Computer Graphics

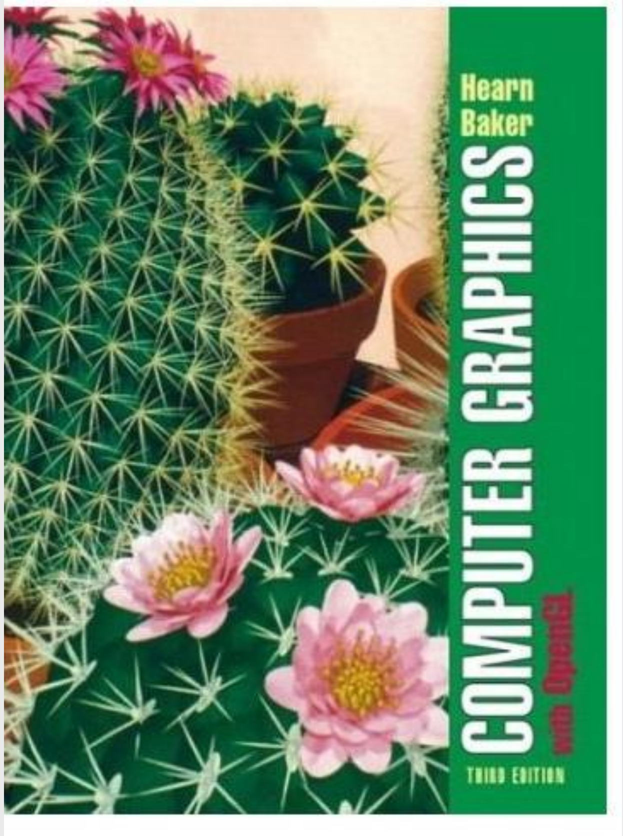
Prof. Dr. Elnomery Zanaty

Aims

- Introduce basic graphics concepts and terminology
- Base for development of 2D interactive computer graphics programmes (OGO 2.3)

The textbook

- D. Hearn, M.P. Baker, "Computer Graphics with OpenGL", 3rd Edition, 2004, ISBN 0-13-015390-7
- Available at the bookstore



Literature

Computer Graphics - Principles and Practice
Foley - van Dam - Feiner - Hughes
2nd edition in C - Addison and Wesley

Computer Graphics - C Version
Donald Hearn - M. Pauline Baker
2nd edition - international edition
Prentice Hall

Overview

Introduction

Geometry

Interaction

Raster graphics

Introduction

What is Computer Graphics?

Applications

Computer Graphics

Raster/vector graphics

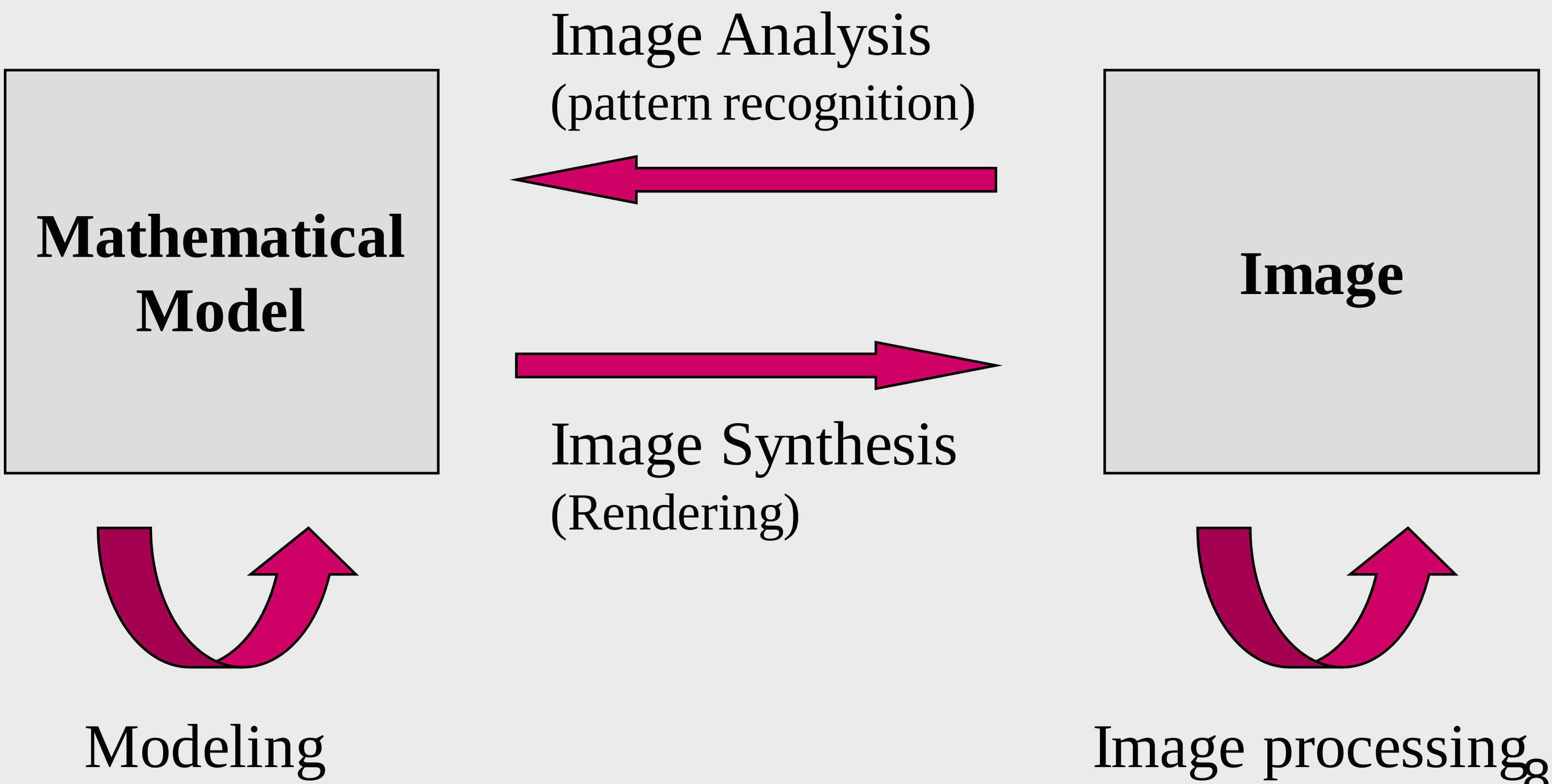
Hardware

Computer Graphics

Computer Graphics is ubiquitous:

- Visual system is most important sense:
  - High bandwidth
  - Natural communication
- Fast developments in
  - Hardware
  - Software

Computer Graphics



Supporting Disciplines

- Computer science *(algorithms, data structures, software engineering, …)*
- Mathematics *(geometry, numerical, …)*
- Physics *(Optics, mechanics, …)*
- Psychology *(Colour, perception)*
- Art and design

Applications

- Computer Aided Design (CAD)
- Computer Aided Geometric Design (CAGD)
- Entertainment (animation, games, …)
- Geographic Information Systems (GIS)
- Visualization (Scientific Vis., Inform. Vis.)
- Medical Visualization
- …

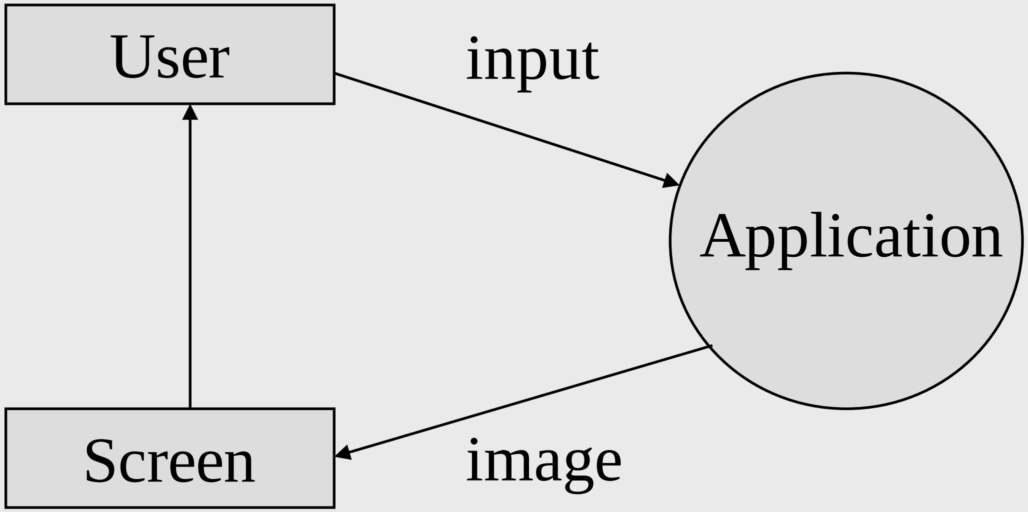
Computer Graphics

Current:

