# Eurographics 2013

## Projective Geometry, Duality and Precision of Computation in Computer Graphics, Visualization and Games



**Tutorial**

**Vaclav Skala**
University of West Bohemia, Plzen, Czech Republic
VSB-Technical University, Ostrava, Czech Republic
http://www.VaclavSkala.eu



# Eurographics 2013

## Plzen (Pilsen) City


Plzen is an old city [first records of Plzen castle 976] city of culture, industry, and brewery.

City, where today's beer fermentation process was invented that is why today's beers are called Pilsner - world wide



# Eurographics 2013

## Ostrava City

Ostrava is

- an industrial city of coal mining & iron melting
- 3rd largest city




# Eurographics 2013

**University of West Bohemia 17530 students + 987 PhD students**

**Computer Science and Engineering** **Mathematics** (+ Geomatics)
**Physics** **Cybernetics** **Mechanics** (Computational)

- Over **50%** of income from research and application projects
- New research center EU project - investment 64 mil. EUR
- 2nd in the ranking of Czech technical / informatics faculties 2009, 2012




# Eurographics 2013

**An overview**

- Precision and robustness
- Euclidean space and projective extension
- Principle of duality and its applications
- Geometric computation in the projective space
- Intersection of two planes in E3 with additional constrains
- Barycentric coordinates and intersections
- Interpolation and intersection algorithms
- Implementation aspects and GPU
- Conclusion and summary



# Eurographics 2013

**Numerical systems**

- Binary system is used nearly exclusively
- Octal & hexadecimal representation is used
- If we would be direct descendants of tetrapods – we would have a great advantage – "simple counting in hexadecimal system"

| | Name | Base | Digits | E min | E max |
|---|---|---|---|---|---|
| | | | **BINARY** | | |
| B 16 | Half | 2 | 10+1 | −14 | 15 |
| B 32 | Single | 2 | 23+1 | −126 | 127 |
| B 64 | Double | 2 | 52+1 | −1022 | 1023 |
| B 128 | Quad | 2 | 112+1 | −16382 | 16383 |
| | | | **DECIMAL** | | |
| D 32 | | 10 | 7 | −95 | 96 |
| D 64 | | 10 | 16 | −383 | 384 |
| D 128 | | 10 | 34 | −6143 | 6144 |

IEEE 758-2008 standard

Courtesy Clive "Max" Maxfield and Alvin Brown

The first tetrapods had eight fingers on each hand

# Eurographics 2013

**Mathematically perfect algorithms fail due to instability**

**Main issues**

- stability, robustness of algorithms
- acceptable speed
- linear speedup – results depends on HW, CPU …. parameters !

**Numerical stability**

- limited precision of float / double
- tests A ? B with floats

  **if** A = B **then** ….. **else** ….. ; **if** A = 0 **then** ….. **else** ….
  should be forbidden in programming languages

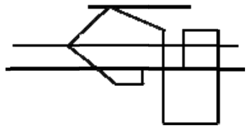
- division operation should be removed or postponed to the last moment if possible - "blue screens", system resets



# Eurographics 2013

**Typical examples of instability**

- intersection of 2 lines in E3
- point lies on a line in E2 or a plane in E3
  Ax + By + C = 0 or Ax + By + Cz + D = 0
- detection if a line intersects a polygon, touches a vertex or passes through



**Typical problem**

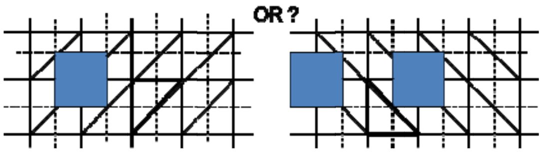
```
double x = -1; double p = ….;
while ( x < +1)
{   if (x == p) Console.Out.WriteLine(" *** ")
    x += p;
}
/*   if p = 0.1 then no output,   if p = 0.25 then expected output */
```



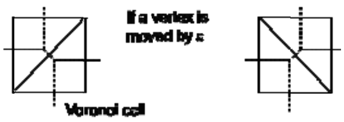
# Eurographics 2013

**Delaunay triangulation & Voronoi diagram**

Point inside of a circle given by three points – problems with meshing points in regular rectangular grid.




It can be seen that the DT & is very sensitive to a point position change



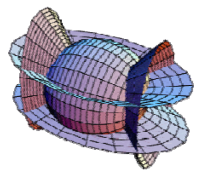

VD

**?? ROBUSTNESS ??**



# Eurographics 2013

**Vectors and Points in Geometry**



Courtesy of http://mathworld.wolfram.com/ConfocalEllipsoidalCoordinates.html

**Vectors** – movable, no fixed position

- Points – no size, position fixed in the GIVEN coordinate system

**Coordinate systems**:

- Cartesian – left / right handed
  right handed system is used
- Polar
- Spherical, Cylindrical
- and many others, e.g. Confocal Ellipsoidal Coordinates
  http://mathworld.wolfram.com/ConfocalEllipsoidalCoordinates.html



# Eurographics 2013

**Floating point**

- Not all numbers are represented correctly
- Logarithmic arithmetic
- Continuous fractions
- Interval arithmetic

$$\pi = \cfrac{4}{1 + \cfrac{1^2}{3 + \cfrac{2^2}{5 + \cfrac{3^2}{\dots}}}}$$

$\pi = [3; 7,15,1,292,1,1,1,2,1,3,1 \dots]$

x + y = [a + c, b + d]   x = [ a , b ]
x - y = [a - d, b - c]   y = [ c , d ]
x × y = [min(ac, ad, bc, bd), max(ac, ad, bc, bd)]
x / y = [min(a/c, a/d, b/c, b/d), max(a/c, a/d, b/c, b/d)] if y ≠ 0

**Numerically NOT valid identities** due to limited precision

- $cos^2\alpha + cos^2\beta = 1$
- $x^2 - y^2 = (x - y)(x + y)$



# Eurographics 2013

**Statements like**

**if** <float> = <float> **then ….** or **if** <float> ≠ <float> **then ….**

**should not be allowed in programming languages**

**Quadratic equation - more reliable results**

$at^2 + bt + c = 0$ usually solved as $t_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

If $b^2 \gg 4ac$ then

$q = -(b + sign(b)\sqrt{b^2 - 4ac})/2$ $\quad t_1 = {}^{q}/_{a}$ $\quad t_2 = {}^{c}/_{a}$

The discriminant should be computed with a twice precision

**Vieta's formula** $\quad t_1 + t_2 = -{}^{b}/_{a}$ $\quad t_1 t_2 = {}^{c}/_{a}$

## Eurographics 2013

**Function value computation** at $x = 77617,\ y = 33096$

$$f(x,y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

$f = 6.33835\ 10^{29}$ single precision

$f = 1{,}1726039400532$ double precision

$f = 1{,}172603940053178631858834904520183{8}$ extended precision

The correct result is "somewhere" in the interval of

$[-0{,}827396059946821368141165095479816292\mathbf{005},$
$-0{,}827396059946821368141165095479816291\mathbf{986}]$

Exact solution

$$f(x,y) = -2 + \frac{x}{2y} = \frac{54767}{66192}$$



## Eurographics 2013

**Summation is one of often used computations.**

$$\sum_{i=1}^{10^3} 10^{-3} = 0.999990701675415$$

or

$$\sum_{i=1}^{10^4} 10^{-4} = 1.000053524971008$$

The result should be only one.

