Describing and implementing basic geometric objects

Vera Sacristán

Computational Geometry
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Basic geometric objects

- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or p, $\overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

Basic geometric objects

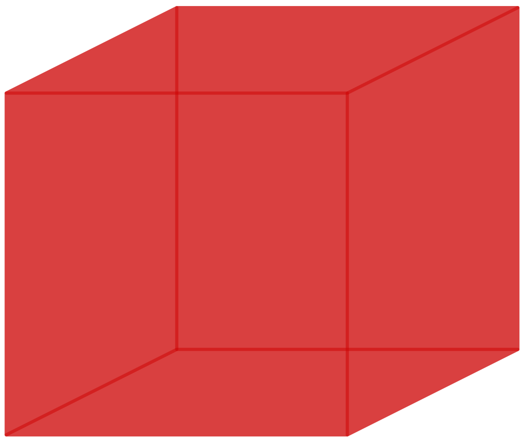
- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or p, $\overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

- Planar graphs such as planar decompositions and polyhedra:
  - For each vertex v:
    - Its coordinates (x, y, z)
    - An edge $e(v)$ incident to it
  - For each face f:
    - An edge $e(f)$ incident to it
  - For each (oriented) edge e:
    - Its starting and ending vertices $v_S(e), v_E(e)$
    - Its left and right incident faces $f_L(e), f_R(e)$
    - Its counterclockwise previous and next edges $e_P(e), e_N(e)$

Basic geometric objects

- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or $p, \overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

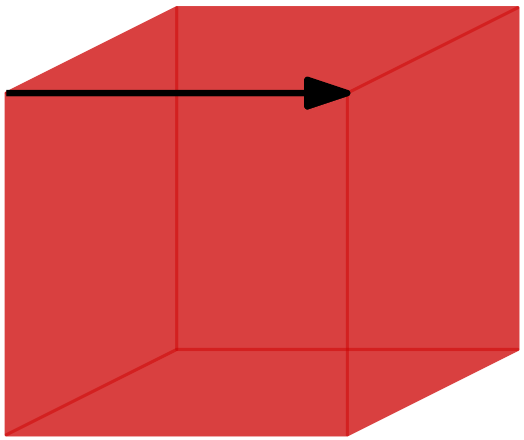
- Planar graphs such as planar decompositions and polyhedra:
  - For each vertex v:
    - Its coordinates (x, y, z)
    - An edge $e(v)$ incident to it
  - For each face f:
    - An edge $e(f)$ incident to it
  - For each (oriented) edge e:
    - Its starting and ending vertices $v_S(e), v_E(e)$
    - Its left and right incident faces $f_L(e), f_R(e)$
    - Its counterclockwise previous and next edges $e_P(e), e_N(e)$



Basic geometric objects

- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or $p, \overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

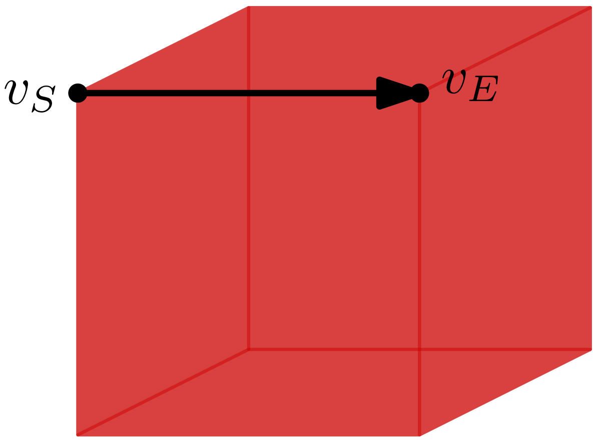
- Planar graphs such as planar decompositions and polyhedra:
  - For each vertex v:
    - Its coordinates (x, y, z)
    - An edge $e(v)$ incident to it
  - For each face f:
    - An edge $e(f)$ incident to it
  - For each (oriented) edge e:
    - Its starting and ending vertices $v_S(e), v_E(e)$
    - Its left and right incident faces $f_L(e), f_R(e)$
    - Its counterclockwise previous and next edges $e_P(e), e_N(e)$