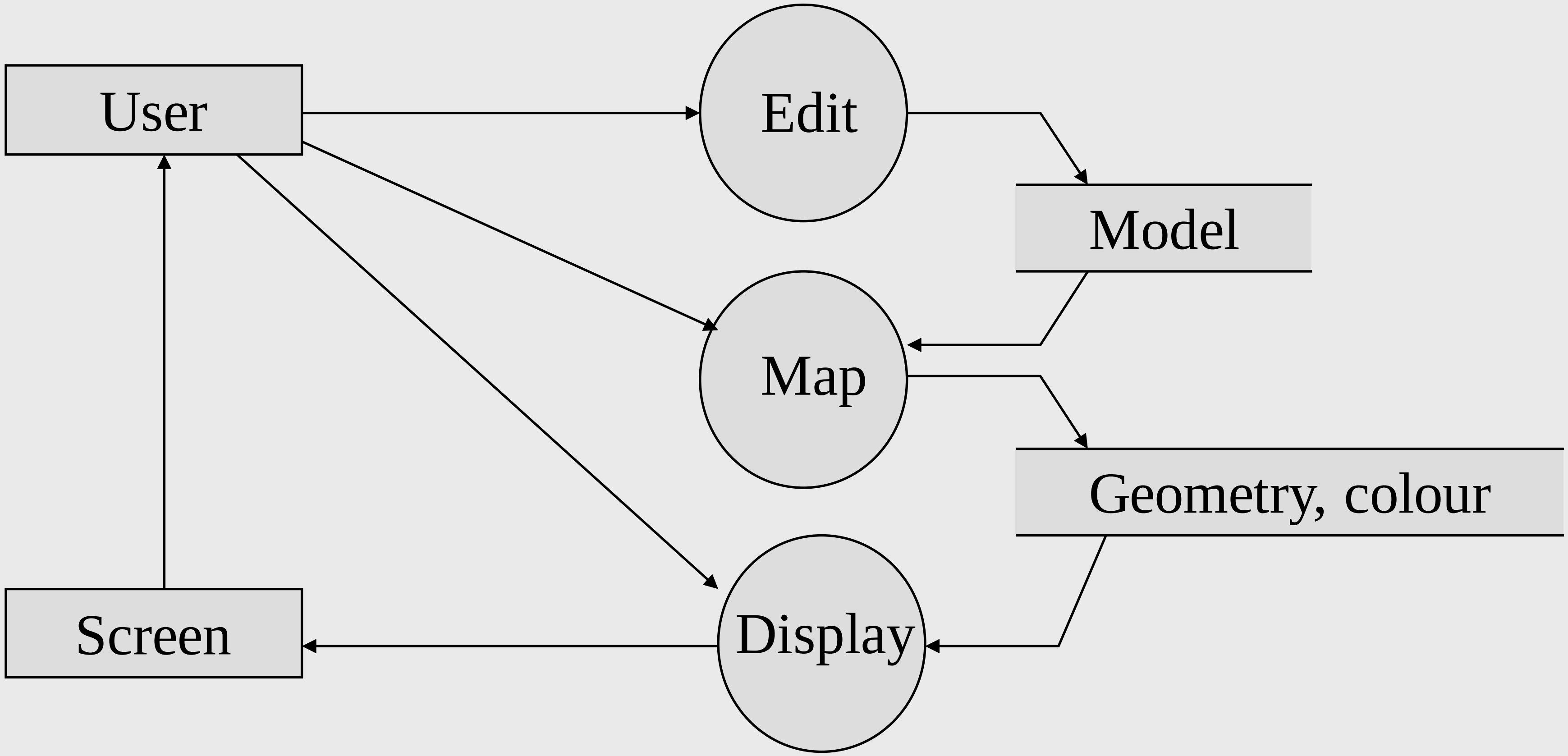
- Information visualisation
- Interactive 3D design
- Virtual reality

Past: Rasterization, Animation

Interactive Computer Graphics



Graphics pipeline



Representations in graphics

Vector Graphics

- *Image is represented by continuous geometric objects: lines, curves, etc.*

Raster Graphics

- *Image is represented as an rectangular grid of coloured squares*

Vector graphics

- Graphics objects: geometry + colour
- Complexity ~ O(number of objects)
- Geometric transformation possible without loss of information (zoom, rotate, …)
- Diagrams, schemes, ...
- Examples: PowerPoint, CorelDraw, ...

Raster graphics

- Generic
- Image processing techniques
- Geometric Transformation: loss of information
- Complexity ~ O(number of pixels)
- Jagged edges, anti-aliasing
- Realistic images, textures, ...
- Examples: Paint, PhotoShop, ...

Conversion

Vector graphics

Rasterization,

Scan conversion

Pattern recognition

Raster graphics

Hardware

- Vector graphics
- Raster graphics
- Colour lookup table
- 3D rendering hardware

Vector Graphics Hardware

continuous & smooth lines

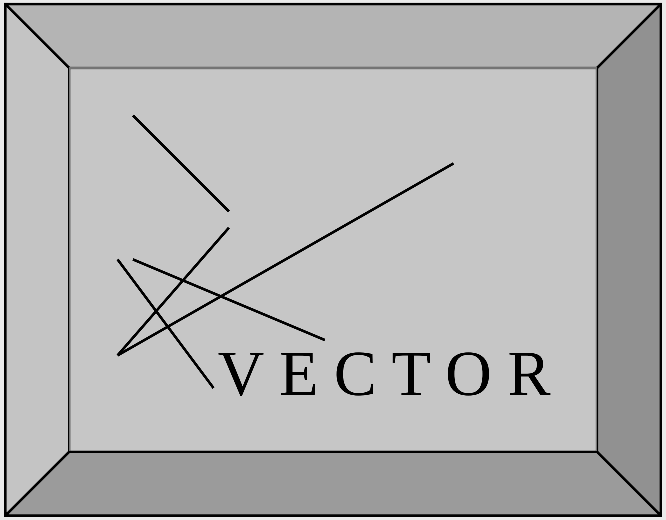
no filled objects

random scan

refresh speed depends on complexity of the scene

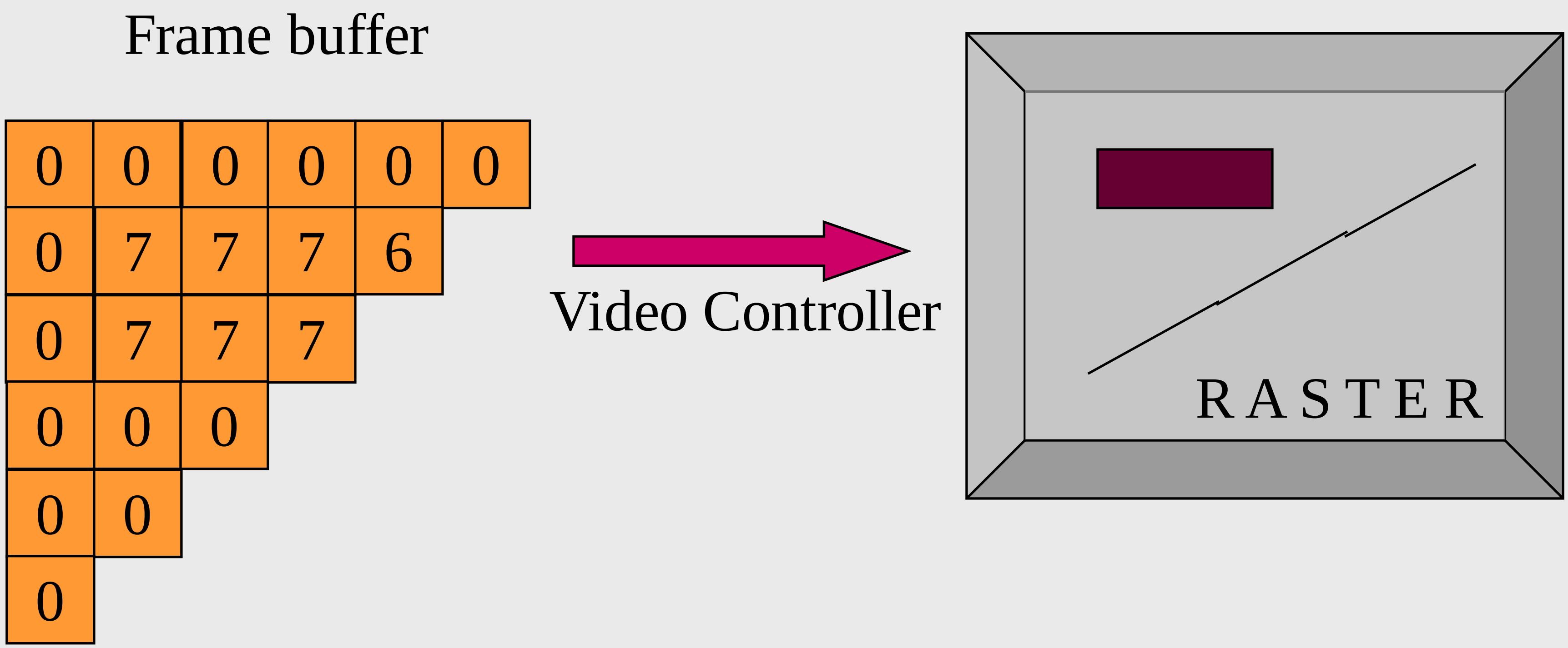
Display list

move 10 20
line 20 40
...
char O
char R



Display Controller

Raster Graphics Hardware



jaggies (stair casing)

filled objects

(anti)aliasing

refresh speed independent of scene complexity

pixel

scan conversion

resolution

bit planes

Colour Lookup Table

Frame buffer

0	0	0	0	0	0
0	7	7	7	6	
0	7	7	7		
0	0	0			
0	0				
0					

colour index	R G B
0	0 0 0
1	102 255 53
2	255 255 204
4	255 102 153
…	
7	102 0 51

CLUT:

pixel = code

True colour:

pixel = R,G,B

3D rendering hardware

Geometric representation: *Triangles*

Viewing: *Transformation*

Hidden surface removal: *z-buffer*

Lighting and illumination: *Gouraud shading*

Realism: *texture mapping*

Special effects: *transparency, antialiasing*

2D geometric modelling

- Coordinates
- Transformations
- Parametric and implicit representations
- Algorithms

