb{R}^3$$

- Dimension?
- Embedding space?
- Boundary?

$$[0,1] \times [0,1] \times [0,1]$$

$$\{p \in \mathbb{R}^3 \ / \ \sqrt{p_x^2 + p_y^2 + p_z^2} = r\}$$

Manifold examples

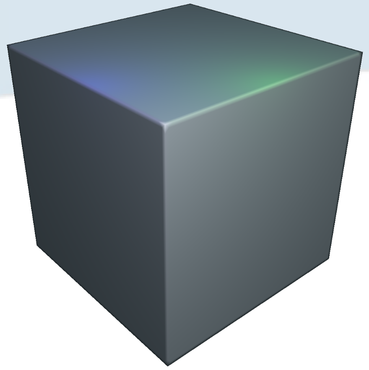
$$\mathbb{R}^3$$

$$[0,1] \times [0,1] \times [0,1]$$

$$\{p \in \mathbb{R}^3 \ / \ \sqrt{p_x^2 + p_y^2 + p_z^2} \leq r\}$$

- Dimension?
- Embedding space?
- Boundary?

Manifold examples



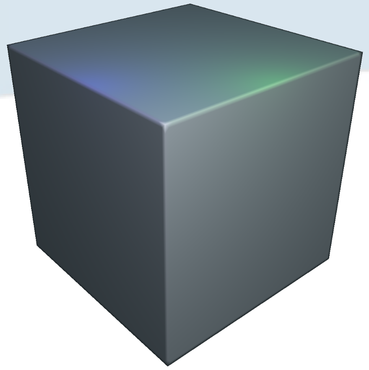
$$\mathbb{R}^3$$

$$[0,1] \times [0,1] \times [0,1]$$

$$\{p \in \mathbb{R}^3 \ / \ \sqrt{p_x^2 + p_y^2 + p_z^2} \leq r\}$$

- Dimension?
- Embedding space?
- Boundary?

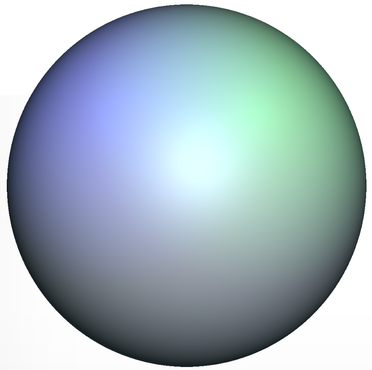
Manifold examples



$$\mathbb{R}^3$$

- Dimension?
- Embedding space?
- Boundary?

$$[0,1] \times [0,1] \times [0,1]$$



$$\{p \in \mathbb{R}^3 \ / \ \sqrt{p_x^2 + p_y^2 + p_z^2} \leq r\}$$

Summary

Summary

- Euclidean representations

Summary

- Euclidean representations
- Non-euclidean representations

Summary

- Euclidean representations
- Non-euclidean representations

- Dimension-specific data-structures

Summary

- Euclidean representations
- Non-euclidean representations

- Dimension-specific data-structures
- Dimension-specific interpolants

Summary

- Euclidean representations
- Non-euclidean representations

- Dimension-specific data-structures
- Dimension-specific interpolants
- Implementation examples

Euclidean spaces

Euclidean spaces

- Named after Greek mathematician Euclid
  - 325 BC, 265 BC

Euclidean spaces

- Named after Greek mathematician Euclid
  - 325 BC, 265 BC

Euclidean spaces

- Named after Greek mathematician Euclid
  - 325 BC, 265 BC
- *“Elements” (300 BC)*

Euclidean spaces

- Named after Greek mathematician Euclid
  - 325 BC, 265 BC
- *“Elements” (300 BC)*
  - First *systematic* math book

Euclidean spaces

- Named after Greek mathematician Euclid
  - 325 BC, 265 BC

- *“Elements” (300 BC)*
  - First *systematic* math book
    - Definitions, axioms, demonstrations

Euclidean spaces

- Named after Greek mathematician Euclid
  - 325 BC, 265 BC

- *“Elements” (300 BC)*
  - First *systematic* math book
    - Definitions, axioms, demonstrations
    - 5 axioms (first tome)

Euclidean spaces on a computer

Euclidean spaces on a computer

- Given an origin

Euclidean spaces on a computer

- Given an origin
  - $o = (0, 0, \ldots, 0) \in \mathbb{R}^n$

Euclidean spaces on a computer

- Given an origin
  - $o = (0, 0, \ldots, 0) \in \mathbb{R}^n$
- And an orthonormal basis

Euclidean spaces on a computer

- Given an origin
  - $o = (0, 0, \ldots, 0) \in \mathbb{R}^n$
- And an orthonormal basis
  - $v_1 = (1, 0, 0, \ldots, 0) \in \mathbb{R}^n$

Euclidean spaces on a computer

- Given an origin
  - $o = (0, 0, \ldots, 0) \in \mathbb{R}^n$
- And an orthonormal basis
  - $v_1 = (1, 0, 0, \ldots, 0) \in \mathbb{R}^n$
  - $v_2 = (0, 1, 0, \ldots, 0) \in \mathbb{R}^n$

Euclidean spaces on a computer

- Given an origin
  - $o = (0, 0, \ldots, 0) \in \mathbb{R}^n$
- And an orthonormal basis
  - $v_1 = (1, 0, 0, \ldots, 0) \in \mathbb{R}^n$
  - $v_2 = (0, 1, 0, \ldots, 0) \in \mathbb{R}^n$
  - $v_3 = (0, 0, 1, \ldots, 0) \in \mathbb{R}^n$

Euclidean spaces on a computer

- Given an origin
  - $o = (0, 0, \ldots, 0) \in \mathbb{R}^n$
- And an orthonormal basis
  - $v_1 = (1, 0, 0, \ldots, 0) \in \mathbb{R}^n$
  - $v_2 = (0, 1, 0, \ldots, 0) \in \mathbb{R}^n$
  - $v_3 = (0, 0, 1, \ldots, 0) \in \mathbb{R}^n$
  - $\ldots$
  - $v_n = (0, 0, 0, \ldots, 1) \in \mathbb{R}^n$

Euclidean spaces on a computer

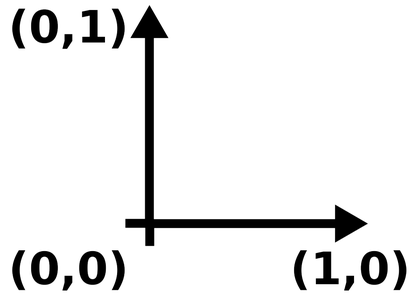
- Given an origin
  - $o = (0, 0, \ldots, 0) \in \mathbb{R}^n$
- And an orthonormal basis
  - $v_1 = (1, 0, 0, \ldots, 0) \in \mathbb{R}^n$
  - $v_2 = (0, 1, 0, \ldots, 0) \in \mathbb{R}^n$
  - $v_3 = (0, 0, 1, \ldots, 0) \in \mathbb{R}^n$
  - $\ldots$
  - $v_n = (0, 0, 0, \ldots, 1) \in \mathbb{R}^n$
- Points can be identified uniquely in that space

Euclidean spaces on a computer

- Given an origin
  - $o = (0, 0, \ldots, 0) \in \mathbb{R}^n$
- And an orthonormal basis
  - $v_1 = (1, 0, 0, \ldots, 0) \in \mathbb{R}^n$
  - $v_2 = (0, 1, 0, \ldots, 0) \in \mathbb{R}^n$
  - $v_3 = (0, 0, 1, \ldots, 0) \in \mathbb{R}^n$
  - $\ldots$
  - $v_n = (0, 0, 0, \ldots, 1) \in \mathbb{R}^n$
- Points can be identified uniquely in that space

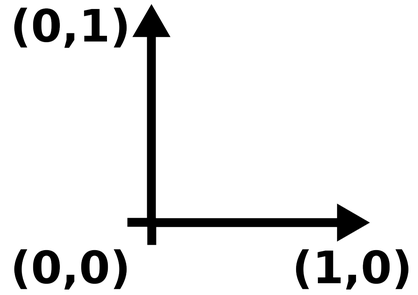
Euclidean spaces on a computer

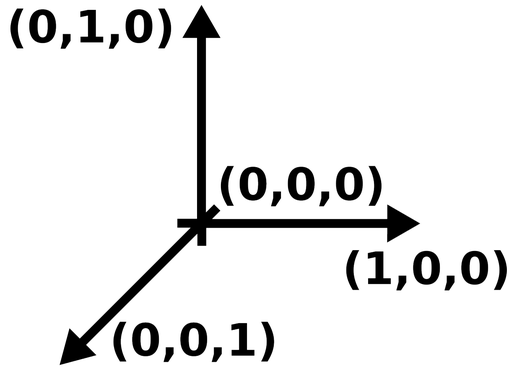
- Given an origin
  - $o = (0, 0, \ldots, 0) \in \mathbb{R}^n$
- And an orthonormal basis
  - $v_1 = (1, 0, 0, \ldots, 0) \in \mathbb{R}^n$
  - $v_2 = (0, 1, 0, \ldots, 0) \in \mathbb{R}^n$
  - $v_3 = (0, 0, 1, \ldots, 0) \in \mathbb{R}^n$
  - $\ldots$
  - $v_n = (0, 0, 0, \ldots, 1) \in \mathbb{R}^n$
- Points can be identified uniquely in that space



Euclidean spaces on a computer

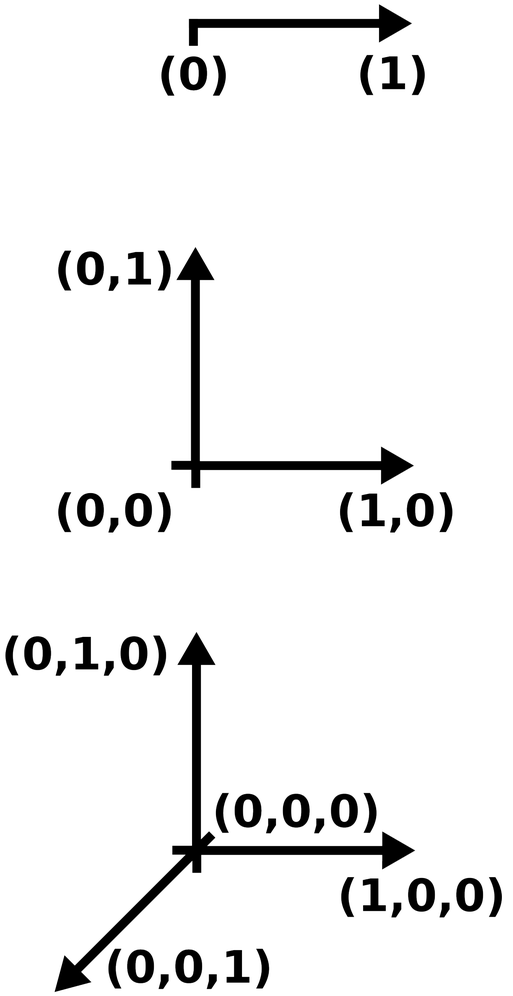
- Given an origin
  - $o = (0, 0, \ldots, 0) \in \mathbb{R}^n$
- And an orthonormal basis
  - $v_1 = (1, 0, 0, \ldots, 0) \in \mathbb{R}^n$
  - $v_2 = (0, 1, 0, \ldots, 0) \in \mathbb{R}^n$
  - $v_3 = (0, 0, 1, \ldots, 0) \in \mathbb{R}^n$
  - $\ldots$
  - $v_n = (0, 0, 0, \ldots, 1) \in \mathbb{R}^n$
- Points can be identified uniquely in that space





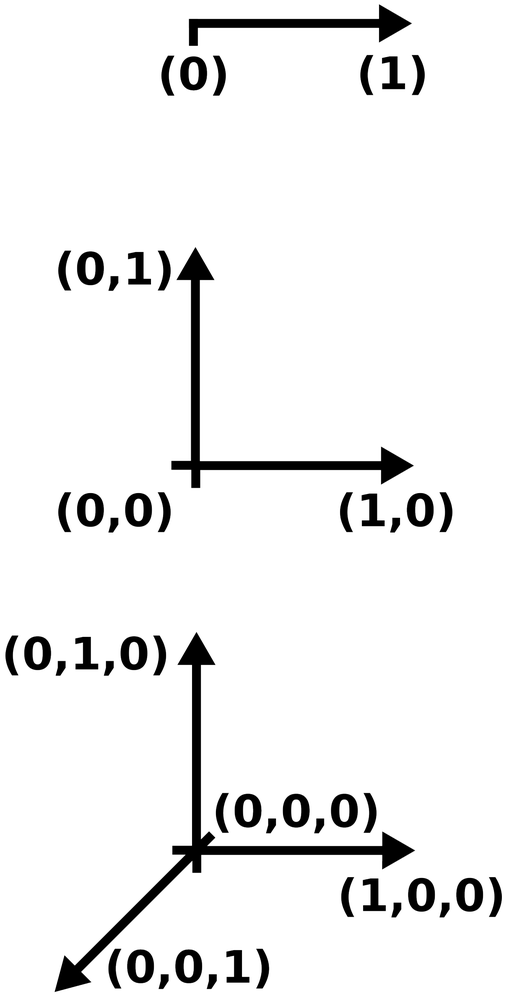
Euclidean spaces on a computer

- Direct product of closed unit intervals



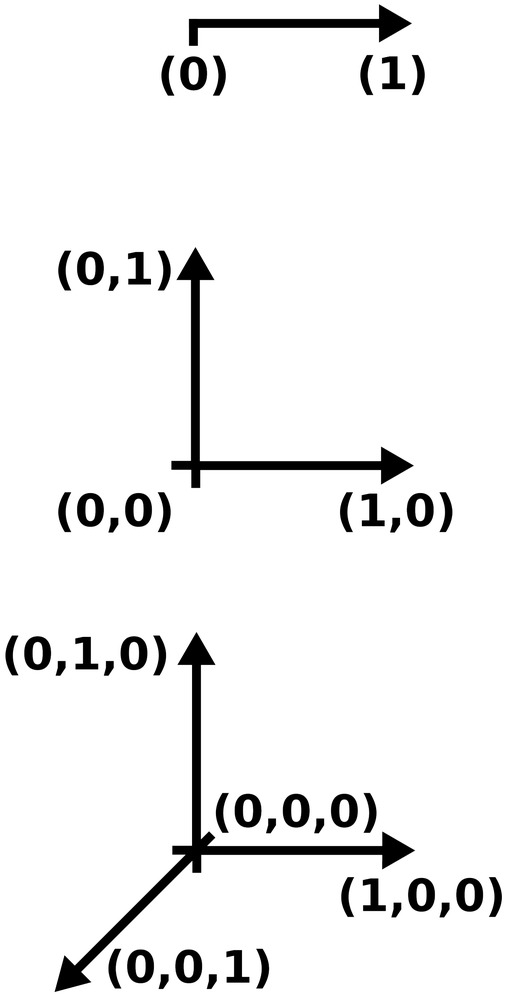
Euclidean spaces on a computer

- Direct product of closed unit intervals
  - $[0,1] \times [0,1] \times \cdots \times [0,1]$



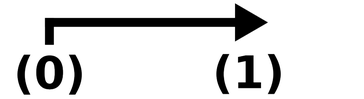
Euclidean spaces on a computer

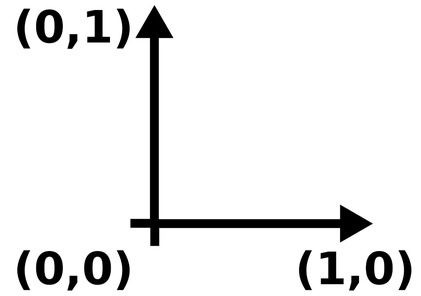
- Direct product of closed unit intervals
  - $[0,1] \times [0,1] \times \cdots \times [0,1]$
  - By construction

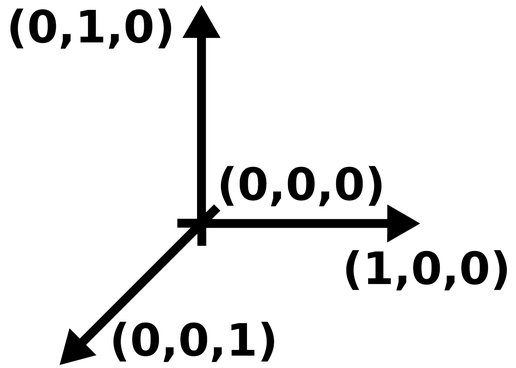


Euclidean spaces on a computer

- Direct product of closed unit intervals
  - $[0,1] \times [0,1] \times \cdots \times [0,1]$
  - By construction
    - Unit translations along the vectors of the orthonormal basis

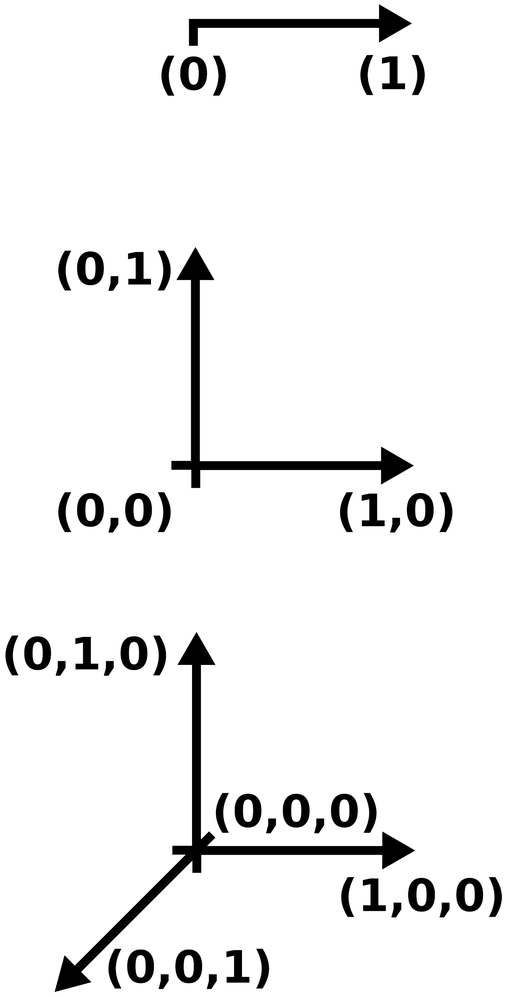






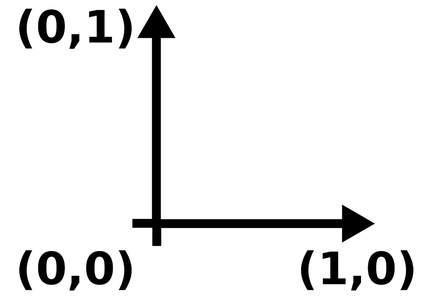
Euclidean spaces on a computer

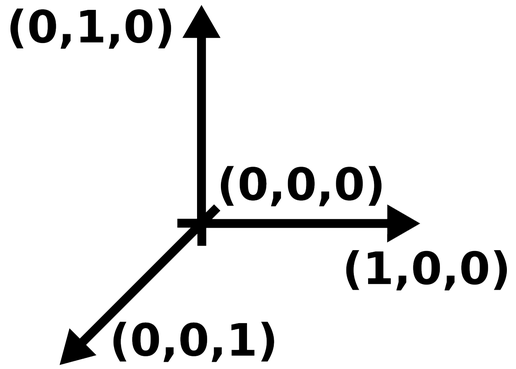
- Direct product of closed unit intervals
  - $[0,1] \times [0,1] \times \cdots \times [0,1]$
  - By construction
    - Unit translations along the vectors of the orthonormal basis
    - Covering a bounded, compact region of $\mathbb{R}^n$



Euclidean spaces on a computer

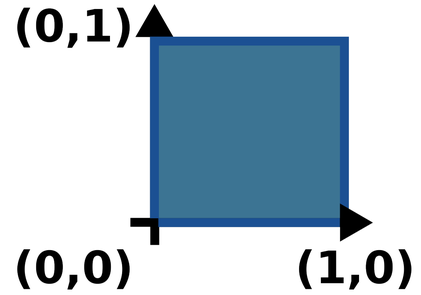
- Direct product of closed unit intervals
  - $[0,1] \times [0,1] \times \cdots \times [0,1]$
  - By construction
    - Unit translations along the vectors of the orthonormal basis
    - Covering a bounded, compact region of $\mathbb{R}^n$

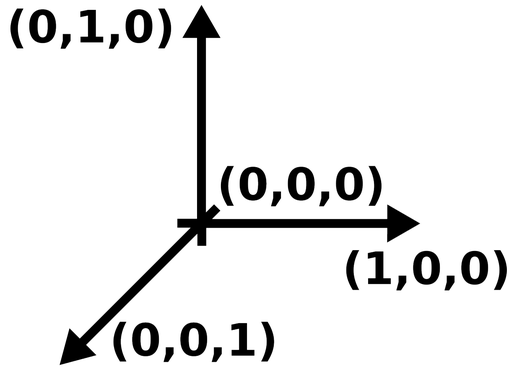




Euclidean spaces on a computer

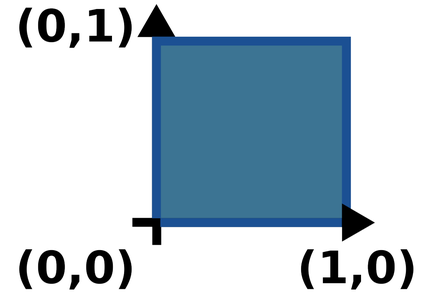
- Direct product of closed unit intervals
  - $[0,1] \times [0,1] \times \cdots \times [0,1]$
  - By construction
    - Unit translations along the vectors of the orthonormal basis
    - Covering a bounded, compact region of $\mathbb{R}^n$

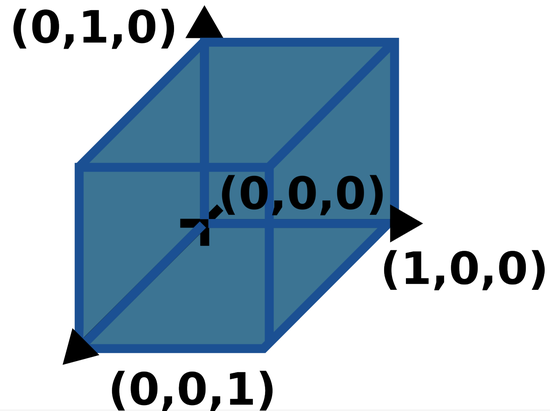




Euclidean spaces on a computer

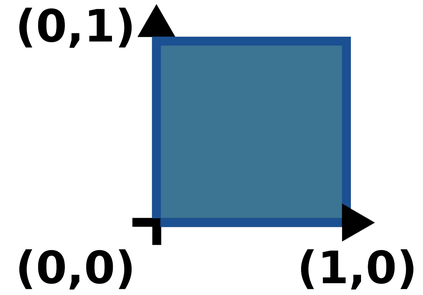
- Direct product of closed unit intervals
  - $[0,1] \times [0,1] \times \cdots \times [0,1]$
  - By construction
    - Unit translations along the vectors of the orthonormal basis
    - Covering a bounded, compact region of $\mathbb{R}^n$

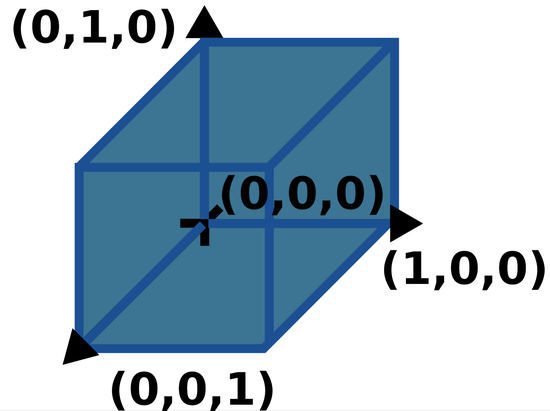




Euclidean spaces on a computer

- Direct product of closed unit intervals
  - $[0,1] \times [0,1] \times \cdots \times [0,1]$
  - By construction
    - Unit translations along the vectors of the orthonormal basis
    - Covering a bounded, compact region of $\mathbb{R}^n$

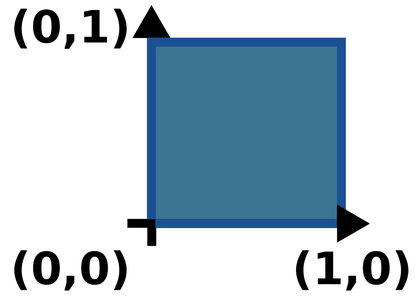


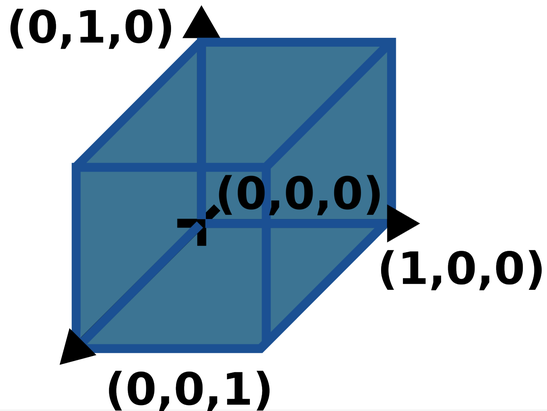


- **Unit cell**

Euclidean spaces on a computer

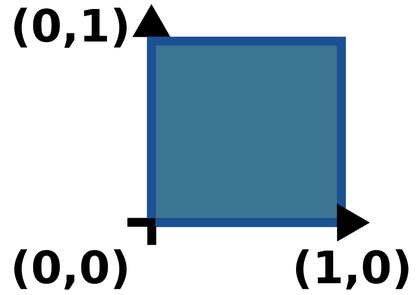
- Given a unit cell of $\mathbb{R}^n$

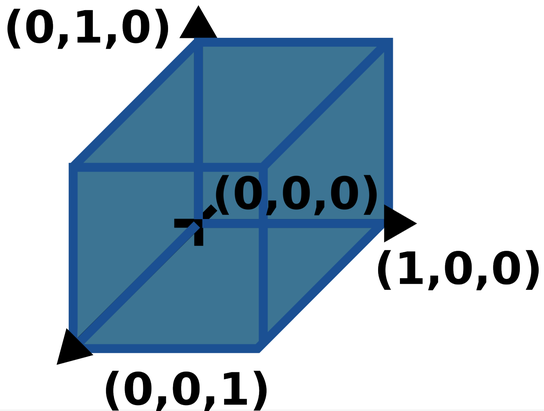




Euclidean spaces on a computer

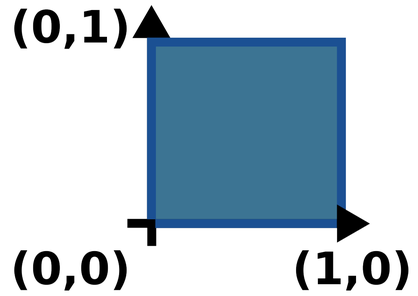
- Given a unit cell of $\mathbb{R}^n$
  - What's the minimum number of samples necessary to describe its convex hull?

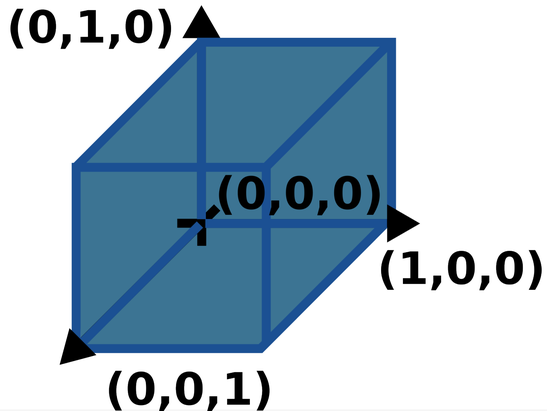




Euclidean spaces on a computer

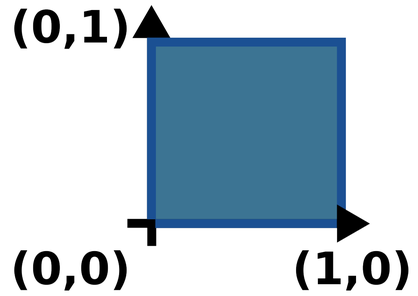
- Given a unit cell of $\mathbb{R}^n$
  - What's the minimum number of samples necessary to describe its convex hull?
  - 2^n

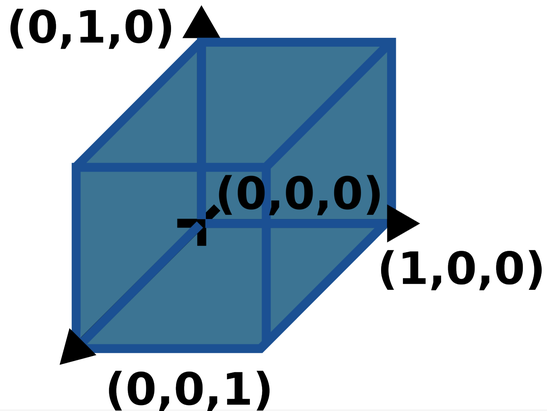




Euclidean spaces on a computer

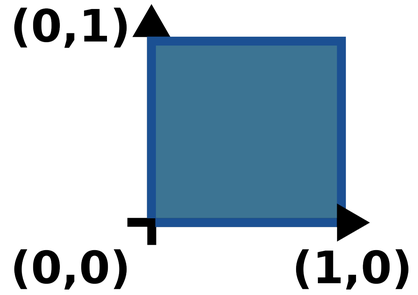
- Given a unit cell of $\mathbb{R}^n$
  - What's the minimum number of samples necessary to describe its convex hull?
  - 2^n
- Arbitrary closed and bounded subsets of $\mathbb{R}^n$

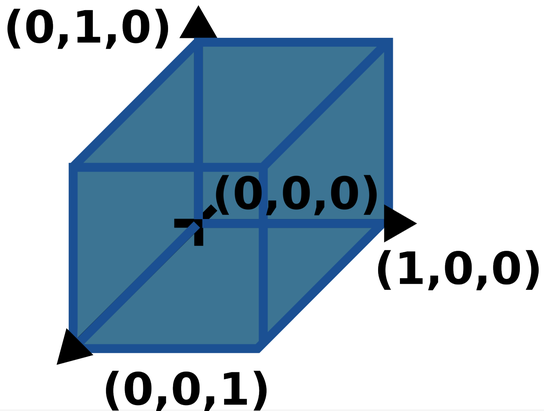




Euclidean spaces on a computer

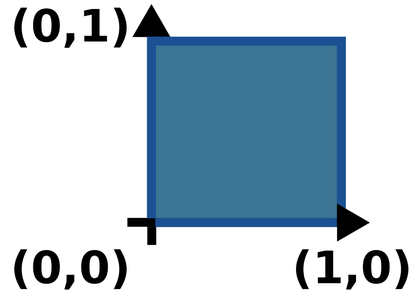
- Given a unit cell of $\mathbb{R}^n$
  - What's the minimum number of samples necessary to describe its convex hull?
  - 2^n
- Arbitrary closed and bounded subsets of $\mathbb{R}^n$
  - Collection of unit cells

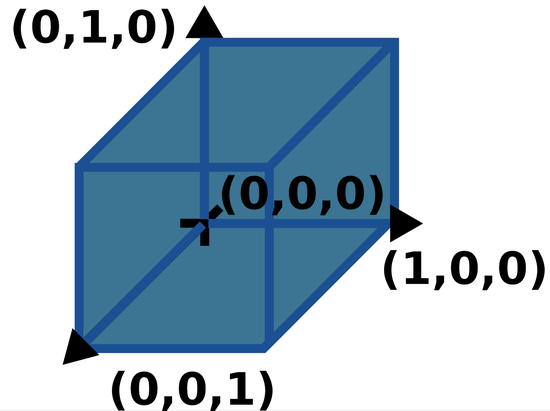




Euclidean spaces on a computer

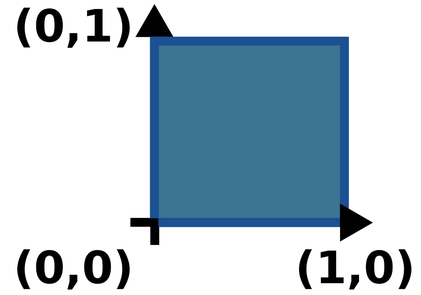
- Given a unit cell of $\mathbb{R}^n$
  - What's the minimum number of samples necessary to describe its convex hull?
  - 2^n
- Arbitrary closed and bounded subsets of $\mathbb{R}^n$
  - Collection of unit cells
  - By construction
    - Unit translations on the unit vectors of the orthonormal basis

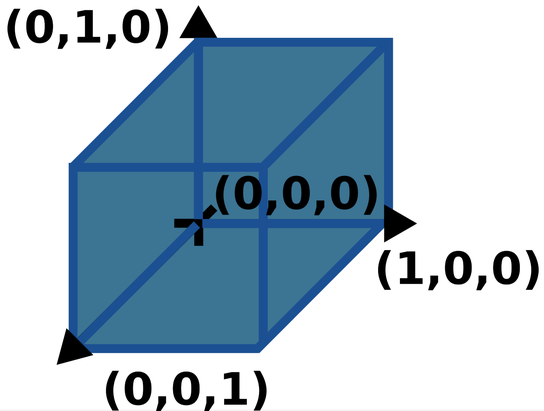




Euclidean spaces on a computer

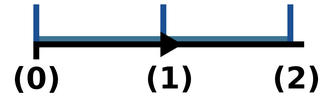
- Notion of **regular grid** $\mathcal{G}$
  - Finite collection of unit cells
  - Entirely covering the direct product of closed intervals, such that
    - $[0, k_1] \times [0, k_2] \times \cdots \times [0, k_n]$
    - $k_i \in \mathbb{N}$

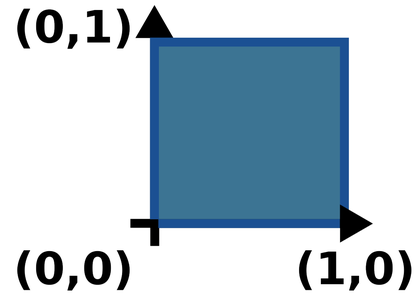


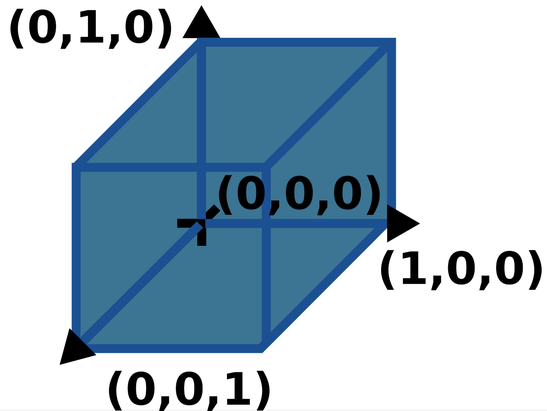


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Finite collection of unit cells
  - Entirely covering the direct product of closed intervals, such that
    - $[0, k_1] \times [0, k_2] \times \cdots \times [0, k_n]$
    - $k_i \in \mathbb{N}$

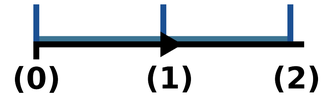


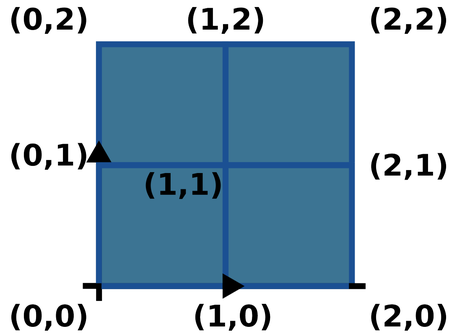


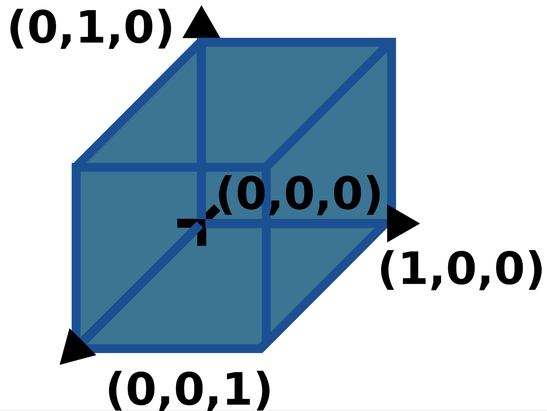


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Finite collection of unit cells
  - Entirely covering the direct product of closed intervals, such that
    - $[0, k_1] \times [0, k_2] \times \cdots \times [0, k_n]$
    - $k_i \in \mathbb{N}$

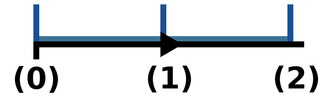


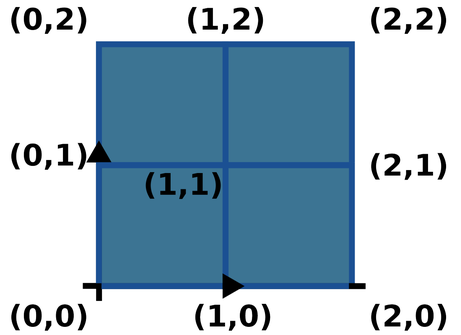


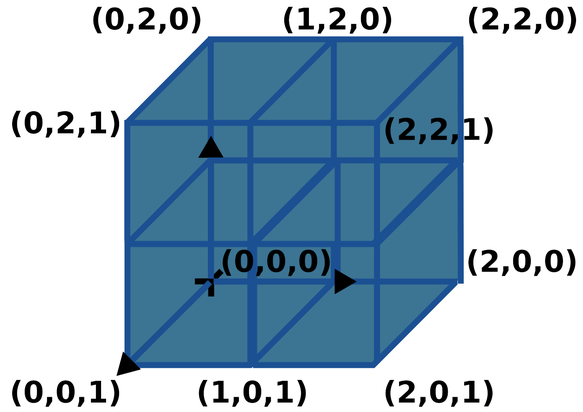


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Finite collection of unit cells
  - Entirely covering the direct product of closed intervals, such that
    - $[0, k_1] \times [0, k_2] \times \cdots \times [0, k_n]$
    - $k_i \in \mathbb{N}$

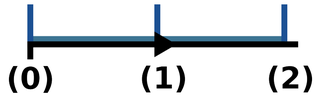


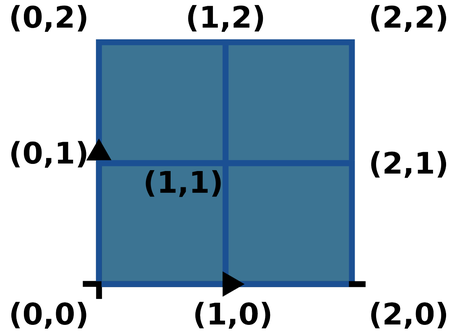


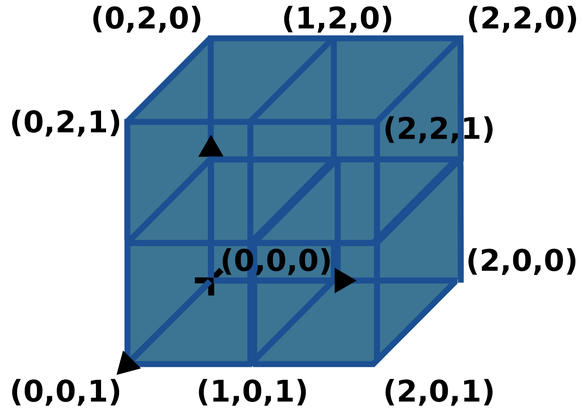


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Finite collection of unit cells
  - Entirely covering the direct product of closed intervals, such that
    - $[0, k_1] \times [0, k_2] \times \cdots \times [0, k_n]$
    - $k_i \in \mathbb{N}$
  - How many unit cells in $[0, k_1] \times [0, k_2] \times [0, k_3]$?

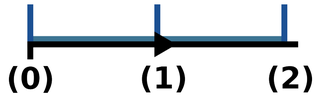


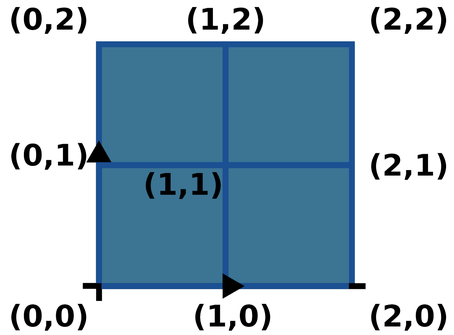


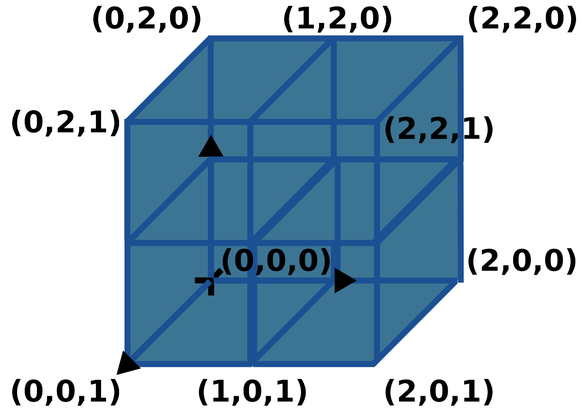


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Finite collection of unit cells
  - Entirely covering the direct product of closed intervals, such that
    - $[0, k_1] \times [0, k_2] \times \cdots \times [0, k_n]$
    - $k_i \in \mathbb{N}$
  - How many unit cells in $[0, k_1] \times [0, k_2] \times [0, k_3]$?
    - $k_1.k_2.k_3$

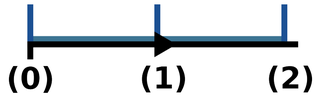


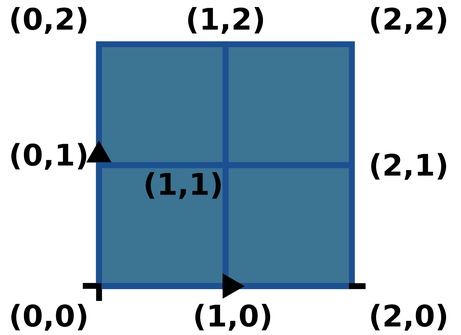


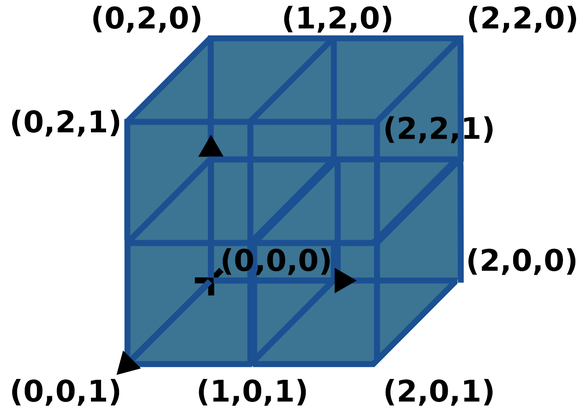


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Finite collection of unit cells
  - Entirely covering the direct product of closed intervals, such that
    - $[0, k_1] \times [0, k_2] \times \cdots \times [0, k_n]$
    - $k_i \in \mathbb{N}$
  - How many unit cells in $[0, k_1] \times [0, k_2] \times [0, k_3]$?
    - $k_1.k_2.k_3$
  - How many samples?

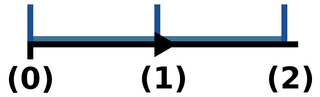


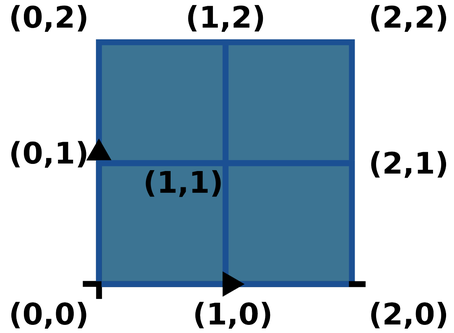


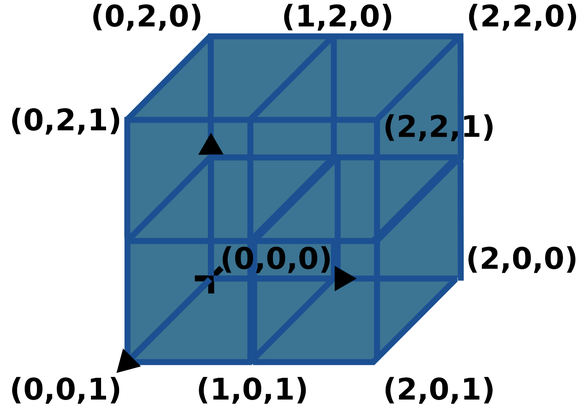


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Finite collection of unit cells
  - Entirely covering the direct product of closed intervals, such that
    - $[0, k_1] \times [0, k_2] \times \cdots \times [0, k_n]$
    - $k_i \in \mathbb{N}$
  - How many unit cells in $[0, k_1] \times [0, k_2] \times [0, k_3]$?
    - $k_1.k_2.k_3$
  - How many samples? $(k_1 + 1).(k_2 + 1).(k_3 + 1)$







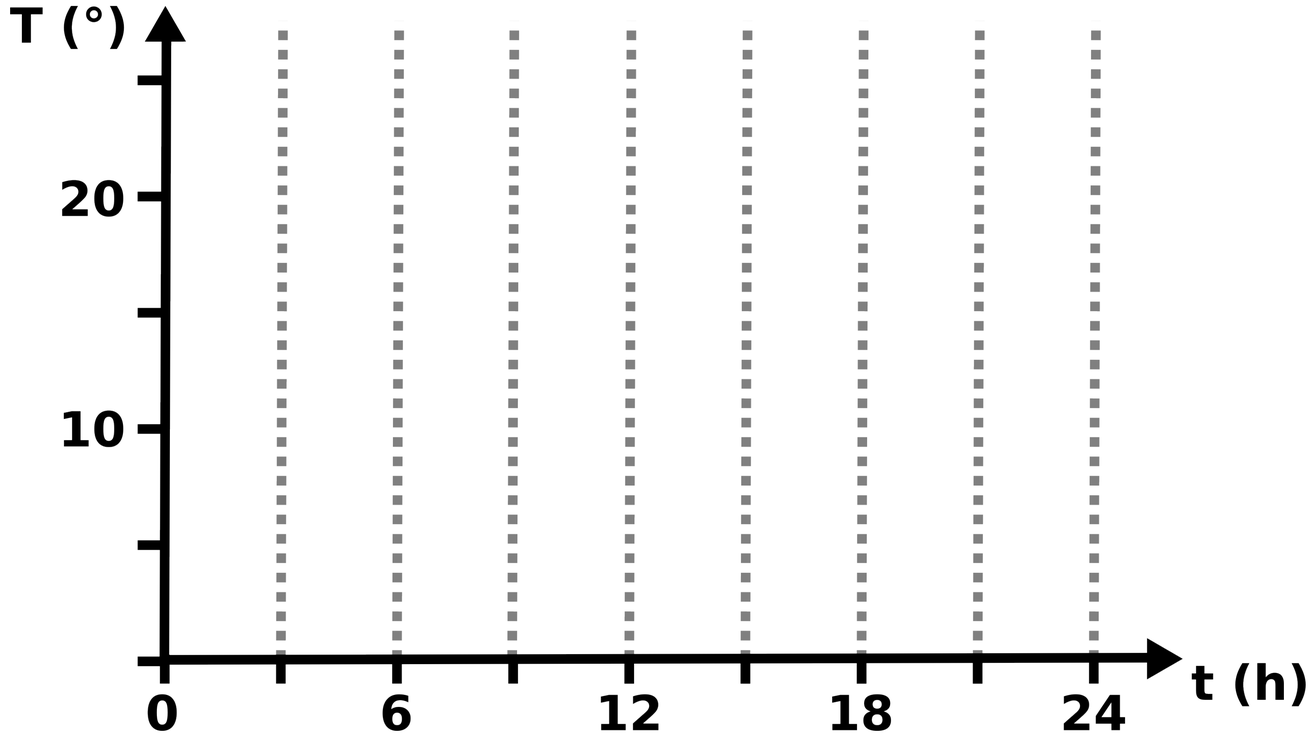
Regular grids of $\mathbb{R}$

Regular grids of $\mathbb{R}$

$$f : [0, 24] \to \mathbb{R}$$

Regular grids of $\mathbb{R}$

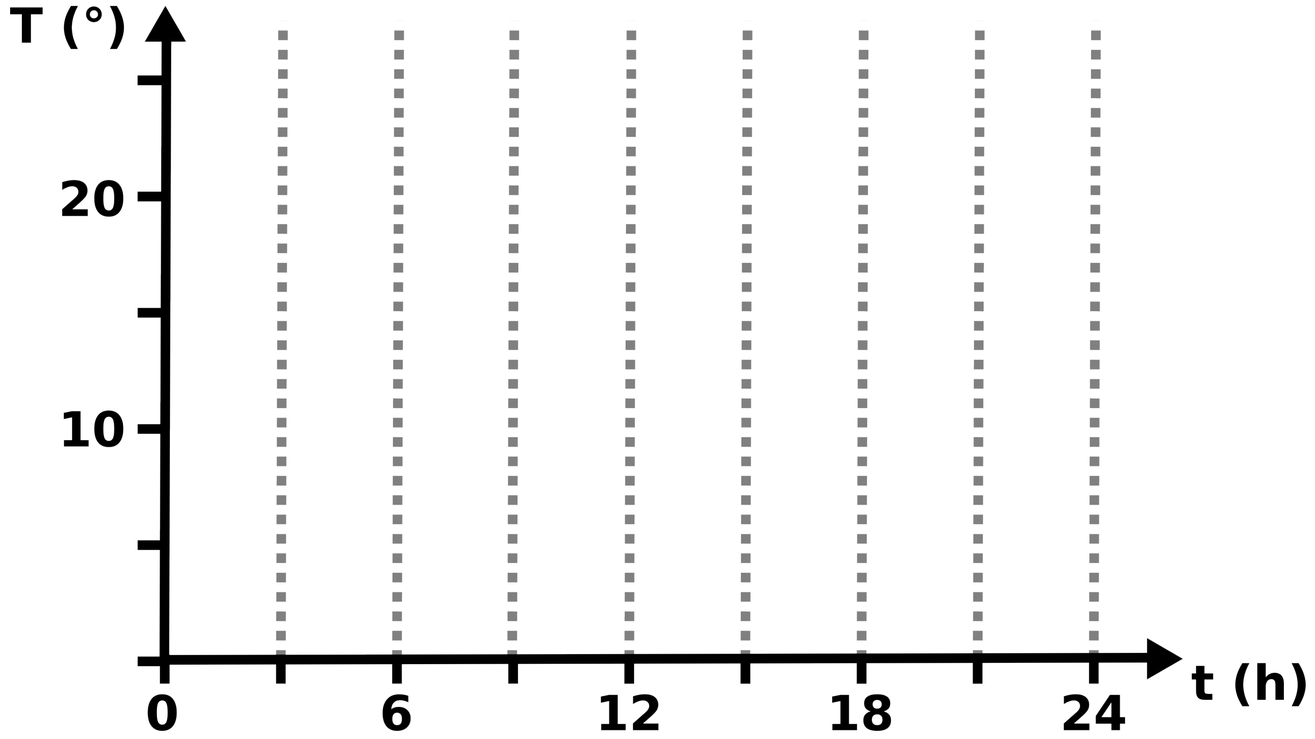
$$f : [0, 24] \to \mathbb{R}$$



Regular grids of $\mathbb{R}$

- We need
  - A one-dimensional structure

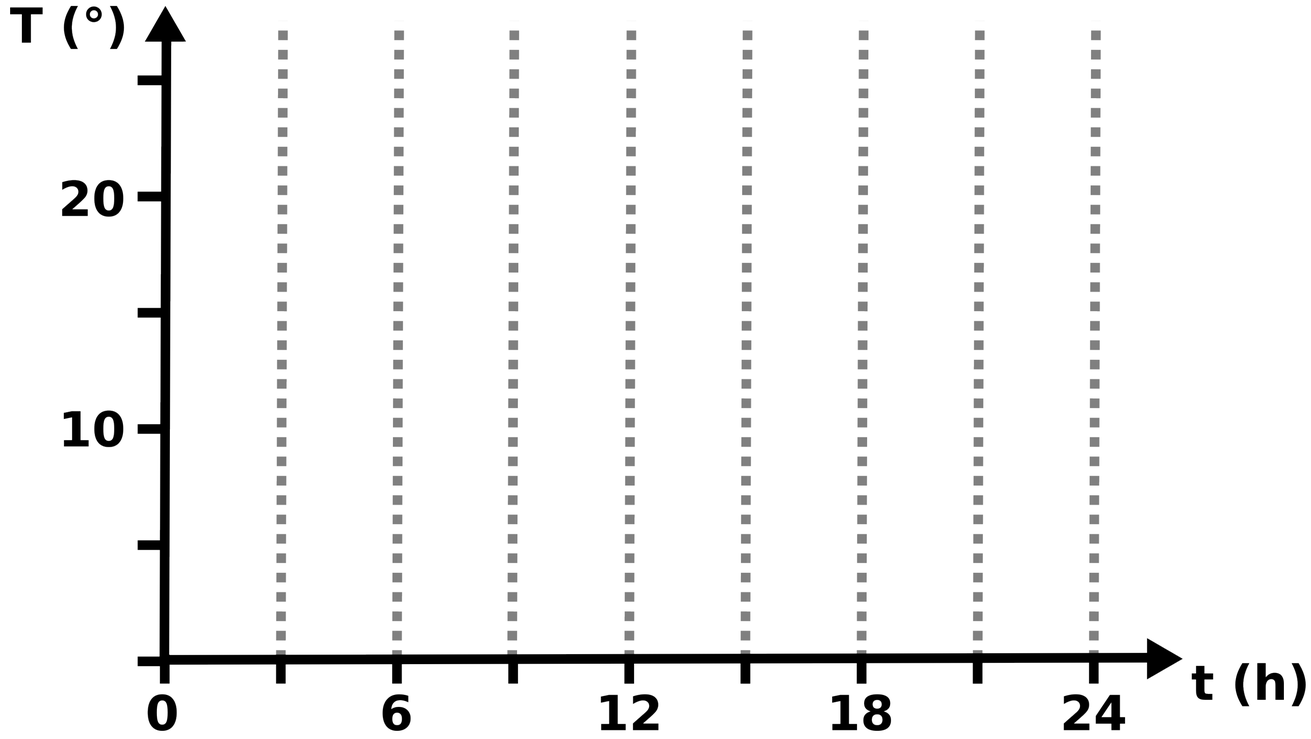
$$f : [0, 24] \rightarrow \mathbb{R}$$



Regular grids of $\mathbb{R}$

- We need
  - A one-dimensional structure
  - To store values
    - Scalars, vectors, tensors...

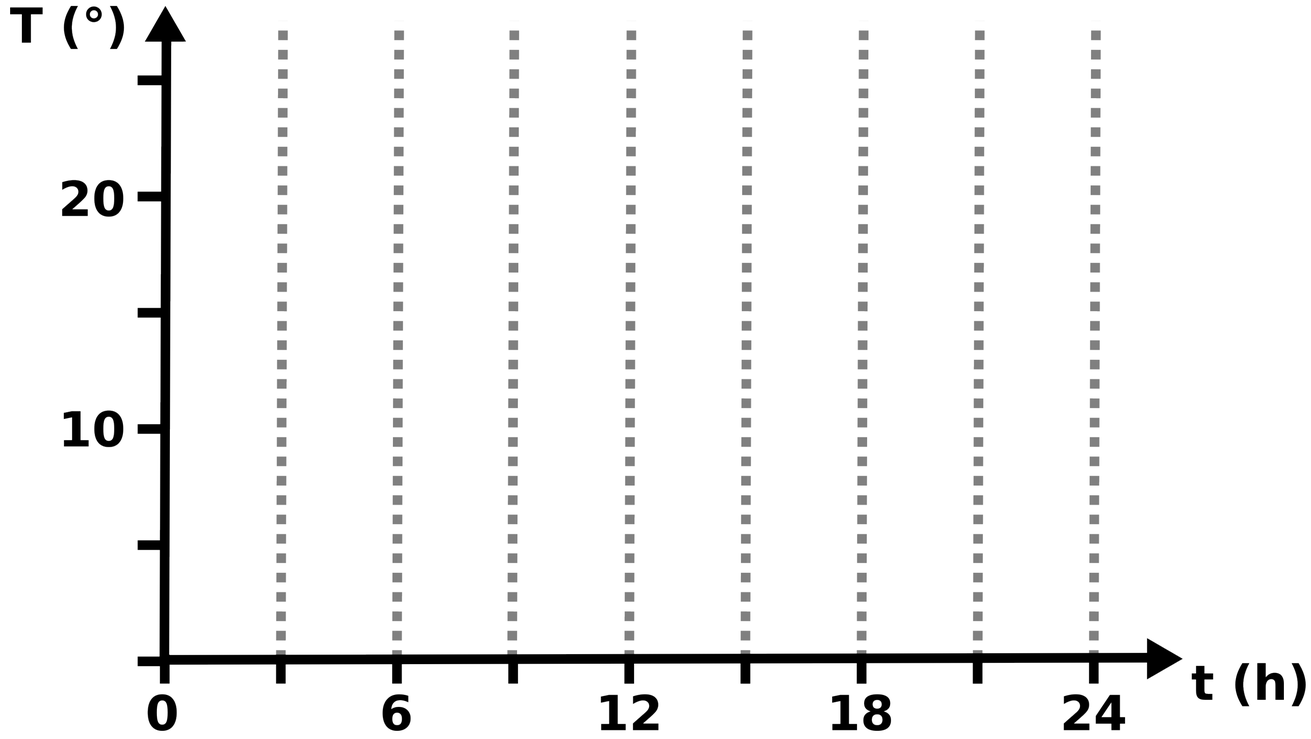
$$f : [0, 24] \to \mathbb{R}$$



Regular grids of $\mathbb{R}$

- We need
  - A one-dimensional structure
  - To store values
    - Scalars, vectors, tensors...
- Arrays :)

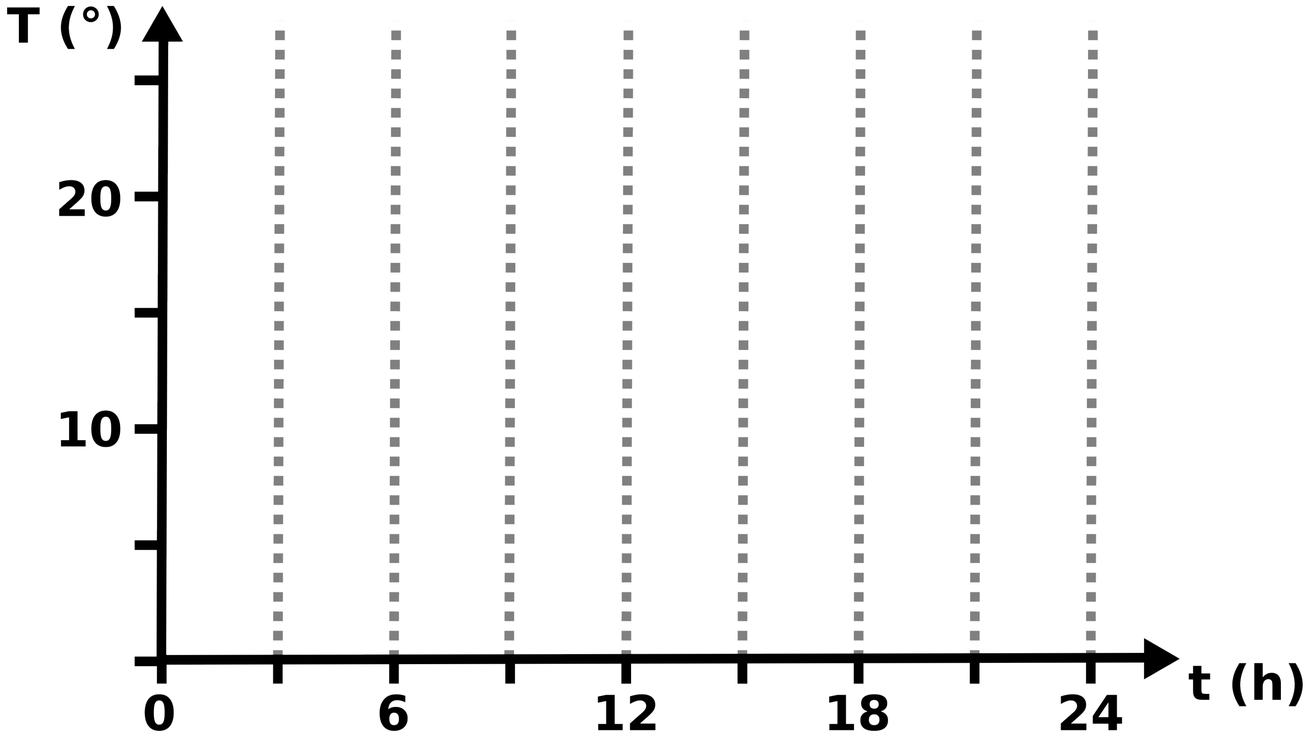
$$f : [0, 24] \to \mathbb{R}$$



Regular grids of $\mathbb{R}$

- We need
  - A one-dimensional structure
  - To store values
    - Scalars, vectors, tensors...
- Arrays :)

$$f : [0, 24] \to \mathbb{R}$$

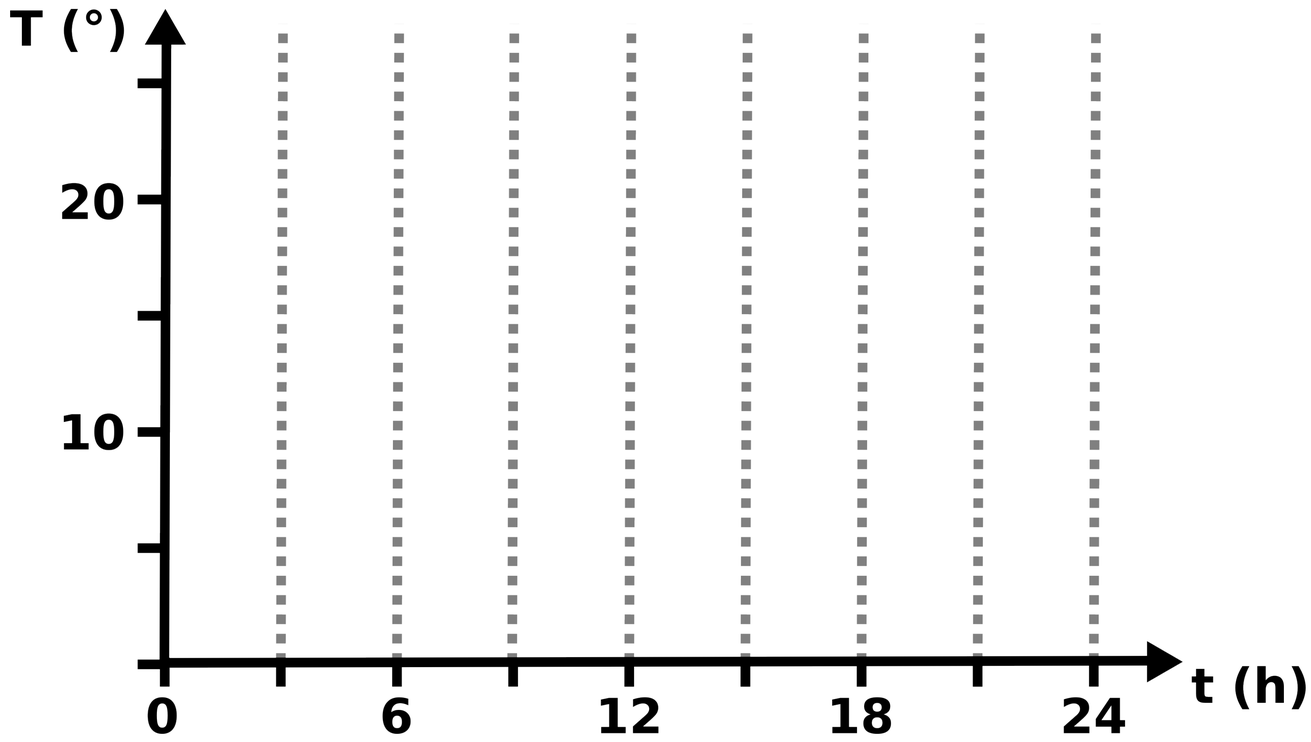


Regular grids of $\mathbb{R}$

- We need
  - A one-dimensional structure
  - To store values
    - Scalars, vectors, tensors...
- Arrays :)

9	7	10	12	16	18	11	10	7

$$f : [0, 24] \to \mathbb{R}$$

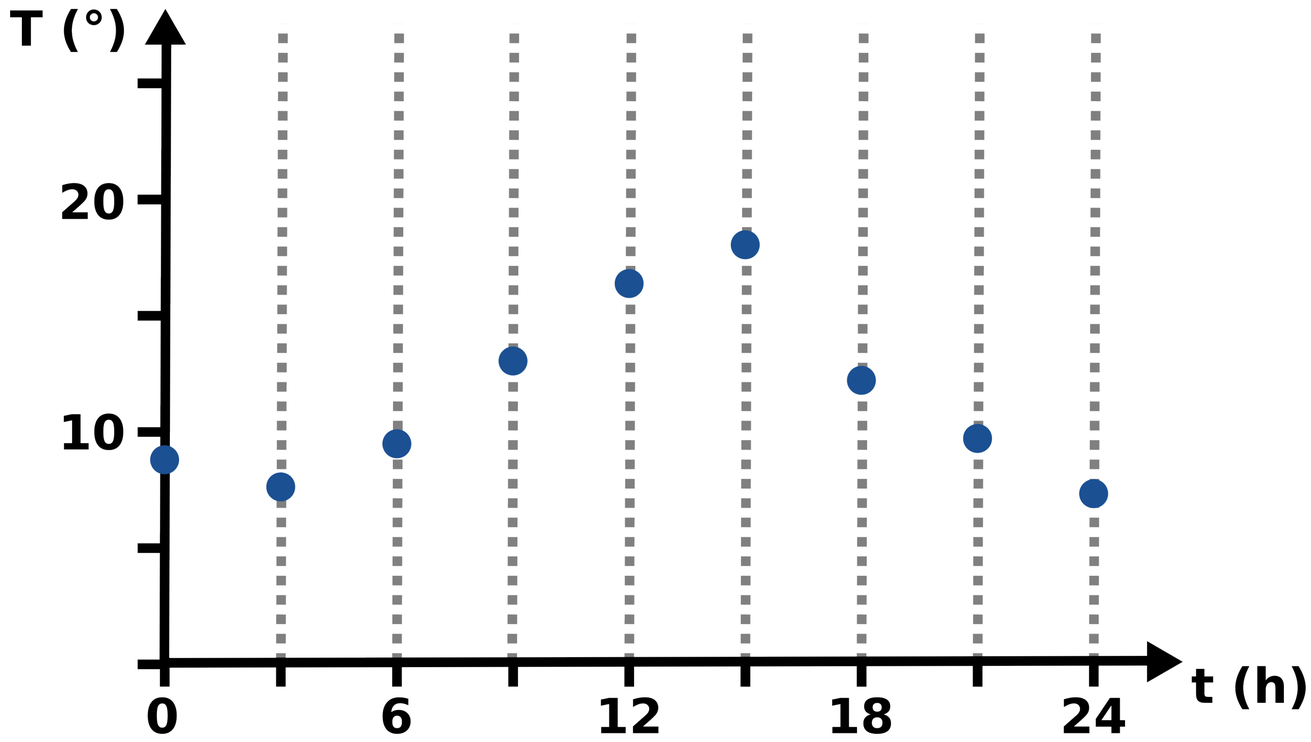


Regular grids of $\mathbb{R}$

- We need
  - A one-dimensional structure
  - To store values
    - Scalars, vectors, tensors...
- Arrays :)