**The correctness in summation is very important in power series computations.**
**!!!!ORDER of summation**

$$\sum_{n=1}^{10^6} \frac{1}{n} = 14.357357 \qquad \sum_{n=10^6}^{1} \frac{1}{n} = 14.392651$$



## Eurographics 2013

**Recursion**

Towers of Hanoi

```
MOVE (A, C, n);
{   MOVE (A, B, n-1);
    MOVE (A, C, 1);
    MOVE (B, C, n-1)
}  # MOVE (from, to, number) #
```

Ackermann function

$$A(m,n) = \begin{cases} n+1 & if\ m = 0 \\ A(m-1,1) & if\ M > 0\ and\ n = 0 \\ A\big(m-1, A(m,n-1)\big) & if\ m > 0\ and\ N > 0 \end{cases}$$

The value of the function grows very fast as

$$A(4,4) = 2^{2^{2^{65536}}} = 2^{2^{10^{197296}}}$$

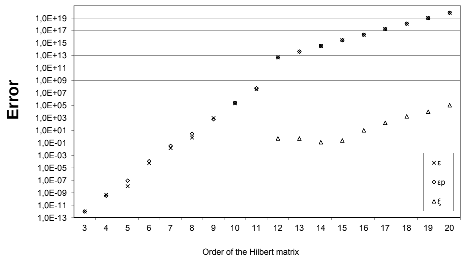


## Eurographics 2013

**Numerical computations** $\boldsymbol{Ax} = \boldsymbol{b}$ $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$

**Hilbert's Matrix Inversion** $H_{ij} = \frac{1}{i+j-1}$

$$H_{ij}^{-1} = (-1)^{i+j}(i+j-1)\binom{n+i-1}{n-j}\binom{n+j-1}{n-i}\binom{i+j-2}{i-1}^2$$






## Eurographics 2013

**Mathematical "forms"** **There are several "forms":**

**Implicit** $F(x,y,z) = 0$ or $F(\boldsymbol{x}) = 0$ or $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$ (system of equations)

There is no orientation, e.g.

- if $F(\boldsymbol{x}) = 0$ is a iso-curve there is no hint how to find another point of this curve, resp. a line segment approximating the curve => tracing algorithms
- if $F(\boldsymbol{x}) = 0$ is a iso-surface there is no hint how to find another point of this surface => iso-surface extraction algorithms

**Parametrical** $\boldsymbol{x} = \boldsymbol{x}(u)$ $\boldsymbol{x} = \boldsymbol{x}(u,v)$

Points of a curve are "ORDERED" according to a parameter $u$, resp. $u, v$

**Explicit** $z = f(x)$ $z = f(x,y)$ [actually 2 ½ D]

For the given value $x$, resp. $x, y$ we get function value $z$



## Eurographics 2013

**Implicit form**

- Is used for separation - for detection if a point is inside or outside, e.g. a half-plane or a circle etc.
- There is always a question how to compute $\boldsymbol{x}$ of $F(\boldsymbol{x}) = 0$ as there are several solutions in general
- Complexity of computations × precision of computation

Compiler optimization is **DANGEROUS** in general - numerical precision

$$x^2 - y^2 = (x+y)(x-y)$$

$$\begin{vmatrix} A_x & A_y & A_x^2 + A_y^2 & 1 \\ B_x & B_y & B_x^2 + B_y^2 & 1 \\ B_x & B_y & B_x^2 + B_y^2 & 1 \\ B_x & B_y & B_x^2 + B_y^2 & 1 \end{vmatrix} = \begin{vmatrix} A_x - D_x & A_y - D_y & (A_x^2 - D_x^2) + (A_y^2 - D_x^2) \\ B_x - D_x & B_y - D_y & (B_x^2 - D_x^2) + (B_y^2 - D_x^2) \\ C_x - D_x & C_y - D_y & (C_x^2 - D_x^2) + (C_y^2 - D_x^2) \end{vmatrix} > 0$$

## Eurographics 2013

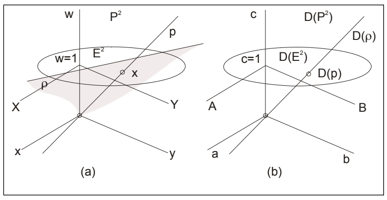
**Projective Space**

$\boldsymbol{X} = [X, Y]^T \qquad \boldsymbol{X} \in E^2$

$\boldsymbol{x} = [x, y : w]^T \qquad \boldsymbol{x} \in P^2$

**Conversion:**

$\boldsymbol{X} = [x/w, y/w]^T \qquad w \neq 0$




If $w = 0$ then $\boldsymbol{x}$ represents "an ideal point" - a point in infinity, i.e. it is a directional vector.

The Euclidean space $E^2$ is represented as a plane $w = 1$.

Equivalent "mathematical" notation often used:

$\boldsymbol{x} = [w : x, y]^T$ generally for $E^n$ $\boldsymbol{x} = [x_0 : x_1, \dots, x_n]^T$

i.e. homogeneous coordinate is the first



## Eurographics 2013

**Points and vectors**

- Vectors **are "freely movable"** – not having a fixed position

$$\boldsymbol{a}_1 = [x_1, y_1 : 0]^T$$

- Points are **not "freely movable"** – they are fixed to an origin of the current coordinate system

$$\boldsymbol{x}_1 = [x_1, y_1 : w_1]^T \quad \text{and} \quad \boldsymbol{x}_2 = [x_2, y_2 : w_2]^T$$

usually in textbooks $w_1 = w_2 = 1$

A vector $\boldsymbol{A} = \boldsymbol{X}_2 - \boldsymbol{X}_1$ in the Euclidean coordinate system - **CORRECT**

**Horrible "construction" DO NOT USE IT – IT IS TOTALLY WRONG**

$$\boldsymbol{a} = \boldsymbol{x}_2 - \boldsymbol{x}_1 = [x_2 - x_1, y_2 - y_1 : w_2 - w_1]^T = [x_2 - x_1, y_2 - y_1 : 1 - 1]^T = [x_2 - x_1, y_2 - y_1 : 0]^T$$

This was presented as "How a vector" is constructed in the projective space $P^k$ in a textbook!! **WRONG, WRONG, WRONG**

*This construction has been found in SW as well*!!



## Eurographics 2013

**Points and vectors**

A Euclidean vector given by two points in the homogeneous coordinates

$$\boldsymbol{a} = \boldsymbol{x}_2 - \boldsymbol{x}_1 = [w_1 x_2 - w_2 x_1, w_1 y_2 - w_2 y_1 : w_1 w_2]^T$$

This is the **CORRECT SOLUTION**, but what is the interpretation?

**A "difference" of coordinates of two points is a vector in the mathematical meaning and $w_1 w_2$ is a "scaling" factor actually**

Actually the division operation is postponed and not performed immediately. A vector in projective notation

$$\boldsymbol{a} = \boldsymbol{x}_2 - \boldsymbol{x}_1 = [w_1 x_2 - w_2 x_1, w_1 y_2 - w_2 y_1 : w_1 w_2]^T \triangleq \left[\frac{w_1 x_2 - w_2 x_1}{w_1 w_2}, \frac{w_1 y_2 - w_2 y_1}{w_1 w_2} : 0\right]^T$$

where: $\triangleq$ means projectively equivalent



## Eurographics 2013

A Euclidean vector in the projective representation (if the vector length matters)

$$\boldsymbol{a} = \boldsymbol{x}_2 - \boldsymbol{x}_1 = [w_1 x_2 - w_2 x_1, w_1 y_2 - w_2 y_1 : w_1 w_2]^T \triangleq \left[\frac{w_1 x_2 - w_2 x_1}{w_1 w_2}, \frac{w_1 y_2 - w_2 y_1}{w_1 w_2} : 0\right]^T$$

A vector in the projective space is given by coordinates $x, y, w$ as

$$\boldsymbol{a} = \boldsymbol{x}_2 - \boldsymbol{x}_1 = [x_2 - x_1, y_2 - y_1, w_2 - w_1]^T$$

[=>Linear interpolation with a non-linear monotonic interpolation]

We have to strictly distinguish meaning of one *dimensional array* [*vector*], i.e. if we are working with:

- **points**, i.e. a data structure represent point coordinates, or
- **vectors**, i.e. a data structure represent a vector in the mathematical meaning

**VECTORS x POINTS**



## Eurographics 2013

**Duality**

For simplicity, let us consider a line $p$ defined as:

$$aX + bY + c = 0$$

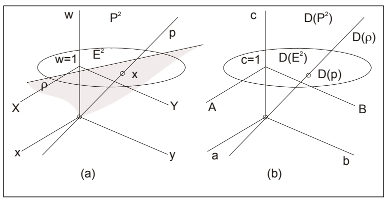
We can multiply it by $w \neq 0$ and we get:

$$awX + bwY + cw = 0$$

As $x = wX$ and $y = wY$

$ax + by + cw = 0$ i.e. $\boldsymbol{p}^T \boldsymbol{x} = 0$

$\boldsymbol{p} = [a, b : c]^T \qquad \boldsymbol{x} = [x, y : w]^T = [wX, wY : w]^T$




A line $p \in E^2$ is actually a plane in the projective space $P^2$ (point $\boldsymbol{x} = [0,0 : 0]^T$ excluded)



## Eurographics 2013

**Duality**

From the mathematical notation $\qquad \boldsymbol{p}^T \boldsymbol{x} = 0$

we cannot distinguish whether $\boldsymbol{p}$ is a line and $\boldsymbol{x}$ is a point or vice versa in the case of $P^2$. It means that

- a *point* and a *line* **are dual** in the case of $P^2$, and
- a *point* and a *plane* **are dual** in the case of $P^3$.