Coordinates

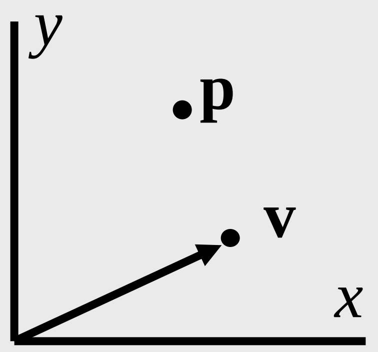
- Point: position on plane

  $\mathbf{p} = (p_x, p_y)$

  $\mathbf{x} = (x, y)$

  $\mathbf{x} = (x_1, x_2)$

  $\mathbf{x} = x_1 \mathbf{e}_1 + x_2 \mathbf{e}_2, \quad \mathbf{e}_1 = (1, 0), \quad \mathbf{e}_2 = (0, 1)$



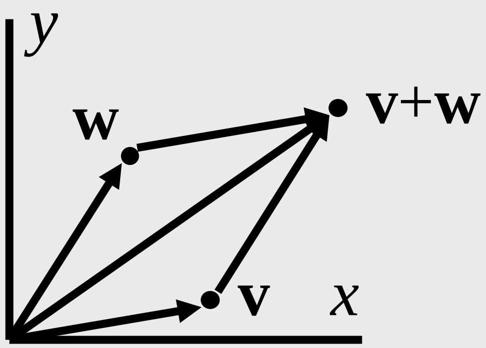
- Vector: direction and magnitude

  $\mathbf{v} = (v_x, v_y)$, etc.

Vector arithmetic

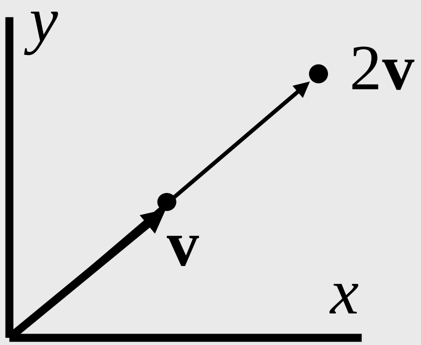
- Addition of two vectors:

  $$\mathbf{v} + \mathbf{w} = (v_x + w_x, v_y + w_y)$$

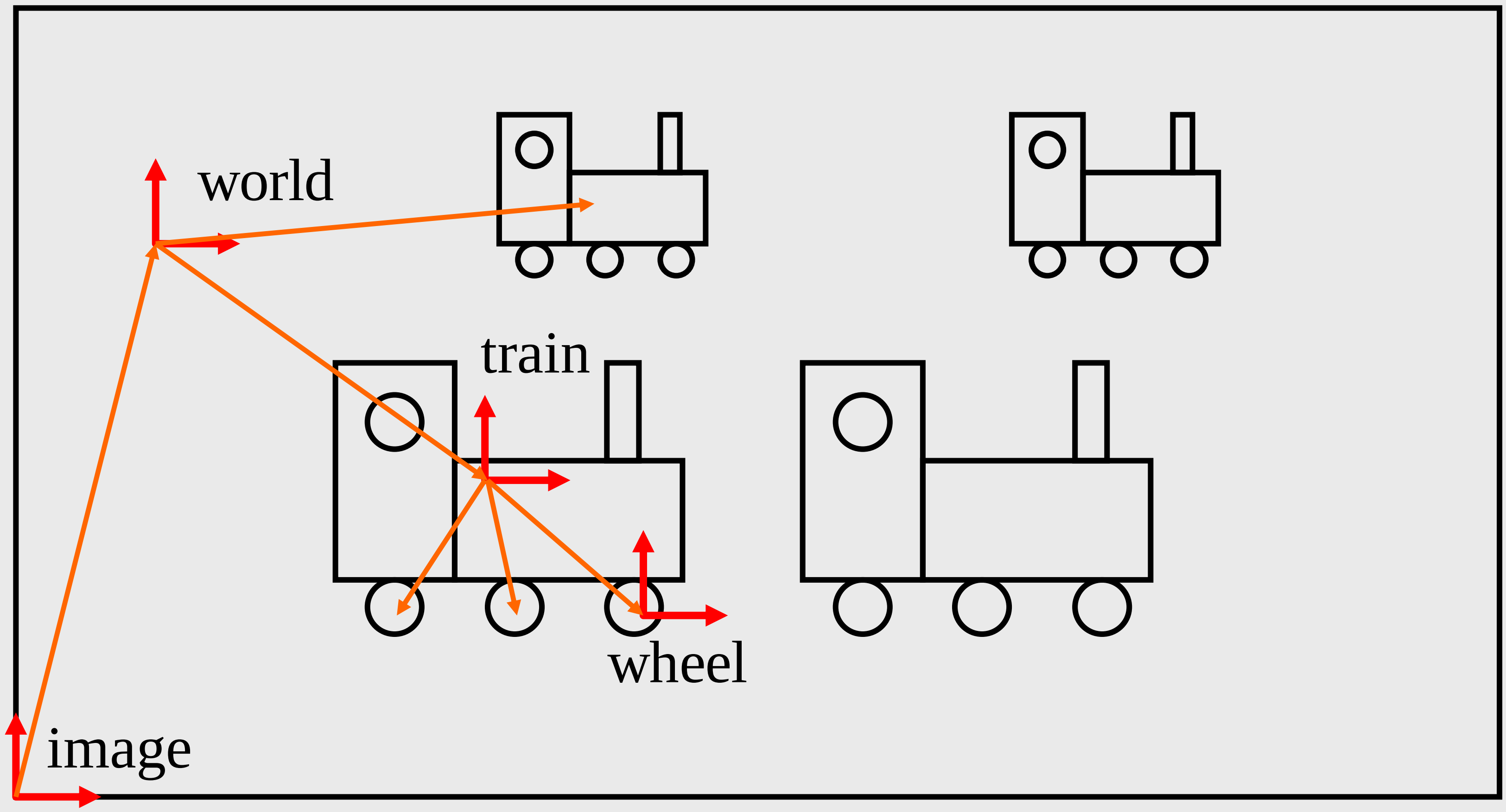


- Multiplication vector-scalar:

  $$\alpha\mathbf{v} = (\alpha v_x, \alpha v_y)$$



Coordinate systems



Why transformations?

- Model of objects

  world coordinates: *km*, *mm*, *etc*.

  hierarchical models:

  human = torso + arm + arm + head + leg + leg

  arm = upperarm + lowerarm + hand …

- Viewing

  zoom in, move drawing, etc.

Transformation types

- Translate according to vector **v**:

  $\mathbf{t} = \mathbf{p} + \mathbf{v}$

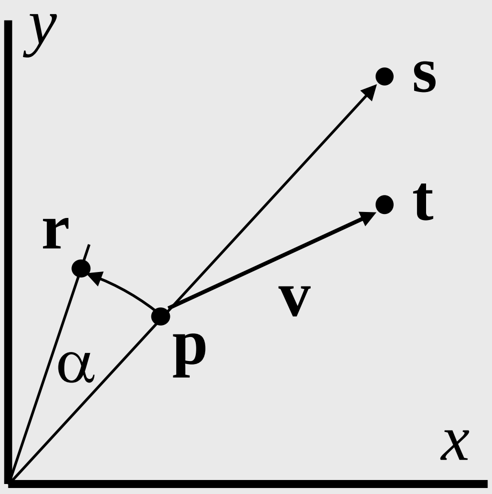
- Scale with factor *s*:

  $\mathbf{s} = s\mathbf{p}$

- Rotate over angle α:

  $r_x = \cos(\alpha)p_x - \sin(\alpha)p_y$

  $r_y = \sin(\alpha)p_x + \cos(\alpha)p_y$



Homogeneous coordinates

- Unified representation of rotation, scaling, translation
- Unified representation of points and vectors
- Compact representation for sequences of transformations
- Here: convenient notation, much more to it

Homogeneous coordinates

- Extra coordinate added:

  $\mathbf{p} = (p_x , p_y , p_w)$ or

  $\mathbf{x} = (x, y, w)$

- Cartesian coordinates: divide by w

  $\mathbf{x} = (x/w, y/w)$

- Here: for a point $w = 1$, for a vector $w = 0$

Matrices for transformation

$\mathbf{x}' = \mathrm{M}\mathbf{x}$, or

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}, \text{ or}$$

$$x' = m_{11}x + m_{12}y + m_{13}w$$

$$y' = m_{21}x + m_{22}y + m_{23}w$$

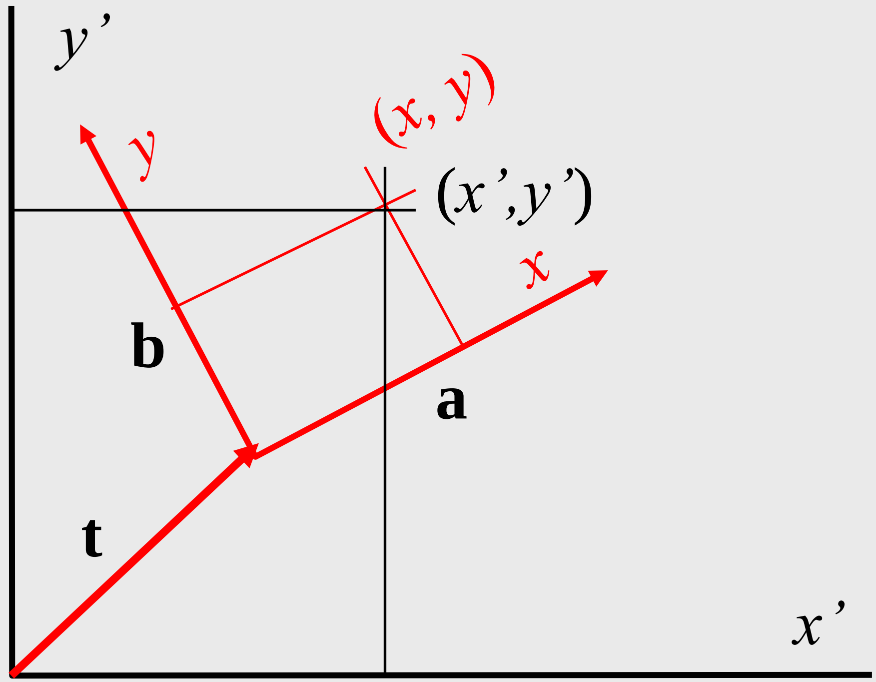
$$w' = m_{31}x + m_{32}y + m_{33}w$$

Direct interpretation

$$\mathbf{x}' = \mathrm{M}\,\mathbf{x}, \text{ or}$$

$$\mathbf{x}' = \begin{pmatrix}\mathbf{a} & \mathbf{b} & \mathbf{t}\end{pmatrix}\mathbf{x}, \text{ or}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_x & b_x & t_x \\ a_y & b_y & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



Translation matrix

$$\text{Translation:}$$

$$\mathbf{x}' = \mathrm{T}(t_x, t_y)\mathbf{x}\text{, with}$$

$$\mathrm{T}(t_x, t_y) = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

Scaling matrix

Scaling :

$\mathbf{x}' = \mathrm{S}(s_x, s_y)\mathbf{x}$, with

$$\mathrm{S}(s_x, s_y) = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

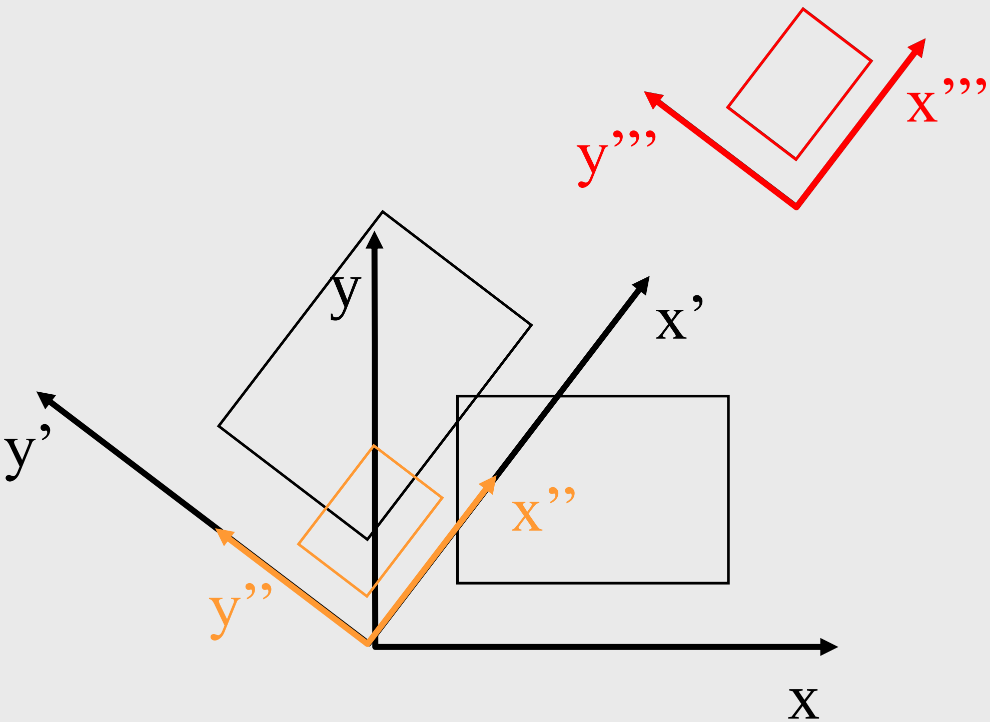
Rotation matrix

$$\text{Rotation:}$$

$$\mathbf{x}' = \mathrm{R}(\alpha)\,\mathbf{x}\,,\ \text{with}$$

$$\mathrm{R}(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Sequences of transformations



$$\mathbf{x}' = R(\pi/2)\mathbf{x}$$

$$\mathbf{x}'' = S(1/2)\mathbf{x}'$$

$$\mathbf{x}''' = T(5,4)\mathbf{x}''$$

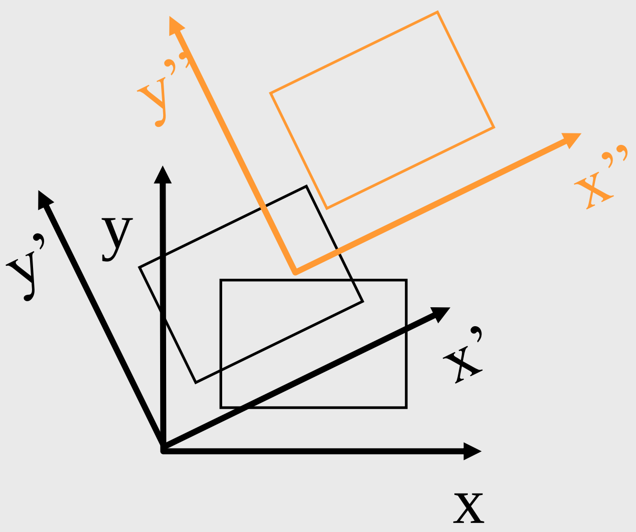
or

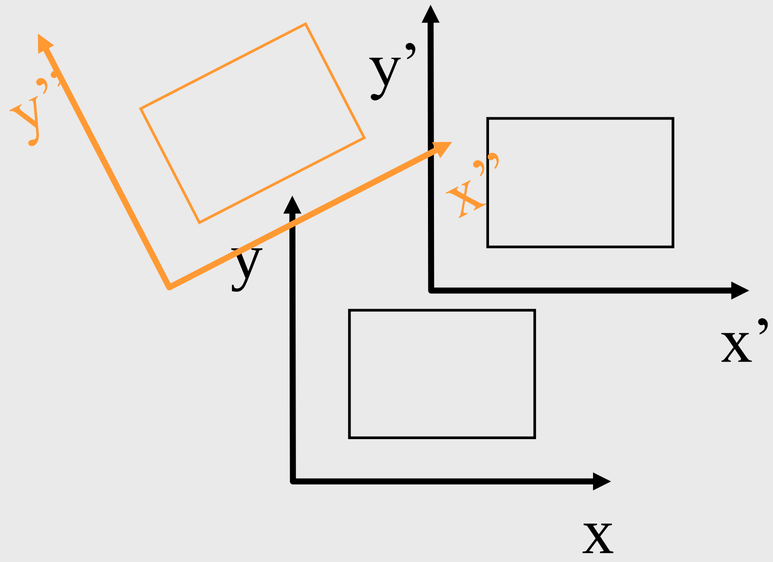
$$\mathbf{x}''' = M\mathbf{x}, \text{ with}$$

$$M = T(5,4)S(1/2)R(\pi/2)$$

Sequences of transformations can be described with a single transformation matrix, which is the result of concatenation of all transformations.

Order of transformations





$\mathbf{x}'' = T(2,3)R(30)\mathbf{x}$

$\mathbf{x}'' = R(30)T(2,3)\mathbf{x}$

Matrix multiplication is not commutative. Different orders of multiplication give different results.

Order of transformations

- Pre-multiplication:

  $\mathbf{x'} = M_n M_{n-1} \ldots M_2 M_1 \mathbf{x}$

  Transformation M_n in global coordinates
- Post-multiplication:

  $\mathbf{x'} = M_1 M_2 \ldots M_{n-1} M_n \mathbf{x}$

  Transformation M_n in local coordinates, i.e., the coordinate system that results from application of $M_1 M_2 \ldots M_{n-1}$

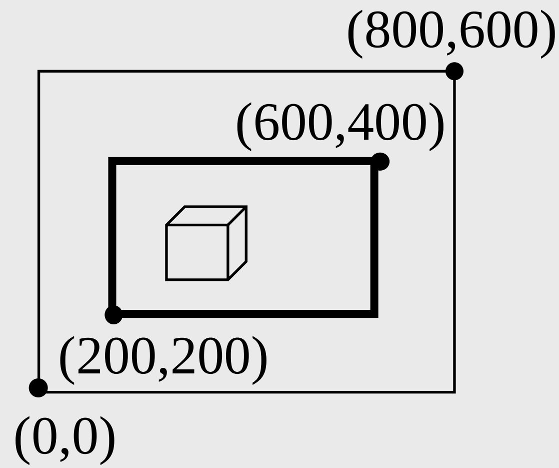
Window and viewport

Viewport:

Area on screen to be used for drawing.