Basic geometric objects

- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or $p, \overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

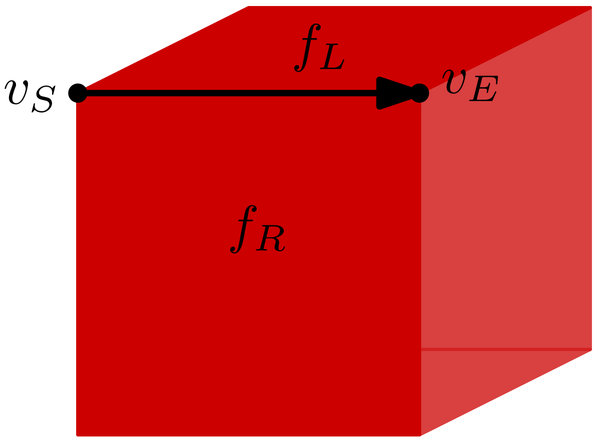
- Planar graphs such as planar decompositions and polyhedra:
  - For each vertex v:
    - Its coordinates (x, y, z)
    - An edge $e(v)$ incident to it
  - For each face f:
    - An edge $e(f)$ incident to it
  - For each (oriented) edge e:
    - Its starting and ending vertices $v_S(e), v_E(e)$
    - Its left and right incident faces $f_L(e), f_R(e)$
    - Its counterclockwise previous and next edges $e_P(e), e_N(e)$



Basic geometric objects

- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or $p, \overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

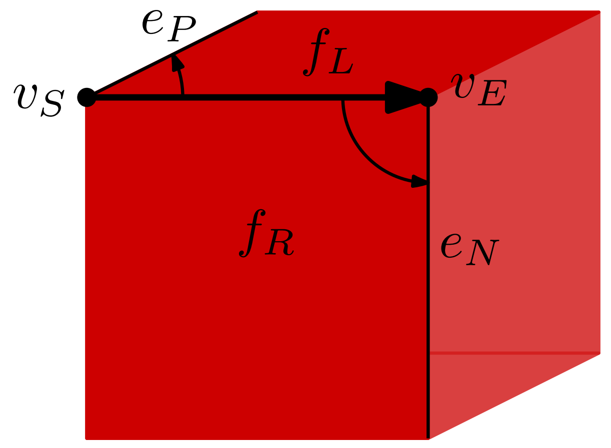
- Planar graphs such as planar decompositions and polyhedra:
  - For each vertex v:
    - Its coordinates (x, y, z)
    - An edge $e(v)$ incident to it
  - For each face f:
    - An edge $e(f)$ incident to it
  - For each (oriented) edge e:
    - Its starting and ending vertices $v_S(e), v_E(e)$
    - Its left and right incident faces $f_L(e), f_R(e)$
    - Its counterclockwise previous and next edges $e_P(e), e_N(e)$



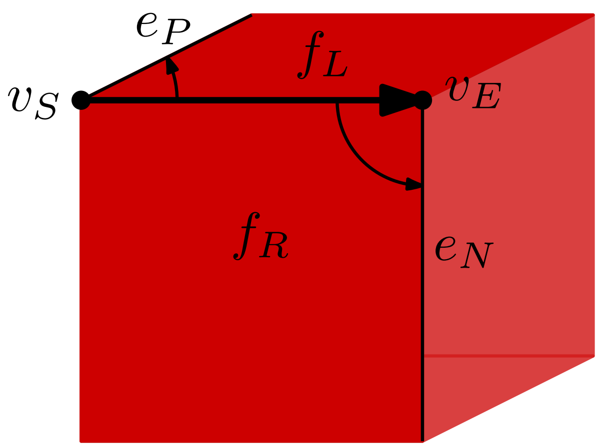
Basic geometric objects

- Points: $p = (x, y) \in \mathbb{R}^2$ and $(x, y, z) \in \mathbb{R}^3$
- Lines and line segments: p_1, p_2
- Halflines: p_1, p_2 or $p, \overrightarrow{v}$
- Polygonal lines and polygons: $p_1, p_2, \ldots, p_n$
- Circles and discs: p_1, p_2, p_3 non collinear
- Balls and spheres: p_1, p_2, p_3, p_4 non coplanar

