Mathematical Foundations for Computer Graphics, Computer Vision and Computation in Projective Space

Vaclav Skala

skala@kiv.zcu.cz   subj. 3DTV – Tutorial

Center of Computer Graphics and Visualization   http://herakles.zcu.cz
Department of Computer Science and Engineering   http://www.kiv.zcu.cz
Faculty of Applied Sciences   http://www.fav.zcu.cz
University of West Bohemia   http://www.zcu.cz

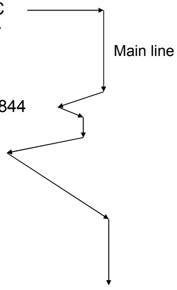
Plzen, Czech Republic

"Real science" in XXI century

Contents

History of Geometry

- Euclid - synthetic geometry 300 BC
- Descartes - analytic geometry 1637
- Gauss – complex algebra 1798
- Hamilton – quaternions 1843
- Grassmann – Grasmann Algebra 1844
- Cayley – Matrix Algebra 1854
- Clifford – Clifford algebra 1878
- Gibbs – vector calculus 1881
- Sylvester – determinants 1878
- Ricci – tensor calculus 1890
- Cartan – differential forms 1908
- Dirac, Pauli – spin algebra 1928
- Hestenes – Space-time algebra 1966 ➔ Geometry Algebra 1984