Unit: pixels (screen coordinates)

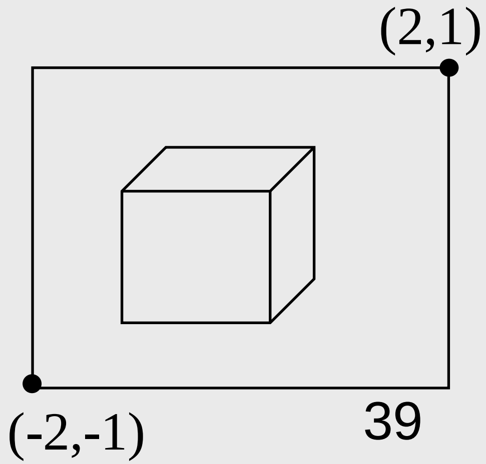
Note: *y*-axis often points down



Window:

Virtual area to be used by application

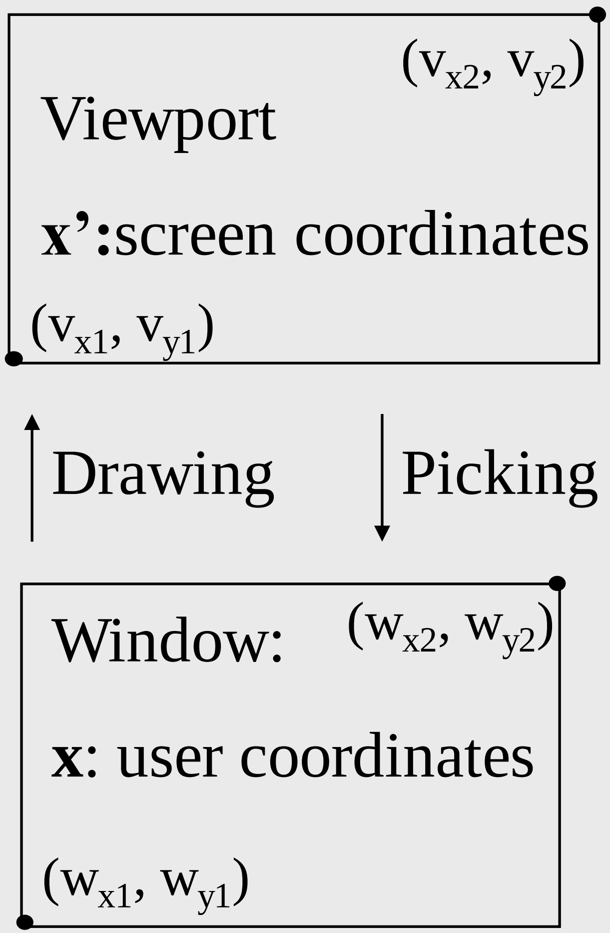
Unit: km, mm,… (world coordinates)



Window/viewport transform

- Determine a matrix M, such that the *window* (w_{x1}, w_{x2}, w_{y1}, w_{y2}) is mapped on the *viewport* (v_{x1}, v_{x2}, v_{y1}, v_{y2}):
- $A = T(-w_{x1}, -w_{y1})$
- $B = S(1/(w_{x2}-w_{x1}), 1/(w_{y2}-w_{y1}))\ A$
- $C = S(v_{x2}-v_{x1}, v_{y2}-v_{y1})B$
- $M = T(v_{x1}, v_{y1})\ C$

Forward and backward



Drawing**:** (meters to pixels)

Use $\mathbf{x'} = M\mathbf{x}$

Picking**:**(pixels to meters)

Use $\mathbf{x} = M^{-1}\mathbf{x'}$

Implementation example

Suppose, basic library supports two functions:

- MoveTo(x, y: integer);
- LineTo(x, y: integer);
- x and y in pixels.

How to make life easier?

State variables

- Define state variables:

  Viewport: array[1..2, 1..2] of integer;

  Window: array:[1..2, 1..2] of real;

  Mwv, Mobject: array[1..3, 1..3] of real;

  Mwv: transformation from world to view

  Mobject: extra object transformation

Procedures

- Define coordinate system:

  SetViewPort(x1, x2, y1, y2):

    Update Viewport and Mwv

  SetWindow(x1, x2, y1, y2):

    Update Window and Mwv

Procedures (continued)

- Define object transformation:

  ResetTrans:

  Mobject := IdentityMatrix

  Translate(tx, ty):

  Mobject := T(tx,ty)* Mobject

  Rotate(alpha):

  Mobject := R(tx,ty)* Mobject

  Scale(sx, sy):

  Mobject := S(sx, sy)* Mobject

Procedures (continued)

- Handling hierarchical models:
  - PushMatrix();

    Push an object transformation on a stack;
  - PopMatrix()

    Pop an object transformation from the stack.

Or:

  - GetMatrix(M);
  - SetMatrix(M);

Procedures (continued)

- Drawing procedures:

  MyMoveTo(x, y):

    (x’, y’) = Mwv*Mobject*(x,y);

    MoveTo(x’, y’)

  MyLineTo(x,y):

    (x’, y’) = Mwv*Mobject*(x,y);

    LineTo(x’, y’)

Application

```
DrawUnitSquare:
   MyMoveTo(0, 0);
   MyLineTo(1, 0);
   MyLineTo(1, 1);
   MyLineTo(0, 1);
   MyLineTo(0, 0);

Initialize:
   SetViewPort(0, 100, 0, 100);
   SetWindow(0, 1, 0, 1);
```

```
Main program:
   Initialize;
   Translate(-0.5, -0.5);
   for i := 1 to 10 do
   begin
       Rotate(pi/20);
       Scale(0.9, 0.9);
       DrawUnitSquare;
   end;
```

Puzzles

- Modify the window/viewport transform for a display y-axis pointing downwards.
- How to maintain aspect-ratio world->view? Which state variables?
- Define a transformation that transforms a unit square into a “wybertje”, centred around the origin with width *w* and height *h*.

Geometry

- Dot product, determinant
- Representations
- Line
- Ellipse
- Polygon

Good and bad

- Good: symmetric in x and y
- Good: matrices, vectors
- Bad: $y = f(x)$

- Good: dot product, determinant
- Bad: arcsin, arccos

Dot product

Notation : $\mathbf{v} \cdot \mathbf{w}$ (sometimes (v, w))

Definition :

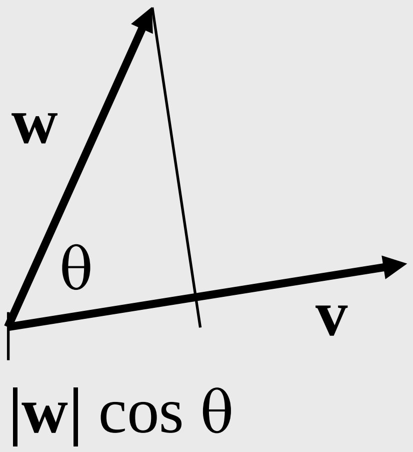
$$\mathbf{v} \cdot \mathbf{w} = v_x w_x + v_y w_y$$

Also :

$$\mathbf{v} \cdot \mathbf{w} = |\,\mathbf{v}\,||\,\mathbf{w}\,|\cos\theta \quad (0 \le \theta \le \pi)$$

with θ angle between **v** and **w**,

and $|\,\mathbf{v}\,|$ is the length of vector **v**



Dot product properties

$$\mathbf{v} \cdot \mathbf{w} = \mathbf{w} \cdot \mathbf{v}$$

$$(\mathbf{v} + \mathbf{w}) \cdot \mathbf{u} = \mathbf{v} \cdot \mathbf{u} + \mathbf{w} \cdot \mathbf{u}$$

$$(\lambda \mathbf{v}) \cdot \mathbf{w} = \lambda \ \mathbf{v} \cdot \mathbf{w}$$

$$\mathbf{v} \cdot \mathbf{v} = |\mathbf{v}|^2$$

$$\mathbf{v} \cdot \mathbf{w} = 0 \ \text{ iff } \ \mathbf{v} \text{ and } \mathbf{w} \text{ are perpendicular}$$

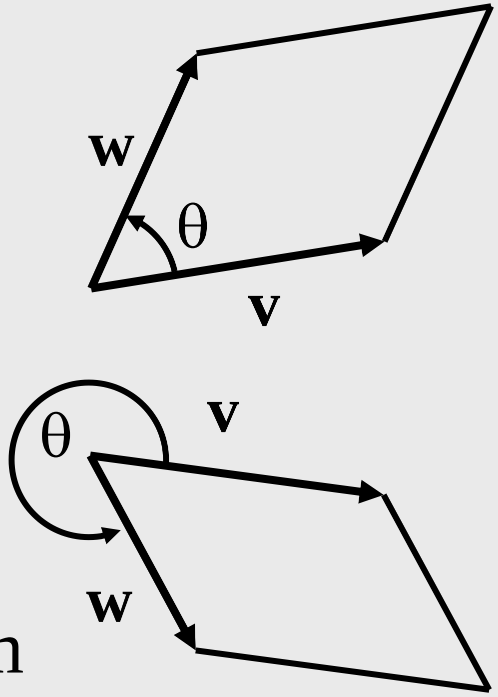
Determinant

$$Det(\mathbf{v},\mathbf{w}) = v_x w_y - v_y w_x$$
$$= |\mathbf{v}||\mathbf{w}|\sin\theta$$

θ is angle from **v** to **w**

$0 < \theta < \pi : Det(\mathbf{v},\mathbf{w}) > 0$

$\pi < \theta < 2\pi : Det(\mathbf{v},\mathbf{w}) < 0$



Det(**v**, **w**): signed area of parallellogram

Det(**v**, **w**) = 0 iff **v** and **w** are parallel

Curve representations

- Parametric: $\mathbf{x}(t) = (x(t), y(t))$
- Implicit: $\mathrm{f}(\mathbf{x}) = 0$