- Planar graphs such as planar decompositions and polyhedra:
  - For each vertex v:
    - Its coordinates (x, y, z)
    - An edge $e(v)$ incident to it
  - For each face f:
    - An edge $e(f)$ incident to it
  - For each (oriented) edge e:
    - Its starting and ending vertices $v_S(e), v_E(e)$
    - Its left and right incident faces $f_L(e), f_R(e)$
    - Its counterclockwise previous and next edges $e_P(e), e_N(e)$

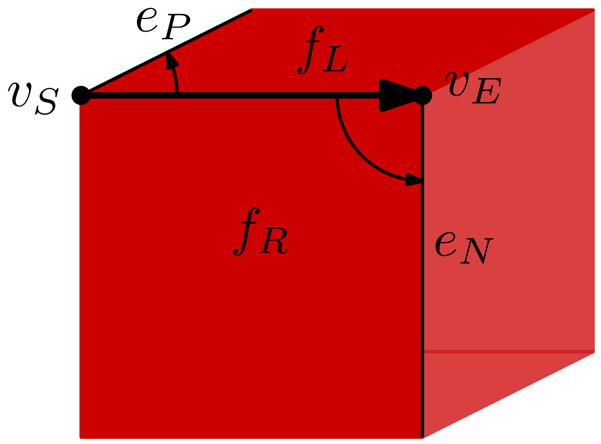


Why do we use a DCEL?



Why do we use a DCEL?

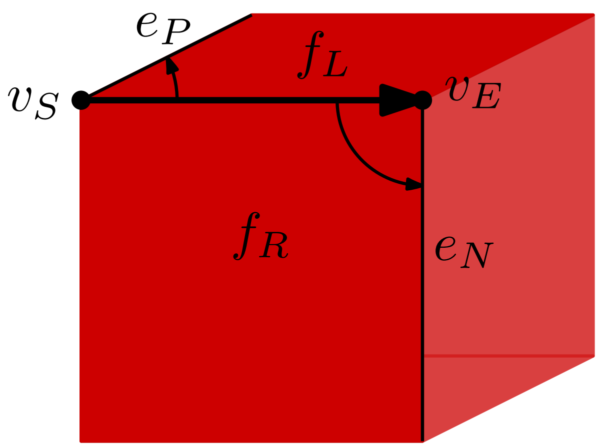
- Time:
  - It allows to obtain the sorted lists of vertices and edges of any face in time proportional to the size of the face.
  - It allows to obtain the sorted lists of edges and faces indicent to any vertex in time proportional to the degree of the vertex.



Why do we use a DCEL?

- Time:
  - It allows to obtain the sorted lists of vertices and edges of any face in time proportional to the size of the face.
  - It allows to obtain the sorted lists of edges and faces indicent to any vertex in time proportional to the degree of the vertex.

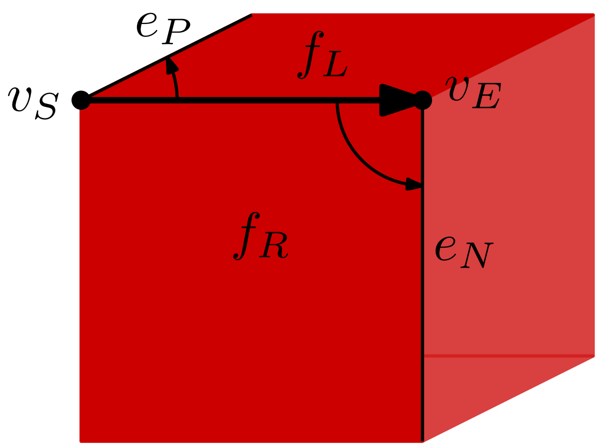
How?



Why do we use a DCEL?

- Time:
  - It allows to obtain the sorted lists of vertices and edges of any face in time proportional to the size of the face.
  - It allows to obtain the sorted lists of edges and faces indicent to any vertex in time proportional to the degree of the vertex.