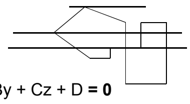


Geometry & Computers

- 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

Geometry & Computers

- 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



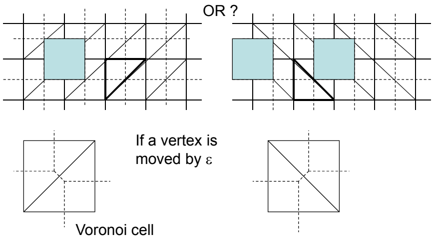
Typical problem

```
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

Geometry & Computers

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



Geometry & Computers

- Precision issues – basic rules
  - [X] + [Y] = [Xmin+Ymin, Xmax+Ymax]
  - [X] - [Y] = [Xmin-Ymax, Xmax-Ymin]
  - [X] * [Y] = [min{Xmin*Ymin, Xmin*Ymax, Xmax*Ymin, Xmax*Ymax}, max{Xmin*Ymin, Xmin*Ymax, Xmax*Ymin, Xmax*Ymax}]
  - [X] / [Y] = (-∞,∞) if Y=0 ; [X] / [Y] = **[X] * [1/X, 1/Y] otherwise !!!!!**
  - √[X] = [√ Xmin, √ Xmax]
- Typical example of wrong computational result:

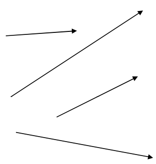
  $F(x,y)=333.75\, y^6+x^2\, (11x^2y^2-y^6-121y^4-2) +5.5y^8 +x/(2y)$ at [x,y]=[77617, 33096]

  single 6.33 10^29, double 1,172…, exact [-0.82739…± 1^{-34}] if interval arithmetic used

  [Lecrerc,A.: Should we be concerned about round-off error?]

Vectors and Points in Geometry

- A vector ***v*** has a magnitude (length) and direction.
- Normalized vectors have magnitude 1, e.g. ‖***v***‖=1
- Zero vector ***0*** has magnitude zero, no direction
- Vectors do not have a location!!
- **Vectors and points have only a similar representation** !!



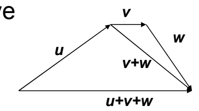
Vectors and Points in Geometry

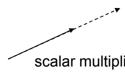
Algebraic rules - properties

- ***v***+***w***=***w***+***v*** commutative
- ***u***+(***v***+***w***)=(***u***+***v***)+***w*** associative
- ***u***+***0***=***u*** additive identity

For every ***v*** there is a vector –***v*** such that ***v***+(-***v***)=***0***

- (ab)***v***=a(b***v***) associative
- (a+b)***v***=a***v***+b***v*** distributive
- a(***v***+***w***)=a***v***+b***w*** distributive
- 1.***v***=***v*** multiplicative identity





scalar multiplication

Vectors and Points in Geometry

Vector representation in R^2

$\boldsymbol{x}_0+\boldsymbol{x}_1 = (x_0,y_0)+(x_1,y_1) = (x_0+x_1,y_0+y_1)$

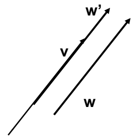
$a\, \boldsymbol{x}_0 = a(x_0,y_0) = (ax_0,ay_0)$

Direct generalization to R^3 and R^n

Linear combination

$\boldsymbol{v} = a_0\boldsymbol{v}_0+ a_1\boldsymbol{v}_1+\ldots+ a_{n-1}\boldsymbol{v}_{n-1}$

If $\boldsymbol{v}_i = a_0\boldsymbol{v}_0+ a_1\boldsymbol{v}_1+\ldots a_{i-1}\boldsymbol{v}_{i-1}+ a_{i+1}\boldsymbol{v}_{i+1}+\ldots+ a_{n-1}\boldsymbol{v}_{n-1}$ then $\boldsymbol{v}_i$ is linearly dependent – two linearly dependent vectors ***v*** and ***w*** are said to be parallel, e.g. ***w*** = a***v***



Vectors and Points in Geometry

Standard vector basis for R^3

$\boldsymbol{e}_0 = (1,0,0)$, $\boldsymbol{e}_1 = (0,1,0)$, $\boldsymbol{e}_2 = (0,0,1)$

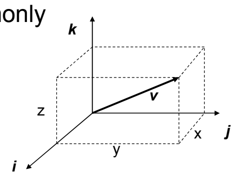
usually as {***i***, ***j***,***k***} , where

i = (1,0,0), ***j*** = (0,1,0), ***k*** = (0,0,1)

coefficients a_0, a_1, a_2, commonly represented as x, y, z

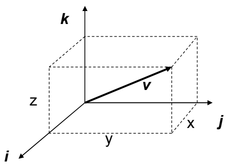
Vector ***v*** in R^3

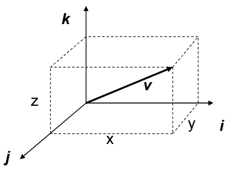
v = x***i*** + y***j***+z***k***



Vectors and Points in Geometry

Right handed R^3 | Left handed R^3

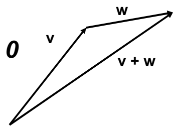




! **Be careful** – column x row vector notation – matrices of geometric transformations are transposed

Vectors and Points in Geometry

Vector length and norm



$\|\boldsymbol{v}\| \geq 0 \qquad \|\boldsymbol{v}\| = 0 \Leftrightarrow \boldsymbol{v} = \boldsymbol{0}$

$\|a\,\boldsymbol{v}\| = |a|\,\|\boldsymbol{v}\|$

$\|\boldsymbol{v} + \boldsymbol{w}\| \leq \|\boldsymbol{v}\| + \|\boldsymbol{w}\|$

$\|\boldsymbol{v}\| = (\sum v_i^2)^{1/2}$ Euclidean norm usually

Pythagorean theorem $x^2 + y^2 = d^2$

in R^3 $\quad x^2 + y^2 + z^2 = \|\boldsymbol{v}\|^2$

Vectors and Points in Geometry

Inner product

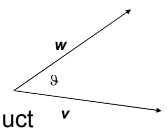
$\langle \boldsymbol{v}, \boldsymbol{w} \rangle = \langle \boldsymbol{w}, \boldsymbol{v} \rangle$ - symmetry

$\langle \boldsymbol{u}+\boldsymbol{v}, \boldsymbol{w} \rangle = \langle \boldsymbol{u}, \boldsymbol{w} \rangle + \langle \boldsymbol{v}, \boldsymbol{w} \rangle$ - additivity

$a \langle \boldsymbol{v}, \boldsymbol{w} \rangle = \langle a\,\boldsymbol{v}, \boldsymbol{w} \rangle = \langle \boldsymbol{v}, a\,\boldsymbol{w} \rangle$ - homogenity

$\langle \boldsymbol{v}, \boldsymbol{v} \rangle \geq 0$ - positivity

$\langle \boldsymbol{v}, \boldsymbol{v} \rangle = 0 \Leftrightarrow \boldsymbol{v} = \boldsymbol{0}$ - definiteness



Euclidean inner product – dot product

$\langle \boldsymbol{v}, \boldsymbol{w} \rangle = \boldsymbol{v} \cdot \boldsymbol{w} = \|\boldsymbol{v}\|\,\|\boldsymbol{w}\| \cos\vartheta$

Vectors and Points in Geometry

Cross (vector) product

$\boldsymbol{v} \times \boldsymbol{w} = -(\boldsymbol{w} \times \boldsymbol{v})$ – it is not commutative !!

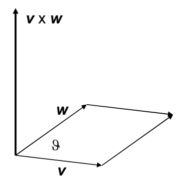
$\boldsymbol{u} \times (\boldsymbol{v} + \boldsymbol{w}) = \boldsymbol{u} \times \boldsymbol{v} + \boldsymbol{u} \times \boldsymbol{w}$

$(\boldsymbol{u} + \boldsymbol{v}) \times \boldsymbol{w} = \boldsymbol{u} \times \boldsymbol{w} + \boldsymbol{v} \times \boldsymbol{w}$

$a\,(\boldsymbol{u} \times \boldsymbol{v}) = (a\,\boldsymbol{u}) \times \boldsymbol{v} = \boldsymbol{u} \times (a\boldsymbol{v})$

$\boldsymbol{u} \times \boldsymbol{0} = \boldsymbol{0} \times \boldsymbol{u} = \boldsymbol{0}$

$\boldsymbol{v} \times \boldsymbol{v} = \boldsymbol{0}$



$\|\boldsymbol{v} \times \boldsymbol{w}\| = \|\boldsymbol{v}\|\,\|\boldsymbol{w}\| \sin\vartheta$

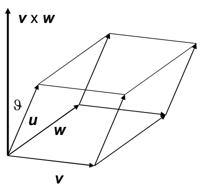
The length of the $\boldsymbol{v} \times \boldsymbol{w}$ equals to area of parallelogram

$\|\boldsymbol{v} \times \boldsymbol{w}\| = (\mathbf{v}.\mathbf{v})\ (\mathbf{w}.\mathbf{w} - (\mathbf{v}.\mathbf{w})^2$

Vectors and Points in Geometry