The principle of duality in $P^2$ states that:

Any theorem in $E^2$ remains true when we interchange the words "point" and "line", "lie on" and "pass through", "join" and "intersection", "collinear" and "concurrent" and so on.

Similarly for the $E^3$ case.

**Once the theorem has been established, the dual theorem is obtained as described above.**

This helps a lot to solve some geometrical problems.

# Eurographics 2013

**Examples of dual objects and operators**

| | **Primitive** | **Dual primitive** |
|---|---|---|
| $P^2$ | Point | Line |
| | Line | Point |
| $P^3$ | Point | Plane |
| | Plane | Point |

| **Operator** | **Dual operator** |
|---|---|
| Join | Intersect |
| Intersect | Join |

***Computational sequence for a problem is the same for a dual problem.***



# Eurographics 2013

**Intersection of two lines**

Let two lines $p_1$ and $p_2$ are given by

$$p_1 = [a_1, b_1 : c_1]^T \quad \text{and} \quad p_2 = [a_2, b_2 : c_2]^T$$

We have to solve a system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} -c_1 \\ -c_2 \end{bmatrix} *$$

Then well known formula is used

$$x = \frac{Det_x}{Det} = \frac{det\begin{bmatrix} q_1 & b_1 \\ q_2 & b_2 \end{bmatrix}}{det\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}} \qquad y = \frac{Det_x}{Det} = \frac{det\begin{bmatrix} a_1 & q_1 \\ a_2 & q_2 \end{bmatrix}}{det\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}}$$

But what if $Det$ is small?

Usually a sequence like $\boldsymbol{if}\ abs(\det(..)) \le eps\ \boldsymbol{then}$ is used. What is $eps$?

**Note** * usually a line is in its explicit form as $ax + by = q$ instead of $ax + by + c = 0$, i.e. the implicit form



# Eurographics 2013

**Definition**

The cross product of the two vectors

$$\boldsymbol{x}_1 = [x_1, y_1 : w_1]^T \text{ and } \boldsymbol{x}_2 = [x_2, y_2 : w_2]^T$$

is defined as:

$$\boldsymbol{x}_1 \times \boldsymbol{x}_2 = det\begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix}$$

where: $\boldsymbol{i} = [1,0:0]^T \quad \boldsymbol{j} = [0,1:0]^T \quad \boldsymbol{k} = [0,0:1]^T$

or as

$$\boldsymbol{x}_1 \times \boldsymbol{x}_2 = \begin{bmatrix} 0 & -w_1 & y_1 \\ w_1 & 0 & -x_1 \\ -y_1 & x_1 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ w_2 \end{bmatrix} = \boldsymbol{T}\boldsymbol{x}_2$$

Please, note that homogeneous coordinates are used.



# Eurographics 2013

**Theorem**

Let two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be given in the projective space. Then the coefficients of the $\boldsymbol{p}$ line, which is defined by those two points, are determined as the cross product of their homogeneous coordinates

$$\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 = [a, b : c]^T$$

**Proof**

Let the line $P^2$ be defined in homogeneous coordinates as

$$ax + by + cw = 0$$

We are actually looking for a solution to the following equations:

$$\boldsymbol{p}^T \boldsymbol{x}_1 = 0 \qquad \boldsymbol{p}^T \boldsymbol{x}_2 = 0$$

where: $\boldsymbol{p} = [a, b : c]^T$

Note that $c$ represents a "distance" from the origin of the coordinate system.



# Eurographics 2013

It means that any point $\boldsymbol{x}$ that lies on the $p$ line must satisfy both the equation above and the equation $\boldsymbol{p}^T\boldsymbol{x} = 0$ in other words the $\boldsymbol{p}$ vector is defined as

$$\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 = det\begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix}$$

We can write

$$(\boldsymbol{x}_1 \times \boldsymbol{x}_2)^T \boldsymbol{x} = 0 \qquad \text{i.e.} \quad det\begin{bmatrix} x & y & w \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} = 0$$

Note that the Cross product and the Dot product is an instruction in Cg/HLSL on GPU.



# Eurographics 2013

Evaluating the determinant $det\begin{bmatrix} a & b & c \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} = 0$

we get the line coefficients of the line $p$ as:

$$a = det\begin{bmatrix} y_1 & w_1 \\ y_2 & w_2 \end{bmatrix} \qquad b = -det\begin{bmatrix} x_1 & w_1 \\ x_2 & w_2 \end{bmatrix} \qquad c = det\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix}$$

**Note**:

1. A line $ax + by + c = 0$ is a one parametric set of coefficients
   $\boldsymbol{p} = [a, b : c]^T$
   From two values $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ we have to compute 3 values, coefficients $a$, $b$ and $c$

2. For w = 1 we get the standard cross product formula and the cross product defines the $p$ line, i.e. $\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2$ where:
   $\boldsymbol{p} = [a, b : c]^T$

# Eurographics 2013

We have seen that computation of

- an intersection of two lines is given as $\boldsymbol{Ax} = \boldsymbol{b}$
- a line given by two points is given as $\boldsymbol{Ax} = \boldsymbol{0}$

**Different scheme BUT**

**Those problems are DUAL.**

**Why algorithms are different??**

**Cross product is equivalent to a solution of a linear system of equations!**

**No division operations!**



# Eurographics 2013

**DUALITY APPLICATION**

In the projective space $P^2$ points and lines are dual. Due to duality we can directly intersection of two lines as

$$\boldsymbol{x} = \boldsymbol{p}_1 \times \boldsymbol{p}_2 = det \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{bmatrix} = [x, y: w]^T$$

If the lines are parallel or close to parallel, the homogeneous coordinate $w \to 0$ and users have to take a decision – so there is no sequence in the code like $\boldsymbol{if}\ abs(\det(..)) \le eps\ \boldsymbol{then}$ …in the procedure.

Generally computation can continue even if $w \to 0$ if projective space is used.



# Eurographics 2013

**DISTANCE**

Geometry is strongly connected with distances and their measurement, geometry education is strictly sticked to the Euclidean geometry, where the distance is measured as

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2} \quad \text{, resp.} \quad d = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}\ .$$

This concept is convenient for a solution of basic geometric problems, but in many cases it results into quite complicated formula and there is a severe question of stability and robustness in many cases.

***The main objection against the projective representation is that there is no metric.***



# Eurographics 2013

The distance of two points can be easily computed as

$$dist = \sqrt{\xi^2 + \eta^2}/(w_1 w_2)$$

where: $\xi = w_1 x_2 - w_2 x_1$ $\quad \eta = w_1 y_2 - w_2 y_1$

Also a distance of a point $\boldsymbol{x}_0$ from a line in $E^2$ can be computed as

$$dist = \frac{\boldsymbol{a}^T \boldsymbol{x}_0}{w_0 \sqrt{a^2 + b^2}}$$

where: $\boldsymbol{x_0} = [x_0, y_0: w_0]^T$ $\quad \boldsymbol{a} = [a, b: c]^T$

The extension to $E^3/P^3$ is simple and the distance of a point $\boldsymbol{x}_0$ from a plane in $E^3$ can be computed as

$$dist = \frac{\boldsymbol{a}^T \boldsymbol{x}_0}{w_0 \sqrt{a^2 + b^2 + c^2}}$$

where: $\boldsymbol{x_0} = [x_0, y_0, z_0: w_0]^T$ $\quad \boldsymbol{a} = [a, b, c: d]^T$.