9	7	10	12	16	18	11	10	7

$$f : [0, 24] \to \mathbb{R}$$



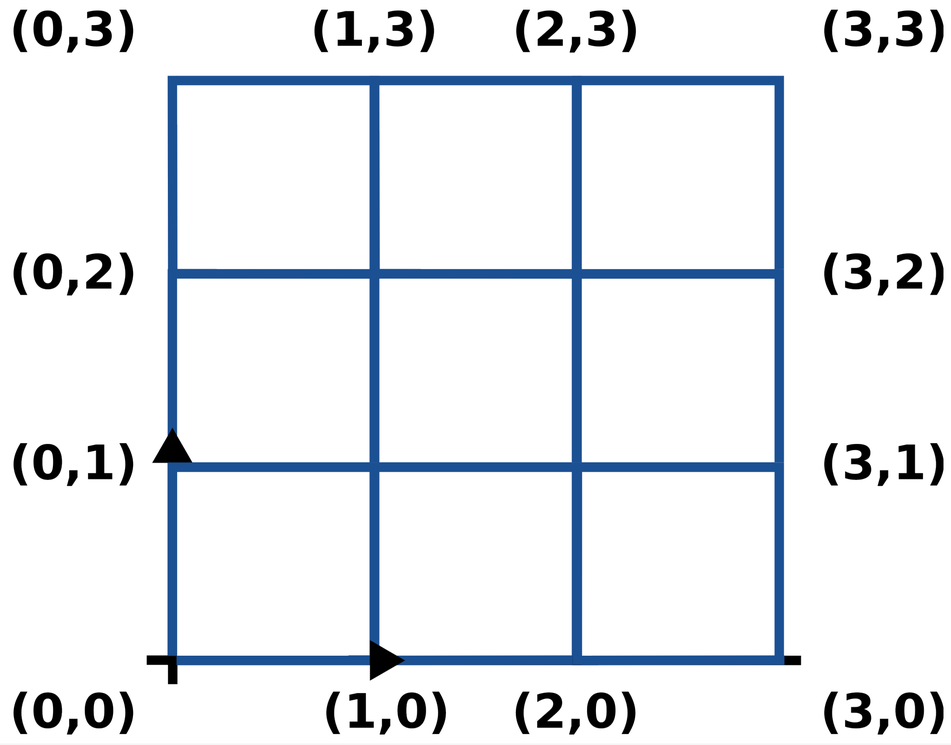
Regular grids of $\mathbb{R}^2$

Regular grids of $\mathbb{R}^2$

$$[0,3] \times [0,3]$$

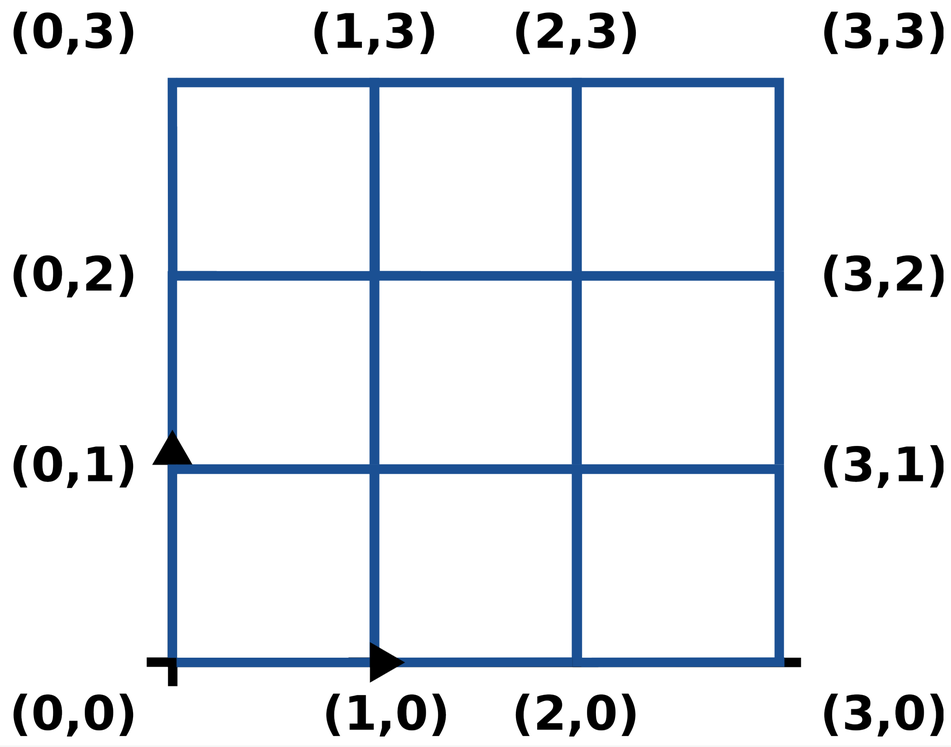
Regular grids of $\mathbb{R}^2$

$$[0,3] \times [0,3]$$



Regular grids of $\mathbb{R}^2$

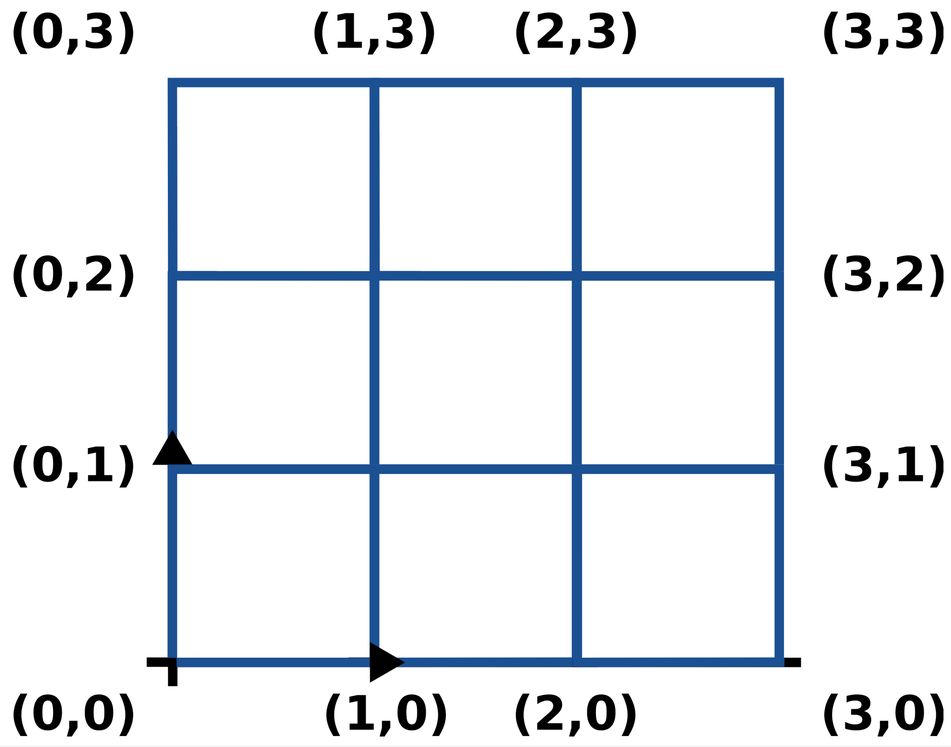
$$f : [0,3] \times [0,3] \to \mathbb{R}$$



Regular grids of $\mathbb{R}^2$

$$f(p) = \sqrt{p_x^2 + p_y^2}$$

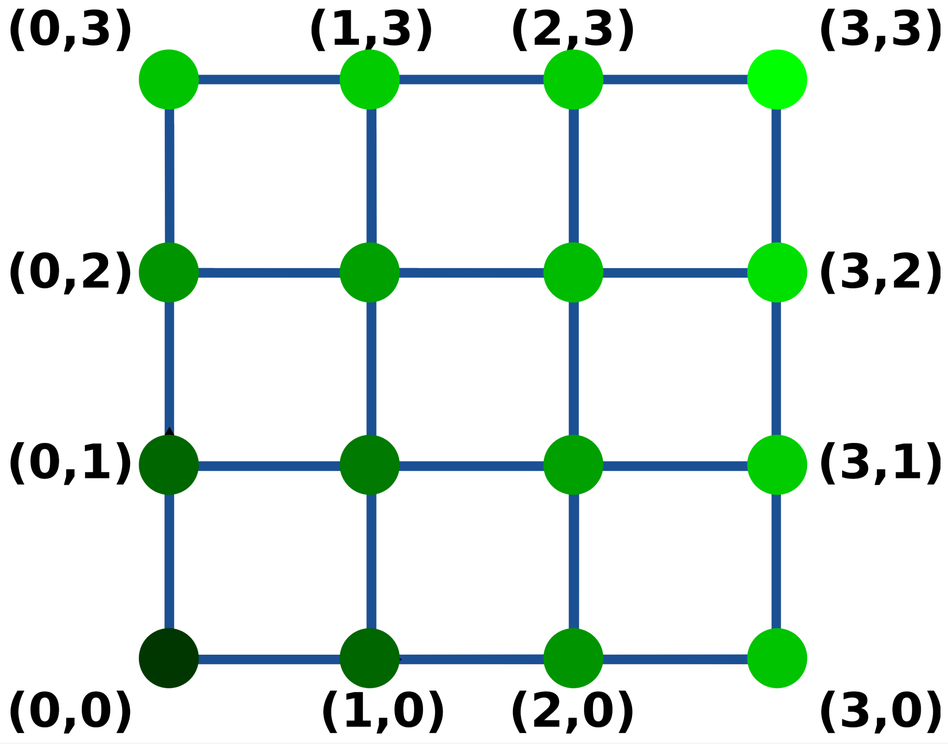
$$f : [0,3] \times [0,3] \rightarrow \mathbb{R}$$



Regular grids of $\mathbb{R}^2$

$$f(p) = \sqrt{p_x^2 + p_y^2}$$

$$f : [0,3] \times [0,3] \rightarrow \mathbb{R}$$

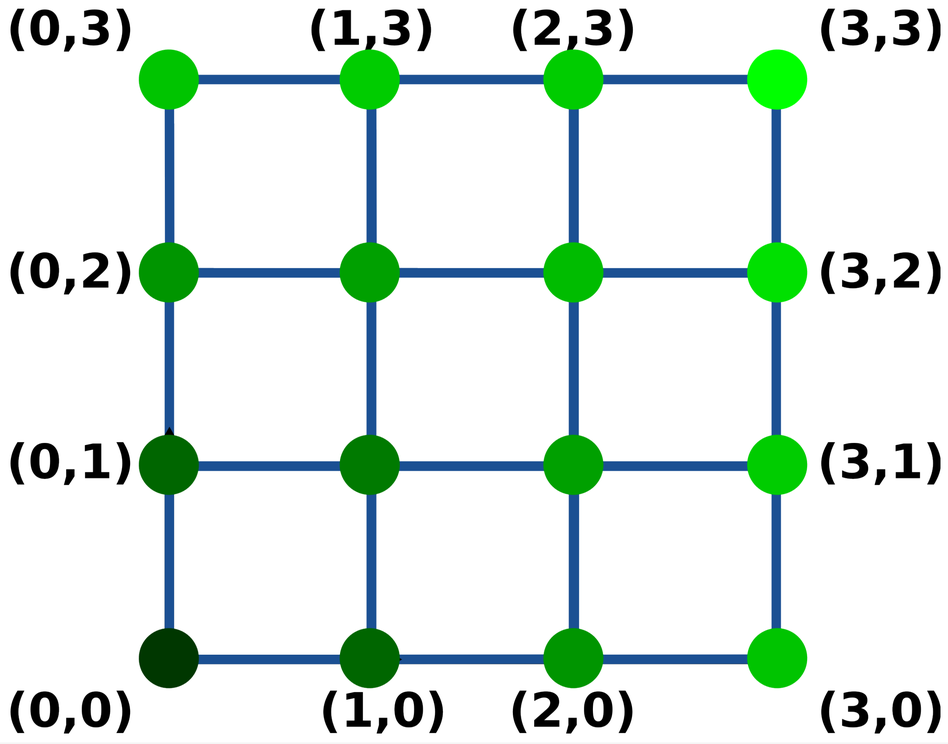


Regular grids of $\mathbb{R}^2$

$$f(p) = \sqrt{p_x^2 + p_y^2}$$

$$f : [0,3] \times [0,3] \rightarrow \mathbb{R}$$

- Problem
  - Computer memory is one-dimensional!

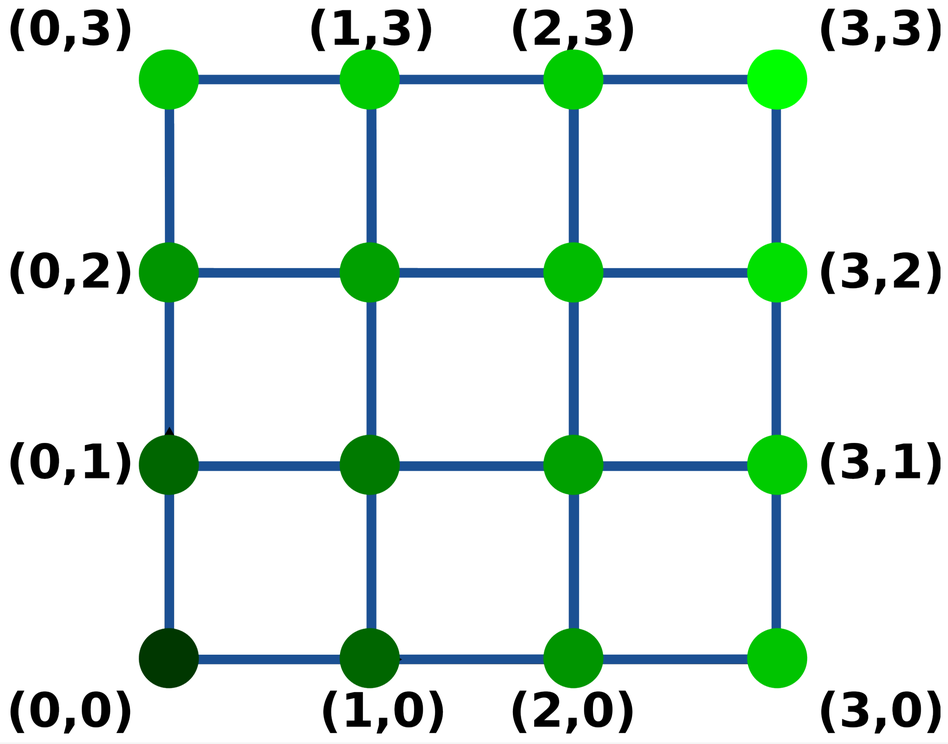


Regular grids of $\mathbb{R}^2$

$$f(p) = \sqrt{p_x^2 + p_y^2}$$

$$f : [0,3] \times [0,3] \rightarrow \mathbb{R}$$

- Problem
  - Computer memory is one-dimensional!

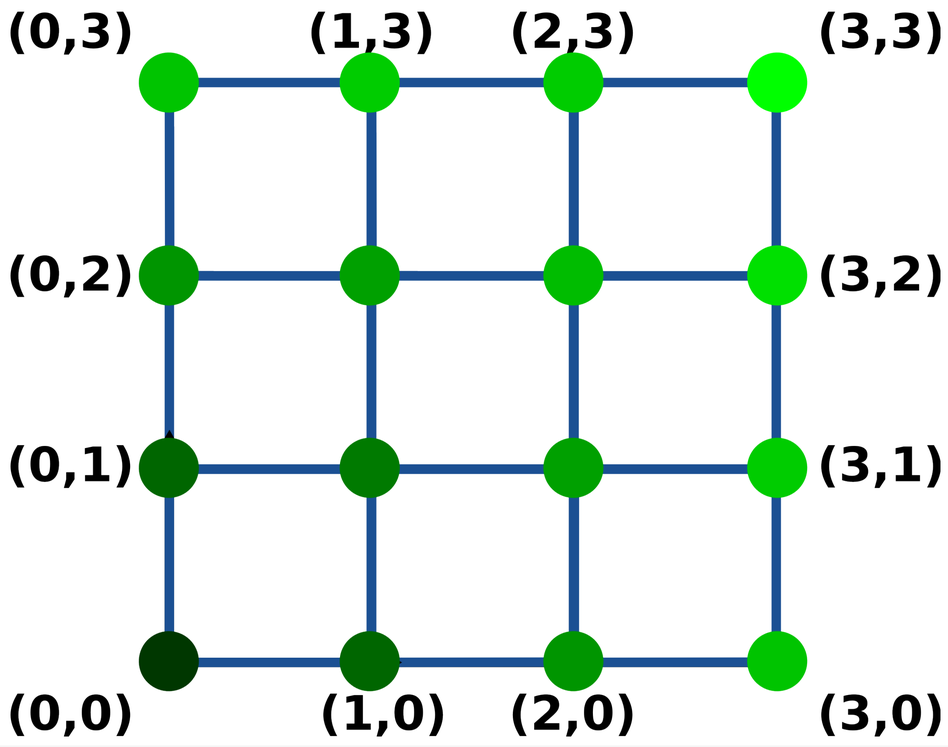


Regular grids of $\mathbb{R}^2$

$$f(p) = \sqrt{p_x^2 + p_y^2}$$

$$f : [0,3] \times [0,3] \rightarrow \mathbb{R}$$

- Problem
  - Computer memory is one-dimensional!
  - Order samples with a space-filling curve

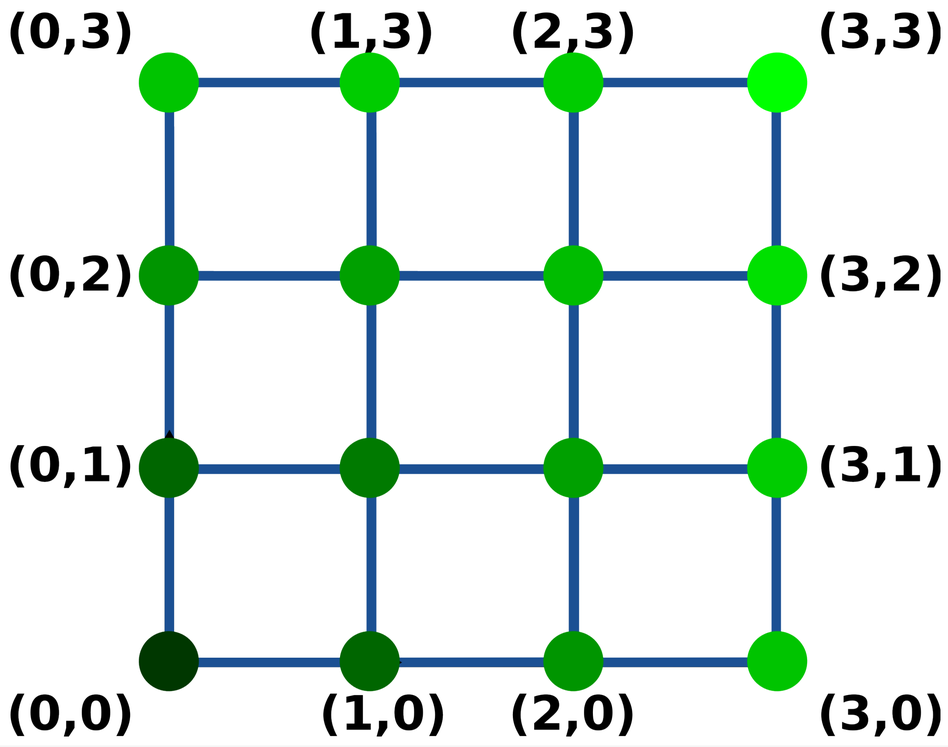


Regular grids of $\mathbb{R}^2$

$$f(p) = \sqrt{p_x^2 + p_y^2}$$

$$f : [0,3] \times [0,3] \rightarrow \mathbb{R}$$

- Problem
  - Computer memory is one-dimensional!
  - Order samples with a space-filling curve
    - One dimension at a time

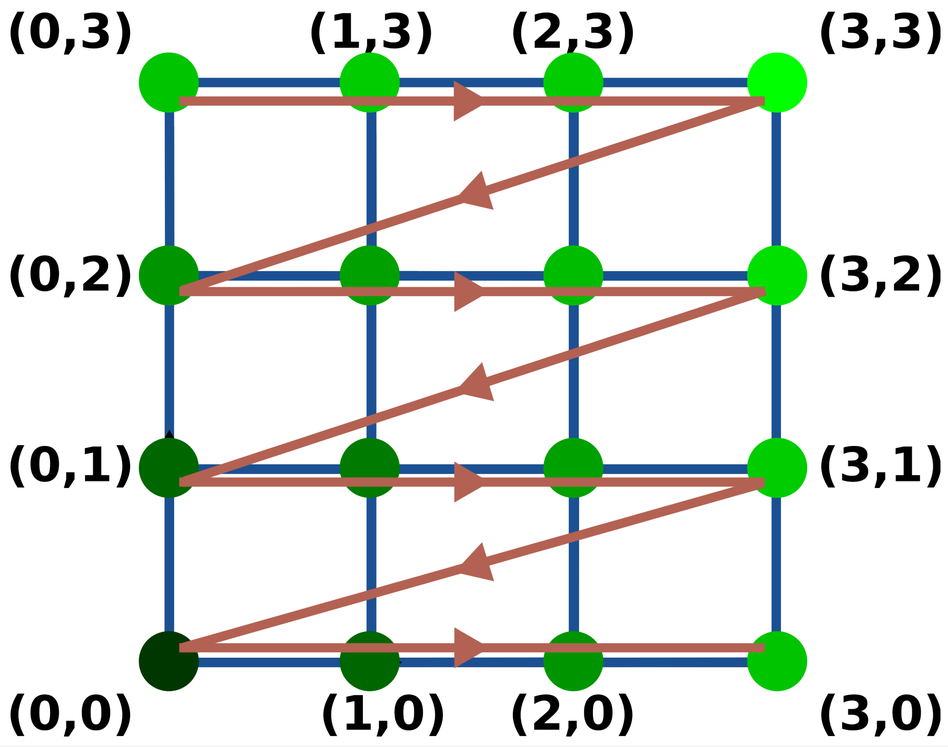


Regular grids of $\mathbb{R}^2$

$$f(p) = \sqrt{p_x^2 + p_y^2}$$

$$f : [0,3] \times [0,3] \rightarrow \mathbb{R}$$

- Problem
  - Computer memory is one-dimensional!
  - Order samples with a space-filling curve
    - One dimension at a time

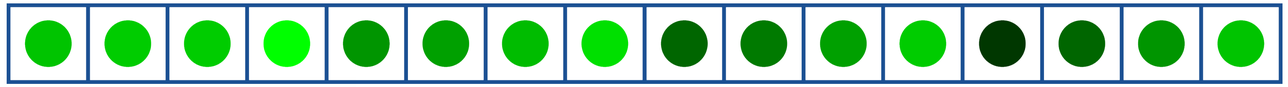


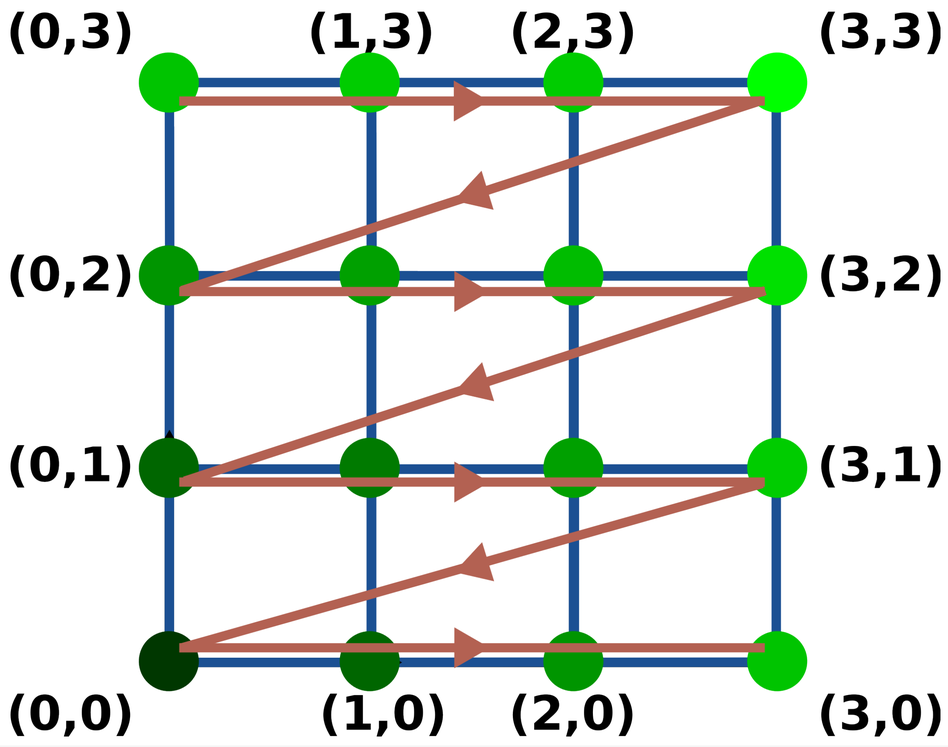
Regular grids of $\mathbb{R}^2$

$$f(p) = \sqrt{p_x^2 + p_y^2}$$

$$f : [0,3] \times [0,3] \rightarrow \mathbb{R}$$

- Problem
  - Computer memory is one-dimensional!
  - Order samples with a space-filling curve
    - One dimension at a time



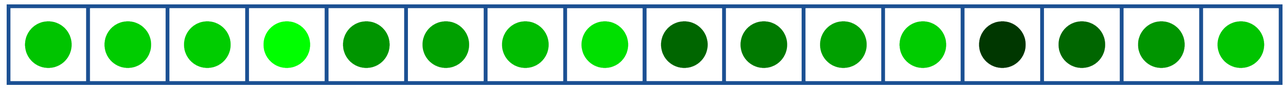


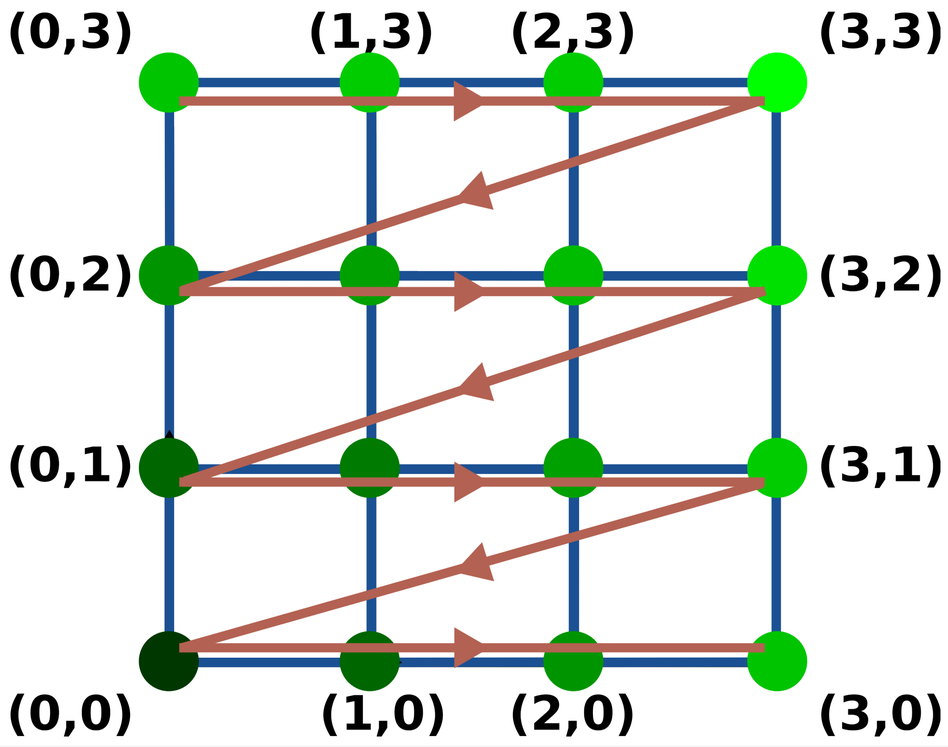
Regular grids of $\mathbb{R}^2$

$$f(p) = \sqrt{p_x^2 + p_y^2}$$

$$f : [0,3] \times [0,3] \rightarrow \mathbb{R}$$

- Problem
  - Computer memory is one-dimensional!
  - Order samples with a space-filling curve
    - One dimension at a time
    - Peano's curves
    - Hilbert's curves, ...



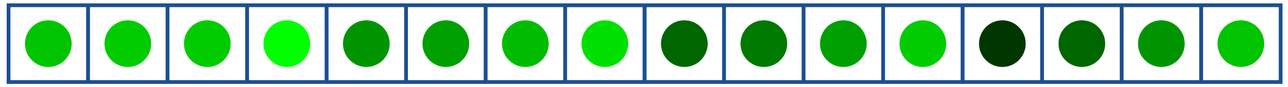


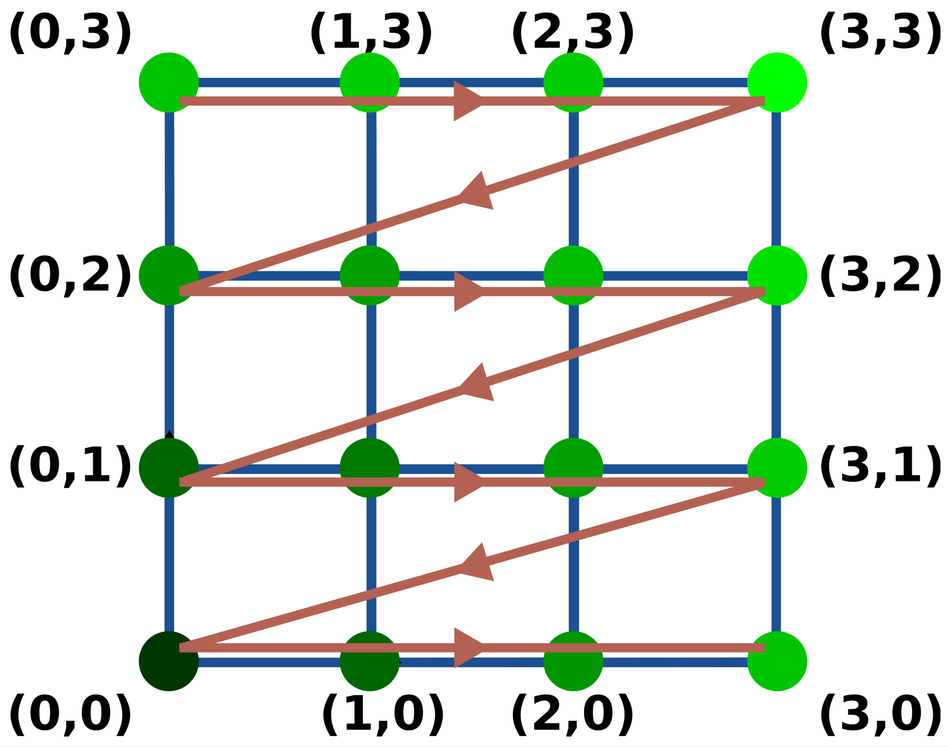
Regular grids of $\mathbb{R}^2$

$$f(p) = \sqrt{p_x^2 + p_y^2}$$

$$f : [0,3] \times [0,3] \rightarrow \mathbb{R}$$

- Problem
  - Computer memory is one-dimensional!
  - Order samples with a space-filling curve
    - One dimension at a time
    - Peano's curves
    - Hilbert's curves, ...
    - Lebesgue's curves (z-order)



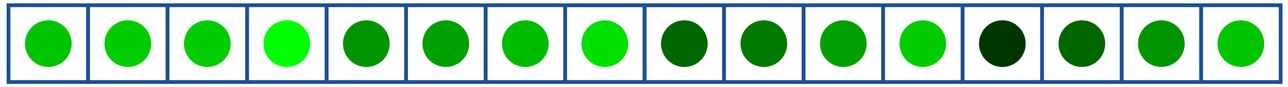


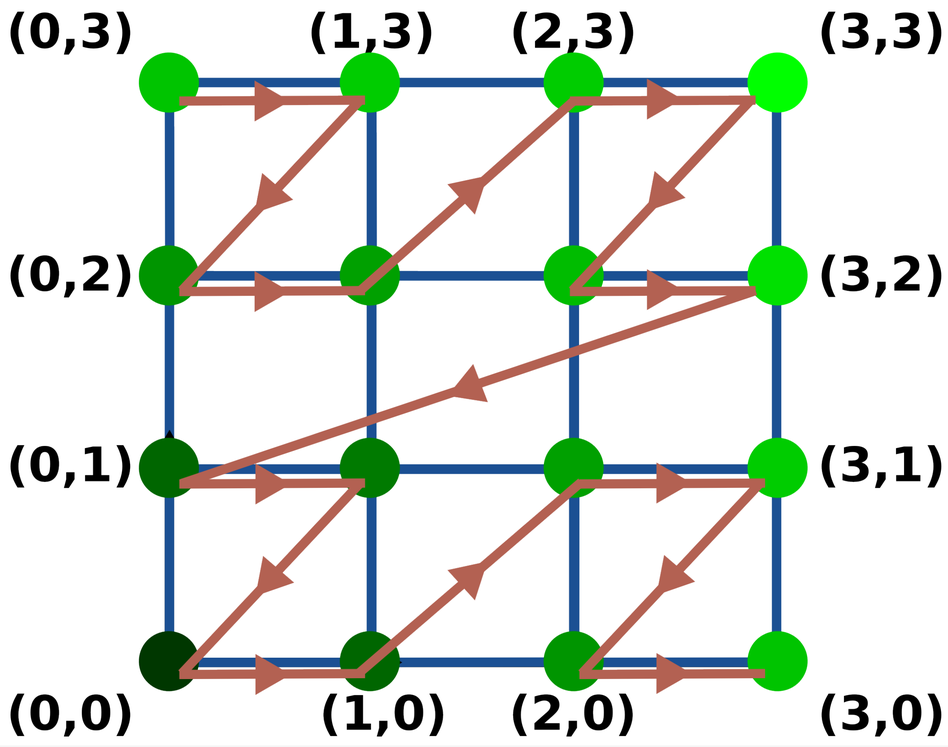
Regular grids of $\mathbb{R}^2$

$$f(p) = \sqrt{p_x^2 + p_y^2}$$

$$f : [0,3] \times [0,3] \rightarrow \mathbb{R}$$

- Problem
  - Computer memory is one-dimensional!
  - Order samples with a space-filling curve
    - One dimension at a time
    - Peano's curves
    - Hilbert's curves, ...
    - Lebesgue's curves (z-order)



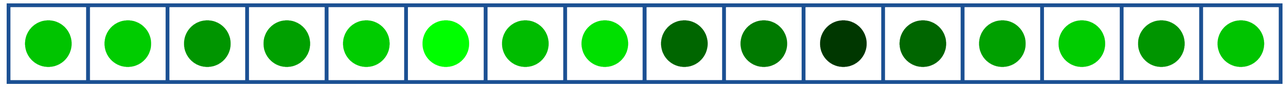


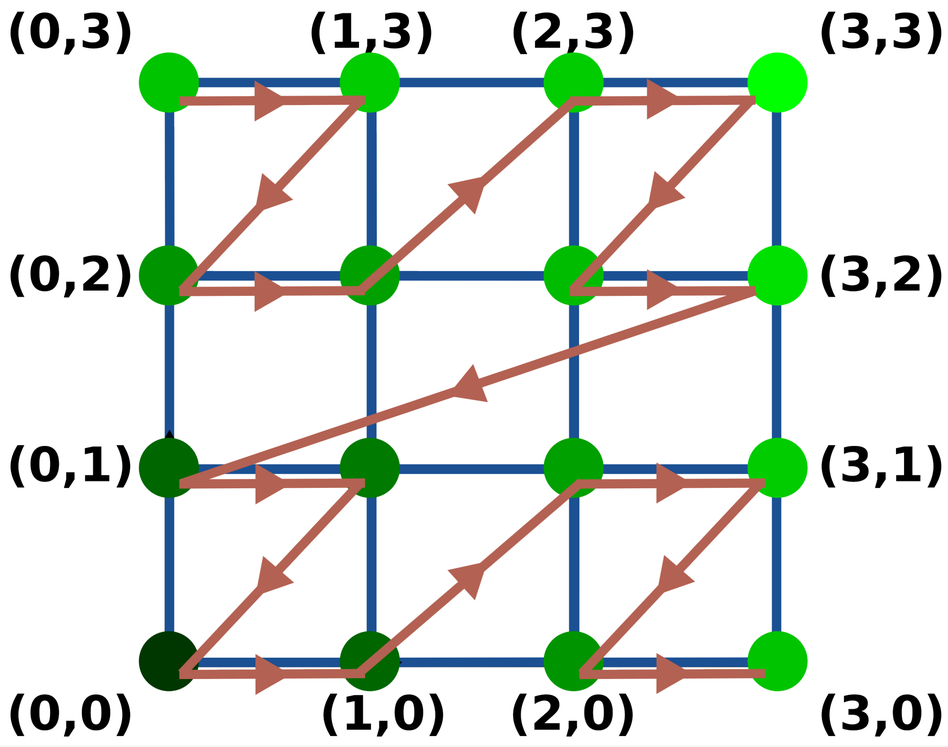
Regular grids of $\mathbb{R}^2$

$$f(p) = \sqrt{p_x^2 + p_y^2}$$

$$f : [0,3] \times [0,3] \rightarrow \mathbb{R}$$

- Problem
  - Computer memory is one-dimensional!
  - Order samples with a space-filling curve
    - One dimension at a time
    - Peano's curves
    - Hilbert's curves, ...
    - Lebesgue's curves (z-order)





Regular grids of $\mathbb{R}^3$

[Wikipedia]

Regular grids of $\mathbb{R}^3$

- Same problem
  - Computer memory is one-dimensional!

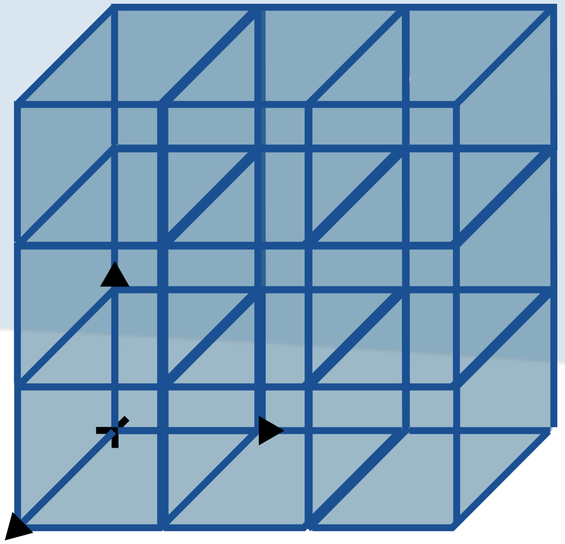
[Wikipedia]

Regular grids of $\mathbb{R}^3$

- Same problem
  - Computer memory is one-dimensional!
  - Same solutions
    - One dimension at a time
    - Peano's curves
    - Hilbert's curves, ...
    - Lebesgue's curves (z-order)

[Wikipedia]

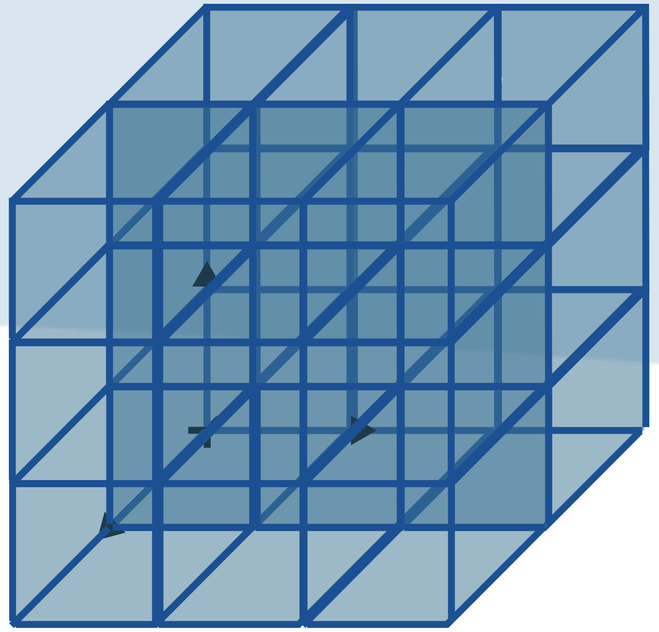
Regular grids of $\mathbb{R}^3$



- Same problem
  - Computer memory is one-dimensional!
  - Same solutions
    - One dimension at a time
    - Peano's curves
    - Hilbert's curves, ...
    - Lebesgue's curves (z-order)

[Wikipedia]

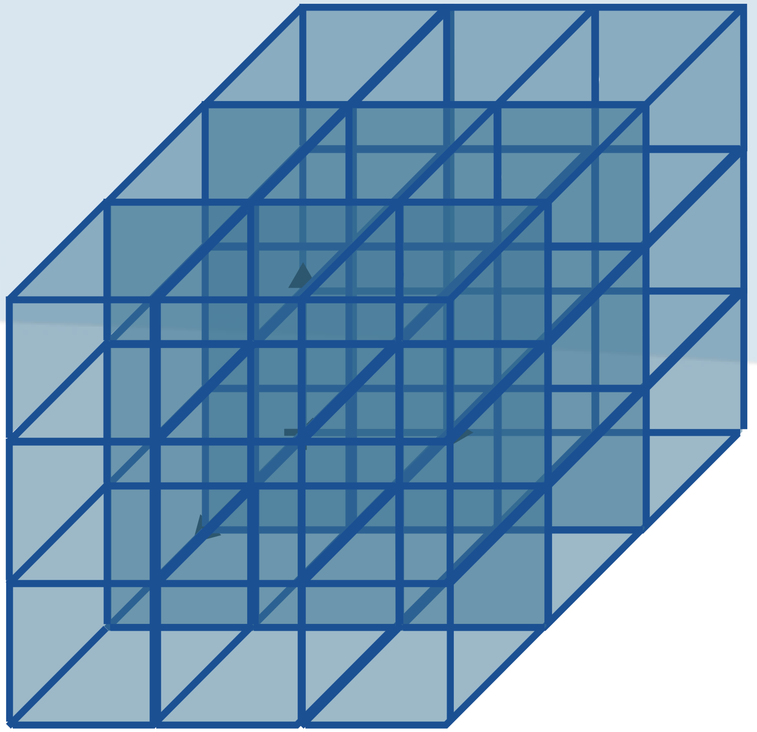
Regular grids of $\mathbb{R}^3$



- Same problem
  - Computer memory is one-dimensional!
  - Same solutions
    - One dimension at a time
    - Peano's curves
    - Hilbert's curves, ...
    - Lebesgue's curves (z-order)

[Wikipedia]

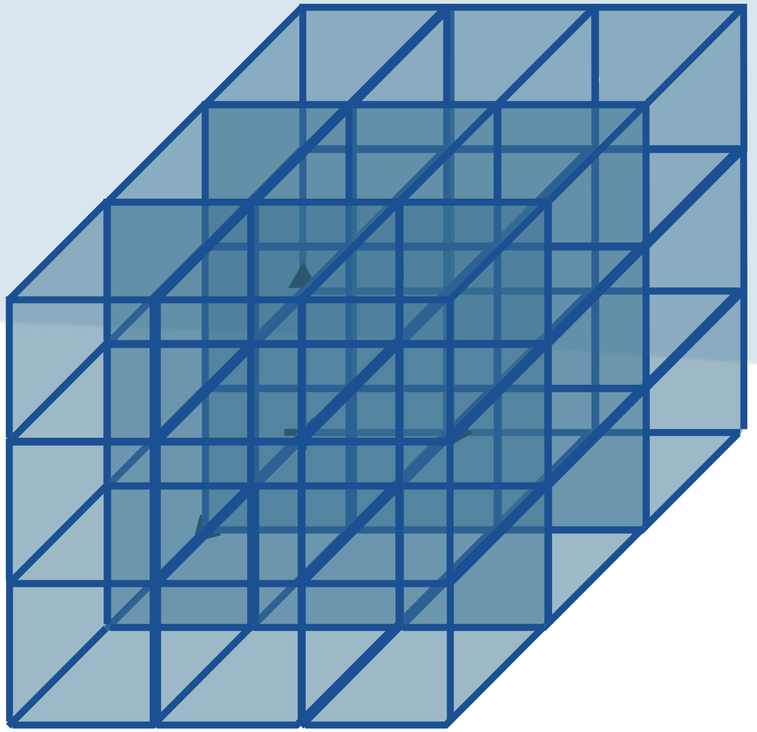
Regular grids of $\mathbb{R}^3$



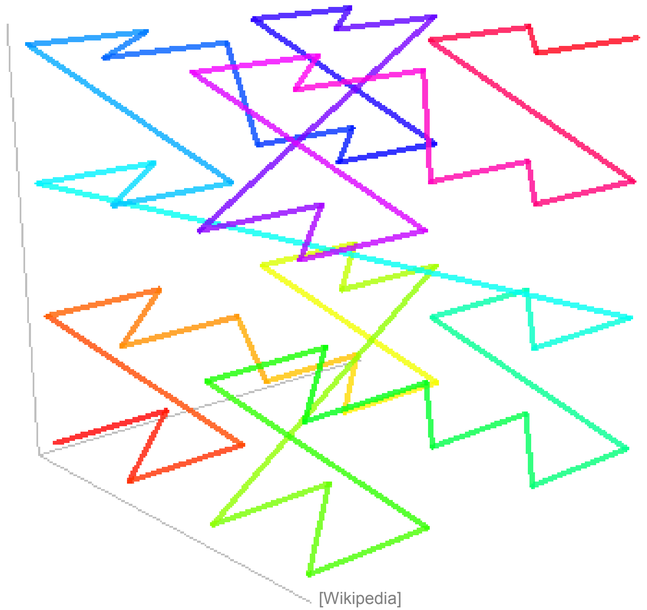
- Same problem
  - Computer memory is one-dimensional!
  - Same solutions
    - One dimension at a time
    - Peano's curves
    - Hilbert's curves, ...
    - Lebesgue's curves (z-order)

[Wikipedia]

Regular grids of $\mathbb{R}^3$

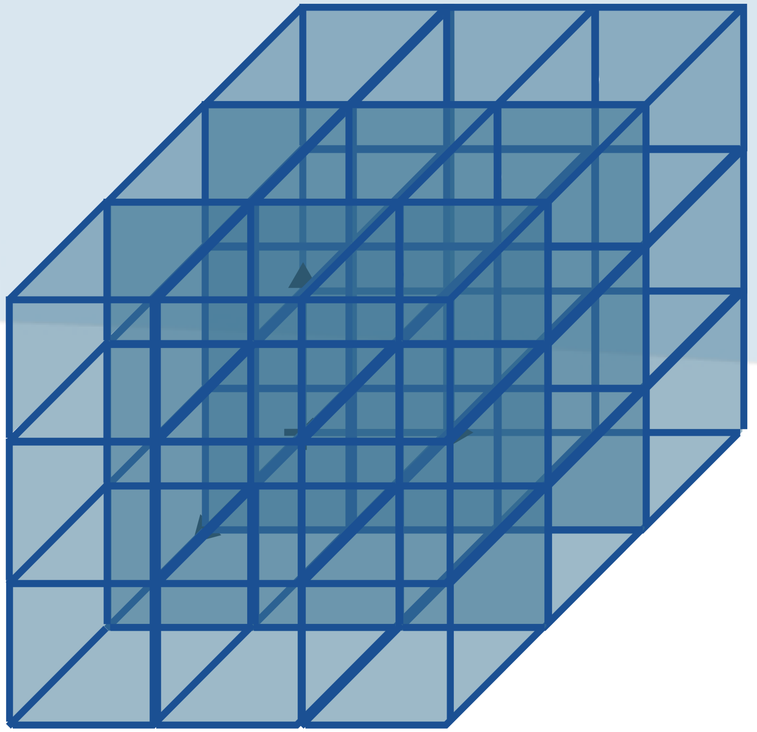


- Same problem
  - Computer memory is one-dimensional!
  - Same solutions
    - One dimension at a time
    - Peano's curves
    - Hilbert's curves, ...
    - Lebesgue's curves (z-order)

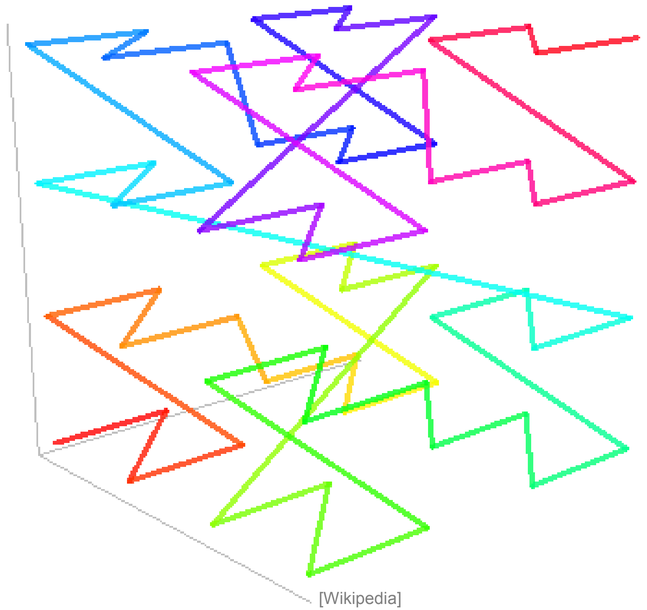


[Wikipedia]

Regular grids of $\mathbb{R}^n$

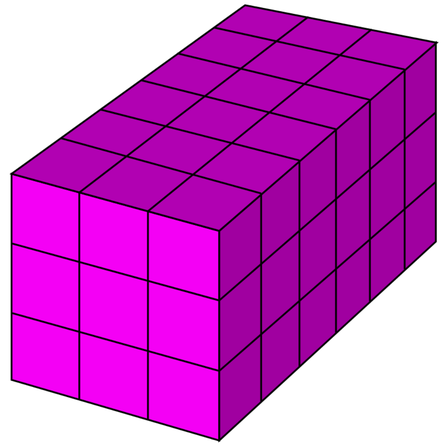


- Same problem
  - Computer memory is one-dimensional!
  - Same solutions
    - One dimension at a time
    - Peano's curves
    - Hilbert's curves, ...
    - Lebesgue's curves (z-order)
- Generalizes to higher dimensions



[Wikipedia]

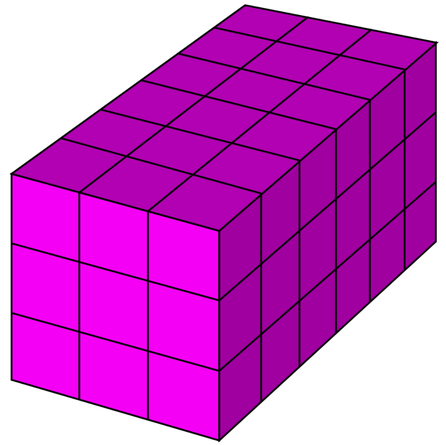
Embedding functions



[Wikipedia]

Embedding functions

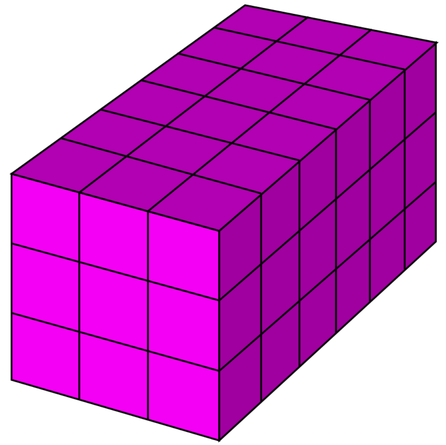
- Problem
  - What if we want to adapt the sampling to the geometry of the data?



[Wikipedia]

Embedding functions

- Problem
  - What if we want to adapt the sampling to the geometry of the data?

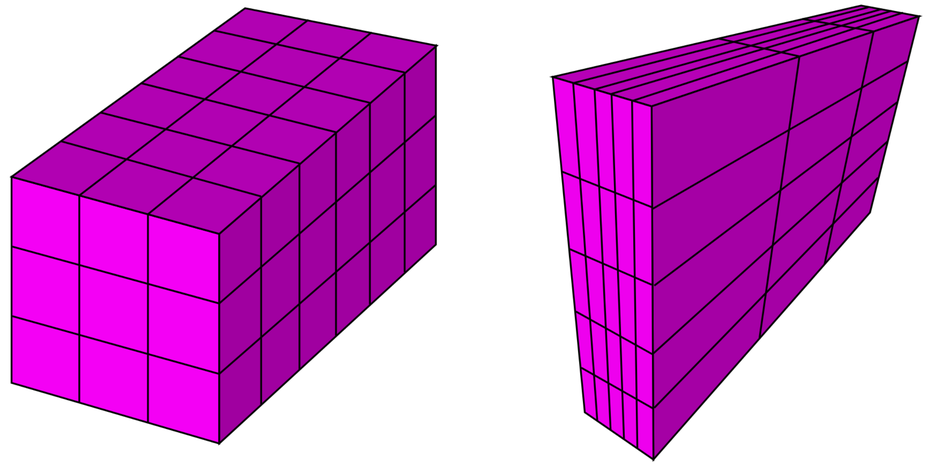


- Embedding functions
  - $e : \mathcal{G} \rightarrow \mathbb{R}^n$

[Wikipedia]

Embedding functions

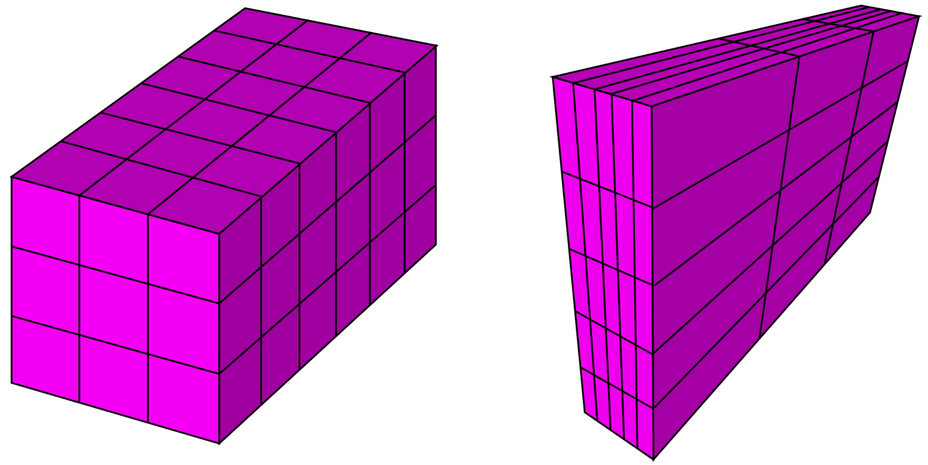
- Problem
  - What if we want to adapt the sampling to the geometry of the data?
- Embedding functions
  - $e : \mathcal{G} \rightarrow \mathbb{R}^n$



[Wikipedia]

Embedding functions

- Problem
  - What if we want to adapt the sampling to the geometry of the data?

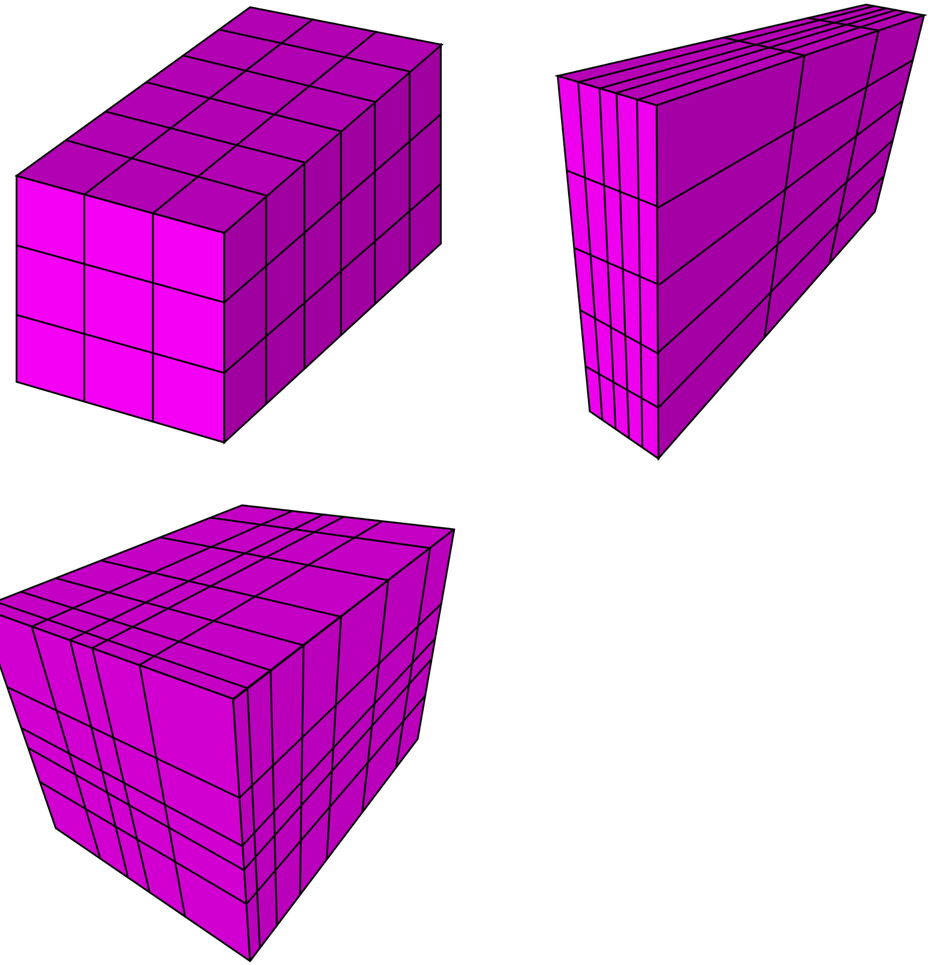


- Embedding functions
  - $e : \mathcal{G} \rightarrow \mathbb{R}^n$
  - Attributes attached to the grid

[Wikipedia]

Embedding functions

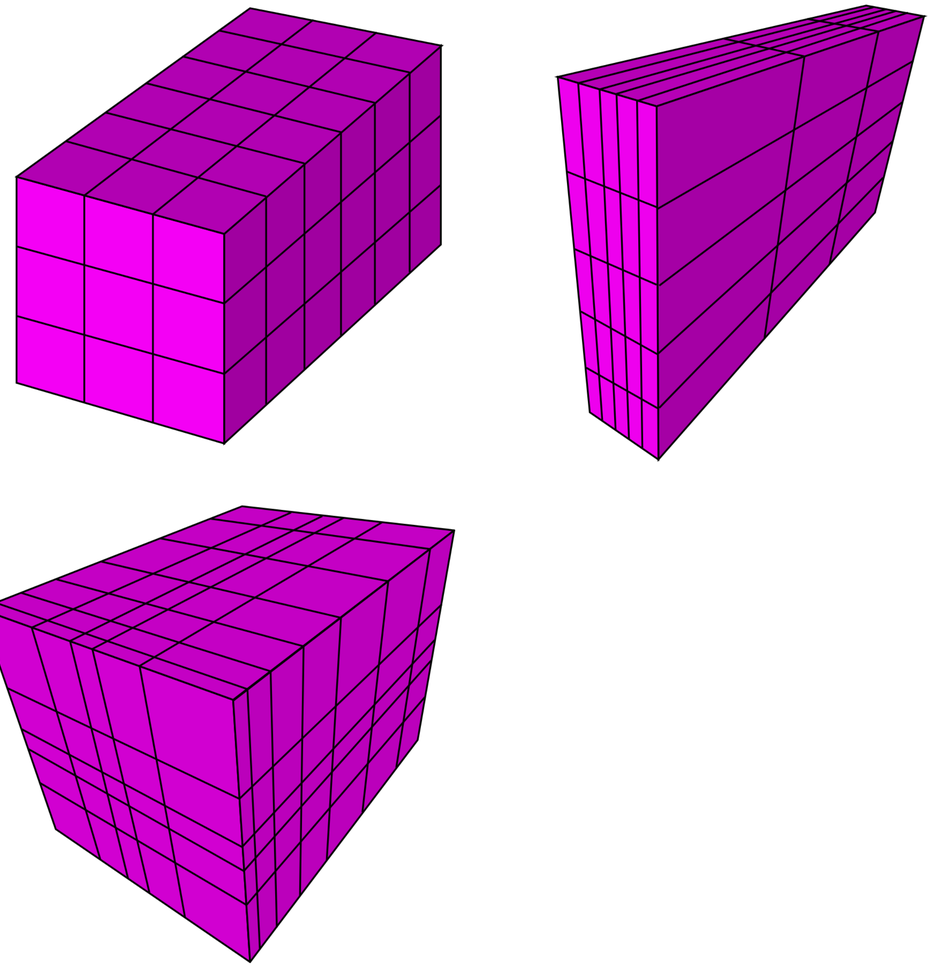
- Problem
  - What if we want to adapt the sampling to the geometry of the data?
- Embedding functions
  - $e : \mathcal{G} \rightarrow \mathbb{R}^n$
  - Attributes attached to the grid
- Non uniform sampling



[Wikipedia]

Embedding functions

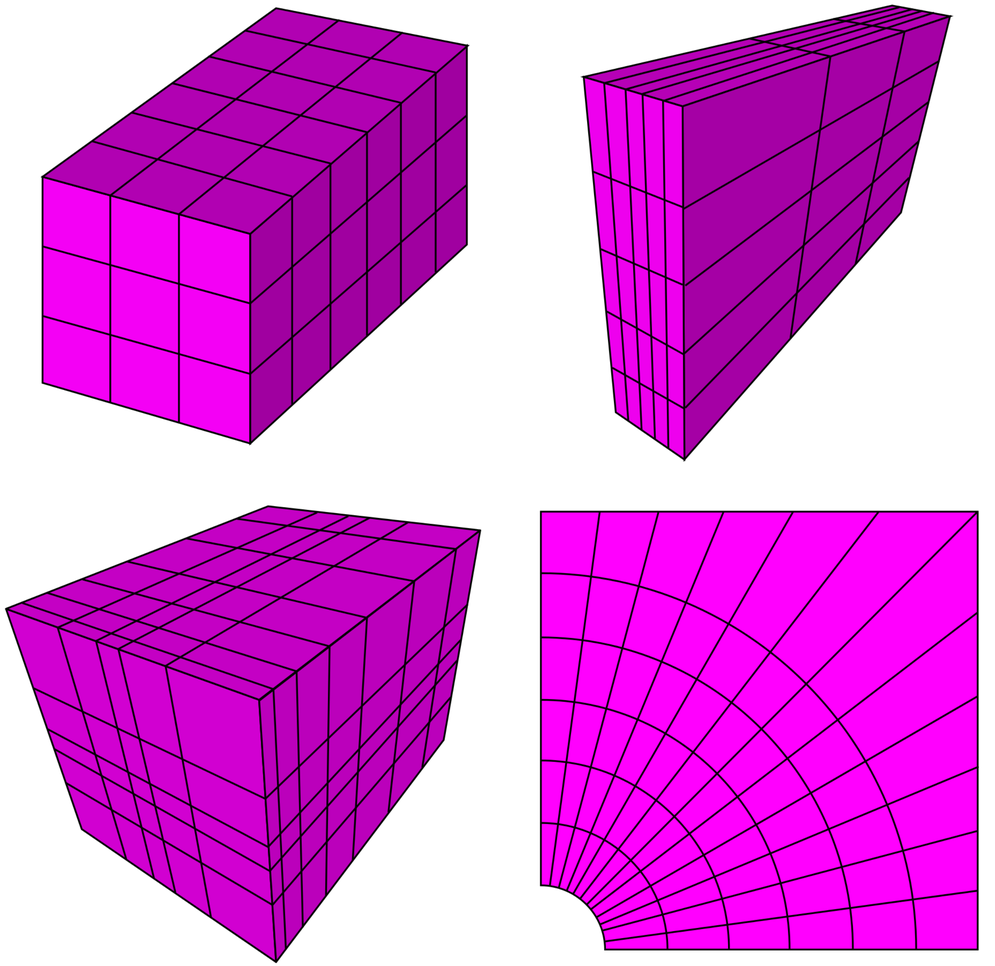
- Problem
  - What if we want to adapt the sampling to the geometry of the data?
- Embedding functions
  - $e : \mathcal{G} \rightarrow \mathbb{R}^n$
  - Attributes attached to the grid
- Non uniform sampling
  - *Rectilinear* grid



[Wikipedia]

Embedding functions

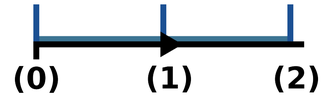
- Problem
  - What if we want to adapt the sampling to the geometry of the data?
- Embedding functions
  - $e : \mathcal{G} \rightarrow \mathbb{R}^n$
  - Attributes attached to the grid
- Non uniform sampling
  - *Rectilinear* grid

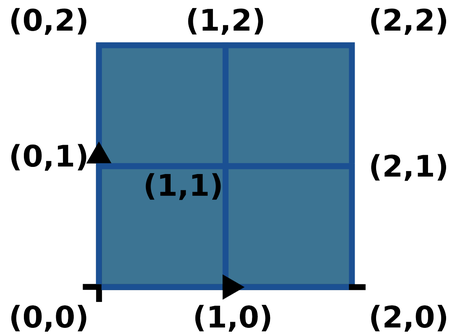


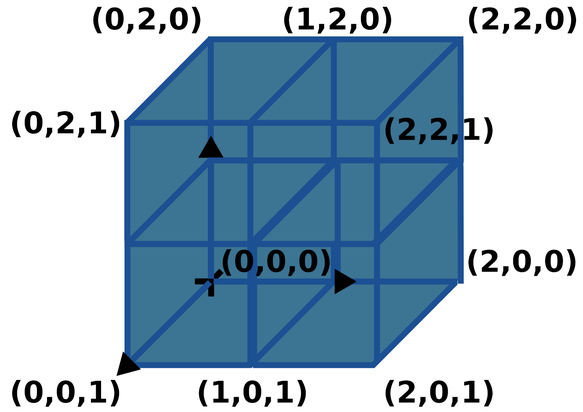
[Wikipedia]

Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$

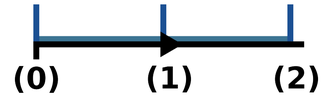


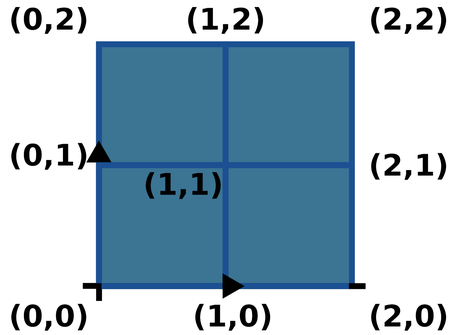


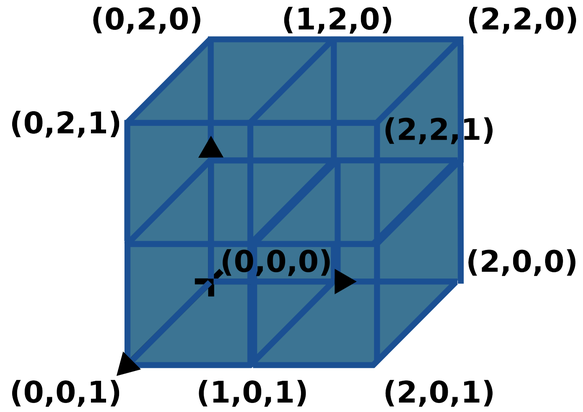


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Set of **vertices** (samples) in $\mathbb{R}^n$
    - With embedding functions to $\mathbb{R}^n$

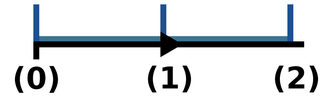


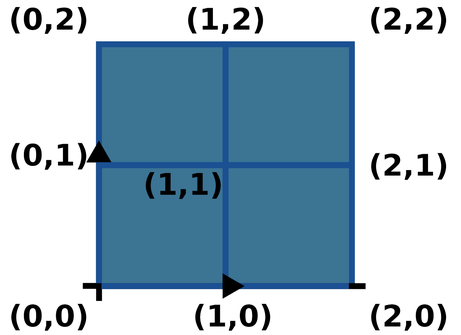


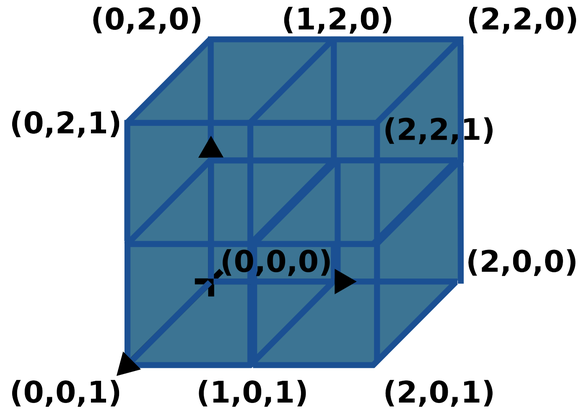


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Set of **vertices** (samples) in $\mathbb{R}^n$
    - With embedding functions to $\mathbb{R}^n$
  - **Connectivity**

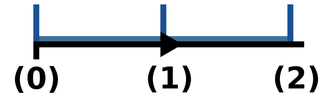


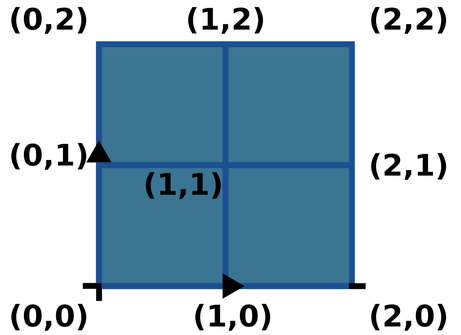


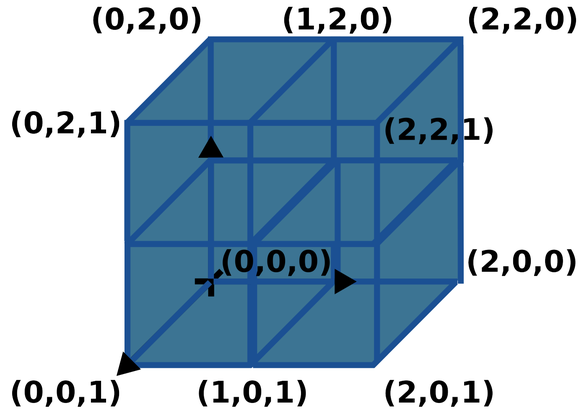


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Set of **vertices** (samples) in $\mathbb{R}^n$
    - With embedding functions to $\mathbb{R}^n$
  - **Connectivity**
    - Unit cells (pixels, voxels)

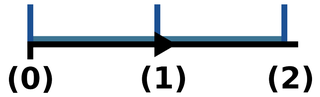


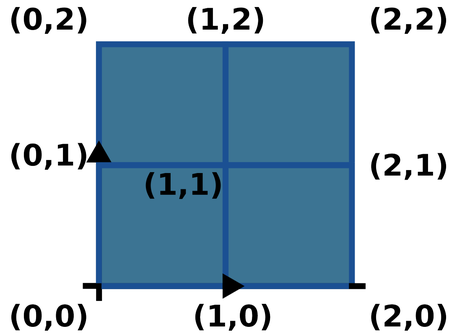


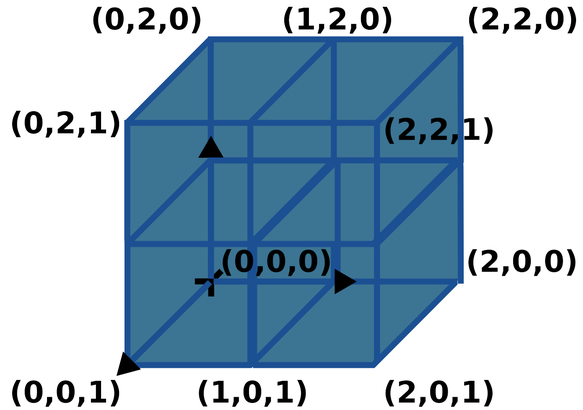


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Set of **vertices** (samples) in $\mathbb{R}^n$
    - With embedding functions to $\mathbb{R}^n$
  - **Connectivity**
    - Unit cells (pixels, voxels)
    - Implicitly encoded by the samples (space fill)

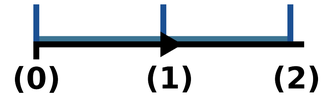


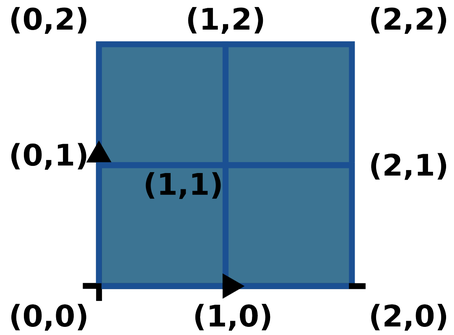


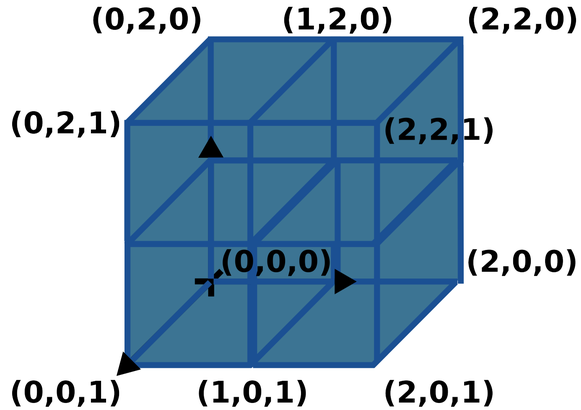


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Set of **vertices** (samples) in $\mathbb{R}^n$
    - With embedding functions to $\mathbb{R}^n$
  - **Connectivity**
    - Unit cells (pixels, voxels)
    - Implicitly encoded by the samples (space fill)
  - What's missing?