How?

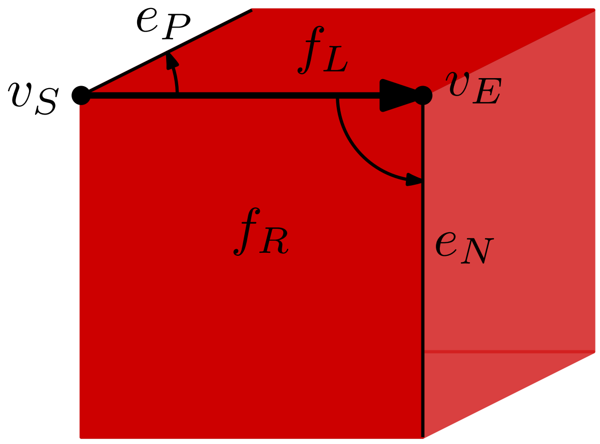


- Space:
  - The space it occupies is linear in the number of vertices:

Why do we use a DCEL?

- Time:
  - It allows to obtain the sorted lists of vertices and edges of any face in time proportional to the size of the face.
  - It allows to obtain the sorted lists of edges and faces indicent to any vertex in time proportional to the degree of the vertex.

How?



- Space:
  - The space it occupies is linear in the number of vertices:

Proof: Since the graph is planar, $V + F = E + 2$.
Since the faces are n-gons (with $n \geq 3$), $2E \geq 3F$.

Therefore:

$$2V + 2F = 2E + 4 \geq 3F + 4 \Longrightarrow F \leq 2V - 4.$$

$$3E + 6 = 3V + 3F \leq 3V + 2E \Longrightarrow E \leq 3V - 6.$$

How to deal with real numbers in a robust way?

How to deal with real numbers in a robust way?

$\mathbb{N}$ and $\mathbb{Z}$ are easy to deal with.

$\mathbb{Q}$ is $\mathbb{Z} \times \mathbb{Z}/\sim$, where $(a, b) \sim (a', b') \iff ab' = a'b$.

Algebraic numbers in $\mathbb{R}$ can be symbollically treated, but trascendental cannot.

$\mathbb{C}$ can be identified with $\mathbb{R}^2$.

$\mathbb{H}$ can be identified with $\mathbb{R}^4$.

DESCRIBING AND IMPLEMENTING BASIC GEOMETRIC OBJECTS

How to deal with real numbers in a robust way?

$\mathbb{N}$ and $\mathbb{Z}$ are easy to deal with.

$\mathbb{Q}$ is $\mathbb{Z} \times \mathbb{Z} / \sim$, where $(a, b) \sim (a', b') \iff ab' = a'b$.

Algebraic numbers in $\mathbb{R}$ can be symbollically treated, but trascendental cannot.

$\mathbb{C}$ can be identified with $\mathbb{R}^2$.

$\mathbb{H}$ can be identified with $\mathbb{R}^4$.

Recommended libraries

The GNU multiple-precision arithmetic library: https://gmplib.org/

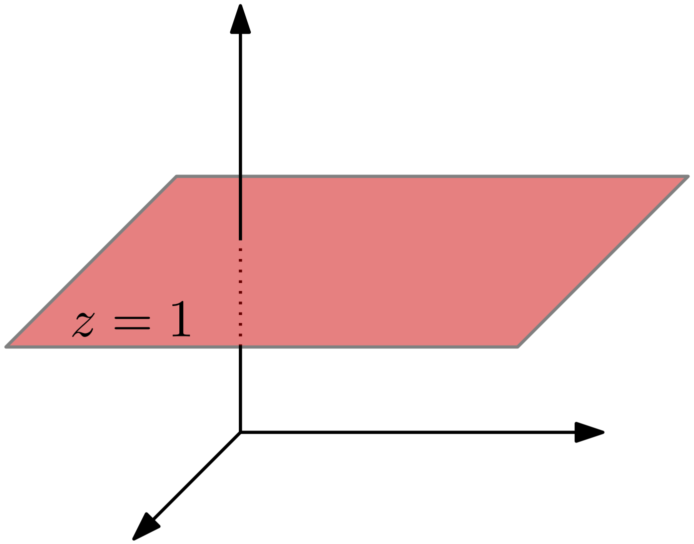
The GNU multiple-precision floating-point library: http://www.mpfr.org/