Cross (vector) product

$$\boldsymbol{v} \times \boldsymbol{w} = \det \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{vmatrix}$$



Vector triple product

$\boldsymbol{u} \times (\boldsymbol{v} \times \boldsymbol{w}) = (\mathbf{u}.\mathbf{w})\mathbf{v} - (\mathbf{u}.\mathbf{v})\mathbf{w}$

Scalar triple product – equals to the volume of a parallelopiped

$\boldsymbol{u} \cdot (\boldsymbol{v} \times \boldsymbol{w}) = \|\boldsymbol{u}\|\,\|\boldsymbol{v} \times \boldsymbol{w}\| \cos\vartheta$

Vectors and points in geometry

Algebraic rules:

Vector triple product

$\boldsymbol{u} \times (\boldsymbol{v} \times \boldsymbol{w}) = (\boldsymbol{u}.\boldsymbol{w})\boldsymbol{v} - (\boldsymbol{u}.\boldsymbol{v})\boldsymbol{w}$

$(\boldsymbol{u} \times \boldsymbol{v}) \times \boldsymbol{w} = (\boldsymbol{u}.\boldsymbol{w})\boldsymbol{v} - (\boldsymbol{v}.\boldsymbol{w})\boldsymbol{u}$

$\boldsymbol{u}.(\boldsymbol{v} \times \boldsymbol{w}) = \boldsymbol{w}.(\boldsymbol{u} \times \boldsymbol{v}) = \boldsymbol{v}.(\boldsymbol{w} \times \boldsymbol{u})$

Outer (tensor) product – result is a ***MATRIX Q***

$\boldsymbol{Q} = \boldsymbol{u}^T\boldsymbol{v} \quad q_{ij} = u_i\,v_j$ - will be defined latter

Tip:

CPU and GPU optimized for vector/parallel comp.

Vectors and Points in Geometry

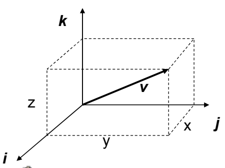
Vectors – movable, no fixed position

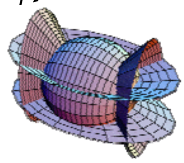
Points – no size, position fixed in the GIVEN coordinate system

Coordinate systems:

- Cartesian – right handed system is used
- Polar
- Spherical
- and many others

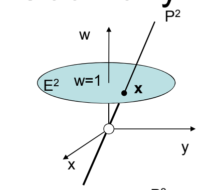
  e.g. Confocal Ellipsoidal Coordinates (http://mathworld.wolfram.com/ConfocalEllipsoidalCoordinates.html)

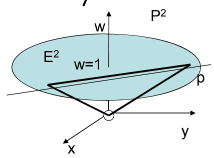




Vectors and Points in Geometry

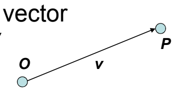
- Projective extension of Affine space
- A point $\boldsymbol{x}$ in E^2 is defined with coordinates $\boldsymbol{X} = (X,Y)$ or as a point $\boldsymbol{x} = [x,y,w]^T$ in homogeneous coordinates $X = x/w \quad Y = y/w, \quad w \neq 0$
- The point $\boldsymbol{x}$ is a line (without an origin) in the projective space P^2
- A point $\boldsymbol{x}$ in E^3 is defined with coordinates $\boldsymbol{X} = (X,Y,Z)$ or as a point $\boldsymbol{x} = [x,y,z,w]^T$, in homogeneous coordinates $X = x/w \quad Y = y/w, \ Z = z/w \ \ w \neq 0$
- A line in E^2 is a plane (without an origin) in the projective space P^3





Vectors and Points in Geometry

An ***affine space*** is formed by a vector space $\boldsymbol{V}$ and set of points $\boldsymbol{W}$



Let us define relations

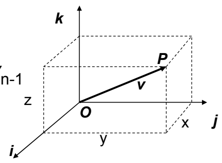
$\boldsymbol{v} = \boldsymbol{P} - \boldsymbol{O}$

$\boldsymbol{P} = \boldsymbol{O} + \boldsymbol{v}$

using n basis vectors of $\boldsymbol{V}$

$\boldsymbol{P} = \boldsymbol{O} + a_0\boldsymbol{v}_0 + a_1\boldsymbol{v}_1 + \ldots + a_{n-1}\boldsymbol{v}_{n-1}$

Coordinate Frame



Vectors and Points in Geometry

An ***affine combinations***

$\boldsymbol{P} = a_0\boldsymbol{P}_0 + a_1\boldsymbol{P}_1 + \ldots + a_{k-1}\boldsymbol{P}_{k-1}$

and $a_0 + a_1 + \ldots + a_{k-1} = 1$ or $a_0 = 1 - a_1 - \ldots - a_{k-1}$

Convex combination $0 \leq a_0 + a_1 + \ldots + a_{k-1} \leq 1$

$\boldsymbol{P} = \boldsymbol{P}_0 + a_1(\boldsymbol{P}_1 - \boldsymbol{P}_0) \ldots + a_{k-1}(\boldsymbol{P}_{k-1} - \boldsymbol{P}_0)$

if $\boldsymbol{v}_i = \boldsymbol{P}_i - \boldsymbol{P}_0$ then

$\boldsymbol{P} = \boldsymbol{P}_0 + a_1\boldsymbol{v}_1 \ldots + a_{k-1}\boldsymbol{v}_{k-1}$

Vectors and Points in Geometry

Vector representation $\boldsymbol{v} = (v_x, v_y, v_z, 0)$

Point representation $\boldsymbol{P} = (P_x, P_y, P_z, 1)$

Many libraries do not distinguish between points and vectors and treat them in the same manner

!! **BE CAREFUL** !!

$\boldsymbol{v} = \boldsymbol{P}_1 - \boldsymbol{P}_0 = (P_{x1}, P_{y1}, P_{z1}, 1) - (P_{x0}, P_{y0}, P_{z0}, 1) = (P_{x1} - P_{x0}, P_{y1} - P_{y0}, P_{z1} - P_{z0}, 0) = (v_x, v_y, v_z, 0)$

!!! Do not make it on CPU/GPU – result $(v_x, v_y, v_z, \varepsilon)$ that is a point $(v_x/\varepsilon, v_y/\varepsilon, v_z/\varepsilon)$ in E^3

Vectors and Points in Geometry

How to handle vectors?

$$\mathbf{v} = \mathbf{X}_1 - \mathbf{X}_0 = \left[\frac{x_1}{w_1} - \frac{x_0}{w_0}, \frac{y_1}{w_1} - \frac{y_0}{w_0}, \frac{z_1}{w_1} - \frac{z_0}{w_0}\right]^T =$$

$$= [\mathbf{x}_1 : w_1]^T - [\mathbf{x}_0 : w_0]^T = [w_0\mathbf{x}_1 - w_1\mathbf{x}_0 : w_0 w_1]^T$$

Other operations can be defined in a similar way

No problems with a stability of computation – we do not need to solve instability

$$[\mathbf{x}_1 :\cong 1]^T - [\mathbf{x}_0 :\cong 1]^T = [\mathbf{x}_1 - \mathbf{x}_0 :\cong 0]$$

e.g. $(X,Y) = (v_x/\varepsilon, v_y/\varepsilon)$ where $\varepsilon \to 0$

Vectors and Points in Geometry

Projective extensions

$$\mathbf{A.B} = \left[\frac{a_x}{w_a}\frac{b_x}{w_b}, \frac{a_y}{w_a}\frac{b_y}{w_b}\right] = \frac{1}{w_a w_b}\left[\mathbf{a} : w_a\right].\left[\mathbf{b} : w_b\right] = \frac{1}{w_a w_b}\mathbf{a.b}$$

$$\mathbf{A.B} = \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ A_x & A_y & A_z \\ B_x & B_y & B_y \end{bmatrix} = \frac{1}{w_a w_b}\begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_y \end{bmatrix} =$$

$$\frac{1}{w_a w_b}\left[\mathbf{a} : w_a\right]\times\left[\mathbf{b} : w_b\right] = \frac{1}{w_a w_b}\begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_y \end{bmatrix}$$

Details: PLib – Library for Computation in the Projective Space [8]

Vectors and Points in Geometry

Geometric product

$\mathbf{ab} = \mathbf{a.b} + \mathbf{a} \wedge \mathbf{b}$ $\quad$ $\mathbf{ba} = \mathbf{b.a} + \mathbf{b} \wedge \mathbf{a} = \mathbf{a.b} - \mathbf{a} \wedge \mathbf{b}$

It can be seen

$\mathbf{a.b} = \frac{1}{2}(\mathbf{ab} + \mathbf{ba})$ $\quad$ $\mathbf{a} \wedge \mathbf{b} = \frac{1}{2}(\mathbf{ab} - \mathbf{ba})$