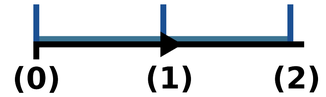


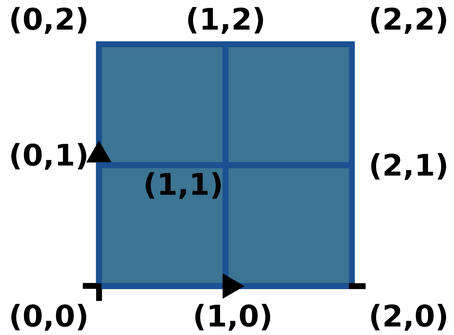


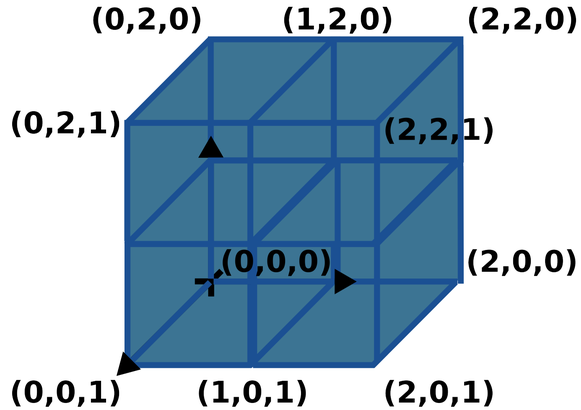


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Set of **vertices** (samples) in $\mathbb{R}^n$
    - With embedding functions to $\mathbb{R}^n$
  - **Connectivity**
    - Unit cells (pixels, voxels)
    - Implicitly encoded by the samples (space fill)
  - What's missing?
    - The **interpolation** scheme

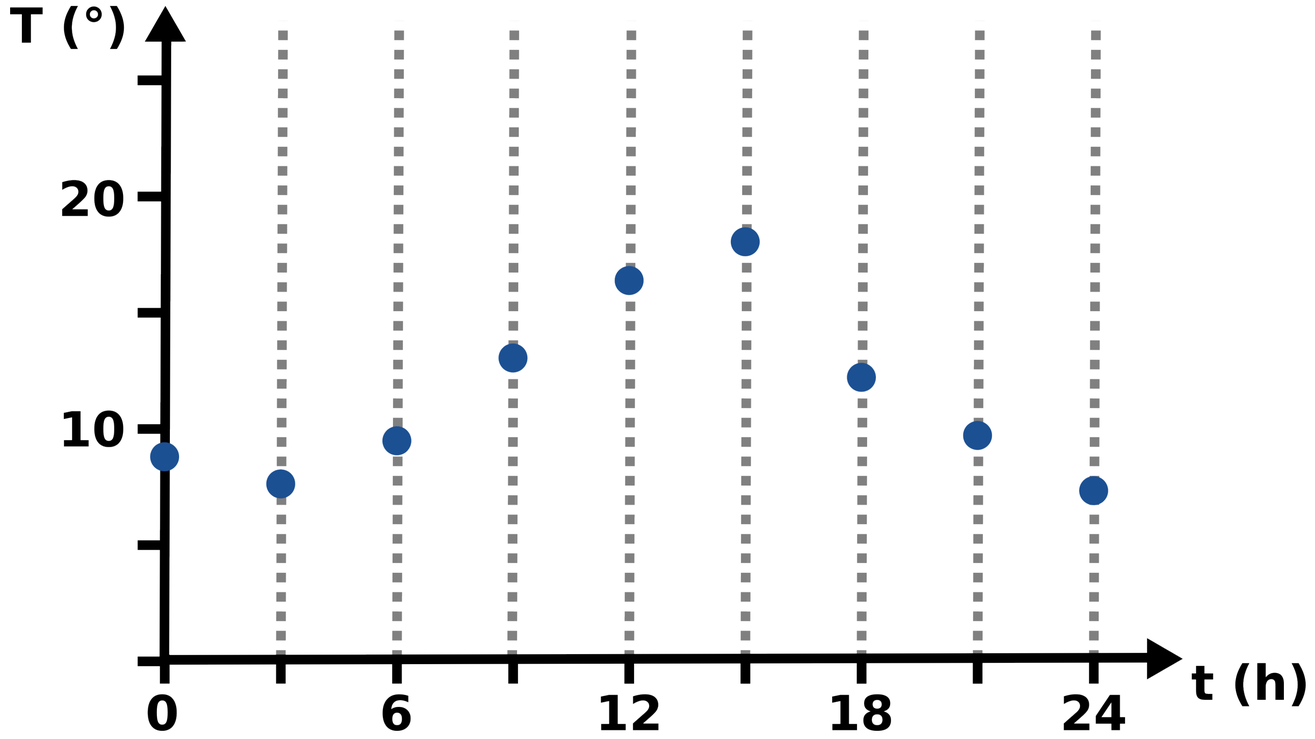






Interpolants for regular grids

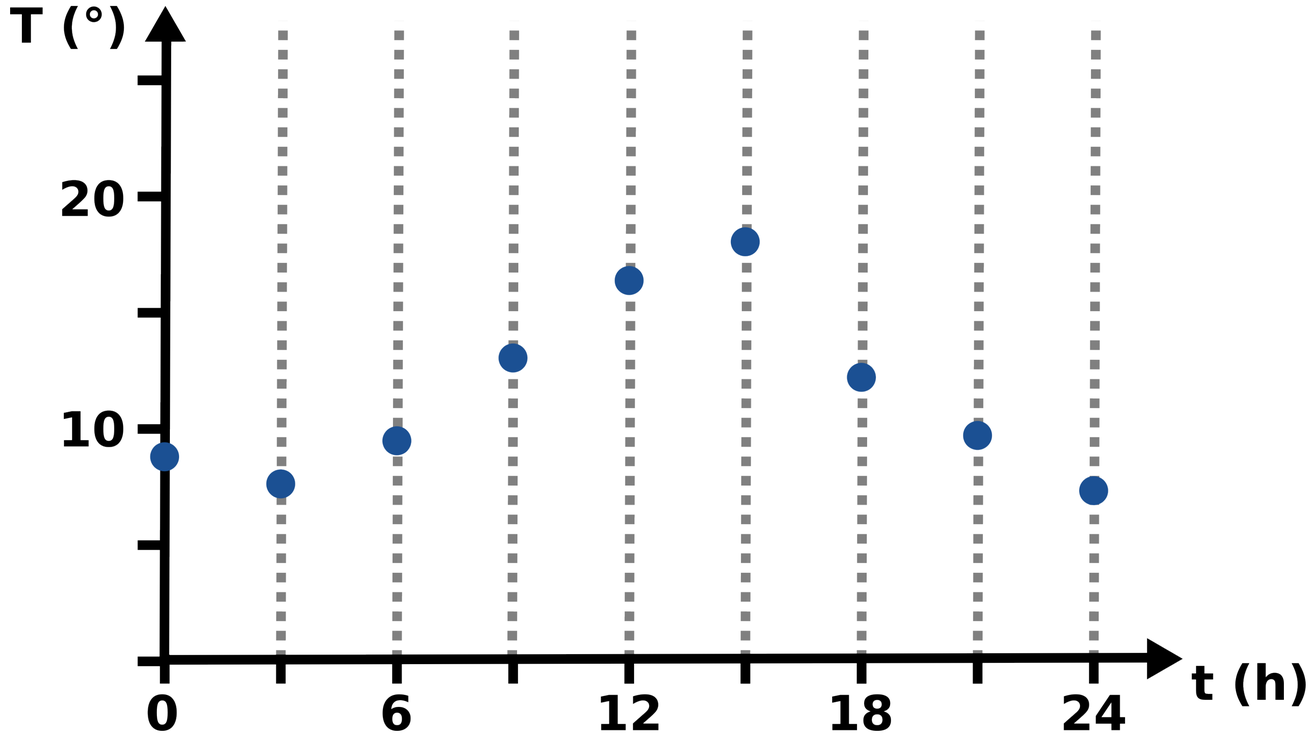
$$f : [0, 24] \to \mathbb{R}$$



Interpolants for regular grids

- Given values on the vertices

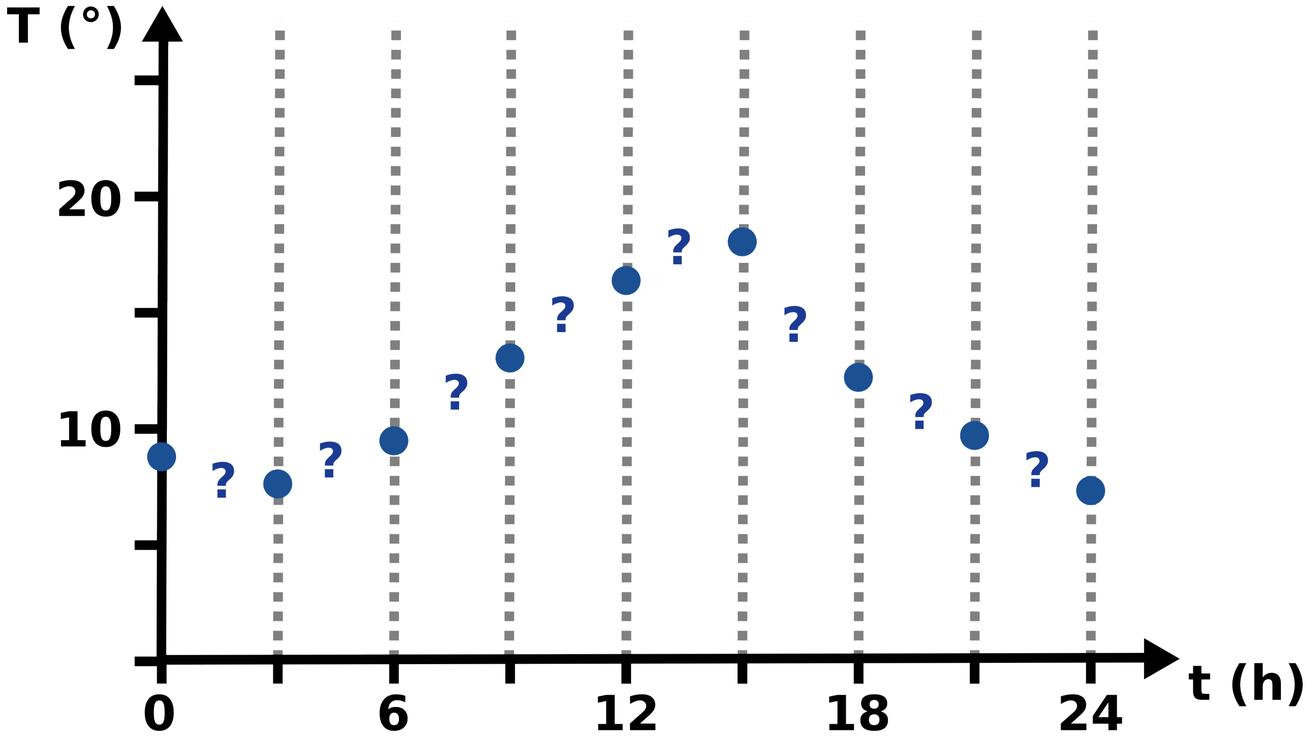
$$f : [0, 24] \to \mathbb{R}$$



Interpolants for regular grids

- Given values on the vertices
  - How can we deduce the values within the unit cells?

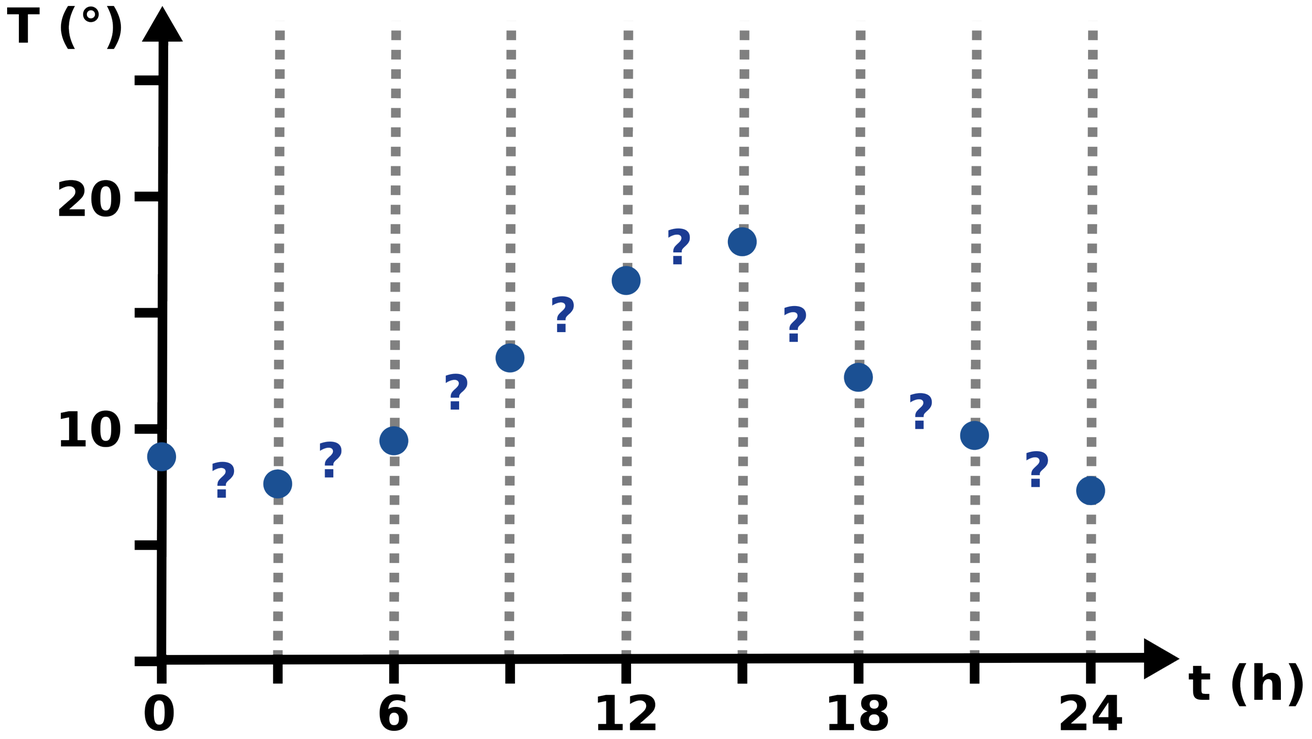
$$f : [0, 24] \to \mathbb{R}$$



Interpolants for regular grids

- Given values on the vertices
  - How can we deduce the values within the unit cells?
- For regular grids of $\mathbb{R}$
  - $\mathbb{I} : \ ]0,1[\ \rightarrow \ ]0,1[$

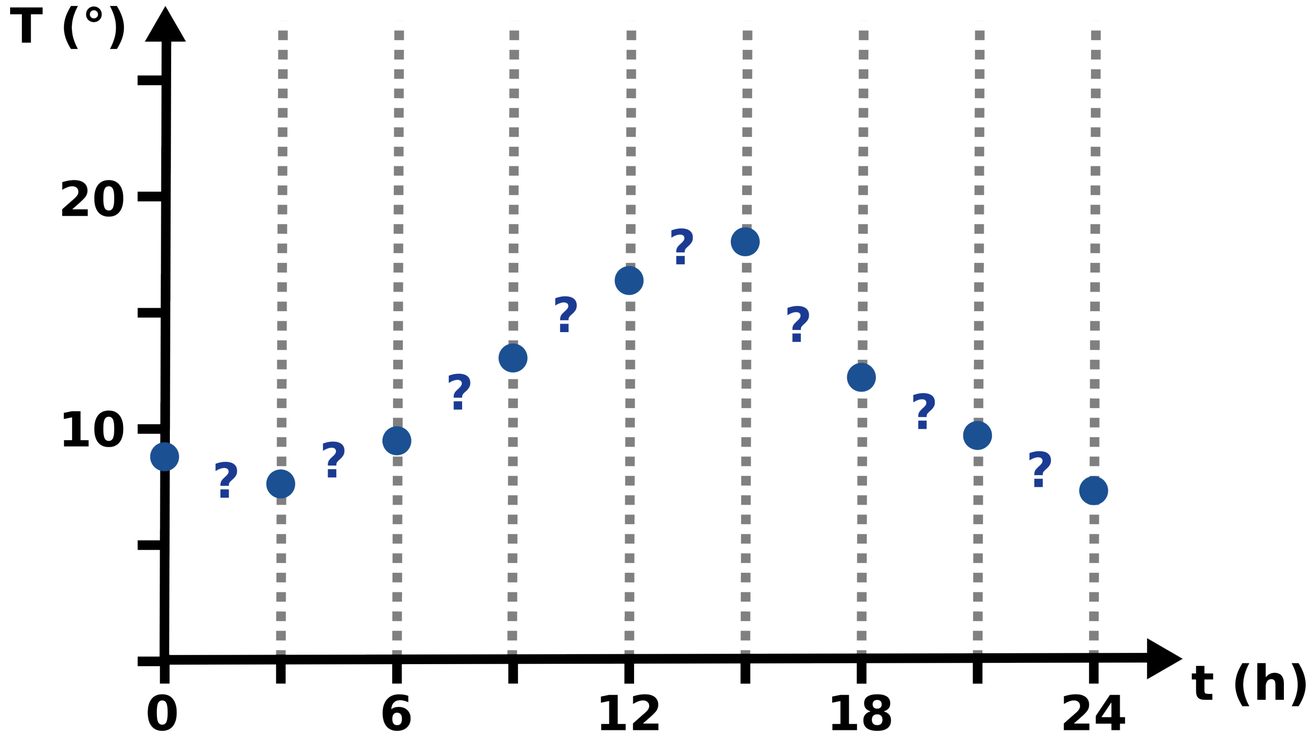
$$f : [0, 24] \rightarrow \mathbb{R}$$



Interpolants for regular grids

- Given values on the vertices
  - How can we deduce the values within the unit cells?
- For regular grids of $\mathbb{R}$
  - $\mathbb{I} : \ ]0,1[\ \rightarrow \ ]0,1[$
  - $lim_{x \rightarrow 0} \ \mathbb{I}(x) = 0$
  - $lim_{x \rightarrow 1} \ \mathbb{I}(x) = 1$

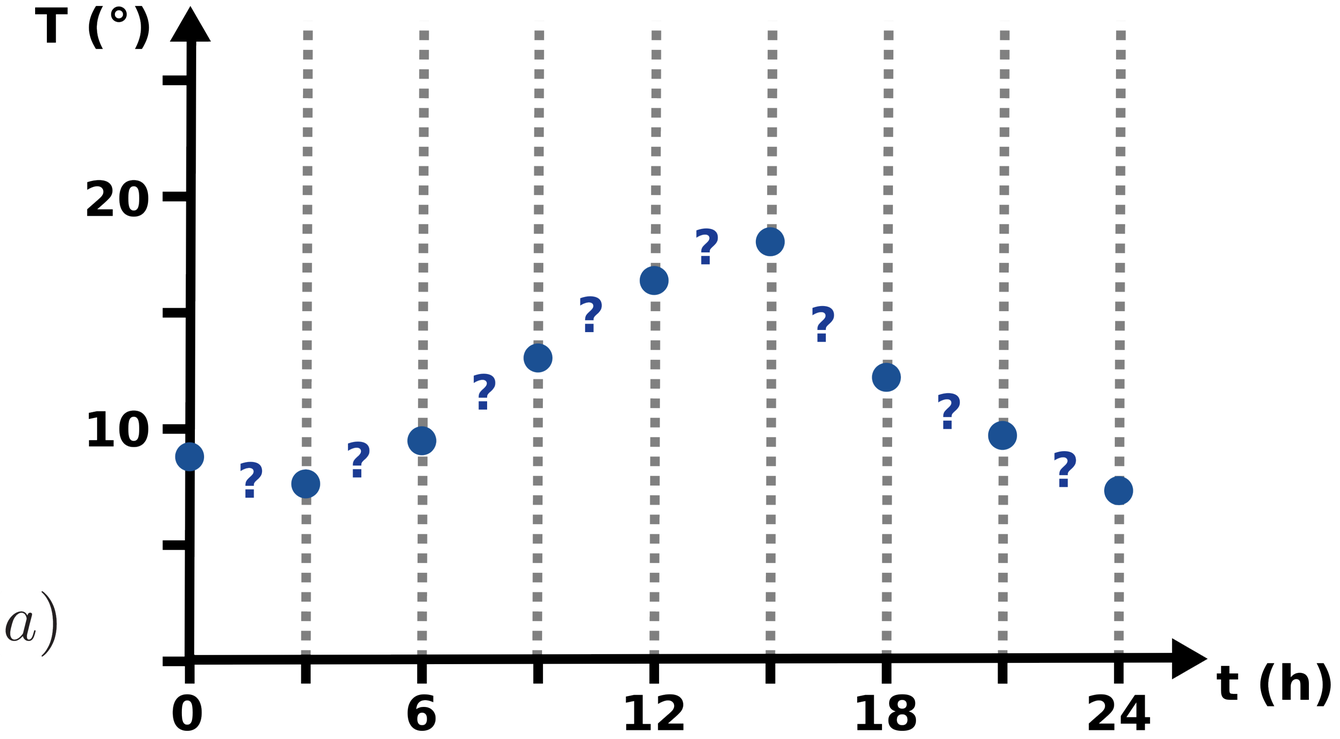
$$f : [0, 24] \rightarrow \mathbb{R}$$



Interpolants for regular grids

- Given values on the vertices
  - How can we deduce the values within the unit cells?
- For regular grids of $\mathbb{R}$
  - $\mathbb{I} : \ ]0,1[\ \rightarrow \ ]0,1[$
  - $lim_{x \rightarrow 0} \ \mathbb{I}(x) = 0$
  - $lim_{x \rightarrow 1} \ \mathbb{I}(x) = 1$
  - $\forall x \in [a,b]$
    - $f(x) = \mathbb{I}(\frac{x-a}{b-a}) \, (f(b) - f(a)) + f(a)$

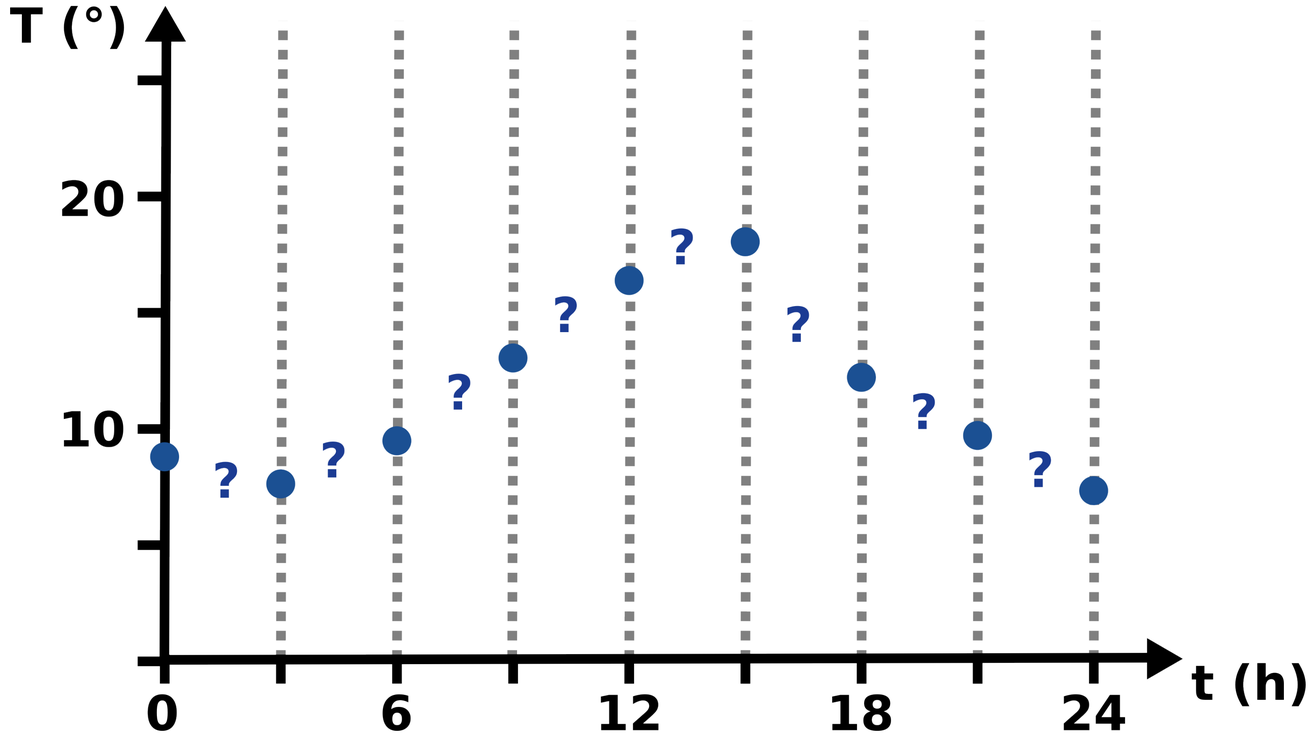
$$f : [0,24] \rightarrow \mathbb{R}$$



Interpolants for regular grids

- Interpolation function

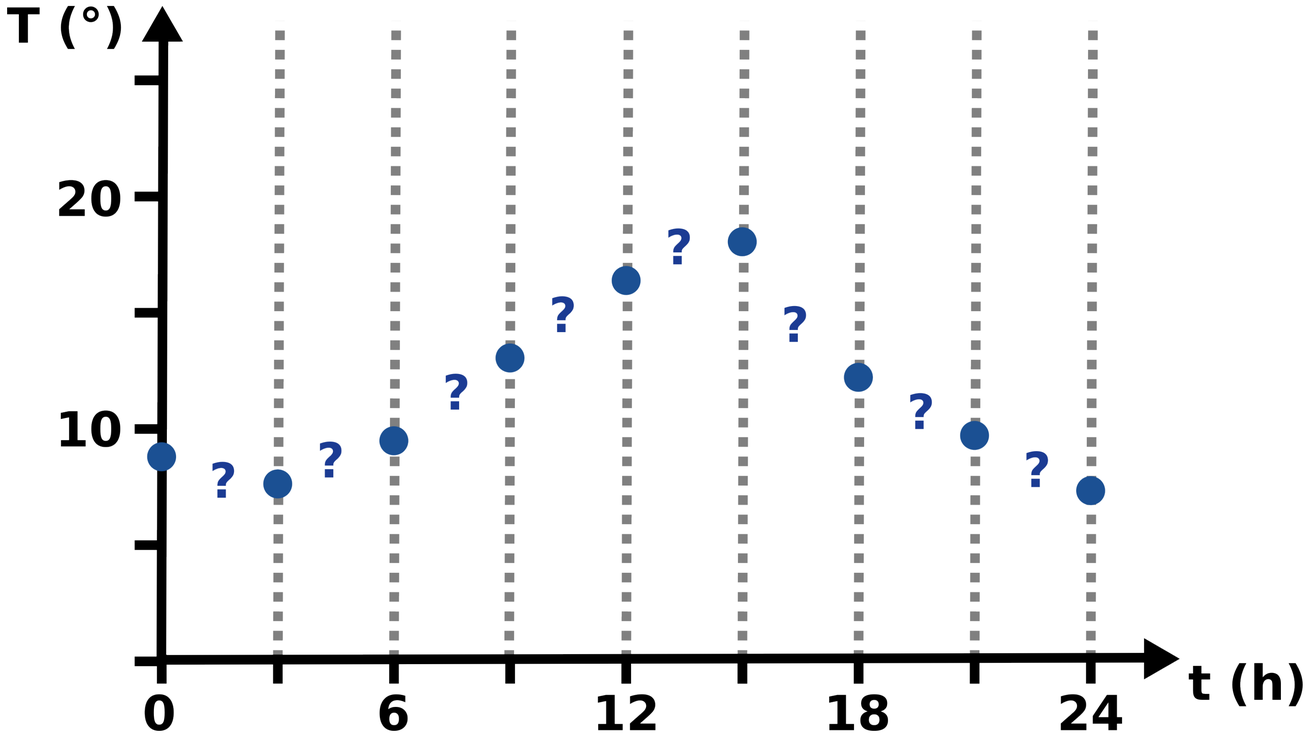
$$f : [0, 24] \to \mathbb{R}$$



Interpolants for regular grids

- Interpolation function
  - Given with the data

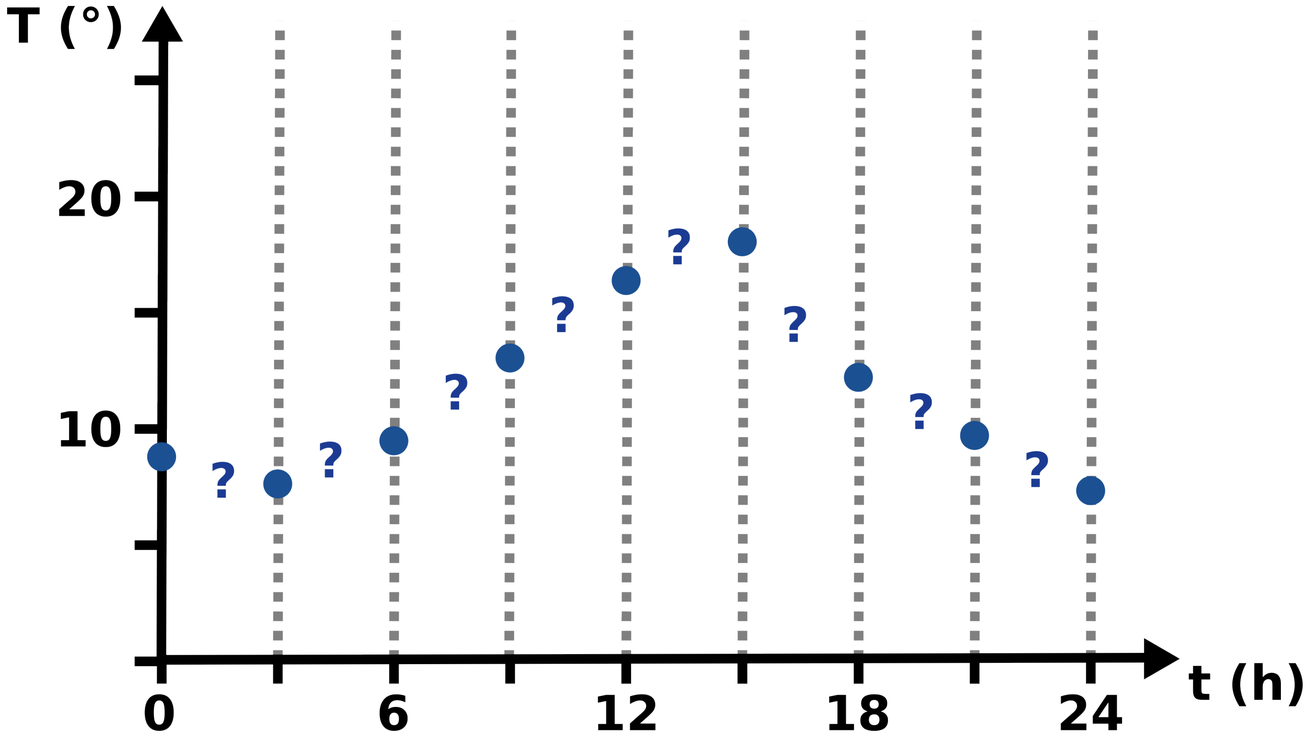
$$f : [0, 24] \to \mathbb{R}$$



Interpolants for regular grids

- Interpolation function
  - Given with the data
  - Can be defined per unit cell

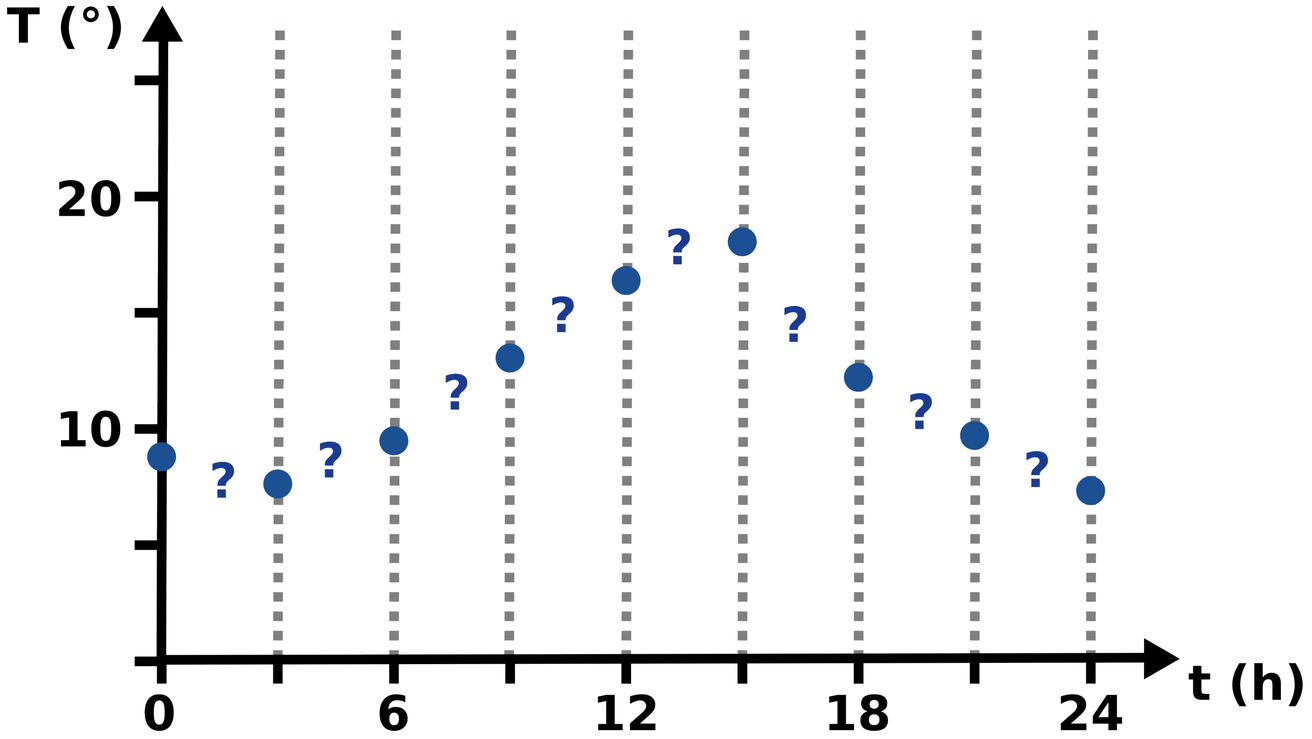
$$f : [0, 24] \to \mathbb{R}$$



Interpolants for regular grids

- Interpolation function
  - Given with the data
  - Can be defined per unit cell
  - Computed on demand

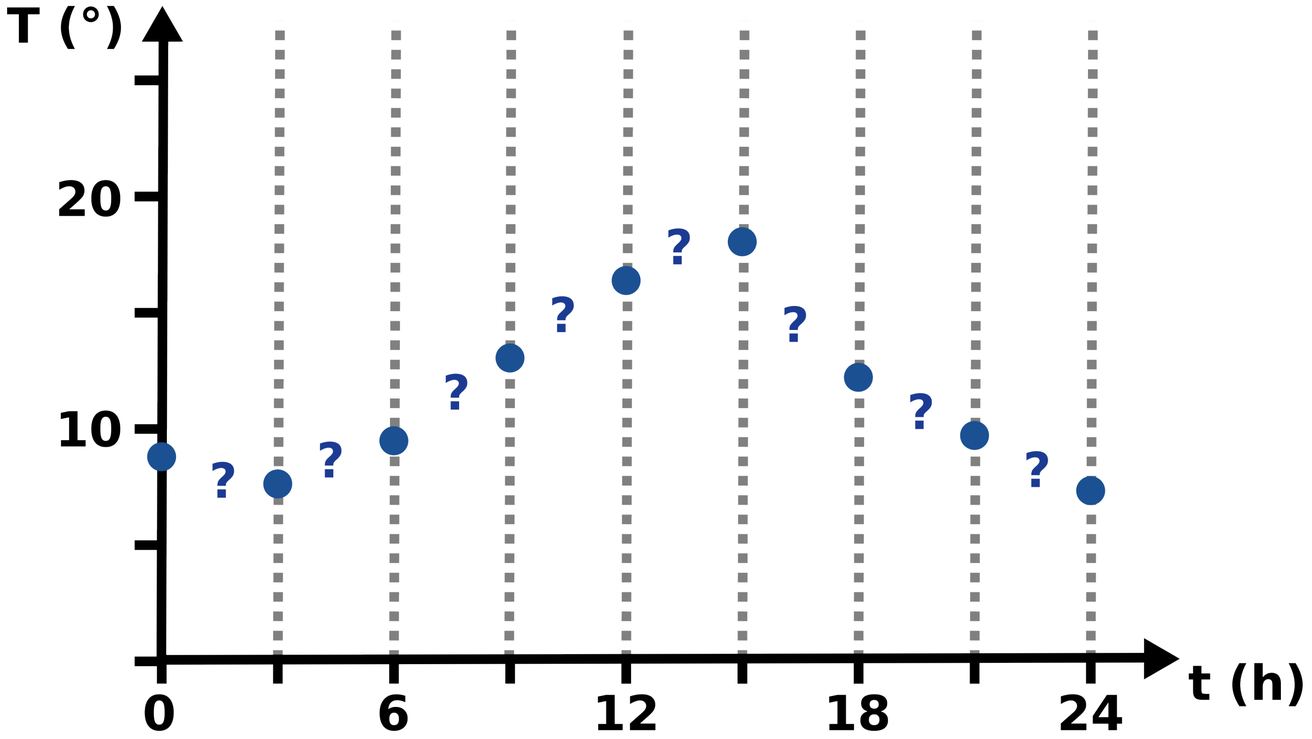
$$f : [0, 24] \rightarrow \mathbb{R}$$



Interpolants for regular grids

- Interpolation function
  - Given with the data
  - Can be defined per unit cell
  - Computed on demand
- Examples $\mathbb{I}: \ ]0,1[\ \rightarrow \ ]0,1[$

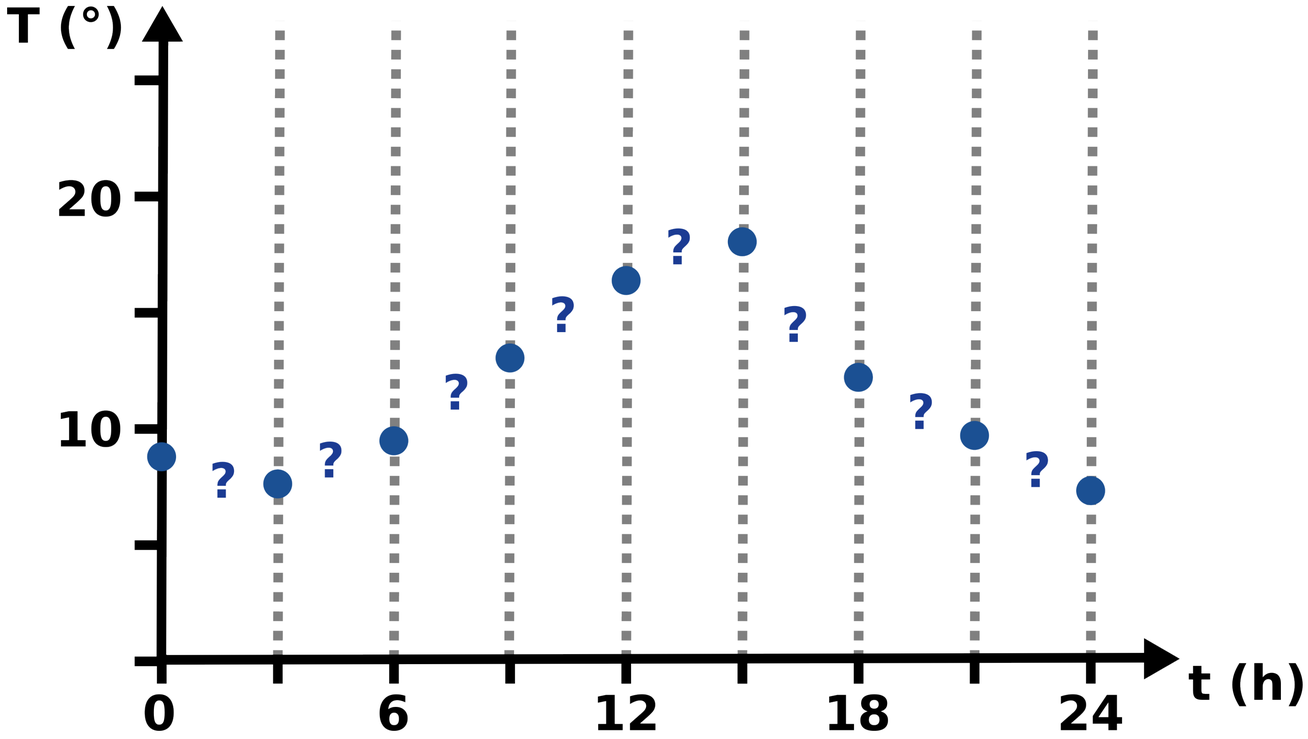
$$f : [0, 24] \rightarrow \mathbb{R}$$



Interpolants for regular grids

- Interpolation function
  - Given with the data
  - Can be defined per unit cell
  - Computed on demand
- Examples $\mathbb{I} : \ ]0,1[\ \rightarrow \ ]0,1[$
  - Piecewise constant
    - $\mathbb{I}(x) = \delta(x-1)$

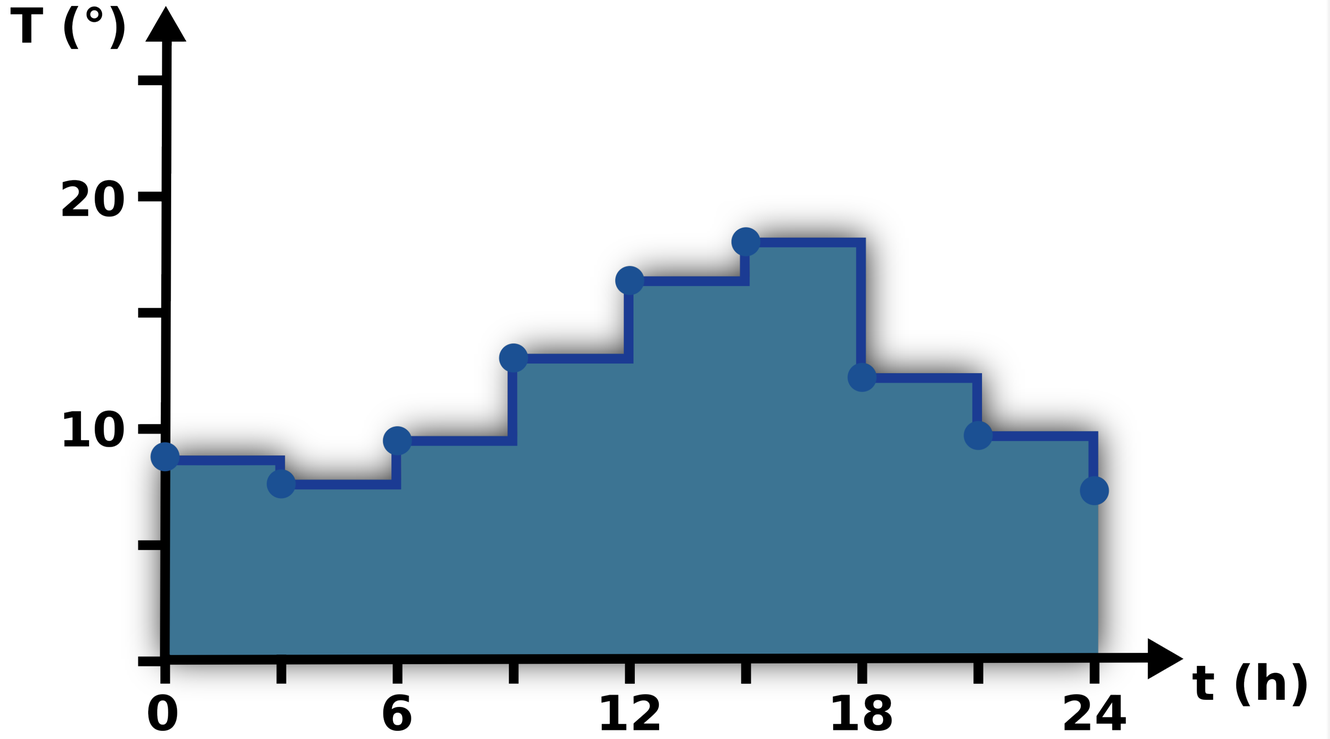
$$f : [0, 24] \rightarrow \mathbb{R}$$



Interpolants for regular grids

- Interpolation function
  - Given with the data
  - Can be defined per unit cell
  - Computed on demand
- Examples $\mathbb{I} : \ ]0,1[\ \rightarrow \ ]0,1[$
  - Piecewise constant
    - $\mathbb{I}(x) = \delta(x-1)$

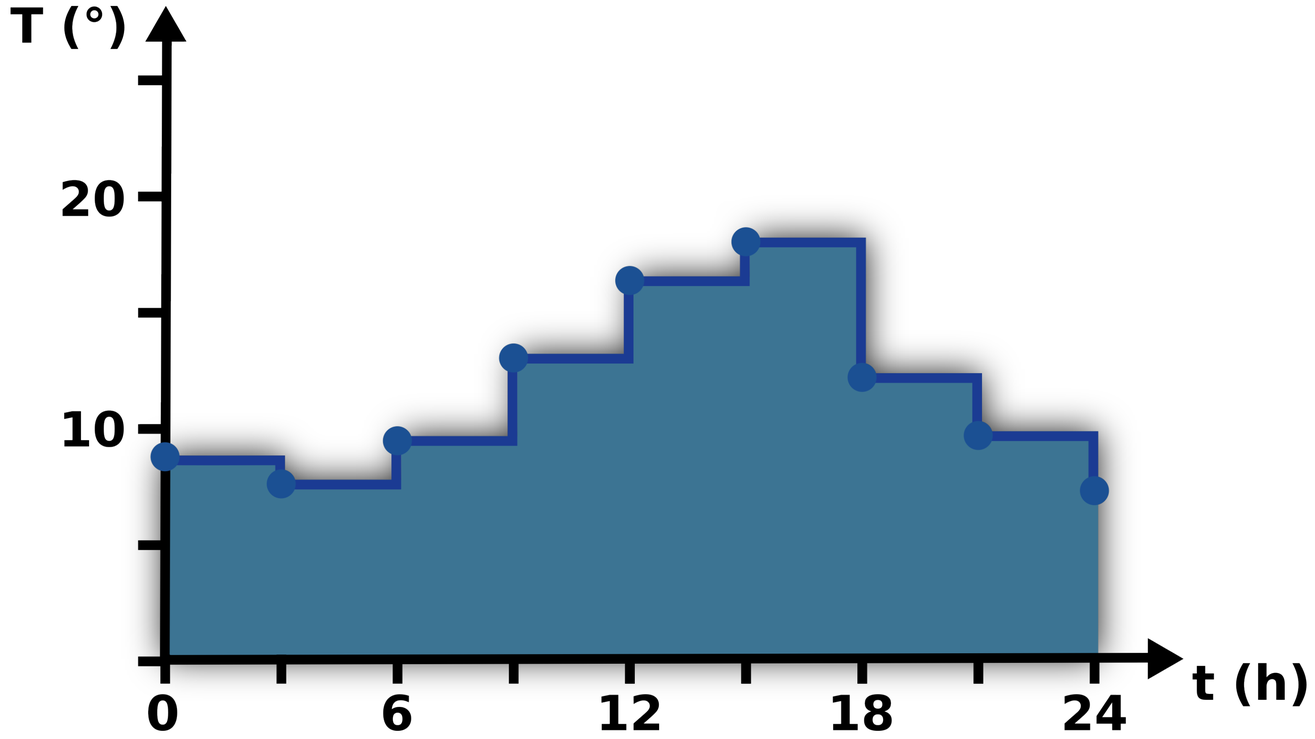
$$f : [0, 24] \rightarrow \mathbb{R}$$



Interpolants for regular grids

- Interpolation function
  - Given with the data
  - Can be defined per unit cell
  - Computed on demand
- Examples $\mathbb{I} : \ ]0,1[\ \rightarrow \ ]0,1[$
  - Piecewise constant
    - $\mathbb{I}(x) = \delta(x-1)$
  - Piecewise linear
    - $\mathbb{I}(x) = x$

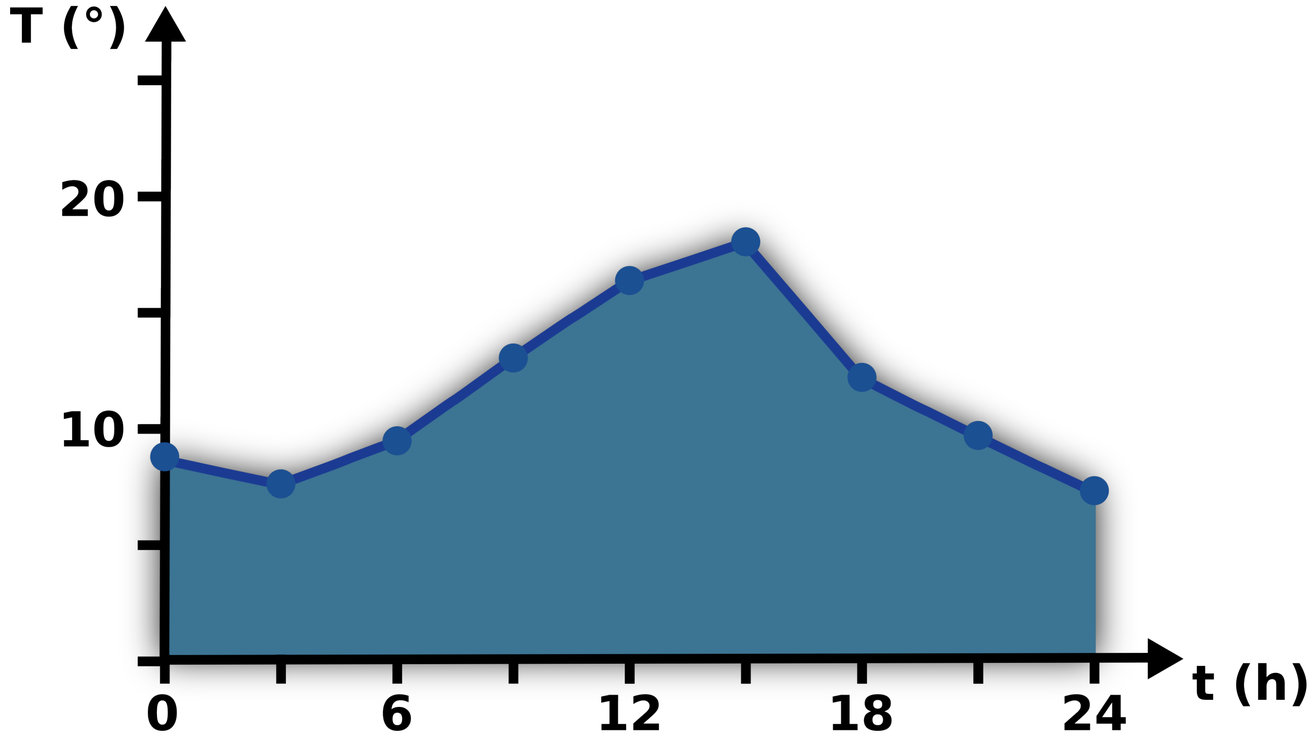
$$f : [0, 24] \rightarrow \mathbb{R}$$



Interpolants for regular grids

- Interpolation function
  - Given with the data
  - Can be defined per unit cell
  - Computed on demand
- Examples $\mathbb{I} : \ ]0,1[\ \rightarrow \ ]0,1[$
  - Piecewise constant
    - $\mathbb{I}(x) = \delta(x-1)$
  - Piecewise linear
    - $\mathbb{I}(x) = x$

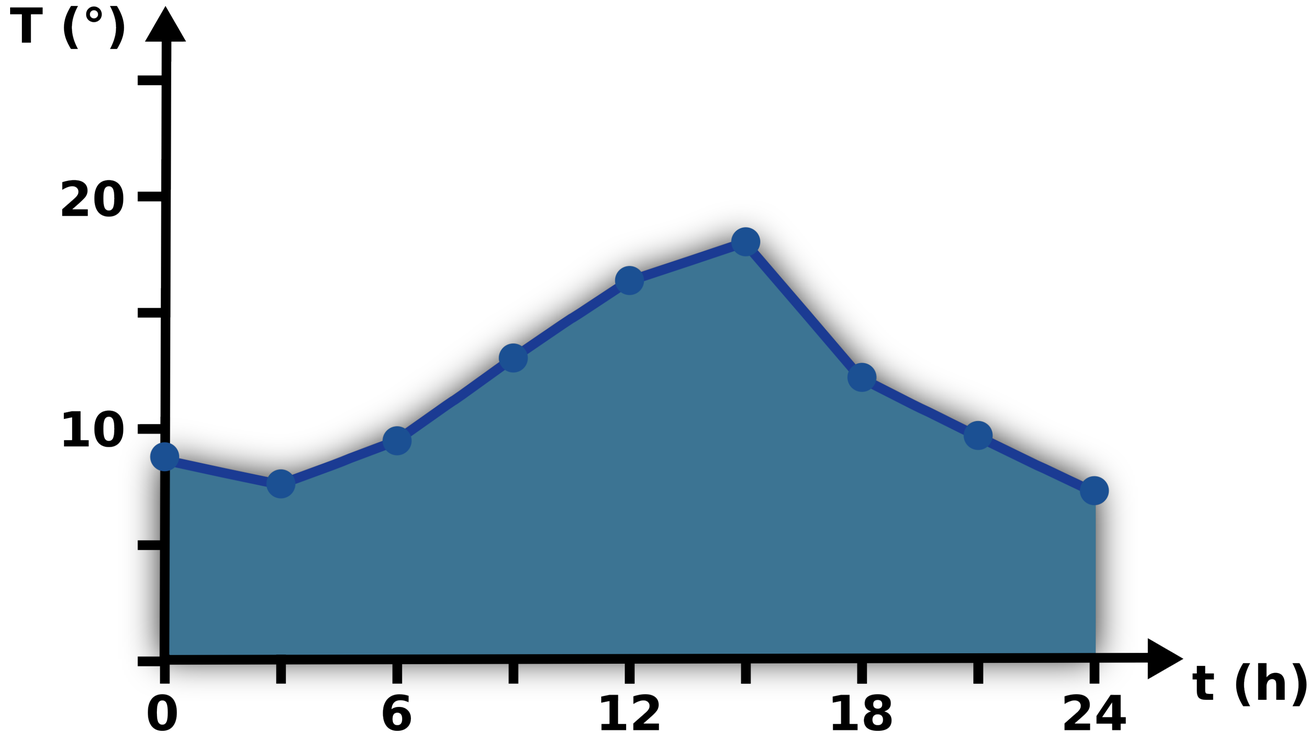
$$f : [0, 24] \rightarrow \mathbb{R}$$



Interpolants for regular grids

- Interpolation function
  - Given with the data
  - Can be defined per unit cell
  - Computed on demand
- Examples $\mathbb{I} : \ ]0,1[\ \rightarrow \ ]0,1[$
  - Piecewise constant
    - $\mathbb{I}(x) = \delta(x-1)$
  - Piecewise linear
    - $\mathbb{I}(x) = x$
  - Piecewise polynomials, etc.

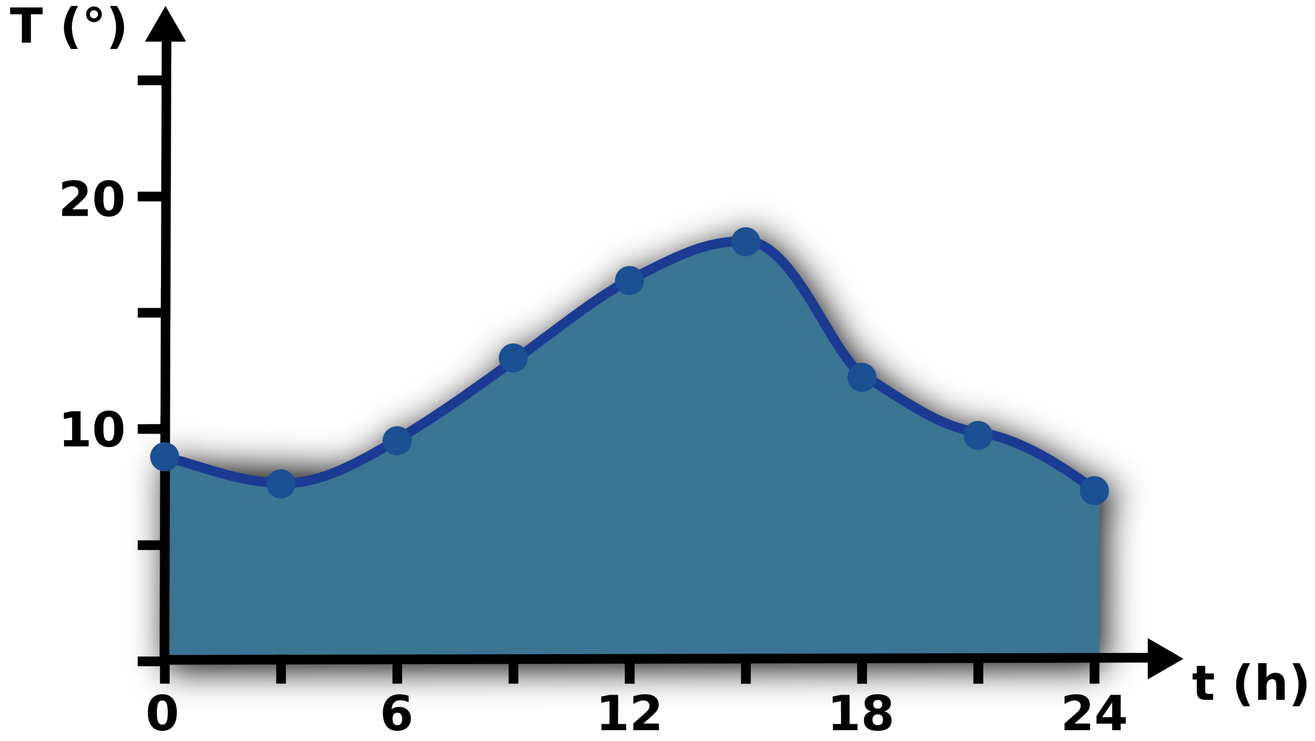
$$f : [0, 24] \rightarrow \mathbb{R}$$



Interpolants for regular grids

- Interpolation function
  - Given with the data
  - Can be defined per unit cell
  - Computed on demand
- Examples $\mathbb{I} : \ ]0,1[\ \rightarrow \ ]0,1[$
  - Piecewise constant
    - $\mathbb{I}(x) = \delta(x-1)$
  - Piecewise linear
    - $\mathbb{I}(x) = x$
  - Piecewise polynomials, etc.