Parametric line representation

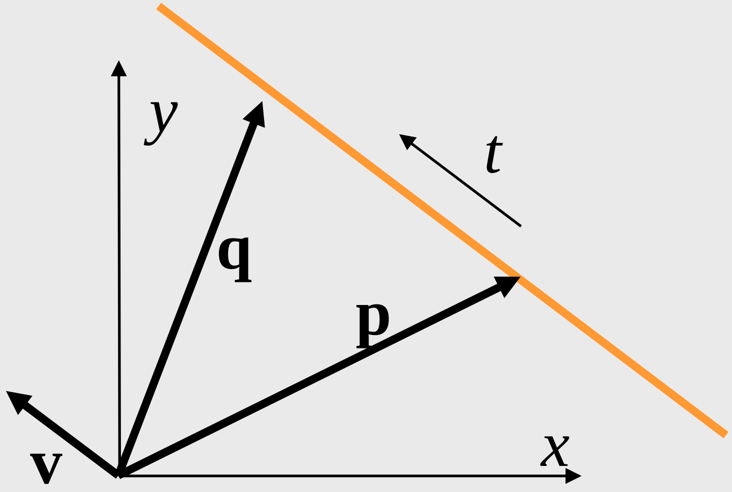
Given point **p** and vector **v:**

$$\mathbf{x}(t) = \mathbf{p} + \mathbf{v}t$$

Given two points **p** and **q:**

$$\mathbf{x}(t) = \mathbf{p} + (\mathbf{q}-\mathbf{p})t \text{ , or}$$

$$= \mathbf{p}t + \mathbf{q}(1-t)$$



Parametric representation

- $\mathbf{x}(t) = (x(t), y(t))$
- Trace out curve:

  MoveTo($\mathbf{x}$(0));

  for i := 1 to N do LineTo($\mathbf{x}$(i*Δt));
- Define segment: $t_{min} \leq t \leq t_{max}$

Implicit line representation

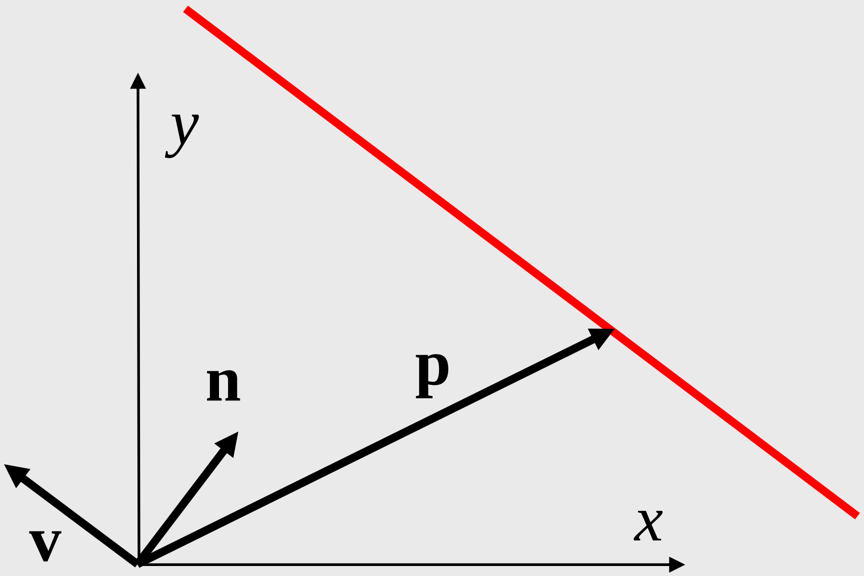
- $(\mathbf{x}-\mathbf{p}).\mathbf{n} = 0$

  with $\mathbf{n}.\mathbf{v} = 0$

  n is normal vector:

  $\mathbf{n} = [-v_y , v_x]$
- Also:

  $ax+by+c=0$



Implicit representation

$f(\mathbf{x}) = 0$: curve

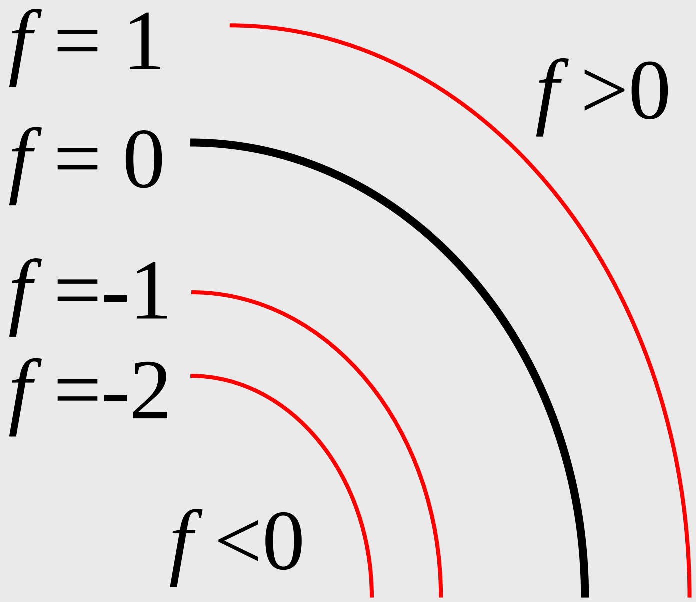
$f(\mathbf{x}) = C$: contours

$f = 0$ divides plane in

two areas: $f > 0$ and $f < 0$

$|f(\mathbf{x})|$: measure of distance

of $\mathbf{x}$ to curve



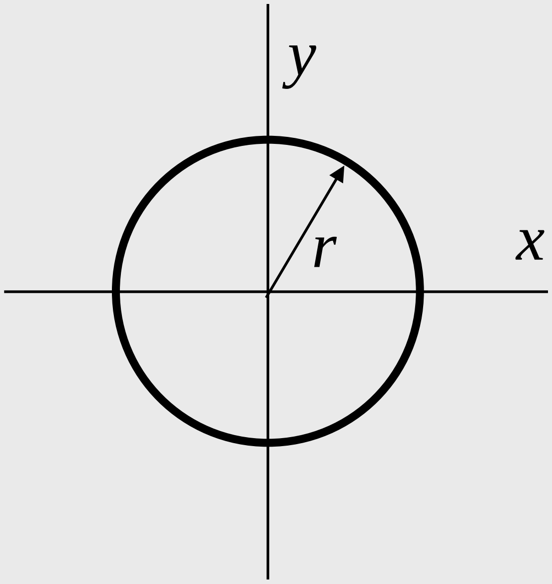
Circle

Parametric:

$$(x, y) = (r\cos\alpha, r\sin\alpha)$$

Implicit:

$$x^2 + y^2 - r^2 = 0$$



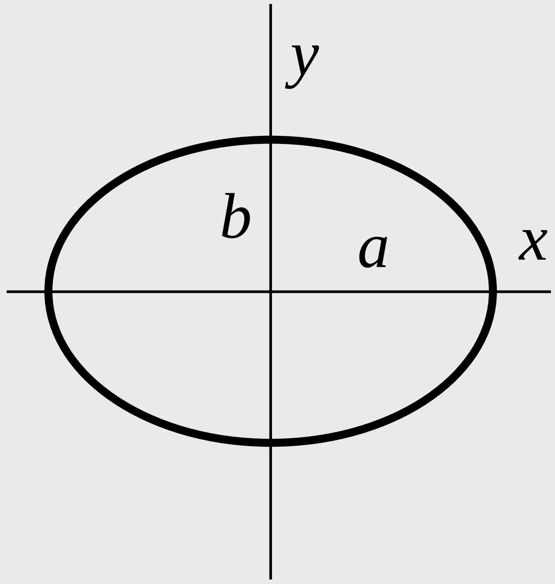
Ellipse

$$\text{Parametric:}$$

$$(x, y) = (a\cos\alpha, b\sin\alpha)$$

$$\text{Implicit:}$$

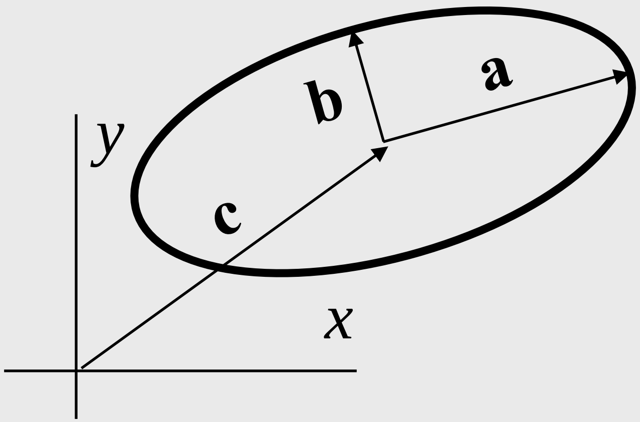
$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 - 1 = 0$$



Generic ellipse

Parametric:

$$\mathbf{x}(\alpha) = \mathbf{c} + \mathbf{a}\cos\alpha + \mathbf{b}\sin\alpha$$



Implicit:

$$|M\mathbf{x}| = 1, \text{ with } M = \begin{pmatrix}\mathbf{a} & \mathbf{b} & \mathbf{c}\end{pmatrix}^{-1}$$

Some standard puzzles

- Conversion of line representation
- Projection of point on line
- Line/Line intersection
- Position points/line
- Line/Circle intersection

Conversion line representations

Given line :

$$\mathbf{p}(s) = \mathbf{a} + \mathbf{u}s;$$

Find implicit representation :

$$\mathbf{n} \cdot \mathbf{x} + c = 0.$$

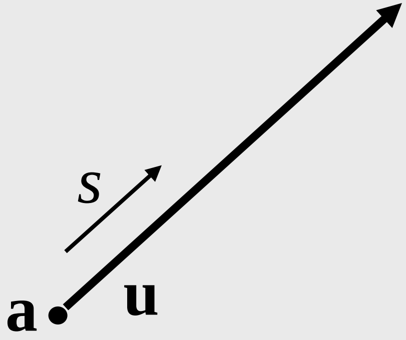
First, determine normal $\mathbf{n}$.