# Eurographics 2013

In many cases we do not need actually a *distance*, e.g. for a decision which object is closer, and *distance*$^2$ can be used instead, i.e. for the $E^2$ case

$$dist^2 = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{w_0{}^2 (a^2 + b^2)} = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{w_0{}^2\, \boldsymbol{n}^T \boldsymbol{n}}$$

where: $\boldsymbol{a} = [a, b : c]^T = [\boldsymbol{n} : c]^T$ and the normal vector $\boldsymbol{n}$ is not normalized

If we are comparing distances of points from the given line $p$ we can use "*pseudo-distance*" for comparisons

$$(pseudo\_dist)^2 = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{w_0{}^2}$$

Similarly for a plane $\rho$ in the case of $E^3$

$$dist^2 = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{w_0{}^2 (a^2 + b^2 + c^2)} = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{w_0{}^2\, \boldsymbol{n}^T \boldsymbol{n}} \quad \text{and} \quad (pseudo\_dist)^2 = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{w_0{}^2}$$

where: $\boldsymbol{a} = [a, b, c : d]^T = [\boldsymbol{n} : d]^T$



# Eurographics 2013

**Computation in Projective Space**

- Cross product definition
- A plane $\boldsymbol{\rho}$ is determined as a cross product of three given points

$$\boldsymbol{x}_1 \times \boldsymbol{x}_2 \times \boldsymbol{x}_3 = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} & \boldsymbol{l} \\ x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \\ x_3 & y_3 & z_3 & w_3 \end{vmatrix}$$

**Due to the duality**

- An intersection point $\boldsymbol{x}$ of three planes is determined as a cross product of three given planes.

$$\boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_3 = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} & \boldsymbol{l} \\ a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{vmatrix}$$

**Computation of generalized cross product is equivalent to a solution of a linear system of equations**
**=> no division operation!**

## Eurographics 2013

**Geometric transformations with points**

(note $X = {}^{x}/_{w}$ , $Y = {}^{y}/_{w}$ , $w \neq 0$ ):

**Translation** by a vector $(A, B) \triangleq [a, b: c]^T$ , i.e. $A = a/c$ , $B = b/c$ , $c \neq 0$:

In the Euclidean space: $\boldsymbol{x}' = \boldsymbol{T}\boldsymbol{x}$

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & A \\ 0 & 1 & B \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} x + Aw \\ y + Bw \\ w \end{bmatrix} \triangleq \begin{bmatrix} x/w + A \\ y/w + B \\ 1 \end{bmatrix} = \begin{bmatrix} X + A \\ Y + A \\ 1 \end{bmatrix}$$

In the projective space: $\boldsymbol{x}' = \boldsymbol{T}'\boldsymbol{x}$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} c & 0 & a \\ 0 & c & b \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} cx + aw \\ cy + bw \\ cw \end{bmatrix} \triangleq \begin{bmatrix} (cx + aw)/(cw) \\ (cy + bw)/(cw) \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{x}/_{w} + {}^{a}/_{c} \\ {}^{y}/_{w} + {}^{b}/_{c} \\ 1 \end{bmatrix} = \begin{bmatrix} X + A \\ Y + B \\ 1 \end{bmatrix}$$

and $\det(\boldsymbol{T}') = c^3$. For $c = 1$ we get a standard formula

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & A \\ 0 & 1 & B \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



## Eurographics 2013

**Rotation** by an angle $(cos\varphi, sin\varphi) = \left(\frac{a}{c}, \frac{b}{c}\right) \triangleq [a, b: c]^T$

In the Euclidean space: $\boldsymbol{x}' = \boldsymbol{R}\boldsymbol{x}$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} cos\varphi & -sin\varphi & 0 \\ sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \triangleq \begin{bmatrix} cos\varphi & -sin\varphi & 0 \\ sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

In the projective space: $\boldsymbol{x}' = \boldsymbol{R}'\boldsymbol{x}$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & -b & 0 \\ b & a & 0 \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} ax - by \\ bx + ay \\ cw \end{bmatrix} \triangleq$$

$$\begin{bmatrix} (ax - by)/(cw) \\ (bx + ay)/(cw) \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x}{w}\frac{a}{c} - \frac{y}{w}\frac{b}{c} \\ \frac{x}{w}\frac{b}{c} + \frac{y}{w}\frac{a}{c} \\ 1 \end{bmatrix} = \begin{bmatrix} Xcos\varphi - Ysin\varphi \\ Xsin\varphi + Ycos\varphi \\ 1 \end{bmatrix}$$

as $c^2 = (a^2 + b^2)$ by definition, $\det(\boldsymbol{R}') = (a^2 + b^2)c = c^3$



## Eurographics 2013

**Scaling** by a factor $(S_x, S_y) = (\frac{s_x}{w_s}, \frac{s_y}{w_s}) \triangleq [s_x, s_y: w_s]^T$

$$\boldsymbol{x}' = \boldsymbol{S}\boldsymbol{x} \qquad \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\boldsymbol{x}' = \boldsymbol{S}'\boldsymbol{x} \qquad \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & w_s \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\det(\boldsymbol{S}') = s_x\, s_y\, w_s$$

It is necessary to note that the determinant of a transformation matrix $\boldsymbol{Q}$ , i.e. matrices $\boldsymbol{T}', \boldsymbol{R}', \boldsymbol{S}'$, is $\det(\boldsymbol{Q}) \neq 1$ in general, but as the formulation is in the projective space, there is no need to "normalize" transformations to $\det(\boldsymbol{Q}) = 1$ even for rotation.

It can be seen that if the parameters of a geometric transformation are given in the homogeneous coordinates, no division operation is needed at all.



## Eurographics 2013

**Transformation of lines and planes**

| | $E^2$ | $E^3$ |
|---|---|---|
| Dual problem | $\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2$<br>$\boldsymbol{x} = \boldsymbol{p}_1 \times \boldsymbol{p}_2$ | $\boldsymbol{\rho} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 \times \boldsymbol{x}_3$<br>$\boldsymbol{x} = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_3$ |

In graphical applications position of points are changed by an interaction, i.e. $\boldsymbol{x}' = \boldsymbol{T}\boldsymbol{x}$.

The question is how coefficients of a line, resp. a plane are changed without need to recompute them from the definition.

It can be proved that

$$\boldsymbol{p}' = (\boldsymbol{T}\boldsymbol{x}_1) \times (\boldsymbol{T}\boldsymbol{x}_2) = det(\boldsymbol{T})(\boldsymbol{T}^{-1})^T\boldsymbol{p} \triangleq (\boldsymbol{T}^{-1})^T\boldsymbol{p}$$

or

$$\boldsymbol{\rho}' = (\boldsymbol{T}\boldsymbol{x}_1) \times (\boldsymbol{T}\boldsymbol{x}_2) \times (\boldsymbol{T}\boldsymbol{x}_3) = det(\boldsymbol{T})(\boldsymbol{T}^{-1})^T\boldsymbol{\rho} \triangleq (\boldsymbol{T}^{-1})^T\boldsymbol{\rho}$$



## Eurographics 2013

**Transformation of lines and planes**

As the computation is made **in the projective space** we can write

$$\boldsymbol{p}' = (\boldsymbol{T}^{-1})^T\boldsymbol{p} = [a',\ b': c']^T \qquad \text{for lines in } E^2$$

or

$$\boldsymbol{\rho}' = (\boldsymbol{T}^{-1})^T\boldsymbol{\rho} = [a',\ b', c': d']^T \qquad \text{for planes in } E^3$$

***THIS SIMPLIFIES COMPUTATIONS***

Transformation matrices for lines, resp. for planes are **DIFFERENT** from transformations for points! Note that a normal vector of a line is actually a co-vector, i.e. an oriented "surface".

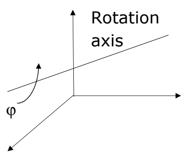


## Eurographics 2013

**Transformation of lines and planes**

Transformation about a general axis in $E^3$ / $P^3$

$$\boldsymbol{x}(t) = \boldsymbol{x}_A + \boldsymbol{s}t$$




Usually used transformation ($\boldsymbol{T}$ is translation):

$$\boldsymbol{Q} = \boldsymbol{T}^{-1}\boldsymbol{R}_{zx}^{-1}\boldsymbol{R}_{yz}^{-1}\boldsymbol{R}(\varphi)\boldsymbol{R}_{zx}\boldsymbol{R}_{xy}\boldsymbol{T}$$

$$\boldsymbol{R}_{yz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c}{\sqrt{b^2 + c^2}} & \frac{-b}{\sqrt{b^2 + c^2}} & 0 \\ 0 & \frac{b}{\sqrt{b^2 + c^2}} & \frac{c}{\sqrt{b^2 + c^2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\boldsymbol{s} = [a, b, c]^T$ is an axis directional vector. This is unstable if $\sqrt{b^2 + c^2} \to 0$ and not precise if $b^2 \gg c^2$ or vice versa.