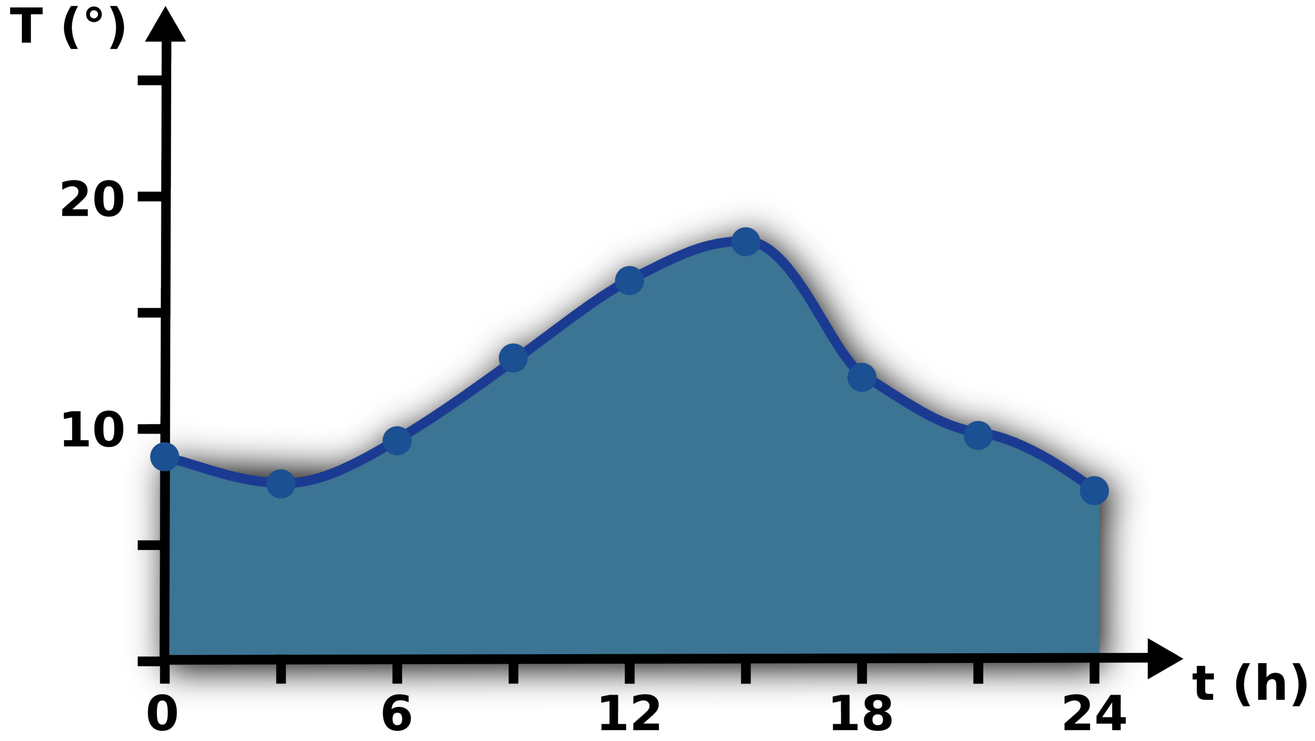
$$f : [0, 24] \rightarrow \mathbb{R}$$



Interpolants for regular grids

- Interpolation function
  - Given with the data
  - Can be defined per unit cell
  - Computed on demand
- Examples $\mathbb{I} : \ ]0,1[\ \rightarrow \ ]0,1[$
  - Piecewise constant
    - $\mathbb{I}(x) = \delta(x-1)$
  - **Piecewise linear**
    - $\mathbb{I}(x) = x$
  - Piecewise polynomials, etc.

$$f : [0, 24] \rightarrow \mathbb{R}$$



Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$

Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$

Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples

Interpolants for regular grids

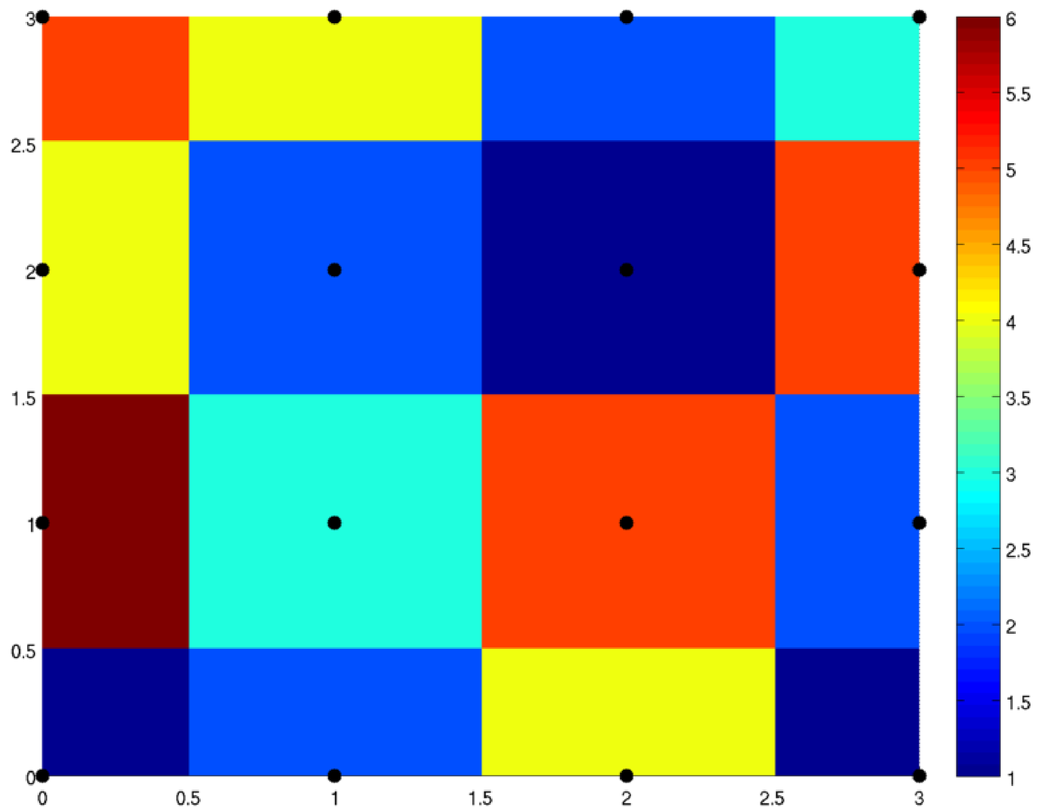
- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Piecewise constant
    - Nearest neighbor

Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Piecewise constant
    - Nearest neighbor
      - Value of the nearest vertex

Interpolants for regular grids

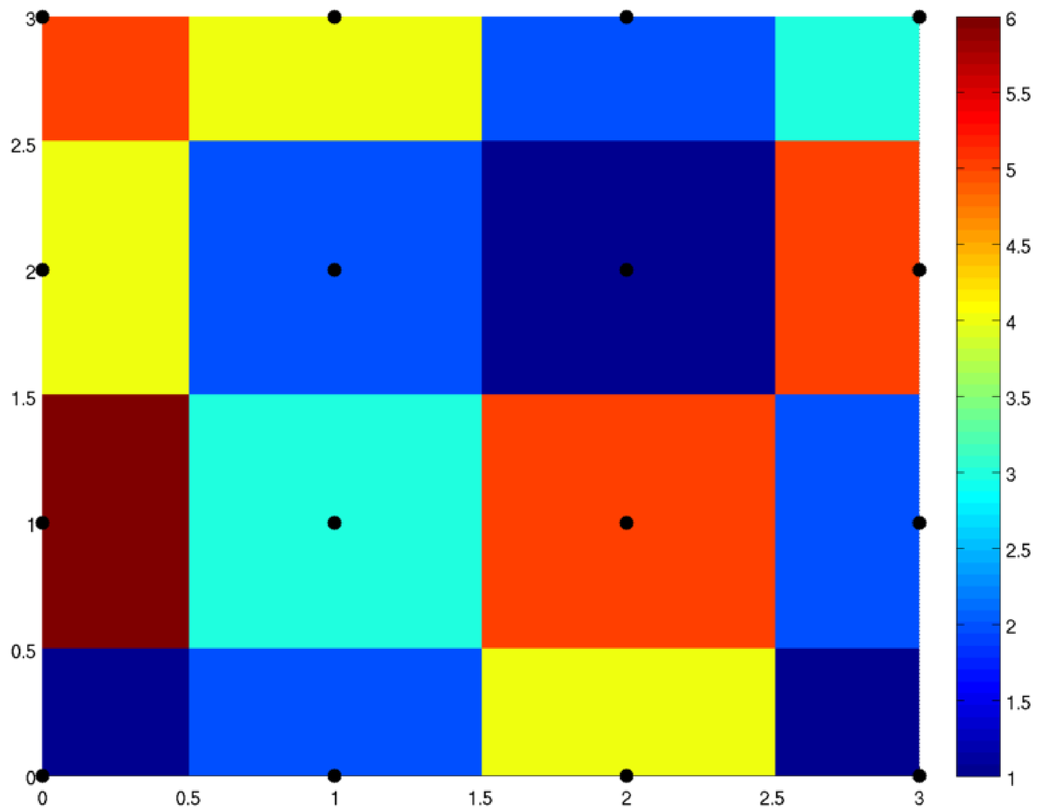
- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Piecewise constant
    - Nearest neighbor
      - Value of the nearest vertex



[Wikipedia]

Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Bilinear interpolation
      - One dimension at a time

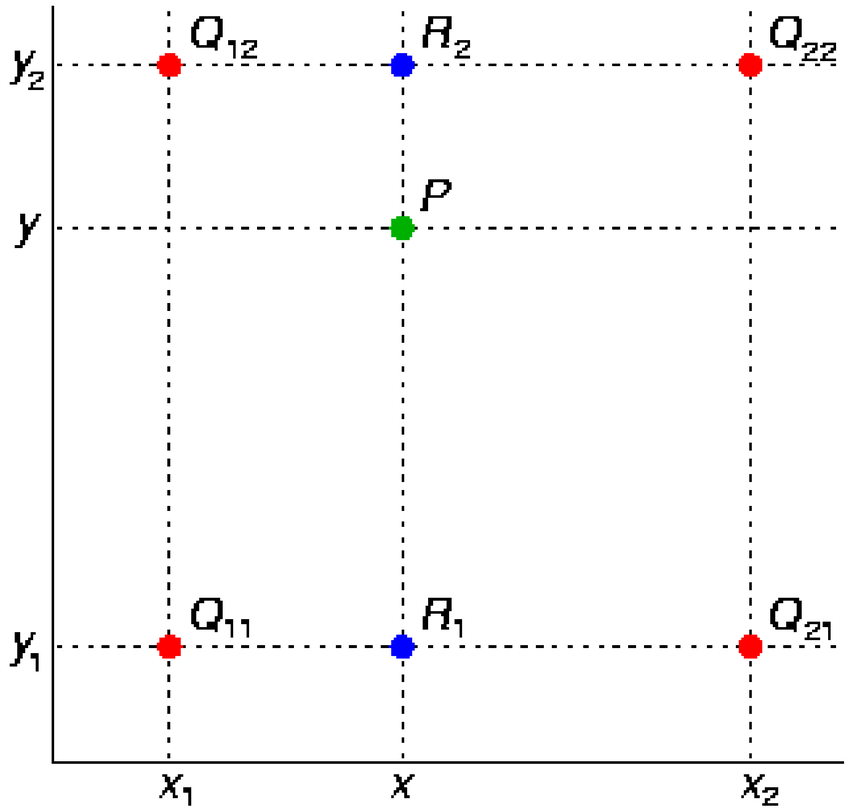


[Wikipedia]

Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Bilinear interpolation
      - One dimension at a time

$$f_{\mathbb{I}}(x, y_1) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_1) - f(x_1, y_1)) + f(x_1, y_1)$$



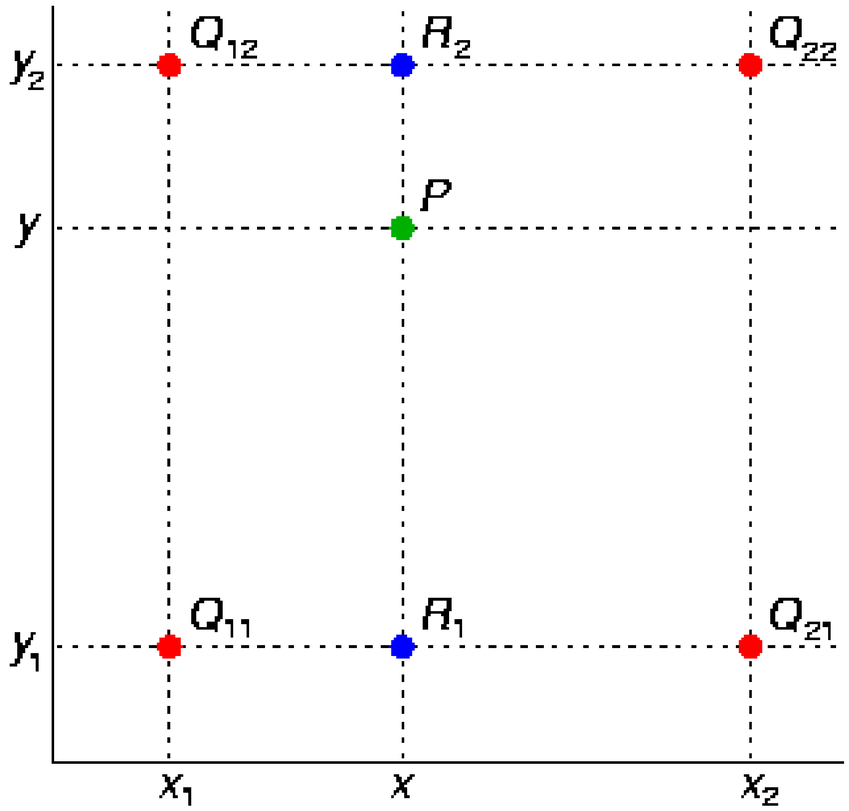
[Wikipedia]

Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Bilinear interpolation
      - One dimension at a time

$$f_{\mathbb{I}}(x, y_1) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_1) - f(x_1, y_1)) + f(x_1, y_1)$$

$$f_{\mathbb{I}}(x, y_2) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_2) - f(x_1, y_2)) + f(x_1, y_2)$$



[Wikipedia]

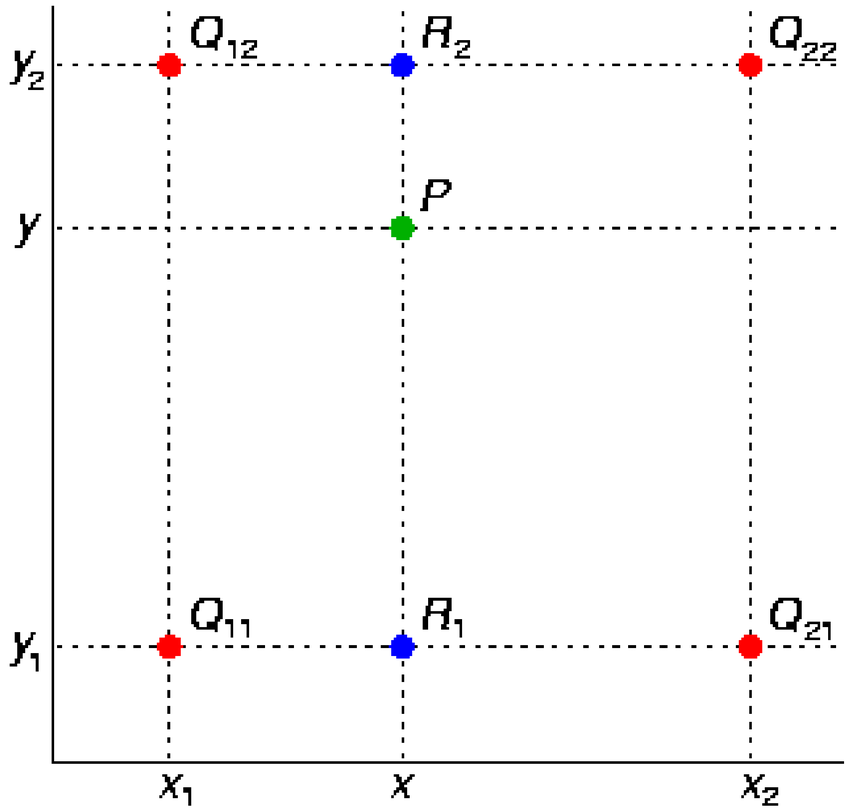
Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Bilinear interpolation
      - One dimension at a time

$$f_{\mathbb{I}}(x, y_1) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_1) - f(x_1, y_1)) + f(x_1, y_1)$$

$$f_{\mathbb{I}}(x, y_2) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_2) - f(x_1, y_2)) + f(x_1, y_2)$$

$$f_{\mathbb{I}}(x, y) = \frac{y - y_1}{y_2 - y_1}(f(x, y_1) - f(x, y_2)) + f(x, y_1)$$



[Wikipedia]

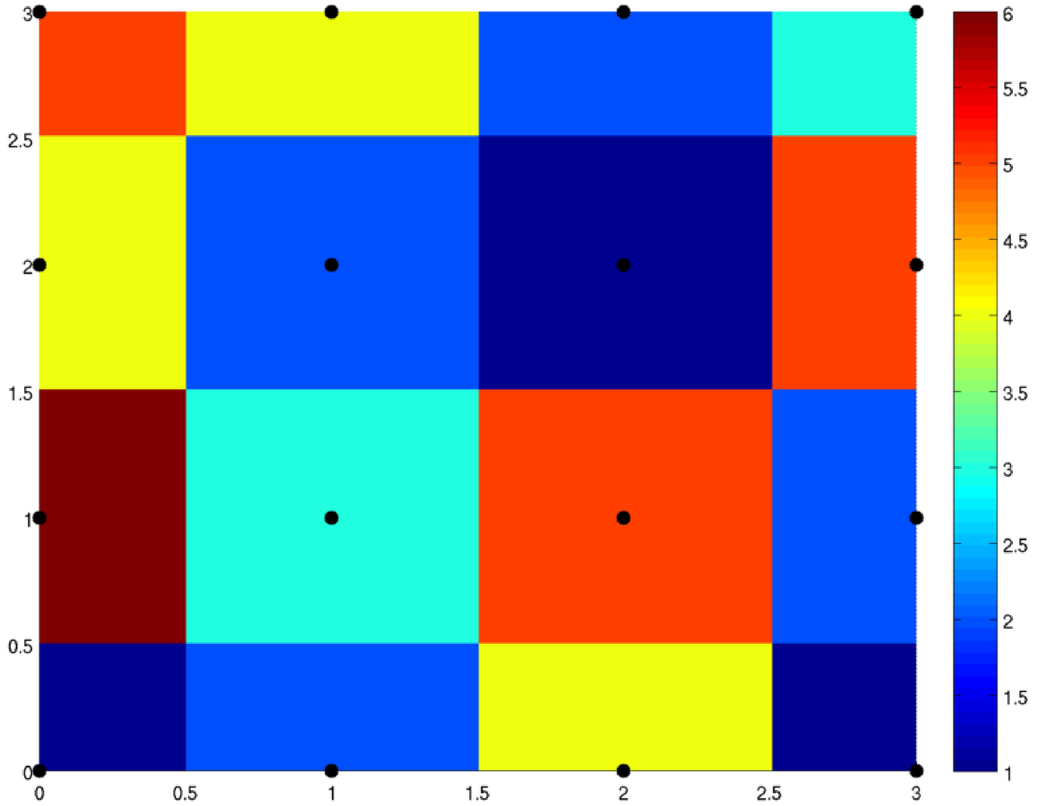
Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Bilinear interpolation
      - One dimension at a time

$$f_{\mathbb{I}}(x, y_1) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_1) - f(x_1, y_1)) + f(x_1, y_1)$$

$$f_{\mathbb{I}}(x, y_2) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_2) - f(x_1, y_2)) + f(x_1, y_2)$$

$$f_{\mathbb{I}}(x, y) = \frac{y - y_1}{y_2 - y_1}(f(x, y_1) - f(x, y_2)) + f(x, y_1)$$



[Wikipedia]

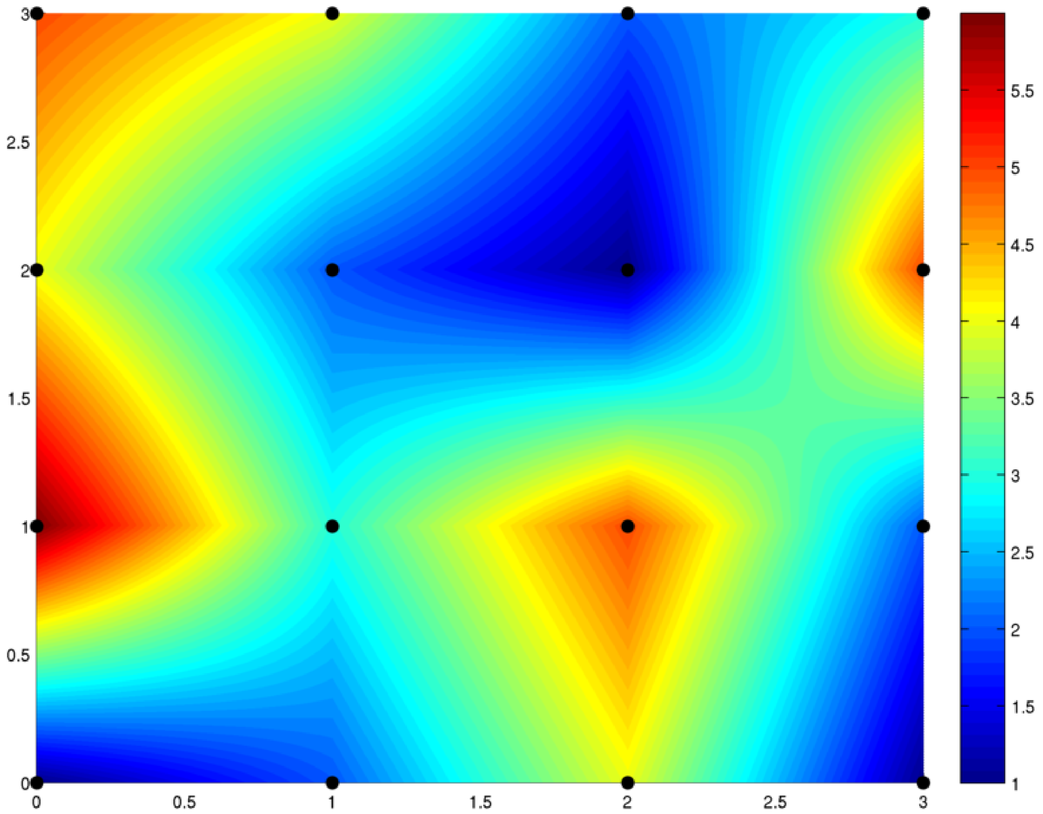
Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Bilinear interpolation
      - One dimension at a time

$$f_{\mathbb{I}}(x, y_1) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_1) - f(x_1, y_1)) + f(x_1, y_1)$$

$$f_{\mathbb{I}}(x, y_2) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_2) - f(x_1, y_2)) + f(x_1, y_2)$$

$$f_{\mathbb{I}}(x, y) = \frac{y - y_1}{y_2 - y_1}(f(x, y_1) - f(x, y_2)) + f(x, y_1)$$



[Wikipedia]

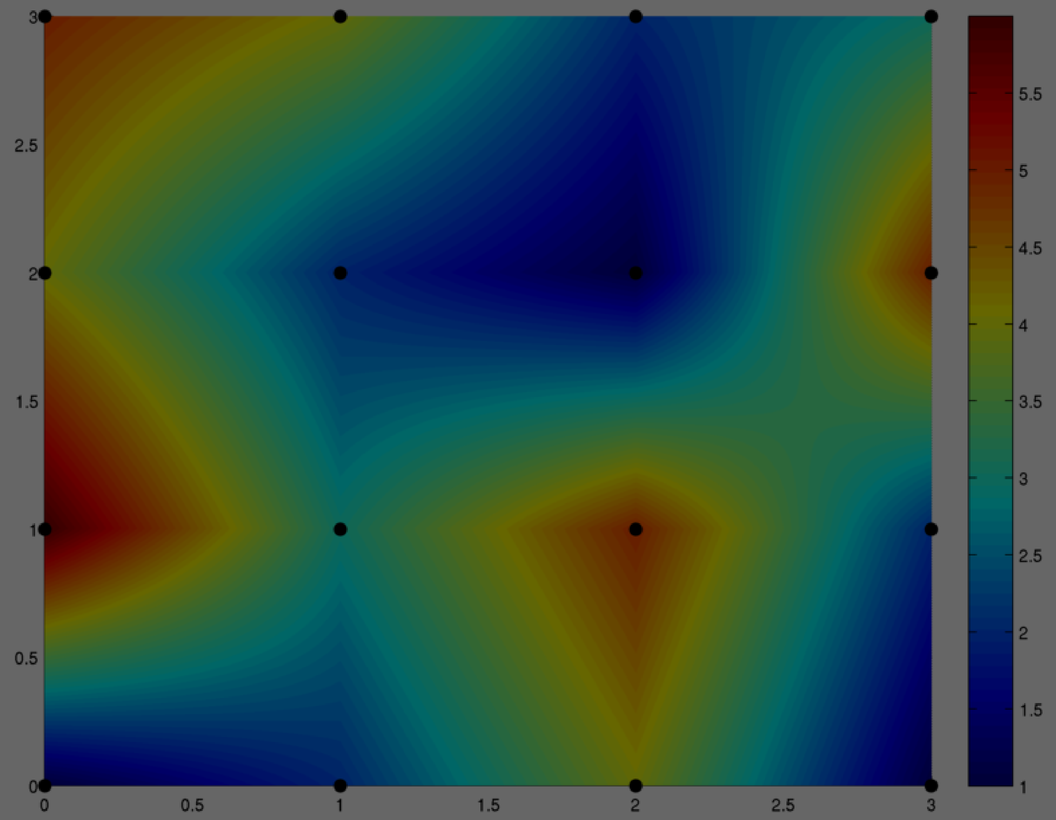
Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Bilinear interpolation
      – One dimension at a time

$$f_{\mathbb{I}}(x, y_1) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_1) - f(x_1, y_1)) + f(x_1, y_1)$$

$$f_{\mathbb{I}}(x, y_2) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_2) - f(x_1, y_2)) + f(x_1, y_2)$$

$$f_{\mathbb{I}}(x, y) = \frac{y - y_1}{y_2 - y_1}(f(x, y_1) - f(x, y_2)) + f(x, y_1)$$



[Wikipedia]

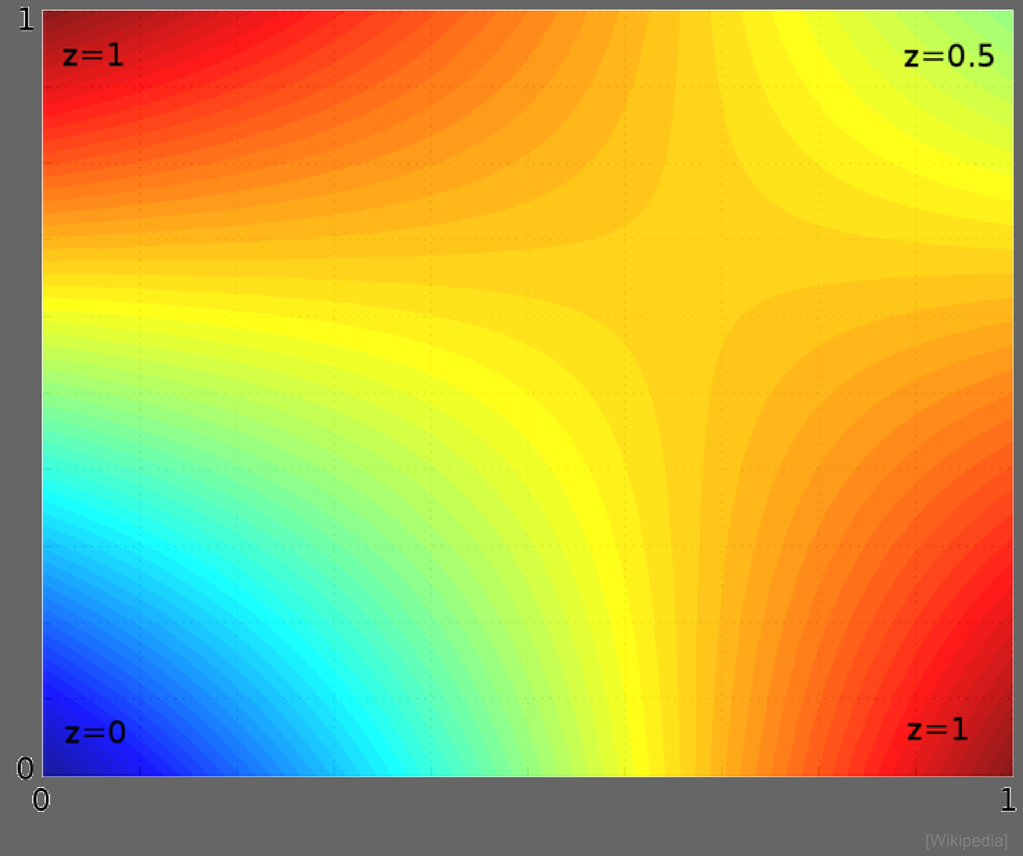
Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Bilinear interpolation
      – One dimension at a time

$$f_{\mathbb{I}}(x, y_1) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_1) - f(x_1, y_1)) + f(x_1, y_1)$$

$$f_{\mathbb{I}}(x, y_2) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_2) - f(x_1, y_2)) + f(x_1, y_2)$$

$$f_{\mathbb{I}}(x, y) = \frac{y - y_1}{y_2 - y_1}(f(x, y_1) - f(x, y_2)) + f(x, y_1)$$



[Wikipedia]

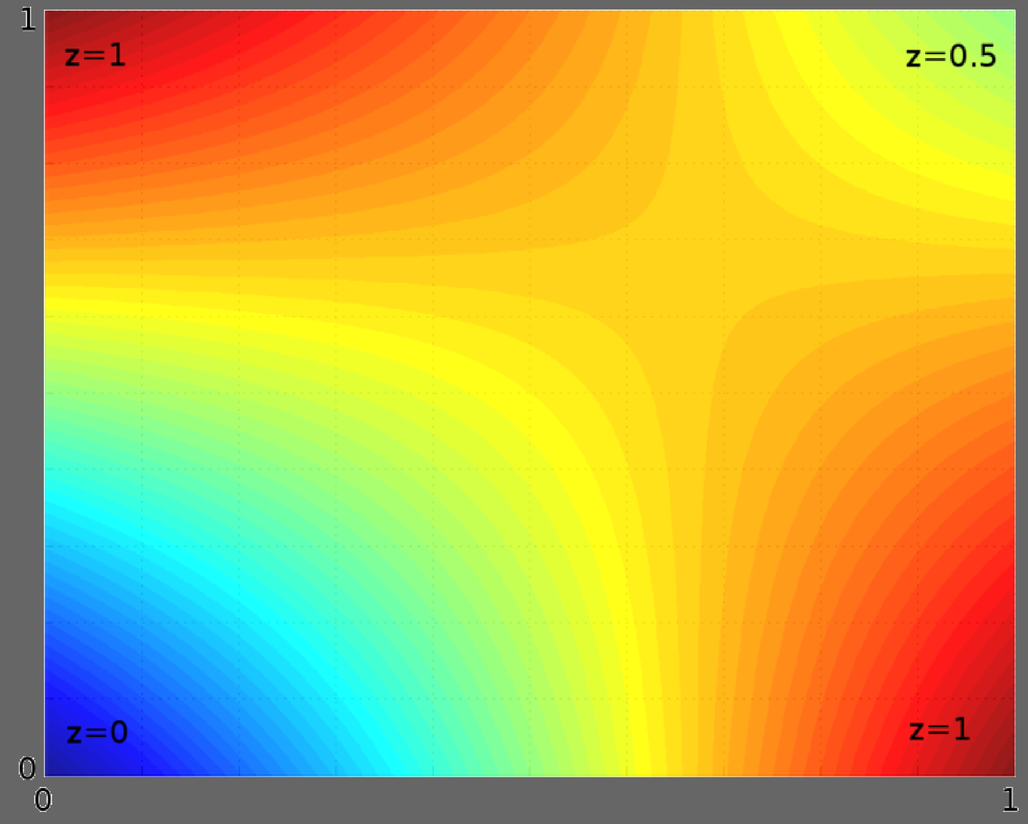
Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - Problem

$$\frac{x_2 - x_1}{} (x_1, y_1)$$

$$f_{\mathbb{I}}(x, y_2) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_2) - f(x_1, y_2)) + f(x_1, y_2)$$

$$f_{\mathbb{I}}(x, y) = \frac{y - y_1}{y_2 - y_1}(f(x, y_1) - f(x, y_2)) + f(x, y_1)$$



[Wikipedia]

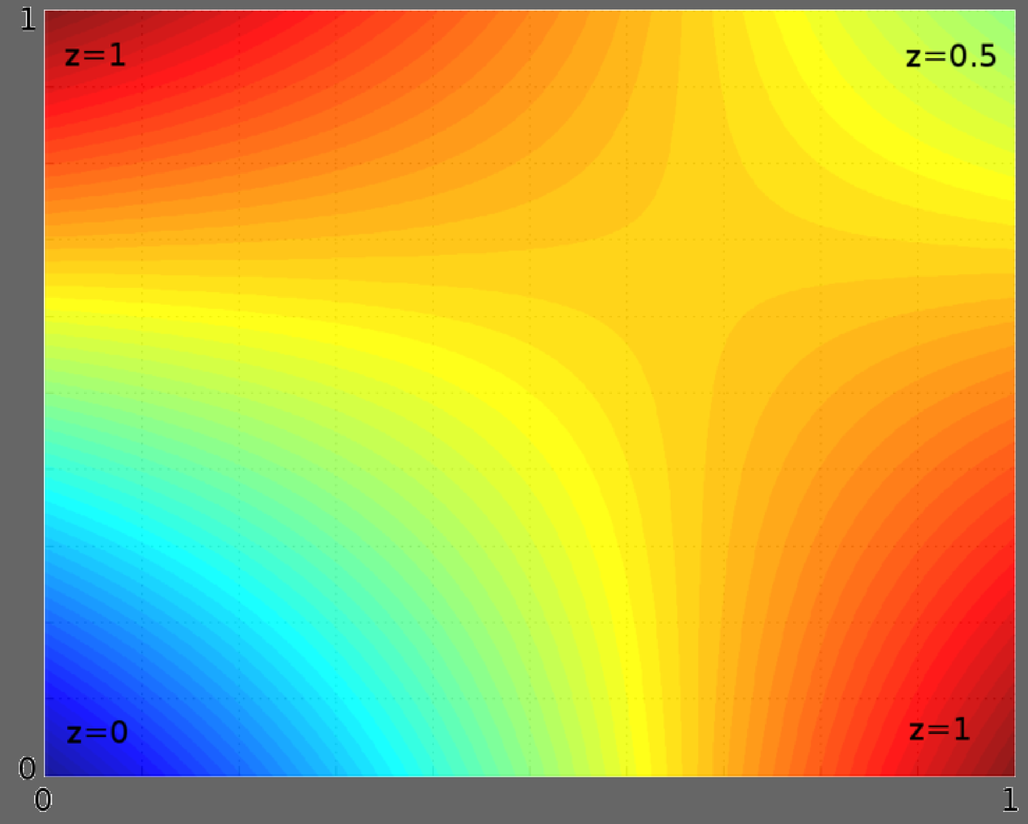
Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - Problem
    - Critical points in the interior

$$\frac{}{x_2 - x_1}(x_1, y_1)$$

$$f_{\mathbb{I}}(x, y_2) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_2) - f(x_1, y_2)) + f(x_1, y_2)$$

$$f_{\mathbb{I}}(x, y) = \frac{y - y_1}{y_2 - y_1}(f(x, y_1) - f(x, y_2)) + f(x, y_1)$$



[Wikipedia]

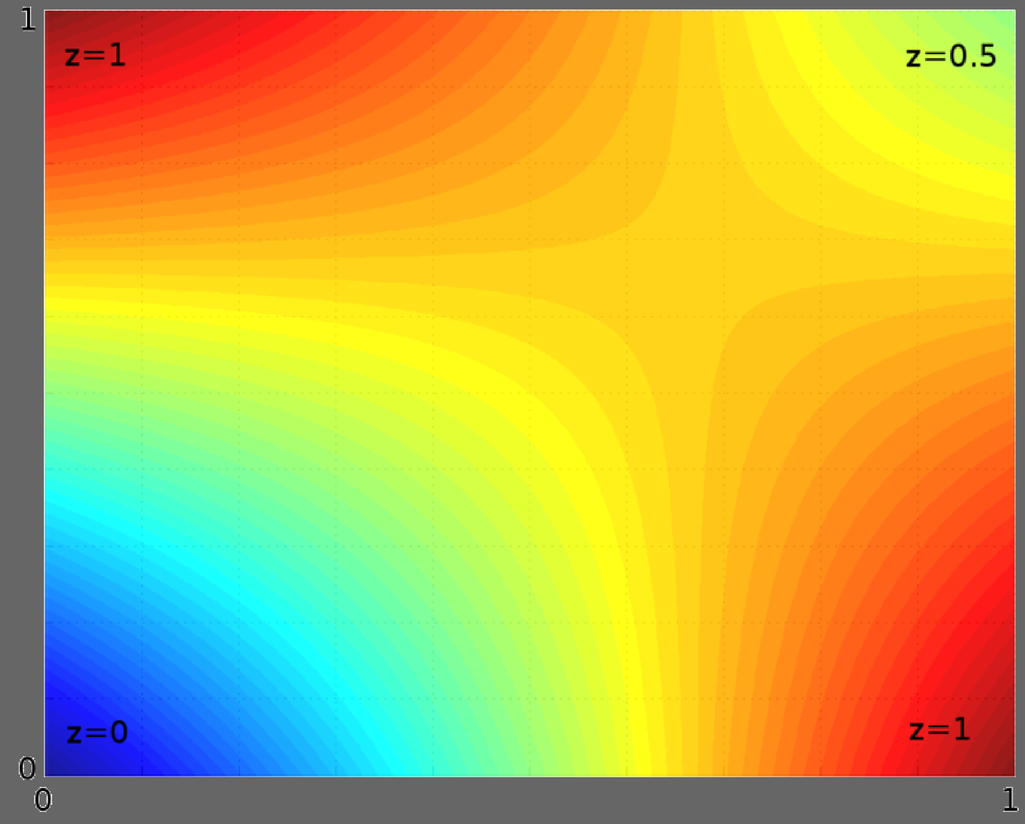
Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - Problem
    - Critical points in the interior
    - Source of many ambiguities in visualization

$$\frac{x_2 - x_1}{} (x_1, y_1)$$

$$f_{\mathbb{I}}(x, y_2) = \frac{x - x_1}{x_2 - x_1}(f(x_2, y_2) - f(x_1, y_2)) + f(x_1, y_2)$$

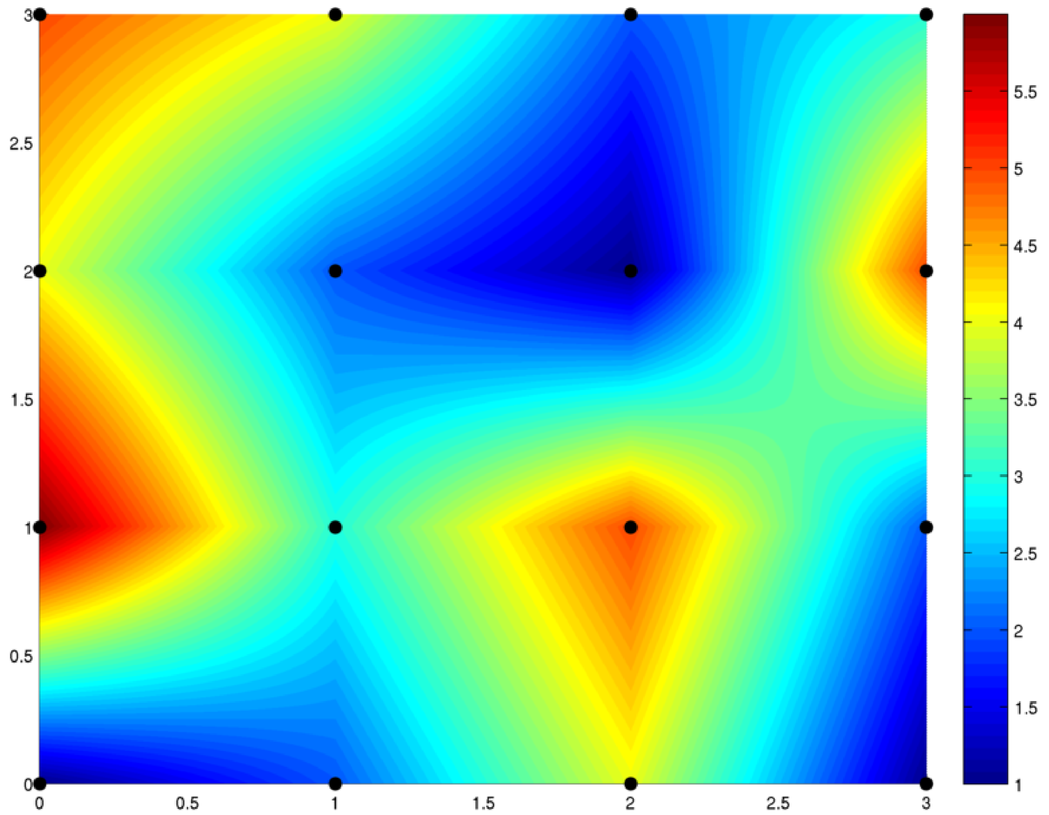
$$f_{\mathbb{I}}(x, y) = \frac{y - y_1}{y_2 - y_1}(f(x, y_1) - f(x, y_2)) + f(x, y_1)$$



[Wikipedia]

Interpolants for regular grids

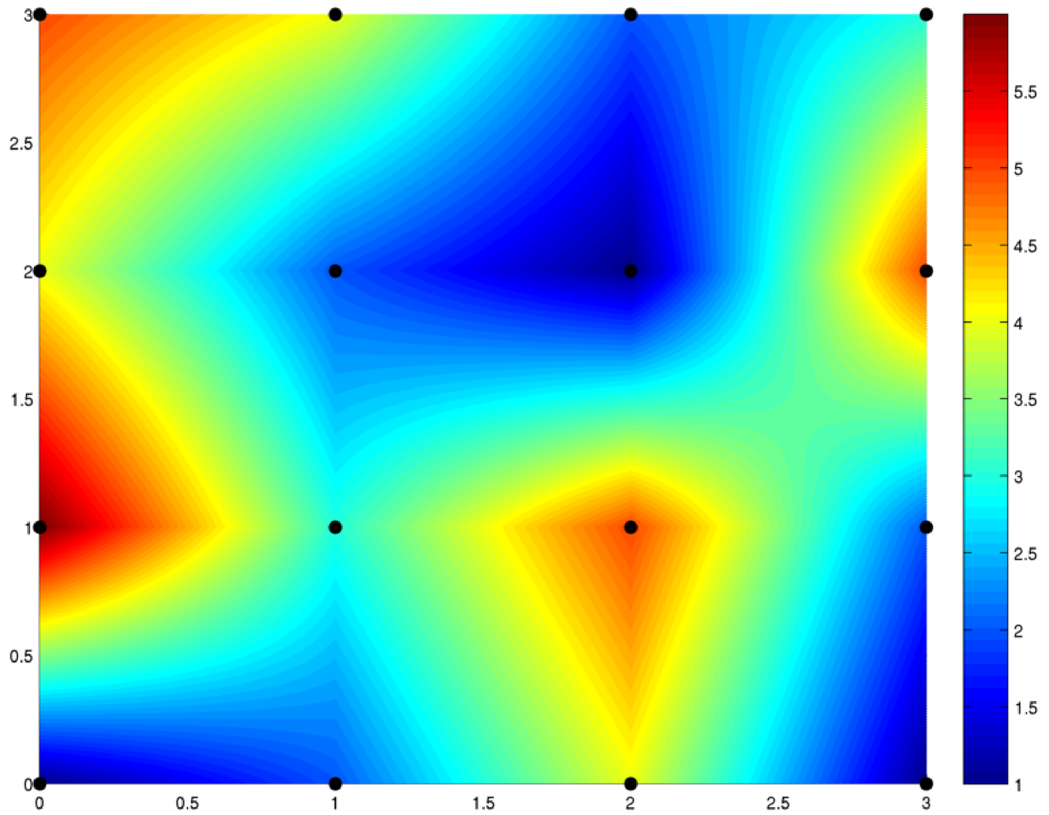
- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Bicubic interpolation



[Wikipedia]

Interpolants for regular grids

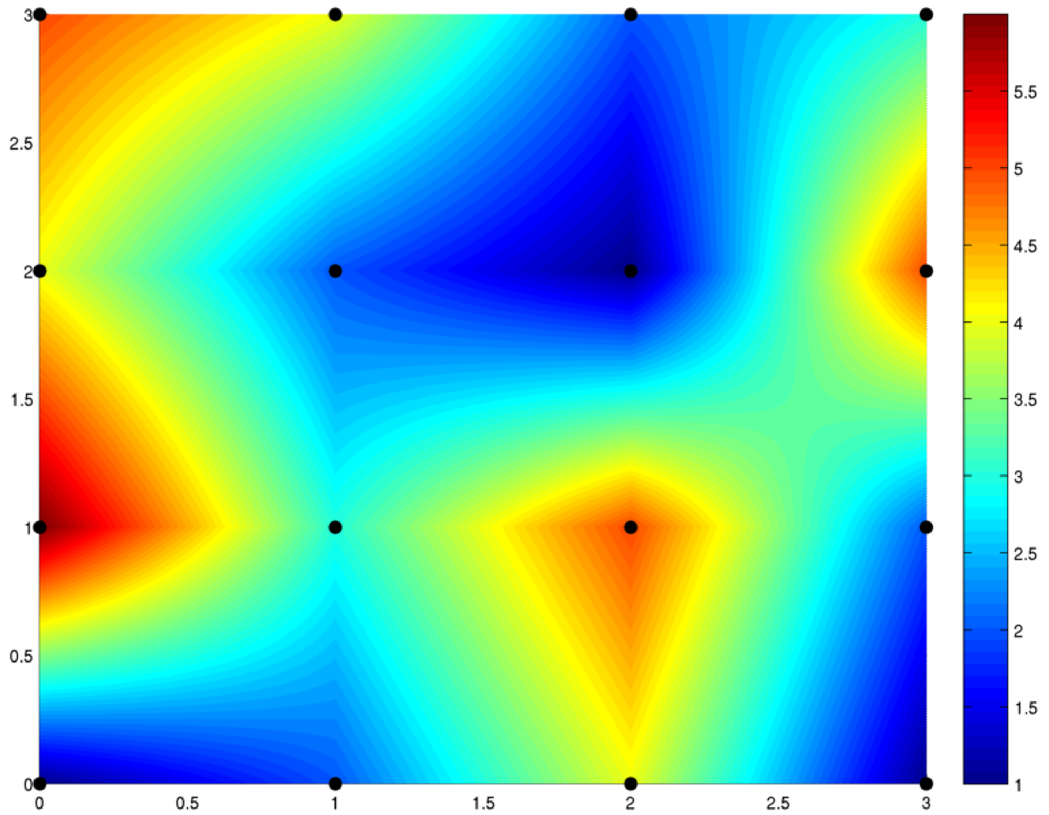
- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Bicubic interpolation
      - Piecewise polynomial



[Wikipedia]

Interpolants for regular grids

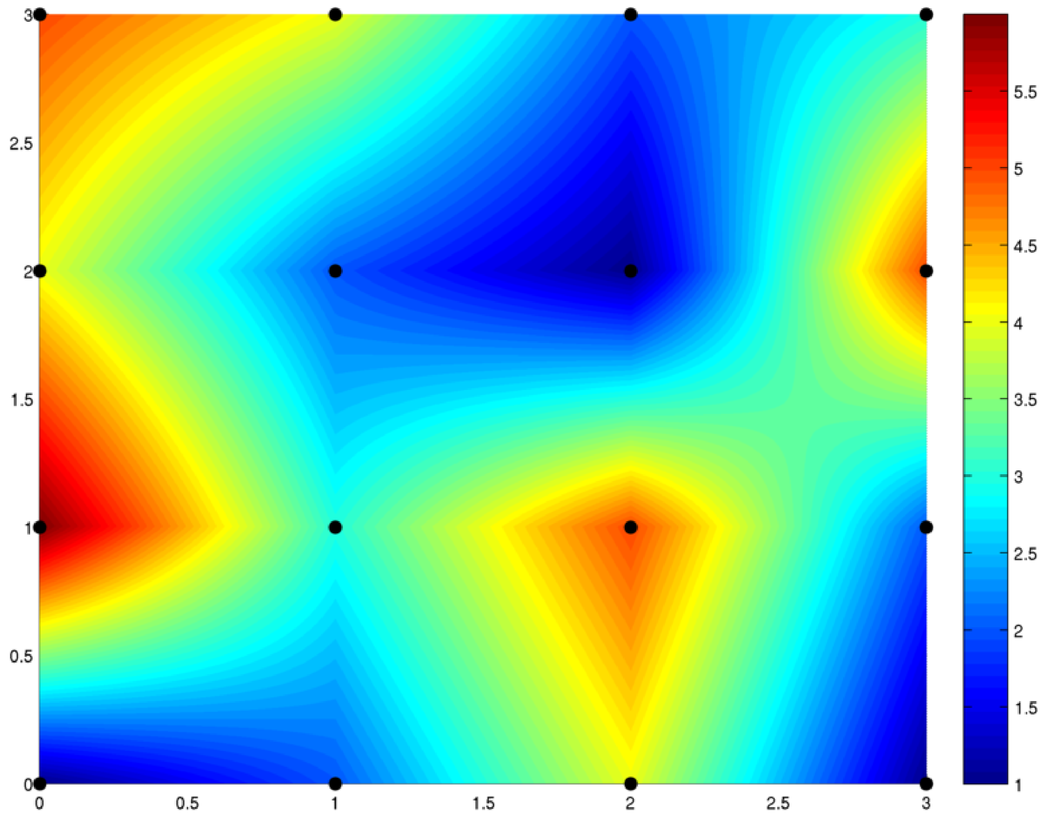
- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Bicubic interpolation
      - Piecewise polynomial
      - Smoother interpolant



[Wikipedia]

Interpolants for regular grids

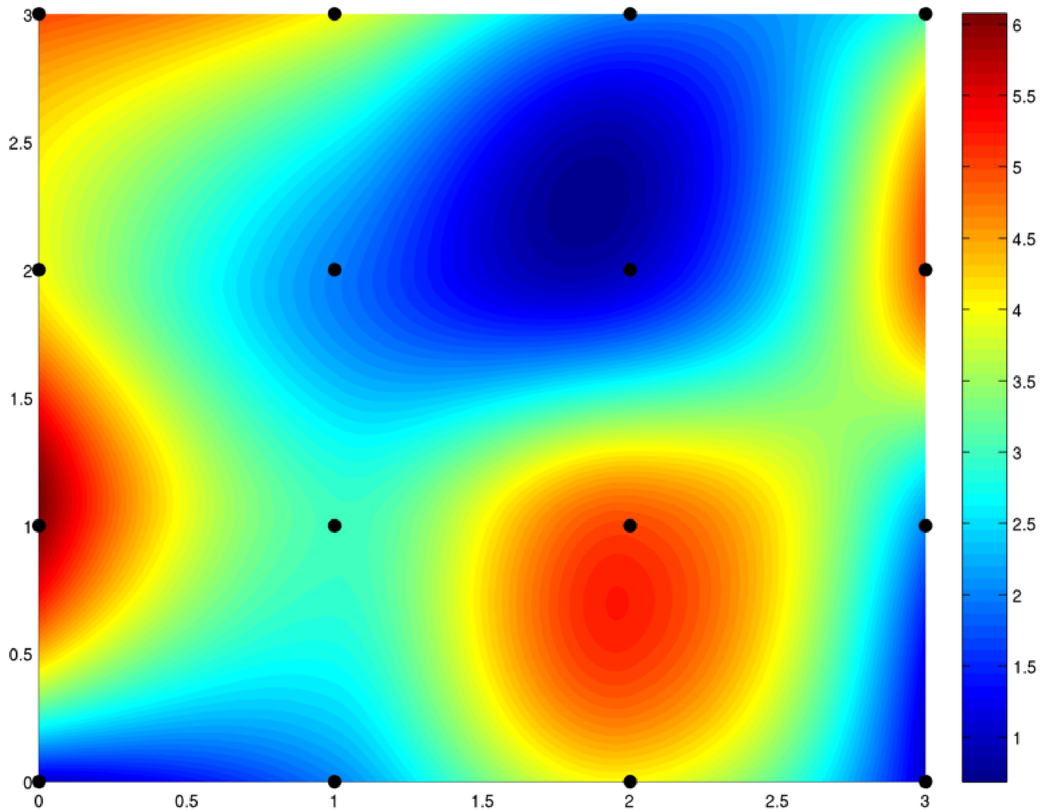
- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Bicubic interpolation
      - Piecewise polynomial
      - Smoother interpolant
      - Takes adjacent cells into account



[Wikipedia]

Interpolants for regular grids

- For regular grids of $\mathbb{R}^2$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Bicubic interpolation
      - Piecewise polynomial
      - Smoother interpolant
      - Takes adjacent cells into account



[Wikipedia]

Interpolants for regular grids

- For regular grids of $\mathbb{R}^3$

Interpolants for regular grids

- For regular grids of $\mathbb{R}^3$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \times \ ]z_1, z_2[\ \rightarrow \mathbb{R}$

Interpolants for regular grids

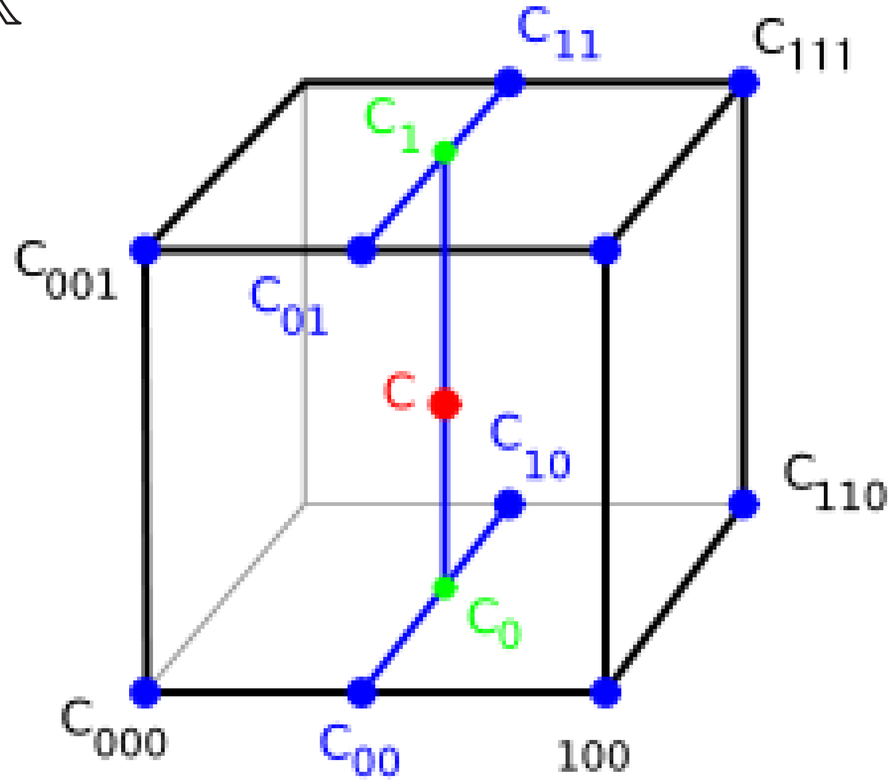
- For regular grids of $\mathbb{R}^3$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \times \ ]z_1, z_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Piecewise constant
      - Nearest neighbor
    - Trilinear
    - Tricubic

Interpolants for regular grids

- For regular grids of $\mathbb{R}^3$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \times \ ]z_1, z_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Piecewise constant
      - Nearest neighbor
    - **Trilinear**
    - Tricubic

Interpolants for regular grids

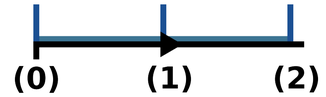
- For regular grids of $\mathbb{R}^3$
  - $f_{\mathbb{I}} : \ ]x_1, x_2[\ \times \ ]y_1, y_2[\ \times \ ]z_1, z_2[\ \rightarrow \mathbb{R}$
  - Examples
    - Piecewise constant
      - Nearest neighbor
    - **Trilinear**
    - Tricubic

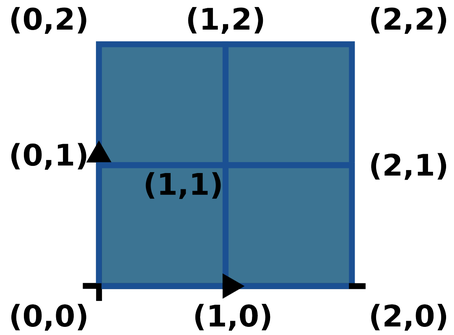


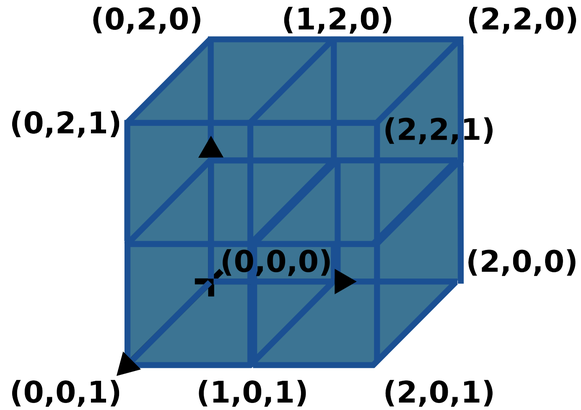
[Wikipedia]

Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Set of **vertices** (samples) in $\mathbb{R}^n$
    - With embedding functions to $\mathbb{R}^n$
  - **Connectivity**
    - Unit cells (pixels, voxels)
    - Implicitly encoded by the samples (space fill)
  - **Interpolant**

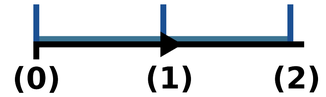


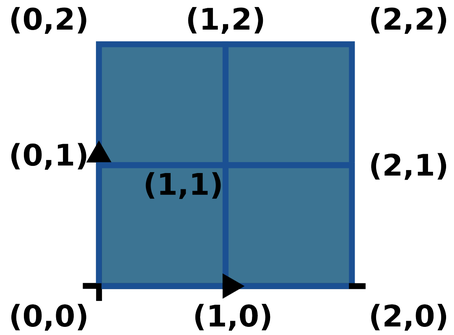


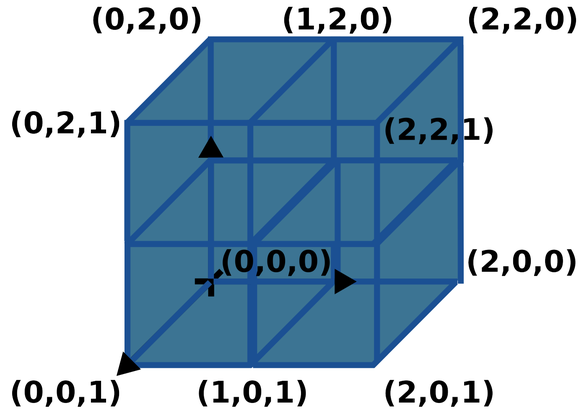


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Set of **vertices** (samples) in $\mathbb{R}^n$
    - With embedding functions to $\mathbb{R}^n$
  - **Connectivity**
    - Unit cells (pixels, voxels)
    - Implicitly encoded by the samples (space fill)
  - **Interpolant**
    - Computed on demand per cell

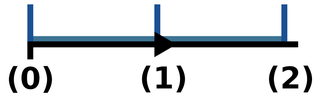


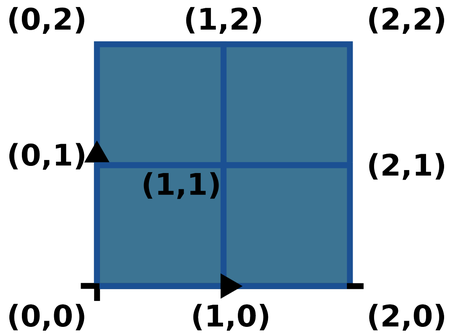


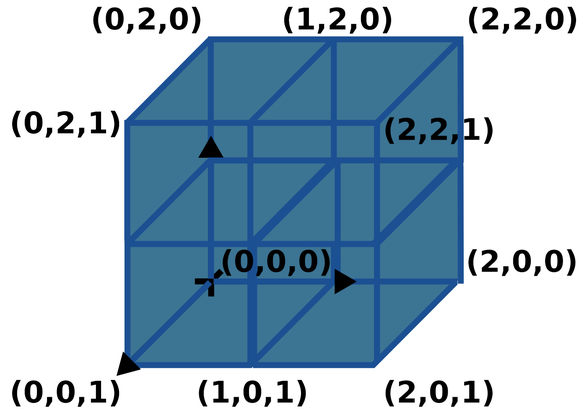


Euclidean spaces on a computer

- Notion of **regular grid** $\mathcal{G}$
  - Set of **vertices** (samples) in $\mathbb{R}^n$
    - With embedding functions to $\mathbb{R}^n$
  - **Connectivity**
    - Unit cells (pixels, voxels)
    - Implicitly encoded by the samples (space fill)
  - **Interpolant**
    - Computed on demand per cell
    - Usually linear functions



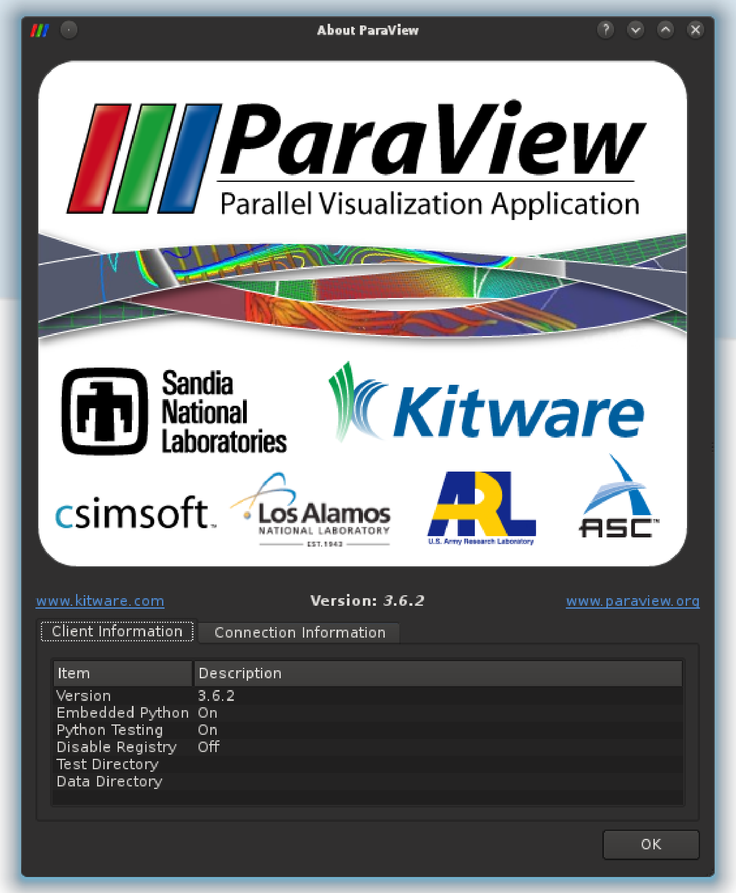




In practice

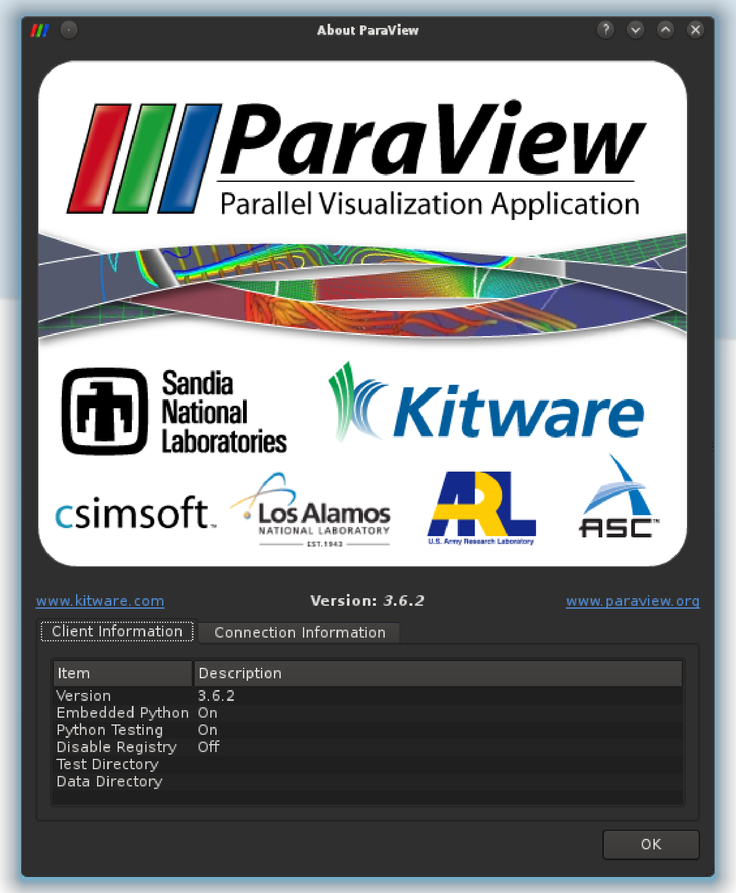
In practice

- Paraview
  - Open-source, interactive visualization platform



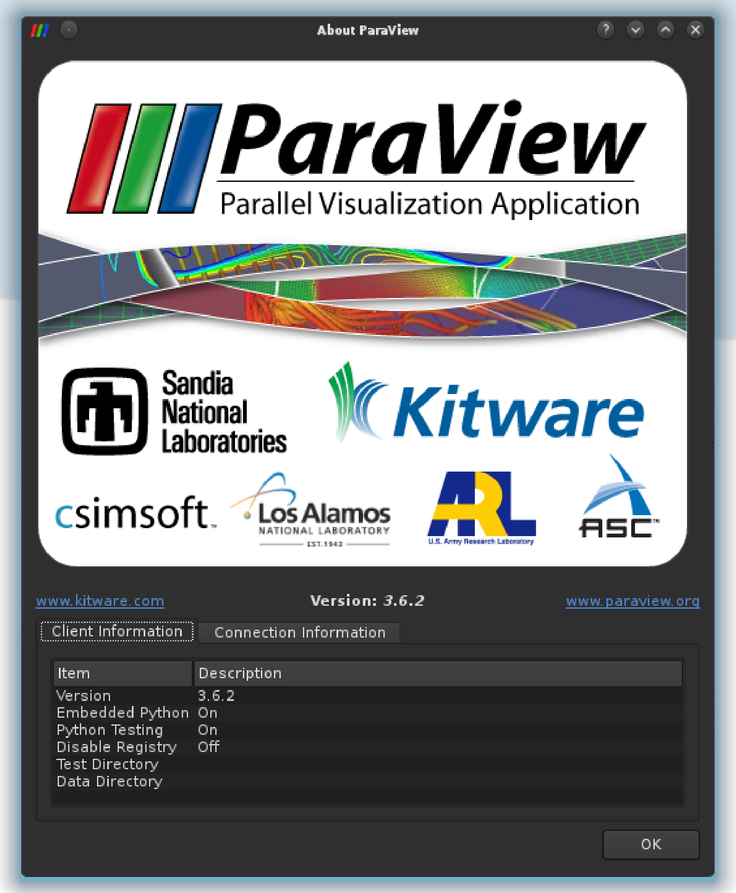
In practice

- Paraview
  - Open-source, interactive visualization platform
  - User interface front-end for the Visualization Tool Kit (VTK) C++ library (open-source)