$\mathbf{n}$ must be $\perp$ on $\mathbf{u}$, hence we set :

$$\mathbf{n} = (-u_y, u_x)$$

$\mathbf{a}$ must be on the line, hence :

$$c = -\mathbf{n} \cdot \mathbf{a}$$



Projection point on line

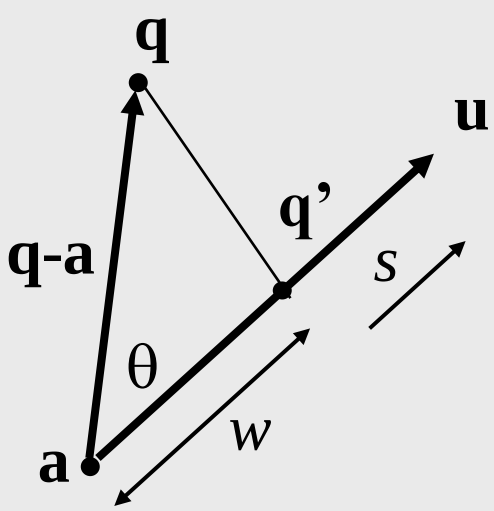
Project point $\mathbf{q}$ on line $\mathbf{p}(s) = \mathbf{a} + \mathbf{u}s$:

$$\mathbf{q}' = \mathbf{a} + \cos\theta \, |\mathbf{q} - \mathbf{a}| \frac{\mathbf{u}}{|\mathbf{u}|}$$

Use $(\mathbf{q} - \mathbf{a}) \cdot \mathbf{u} = |\mathbf{q} - \mathbf{a}| |\mathbf{u}| \cos\theta$:

$$\mathbf{q}' = \mathbf{a} + \frac{(\mathbf{q} - \mathbf{a}) \cdot \mathbf{u}}{|\mathbf{u}||\mathbf{u}|}\mathbf{u}, \quad \text{or}$$

$$\mathbf{q}' = \mathbf{a} + \frac{(\mathbf{q} - \mathbf{a}) \cdot \mathbf{u}}{\mathbf{u} \cdot \mathbf{u}}\mathbf{u}$$



$\cos\theta \, |\mathbf{q} - \mathbf{a}|$: length w

$\frac{\mathbf{u}}{|\mathbf{u}|}$: unit vector along $\mathbf{u}$

Intersection of line segments

Find intersection of line segments :

$$\mathbf{p}(s) = \mathbf{a} + \mathbf{u}s,\ \ 0 \le \mathrm{s} \le 1 \text{ and}$$

$$\mathbf{q}(t) = \mathbf{b} + \mathbf{v}t,\ \ 0 \le t \le 1.$$

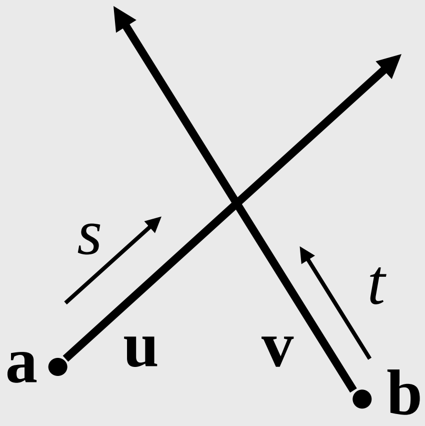
At intersection :

$$\mathbf{p}(s) = \mathbf{q}(t)$$

Solve for s and t (next sheet);

Check if $0 \le s \le 1$ and $0 \le t \le 1$;

If so, intersection is $\mathbf{p}(s)$.



Solving for s and t

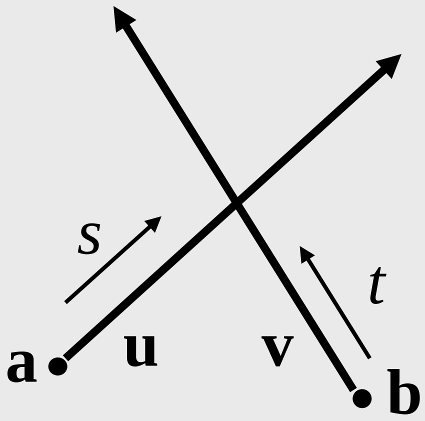
$$\mathbf{p}(s) = \mathbf{q}(t), \text{ or}$$

$$\mathbf{a} + \mathbf{u}s = \mathbf{b} + \mathbf{v}t, \text{ or}$$

$$\begin{pmatrix}\mathbf{u} & \mathbf{v}\end{pmatrix}\begin{pmatrix} s \\ t \end{pmatrix} = \mathbf{b} - \mathbf{a}, \text{ or}$$

$$\begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix}\mathbf{u} & \mathbf{v}\end{pmatrix}^{-1}(\mathbf{b} - \mathbf{a}), \text{ or}$$

$$\begin{pmatrix} s \\ t \end{pmatrix} = \frac{1}{u_x v_y - u_y v_x}\begin{pmatrix} v_y & -v_x \\ -u_y & u_x \end{pmatrix}\begin{pmatrix} b_x - a_x \\ b_y - a_y \end{pmatrix}$$

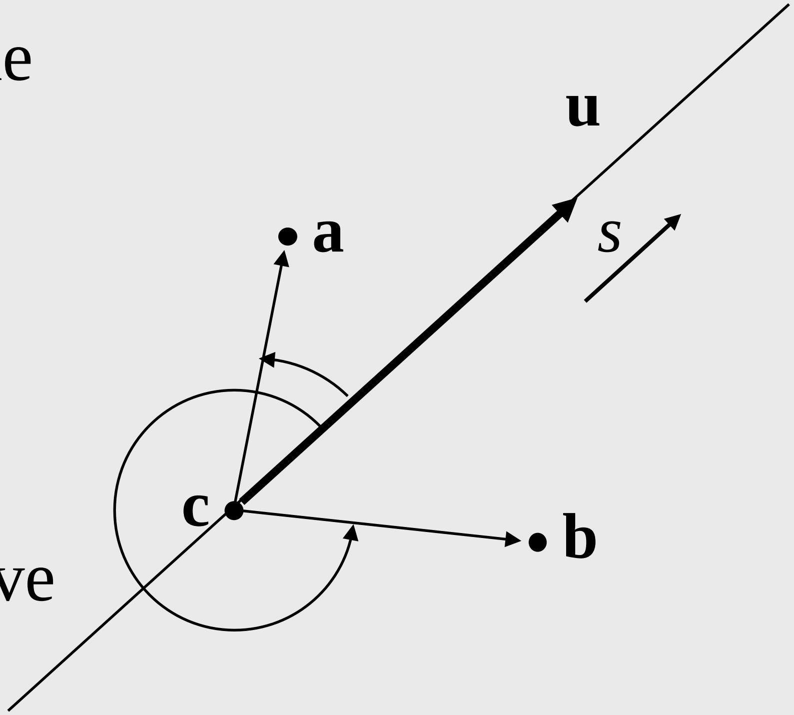


Position points/line

Check if points $\mathbf{a}$ and $\mathbf{b}$ are on the same side of line $\mathbf{p}(s) = \mathbf{c} + \mathbf{u}s$

Use $Det(\mathbf{u}, \mathbf{v}) = |\mathbf{u}||\mathbf{v}|\sin\theta$:

Points are on the same side if $Det(\mathbf{u}, \mathbf{a} - \mathbf{c})$ and $Det(\mathbf{u}, \mathbf{b} - \mathbf{c})$ have the same sign.



Line/circle intersection

Find intersections of :

line : $\mathbf{p}(t) = \mathbf{a} + \mathbf{u}t,\ 0 \le \mathrm{t} \le 1$ and

circle : $\mathbf{x} \cdot \mathbf{x} = r^2.$

At intersection :

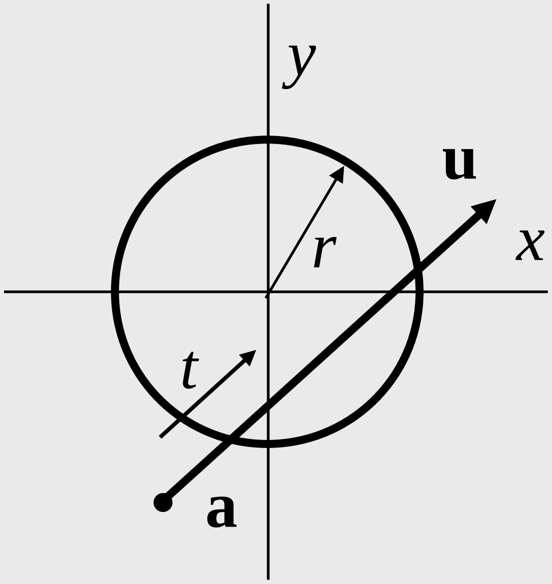
$$\mathbf{p}(t) \cdot \mathbf{p}(t) = r^2, \text{ or}$$

$$(\mathbf{a} + \mathbf{u}t) \cdot (\mathbf{a} + \mathbf{u}t) = r^2, \text{ or}$$

$$\mathbf{u} \cdot \mathbf{u}t^2 + \mathbf{a} \cdot \mathbf{u}t + \mathbf{a} \cdot \mathbf{a} - r^2 = 0.$$

Solve quadratic equation for t :

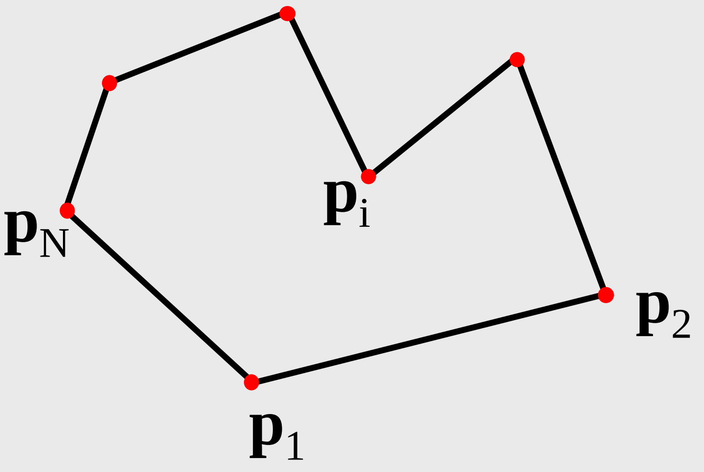
0, 1, or 2 solutions.