Result is a scalar and bivector in E^n

In the case of E^3 the operator $\wedge$ is equivalent to $\times$

Extension to projective case is analogical

Vectors and Matrices

Vector – any vector in n-dimensional vector space ***V*** can be represented as

$$\boldsymbol{x} = x_0\boldsymbol{v}_0 + x_1\boldsymbol{v}_1 + \ldots\ldots + x_{n-1}\boldsymbol{v}_{n-1}$$

where $\{\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n-1}\}$ is a basis vector of ***V***

Vectors and Matrices

Matrix – square matrix n = m

$\boldsymbol{A} = [a_{ij}]$

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \ldots & a_{1,m} \\ : & & : \\ a_{n,m} & \ldots & a_{n,m} \end{bmatrix}$$

Special cases

m = 1 - column matrix – vector (used in the right handed coordinate system)

n = 1 - row matrix - vector

Vectors and Matrices

Algebraic rules

A + ***B*** = ***B*** + ***A***

A + (***B*** + ***C***) = (***A*** + ***B***) + ***C***

A + ***0*** = ***A***

A + (-***A***) = ***0***

a(***A*** + ***B***) = a***A*** + Ab

(a+b)***A*** = a***A*** + b***A***

1 . ***A*** = ***A***

Transpose of a matrix ***A*** is a matrix ***B*** = $\boldsymbol{A}^T$

where: $b_{ij} = a_{ij}$

Additional rules

$(\boldsymbol{A}^T)^T = \boldsymbol{A}$

$\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$

$(\boldsymbol{A}^{-1})^T = (\boldsymbol{A}^T)^{-1} = \boldsymbol{A}^{-T}$ (n x n)

$(a\boldsymbol{A})^T = a\boldsymbol{A}^T$

$(\boldsymbol{A}+\boldsymbol{B})^T = \boldsymbol{A}^T + \boldsymbol{B}^T$

Vectors and Matrices

Matrix multiplication

C = ***A B***

$$c_{ik} = \sum_{k=1}^{p} a_{ik} b_{kj}$$

$\boldsymbol{A} = [a_{ik}]$ $\quad$ i = 1,…,n, k = 1,…,p

$\boldsymbol{B} = [b_{kj}]$ $\quad$ k = 1,…,p, j = 1,…,m

$\boldsymbol{C} = [c_{ij}]$ $\quad$ i = 1,…,n, j = 1,…,m

NOTE!

A B ≠ B A

How to compute **C**?

	B
A	**C**

Vectors and Matrices

$\boldsymbol{A}(\boldsymbol{BC}) = (\boldsymbol{AB})\boldsymbol{C}$ a$(\boldsymbol{BC}) = ($a$\boldsymbol{B})\boldsymbol{C}$

$\boldsymbol{A}(\boldsymbol{B} + \boldsymbol{C}) = \boldsymbol{AB} + \boldsymbol{AC}$ $(\boldsymbol{A} + \boldsymbol{B})\boldsymbol{C} = \boldsymbol{AC} + \boldsymbol{BC}$

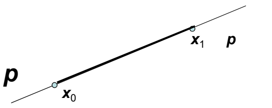
$(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$

Block matrices

$$\left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right]\left[\begin{array}{c|c} \mathbf{E} & \mathbf{F} \\ \hline \mathbf{G} & \mathbf{H} \end{array}\right] = \left[\begin{array}{c|c} \mathbf{AE} + \mathbf{BG} & \mathbf{AF} + \mathbf{BH} \\ \hline \mathbf{CE} + \mathbf{DG} & \mathbf{CF} + \mathbf{DH} \end{array}\right]$$

Lines and Line Segments

Lines in E^2
two points define a line ***p***



- implicit description

$$a x + b y + d = 0$$
$$\mathbf{a}^T \mathbf{x} + d = 0$$

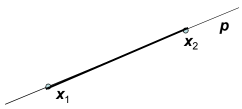
- parametric description

$$\mathbf{x}(t) = \mathbf{x}_0 + (\mathbf{x}_1 - \mathbf{x}_0)t$$
$$t \in (-\infty, \infty), \quad \text{resp. } t \in \langle 0, 1 \rangle$$

Lines and Line Segments

Lines in E^2

- two points define a line ***p***
- we have 2 equations for 3 parameters – a, b, d
- we have to solve linear homogeneous system, i.e. one parametric solution



$$a x_1 + b y_1 + d = 0$$
$$a x_2 + b y_2 + d = 0$$

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix}\begin{bmatrix} a \\ b \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Lines and Line Segments

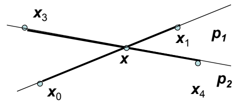
A point ***x*** as an intersection of two lines in E^2

- system of linear equations must be solved
- numerical stability of a solution

$$a_1 x + b_1 y + d_1 = 0$$
$$a_2 x + b_2 y + d_2 = 0$$

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -d_1 \\ -d_2 \end{bmatrix}$$

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}$$



Points and Planes in E^3

A point ***x*** as an intersection of three planes in E^3

- system of linear equations must be solved
- numerical stability of the solution if |det A| < ε then "singular case"?? What is ε value ??

$$a_1 x + b_1 y + c_1 z + d_1 = 0$$
$$a_2 x + b_2 y + c_2 z + d_2 = 0$$
$$a_3 x + b_3 y + c_3 z + d_3 = 0$$

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -d_1 \\ -d_2 \\ -d_3 \end{bmatrix}$$

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}$$

Principle of Duality

Any theorems remain true in E^2 when we interchange words "point" and "line", "lie on" and "pass through", "join" and "intersection" etc.

Points and planes are dual in E^3 etc. (not points and lines!)

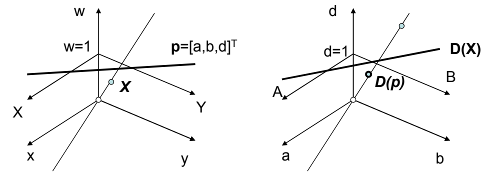
It means that intersection computation of two lines and a line given by two points should be same if we use the principle of duality.

Principle of Duality

- In E^2
  - a line **p** is given by two points (how to compute 3 coefficients $[a,b,d]^T$ of the line ax+by+d=0 ?)
  - a point **x** as an intersection of two lines
- In E^3
  - a plane **ρ** is given by three points (how to compute 4 coefficients $[a,b,c,d]^T$ of the plane ax+by+cz+d=0 ?)
  - a point **x** as an intersection of three planes

Dual problems – but computations are not "symmetrical"
Stability and robustness??
What happen if lines in E^2 are nearly collinear?

Principle of Duality



$$ax + by + d = 0$$

Computation in Projective Space

Intersection point ***x*** given by two lines $\mathbf{p}_1=[a_1,b_1,d_1]^T$ and $\mathbf{p}_2=[a_2,b_2,d_2]^T$

$$a_1X + b_1Y + d_1 = 0$$
$$a_2X + b_2Y + d_2 = 0$$
$$\mathbf{A}\,\mathbf{X} = -\mathbf{d} \qquad / * \ \xi \neq 0$$

$$\boldsymbol{x} = \mathbf{p}_1 \times \mathbf{p}_2$$

$$a_1x + b_1y + d_1w = 0$$
$$a_2x + b_2y + d_2w = 0$$
$$\mathbf{B}\,\mathbf{x} = \mathbf{0} \qquad \mathbf{x} = [x, y, w]^T$$

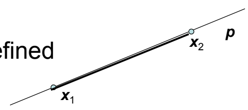
$$\mathbf{p}_1 \times \mathbf{p}_2 = \det\begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & b_1 & d_1 \\ a_2 & b_2 & d_2 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} a_1 & b_1 & d_1 \\ a_2 & b_2 & d_2 \end{bmatrix}$$