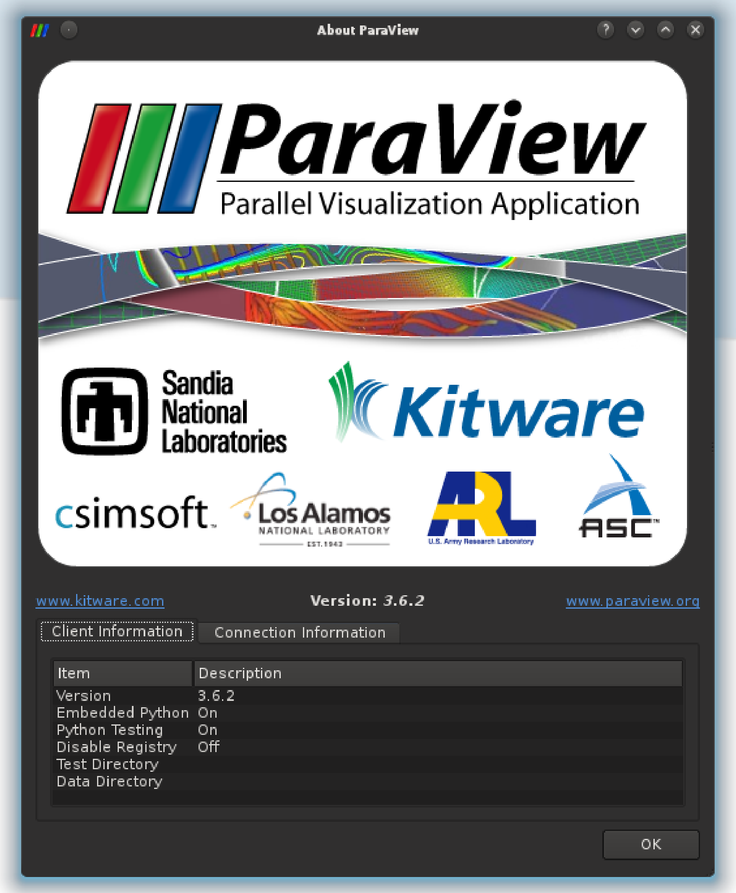
In practice

- Paraview
  - Open-source, interactive visualization platform
  - User interface front-end for the Visualization Tool Kit (VTK) C++ library (open-source)
  - Started in 1993
    - Initially General Electric, then Kitware (1998)



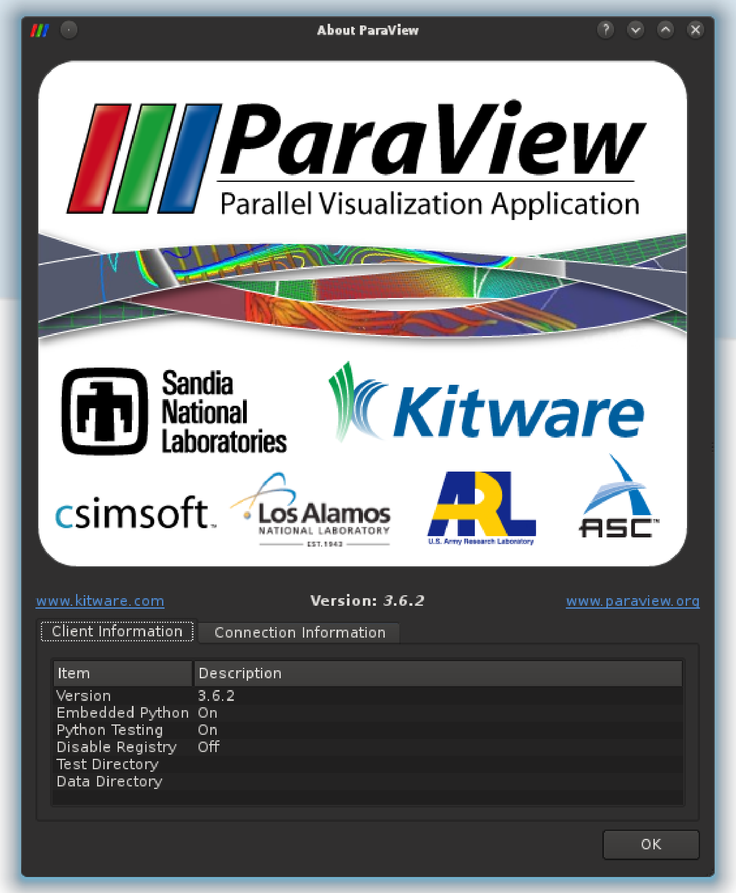
In practice

- Paraview
  - Open-source, interactive visualization platform
  - User interface front-end for the Visualization Tool Kit (VTK) C++ library (open-source)
  - Started in 1993
    - Initially General Electric, then Kitware (1998)
    - Many institutional contributors (companies, national labs, universities)

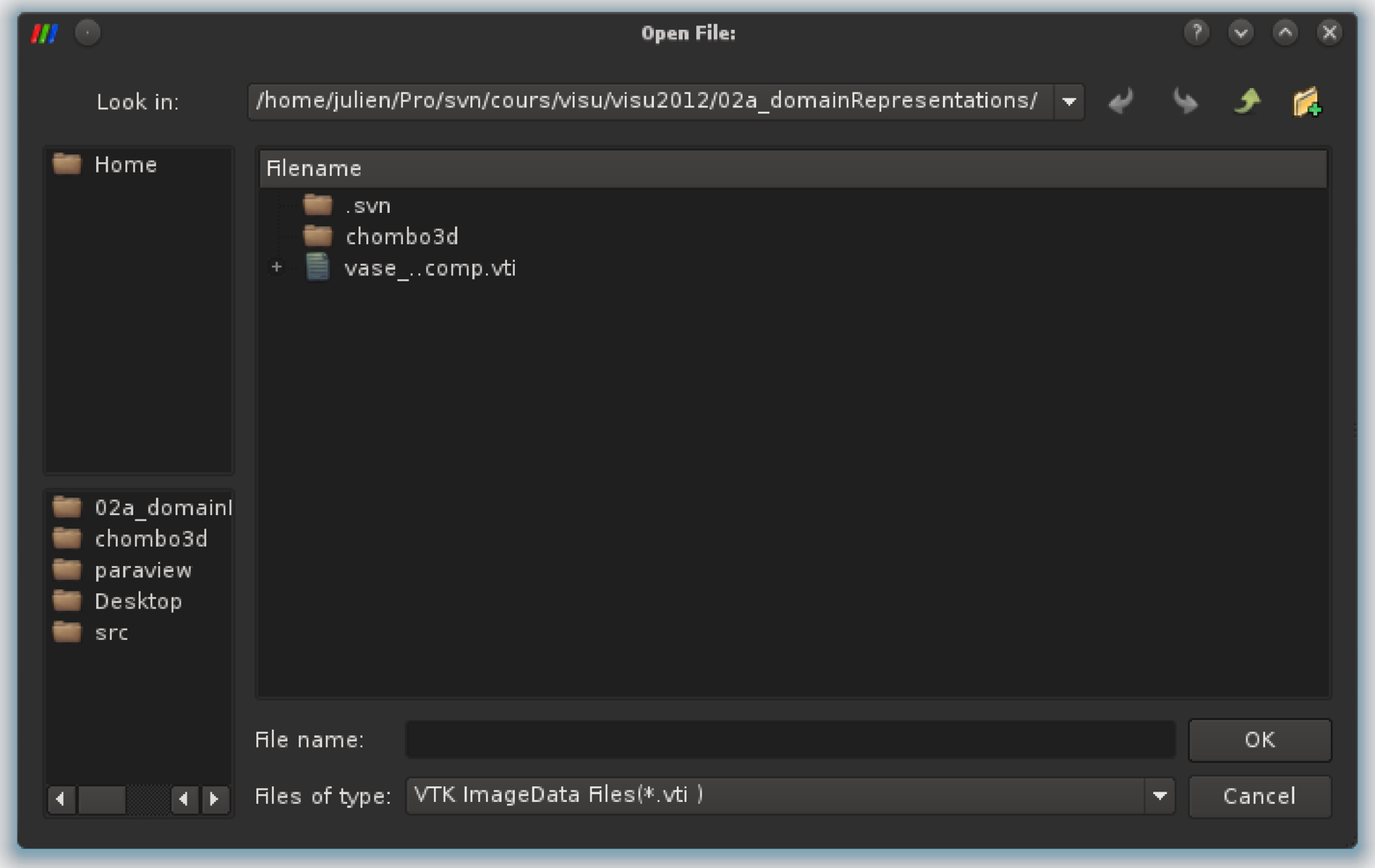


In practice

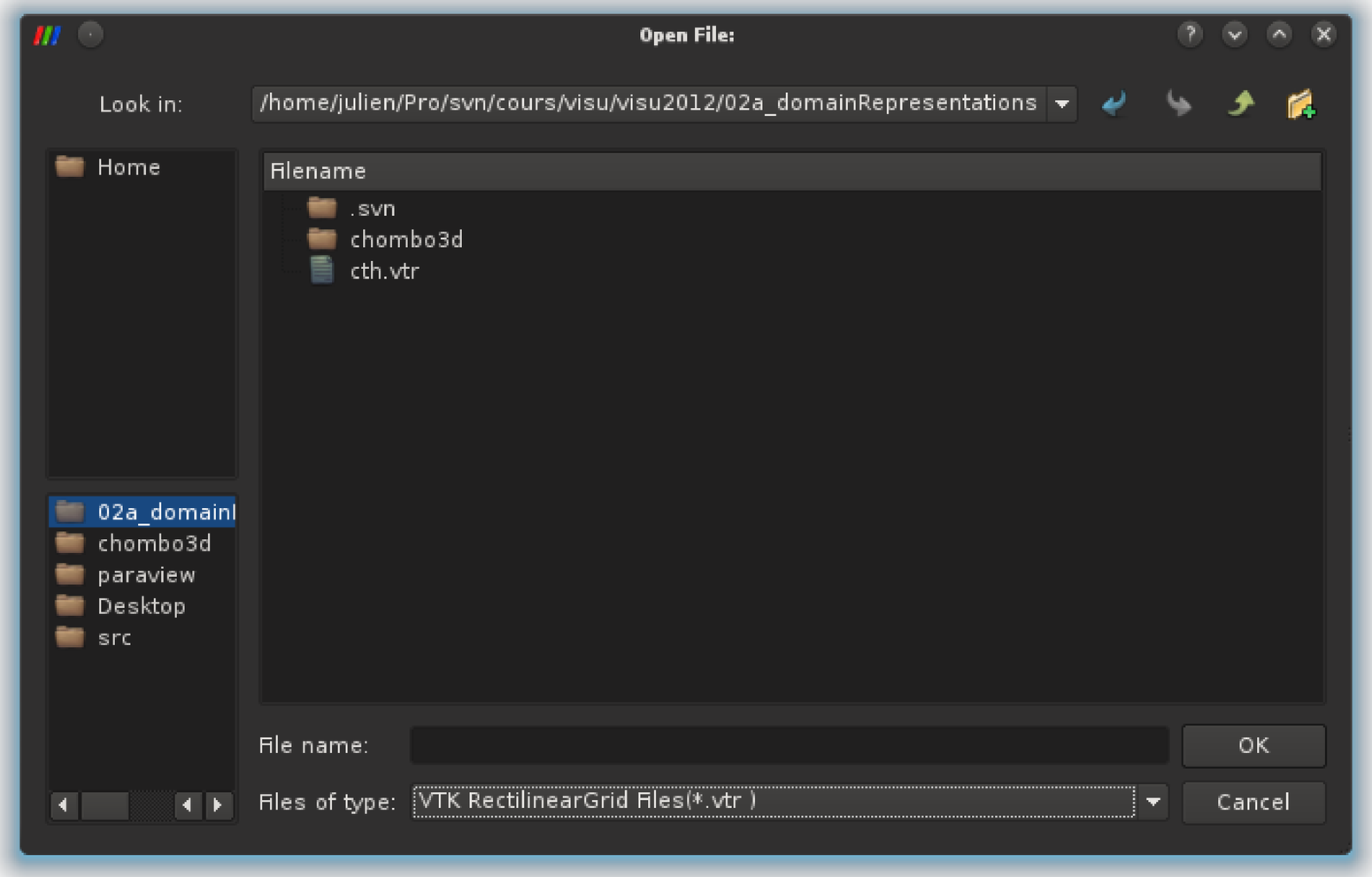
- Paraview
  - Open-source, interactive visualization platform
  - User interface front-end for the Visualization Tool Kit (VTK) C++ library (open-source)
  - Started in 1993
    - Initially General Electric, then Kitware (1998)
    - Many institutional contributors (companies, national labs, universities)
    - Over a million lines of code
    - Tens of thousands of users worldwide



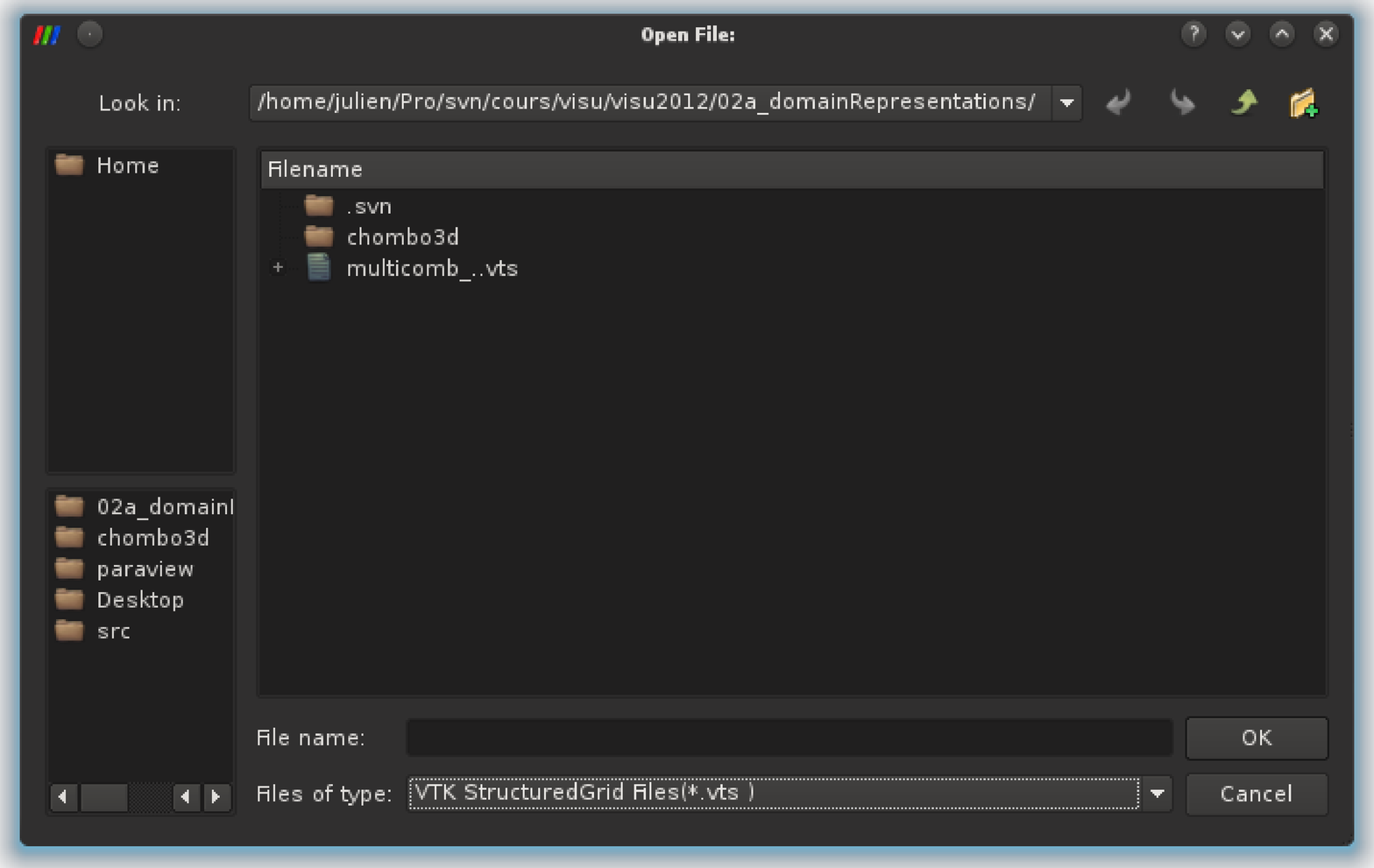
In practice


Open File:
Look in:
/home/julien/Pro/svn/cours/visu/visu2012/02a_domainRepresentations/
Home
Filename
.svn
chombo3d
vase_..comp.vti
02a_domainI
chombo3d
paraview
Desktop
src
File name:
OK
Files of type:
VTK ImageData Files(*.vti)
Cancel

In practice


Open File:
Look in:
/home/julien/Pro/svn/cours/visu/visu2012/02a_domainRepresentations
Home
Filename
.svn
chombo3d
cth.vtr
02a_domainI
chombo3d
paraview
Desktop
src
File name:
OK
Files of type:
VTK RectilinearGrid Files(*.vtr)
Cancel

In practice


Open File:
Look in:
/home/julien/Pro/svn/cours/visu/visu2012/02a_domainRepresentations/
Home
Filename
.svn
chombo3d
multicomb_..vts
02a_domainl
chombo3d
paraview
Desktop
src
File name:
OK
Files of type:
VTK StructuredGrid Files(*.vts)
Cancel

Kitware ParaView 3.6.2

File Edit View Sources Filters Animation Tools Help

Time: 0 0

SLCImage | Surface With Edges

Edit Color Map

Pipeline Browser

- builtin:
  - vase_*
    - ExtractCellsByRegion1
  - vase_*
  - vase_*

Object Inspector

Properties | Display | Information

Map Scalars

Apply Texture: None

Color by: SLCImage

Edit Color Map... | Rescale to Data Range

Slice

Slice Direction: XY Plane

Slice: 0

Annotation

Show cube axes | Edit

Style

Representation: Surface With Edges

Interpolation: Gouraud

Material: None

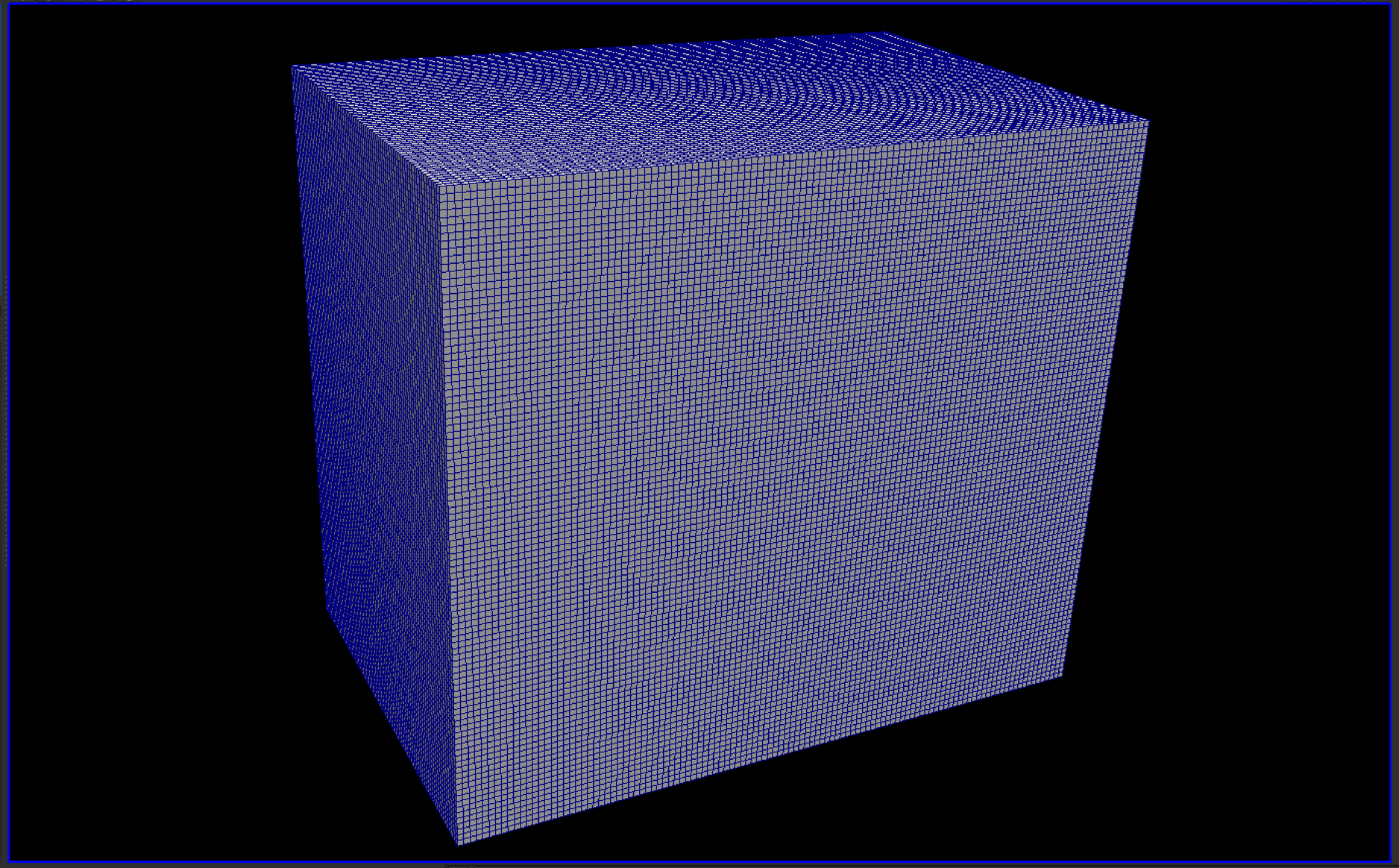
Point size: 2.00

Line width: 1.00

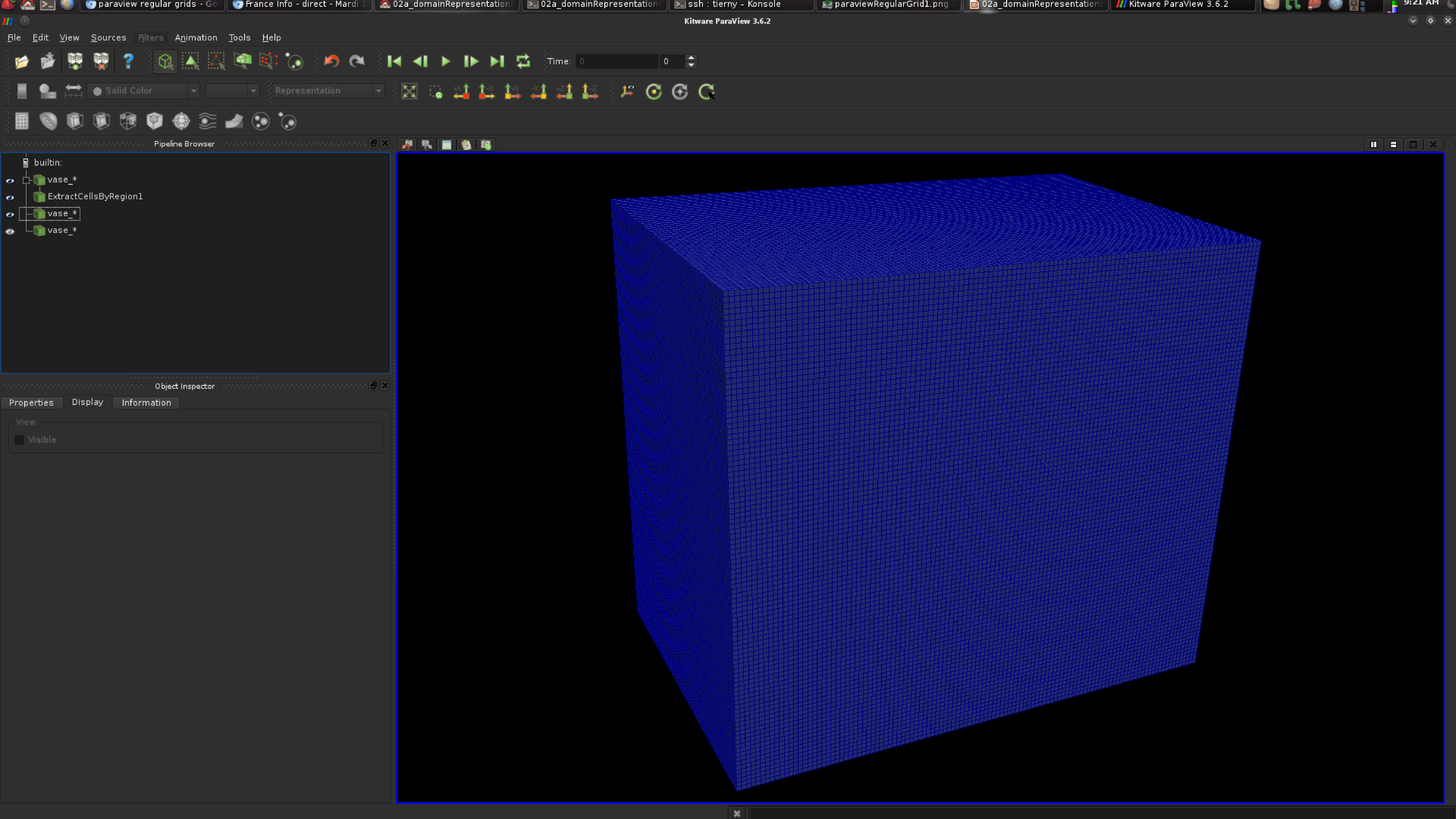
Opacity: 1.00

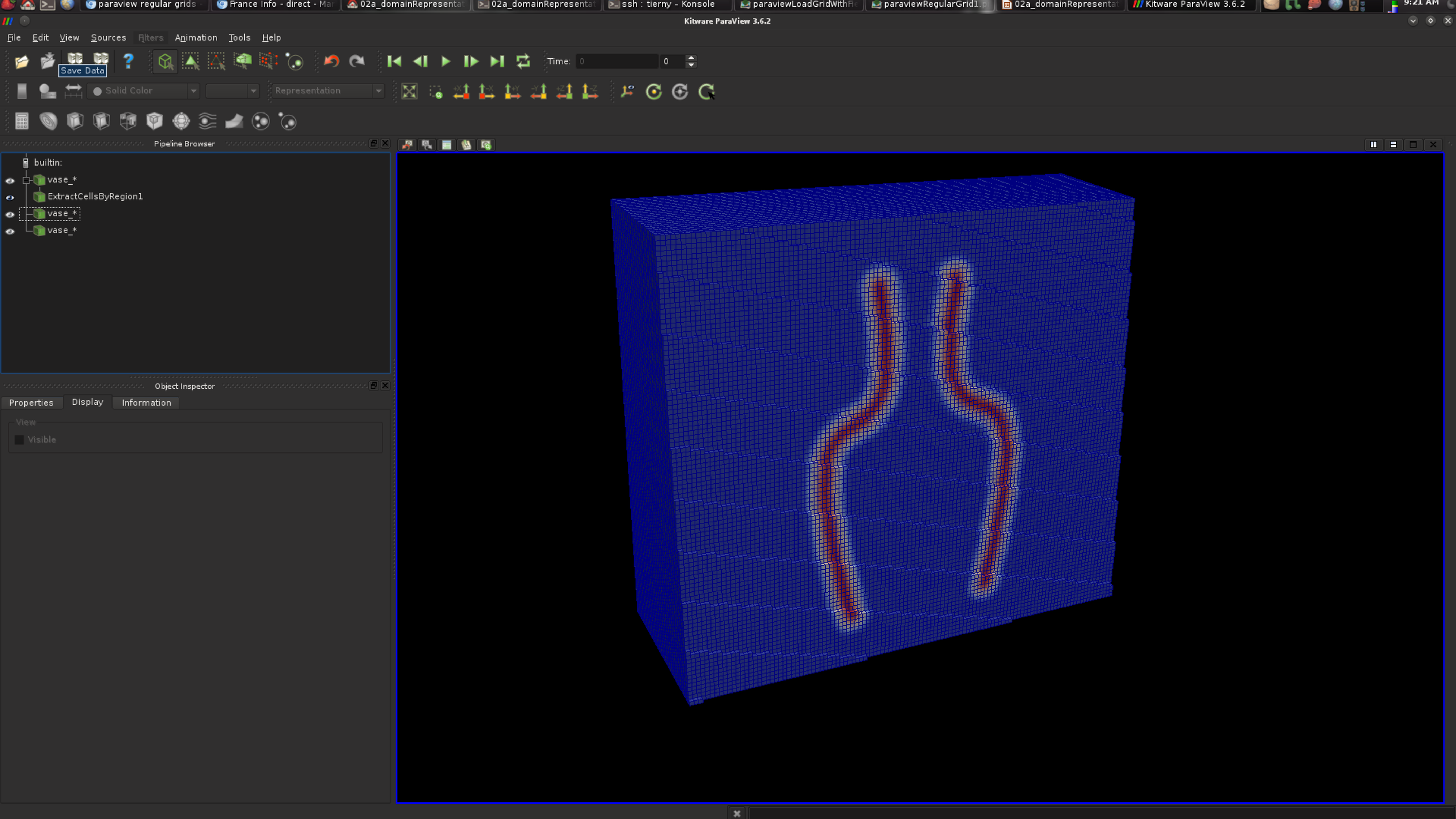
Volume mapper

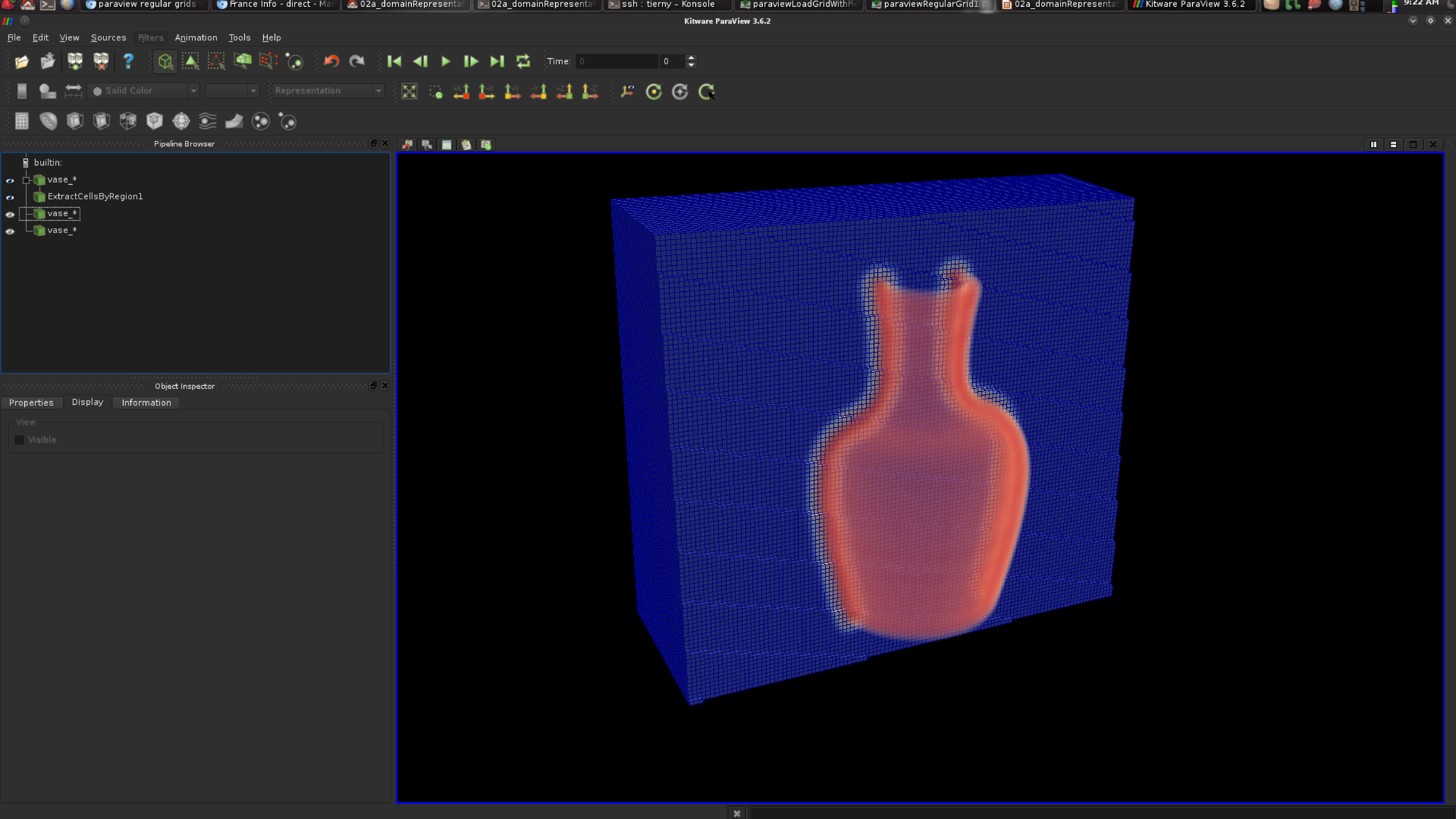
Edge Style



Edit Color Map


paraview regular grids - Go
France Info - direct - Mardi
02a_domainRepresentation
02a_domainRepresentation
ssh : tierny - Konsole
paraviewRegularGrid1.png
02a_domainRepresentation
Kitware ParaView 3.6.2
9:21 AM
Kitware ParaView 3.6.2
File Edit View Sources Filters Animation Tools Help
Time: 0 0
Solid Color
Representation
Pipeline Browser
builtin:
vase_*
ExtractCellsByRegion1
vase_*
vase_*
Object Inspector
Properties Display Information
View
Visible


Kitware ParaView 3.6.2
File Edit View Sources Filters Animation Tools Help
Save Data
Time:
Solid Color
Representation
Pipeline Browser
builtin:
vase_*
ExtractCellsByRegion1
vase_*
vase_*
Object Inspector
Properties Display Information
View
Visible


paraview regular grids
France Info - direct - Mar
02a_domainRepresentat
02a_domainRepresentat
ssh : tierny - Konsole
paraviewLoadGridWithFie
paraviewRegularGrid1.p
02a_domainRepresentat
Kitware ParaView 3.6.2
9:22 AM
Kitware ParaView 3.6.2
File Edit View Sources Filters Animation Tools Help
Time: 0 0
Solid Color
Representation
Pipeline Browser
builtin:
vase_*
ExtractCellsByRegion1
vase_*
vase_*
Object Inspector
Properties Display Information
View
Visible

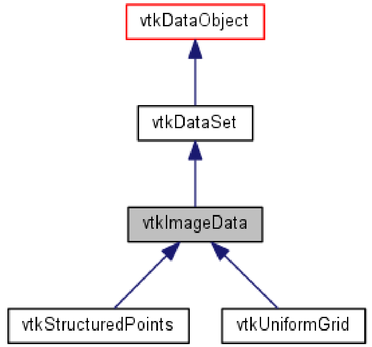
VTK

vtkImageData Class Reference

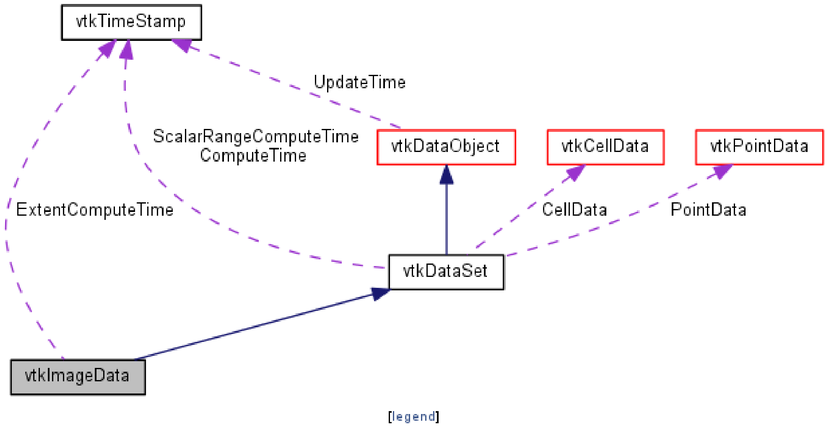
topologically and geometrically regular array of data More...

`#include <vtkImageData.h>`

Inheritance diagram for vtkImageData:



Collaboration diagram for vtkImageData:



Public Types

typedef **vtkDataSet** **Superclass**

Public Member Functions

virtual int	**IsA** (const char *type)
vtkImageData *	**NewInstance** () const
void	**PrintSelf** (ostream &os, **vtkIndent** indent)
virtual void	**CopyStructure** (**vtkDataSet** *ds)
virtual int	**GetDataObjectType** ()
virtual void	**Initialize** ()
virtual void	**SetDimensions** (**int** i, **int** j, **int** k)
virtual void	**SetDimensions** (const **int** dims[3])
virtual int	**GetDataDimension** ()
virtual void	**AllocateScalars** (**int** dataType, **int** numComponents)
virtual void	**AllocateScalars** (**vtkInformation** *pipeline_info)
virtual void	**Crop** (const **int** *updateExtent)
virtual unsigned long	**GetActualMemorySize** ()
int	**GetScalarType** ()
const char *	**GetScalarTypeAsString** ()
virtual void	**CopyInformationFromPipeline** (**vtkInformation** *information)
virtual void	**PrepareForNewData** ()
void	**GetArrayIncrements** (**vtkDataArray** *array, **vtkIdType** increments[3])
void	**ComputeInternalExtent** (**int** *intExt, **int** *tgtExt, **int** *bnds)
virtual int	**GetExtentType** ()
virtual vtkIdType	**GetNumberOfCells** ()
virtual vtkIdType	**GetNumberOfPoints** ()
virtual double *	**GetPoint** (**vtkIdType** ptId)
virtual void	**GetPoint** (**vtkIdType** id, **double** x[3])
virtual vtkCell *	**GetCell** (**vtkIdType** cellId)
virtual void	**GetCell** (**vtkIdType** cellId, **vtkGenericCell** *cell)
virtual void	**GetCellBounds** (**vtkIdType** cellId, **double** bounds[6])
virtual vtkIdType	**FindPoint** (**double** x, **double** y, **double** z)
virtual vtkIdType	**FindPoint** (**double** x[3])
virtual vtkIdType	**FindCell** (**double** x[3], **vtkCell** *cell, **vtkIdType** cellId, **double** tol2, **int** &subId, **double** pcoords[3], **double** *weights)
virtual vtkIdType	**FindCell** (**double** x[3], **vtkCell** *cell, **vtkGenericCell** *gencell, **vtkIdType** cellId, **double** tol2, **int** &subId, **double** pcoords[3], **double** *weights)
virtual vtkCell *	**FindAndGetCell** (**double** x[3], **vtkCell** *cell, **vtkIdType** cellId, **double** tol2, **int** &subId, **double** pcoords[3], **double** *weights)
virtual int	**GetCellType** (**vtkIdType** cellId)
virtual void	**GetCellPoints** (**vtkIdType** cellId, **vtkIdList** *ptIds)
virtual void	**GetPointCells** (**vtkIdType** ptId, **vtkIdList** *cellIds)
virtual void	**ComputeBounds** ()
virtual int	**GetMaxCellSize** ()
virtual int *	**GetDimensions** ()
virtual void	**GetDimensions** (**int** dims[3])
virtual int	**ComputeStructuredCoordinates** (const **double** x[3], **int** ijk[3], **double** pcoords[3])

Recap

- Euclidean domains

Recap

- Euclidean domains
  - Easily represented with regular grids

Recap

- Euclidean domains
  - Easily represented with regular grids

- What about other spaces?

Recap

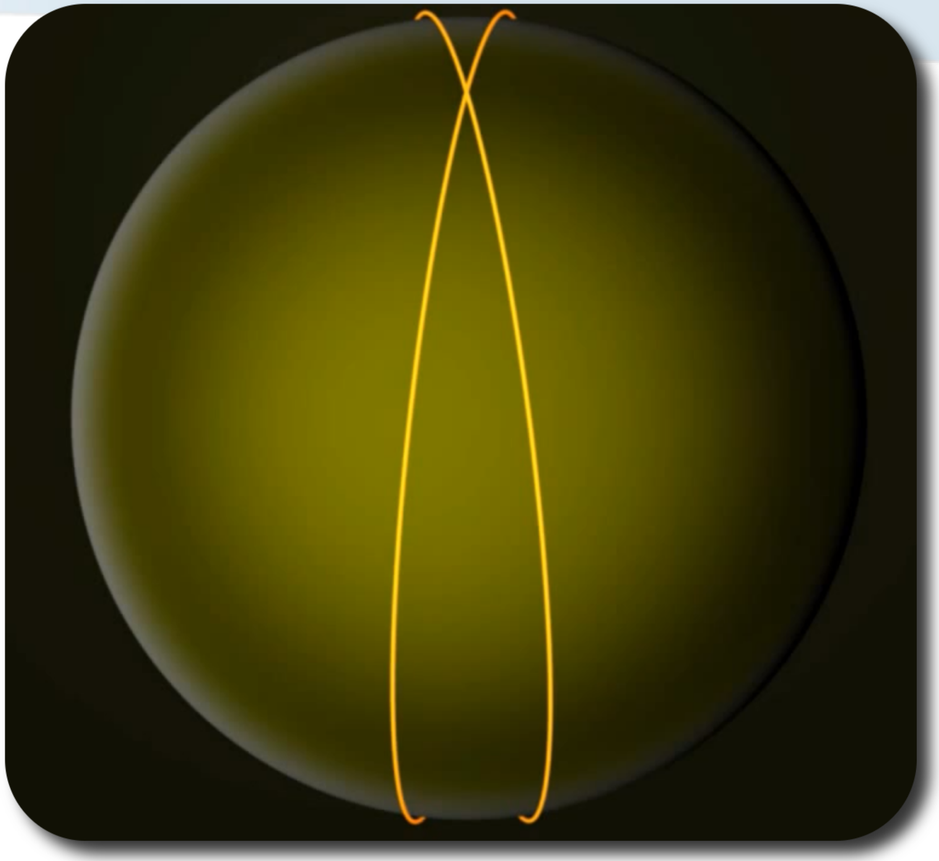
- Euclidean domains
  - Easily represented with regular grids

- What about other spaces?
  - 5th axiom of the *Elements*:
    - *Through a point not on a given straight line, at most one line can be drawn that never meets the given line*

Recap

- Euclidean domains
  - Easily represented with regular grids

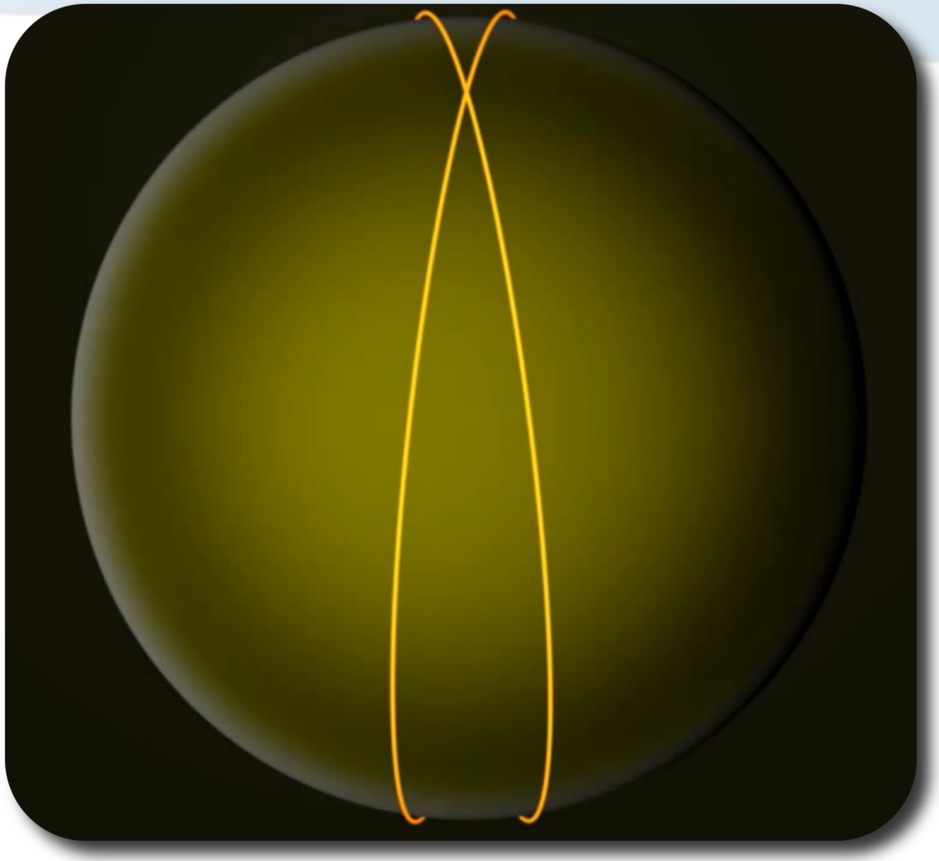
- What about other spaces?
  - 5th axiom of the *Elements*:
    - *Through a point not on a given straight line, at most one line can be drawn that never meets the given line*



Recap

- Euclidean domains
  - Easily represented with regular grids

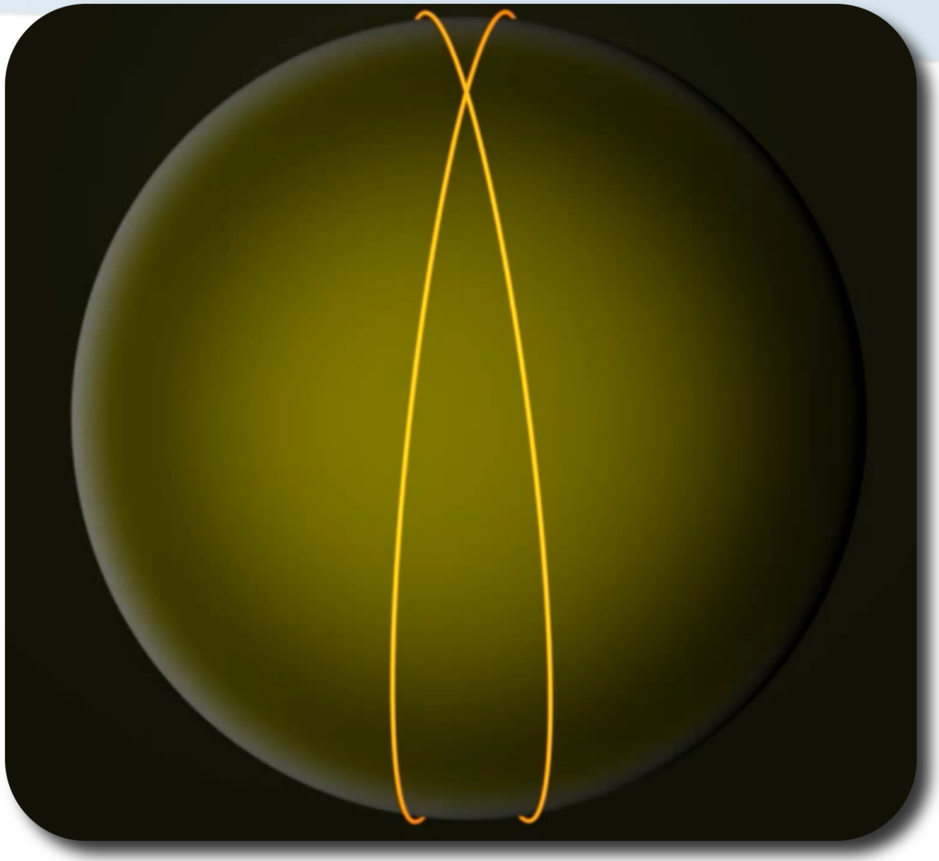
- What about other spaces?
  - 5th axiom of the *Elements*:
    - *Through a point not on a given straight line, at most one line can be drawn that never meets the given line*
  - What if this axiom was not true?



Recap

- Euclidean domains
  - Easily represented with regular grids

- What about other spaces?
  - 5th axiom of the *Elements*:
    - *Through a point not on a given straight line, at most one line can be drawn that never meets the given line*
  - What if this axiom was not true?
  - What would geometry look like?



Recap

- Euclidean domains
  - Easily represented with regular grids
- What about other spaces?
  - Riemannian geometry

Recap

- Euclidean domains
  - Easily represented with regular grids
- What about other spaces?
  - Riemannian geometry
  - 19th century

Recap

- Euclidean domains
  - Easily represented with regular grids

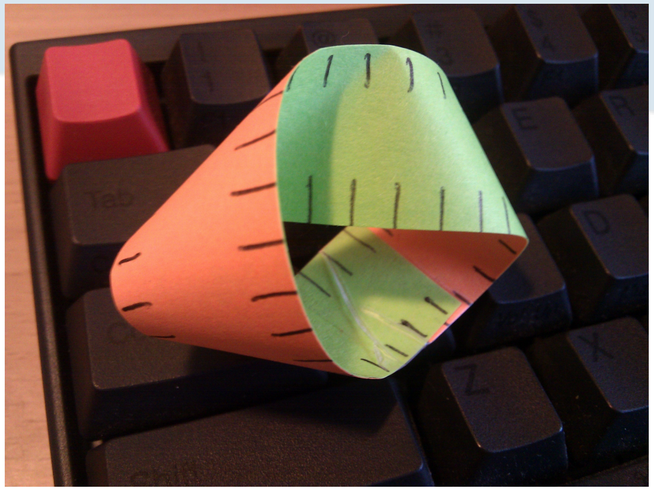
- What about other spaces?
  - Riemannian geometry
  - 19th century
  - Gauss, Lobachevsky, Poincaré, Beltrami, etc...

Recap

- Euclidean domains
  - Easily represented with regular grids

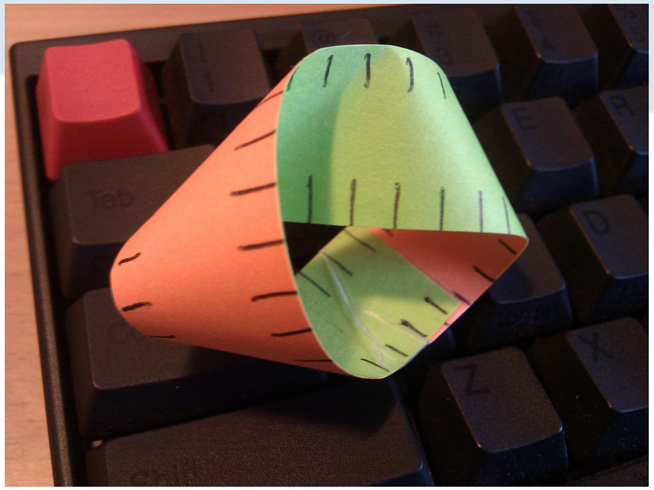
- What about other spaces?
  - Riemannian geometry
  - 19th century
  - Gauss, Lobachevsky, Poincaré, Beltrami, etc...
  - General notion of manifold

Manifolds on a computer



- Notion of simplex
  - A d-simplex σ is the convex hull of (d + 1) affinely independent points of an euclidean space $\mathbb{R}^n$
  - $0 \leq d \leq n$
  - d is the dimension of σ

Manifolds on a computer



- Notion of simplex
  - A d-simplex σ is the convex hull of (d + 1) affinely independent points of an euclidean space $\mathbb{R}^n$
  - $0 \leq d \leq n$
  - d is the dimension of σ
  - Smallest combinatorial construction to represent a cell of dimension d

Manifolds on a computer

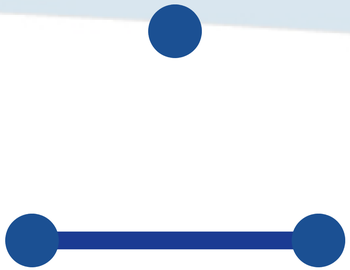
- Notion of simplex
  - A d-simplex σ is the convex hull of (d + 1) affinely independent points of an euclidean space $\mathbb{R}^n$
  - $0 \leq d \leq n$

Manifolds on a computer

- Notion of simplex
  - A d-simplex σ is the convex hull of (d + 1) affinely independent points of an euclidean space $\mathbb{R}^n$
  - $0 \leq d \leq n$
  - 0-simplex: *vertex*

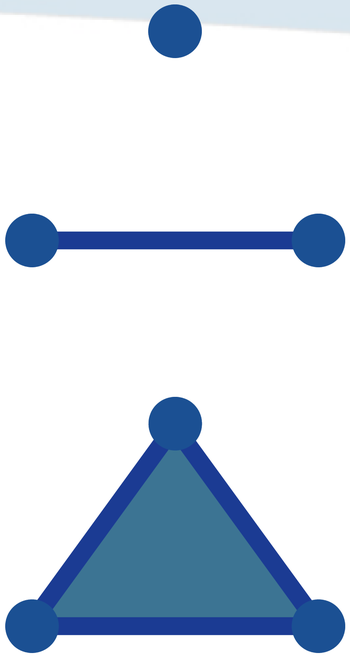
Manifolds on a computer

- Notion of simplex
  - A d-simplex σ is the convex hull of (d + 1) affinely independent points of an euclidean space $\mathbb{R}^n$
  - $0 \leq d \leq n$
  - 0-simplex: *vertex*
  - 1-simplex: *edge*



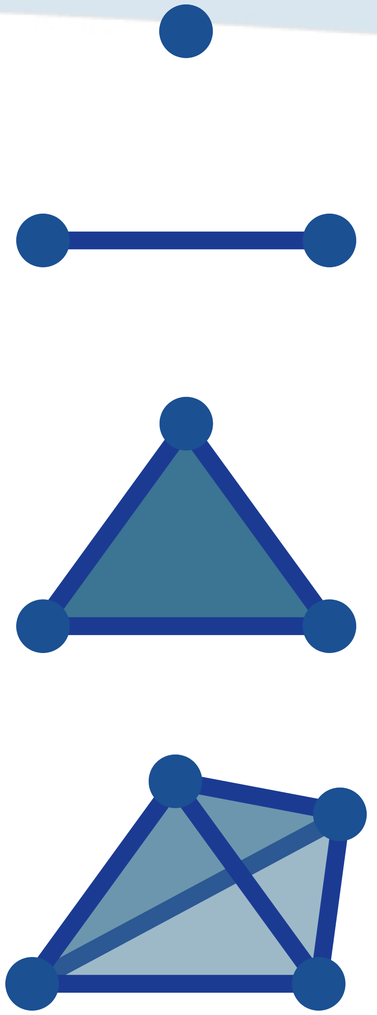
Manifolds on a computer

- Notion of simplex
  - A d-simplex σ is the convex hull of (d + 1) affinely independent points of an euclidean space $\mathbb{R}^n$
  - $0 \leq d \leq n$
  - 0-simplex: *vertex*
  - 1-simplex: *edge*
  - 2-simplex: *triangle*



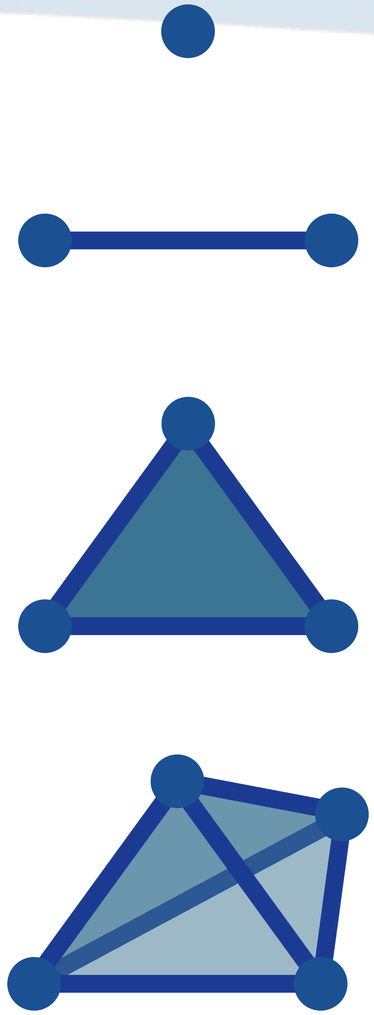
Manifolds on a computer

- Notion of simplex
  - A d-simplex σ is the convex hull of (d + 1) affinely independent points of an euclidean space $\mathbb{R}^n$
  - $0 \leq d \leq n$
  - 0-simplex: *vertex*
  - 1-simplex: *edge*
  - 2-simplex: *triangle*
  - 3-simplex: *tetrahedron*



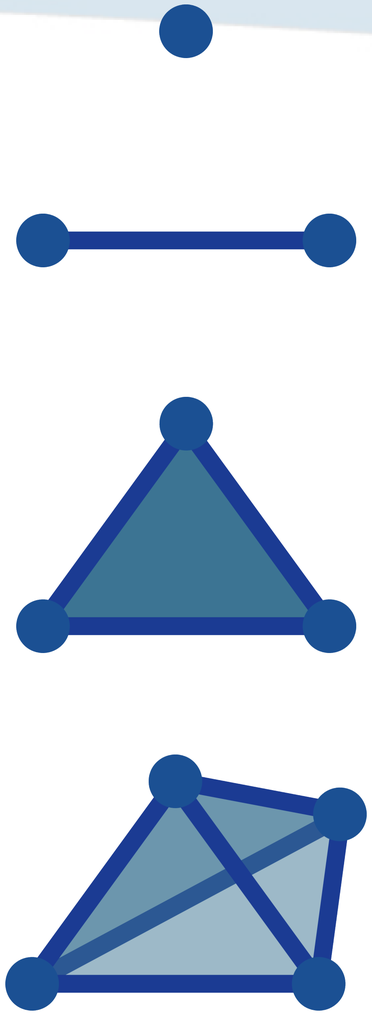
Manifolds on a computer

- Notion of face



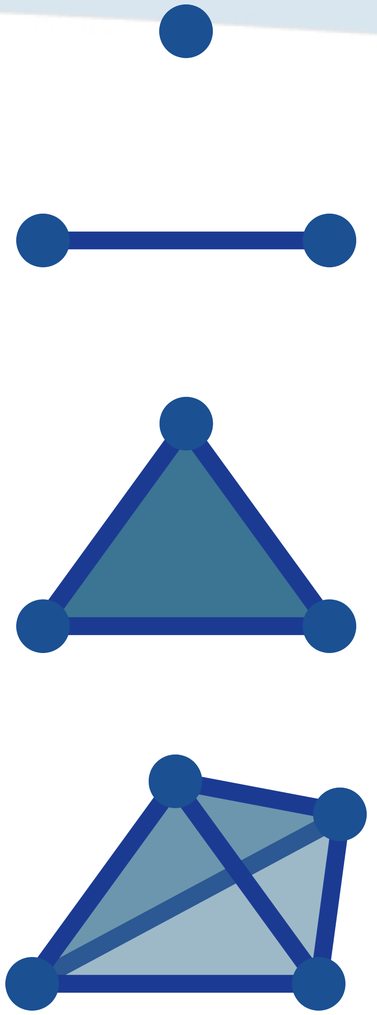
Manifolds on a computer

- Notion of face
  - A face τ of a d-simplex σ is the simplex defined by a non-empty subset of the d+1 points of σ
  - Noted $\tau \leq \sigma$



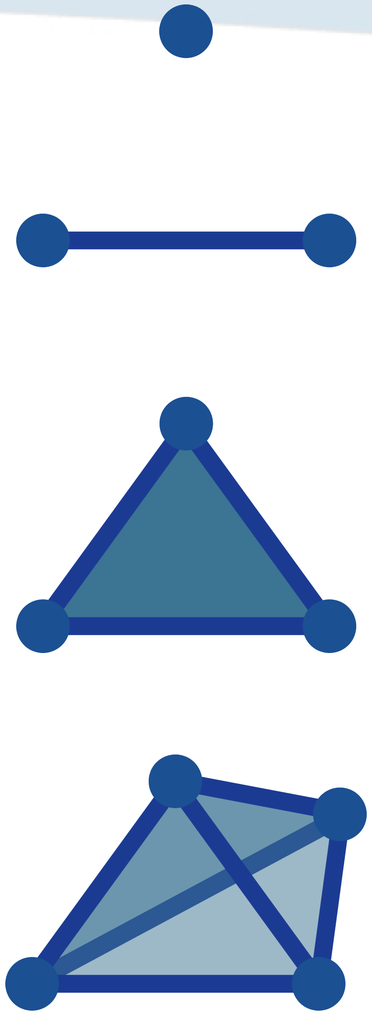
Manifolds on a computer

- Notion of face
  - A face τ of a d-simplex σ is the simplex defined by a non-empty subset of the d+1 points of σ
  - Noted $\tau \leq \sigma$
- Recursive construction!



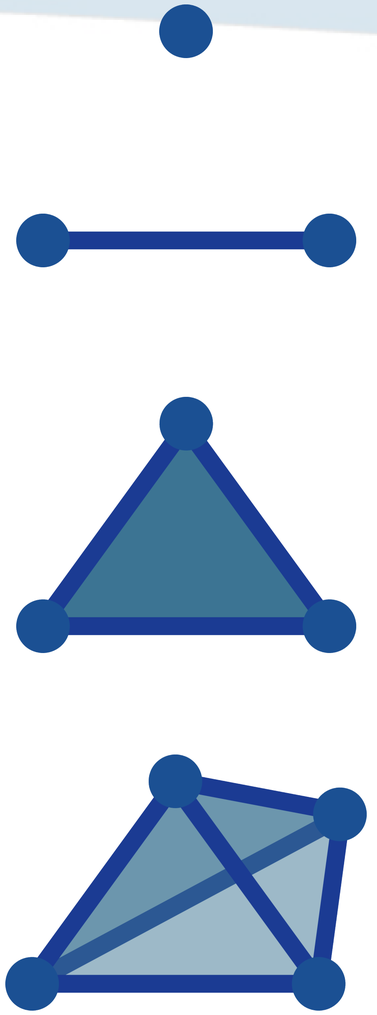
Manifolds on a computer

- Notion of face
  - A face τ of a d-simplex σ is the simplex defined by a non-empty subset of the d+1 points of σ
  - Noted $\tau \leq \sigma$
- Recursive construction!
  - A 3-simplex has 4 2-simplices as faces



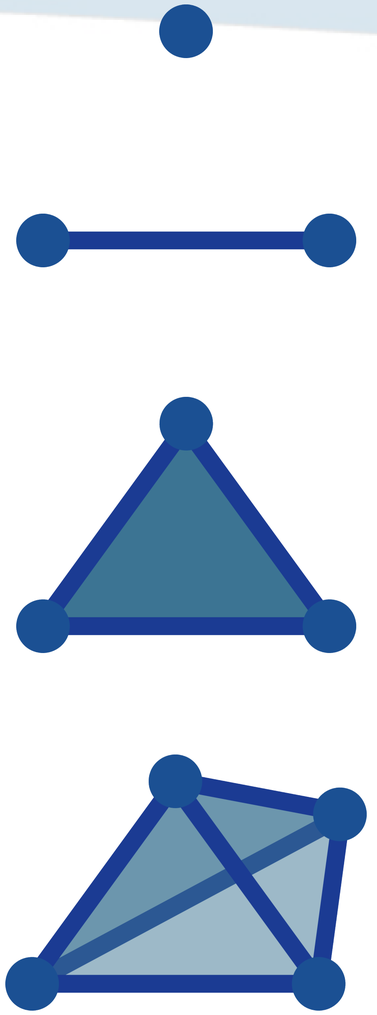
Manifolds on a computer

- Notion of face
  - A face τ of a d-simplex σ is the simplex defined by a non-empty subset of the d+1 points of σ
  - Noted $\tau \leq \sigma$
- Recursive construction!
  - A 3-simplex has 4 2-simplices as faces
  - A 2-simplex has 3 1-simplices as faces



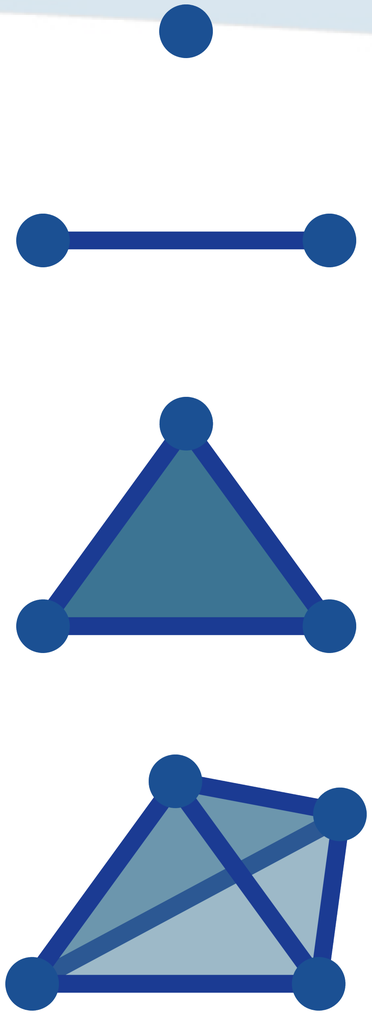
Manifolds on a computer

- Notion of face
  - A face τ of a d-simplex σ is the simplex defined by a non-empty subset of the d+1 points of σ
  - Noted $\tau \leq \sigma$
- Recursive construction!
  - A 3-simplex has 4 2-simplices as faces
  - A 2-simplex has 3 1-simplices as faces
  - A 1-simplex has 2 0-simplices as faces



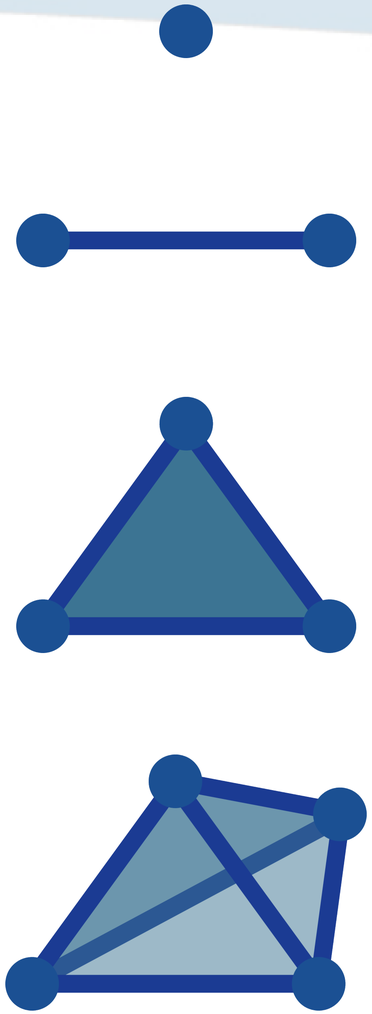
Manifolds on a computer

- Notion of face
  - A face τ of a d-simplex σ is the simplex defined by a non-empty subset of the d+1 points of σ
  - Noted $\tau \leq \sigma$
- Recursive construction!
  - A 3-simplex has 4 2-simplices as faces
  - A 2-simplex has 3 1-simplices as faces
  - A 1-simplex has 2 0-simplices as faces
  - How many faces in a 3-simplex (total)?