Polygons

- Sequence of points $\mathbf{p_i}$, $i = 1, \ldots, N$, connected by straight lines
- Index arithmetic: modulo N

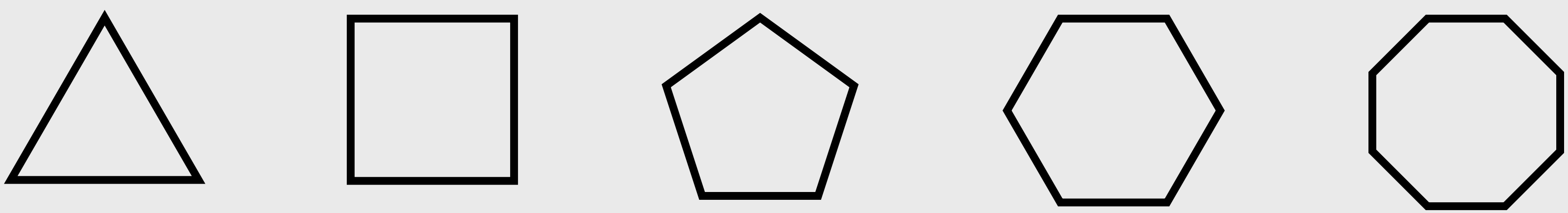
$\mathbf{p_0} = \mathbf{p_N}$, $\mathbf{p_{N+1}} = \mathbf{p_1}$, etc.



Regular N-gon

$$\mathbf{p}_i = (r\cos\alpha_i, r\sin\alpha_i)$$

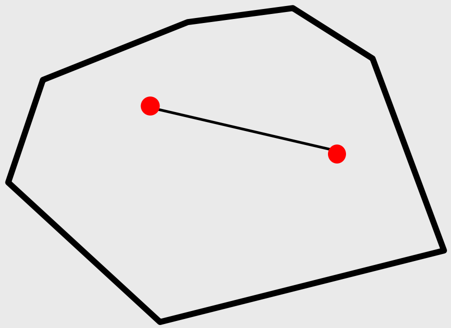
$$\alpha_i = 2\pi(i+1/2)/N - \pi/2$$



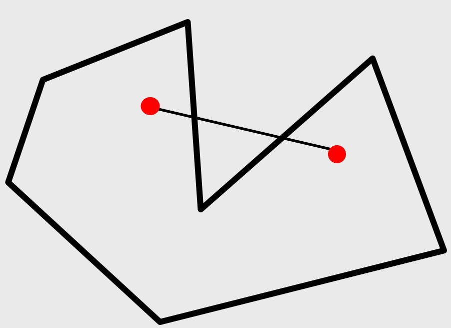
triangle square pentagon hexagon octagon

Convex and concave

- Convex:
  - each line between two arbitrary points inside the polygon does not cross its boundary



- Concave:
  - not convex

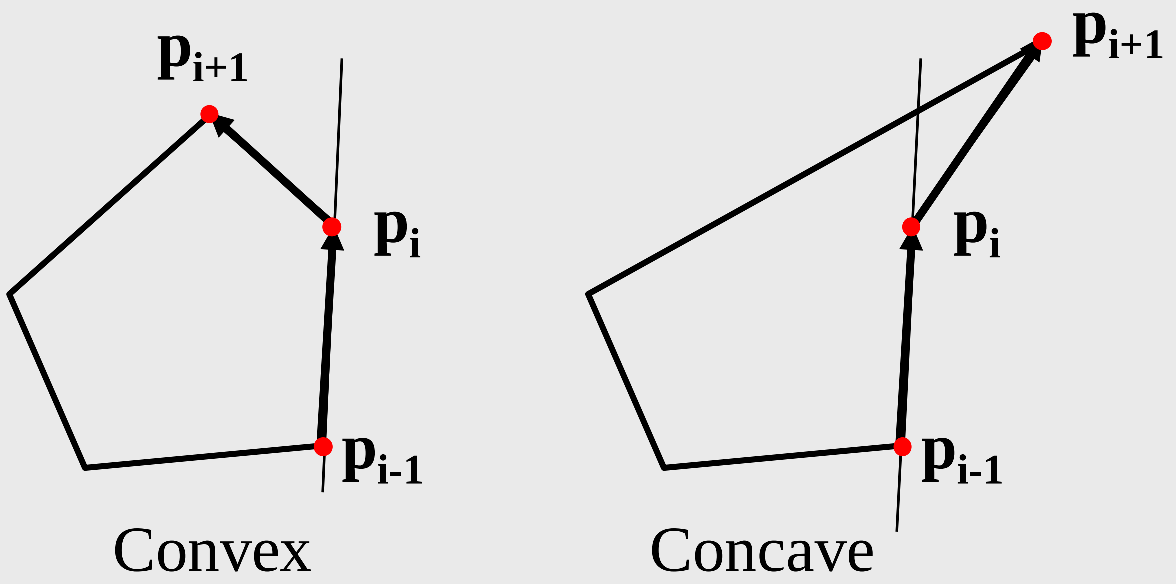


Convexity test

Assume polygon is oriented counterclockwise.

Polygon is concave, if

$\mathrm{Det}(\mathbf{p}_{\mathrm{i}} - \mathbf{p}_{\mathrm{i\text{-}1}}, \mathbf{p}_{\mathrm{i+1}} - \mathbf{p}_{\mathrm{i}}) > 0$ for all i



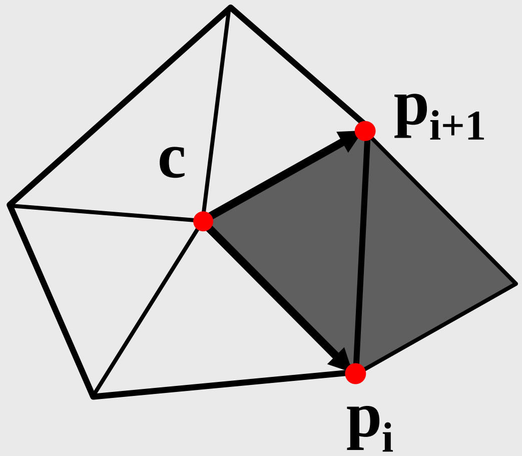
Polygon area and orientation

$$a = \sum_{\mathrm{i}}^{N} Det(\mathbf{p}_{\mathrm{i}} - \mathbf{c}, \mathbf{p}_{\mathrm{i+1}} - \mathbf{c}) / 2, \quad \mathbf{c} \text{ is } \textit{arbitrary} \text{ point}$$

$$area = |\, a \,|$$

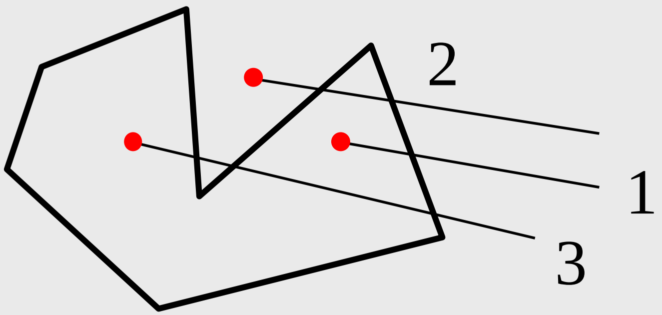
$$a > 0 : \text{counterclockwise orientation}$$

$$a < 0 : \text{clockwise orientation}$$



Point/polygon test

Given a polygon. Test if a point $\mathbf{c}$ is inside or outside.



Solution :

Define a line $L = \mathbf{c} + \mathbf{v}t, t \geq 0$.

$\mathbf{v}$ can be chosen arbitrarily, f.i. $(1, 0)$.

Let n be the number of crossings of L with the polygon. If n is odd : point is inside, else it is outside.

Point/polygon test (cntd.)

- Beware of special cases:
  - Point at boundary
  - $\mathbf{v}$ parallel to edge
  - $\mathbf{c} + \mathbf{v}t$ through vertex

Puzzles

- Define a procedure to clip a line segment against a rectangle.
- Define a procedure to calculate the intersection of two polygons.
- Define a procedure to draw a star.
- Same, with the constraint that the edges $\mathbf{p}_{i-1}\,\mathbf{p}_i$ and $\mathbf{p}_{i+2}\,\mathbf{p}_{i+3}$ are parallel.