Computation in Projective Space

Lines in E^2

- a line ***p*** = $[a,b,d]^T$ is defined by points
  $\boldsymbol{x}_1=[x_1, y_1, w_1]^T$
  $\boldsymbol{x}_2=[x_2, y_2, w_2]^T$

  $\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2$



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

- we do not need to solve linear equations and no division operation is needed!
- stability evaluation AFTER computation

Computation in Projective Space

$$\mathbf{x}_1 \times \mathbf{x}_2 = \det\begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} \qquad \mathbf{a} = \begin{bmatrix} a & b & d \end{bmatrix}^T$$

It means

$$\mathbf{a}^T(\mathbf{x}_1 \times \mathbf{x}_2) = 0 \qquad \det\begin{bmatrix} a & b & d \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} = 0$$

Computation in Projective Space

In E^2

- A line ***p*** can be determined by cross product of two points $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$
- An intersection ***x*** point can be determined by cross product of two lines $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$
- If $\boldsymbol{x_1}$ or $\boldsymbol{x_2}$ in homogeneous coordinates, e.g. $w_i \neq 1$ no division is needed

How the E^3 case is handled?

- A point is dual to a plane ***ρ*** $\boldsymbol{\rho}=[a,b,c,d]^T$
- There is no "direct duality" for a line in E^3

Computation in Projective Space

- Cross product definition

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

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

$$ax + by + cz + dw = 0 \qquad \mathbf{a}^T \mathbf{x} = 0$$

Due to the duality

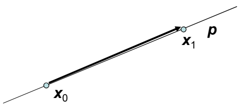
- An intersection point $\boldsymbol{x}$ of three planes is determined as a cross product of three given planes

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

Computation in Projective Space

- No division operation!
- An "intersection of parallel lines" can be computed - it leads to $[x,y,0]^T$, resp. $[x,y,\varepsilon]^T$ – a point in, resp.close to infinity
- More robust computations in general no IF clauses (conditions) are needed
- Substantial speed-up on CPU or GPU can be expected due to vector-vector operations support

Computation in Projective Space

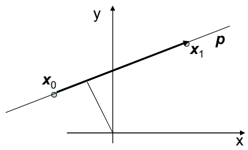


A line in E^3

- parametric form
- as an intersection of two planes
- Plücker coordinates

$$\mathbf{x}(t) = \mathbf{x}_0 + (\mathbf{x}_1 - \mathbf{x}_0)t$$
$$t \in (-\infty, \infty)$$

Computation in Projective Space

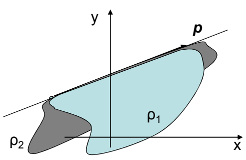


Plücker coordinates

$$\mathbf{p} = \left[(\mathbf{x}_1 - \mathbf{x}_2)^T : (\mathbf{x}_1 \times \mathbf{x}_2)^T \right]^T$$
$$\left[\boldsymbol{\omega}^T : \mathbf{v}^T \right]^T \quad \text{sometimes as } [\boldsymbol{\omega} : \mathbf{v}]^T$$

$$\mathbf{x}_i = [x_i, y_i, z_i, w_i]^T \qquad i = 1,2$$
$$\mathbf{L} = \mathbf{x}_0 \mathbf{x}_1^T - \mathbf{x}_1 \mathbf{x}_0^T \quad \textit{tensor product – matrix}$$
$$\boldsymbol{\omega} = [l_{41} \quad l_{42} \quad l_{43}]^T \qquad \mathbf{v} = [l_{23} \quad l_{31} \quad l_{12}]^T$$
$$\mathbf{q}(\tau) = \frac{\mathbf{v} \times \boldsymbol{\omega}}{\|\boldsymbol{\omega}\|^2} + \boldsymbol{\omega}\tau \qquad \mathbf{q}(\tau) = \begin{bmatrix} \mathbf{v} \times \boldsymbol{\omega} + \boldsymbol{\omega}\tau\|\boldsymbol{\omega}\|^2 \\ \|\boldsymbol{\omega}\|^2 \end{bmatrix}$$

Computation in Projective Space



Plücker coordinates

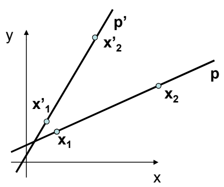
- a line given by two planes – due to **DUALITY**

$$\mathbf{q}(\tau) = \frac{\boldsymbol{\omega} \times \mathbf{v}}{\|\mathbf{v}\|^2} + \mathbf{v}\tau$$
$$\boldsymbol{\omega} = [l_{41} \quad l_{42} \quad l_{43}]^T \qquad \mathbf{v} = [l_{23} \quad l_{31} \quad l_{12}]^T$$
$$\mathbf{L} = \mathbf{a}_0 \mathbf{a}_1^T - \mathbf{a}_1 \mathbf{a}_0^T \quad \textit{tensor product – matrix}$$
$$\mathbf{a}_i = [a_i, b_i, c_i, d_i]^T \qquad i = 1,2$$

Computation in Projective Space

- Geometric transformations
  - points

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

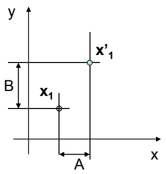


$$\mathbf{x}' = \mathbf{Q}\,\mathbf{x}$$
$$\mathbf{p}' = ? \quad \text{but}$$
$$\mathbf{Q}\,\mathbf{x}_1 \times \mathbf{Q}\,\mathbf{x}_2 = \left(\mathbf{Q}^{-1}\right)^T \mathbf{x}_1 \times \mathbf{x}_2$$

Computation in Projective Space

- Translation
  - distance given in E^2
  - distance given in P^2

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



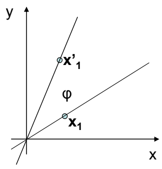
$$\mathbf{x}' = \begin{bmatrix} w & 0 & a \\ 0 & w & b \\ 0 & 0 & w \end{bmatrix} \mathbf{x} \quad (A, B) = \left(\frac{a}{w}, \frac{b}{w}\right) \quad w \neq 0$$

if A or B are fractions, we can avoid division !

Computation in Projective Space

- Rotation

$$\mathbf{x}' = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}$$

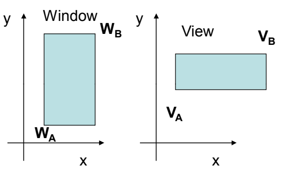


$$\mathbf{x}' = \begin{bmatrix} A & -B & 0 \\ B & A & 0 \\ 0 & 0 & C \end{bmatrix} \mathbf{x}$$

$$\cos\varphi = \frac{A}{C} \quad \sin\varphi = \frac{B}{C}$$

Computation in Projective Space

- Window-Viewport



$$\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & -Wx_A \\ 0 & 1 & -Wy_A \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{T}_2 = \begin{bmatrix} 1 & 0 & -Vx_A \\ 0 & 1 & -Vy_A \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{S} = \begin{bmatrix} \Delta Vx/\Delta Wx & 0 & 0 \\ 0 & \Delta Vy/\Delta Wy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{Q} = \mathbf{T}_2\ \mathbf{S}\ \mathbf{T}_1$$

$$\mathbf{S}' = \begin{bmatrix} \Delta Vx \Delta Wy & 0 & 0 \\ 0 & \Delta Vy \Delta Wx & 0 \\ 0 & 0 & \Delta Wx \Delta Wy \end{bmatrix} \quad \mathbf{Q}' = \mathbf{T}_2\ \mathbf{S}'\ \mathbf{T}_1$$

No division operation needed !!

Computation in Projective Space

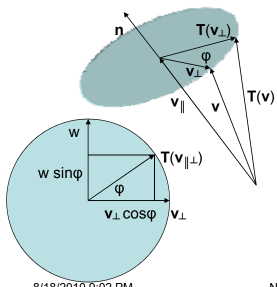
Fundamental geometric transformations

- translation, rotation, reflection, shearing, scaling, projection from E^3 to E^2