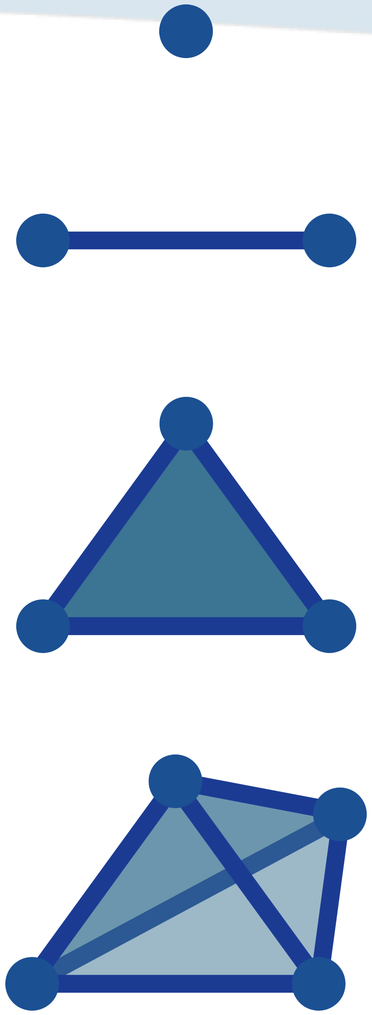
Manifolds on a computer

- Notion of face
  - A face τ of a d-simplex σ is the simplex defined by a non-empty subset of the d+1 points of σ
  - Noted $\tau \leq \sigma$
- Recursive construction!
  - A 3-simplex has 4 2-simplices as faces
  - A 2-simplex has 3 1-simplices as faces
  - A 1-simplex has 2 0-simplices as faces
  - How many faces in a 3-simplex (total)? 15



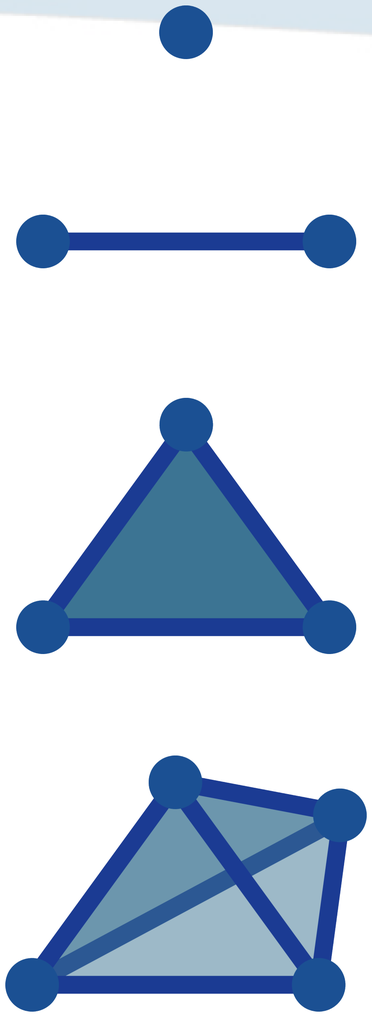
Manifolds on a computer

- Notion of **simplicial complex**



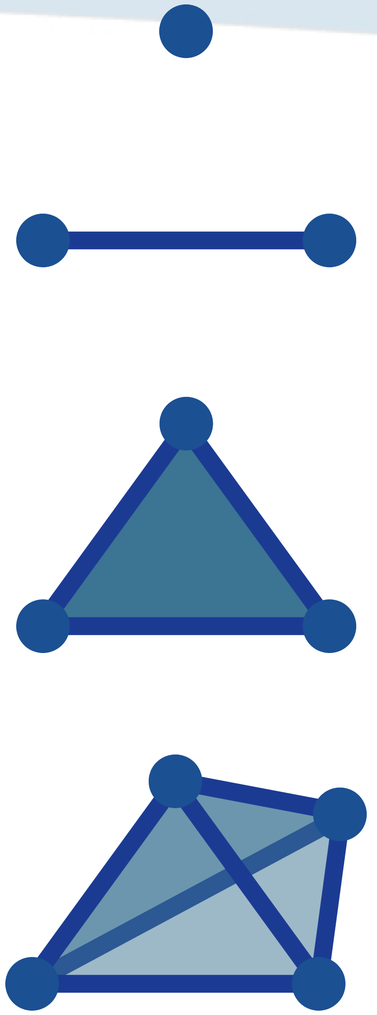
Manifolds on a computer

- Notion of **simplicial complex**
  - A simplicial complex $\mathcal{K}$ is a finite collection of non-empty simplices $\{\sigma_i\}$ such that



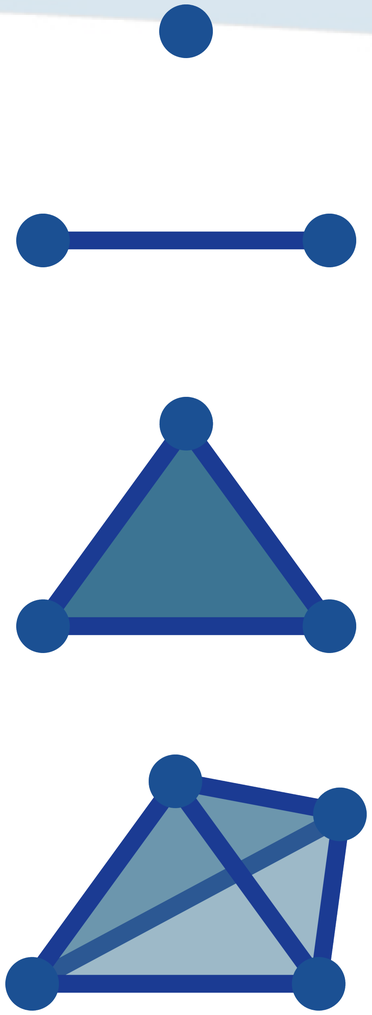
Manifolds on a computer

- Notion of **simplicial complex**
  - A simplicial complex $\mathcal{K}$ is a finite collection of non-empty simplices $\{\sigma_i\}$ such that
    - Every face τ of a simplex σ_i is also in $\mathcal{K}$

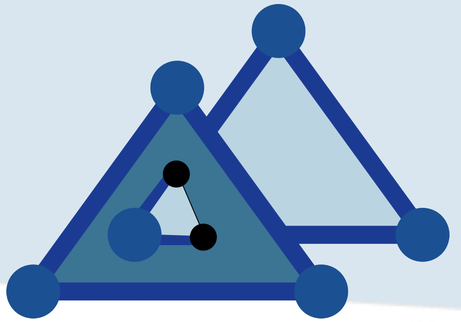


Manifolds on a computer

- Notion of **simplicial complex**
  - A simplicial complex $\mathcal{K}$ is a finite collection of non-empty simplices $\{\sigma_i\}$ such that
    - Every face τ of a simplex σ_i is also in $\mathcal{K}$
    - Any two simplices σ_i and σ_j intersect in a common face or not at all.

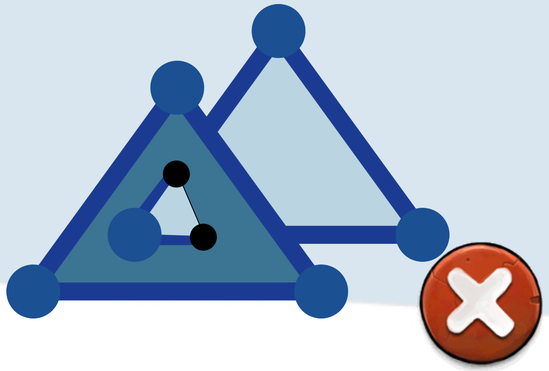


Manifolds on a computer



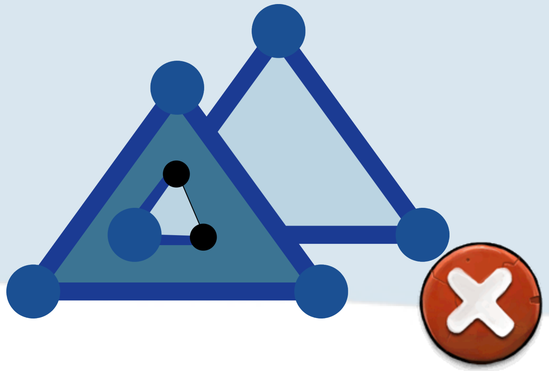
- Notion of **simplicial complex**
  - A simplicial complex $\mathcal{K}$ is a finite collection of non-empty simplices $\{\sigma_i\}$ such that
    - Every face τ of a simplex σ_i is also in $\mathcal{K}$
    - Any two simplices σ_i and σ_j intersect in a common face or not at all.

Manifolds on a computer

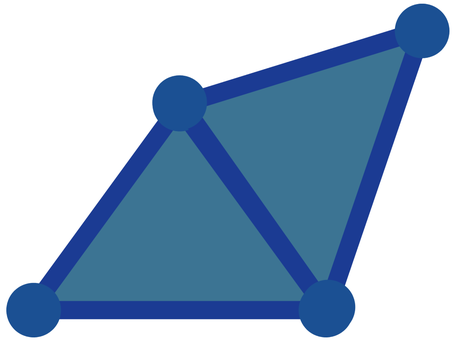


- Notion of **simplicial complex**
  - A simplicial complex $\mathcal{K}$ is a finite collection of non-empty simplices $\{\sigma_i\}$ such that
    - Every face τ of a simplex σ_i is also in $\mathcal{K}$
    - Any two simplices σ_i and σ_j intersect in a common face or not at all.

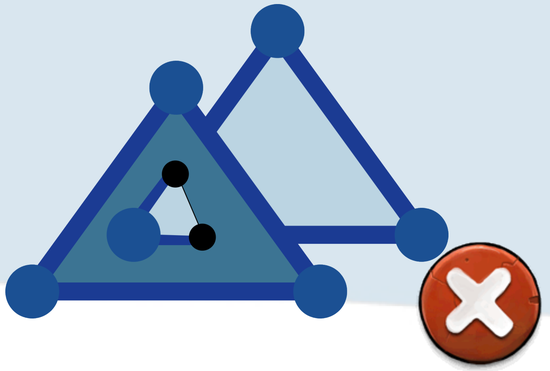
Manifolds on a computer



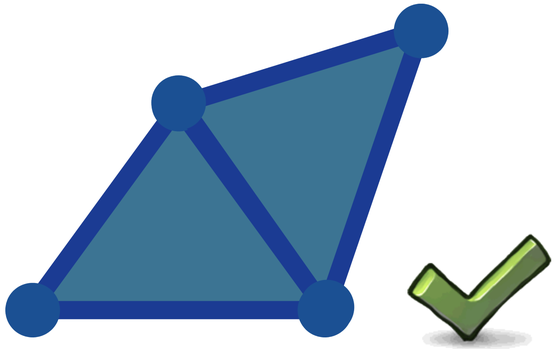
- Notion of **simplicial complex**
  - A simplicial complex $\mathcal{K}$ is a finite collection of non-empty simplices $\{\sigma_i\}$ such that
    - Every face τ of a simplex σ_i is also in $\mathcal{K}$
    - Any two simplices σ_i and σ_j intersect in a common face or not at all.



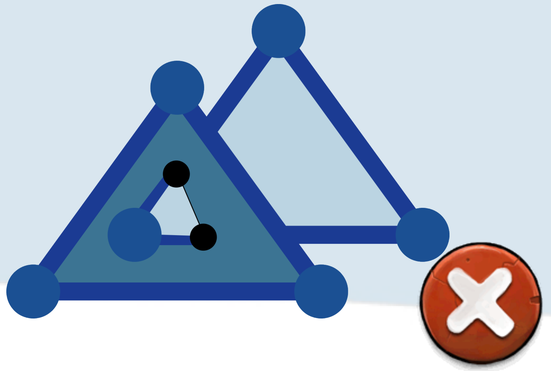
Manifolds on a computer



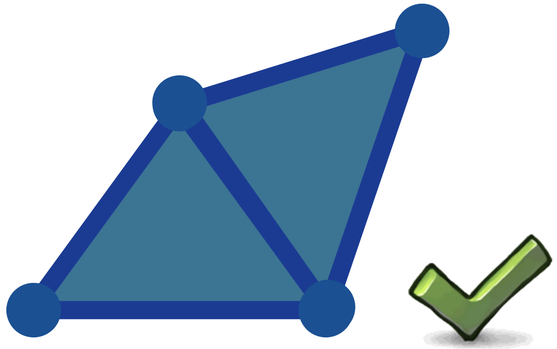
- Notion of **simplicial complex**
  - A simplicial complex $\mathcal{K}$ is a finite collection of non-empty simplices $\{\sigma_i\}$ such that
    - Every face τ of a simplex σ_i is also in $\mathcal{K}$
    - Any two simplices σ_i and σ_j intersect in a common face or not at all.

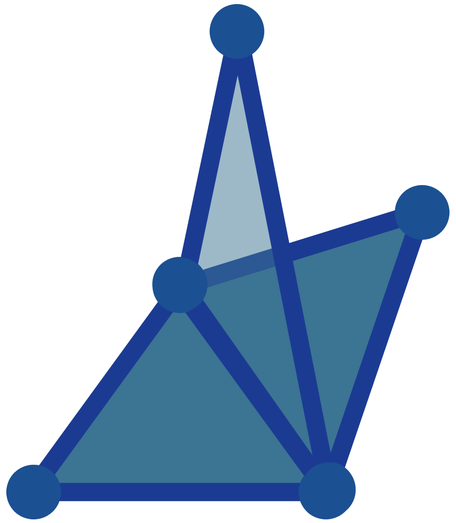


Manifolds on a computer

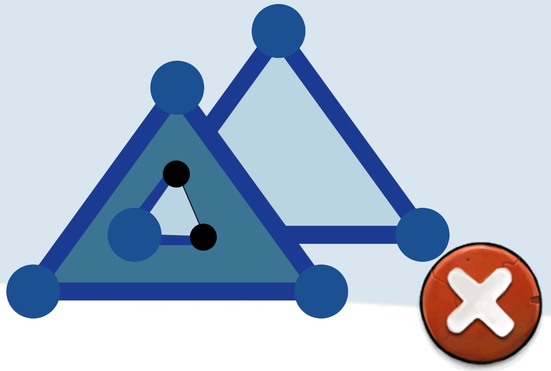


- Notion of **simplicial complex**
  - A simplicial complex $\mathcal{K}$ is a finite collection of non-empty simplices $\{\sigma_i\}$ such that
    - Every face τ of a simplex σ_i is also in $\mathcal{K}$
    - Any two simplices σ_i and σ_j intersect in a common face or not at all.

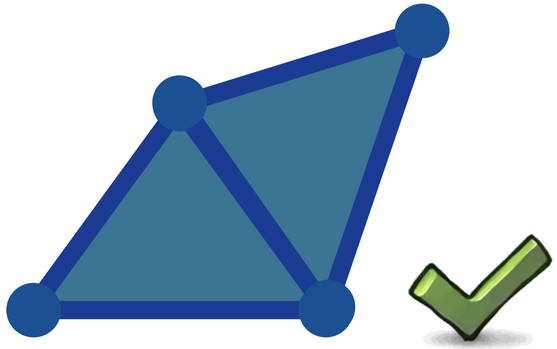


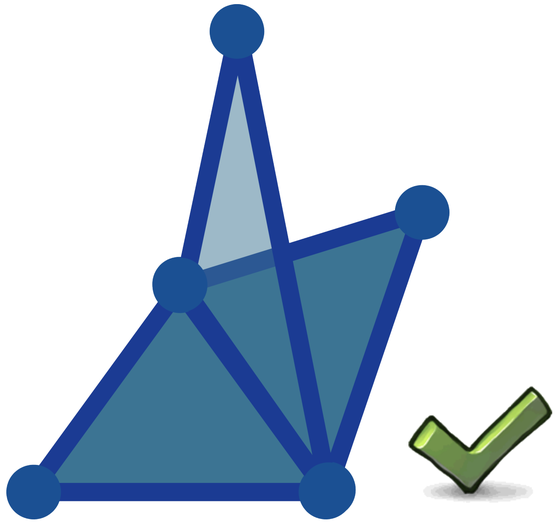


Manifolds on a computer



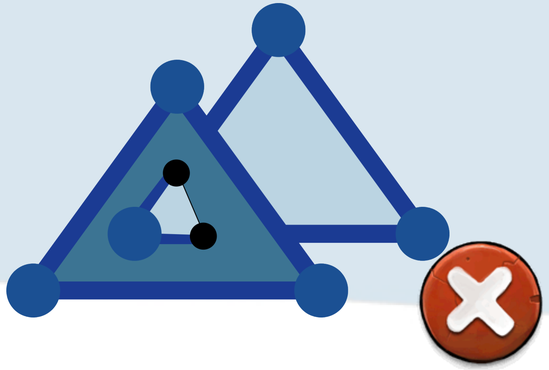
- Notion of **simplicial complex**
  - A simplicial complex $\mathcal{K}$ is a finite collection of non-empty simplices $\{\sigma_i\}$ such that
    - Every face τ of a simplex σ_i is also in $\mathcal{K}$
    - Any two simplices σ_i and σ_j intersect in a common face or not at all.

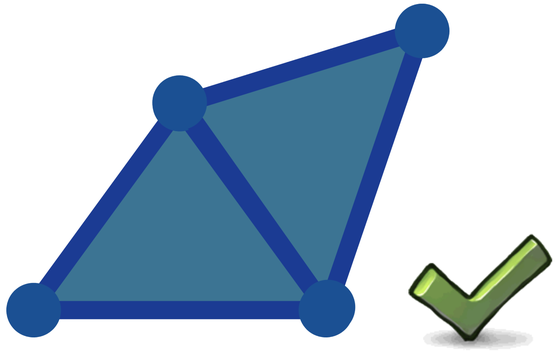


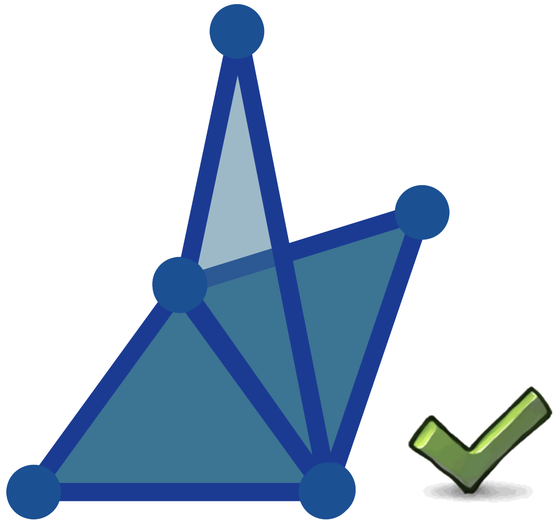


Manifolds on a computer

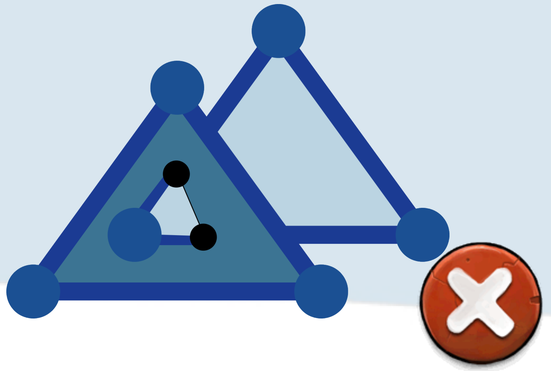
- Notion of **triangulation**



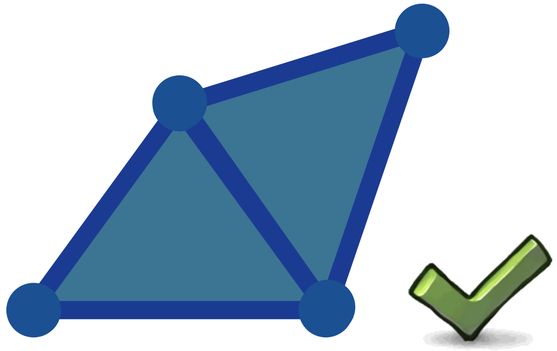


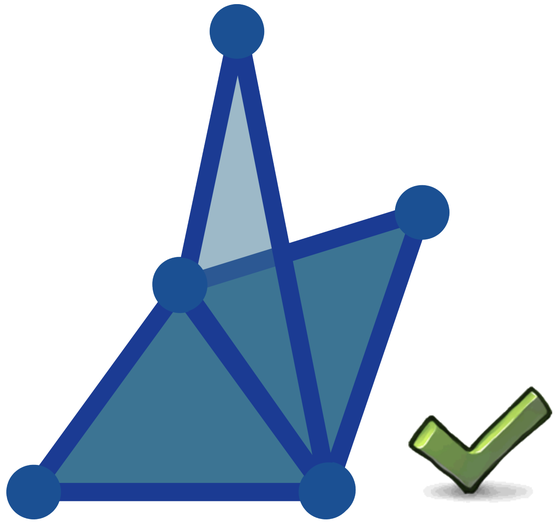


Manifolds on a computer

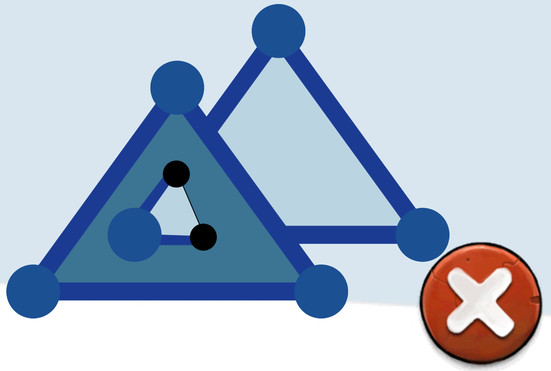


- Notion of **triangulation**
  - The triangulation of a d-manifold $\mathbb{M}$ is a simplicial complex $\mathcal{K}$ such that

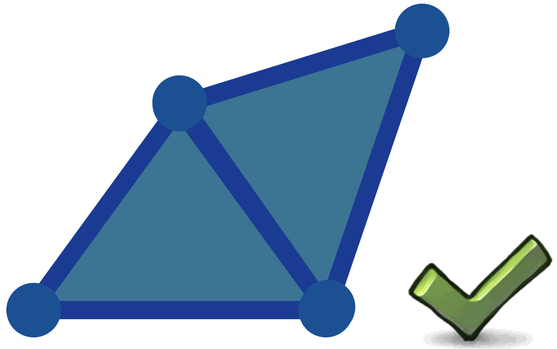


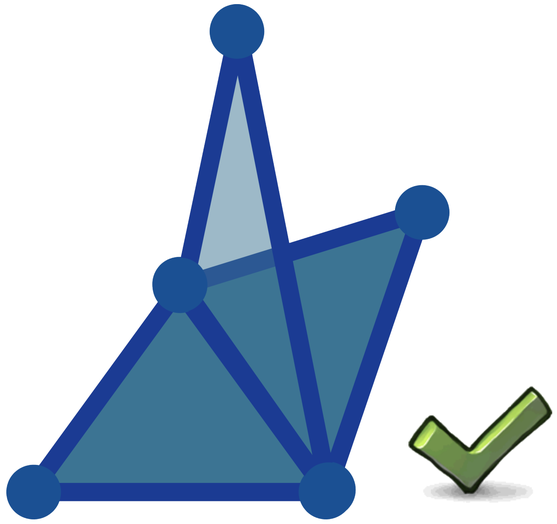


Manifolds on a computer

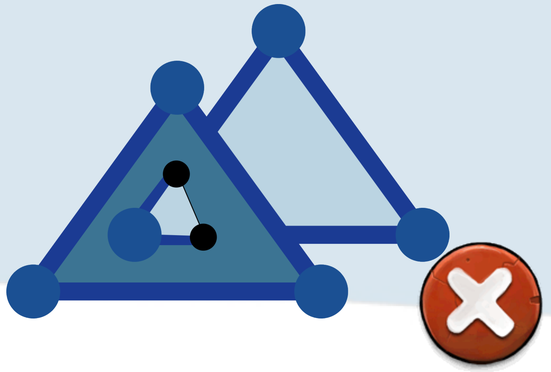


- Notion of **triangulation**
  - The triangulation of a d-manifold $\mathbb{M}$ is a simplicial complex $\mathcal{K}$ such that
    - The union $|\mathcal{K}| = \cup_{\sigma \in \mathcal{K}} \sigma$ of the simplices of $\mathcal{K}$ is homeomorphic to $\mathbb{M}$

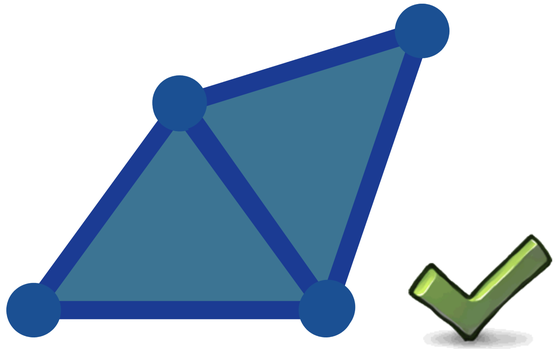


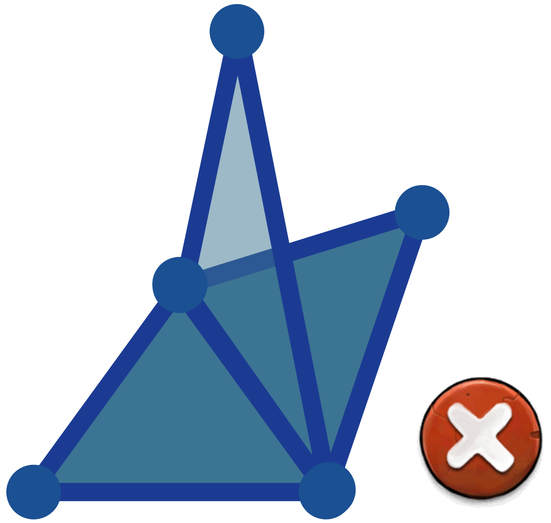


Manifolds on a computer

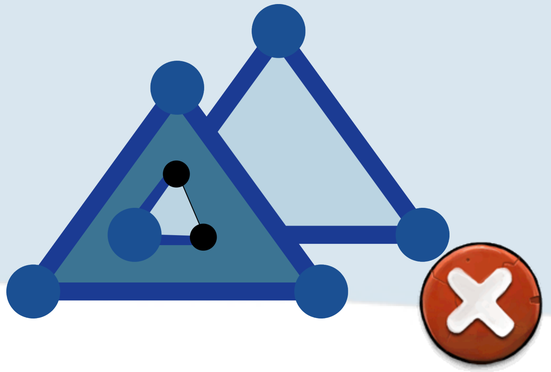


- Notion of **triangulation**
  - The triangulation of a d-manifold $\mathbb{M}$ is a simplicial complex $\mathcal{K}$ such that
    - The union $|\mathcal{K}| = \cup_{\sigma \in \mathcal{K}} \sigma$ of the simplices of $\mathcal{K}$ is homeomorphic to $\mathbb{M}$

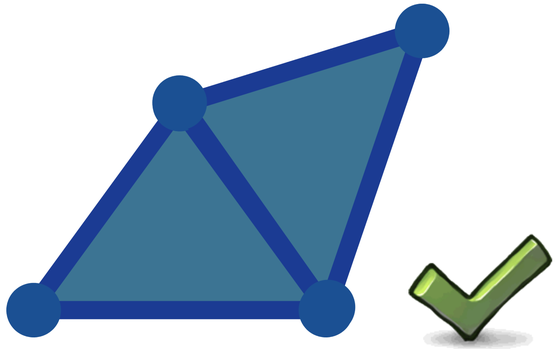


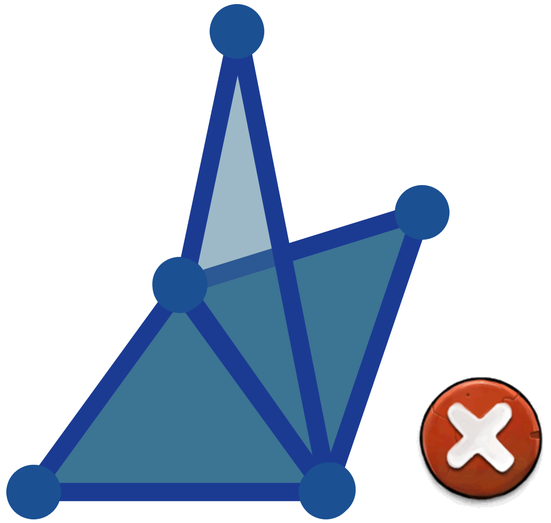


Manifolds on a computer

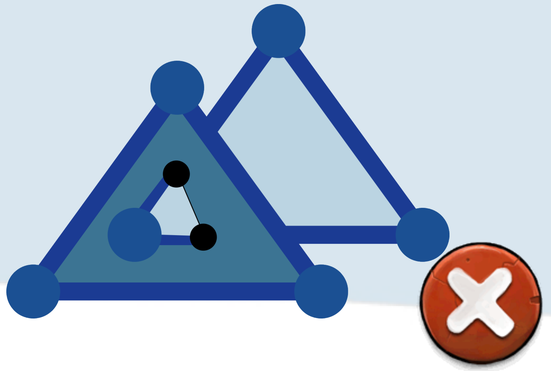


- Notion of **triangulation**
  - The triangulation of a d-manifold $\mathbb{M}$ is a simplicial complex $\mathcal{K}$ such that
    - The union $|\mathcal{K}| = \cup_{\sigma \in \mathcal{K}} \sigma$ of the simplices of $\mathcal{K}$ is homeomorphic to $\mathbb{M}$
  - Any open set of $|\mathcal{K}|$ is homeomorphic to $\mathbb{R}^d$

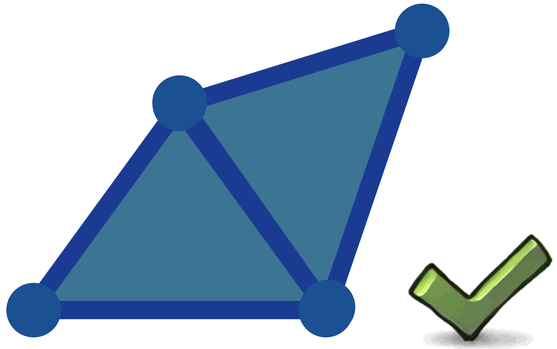




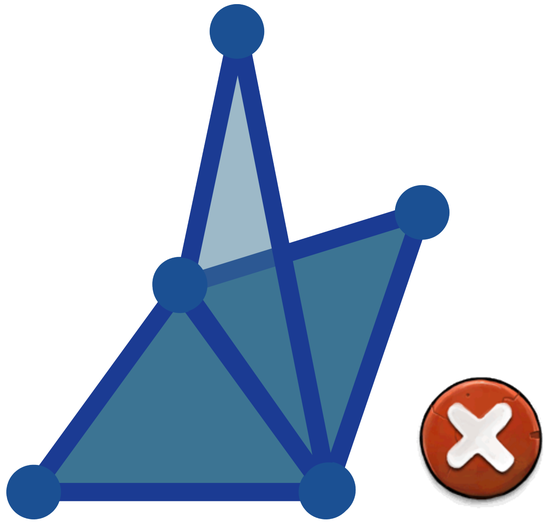
Manifolds on a computer



- Notion of **triangulation**
  - The triangulation of a d-manifold $\mathbb{M}$ is a simplicial complex $\mathcal{K}$ such that
    - The union $|\mathcal{K}| = \cup_{\sigma \in \mathcal{K}} \sigma$ of the simplices of $\mathcal{K}$ is homeomorphic to $\mathbb{M}$
  - Any open set of $|\mathcal{K}|$ is homeomorphic to $\mathbb{R}^d$

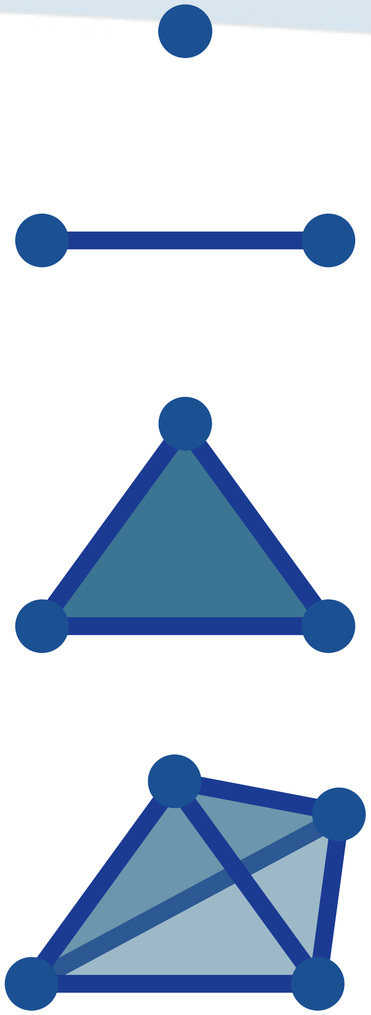


- 2-triangulation: *triangle mesh*
- 3-triangulation: *tetrahedral mesh*



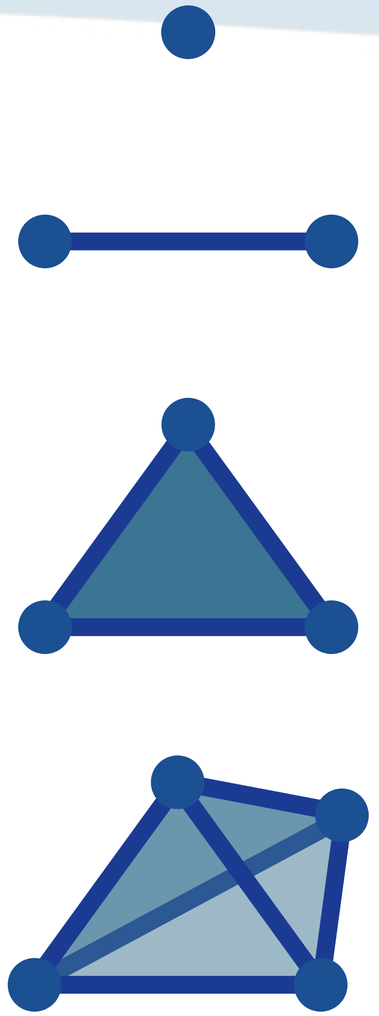
Manifolds on a computer

- How to represent that in memory?



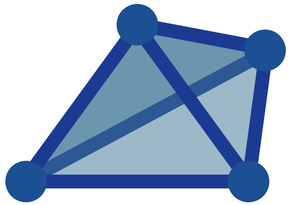
Manifolds on a computer

- How to represent that in memory?
  - Much trickier (arbitrary valence of the vertices)
  - The connectivity cannot be inferred by the geometry
  - It has to be explicitly stored



Manifolds on a computer

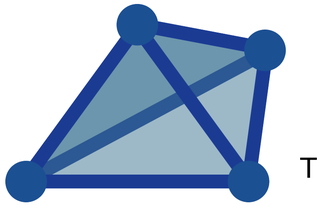
- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices



Manifolds on a computer

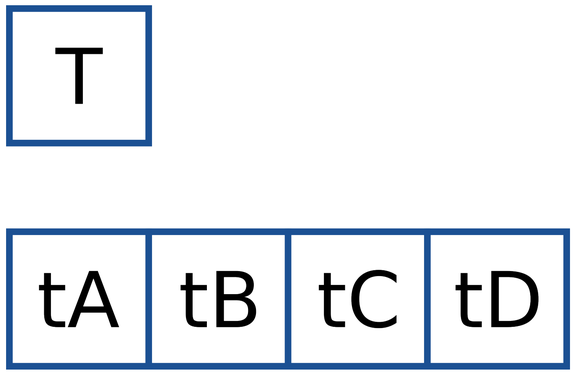
- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices

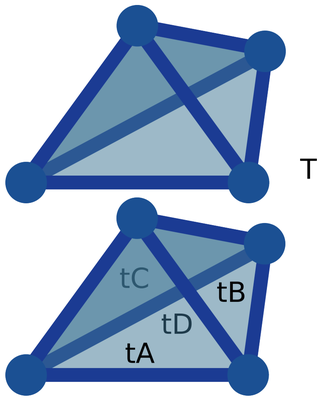
T



Manifolds on a computer

- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices





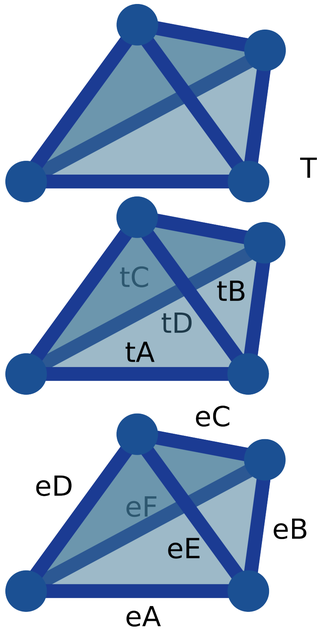
Manifolds on a computer

- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices

T

tA	tB	tC	tD

eA	eB	eC	eD	eE	eF



Manifolds on a computer

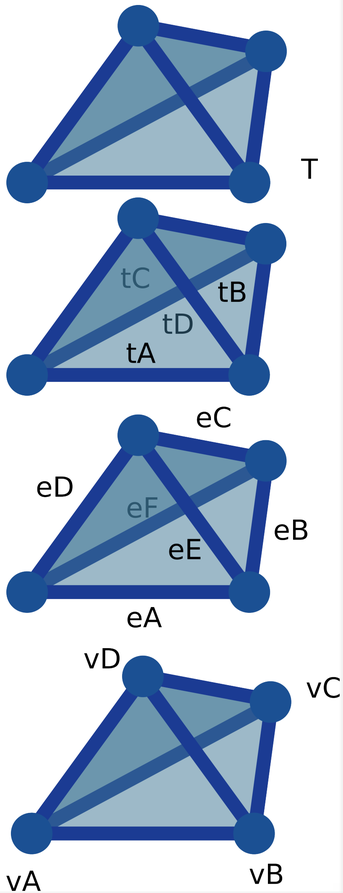
- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices

T

tA	tB	tC	tD

eA	eB	eC	eD	eE	eF

vA	vB	vC	vD



Manifolds on a computer

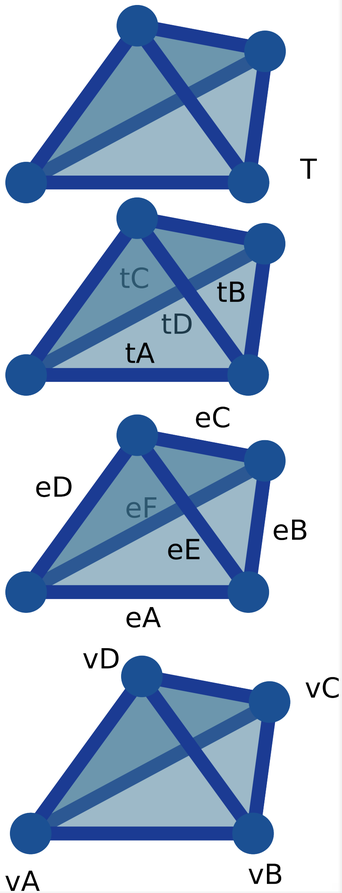
- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices
  - Attributes:
    - Assigned to the vertices

T

tA	tB	tC	tD

eA	eB	eC	eD	eE	eF

vA	vB	vC	vD



Manifolds on a computer

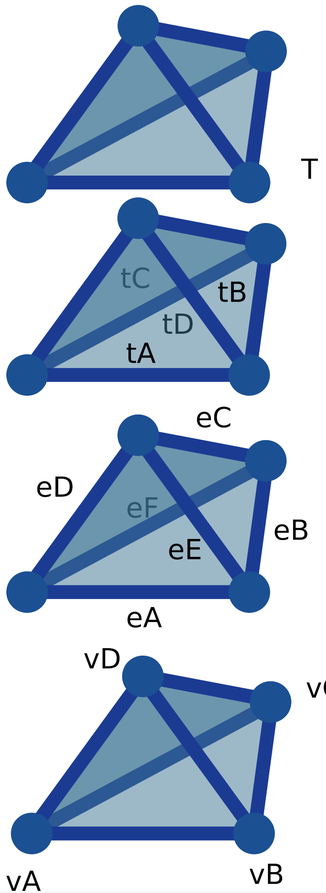
- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices
  - Attributes:
    - Assigned to the vertices
  - Each d-simplex stores pointers to its (d+1) faces of dimension (d-1)

T

tA	tB	tC	tD

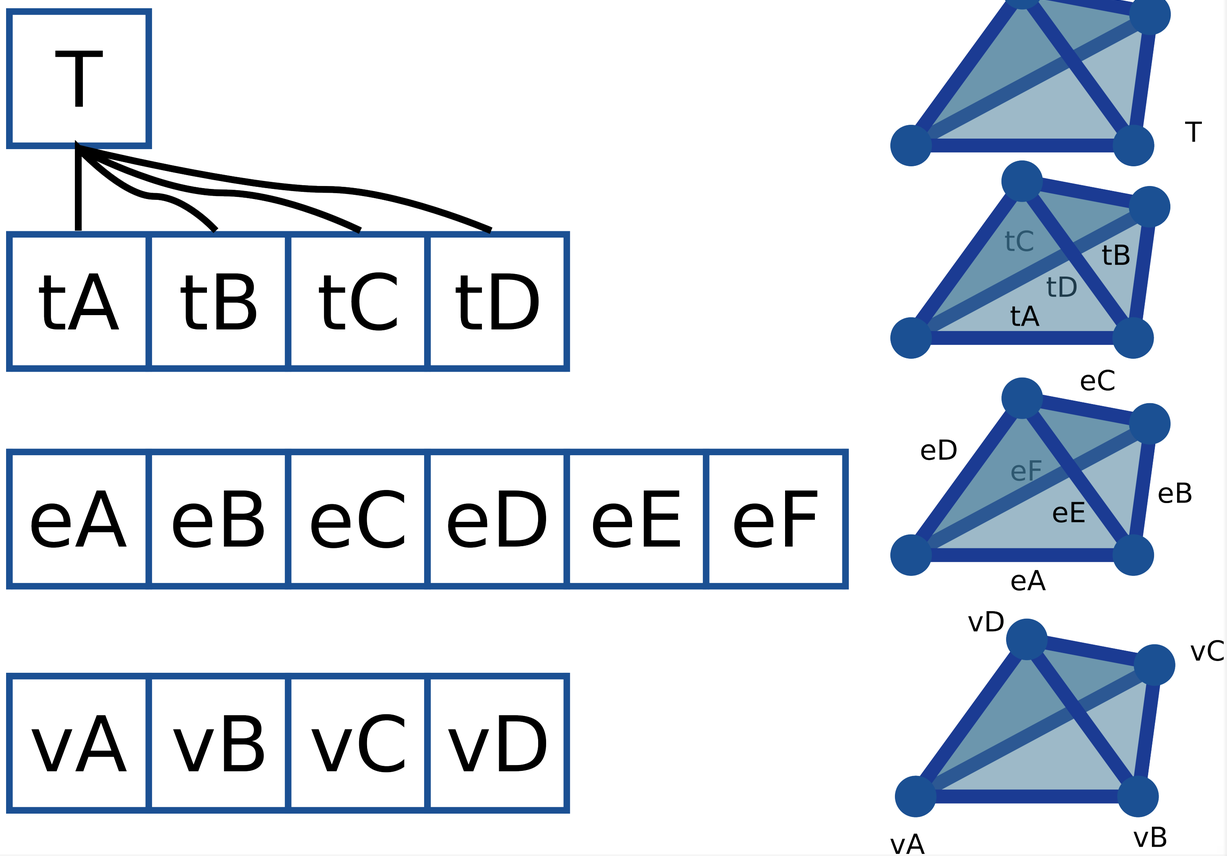
eA	eB	eC	eD	eE	eF

vA	vB	vC	vD



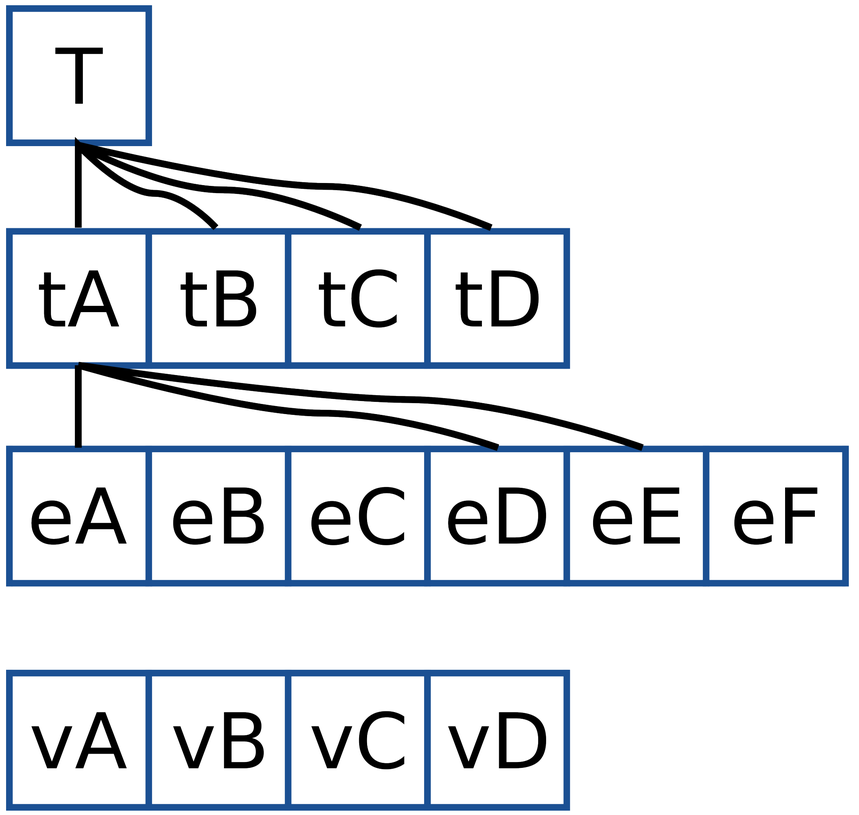
Manifolds on a computer

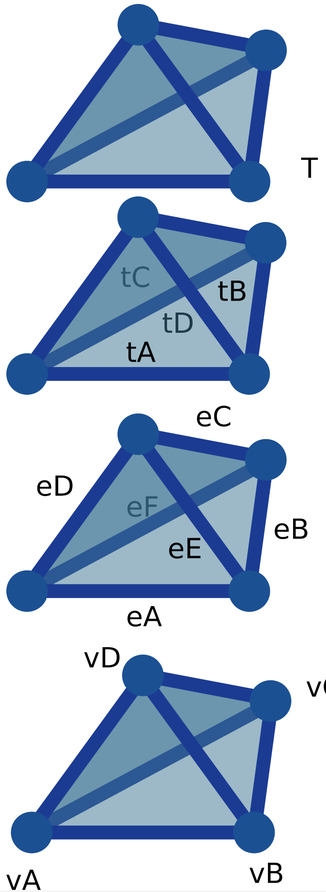
- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices
  - Attributes:
    - Assigned to the vertices
  - Each d-simplex stores pointers to its (d+1) faces of dimension (d-1)



Manifolds on a computer

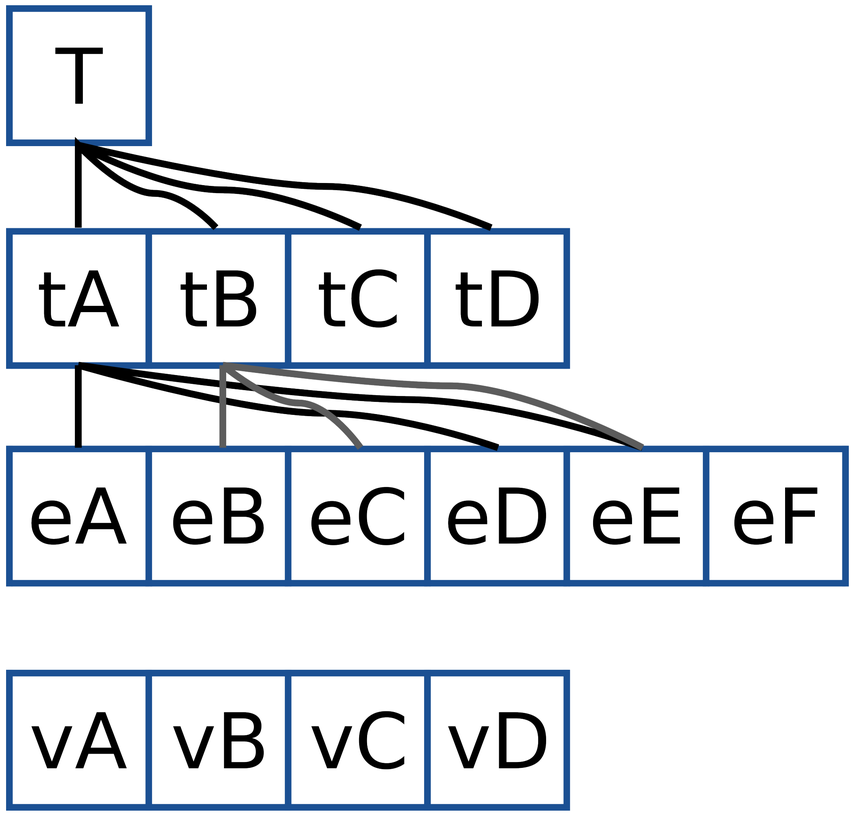
- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices
  - Attributes:
    - Assigned to the vertices
  - Each d-simplex stores pointers to its (d+1) faces of dimension (d-1)

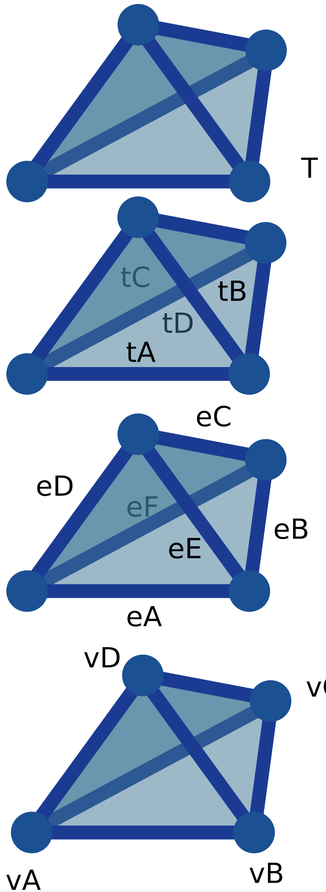




Manifolds on a computer

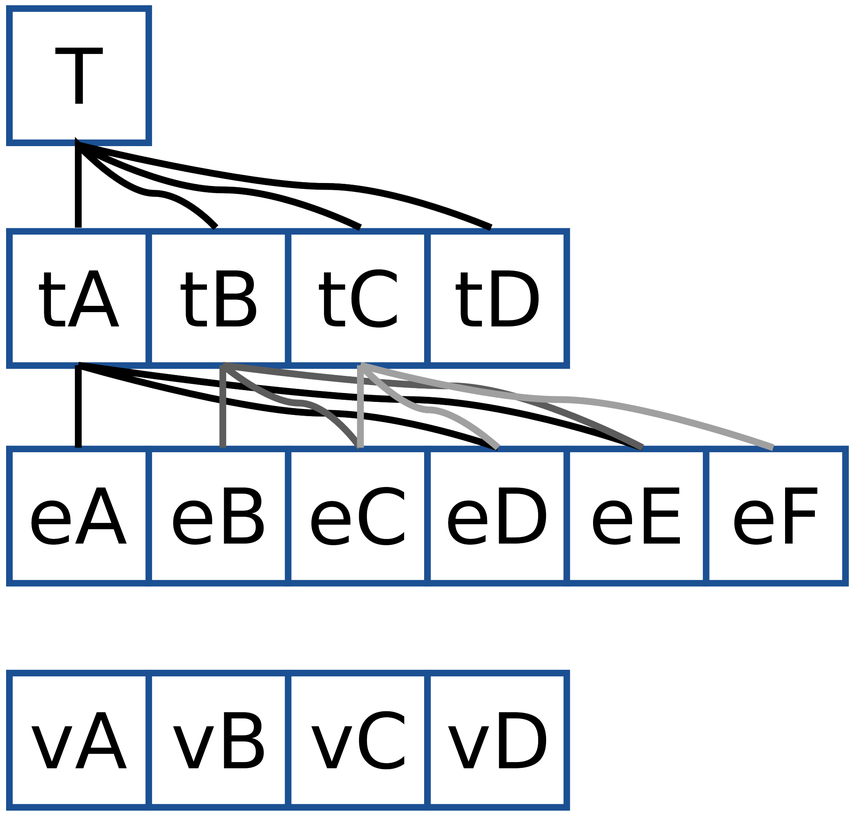
- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices
  - Attributes:
    - Assigned to the vertices
  - Each d-simplex stores pointers to its (d+1) faces of dimension (d-1)

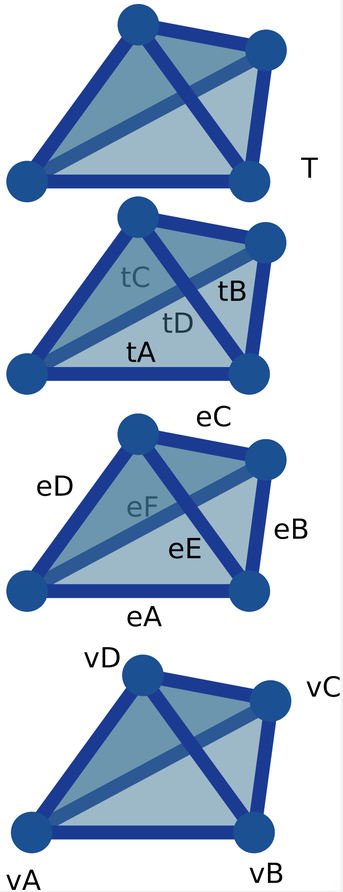




Manifolds on a computer

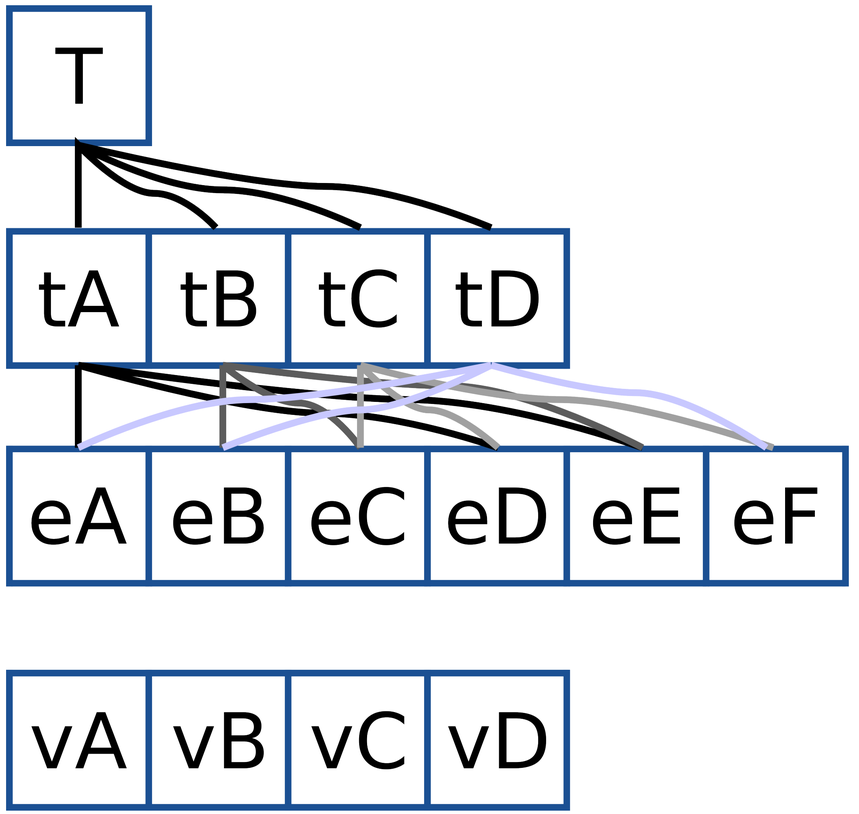
- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices
  - Attributes:
    - Assigned to the vertices
  - Each d-simplex stores pointers to its (d+1) faces of dimension (d-1)

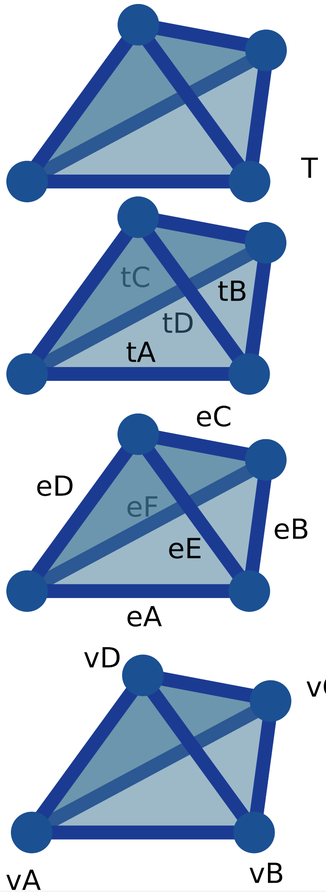




Manifolds on a computer

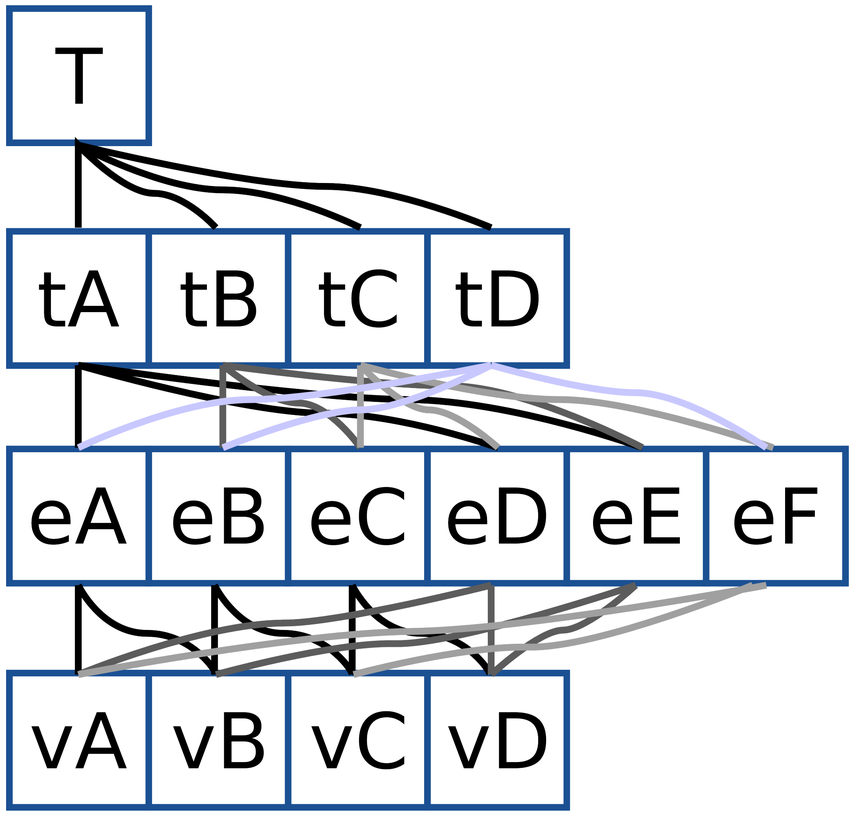
- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices
  - Attributes:
    - Assigned to the vertices
  - Each d-simplex stores pointers to its (d+1) faces of dimension (d-1)

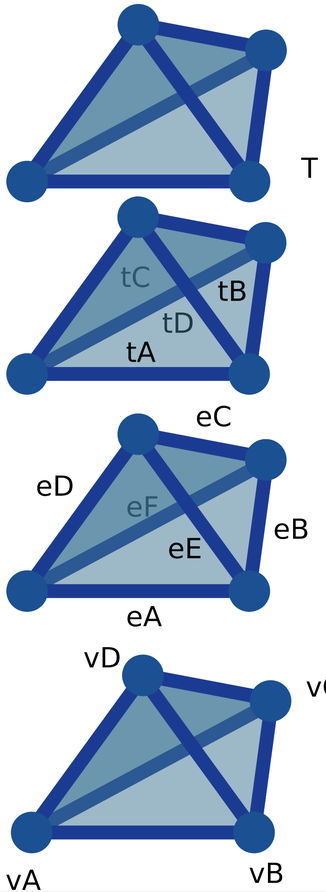




Manifolds on a computer

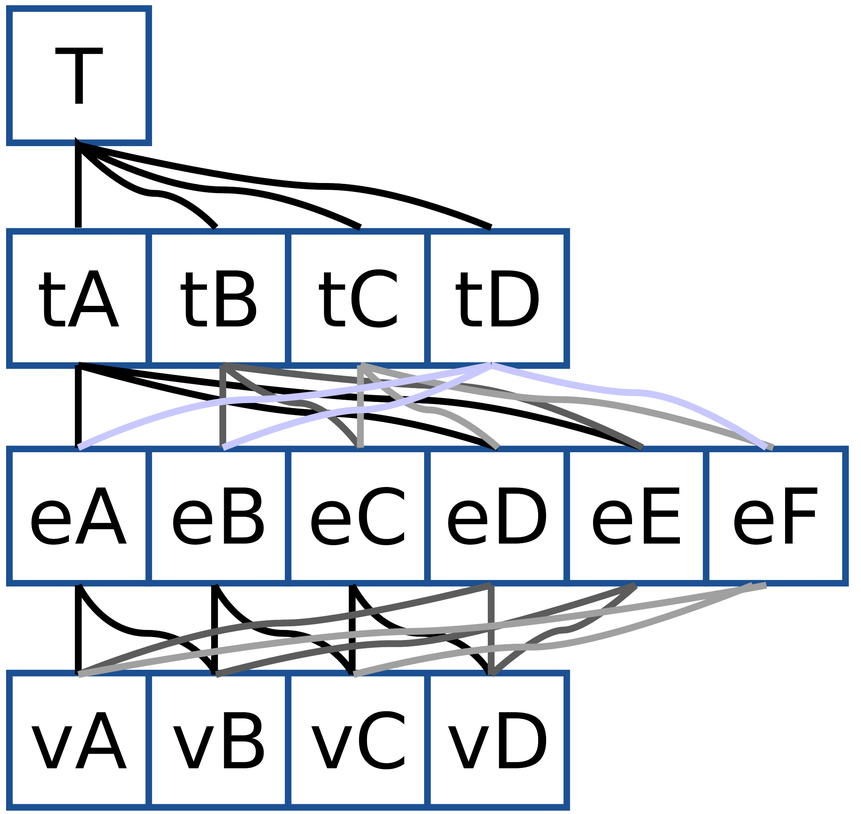
- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices
  - Attributes:
    - Assigned to the vertices
  - Each d-simplex stores pointers to its (d+1) faces of dimension (d-1)

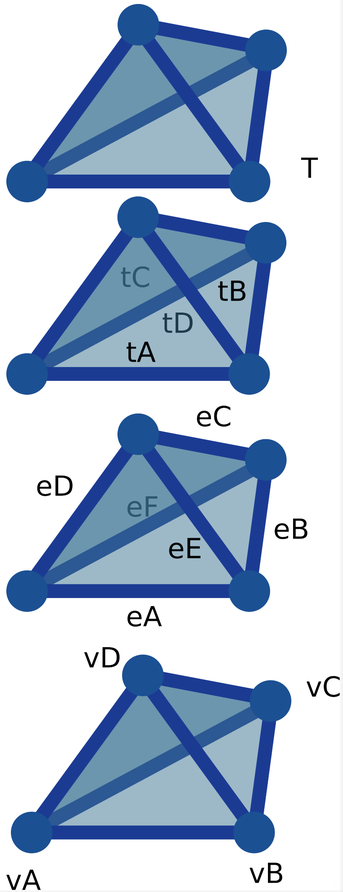




Manifolds on a computer

- General d-triangulation
  - List of d-simplices, (d-1)-simplices, … 0-simplices
  - Attributes:
    - Assigned to the vertices
  - Each d-simplex stores pointers to its (d+1) faces of dimension (d-1)
  - Navigation with bi-directional pointers





Consistency check

- There's a lot of room for bugs....