That is generally computationally complex and unstable as *a user has to select which axis is to be used for a rotation*

# Eurographics 2013

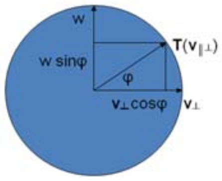
**Transformation of lines and planes**

Transformation about an axis $\boldsymbol{n}$
in the Euclidean space E$^3$

$$\boldsymbol{X} = \boldsymbol{X}\,cos\varphi + (1 - cos\varphi)(\,\boldsymbol{n}^T\boldsymbol{X}).\,\boldsymbol{n} + (\boldsymbol{n}\times\boldsymbol{X})sin\varphi$$

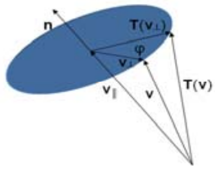
$$\boldsymbol{Q} = \boldsymbol{I}cos\varphi + (1 - cos\varphi)(\boldsymbol{n}\otimes\boldsymbol{n}) + \boldsymbol{W}sin\varphi$$

where: $\boldsymbol{n}\otimes\boldsymbol{n} = \boldsymbol{n}\,.\,\boldsymbol{n}^T$ is a matrix.
In the Euclidean space E$^3$ the vector $\boldsymbol{n}$ has to be normalized




The matrix $\boldsymbol{W}$ is defined as: $\boldsymbol{W}\boldsymbol{v} = \boldsymbol{w}\times\boldsymbol{v}$

$$\boldsymbol{W} = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & w_x & 0 \end{bmatrix} \text{ in our case}$$






# Eurographics 2013

**Projection**

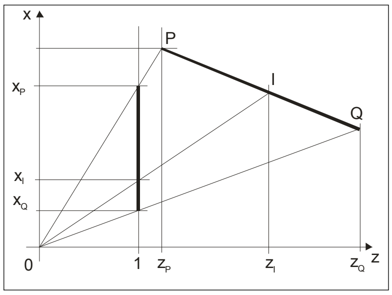
Standard projection

$$\begin{bmatrix} x_I \\ y_I \\ w_I \end{bmatrix} = \boldsymbol{T}_{projection}\begin{bmatrix} x_I \\ y_I \\ 1 \end{bmatrix}$$

How to determine $z_I$ coordinate?

$x_I = (1-\lambda)x_P + \lambda x_Q$

$y_I = (1-\lambda)y_P + \lambda y_Q$

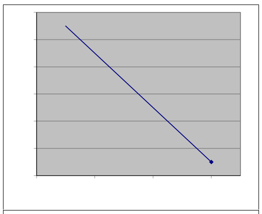



$$\frac{1}{z_I} = (1-\lambda)\frac{1}{z_P} + \lambda\frac{1}{z_Q}$$

⇨ “Reverse” depth computation, e.g. for correct intensity computation if perspective projection is used.



# Eurographics 2013



# Eurographics 2013

**Computation in Projective Space**

Intersection line – plane (Cyrus-Beck clipping algorithm)

$\boldsymbol{x}(t) = \boldsymbol{x}_0 + (\boldsymbol{x}_1 - \boldsymbol{x}_0)\,t = \boldsymbol{x}_0 + \boldsymbol{s}\,t$ Computation in projective space !

$\boldsymbol{s} = \boldsymbol{x}_1 - \boldsymbol{x}_0 = [x_1 - x_0, y_1 - y_0, z_1 - z_0 : w_1 - w_0]^T \quad t \in (-\infty, \infty)$

$\boldsymbol{a}^T\boldsymbol{x} = 0 \quad ax + by + cz + dw = 0 \quad \boldsymbol{a} = [a, b, c : d]^T$

Euclidean solution: $t = -\dfrac{\boldsymbol{a}^T\boldsymbol{x}_0}{\boldsymbol{a}^T\boldsymbol{s}}$

Projective solution: $\tau = -\boldsymbol{a}^T\boldsymbol{x}_0$ and $\tau_w = \boldsymbol{a}^T\boldsymbol{s}$ $\quad \boldsymbol{t} = [\tau, \tau_w]^T$

**if** $\tau_w < 0$ **then** $\boldsymbol{t} := -\boldsymbol{t}$

Test: **if** $t < t_{min}$ **then** …modification:

**if** $\tau * \tau_{min_w} > \tau_w * \tau_{min}$ **then** …..

- An intersection of a plane with a line in $E^2$ / $E^3$ can be computed efficiently, **no division operation**
- Comparison operations must be modified!!!
- => Cyrus-Beck line clipping algorithm 10-25% faster



# Eurographics 2013

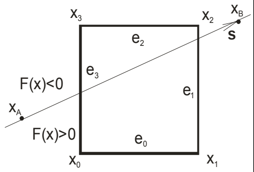
**Line Clipping – convex polygon**

```
procedure CLIP_Line ( x_A , x_B );
/* x_A=[x_A,y_A:w_A]^T x_B=[x_B,y_B:w_B]^T */
begin /* p=[a,b:c]^T given - NO STEP 1 */
{1}p := x_A x x_B;     /* p: ax+by+c = 0 */
{2}for k:=0 to N-1 do /*x_k=[x_k,y_k,w_k]^T*/
{3}    if p^T x_k ≥ 0 then c_k:=1 else c_k:=0;
{4}if c=[0…0]^T or c=[1…1]^T then EXIT;
{5}i:= TAB1[c];    j:= TAB2[c];
{6}x_A := p x e_i ;    x_B := p x e_j ;
{7}DRAW (x_A; x_B )    * e_i – i-th edge */
end /* CLIP_Line */
/* c identifies an edge intersected */
```

**TOO COMPLEX?**

Line clipping algorithms in $E^2$

- Cohen-Sutherland
- Liang-Barsky
- Hodgman
- Skala – modification of Clip_L for line segments




**NO SIMPLE, ROBUST and FAST**

- Skala,V.: A new approach to line and line segment clipping in homogeneous coordinates, The Visual Computer, SpringerVol.21, No.11, pp.905-914, 2005



# Eurographics 2013

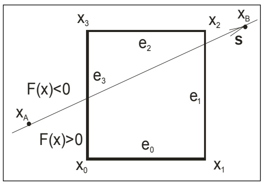
For a rectangular normalized window

$\boldsymbol{x}_A := \boldsymbol{p} \times \boldsymbol{e}_i$

For the edge $y = -1$, i.e. $y + 1 = 0$

$$\boldsymbol{x}_A = [x_A, y_A : w_A]^T = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ a & b & c \\ 0 & 1 & 1 \end{vmatrix} = [b - c, -a : a]$$

$$\triangleq \left[\frac{b-c}{a}, \frac{-a}{a} : 1\right]^T = \left[\frac{b-c}{a}, -1 : 1\right]^T$$




Actually expression for $y_A$ , resp. for $x_A$ is given by the window edge.

**No multiplication or division operations**

A simple modification if a line is given (in the Euclidean or projective space) as $\boldsymbol{x}(t) = \boldsymbol{x}_A + \boldsymbol{s}t$

Simple modification for non-convex polygon but it requires intersections sorting => $O(M\ lgM)$



# Eurographics 2013

/* Additional code for Line Segment clipping colored */

```
function CODE (x); /* Cohen Sutherland window coding */
begin   c:= [0000];     /* land / lor – bitwise  operations and / or */
    if x<x_min then c:=[1000] else if x>x_max then c:= [0100];
    if y<y_min then c:=c lor [1001] else if y>y_max then c:=c lor [0010];
    CODE := c
end /*CODE */;
procedure Clip_LineSegment (x_A , x_B); /* short x long line segments */
begin  c_A := CODE (x_A);    c_B := CODE (x_B);
    if (c_A lor c_B) = 0 then { output (x_A; x_B ); EXIT }      /*  inside */
    if (c_A land c_B) ≠ 0 then EXIT;                          /* outside */
    p := x_A x x_B;                  /* ax+by+c = 0; p = [a,b:c]^T */
    for k:=0 to 3 do                 /* x_k=[x_k,y_k: w_k]^T   c=[ c_3, c_2, c_1, c_0 ]^T */
        if p^T x_k ≥ 0 then c_k:=1 else c_k:=0;
    if c = [0000]^T or c = [1111]^T then EXIT;
    i:= TAB1[c];    j:= TAB2[c];
    /* original code x_A :=  p x e_i ;  x_B :=  p x e_j ; DRAW (x_A; x_B ) */
```