- can be performed without division operation and if parameters are given as fractions, matrices can be easily modified for parameters given in homogeneous representation

No division operation needed !!

Computation in Projective Space

Rotation around general axis



$\mathbf{T}(\mathbf{v}_\perp) = \mathbf{v}_\parallel \cos\varphi + \mathbf{w} \sin\varphi$

$\mathbf{T}(\mathbf{v}) = \mathbf{T}(\mathbf{v}_\parallel) + \mathbf{T}(\mathbf{v}_\perp)$

$$\mathbf{T} = \mathbf{I}\cos\varphi + (1-\cos\varphi)(\mathbf{n} \otimes \mathbf{n}) + (\mathbf{n} \times \mathbf{v})\sin\varphi \qquad \|\mathbf{n}\| = 1$$

$$\mathbf{n} \otimes \mathbf{n} = \mathbf{n}^T \mathbf{n} \quad \text{tensor product-matrix}$$

Interpolation

Linear parametrization

$$\mathbf{X}(t) = \mathbf{X}_0 + (\mathbf{X}_1 - \mathbf{X}_0)t \quad t \in (-\infty, \infty)$$

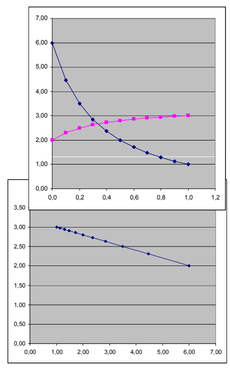
Non-linear (monotonous) parametrization

$$\mathbf{x}(t) = \mathbf{x}_0 + (\mathbf{x}_1 - \mathbf{x}_0)t \quad t \in (-\infty, \infty)$$

$$x(t) = x_0 + (x_1 - x_0)t \quad y(t) = y_0 + (y_1 - y_0)t$$

$$z(t) = z_0 + (z_1 - z_0)t \quad w(t) = w_0 + (w_1 - w_0)t$$

- It means that we can interpolate using homogeneous coordinates without a need of "normalization" to E^k !!
- Homogeneous coordinate $w \geq 0$
- In many algorithms, we need "monotonous" parameterization, only !



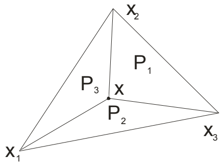
Barycentric coordinates

Let us consider a triangle with vertices $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$,
A position of any point $\mathbf{X} \in E^2$ can be expressed as

$$a_1 X_1 + a_2 X_2 + a_3 X_3 = X$$
$$a_1 Y_1 + a_2 Y_2 + a_3 Y_3 = Y$$

additional condition

$$a_1 + a_2 + a_3 = 1 \quad 0 \le a_i \le 1$$
$$a_i = \frac{P_i}{P} \quad i = 1,...,3$$



Linear system must be solved
If points $\mathbf{x}_i$ are given as $[x_i, y_i, z_i, w_i]^T$ and $w_i \neq 1$
then $\mathbf{x}_i$ must be "normalized"

Barycentric coordinates

It can be modified to:

$$b_1 X_1 + b_2 X_2 + b_3 X_3 + b_4 X = 0$$
$$b_1 Y_1 + b_2 Y_2 + b_3 Y_3 + b_4 Y = 0$$
$$b_1 + b_2 + b_3 + b_4 = 0$$
$$b_i = -a_i b_4 \quad i = 1,...,3 \quad b_4 \neq 0$$

Rewriting

$$\begin{bmatrix} X_1 & X_2 & X_3 & X \\ Y_1 & Y_2 & Y_3 & Y \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \mathbf{0}$$

$$\mathbf{b} = \boldsymbol{\xi} \times \boldsymbol{\eta} \times \mathbf{w}$$
$$\mathbf{b} = [b_1, b_2, b_3, b_4]^T$$
$$\boldsymbol{\xi} = [X_1, X_2, X_3, X]^T$$
$$\boldsymbol{\eta} = [Y_1, Y_2, Y_3, Y]^T$$
$$\mathbf{w} = [1,1,1,1]^T$$

Barycentric coordinates

if $w_i \neq 1$

$$\begin{bmatrix} x_1 & x_2 & x_3 & x \\ y_1 & y_2 & y_3 & y \\ w_1 & w_2 & w_3 & w \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \mathbf{0}$$

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

Barycentric coordinates

if $w_i \neq 1$

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

$$0 \le (-b_1 : w_2 w_3 w) \le 1$$
$$0 \le (-b_2 : w_3 w_1 w) \le 1$$
$$0 \le (-b_3 : w_1 w_2 w) \le 1$$

It means that we can compute barycentric coordinates without division operation

Simple modification for a position in the tetrahedron [4]

Length, Area and Volume

Length, area and volume computation in projective space if an element is given by points in homogeneous coordinates

Line segment length

$$\mathbf{p} = \mathbf{x}_1 \times \mathbf{x}_2 \qquad \mathbf{p} = [a, b, d]^T \quad \mathbf{n} = [a, b]^T$$
$$l = \left| \left(\sqrt{\mathbf{n}^T \mathbf{n}} : w_1 w_2 \right) \right|$$

Triangle area

$$\boldsymbol{\rho} = \mathbf{x}_1 \times \mathbf{x}_2 \times \mathbf{x}_3 \qquad \boldsymbol{\rho} = [a, b, c, d]^T \quad \mathbf{n} = [a, b, c]^T$$
$$S = \left| \left(\sqrt{\mathbf{n}^T \mathbf{n}} : 2 w_1 w_2 w_3 \right) \right|$$

Length, Area and Volume

Tetrahedron volume

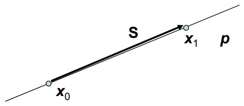
$$\boldsymbol{\rho} = \mathbf{x}_1 \times \mathbf{x}_2 \times \mathbf{x}_3 \times \mathbf{x}_4 \qquad \boldsymbol{\rho} = [a, b, c, d, e]^T \quad \mathbf{n} = [a, b, c, d]^T$$
$$V = \left| \left(\sqrt{\mathbf{n}^T \mathbf{n}} : 6 w_1 w_2 w_3 w_4 \right) \right|$$

General formula

$$Q_k = \left| \left(\sqrt{\mathbf{n}^T \mathbf{n}} : (k-1)! \prod_{i=1}^{k} w_i \right) \right| \quad \text{k = number of end-points}$$

$$\bar{\mathbf{n}} = \frac{\mathbf{n}}{\sqrt{\mathbf{n}^T \mathbf{n}}} = \left(\mathbf{n} : (k-1)!\, Q_k \prod_{i=1}^{k} w_i \right)$$

Intersection Computation in Projective Space



$$\mathbf{X}(t) = \mathbf{X}_0 + (\mathbf{X}_1 - \mathbf{X}_0)t \quad t \in (-\infty, \infty)$$
$$\mathbf{A}^T\mathbf{X} + d = 0 \qquad ax + by + cz + d = 0$$
$$\mathbf{A} = [a,b,c]^T \qquad \mathbf{S} = \mathbf{X}_1 - \mathbf{X}_0$$
$$t = -\frac{\mathbf{A}^T\mathbf{X}_0 + d}{\mathbf{A}^T\mathbf{S}}$$

- Linear interpolation& parameterization very often used
- Intersection of a line and a plane
- Robustness problems if $\mathbf{A}^T\mathbf{S} \rightarrow 0$

??? How to avoid an instability ???

Intersection Computation in Projective Space

$$\mathbf{x}(t) = \mathbf{x}_0 + (\mathbf{x}_1 - \mathbf{x}_0)t \quad t \in (-\infty, \infty)$$
$$\mathbf{a}^T\mathbf{x} = 0 \qquad ax + by + cz + d = 0$$
$$\mathbf{a} = [a,b,c,d]^T \qquad \mathbf{S} = \mathbf{X}_1 - \mathbf{X}_0$$
$$t = -\frac{\mathbf{a}^T\mathbf{x}_0}{\mathbf{a}^T\mathbf{s}}$$
$$\tau = -\mathbf{a}^T\mathbf{x}_0 \qquad \tau_w = \mathbf{a}^T\mathbf{s}$$
$$\mathbf{t} = [\tau : \tau_w] \quad \textbf{if } \tau_w \le 0 \textbf{ then } \mathbf{t} := -\mathbf{t}$$

TEST

if $t > t_{min}$ **then.....**

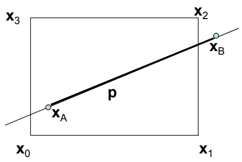
if $\tau * \tau_{min_w} > \tau_w * \tau_{min}$ **then.....** condition $\tau \ge 0$

- An intersection of a plane with a line in E^2 can be computed efficiently [6]
- Comparison operations must be modified !!!
- Cyrus-Beck line clipping algorithm 10-25% faster

Intersection Computation in Projective Space

Line clipping in E^2 algorithms

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