The Computational Geometry Algorithms Library: http://www.cgal.org/

Homogeneous coordinates

Homogeneous coordinates

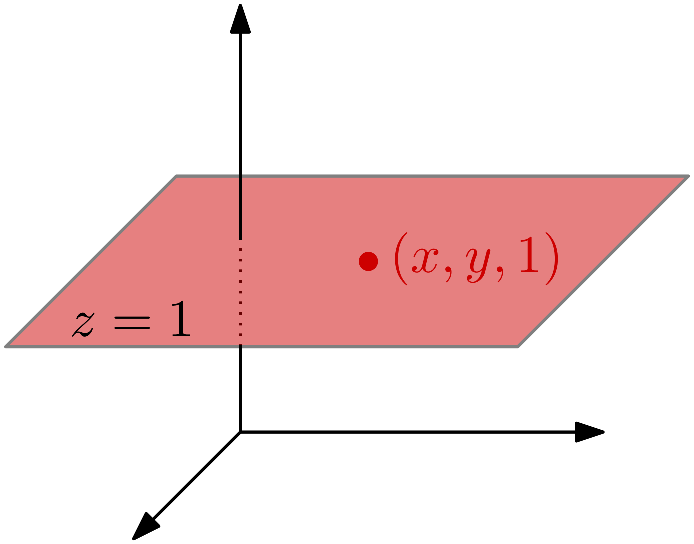
Embed the plane in 3-dimensional space as plane $z = 1$.



Homogeneous coordinates

Embed the plane in 3-dimensional space as plane $z = 1$.

This identifies every 2-dimensional point (x, y) with a 3-dimensional point $(x, y, 1)$.



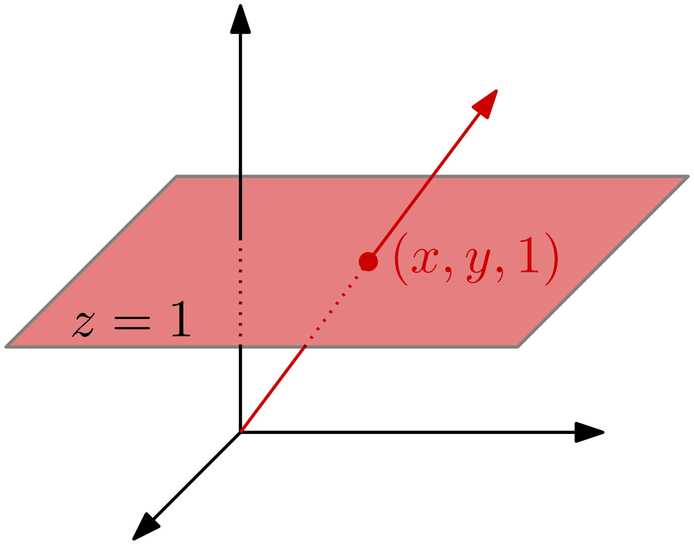
Homogeneous coordinates

Embed the plane in 3-dimensional space as plane $z = 1$.

This identifies every 2-dimensional point (x, y) with a 3-dimensional point $(x, y, 1)$.

Then identify point $(x, y, 1)$ with all the points in the ray connecting it to the origin:

$(x, y) \sim (x, y, 1) \sim (\lambda x : \lambda y : \lambda)$, where $\lambda > 0$.