Consistency check

- There's a lot of room for bugs....

$$\chi = \sum_{i=0}^{d} (-1)^i |\{\sigma^i\}|$$

Consistency check

- There's a lot of room for bugs....

$$\chi = \sum_{i=0}^{d} (-1)^i |\{\sigma^i\}| = \sum_{i=0}^{d} (-1)^i \beta_i$$

Consistency check

- There's a lot of room for bugs....

$$\chi = \#_{vertices} - \#_{edges} + \#_{triangles} - \#_{tetrahedra} = \sum_{i=0}^{d} (-1)^i \beta_i$$

Consistency check

- There's a lot of room for bugs....

$$\chi = \#_{vertices} - \#_{edges} + \#_{triangles} - \#_{tetrahedra} = \sum_{i=0}^{d} (-1)^i \beta_i$$

$$\chi = \#_{connected\_components} - \#_{cycles} + \#_{voids}$$

Consistency check

- There's a lot of room for bugs....

$$\chi = \#_{vertices} - \#_{edges} + \#_{triangles} - \#_{tetrahedra} = \sum_{i=0}^{d} (-1)^i \beta_i$$

$$\chi = \#_{connected\_components} - \#_{cycles} + \#_{voids}$$

Consistency check

- There's a lot of room for bugs....

$$\chi = \#_{vertices} - \#_{edges} + \#_{triangles} - \#_{tetrahedra} = \sum_{i=0}^{d} (-1)^i \beta_i$$

$$\chi = \#_{connected\_components} - \#_{cycles} + \#_{voids}$$

$1, 0, 0, \chi = 1$

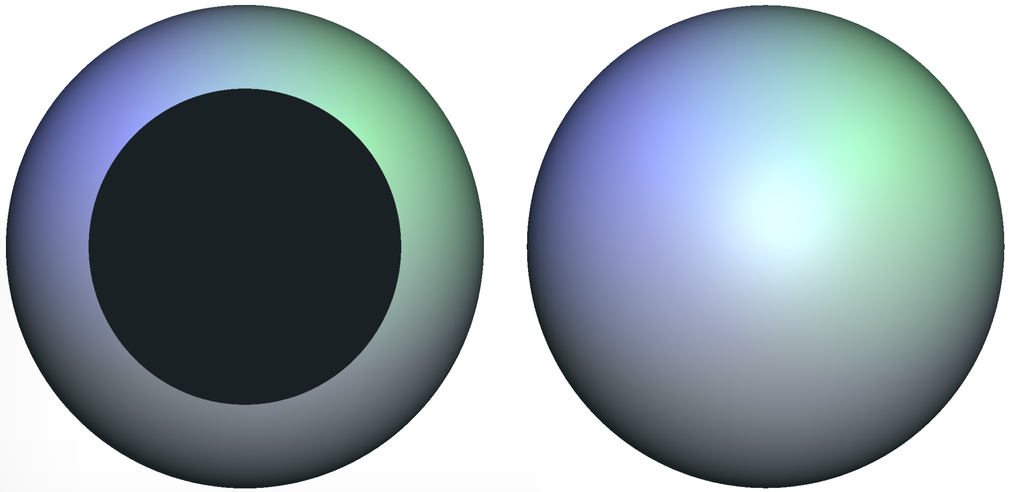
Consistency check

- There's a lot of room for bugs....

$$\chi = \#_{vertices} - \#_{edges} + \#_{triangles} - \#_{tetrahedra} = \sum_{i=0}^{d} (-1)^i \beta_i$$

$$\chi = \#_{connected\_components} - \#_{cycles} + \#_{voids}$$

$1, 0, 0, \chi = 1$



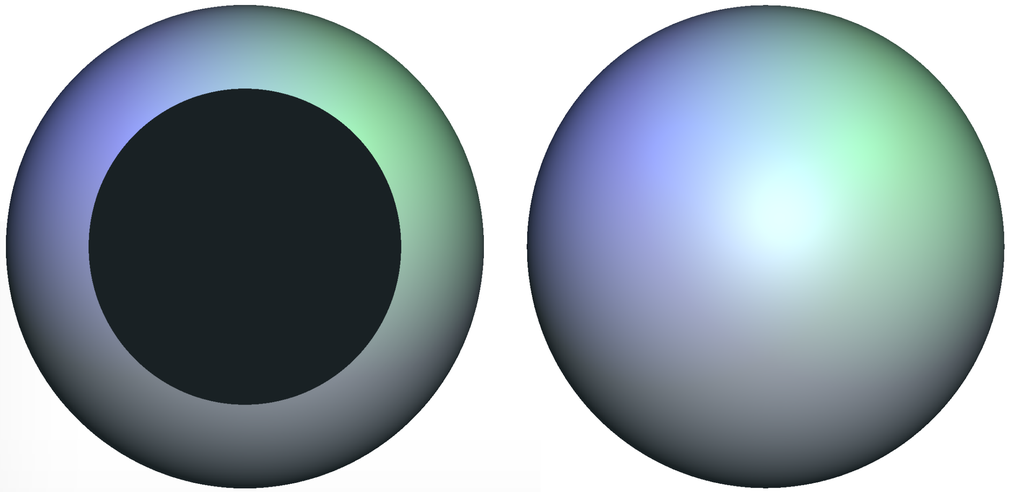
Consistency check

- There's a lot of room for bugs....

$$\chi = \#_{vertices} - \#_{edges} + \#_{triangles} - \#_{tetrahedra} = \sum_{i=0}^{d} (-1)^i \beta_i$$

$$\chi = \#_{connected\_components} - \#_{cycles} + \#_{voids}$$

$1, 0, 0, \chi = 1$ $1, 0, 1, \chi = 2$



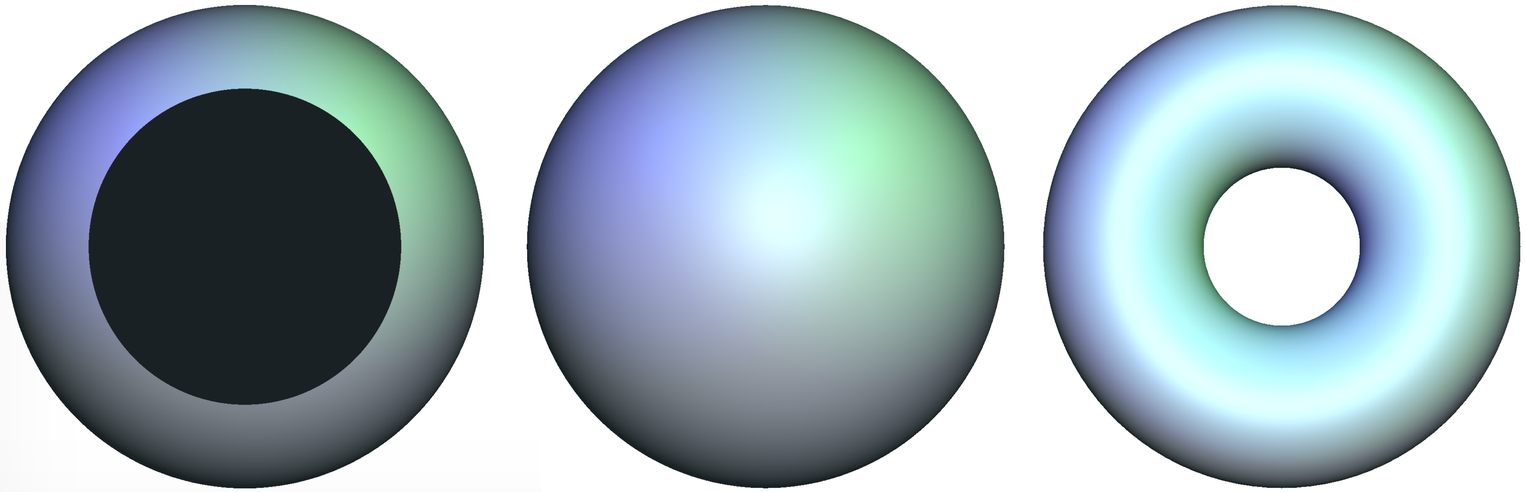
Consistency check

- There's a lot of room for bugs....

$$\chi = \#_{vertices} - \#_{edges} + \#_{triangles} - \#_{tetrahedra} = \sum_{i=0}^{d} (-1)^i \beta_i$$

$$\chi = \#_{connected\_components} - \#_{cycles} + \#_{voids}$$

$1, 0, 0, \chi = 1$ $1, 0, 1, \chi = 2$



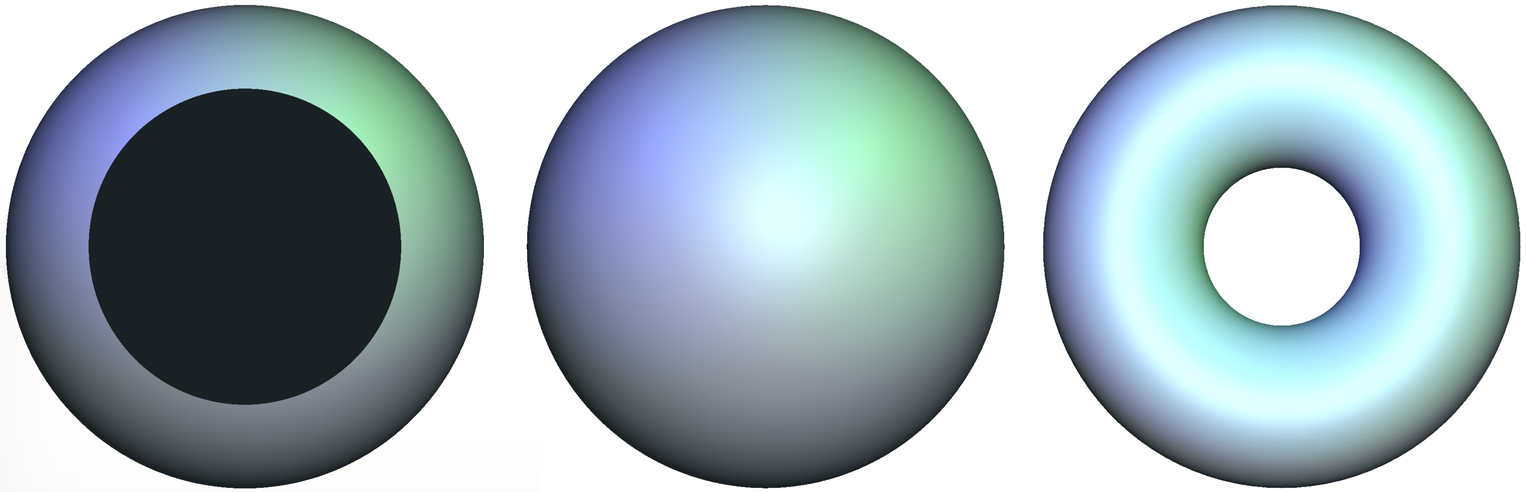
Consistency check

- There's a lot of room for bugs....

$$\chi = \#_{vertices} - \#_{edges} + \#_{triangles} - \#_{tetrahedra} = \sum_{i=0}^{d} (-1)^i \beta_i$$

$$\chi = \#_{connected\_components} - \#_{cycles} + \#_{voids}$$

$1, 0, 0, \chi = 1$ $1, 0, 1, \chi = 2$ $1, 2, 1, \chi = 0$



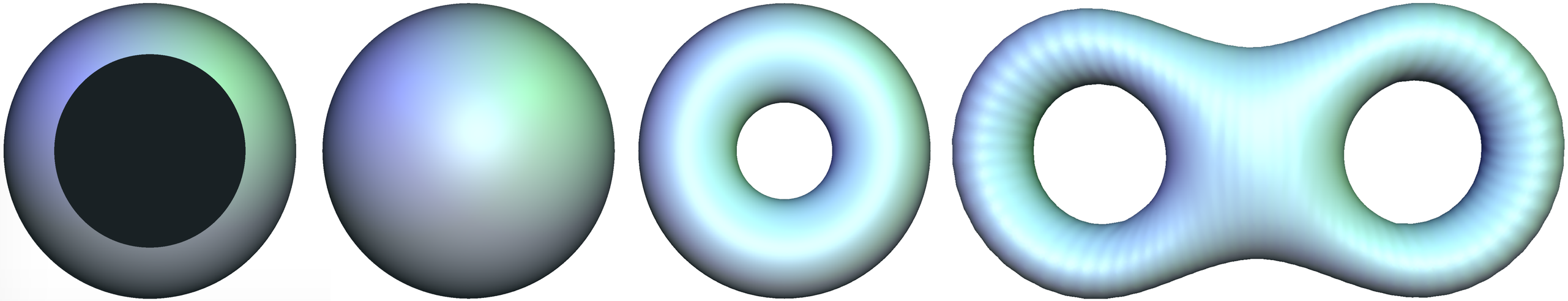
Consistency check

- There's a lot of room for bugs....

$$\chi = \#_{vertices} - \#_{edges} + \#_{triangles} - \#_{tetrahedra} = \sum_{i=0}^{d} (-1)^i \beta_i$$

$$\chi = \#_{connected\_components} - \#_{cycles} + \#_{voids}$$

$1, 0, 0, \chi = 1$ $1, 0, 1, \chi = 2$ $1, 2, 1, \chi = 0$



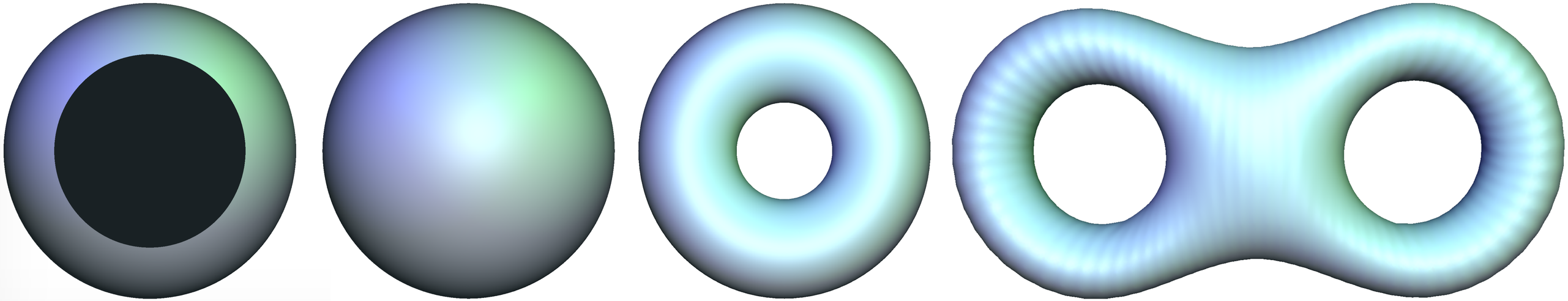
Consistency check

- There's a lot of room for bugs....

$$\chi = \#_{vertices} - \#_{edges} + \#_{triangles} - \#_{tetrahedra} = \sum_{i=0}^{d} (-1)^i \beta_i$$

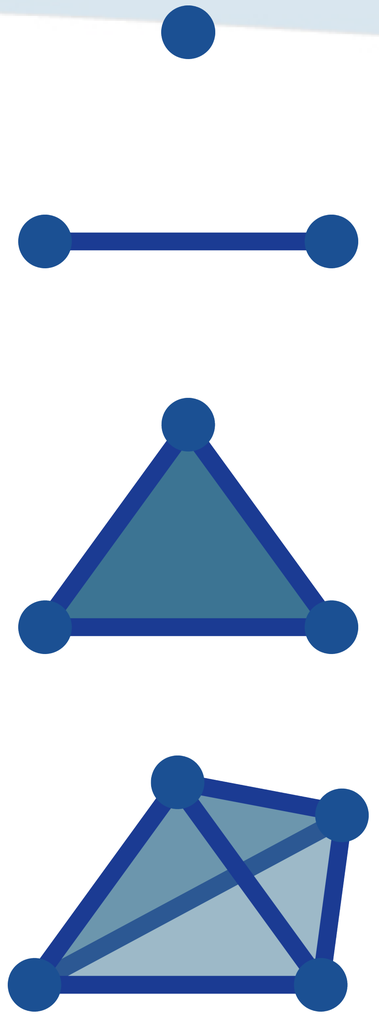
$$\chi = \#_{connected\_components} - \#_{cycles} + \#_{voids}$$

$1, 0, 0, \chi = 1$ $1, 0, 1, \chi = 2$ $1, 2, 1, \chi = 0$ $1, 4, 1, \chi = -2$



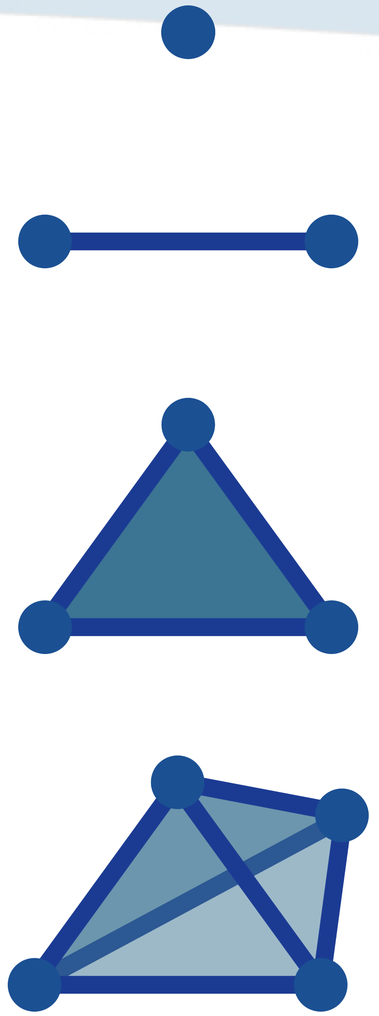
Manifold spaces on a computer

- Notion of **triangulation** $\mathcal{T}$



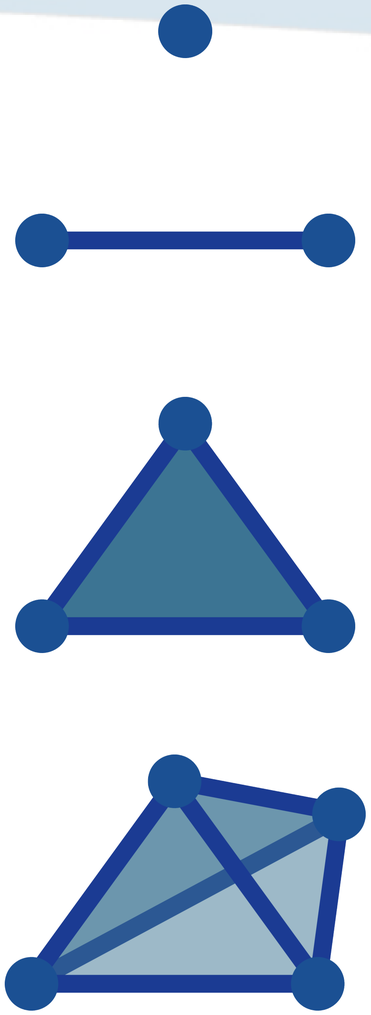
Manifold spaces on a computer

- Notion of **triangulation** $\mathcal{T}$
  - Set of **vertices** (samples)
    - With embedding functions to $\mathbb{R}^n$



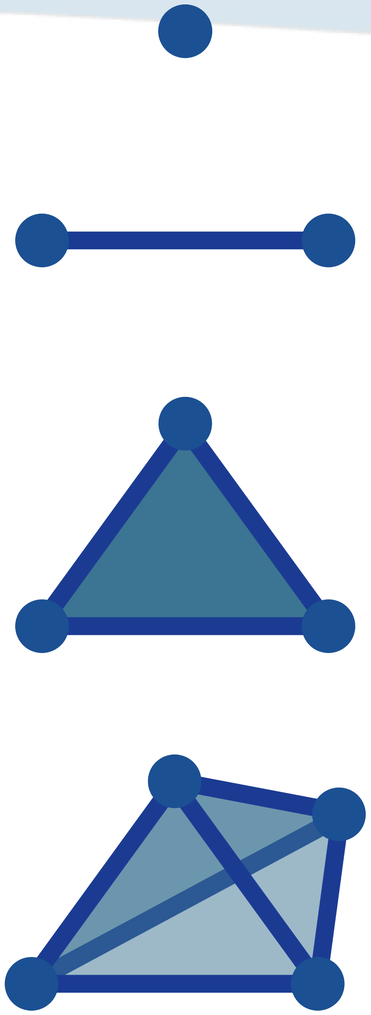
Manifold spaces on a computer

- Notion of **triangulation** $\mathcal{T}$
  - Set of **vertices** (samples)
    - With embedding functions to $\mathbb{R}^n$
  - **Connectivity**
    - **Explicitly** represented
    - Enumeration of simplices and their faces



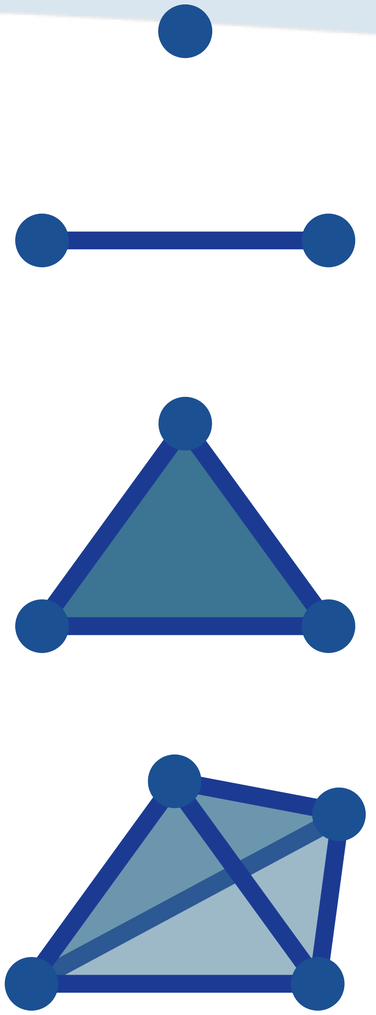
Manifold spaces on a computer

- Notion of **triangulation** $\mathcal{T}$
  - Set of **vertices** (samples)
    - With embedding functions to $\mathbb{R}^n$
  - **Connectivity**
    - **Explicitly** represented
    - Enumeration of simplices and their faces
  - What's missing?



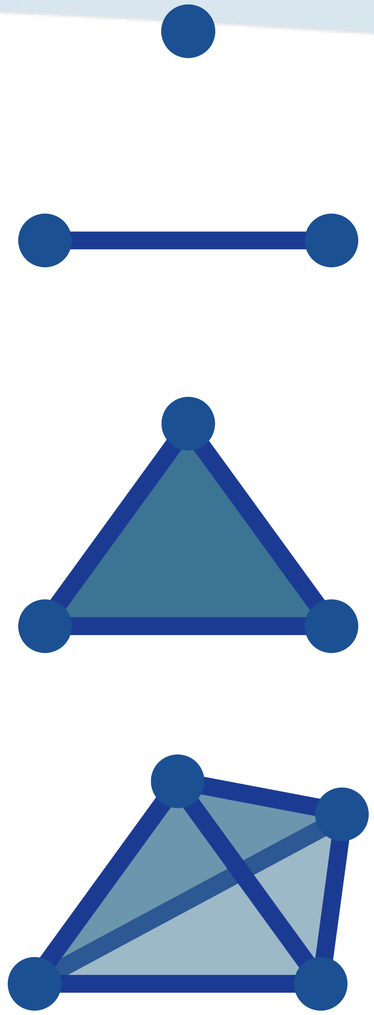
Manifold spaces on a computer

- Notion of **triangulation** $\mathcal{T}$
  - Set of **vertices** (samples)
    - With embedding functions to $\mathbb{R}^n$
  - **Connectivity**
    - **Explicitly** represented
    - Enumeration of simplices and their faces
  - What's missing?
    - The **interpolation** scheme



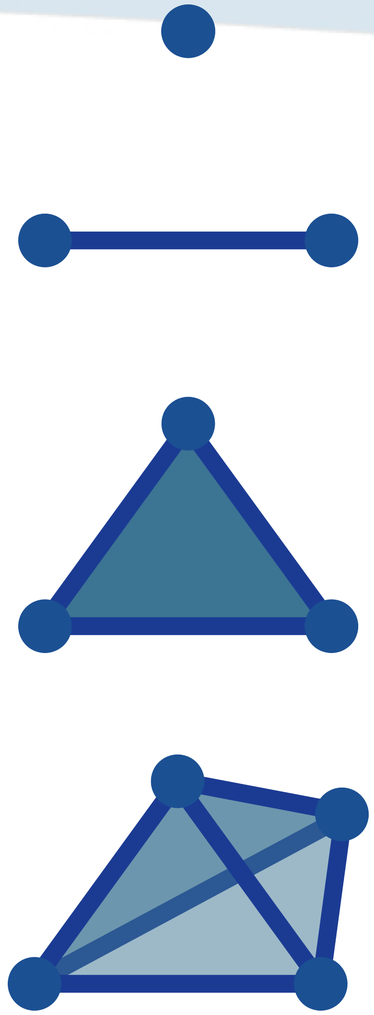
Interpolants for triangulations

- Like regular grids, several options



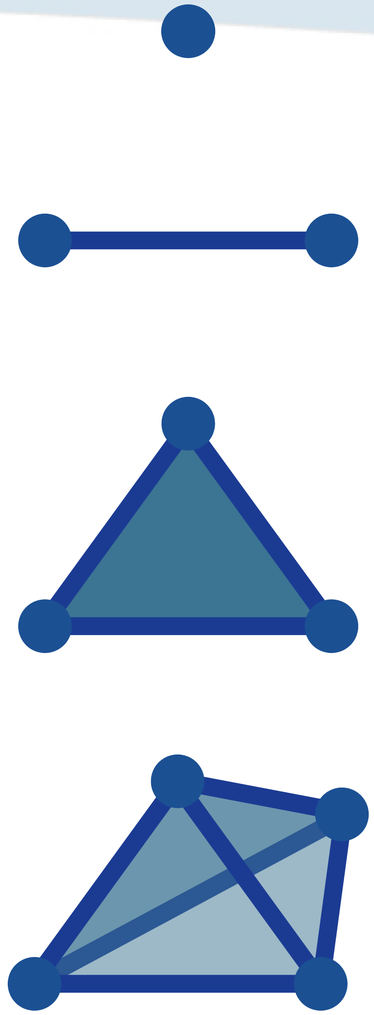
Interpolants for triangulations

- Like regular grids, several options
  - Piecewise constant
  - Piecewise linear
  - Piecewise polynomials,
  - etc.



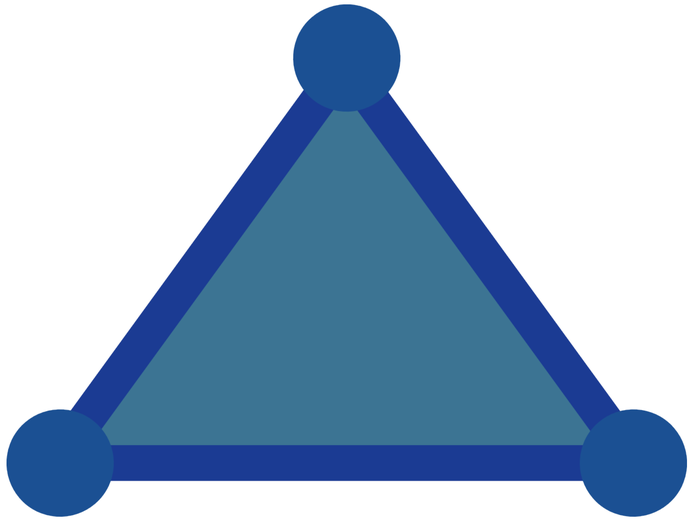
Interpolants for triangulations

- Like regular grids, several options
  - Piecewise constant
  - **Piecewise linear**
  - Piecewise polynomials,
  - etc.



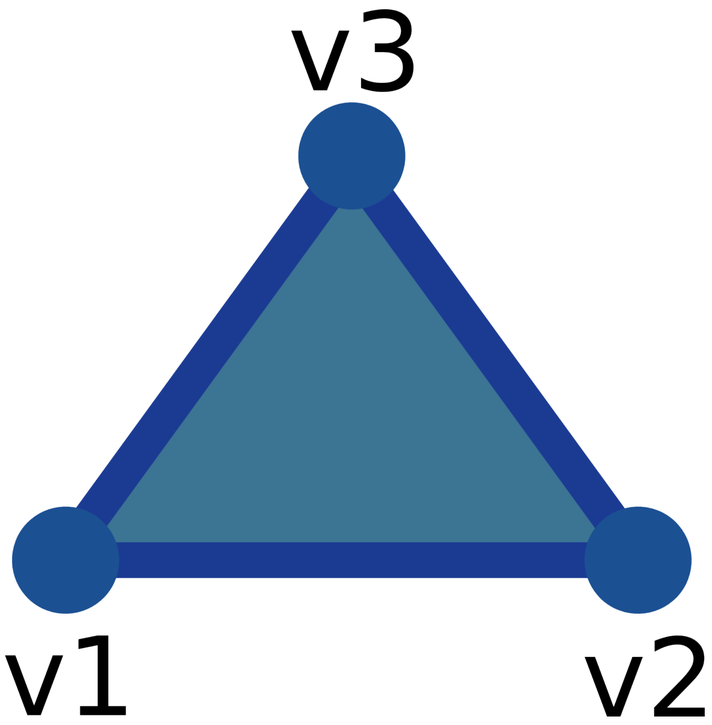
Piecewise linear interpolant on triangulations

- Interpolation as a boundary value problem



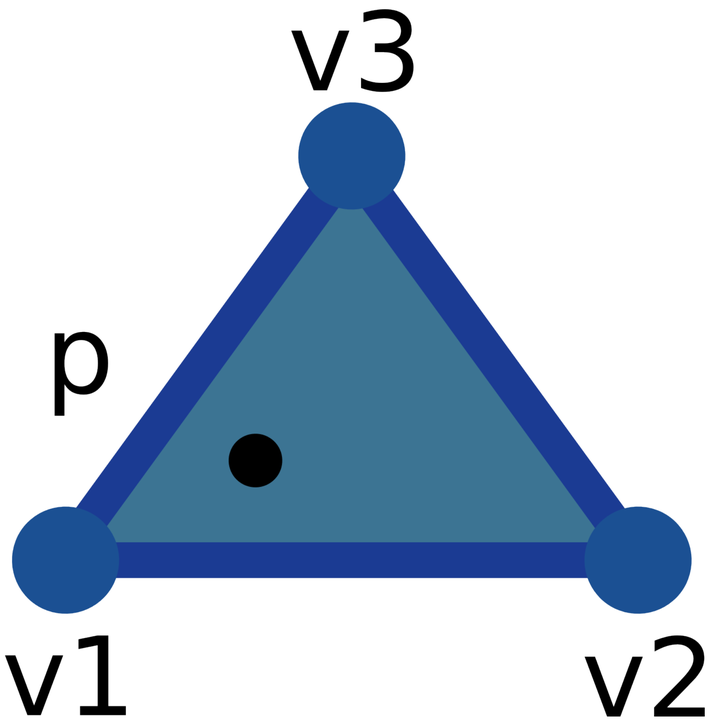
Piecewise linear interpolant on triangulations

- Interpolation as a boundary value problem
  - Given a simplex and its vertices



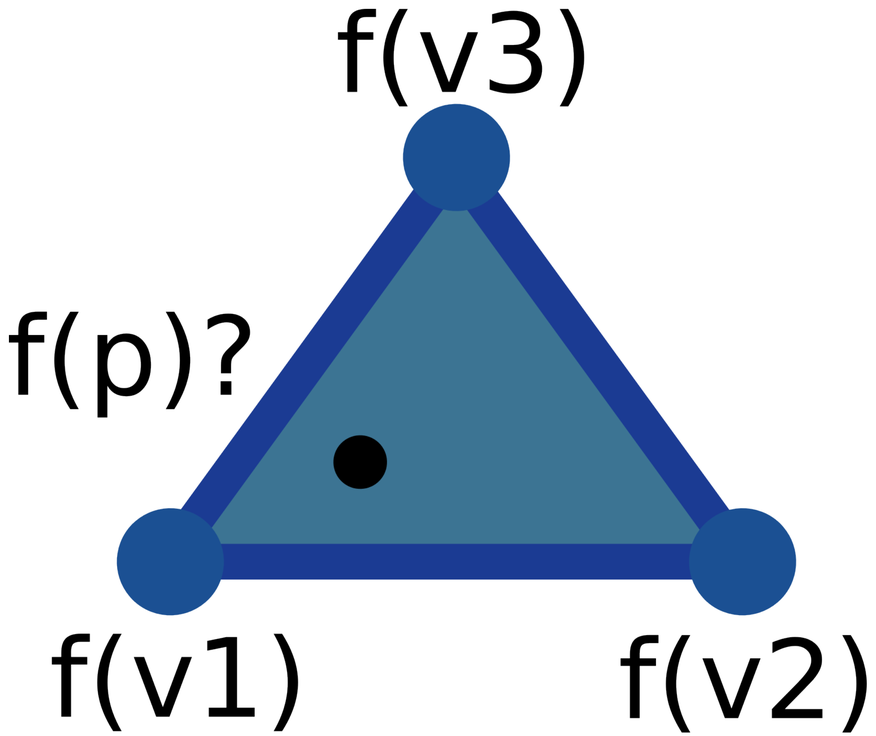
Piecewise linear interpolant on triangulations

- Interpolation as a boundary value problem
  - Given a simplex and its vertices
  - Express the value of any point



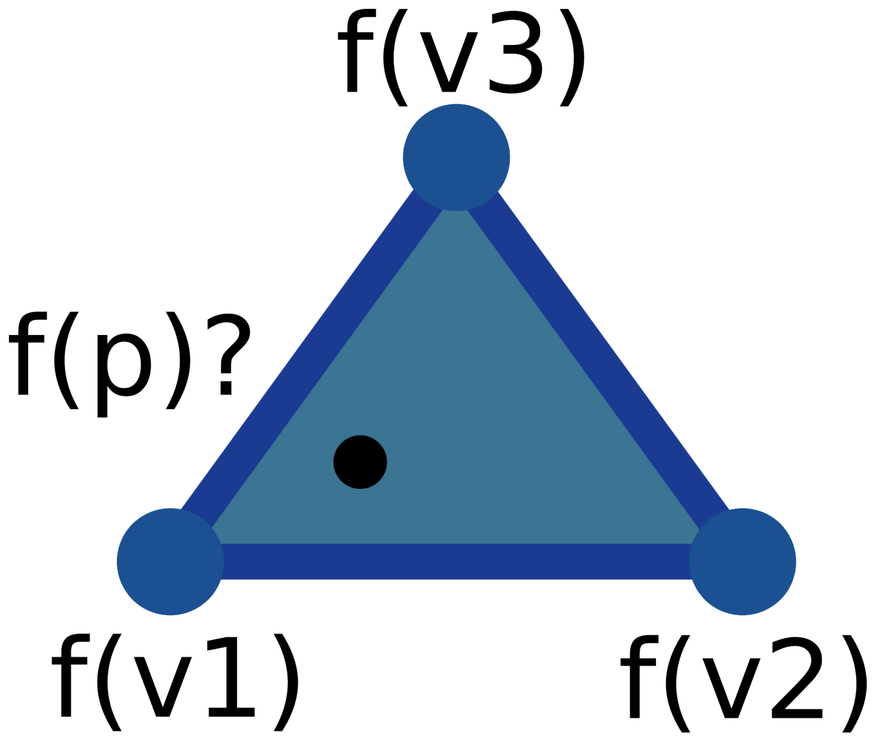
Piecewise linear interpolant on triangulations

- Interpolation as a boundary value problem
  - Given a simplex and its vertices
  - Express the value of any point
  - As a linear combination of those of the vertices



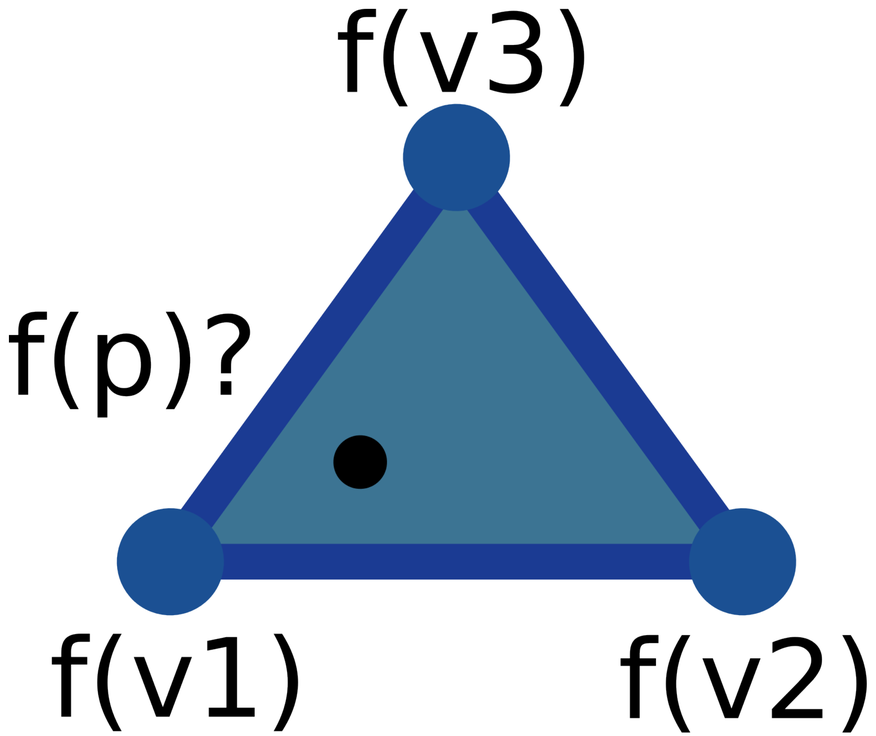
Piecewise linear interpolant on triangulations

- Interpolation as a boundary value problem
  - Given a simplex and its vertices
  - Express the value of any point
  - As a linear combination of those of the vertices
- Notion of barycentric coordinates:
  - $f(p) = \alpha_1 f(v_1) + \alpha_2 f(v_2) + \alpha_3 f(v_3)$



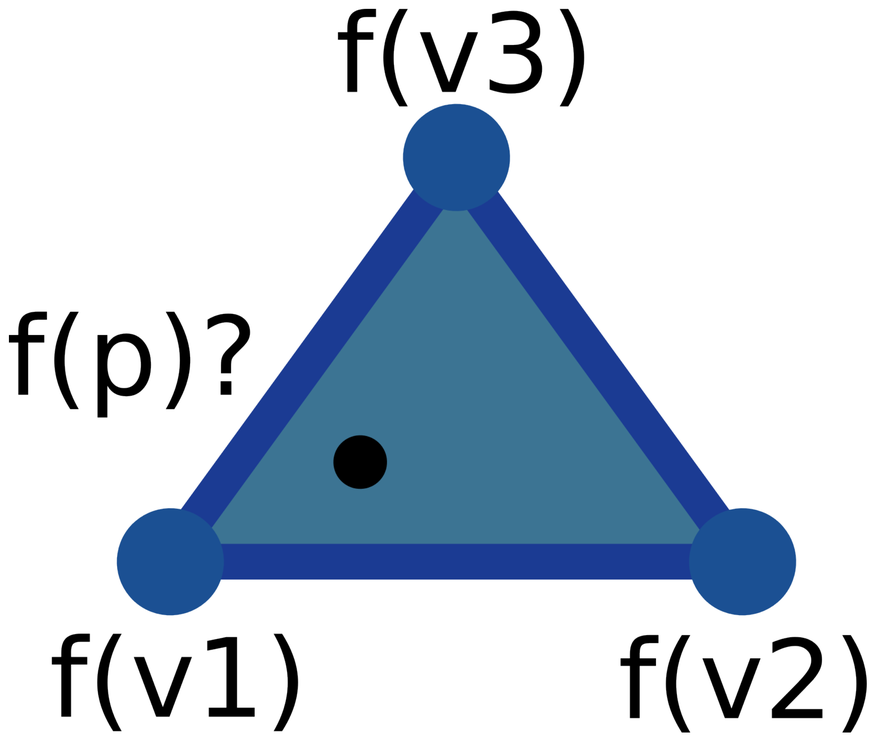
Barycentric coordinates

- There exists many forms
  - With specific properties



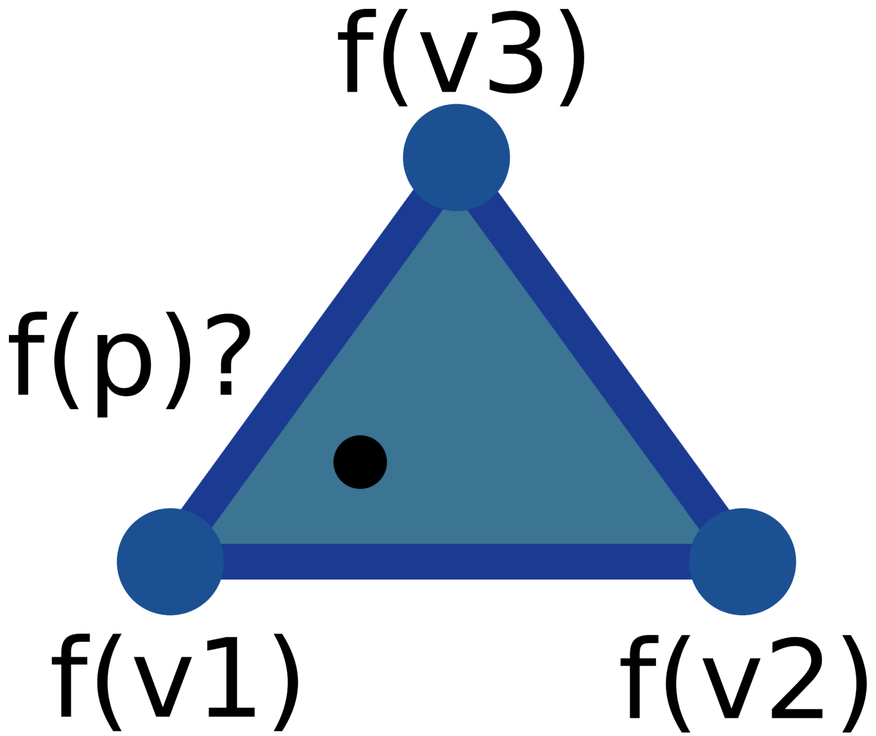
Barycentric coordinates

- There exists many forms
  - With specific properties
- We want to produce linear interpolations



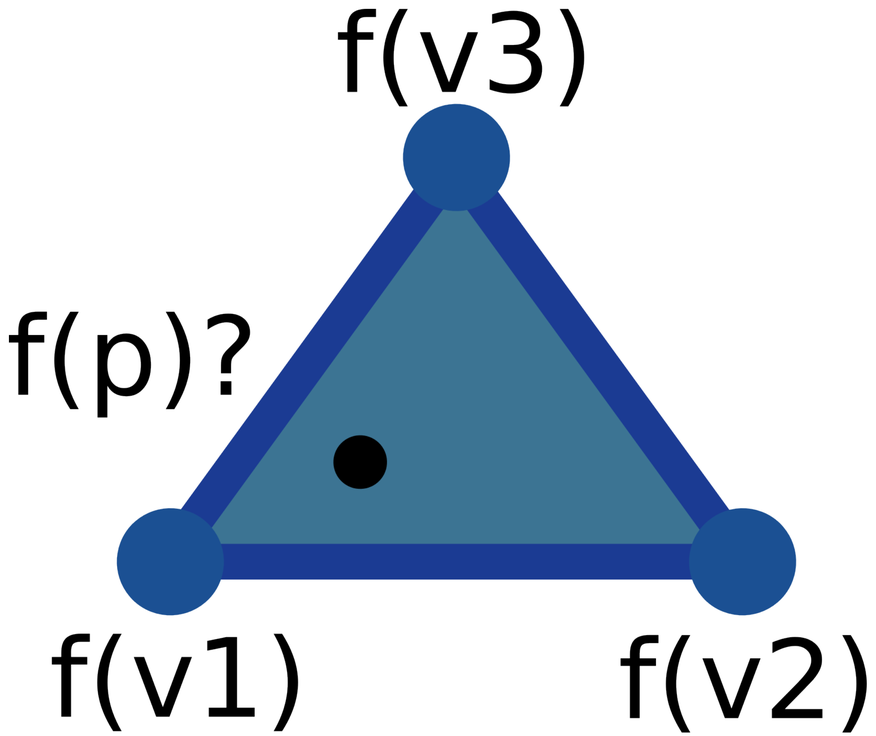
Barycentric coordinates

- There exists many forms
  - With specific properties
- We want to produce linear interpolations
  - $\alpha_1 + \alpha_2 + \alpha_3 = 1$



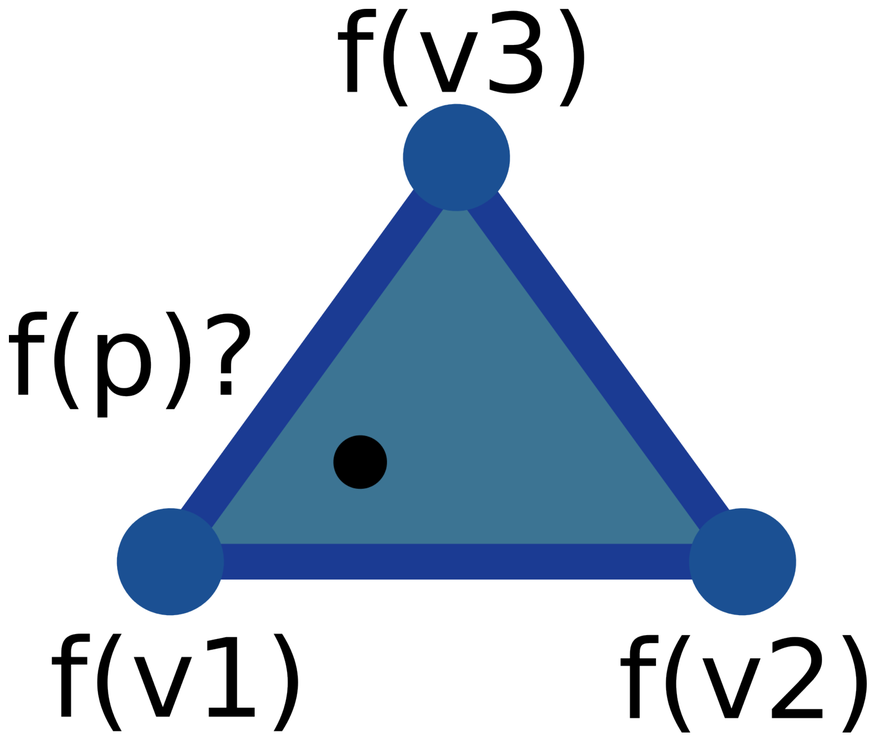
Barycentric coordinates

- There exists many forms
  - With specific properties
- We want to produce linear interpolations
  - $\alpha_1 + \alpha_2 + \alpha_3 = 1$
- Notion of barycentric coordinates:
  - $f(p) = \alpha_1 f(v_1) + \alpha_2 f(v_2) + \alpha_3 f(v_3)$



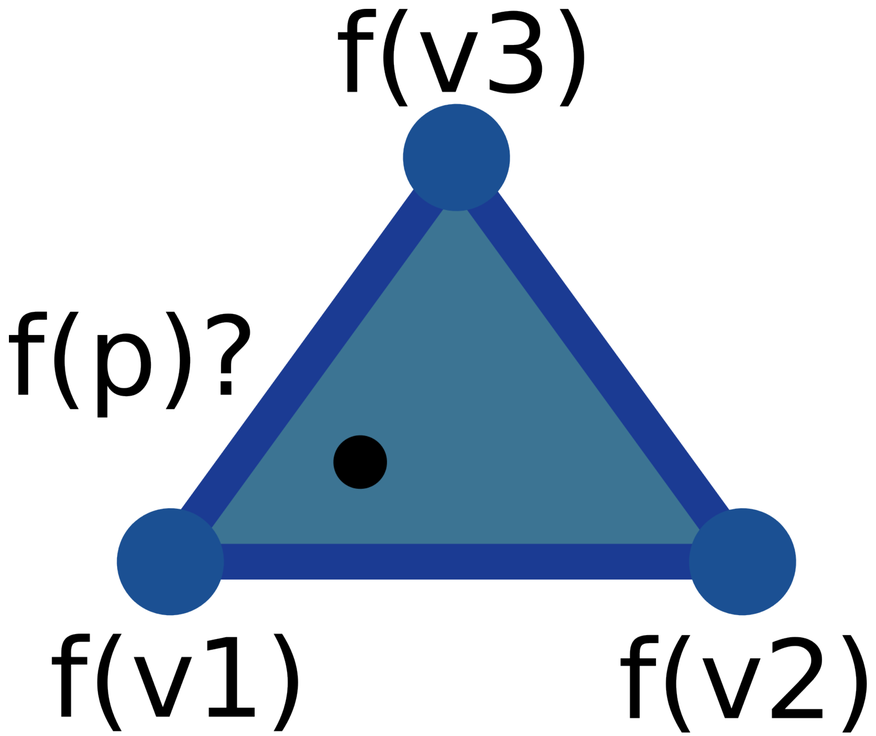
Barycentric coordinates

- There exists many forms
  - With specific properties
- We want to produce linear interpolations
  - $\alpha_1 + \alpha_2 + \alpha_3 = 1$
- Notion of barycentric coordinates:
  - $f(p) = \alpha_1 f(v_1) + \alpha_2 f(v_2) + \alpha_3 f(v_3)$
  - Must hold for any $f : \mathcal{T} \to \mathbb{R}$



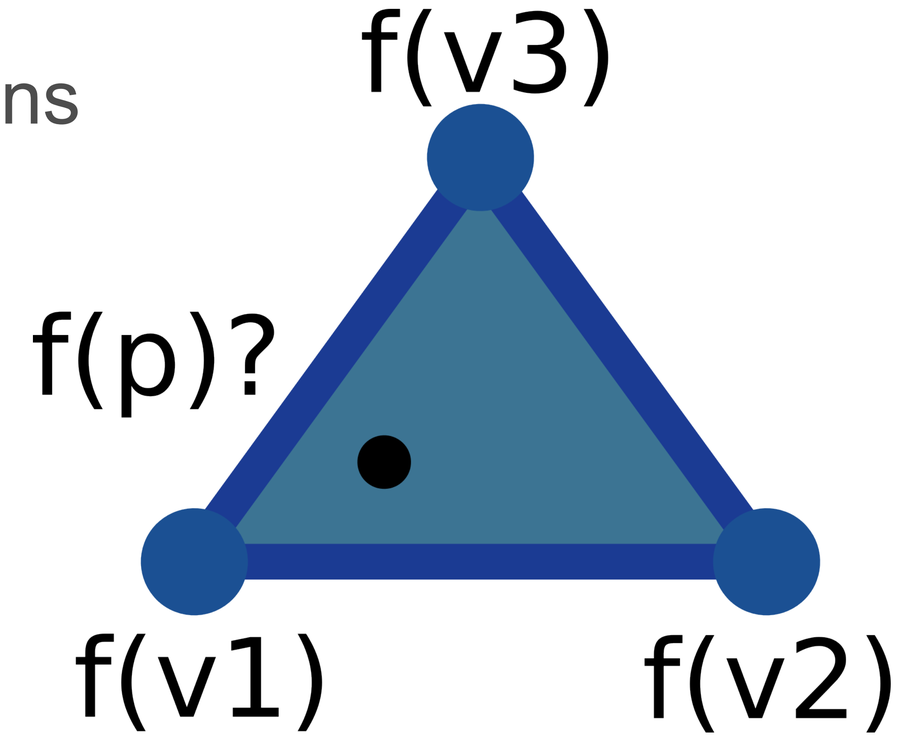
Barycentric coordinates

- In particular



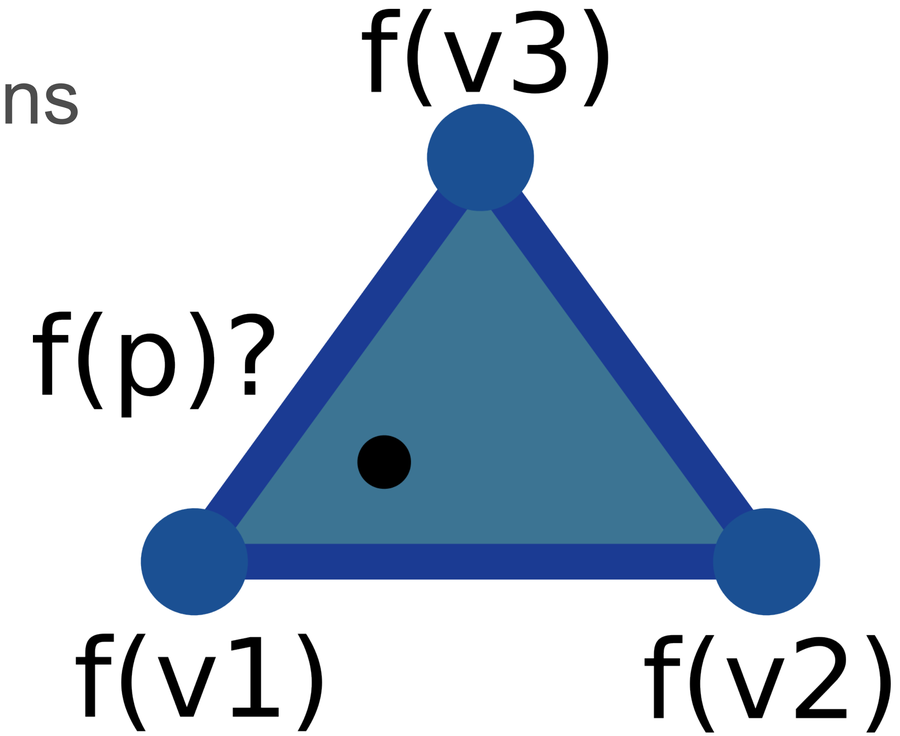
Barycentric coordinates

- In particular
  - It must also hold for the embedding functions



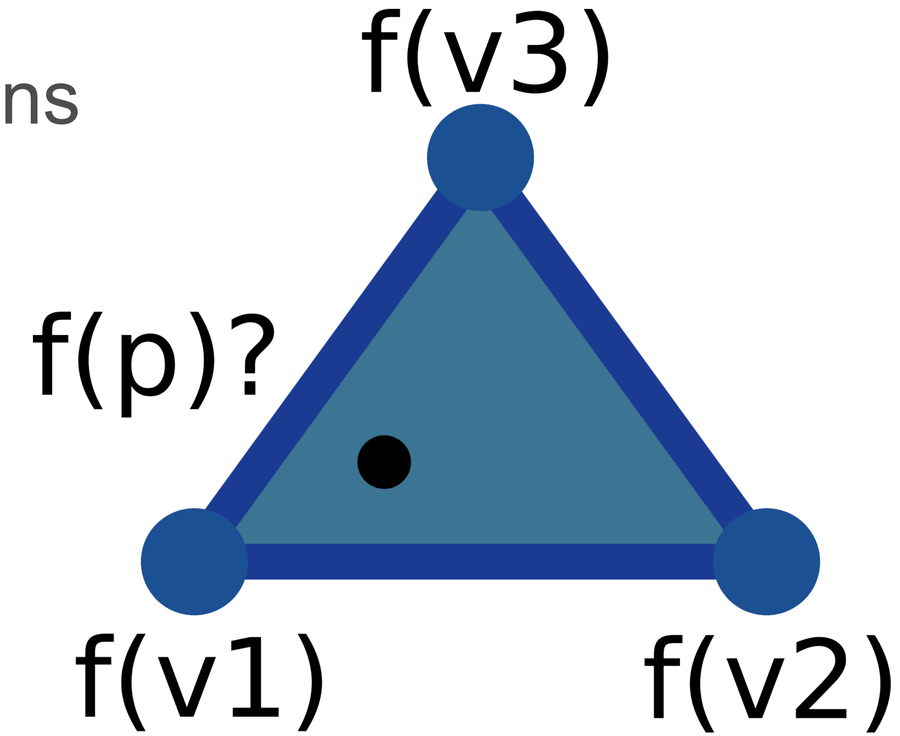
Barycentric coordinates

- In particular
  - It must also hold for the embedding functions
  - $f(p) = \alpha_1 f(v_1) + \alpha_2 f(v_2) + \alpha_3 f(v_3)$
  - $\alpha_1 + \alpha_2 + \alpha_3 = 1$
  - $x(p) = \alpha_1 x(v_1) + \alpha_2 x(v_2) + \alpha_3 x(v_3)$
  - $y(p) = \alpha_1 y(v_1) + \alpha_2 y(v_2) + \alpha_3 y(v_3)$



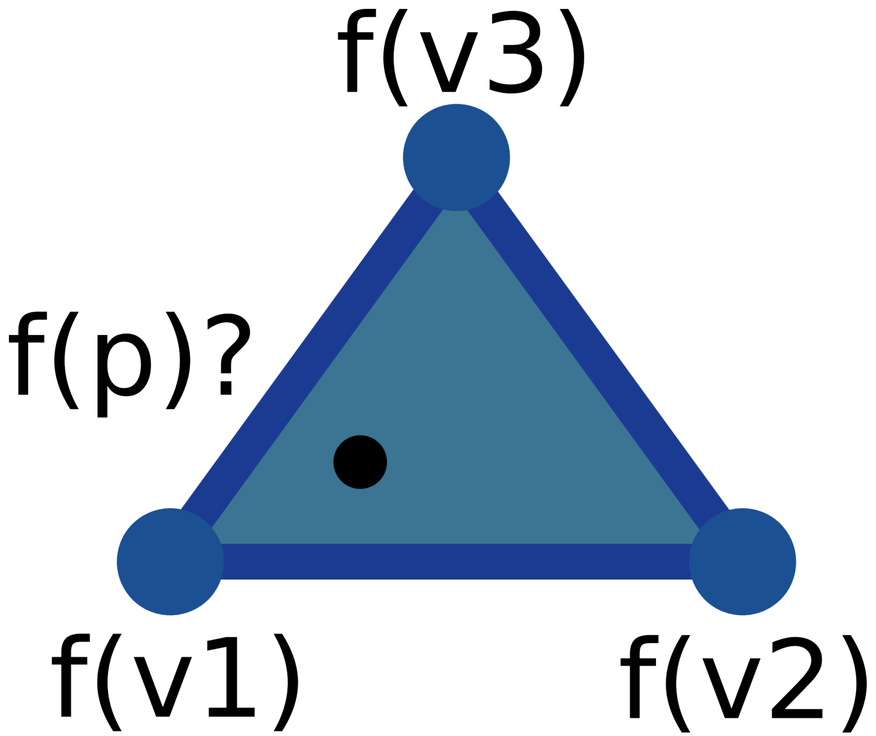
Barycentric coordinates

- In particular
  - It must also hold for the embedding functions
  - $f(p) = \alpha_1 f(v_1) + \alpha_2 f(v_2) + \alpha_3 f(v_3)$
  - $\alpha_1 + \alpha_2 + \alpha_3 = 1$
  - $x(p) = \alpha_1 x(v_1) + \alpha_2 x(v_2) + \alpha_3 x(v_3)$
  - $y(p) = \alpha_1 y(v_1) + \alpha_2 y(v_2) + \alpha_3 y(v_3)$
- 3 linear equations with 3 unknowns



Barycentric coordinates

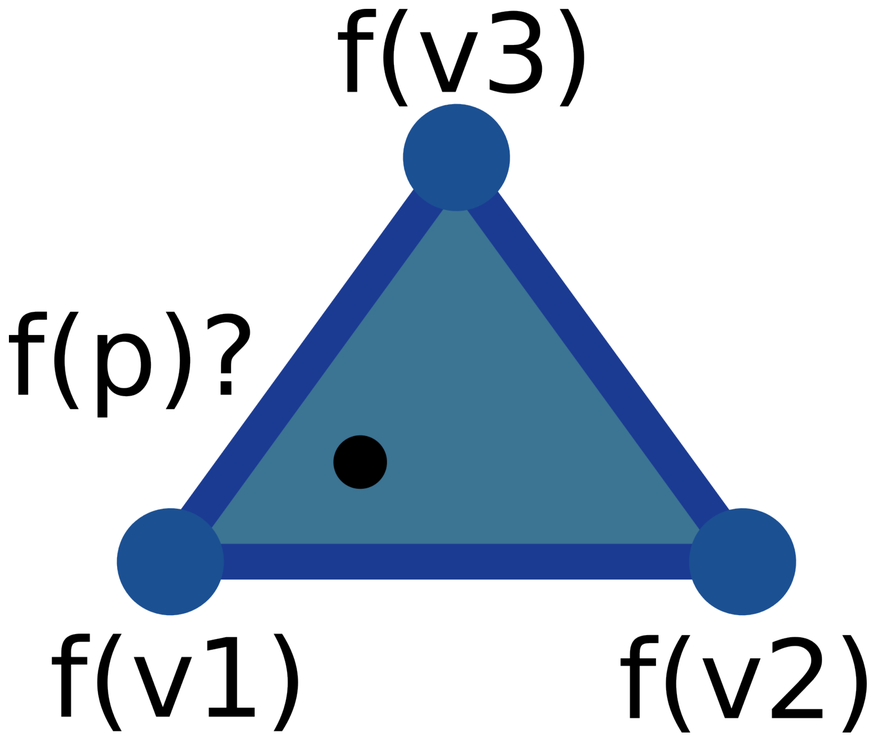
- $f(p) = \alpha_1 f(v_1) + \alpha_2 f(v_2) + \alpha_3 f(v_3)$
- $\alpha_1 + \alpha_2 + \alpha_3 = 1$
- $x(p) = \alpha_1 x(v_1) + \alpha_2 x(v_2) + \alpha_3 x(v_3)$
- $y(p) = \alpha_1 y(v_1) + \alpha_2 y(v_2) + \alpha_3 y(v_3)$



Barycentric coordinates

- $f(p) = \alpha_1 f(v_1) + \alpha_2 f(v_2) + \alpha_3 f(v_3)$
- $\alpha_1 + \alpha_2 + \alpha_3 = 1$
- $x(p) = \alpha_1 x(v_1) + \alpha_2 x(v_2) + \alpha_3 x(v_3)$
- $y(p) = \alpha_1 y(v_1) + \alpha_2 y(v_2) + \alpha_3 y(v_3)$

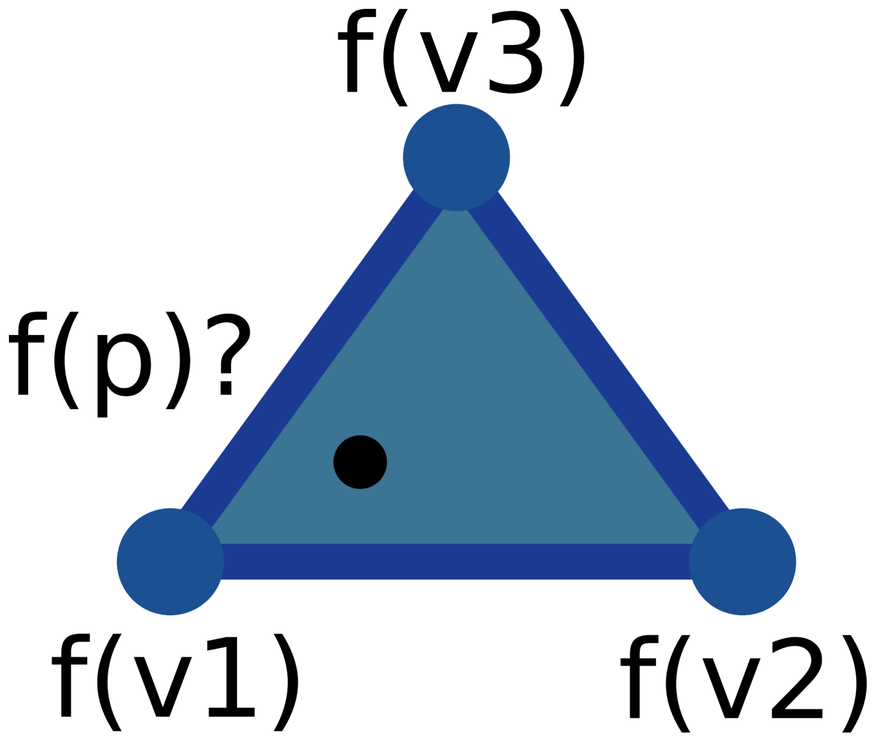
- Automatically interpolates on the edges



Barycentric coordinates

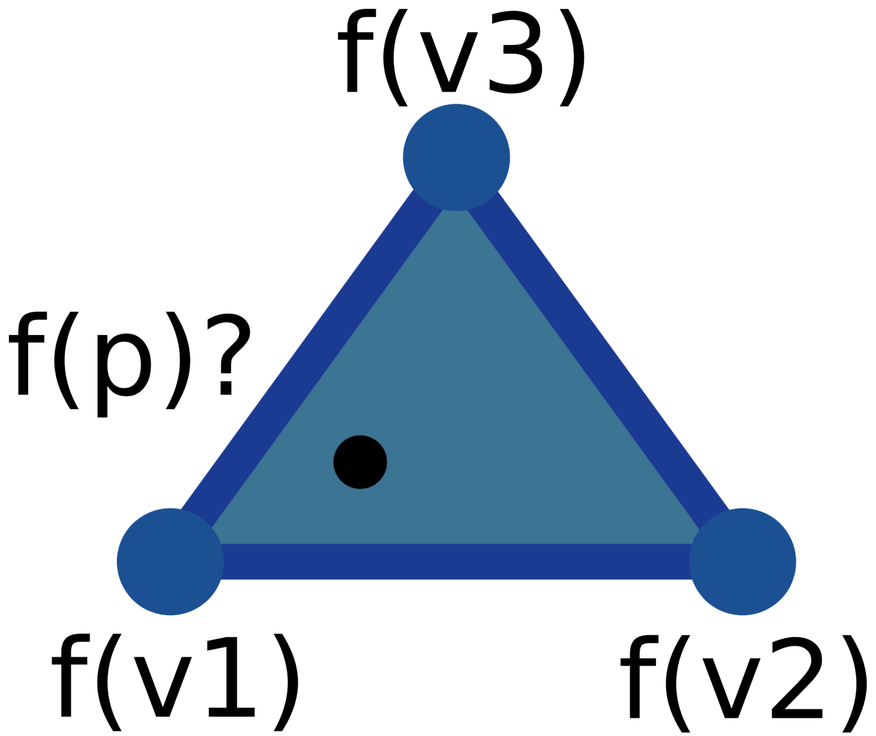
- $f(p) = \alpha_1 f(v_1) + \alpha_2 f(v_2) + \alpha_3 f(v_3)$
- $\alpha_1 + \alpha_2 + \alpha_3 = 1$
- $x(p) = \alpha_1 x(v_1) + \alpha_2 x(v_2) + \alpha_3 x(v_3)$
- $y(p) = \alpha_1 y(v_1) + \alpha_2 y(v_2) + \alpha_3 y(v_3)$

- Automatically interpolates on the edges
- Similar reasoning for d-simplex



Barycentric coordinates

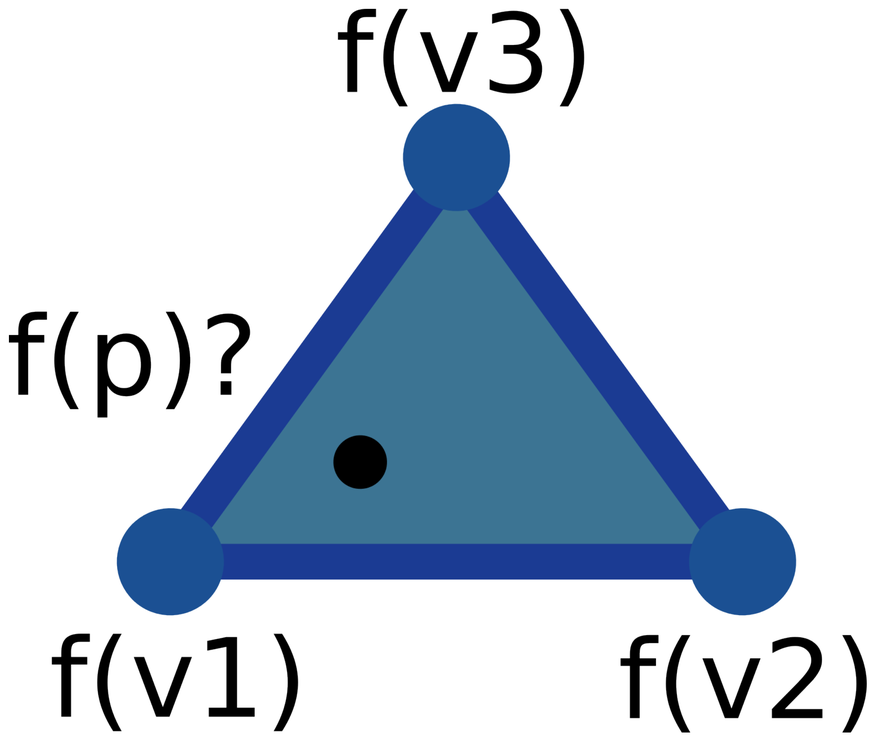
- $f(p) = \alpha_1 f(v_1) + \alpha_2 f(v_2) + \alpha_3 f(v_3)$
- $\alpha_1 + \alpha_2 + \alpha_3 = 1$
- $x(p) = \alpha_1 x(v_1) + \alpha_2 x(v_2) + \alpha_3 x(v_3)$
- $y(p) = \alpha_1 y(v_1) + \alpha_2 y(v_2) + \alpha_3 y(v_3)$



- Automatically interpolates on the edges
- Similar reasoning for d-simplex
- Can be used to determine if a point lies within a simplex