```
procedure CLIP_L; {details in [3]}
{ xA , xB – in homogeneous coordinates }
{ The EXIT ends the procedure }
{ input: xA , xB ; xA=[xA,yA,1]T  p = [a,b,c]T }
begin
{1}    p := xA x xB;  { ax+by+c = 0}
{2}    for k:=0 to N-1 do { xk=[xk,yk,1]T }
{3}           if pTxk ≥ 0 then ck:=1
                      else ck:=0;
{4}    if  c = [0000]T or c = [1111]T
                             then EXIT;
{5}    i:= TAB1[c];      j:= TAB2[c];
{6}    xA := p x ei ;     xB := p x ej ;
{7}    DRAW (xA; xB )     {ei – i-th edge }
end {CLIP_L};
```

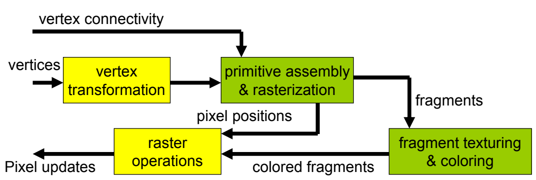

Intersection Computation in Projective Space

Iterative computations

- values are represented as fractions with floats
- exponents grow – need of "exponents normalization"
  - not available on current CPUs
  - necessity of explicit CALL
- solution - see PLib for .NET [8]

Cg / HLSL and GPU Computing

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



Cg / HLSL and GPU Computing

- 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;
  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;
}
```