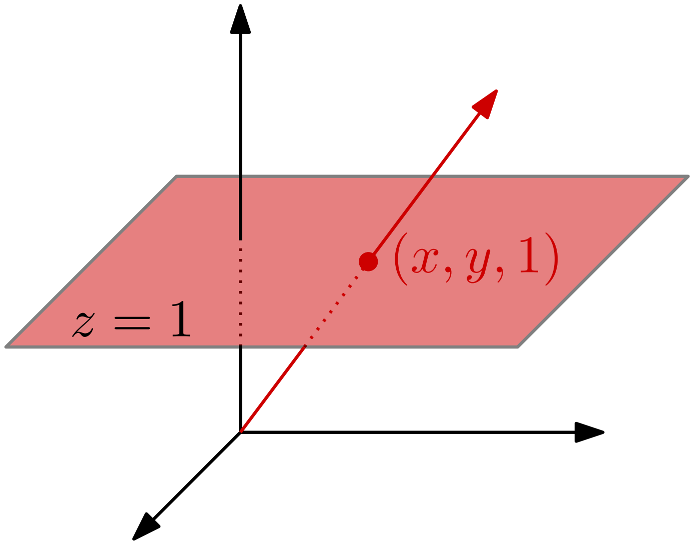
Homogeneous coordinates

Embed the plane in 3-dimensional space as plane $z = 1$.

This identifies every 2-dimensional point (x, y) with a 3-dimensional point $(x, y, 1)$.

Then identify point $(x, y, 1)$ with all the points in the ray connecting it to the origin:

$(x, y) \sim (x, y, 1) \sim (\lambda x : \lambda y : \lambda)$, where $\lambda > 0$.



Welcome to **projective geometry**!

Homogeneous coordinates

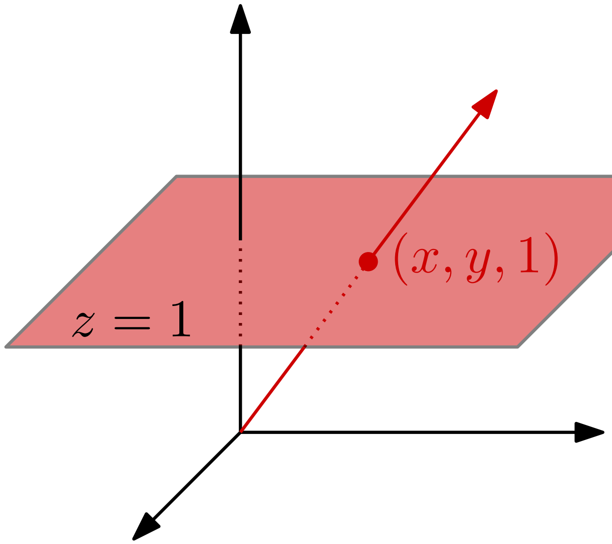
Embed the plane in 3-dimensional space as plane $z = 1$.

This identifies every 2-dimensional point (x, y) with a 3-dimensional point $(x, y, 1)$.

Then identify point $(x, y, 1)$ with all the points in the ray connecting it to the origin:

$(x, y) \sim (x, y, 1) \sim (\lambda x : \lambda y : \lambda)$, where $\lambda > 0$.

Welcome to **projective geometry**!



What about vectors?

- Vectors have coordinates $(x : y : 0)$
  because they are parallel to the plane $z = 1$.
- They correspond to horizontal rays, i.e., to points at infinity.

Homogeneous coordinates are very useful in practice!

Homogeneous coordinates are very useful in practice!

Points:

$$(x, y) \longleftrightarrow (x : y : 1)$$

Lines:

$$ax + by + c = 0 \longleftrightarrow (a : b : c)$$

Homogeneous coordinates are very useful in practice!

Points:

$$(x, y) \longleftrightarrow (x : y : 1)$$

Lines:

$$ax + by + c = 0 \longleftrightarrow (a : b : c)$$

Incidence point-line

$$p = (x : y : 1) \text{ lies in } \ell = (a : b : c) \iff (a, b, c) \cdot (x, y, 1) = 0$$
$$\iff p \perp \ell$$

Homogeneous coordinates are very useful in practice!

Points:

$$(x, y) \longleftrightarrow (x : y : 1)$$

Lines:

$$ax + by + c = 0 \longleftrightarrow (a : b : c)$$

Incidence point-line

$$p = (x : y : 1) \text{ lies in } \ell = (a : b : c) \quad \Longleftrightarrow (a, b, c) \cdot (x, y, 1) = 0$$
$$\Longleftrightarrow p \perp \ell$$

Line through two points

$$\left.\begin{array}{l} p \text{ lies in } \ell \Longrightarrow p \perp \ell \\ q \text{ lies in } \ell \Longrightarrow q \perp \ell \end{array}\right\} \Longrightarrow \ell = p \times q$$

Homogeneous coordinates are very useful in practice!