Barycentric coordinates

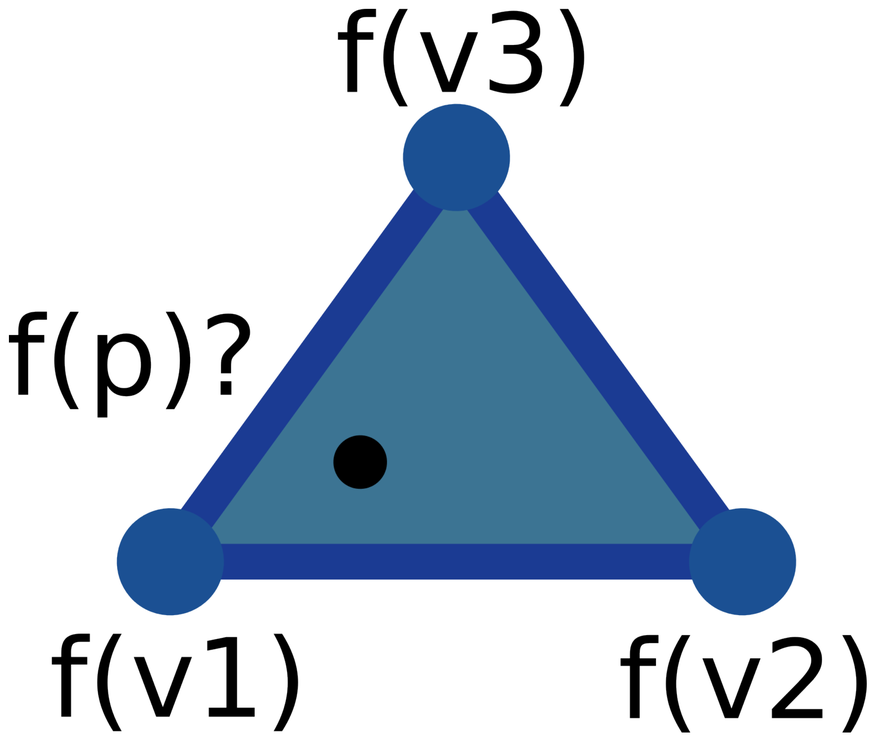
- $f(p) = \alpha_1 f(v_1) + \alpha_2 f(v_2) + \alpha_3 f(v_3)$
- $\alpha_1 + \alpha_2 + \alpha_3 = 1$
- $x(p) = \alpha_1 x(v_1) + \alpha_2 x(v_2) + \alpha_3 x(v_3)$
- $y(p) = \alpha_1 y(v_1) + \alpha_2 y(v_2) + \alpha_3 y(v_3)$



- Automatically interpolates on the edges
- Similar reasoning for d-simplex
- Can be used to determine if a point lies within a simplex $(\alpha_1, \alpha_2, \alpha_3) \in [0,1] \times [0,1] \times [0,1]$

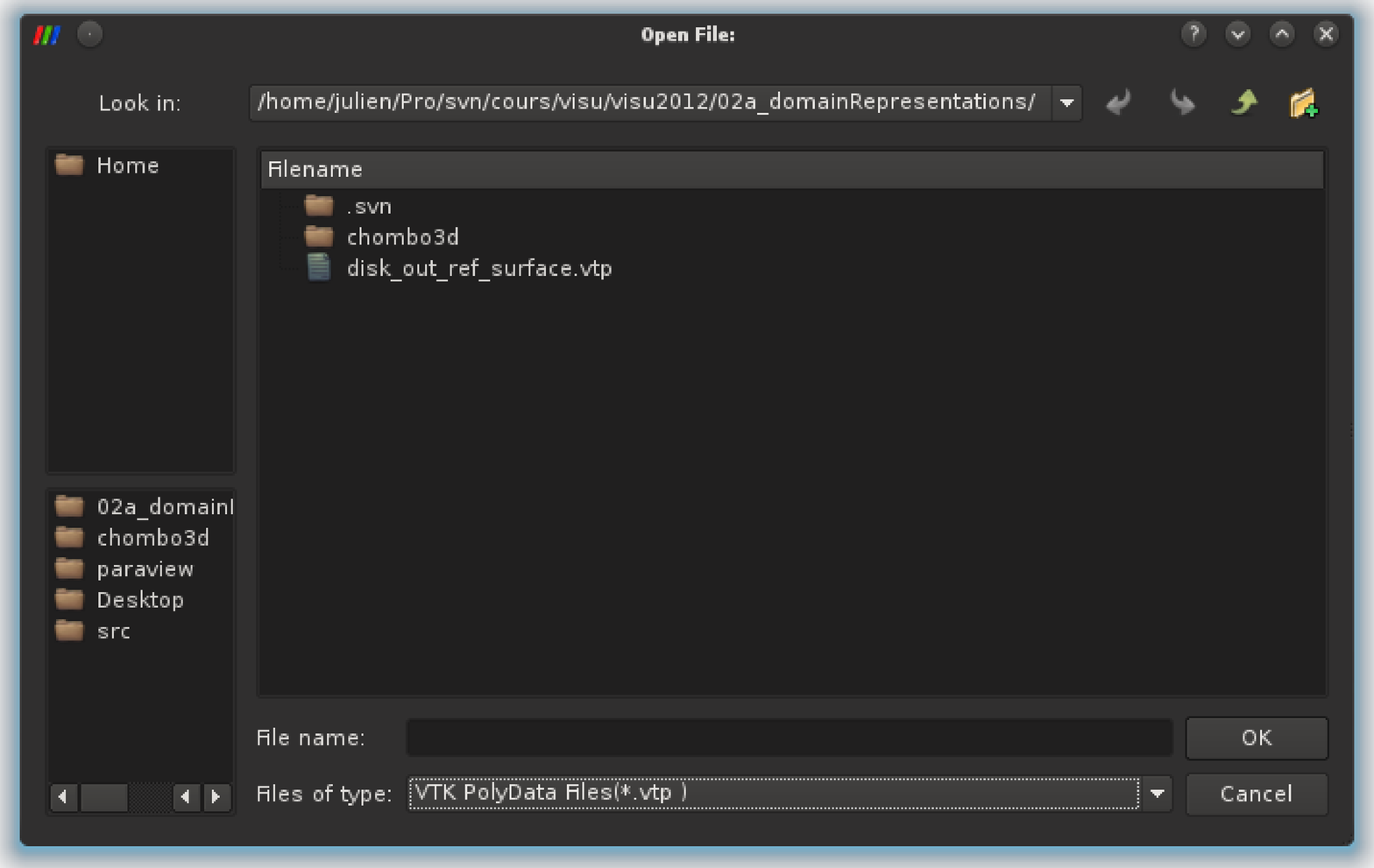
Barycentric coordinates

- $f(p) = \alpha_1 f(v_1) + \alpha_2 f(v_2) + \alpha_3 f(v_3)$
- $\alpha_1 + \alpha_2 + \alpha_3 = 1$
- $x(p) = \alpha_1 x(v_1) + \alpha_2 x(v_2) + \alpha_3 x(v_3)$
- $y(p) = \alpha_1 y(v_1) + \alpha_2 y(v_2) + \alpha_3 y(v_3)$

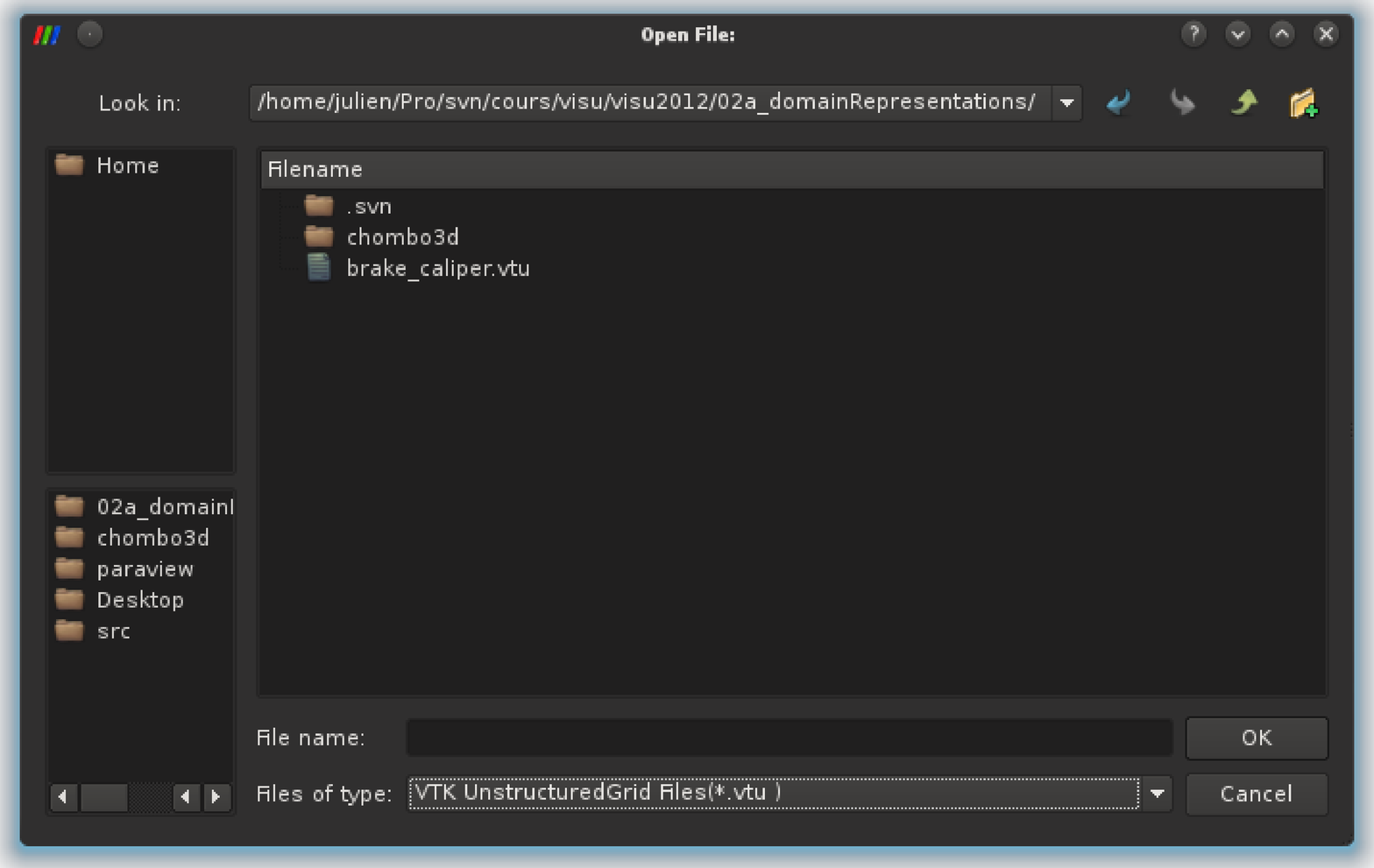


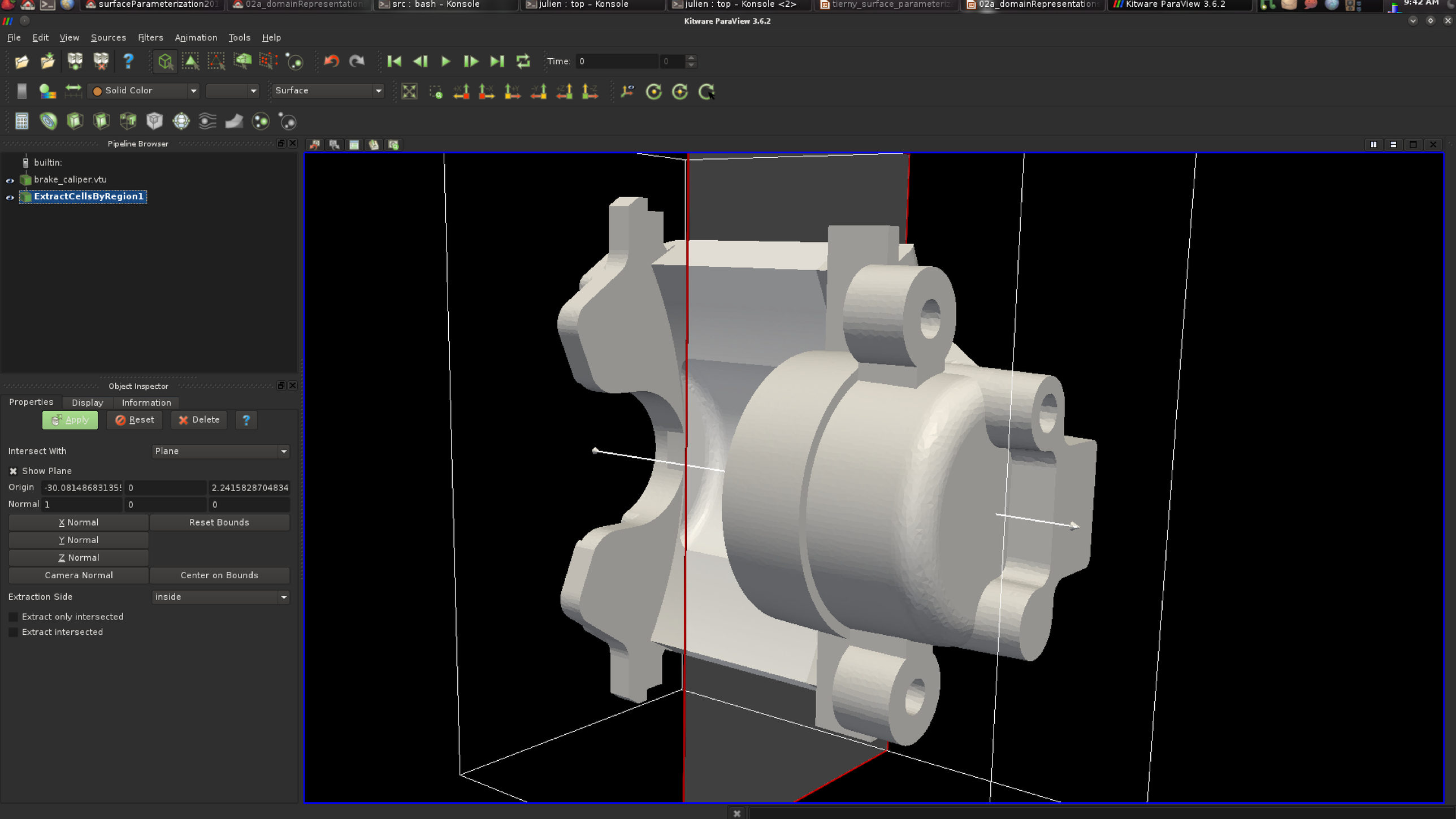
- Automatically interpolates on the edges
- Similar reasoning for d-simplex
- Can be used to determine if a point lies within a simplex $(\alpha_1, \alpha_2, \alpha_3) \in [0,1] \times [0,1] \times [0,1]$
- No critical point in the interior!

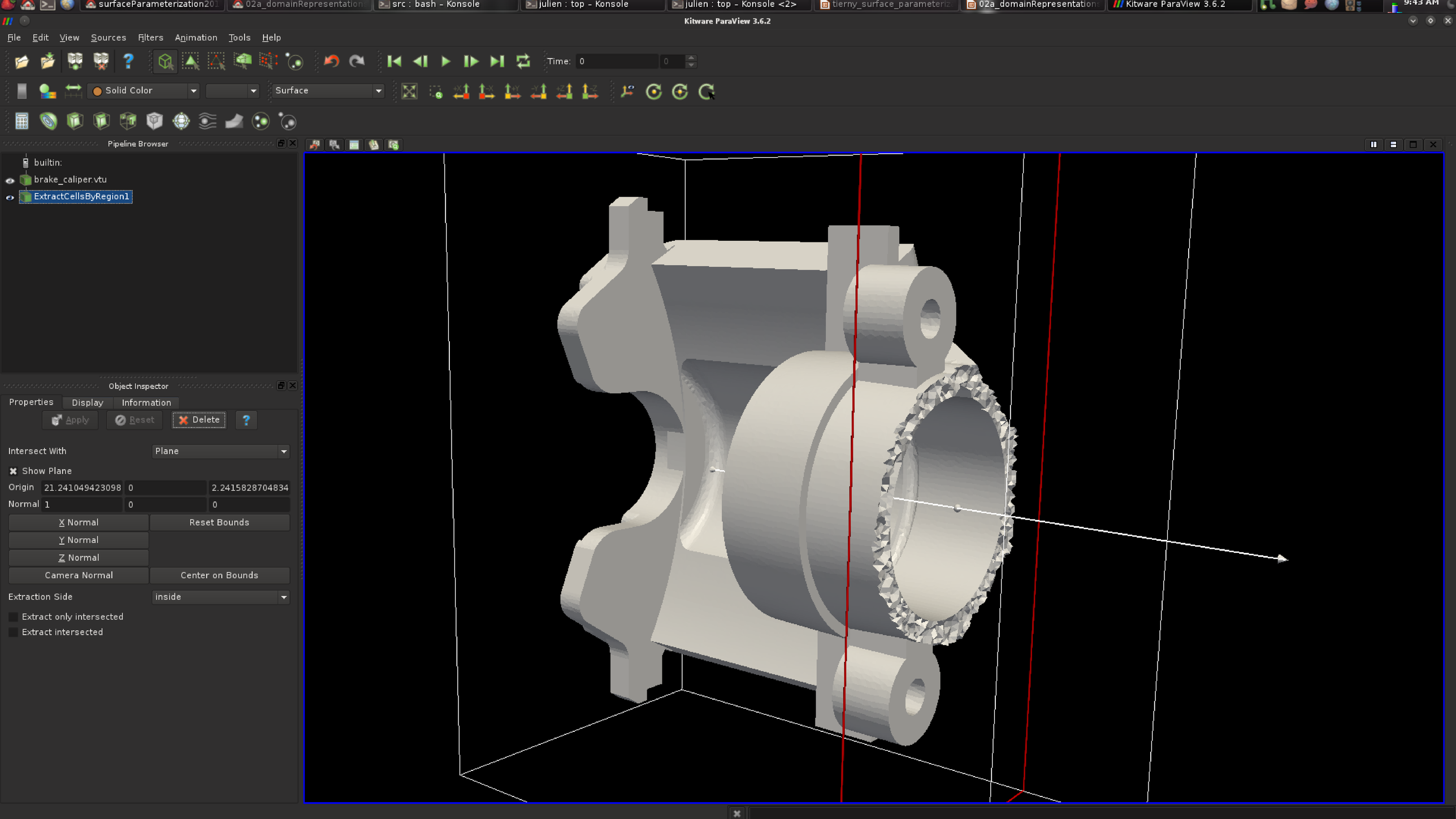
In practice

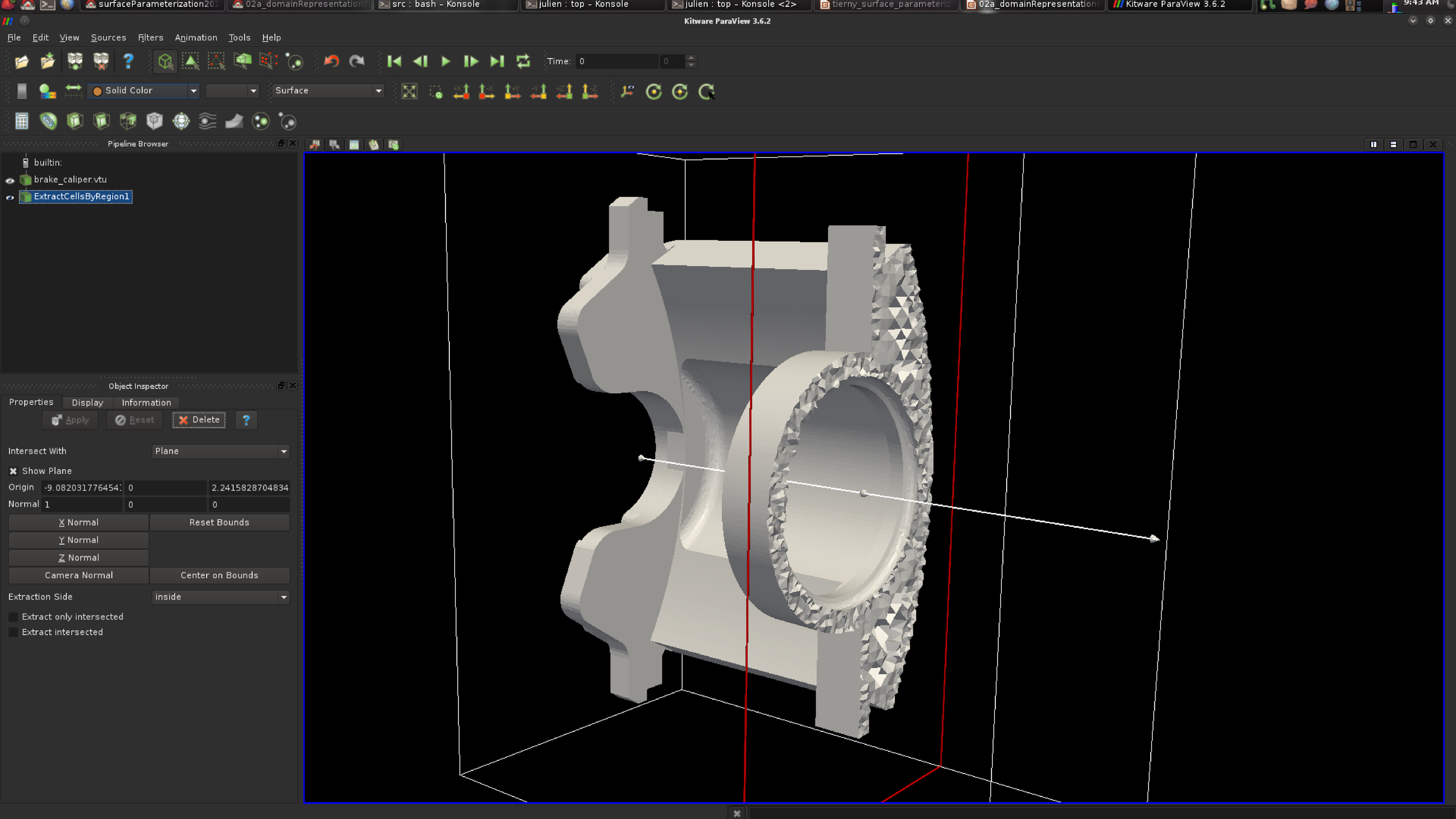

Open File:
Look in:
/home/julien/Pro/svn/cours/visu/visu2012/02a_domainRepresentations/
Home
Filename
.svn
chombo3d
disk_out_ref_surface.vtp
02a_domainI
chombo3d
paraview
Desktop
src
File name:
OK
Files of type:
VTK PolyData Files(*.vtp)
Cancel

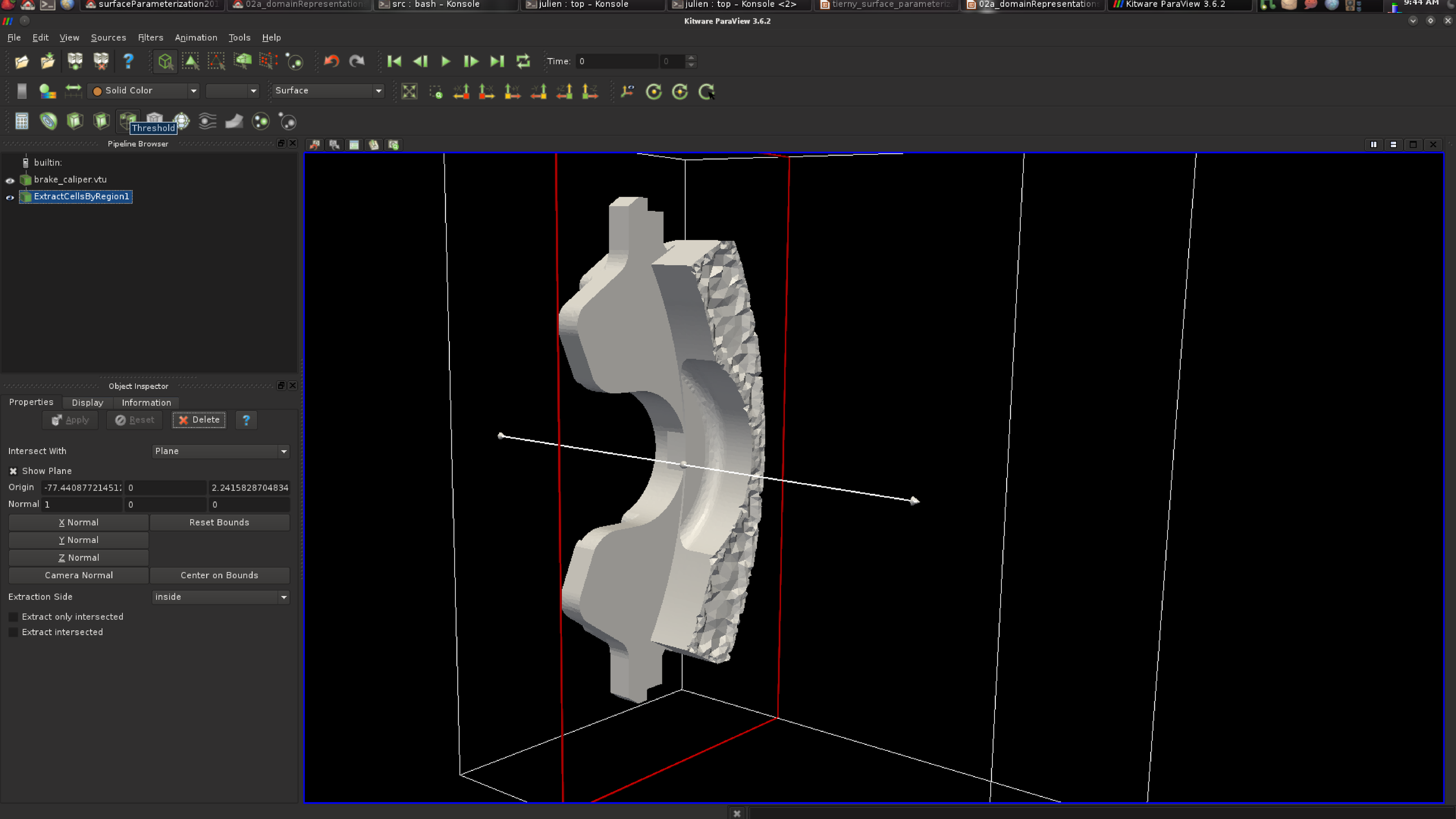
In practice


Open File:
Look in:
/home/julien/Pro/svn/cours/visu/visu2012/02a_domainRepresentations/
Home
Filename
.svn
chombo3d
brake_caliper.vtu
02a_domainI
chombo3d
paraview
Desktop
src
File name:
OK
Files of type:
VTK UnstructuredGrid Files(*.vtu)
Cancel


surfaceParameterization201
02a_domainRepresentation
src : bash - Konsole
julien : top - Konsole
julien : top - Konsole <2>
tierny_surface_parameteriz
02a_domainRepresentation
Kitware ParaView 3.6.2
9:42 AM
Kitware ParaView 3.6.2
File Edit View Sources Filters Animation Tools Help
Time: 0
Solid Color
Surface
Pipeline Browser
builtin:
brake_caliper.vtu
ExtractCellsByRegion1
Object Inspector
Properties Display Information
Apply Reset Delete
Intersect With Plane
Show Plane
Origin -30.08148683135 0 2.2415828704834
Normal 1 0 0
X Normal
Reset Bounds
Y Normal
Z Normal
Camera Normal
Center on Bounds
Extraction Side inside
Extract only intersected
Extract intersected


Kitware ParaView 3.6.2
File Edit View Sources Filters Animation Tools Help
Time: 0
Solid Color
Surface
Pipeline Browser
builtin:
brake_caliper.vtu
ExtractCellsByRegion1
Object Inspector
Properties Display Information
Apply Reset Delete
Intersect With Plane
Show Plane
Origin 21.241049423098 0 2.2415828704834
Normal 1 0 0
X Normal
Reset Bounds
Y Normal
Z Normal
Camera Normal
Center on Bounds
Extraction Side inside
Extract only intersected
Extract intersected


surfaceParameterization201
02a_domainRepresentation
src : bash - Konsole
julien : top - Konsole
julien : top - Konsole <2>
tierny_surface_parameteriz
02a_domainRepresentation
Kitware ParaView 3.6.2
9:43 AM
Kitware ParaView 3.6.2
File Edit View Sources Filters Animation Tools Help
Time: 0
Solid Color
Surface
Pipeline Browser
builtin:
brake_caliper.vtu
ExtractCellsByRegion1
Object Inspector
Properties Display Information
Apply Reset Delete
Intersect With Plane
Show Plane
Origin -9.082031776454 0 2.2415828704834
Normal 1 0 0
X Normal
Reset Bounds
Y Normal
Z Normal
Camera Normal
Center on Bounds
Extraction Side inside
Extract only intersected
Extract intersected


surfaceParameterization201
02a_domainRepresentation
src : bash - Konsole
julien : top - Konsole
julien : top - Konsole <2>
tierny_surface_parameteriz
02a_domainRepresentation
Kitware ParaView 3.6.2
9:44 AM
Kitware ParaView 3.6.2
File Edit View Sources Filters Animation Tools Help
Time: 0
Solid Color
Surface
Threshold
Pipeline Browser
builtin:
brake_caliper.vtu
ExtractCellsByRegion1
Object Inspector
Properties Display Information
Apply Reset Delete
Intersect With Plane
Show Plane
Origin -77.44087721451 0 2.2415828704834
Normal 1 0 0
X Normal
Reset Bounds
Y Normal
Z Normal
Camera Normal
Center on Bounds
Extraction Side inside
Extract only intersected
Extract intersected

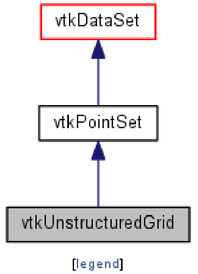
VTK

vtkUnstructuredGrid Class Reference

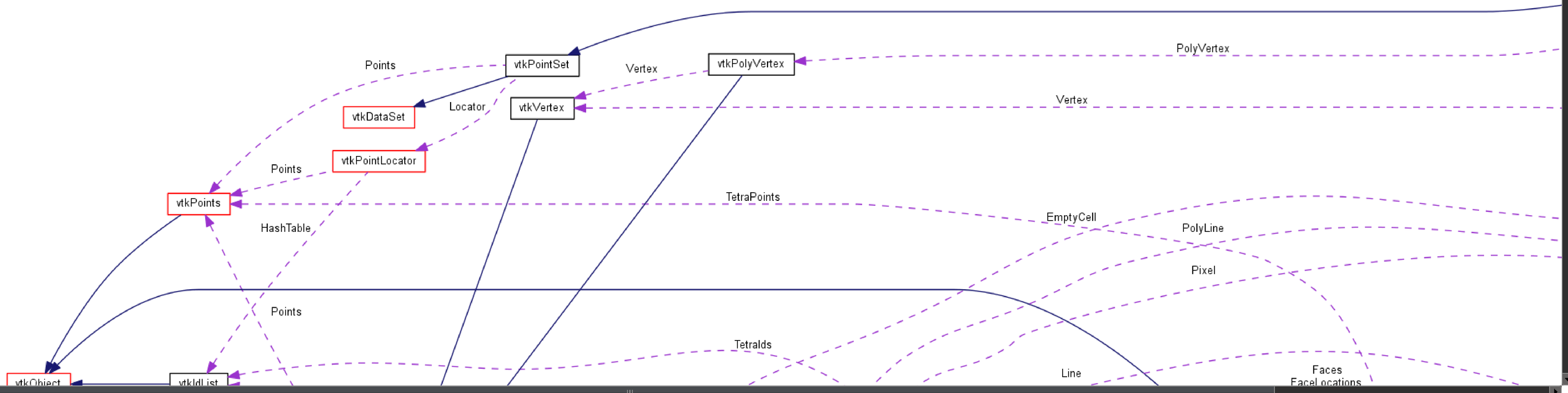
dataset represents arbitrary combinations of all possible cell types More...

#include <vtkUnstructuredGrid.h>

Inheritance diagram for vtkUnstructuredGrid:



Collaboration diagram for vtkUnstructuredGrid:



Public Member Functions

virtual **int** **IsA** (const char *type)
vtkUnstructuredGrid * **NewInstance** () const
void **PrintSelf** (ostream &os, **vtkIndent** indent)
vtkIdType **InsertNextCell** (**int** type, **vtkIdType** npts, **vtkIdType** *ptIds)
vtkIdType **InsertNextCell** (**int** type, **vtkIdList** *ptIds)
vtkIdType **InsertNextCell** (**int** type, **vtkIdType** npts, **vtkIdType** *ptIds, **vtkIdType** nfaces, **vtkIdType** *faces)
int **GetCellType** (**vtkIdType** cellId)
vtkUnsignedCharArray * **GetCellTypesArray** ()
vtkIdTypeArray * **GetCellLocationsArray** ()
void **Squeeze** ()
void **Initialize** ()
int **GetMaxCellSize** ()
void **BuildLinks** ()
vtkCellLinks * **GetCellLinks** ()
virtual void **GetCellPoints** (**vtkIdType** cellId, **vtkIdType** &npts, **vtkIdType** *&pts)
void **GetFaceStream** (**vtkIdType** cellId, **vtkIdList** *ptIds)
void **GetFaceStream** (**vtkIdType** cellId, **vtkIdType** &nfaces, **vtkIdType** *&ptIds)
vtkCellArray * **GetCells** ()
void **ReplaceCell** (**vtkIdType** cellId, **int** npts, **vtkIdType** *pts)
vtkIdType **InsertNextLinkedCell** (**int** type, **int** npts, **vtkIdType** *pts)
void **RemoveReferenceToCell** (**vtkIdType** ptId, **vtkIdType** cellId)
void **AddReferenceToCell** (**vtkIdType** ptId, **vtkIdType** cellId)
void **ResizeCellList** (**vtkIdType** ptId, **int** size)
virtual **int** **GetGhostLevel** ()
unsigned long **GetActualMemorySize** ()
void **GetIdsOfCellsOfType** (**int** type, **vtkIdTypeArray** *array)
int **IsHomogeneous** ()
void **RemoveGhostCells** (**int** level)
vtkIdType * **GetFaces** (**vtkIdType** cellId)
int **InitializeFacesRepresentation** (**vtkIdType** numPrevCells)

int **GetDataObjectType** ()
virtual void **Allocate** (**vtkIdType** numCells=1000, **int** extSize=1000)

void **Reset** ()
virtual void **CopyStructure** (**vtkDataSet** *ds)
vtkIdType **GetNumberOfCells** ()
virtual **vtkCell** * **GetCell** (**vtkIdType** cellId)
virtual void **GetCell** (**vtkIdType** cellId, **vtkGenericCell** *cell)
virtual void **GetCellBounds** (**vtkIdType** cellId, **double** bounds[6])
virtual void **GetCellPoints** (**vtkIdType** cellId, **vtkIdList** *ptIds)
void **GetPointCells** (**vtkIdType** ptId, **vtkIdList** *cellIds)

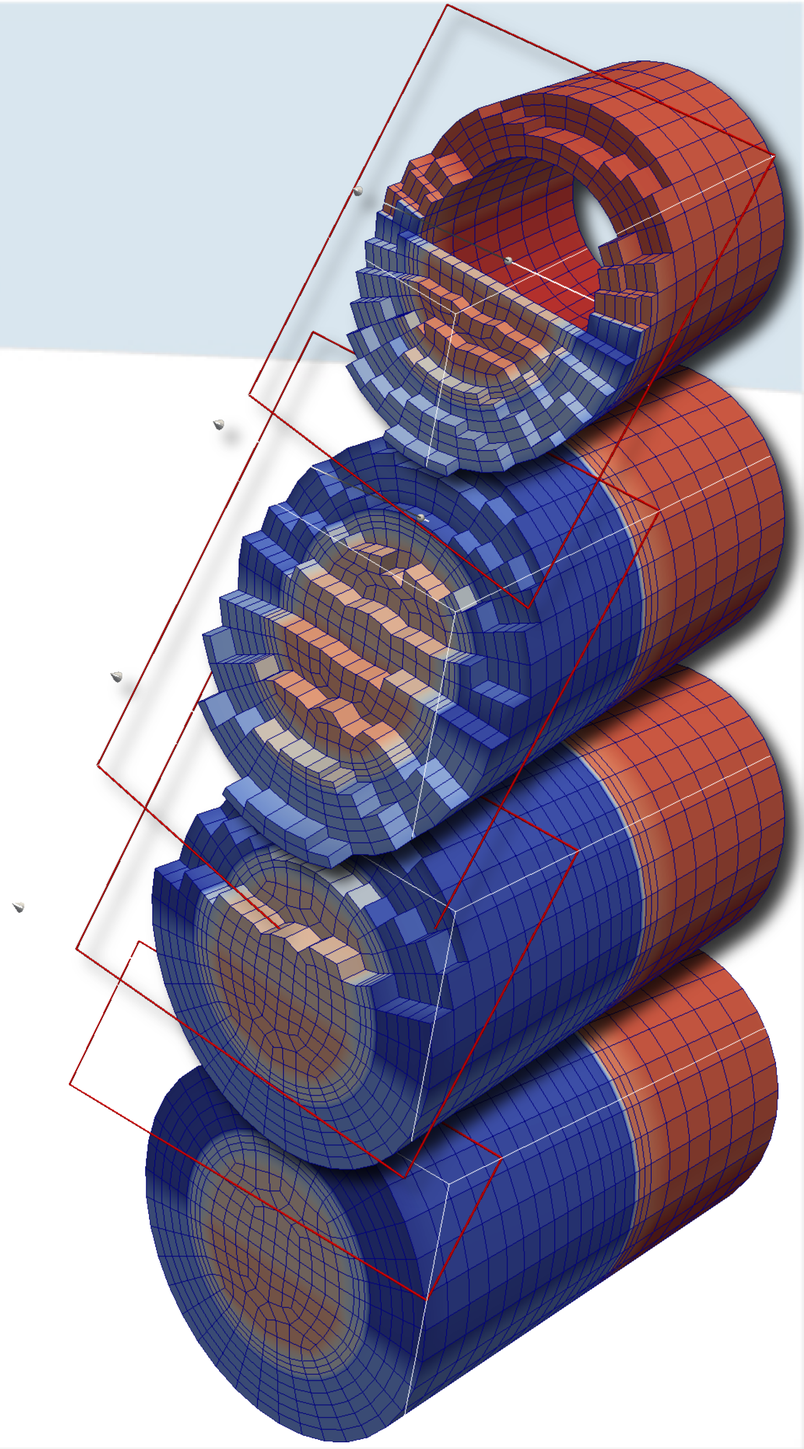
void **SetCells** (**int** type, **vtkCellArray** *cells)

Are we done?

- Nearly

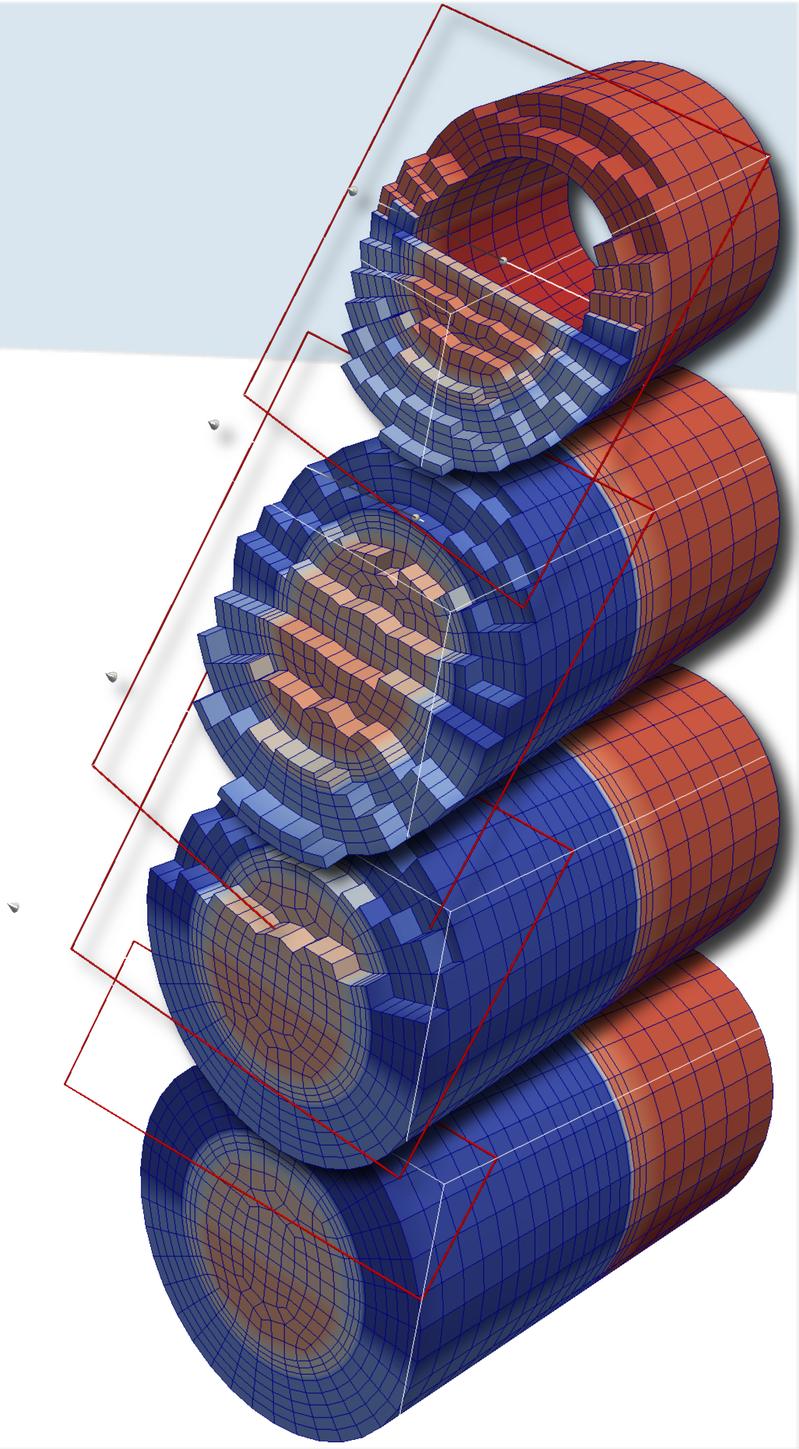
Are we done?

- Nearly
  - What about these objects?



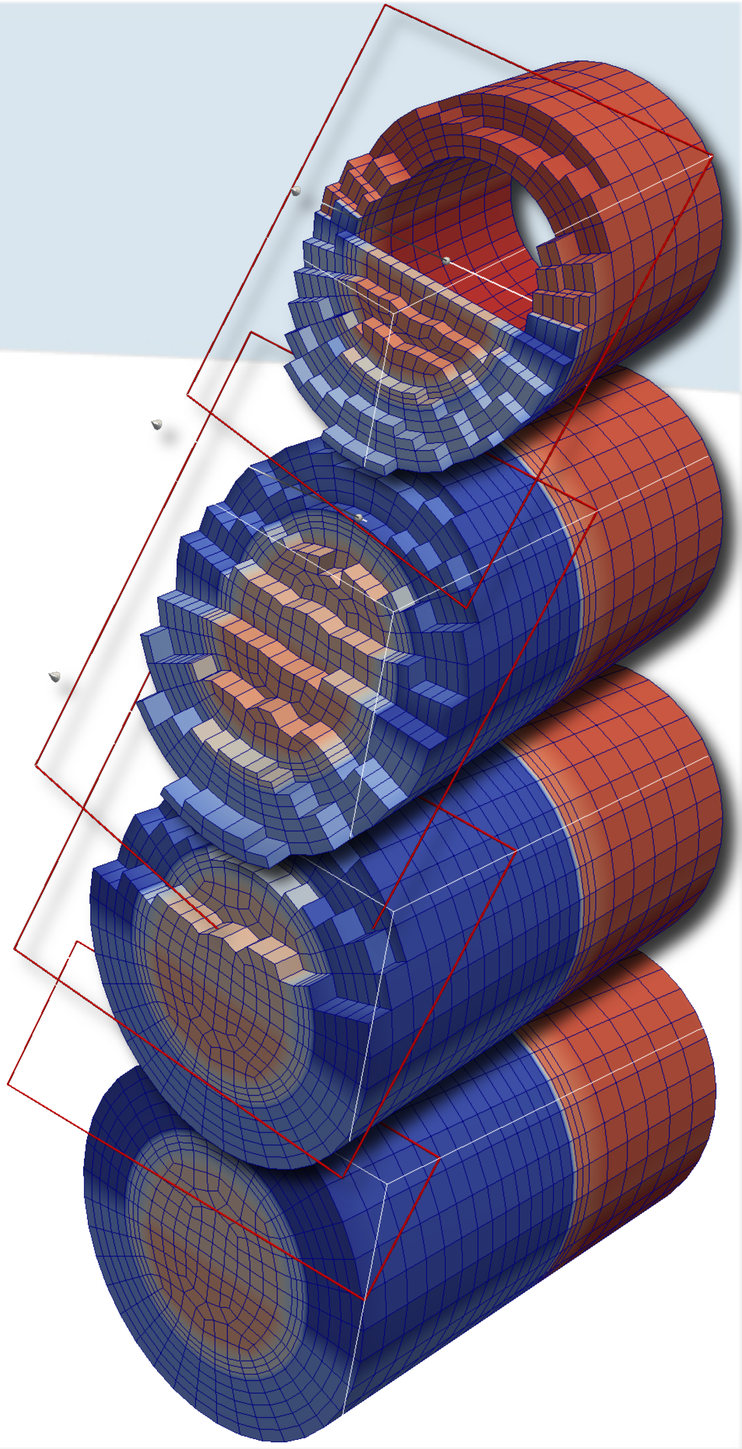
Are we done?

- Nearly
  - What about these objects?
- Other manifold representations



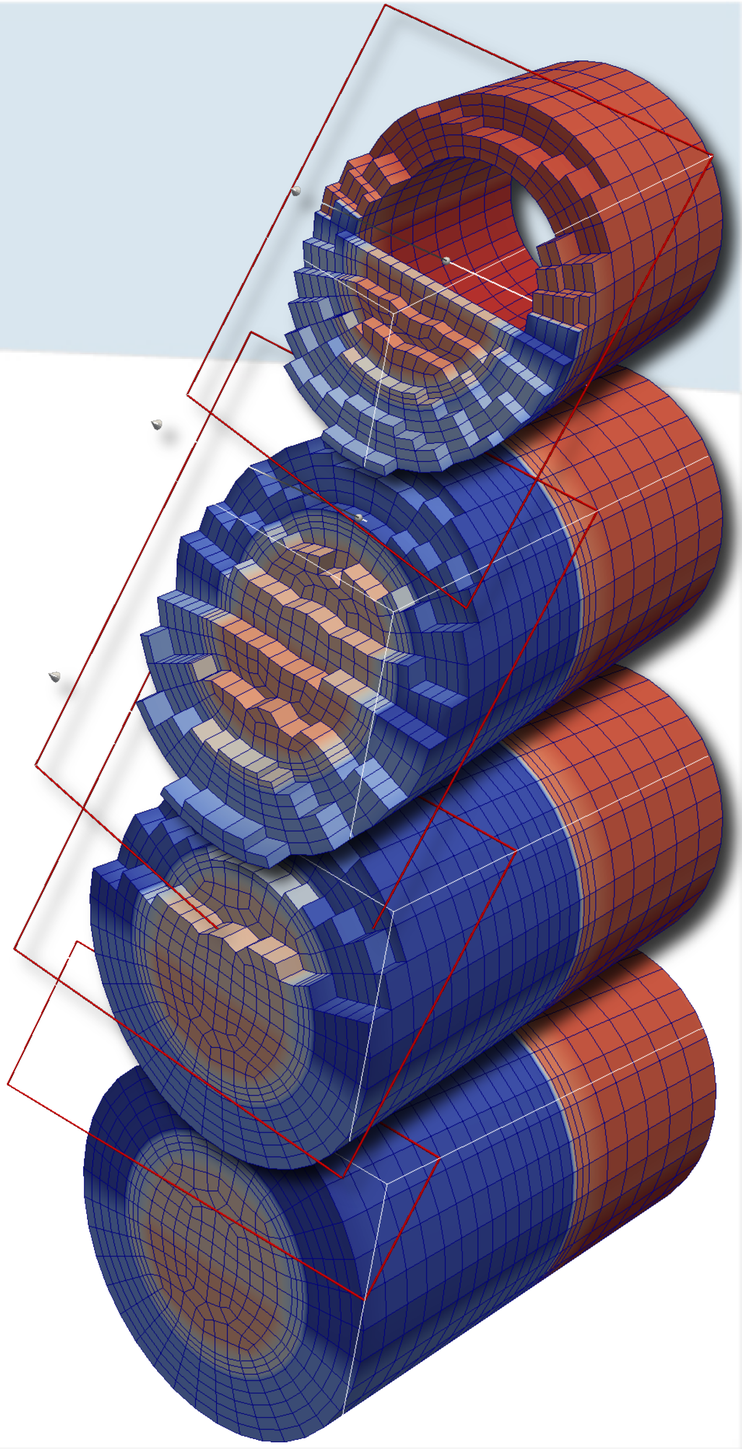
Are we done?

- Nearly
  - What about these objects?
- Other manifold representations
  - Closure-finite Weak-topology (CW) complex



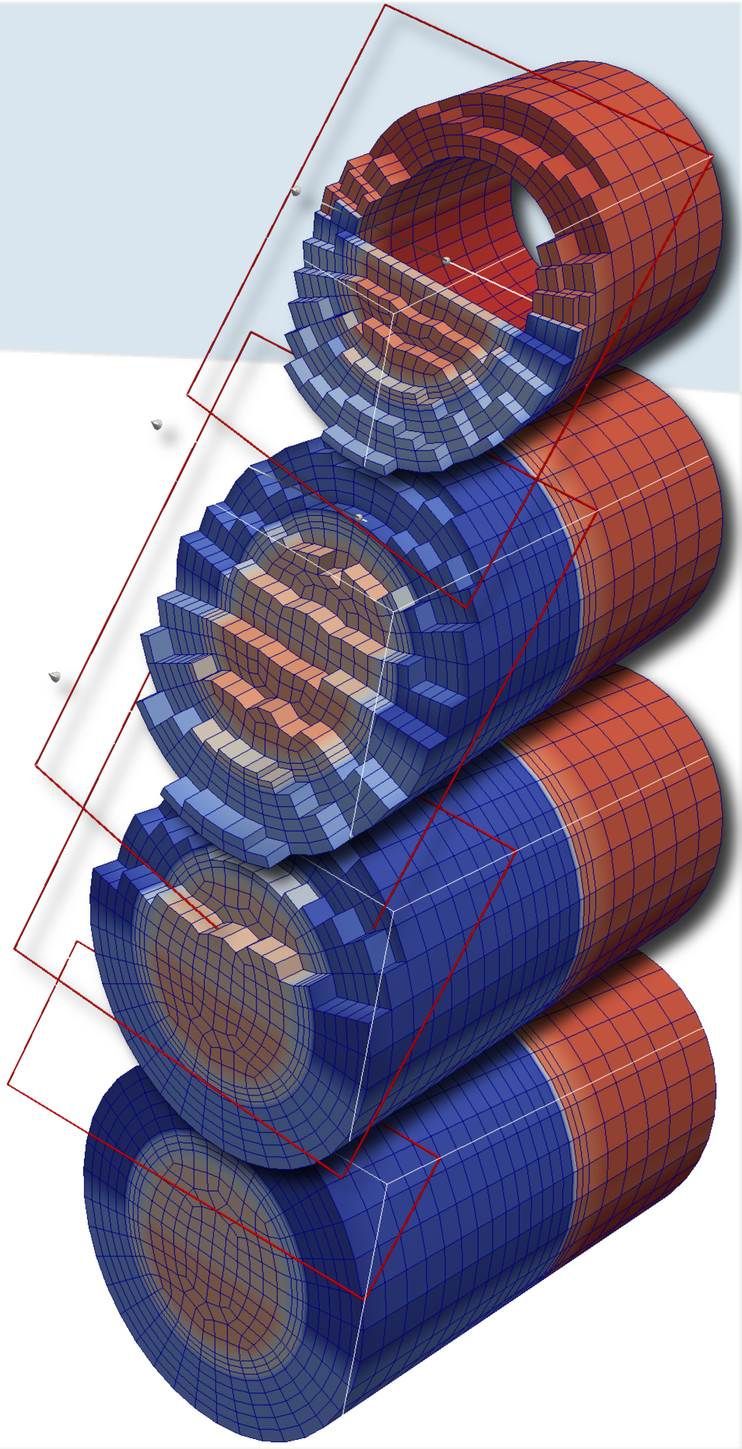
Are we done?

- Nearly
  - What about these objects?
- Other manifold representations
  - Closure-finite Weak-topology (CW) complex
    - Arbitrary number of samples per cell



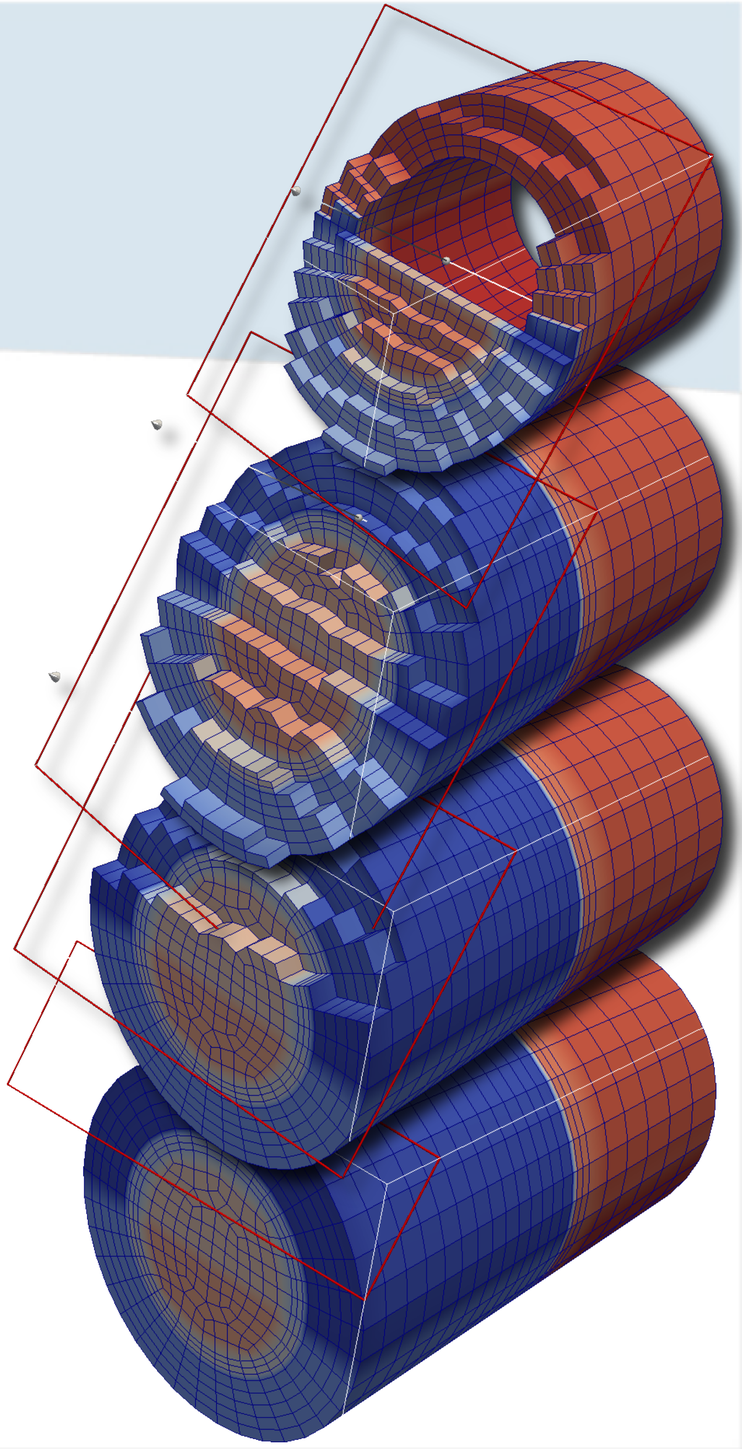
Are we done?

- Nearly
  - What about these objects?
- Other manifold representations
  - Closure-finite Weak-topology (CW) complex
    - Arbitrary number of samples per cell
    - Similar implementation



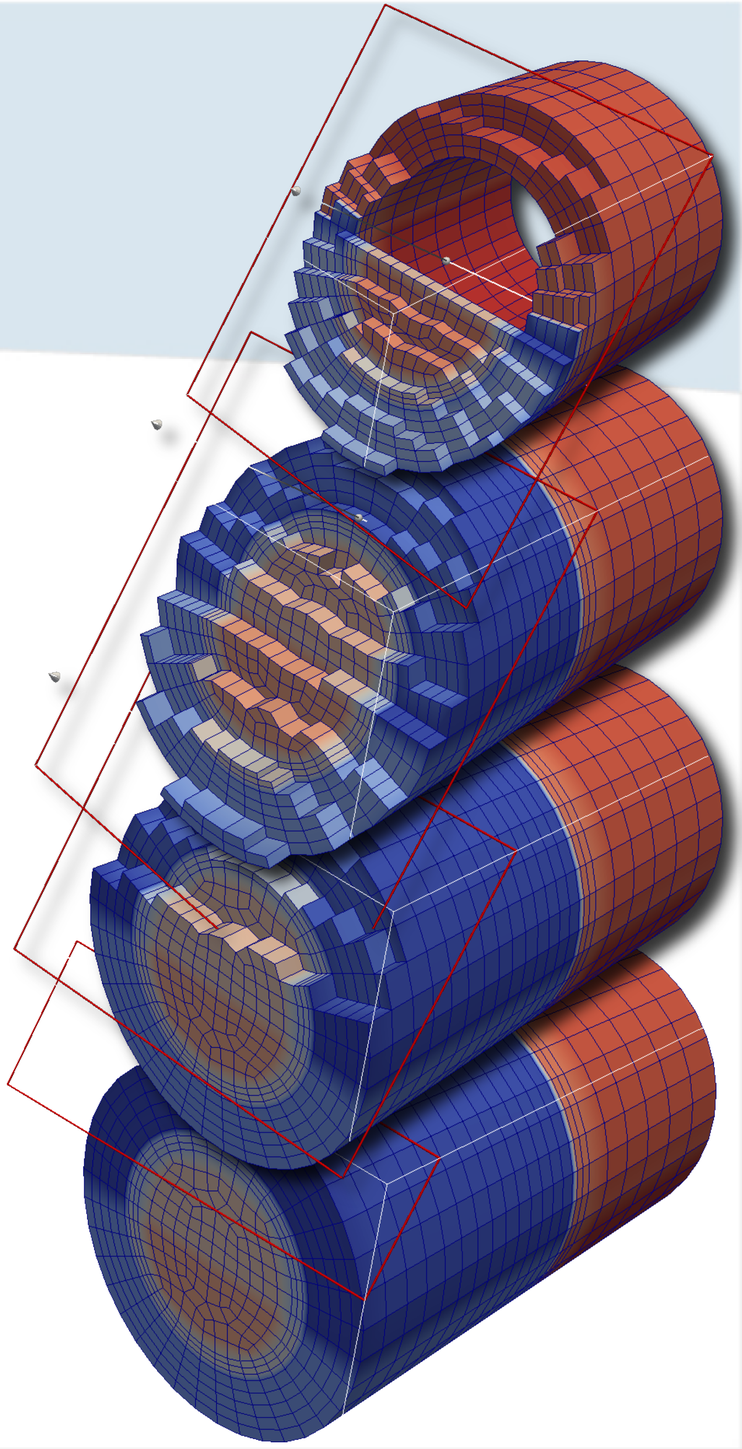
Are we done?

- Nearly
  - What about these objects?
- Other manifold representations
  - Closure-finite Weak-topology (CW) complex
    - Arbitrary number of samples per cell
    - Similar implementation (vtkUnstructuredGrid)



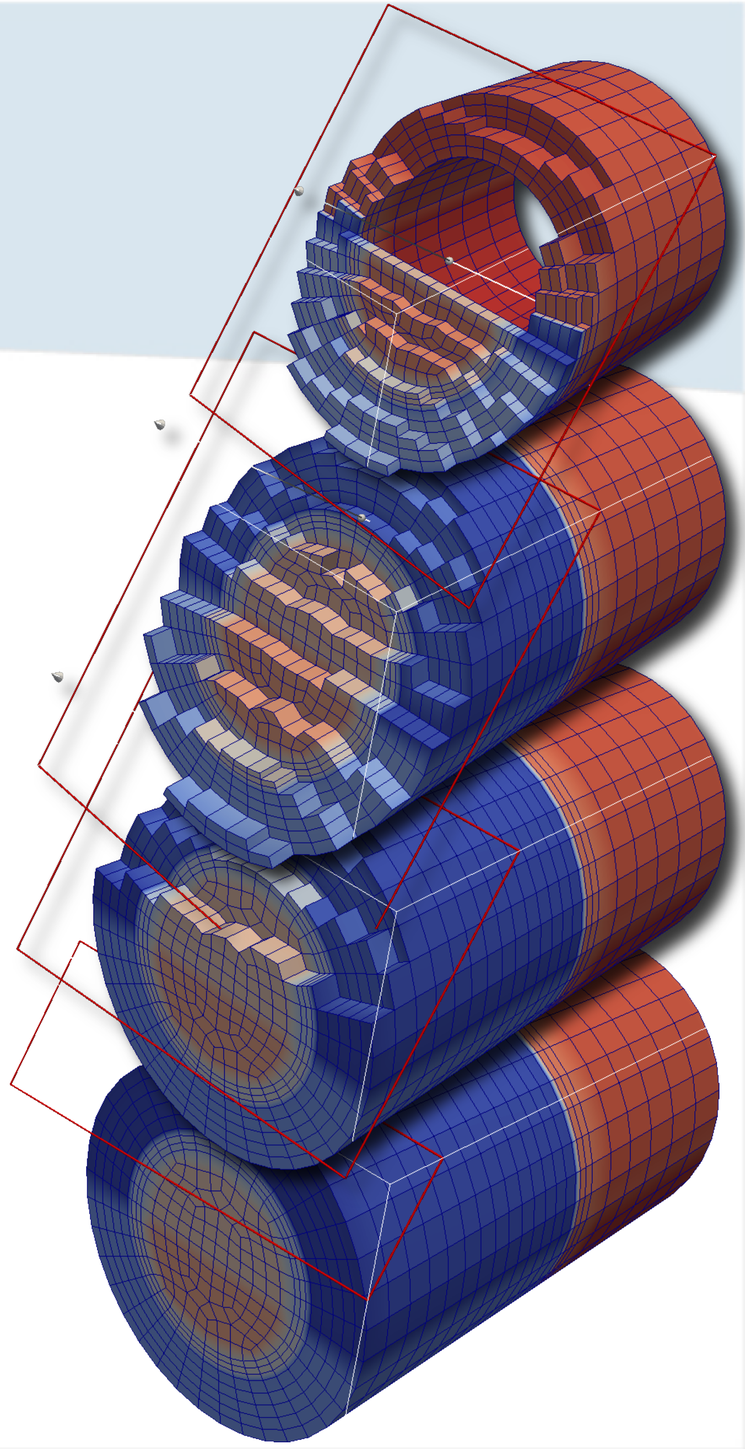
Are we done?

- Nearly
  - What about these objects?
- Other manifold representations
  - Closure-finite Weak-topology (CW) complex
    - Arbitrary number of samples per cell
    - Similar implementation (vtkUnstructuredGrid)
    - Special cases with 2^d samples per cell



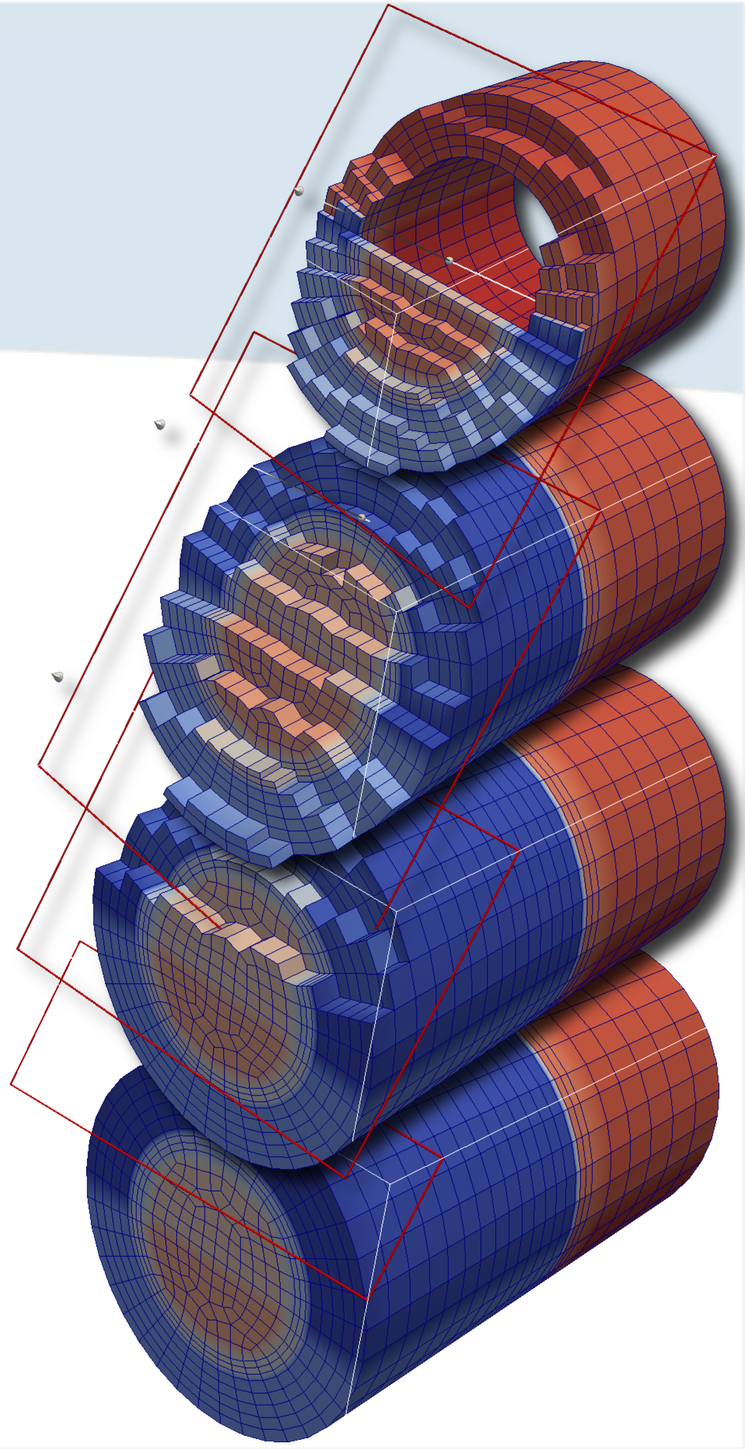
Are we done?

- Nearly
  - What about these objects?
- Other manifold representations
  - Closure-finite Weak-topology (CW) complex
    - Arbitrary number of samples per cell
    - Similar implementation (vtkUnstructuredGrid)
    - Special cases with 2^d samples per cell
      - Quad mesh
      - Hexahedral mesh



Are we done?

- Nearly
  - What about these objects?
- Other manifold representations
  - Closure-finite Weak-topology (CW) complex
    - Arbitrary number of samples per cell
    - Similar implementation (vtkUnstructuredGrid)
    - Special cases with 2^d samples per cell
      - Quad mesh
      - Hexahedral mesh … intriguing properties



In conclusion

- Now you know

In conclusion

- Now you know
  - A classification of possible domains

In conclusion

- Now you know
  - A classification of possible domains
  - How to implement representations for euclidean spaces

In conclusion

- Now you know
  - A classification of possible domains
  - How to implement representations for euclidean spaces
  - How to implement representations for manifold spaces

In conclusion

- Now you know
  - A classification of possible domains
  - How to implement representations for euclidean spaces
  - How to implement representations for manifold spaces
  - How to interpolate numerical values all over these domains

In conclusion

- Now you know
  - A classification of possible domains
  - How to implement representations for euclidean spaces
  - How to implement representations for manifold spaces
  - How to interpolate numerical values all over these domains
  - Example of available implementations