Conclusion

- Fundamentals of computation in projective space have been introduced
- Proposed approach helps to improve robustness of algorithms, but it does not give the ultimate solution - limited numerical precision
- Homogeneous coordinate *w* must be non-negative (simplification comparison operations)
- Comparison operations are a little bit complicated – but tests rely on separation functions – higher robustness
- Due to GPU and CPU architecture algorithms might be significantly faster even in SW implementation

Conclusion

- A new data type for programming languages – float_projective, double_projective should be considered
  - perhaps as a native representation
  - it enables more robust numerical algorithms
  - unfortunately increases a data bus traffic
  - operation “exponent normalization” should be supported on CPU/GPU in HW – significantly slow in SW
  - experimental library PLib is available [8]
- Geometry algebra applications in CG & CV ??

References

[1] van Verth,J.M., Bishop,L.M.: Essential Mathematics for Games and Interactive Applications, Morgan Kaufmann 2005
[2] Skala,V.: GPU Computation in Projective Space Using Homogeneous Coordinates , Game Programming GEMS 6 (Ed.Dickheiser,M.), pp.137-147, Charles River Media, 2006
[3] 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
[4] Skala,V.: Barycentric coordinates computation in homogeneous coordinates, submitted for publication, 2007
[5] Skala,V.: Length, Area and Volume Computation in Homogeneous Coordinates, International Journal of Image and Graphics, Vol.6., No.4, pp.625-639, 2006
[6] Yamaguchi,F.: Computer Aided Design – A totally Four-Dimensional Approach, Springer Verlag 2002
[7] Fernando,R., Kilgard,M.J.: The Cg Tutorial, Addison Wesley, 2003
[8] Skala,V., Kaiser,J., Ondracka,V.: Library for Computation in the Projective Space, Aplimat 2007 conf., 2007

References

Additional readings:

- Nielsen,F: Visual Computing: Geometry, Graphics and Vision, Charles River Media, 2005
- Hartley,R., Zisserman,A.: Multiple View Geometry in Computer Vision, Cambridge Univ.Press, 2000
- Farin,G., Hansford,D.: Practical Linear Algebra – A Geometry Toolbox, A.K.Peters, 2005
- Doran,Ch., Lasenby,A.: Geometric Algebra for Physicist, Cambridge Univ.Press, 2005
- Hanák,I.: The GPU and Graphic Algorithms http://www.kiv.zcu.cz/publications/2005/tr-2005-05.pdf

Acknowledgment

Thanks belong to many colleagues and students at the University of West Bohemia for their critical comments and suggestions.

Activites supported by:

- 3DTV, project FP6 NoE No.511568
- VIRTUAL, project MSMT Czech Rep.,No.2C06002
- LC CPG, project MSMT Czech Rep., No.LC 06008

Thank you for your attention

Questions ??

Contact
Vaclav Skala
skala@kiv.zcu.cz subj. 3DTV - Tutorial

Center of Computer Graphics and Visualization http://herakles.zcu.cz
Department of Informatics and Computer Science http://www.kiv.zcu.cz
Faculty of Applied Sciences http://www.fav.zcu.cz
University of West Bohemia http://www.zcu.cz
Univerzitni 8
CZ 306 14 Plzen
Czech Republic