# Eurographics 2013

```
/* additional code for Line Segment clipping */
if  c_A ≠ 0 and c_B ≠ 0
then { x_A := p x e_i ;  x_B:= p x e_j } /* two intersections */
else /* only one intersection point */
   if  c_A = 0 then /* x_B  is outside */
{ if c_B land MASK[c] ≠ 0
          then  x_B :=  p x e_i
          else   x_B :=  p x e_j
}
  else if  c_B =0 then /* x_A is outside */
     { if c_A land MASK[c] ≠ 0
          then  x_A :=  p x e_i
          else   x_A :=  p x e_j
     };
output (x_A , x_B )
end /* Clip_LineSegment */
```

| c | $c_3$ | $c_2$ | $c_1$ | $c_0$ | TAB1 | TAB2 | MASK |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | None | None | None |
| 1 | 0 | 0 | 0 | 1 | 0 | 3 | 0100 |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0100 |
| 3 | 0 | 0 | 1 | 1 | 1 | 3 | 0010 |
| 4 | 0 | 1 | 0 | 0 | 1 | 2 | 0010 |
| 5 | 0 | 1 | 0 | 1 | N/A | N/A | N / A |
| 6 | 0 | 1 | 1 | 0 | 0 | 2 | 0100 |
| 7 | 0 | 1 | 1 | 1 | 2 | 3 | 1000 |
| 8 | 1 | 0 | 0 | 0 | 2 | 3 | 1000 |
| 9 | 1 | 0 | 0 | 1 | 0 | 2 | 0100 |
| 10 | 1 | 0 | 1 | 0 | N/A | N/A | N / A |
| 11 | 1 | 0 | 1 | 1 | 1 | 2 | 0010 |
| 12 | 1 | 1 | 0 | 0 | 1 | 3 | 0010 |
| 13 | 1 | 1 | 0 | 1 | 0 | 1 | 0100 |
| 14 | 1 | 1 | 1 | 0 | 0 | 3 | 0100 |
| 15 | 1 | 1 | 1 | 1 | None | None | None |

Algorithm can be extended to convex polygon clipping & modified for parametric lines as well

For a convex n-sided polygon the table can be generated synthetically.



# Eurographics 2013

**Computation in Projective Space -** Barycentric coordinates

Let us consider a triangle with vertices $X_1,\ X_2,\ X_{31}$

A position of any point $X \in E^2$ can be expressed as

$$a_1X_1 + a_2X_2 + a_3X_3 = X$$

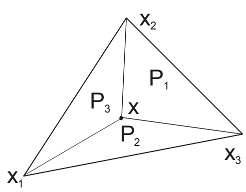
$$a_1Y_1 + a_2Y_2 + a_3Y_3 = Y$$

Additional condition:

$$a_1 + a_2 + a_3 = 1$$

$$0 \le a_i \le 1 \qquad i = 1, \dots, 3$$

$$a_i = \frac{P_i}{P}$$




Linear system has to be solved

If points $x_i$ are given as $\boldsymbol{x}_i = [x_i, y_i, z_i : w_i]^T$ and $w_i \neq 1$ then $x_i$ have to be "normalized" to $w_i = 1$, i.e. 4 * 3 = 12 division operations are used.

# Eurographics 2013

**Computation in Projective Space**

$$b_1X_1 + b_2X_2 + b_3X_3 + b_4X = 0$$
$$b_1Y_1 + b_2Y_2 + b_3Y_3 + b_4Y = 0$$
$$b_1 + b_2 + b_3 + b_4 = 1 \qquad b_i = -a_i b_4 \qquad b_4 \neq 0 \qquad i = 1, \dots, 3$$

Rewriting

$$\begin{bmatrix} X_1 & X_2 & X_3 & X \\ Y_1 & Y_2 & Y_3 & Y \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \mathbf{0} \qquad \begin{aligned} \boldsymbol{b} &= [b_1, b_2, b_3, b_4]^T \\ \boldsymbol{\xi} &= [X_1, X_2, X_3, X]^T \\ \boldsymbol{\eta} &= [Y_1, Y_2, Y_3, Y]^T \\ \boldsymbol{\omega} &= [1, 1, 1, 1]^T \end{aligned}$$

A solution of the linear system of equations (LSE) is equivalent to generalized cross product:

$$\boldsymbol{b} = \boldsymbol{\xi} \times \boldsymbol{\eta} \times \boldsymbol{\omega}$$



# Eurographics 2013

**Computation in Projective Space**

For $w_i \neq 0$ each column can be multiplied by the relevant $w_i$

$$\begin{bmatrix} w_1X_1 & w_2X_2 & w_3X_3 & wX \\ w_1Y_1 & w_2Y_2 & w_3Y_3 & wY \\ w_1 & w_2 & w_3 & w \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \mathbf{0} \qquad \begin{bmatrix} x_1 & x_2 & x_3 & x \\ y_1 & y_2 & y_3 & y \\ w_1 & w_2 & w_3 & w \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \mathbf{0}$$

where again

$$\boldsymbol{b} = [b_1, b_2, b_3, b_4]^T \quad \boldsymbol{\xi} = [x_1, x_2, x_3, x]^T \quad \boldsymbol{\eta} = [y_1, y_2, y_3, y]^T \quad \boldsymbol{\omega} = [w_1, w_2, w_3, w]^T$$

Barycentric coordinates (Euclidean):

$$0 \leq (-b_1 : w_2, w_3, w) \leq 1 \quad 0 \leq (-b_2 : w_3, w_1, w) \leq 1 \quad 0 \leq (-b_3 : w_1, w_2, w) \leq 1$$

=> Tests – point in triangle, point in tetrahedron -
**all in homogeneous coordinates**

=> **new entities**: *projective scalar*, *projective vector*

(Skala,V.: Barycentric coordinates computation in homogeneous coordinates, Computers&Graphics, 2008)



# Eurographics 2013

**Computation in Projective Space**

Area of a triangle — Volume of a tetrahedron

$$P = \frac{1}{2}\boldsymbol{x}_1^T.(\boldsymbol{x}_2 \times \boldsymbol{x}_3)/(w_1w_2w_3) \qquad V = \frac{1}{6}\boldsymbol{x}_1^T.(\boldsymbol{x}_2 \times \boldsymbol{x}_3 \times \boldsymbol{x}_4)/(w_1w_2w_3w_4)$$

As the principle of duality is valid, one could ask: *What is a "dual" value $G$ to a computation of the area $P$ if the triangle is given by three lines in the "normalized" form, e.g.* $\boldsymbol{a}_1^T.(\boldsymbol{a}_2 \times \boldsymbol{a}_3)$ *instead of three points?*

$$G = \boldsymbol{a}_1^T.(\boldsymbol{a}_2 \times \boldsymbol{a}_3) \triangleq \begin{vmatrix} cos\alpha_1 & cos\alpha_2 & cos\alpha_3 \\ sin\alpha_1 & sin\alpha_2 & sin\alpha_3 \\ d_1 & d_2 & d_3 \end{vmatrix} = \begin{vmatrix} 1 & cos\alpha_2 & cos\alpha_3 \\ 0 & sin\alpha_2 & sin\alpha_3 \\ 0 & 0 & d_3 \end{vmatrix} = d_3 sin\alpha_2$$
$$= d_3.a/(2R) = P/R$$

It can be seen that $G = d_3 sin\alpha_2 = P/R$ , where: $a$ is the length of the line segment on $\boldsymbol{a}_3$ and $R$ is a radius of the circumscribing circle.

=> value $G$ can be used as criterion for a quality triangular meshes.