Points:

$$(x, y) \longleftrightarrow (x : y : 1)$$

Lines:

$$ax + by + c = 0 \longleftrightarrow (a : b : c)$$

Incidence point-line

$$p = (x : y : 1) \text{ lies in } \ell = (a : b : c) \iff (a, b, c) \cdot (x, y, 1) = 0 \iff p \perp \ell$$

Line through two points

$$\left.\begin{array}{l} p \text{ lies in } \ell \Longrightarrow p \perp \ell \\ q \text{ lies in } \ell \Longrightarrow q \perp \ell \end{array}\right\} \Longrightarrow \ell = p \times q$$

Intersecting two lines

$$\left.\begin{array}{l} p \text{ lies in } \ell_1 \Longrightarrow p \perp \ell_1 \\ p \text{ lies in } \ell_2 \Longrightarrow p \perp \ell_2 \end{array}\right\} \Longrightarrow p = \ell_1 \times \ell_2$$

Homogeneous coordinates are very useful in practice!

Parallel lines

$$\ell_1 \| \ell_2 \Longleftrightarrow \left\{ \begin{array}{l} \ell_1 = (a, b, c) \\ \ell_2 = (\lambda a, \lambda b, d) \end{array} \right\} \Longleftrightarrow \ell_1 \times \ell_2 = (x, y, 0)$$

Homogeneous coordinates are very useful in practice!

Parallel lines

$$\ell_1 \| \ell_2 \Longleftrightarrow \left\{ \begin{array}{l} \ell_1 = (a, b, c) \\ \ell_2 = (\lambda a, \lambda b, d) \end{array} \right\} \Longleftrightarrow \ell_1 \times \ell_2 = (x, y, 0)$$

The line at infinity

$$(x_1 : y_1 : 0) \times (x_2 : y_2 : 0) = (0 : 0 : 1)$$

Homogeneous coordinates are very useful in practice!

Parallel lines

$$\ell_1 \| \ell_2 \Longleftrightarrow \left\{ \begin{array}{l} \ell_1 = (a, b, c) \\ \ell_2 = (\lambda a, \lambda b, d) \end{array} \right\} \Longleftrightarrow \ell_1 \times \ell_2 = (x, y, 0)$$

The line at infinity

$$(x_1 : y_1 : 0) \times (x_2 : y_2 : 0) = (0 : 0 : 1)$$

Identical lines (or points)

$$\ell_1 = \ell_2 \Longleftrightarrow \left\{ \begin{array}{l} \ell_1 = (a, b, c) \\ \ell_2 = (\lambda a, \lambda b, \lambda c) \end{array} \right\} \Longleftrightarrow \ell_1 \times \ell_2 = (0, 0, 0)$$

Homogeneous coordinates are very useful in practice!

If you implement the cross product, you get a code that is:

- **Correct:** it computes exactly what it should.
- **Reusable:** it is used to solve several (apparently different) problems.
- **Efficient:** 6 products and 3 additions.
- **Robust:** it handles all cases, including degeneracies.

Homogeneous coordinates are very useful in practice!

If you implement the cross product, you get a code that is:

- **Correct:** it computes exactly what it should.
- **Reusable:** it is used to solve several (apparently different) problems.
- **Efficient:** 6 products and 3 additions.
- **Robust:** it handles all cases, including degeneracies.

There is no way you can do the same using affine coordinates!

Homogeneous coordinates are very useful in practice!

If you implement the cross product, you get a code that is:

- **Correct:** it computes exactly what it should.
- **Reusable:** it is used to solve several (apparently different) problems.
- **Efficient:** 6 products and 3 additions.
- **Robust:** it handles all cases, including degeneracies.

There is no way you can do the same using affine coordinates!

FURTHER READING

Read about CGAL!

- At this cours web page (topic 13)
- At CGAL web page