- Skala,V.: Geometric Computation, Duality and Projective Space, IW-LGK workshop proceedings, ISBN 978-3-86780-244-4, pp.105-111, Dresden University of Technology, 2011



# Eurographics 2013

**Computation in Projective Space**

**Line in E$^3$ as Two Plane Intersection**

Standard formula in the Euclidean space

$$\boldsymbol{\rho}_1 = [a_1, b_1, c_1 : d_1]^T = [\boldsymbol{n}_1^T : d_1]^T \qquad \boldsymbol{\rho}_2 = [a_2, b_2, c_2 : d_2]^T = [\boldsymbol{n}_2^T : d_2]^T$$

Line given as an intersection of two planes

$$\boldsymbol{s} = \boldsymbol{n}_1 \times \boldsymbol{n}_2 \equiv [a_3, b_3, c_3]^T \qquad \boldsymbol{x}(t) = \boldsymbol{x}_0 + \boldsymbol{s}t$$

$$x_0 = \frac{d_2 \begin{vmatrix} b_1 & c_1 \\ b_3 & c_3 \end{vmatrix} - d_1 \begin{vmatrix} b_2 & c_2 \\ b_3 & c_3 \end{vmatrix}}{DET} \qquad y_0 = \frac{d_2 \begin{vmatrix} a_3 & c_3 \\ a_1 & c_1 \end{vmatrix} - d_1 \begin{vmatrix} a_3 & c_3 \\ a_2 & c_2 \end{vmatrix}}{DET}$$

$$z_0 = \frac{d_2 \begin{vmatrix} a_1 & b_1 \\ a_3 & b_3 \end{vmatrix} - d_1 \begin{vmatrix} a_2 & b_2 \\ a_3 & b_3 \end{vmatrix}}{DET} \qquad DET = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}$$

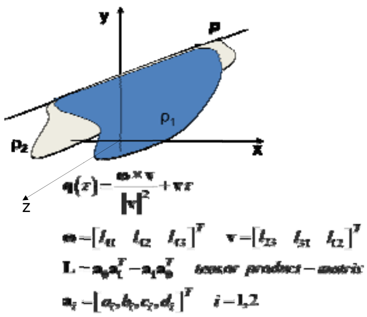
**The formula is quite "horrible" one and for students not acceptable as it is too complex and they do not see from the formula comes from.**



# Eurographics 2013

**Computation in Projective Space**

**Line in E3 as Two Plane Intersection**




- Standard formula in the Euclidean space
- Plücker's coordinates
  a line is given by two points. DUALITY – a point is dual to
  a plane and vice versa =>
  an intersection of two planes can be computed as a dual problem
  But computationally **expensive** computation
- Projective formulation and simple computation



# Eurographics 2013

**Computation in Projective Space**

**Line in E3 as Two Plane Intersection**

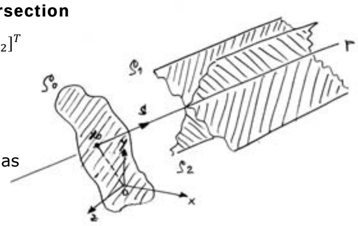
$\boldsymbol{\rho}_1 = [a_1, b_1, c_1 : d_1]^T$ $\boldsymbol{\rho}_2 = [a_2, b_2, c_2 : d_2]^T$

normal vectors are

$\boldsymbol{n}_1 = [a_1, b_1, c_1]^T$ $\boldsymbol{n}_2 = [a_2, b_2, c_2]^T$

Directional vector of a line
of two planes $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_1$ is given as

$\boldsymbol{s} = \boldsymbol{n}_1 \times \boldsymbol{n}_2$



"starting" point $\boldsymbol{x}_0$ **???**

A plane $\boldsymbol{\rho}_0$ passing the origin with a normal vector $\boldsymbol{s}$ , $\boldsymbol{\rho}_0 = [a_0, b_0, c_0 : 0]^T$

The point $\boldsymbol{x}_0$ is defined as $\boldsymbol{x}_0 = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_0$

**Simple formula for matrix-vector architectures like GPU and parallel processing**. *Compare the standard and projective formulas*

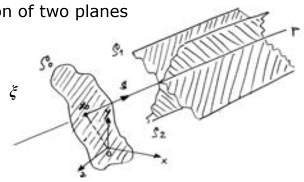
# Eurographics 2013

**Computation in Projective Space – the nearest point**

Find the nearest point on an intersection of two planes to the given point $\xi$

Simple solution:

- Translate planes $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$ so the $\xi$ is in the origin
- Compute intersection of two planes i.s. $\boldsymbol{x}_0$ and $\boldsymbol{s}$
- Translate $\boldsymbol{x}_0$ using $\boldsymbol{T}^{-1}$



Known solution using Lagrange multipliers

$$\begin{bmatrix} 2 & 0 & 0 & n_{1x} & n_{2x} \\ 0 & 2 & 0 & n_{1y} & n_{2y} \\ 0 & 0 & 2 & n_{1z} & n_{2z} \\ n_{1x} & n_{1y} & n_{1z} & 0 & 0 \\ n_{2x} & n_{2y} & n_{2z} & 0 & 0 \end{bmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} 2p_{0x} \\ 2p_{0y} \\ 2p_{0z} \\ \vec{p}_1 \cdot \vec{n}_1 \\ \vec{p}_2 \cdot \vec{n}_2 \end{pmatrix}$$

Krumm,J.: Intersection of Two Planes, Microsoft Research

**Again – an elegant solution, simple formula supporting matrix-vector architectures like GPU and parallel processing**

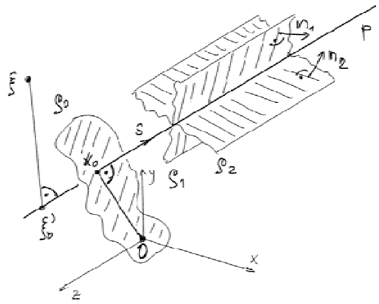
*Solution DETAILS next*



# Eurographics 2013

**The closest point to an intersection of two planes**

In some applications we need to find a closest point on a line given as an intersection of two planes. We want to find a point $\boldsymbol{\xi}_0'$, the closest point to the given point $\boldsymbol{\xi}$, which lies on an intersection of two planes



$\boldsymbol{\rho}_1 \triangleq [\boldsymbol{n}_1^T : d_1]^T$ and $\boldsymbol{\rho}_2 \triangleq [\boldsymbol{n}_2^T : d_2]^T$

This problem was recently solved by using Lagrange multipliers and an optimization approach leading to a solution of a system of linear equations with 5 equations.



# Eurographics 2013

**Solution in the projective space**

1. Translate the given point $\boldsymbol{\xi} = [\xi_x, \xi_y, \xi_z : 1]^T$ to the origin – matrix $\boldsymbol{Q}$
2. Compute parameters of the given planes $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$ after the transformation as $\boldsymbol{\rho}_1' = \boldsymbol{Q}^{-T}\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2' = \boldsymbol{Q}^{-T}\boldsymbol{\rho}_2$,
3. Compute the intersection of those two planes $\boldsymbol{\rho}_1'$ and $\boldsymbol{\rho}_2'$
4. Transform the point $\boldsymbol{\xi}_0$ to the original coordinate system using transformation

$$\boldsymbol{n}_0 = \boldsymbol{n}_1 \times \boldsymbol{n}_2 \qquad \boldsymbol{\rho}_0 \triangleq [\boldsymbol{n}_0^T : 0]^T \qquad \boldsymbol{\xi}_0 = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_0 \qquad \boldsymbol{\xi}_0' = \boldsymbol{Q}^{-1}\,\boldsymbol{\xi}_0$$

$$\boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 & -\xi_x \\ 0 & 1 & 0 & -\xi_y \\ 0 & 0 & 1 & -\xi_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{Q}^{-T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \xi_x & \xi_y & \xi_z & 1 \end{bmatrix} \qquad \boldsymbol{Q}^{-1} = \begin{bmatrix} 1 & 0 & 0 & \xi_x \\ 0 & 1 & 0 & \xi_y \\ 0 & 0 & 1 & \xi_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It is simple, easy to implement on GPU.



# Eurographics 2013

**Curves and surfaces**

Rational Bézier curve – Euclidean $\boldsymbol{X} = [X, Y, Z]^T$

$$\boldsymbol{X}(t) = \frac{\sum_{i=0}^{n} B_i^n(t) w_i q_i}{\sum_{i=0}^{n} B_i^n(t) w_i} \qquad 0 \le t \le 1 \qquad B_i^n \binom{n}{i} (1-t)^{n-1} t^i$$

1st derivative $\qquad \left[\frac{a}{b}\right]' = \frac{a'b - ab'}{b^2} \qquad$ quite complicated

Projective $\boldsymbol{x} = [x, y, z : w]^T$

$$\boldsymbol{x}(t) = \sum_{i=0}^{n} B_i^n(t) q_i \qquad \boldsymbol{x}'(t) = \sum_{i=0}^{n} B_i^n(t) q_i$$

How simple !



# Eurographics 2013

**Computation in Projective Space**

**Disadvantages**

- Careful handling with formula as the projective space
- "Oriented" projective space is to be used, i.e. $w \ge 0$; HW could support it
- Exponents of homogeneous vectors can overflow
  - exponents should be normalized; HW could support it unfortunately not supported by the current hardware
  - P_Lib – library for computation in the projective space - uses SW solution for normalization on GPU (C# and C++)



# Eurographics 2013

**Computation in Projective Space**

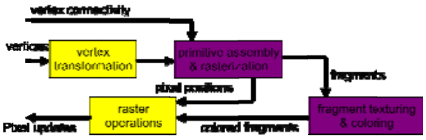
**Advantages**

- "Infinity" is well represented
- No division operation is needed, a division operation can be hidden to the homogeneous coordinate
- Many mathematical formula are simpler and elegant
- One code sequence solve primary and dual problems
- Supports matrix–vector operations in hardware – like GPU etc.
- Numerical computation can be faster
- More robust and stable solutions can be achieved
- System of linear equations can be solved directly without division operation, if exponent normalization is provided

## Eurographics 2013

**Implementation aspects and GPU**

- GPU (Graphical Processing Unit) -optimized for matrix-vector, vector-vector operation – especially for $[x,y,z:w]^T$
- Native arithmetic operations with homogeneous coordinates – without exponent "normalization"
- Programmable HW – parallel processing






## Eurographics 2013

**Implementation aspects and GPU**

4D cross product can be implemented in Cg/HLSL on GPU (not optimal implementation) as:

```
float4 cross_4D(float4 x1, float4 x2, float4 x3)
{ float4 a; # simple formula #
   a.x=dot(x1.yzw, cross(x2.yzw, x3.yzw));
   a.y=-dot(x1.xzw, cross(x2.xzw, x3.xzw));
   a.z=dot(x1.xyw, cross(x2.xyw, x3.xyw));
   a.w=-dot(x1.xyz, cross(x2.xyz, x3.xyz));
   return a;
}
# more compact formula available #
```



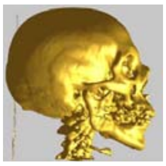
## Eurographics 2013

**Appendix**

**Data processing** - main field in computer science

Data processing itself can be split to two main areas:

- **processing of textual data**
  limited interval of values, unlimited dimensionality
  [a char as one dimension - Methionylthreonylthreonylglutaminylarginyl..isoleucine 189,819 chars]
  No interpolation is defined
- **processing of numerical data**
  unlimited interval of values,
  limited dimensionality – usually 2 or 3
  Interpolation **can be used**



| | Textual | Graphical |
|---|---|---|
| Dim | ∞ | 2, 3 |
| Interval | 0-255(ASCII) | (-∞, ∞) |



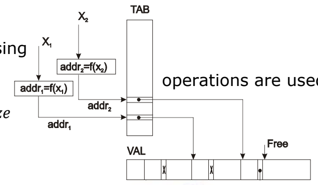
## Eurographics 2013

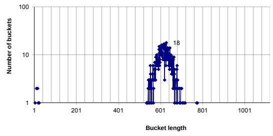
**Hash functions**

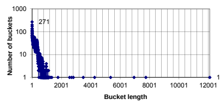
- usually used for textual data processing
- prime numbers and modulo operations are used

Usual form $Addr = \lfloor 3x + 5y + 7z \rfloor \mathbf{mod}\ size$

multiplication int * float needed



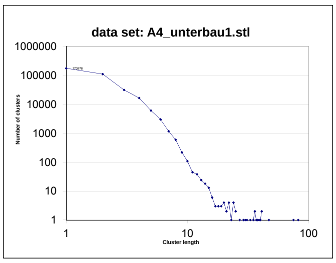



## Eurographics 2013

If the hash function is constructed as

$$Addr = \lfloor C(\alpha x + \beta y + \gamma z) \rfloor \mathbf{mod}\ m$$

where $\alpha, \beta, \gamma$ are "irrational" numbers and $m = 2^k$ better distribution is obtained => much faster processing.




$C$ is a "magic" constant which maps the expression for $< x_{min}, x_{max} >$ to $(0, maxint)$

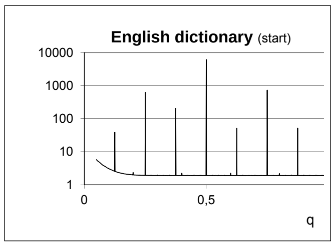
$$Addr = \lfloor C(\alpha x + \beta y + \gamma z) \rfloor \mathbf{and}\ (2^k - 1)$$

- Hradek,J., Skala,V.: Hash Function and Triangular Mesh Reconstruction, Vol.29, No.6., pp.741-751, Computers&Geosciences, Pergamon Press, ISSN 0098-3004, 2003



## Eurographics 2013

**Textual processing**




The has function is constructed as

$$h(\boldsymbol{x}) = \left( C * \sum_{i=1}^{L} q^i x_i \right) \boldsymbol{mod}\ m$$

$q$ „irrational" $0 < q < 1$

$m = 2^k - 1$

Both geometrical and textual hash function design have the same approach coefficients are "irrational" and no division operation is needed.

Some differences for Czech, Hebrew, English, German, Arabic, … languages and "chemical" words.

## Eurographics 2013

**Summary and conclusion**

We have got within this course an understanding of:

- projective representation use for geometric transformations with points, lines and planes
- principle of duality and typical examples of dual problems, influence to computational complexity
- intersection computation of two planes in E3, dual Plücker coordinates and simple projective solution
- geometric problems solution with additional constrains
- intersection computations and interpolation algorithms directly in the projective space
- barycentric coordinates computation on GPU
- avoiding or postponing division operations in computations

Projective space representation supports matrix-vector architectures like GPU – faster, robust and easy to implement algorithms achieved



## Eurographics 2013

**References**


- Skala,V.: Barycentric Coordinates Computation in Homogeneous Coordinates, Computers & Graphics, Elsevier, ISSN 0097-8493, Vol. 32, No.1, pp.120-127, 2008
- Skala,V.: Intersection Computation in Projective Space using Homogeneous Coordinates, Int. Journal of Image and Graphics, ISSN 0219-4678, Vol.7, No.4, pp.615-628, 2008
- Skala,V.: Length, Area and Volume Computation in Homogeneous Coordinates, Int. Journal of Image and Graphics, Vol.6., No.4, pp.625-639, ISSN 0219-4678, 2006
- Skala,V., Kaiser,J., Ondracka,V.: Library for Computation in the Projective Space, 6th Int.Conf. Aplimat, Bratislava, ISBN 978-969562-4-1, pp. 125-130, 2007
- Skala,V.: GPU Computation in Projective Space Using Homogeneous Coordinates , Game Programming GEMS 6 (Ed.Dickheiser,M.), pp.137-147, Charles River Media, 2006
- Skala,V.: A new approach to line and line segment clipping in homogeneous coordinates, The Visual Computer, Vol.21, No.11, pp.905-914, Springer Verlag, 2005
- Skala,V.: Geometric Computation, Duality and Projective Space, IW-LGK workshop proceedings, ISBN 978-3-86780-244-4, pp.105-111, Dresden University of Technology, 2011

Generally: "***Publications with on-line DRAFTs***" via **http://www.VaclavSkala.eu**


**References related**


- Agoston,M.K.: Computer Graphics and Geometric Modeling - Mathematics, Springer, 2005
- Ammeral,L: Programming Principles in Computer Graphics, John Wiley, 1986
- Miller,J.R., The Mathematics of Graphical Transformations: Vector Geometric and Coordinate-Based Approaches, DesignLab, 1997
- Yamaguchi,F.:Computer-Aided Geometric Design: A Totally Four-Dimensional Approach, Springer, 2002




## Eurographics 2013

## ?? Questions ??


**Contact: Vaclav Skala**
c/o University of West Bohemia
CZ 306 14 Plzen, Czech Republic
**http://www.VaclavSkala.eu** skala@kiv.zcu.cz subj. EG 2013


**INVITATION**
**WSCG conference on Computer Graphics, Visualizaction and Computer Vision 2013**
**since 1992**

**http://www.wscg.eu** **http://www.wscg.cz**
**June 24-27, 2013**

Supported by the Ministry of Education of the Czech Republic, projects No. ME10060, LH12181, LG13047