



Chapter 1 Elementary Concepts

- Lines and Coordinates
- Device Coordinates
- Logical Coordinates
- Converting Between Logical and Device Coordinates
- Mapping From Logical to Device Coordinates
 - Anisotropic mapping
 - Isotropic mapping

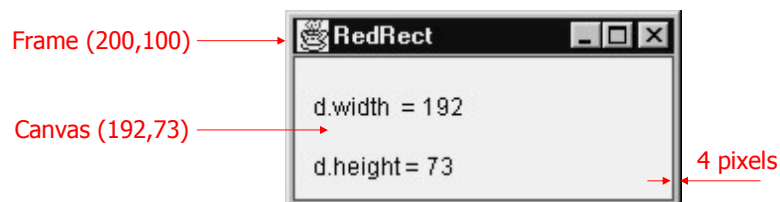
©2006 Wiley & Sons

1



Lines and Coordinates

- In Java, to draw a line
 - `g.drawLine(xA, yA, xB, yB)`
 - Same as `g.drawLine(xB, yB, xA, yA)`
- E.g. to draw the largest possible rectangle on a canvas:



©2006 Wiley & Sons

2



Example: Red Rectangle

```
.....  
class CvRedRect extends Canvas  
{ public void paint(Graphics g)  
  { Dimension d = getSize();  
    int maxX = d.width - 1, maxY = d.height - 1;  
    g.drawString("d.width = " + d.width, 10, 30);  
    g.drawString("d.height = " + d.height, 10, 60);  
    g.setColor(Color.red);  
    g.drawLine(0, 0, maxX, 0);           // Top edge  
    g.drawLine(maxX, 0, maxX, maxY);     // Right edge  
    g.drawLine(maxX, maxY, 0, maxY);     // Bottom edge  
    g.drawLine(0, maxY, 0, 0);          // Left edge  
  }  
}
```

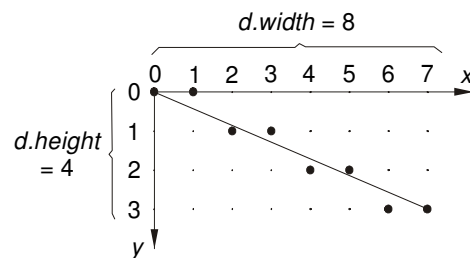
`g.drawRect(0, 0, maxX, maxY);`

©2006 Wiley & Sons

3



Device Coordinate System



- The rectangle size drawn by
 - `g.drawLine(x, y, maxX, maxY)` is **(maxX + 1) by maxY + 1)**
- The smallest rectangle is a square 2 x 2, using
 - `g.drawRect(x, y, 1, 1)`
- To draw one dot
 - `g.drawLine(x, y, x, y)`

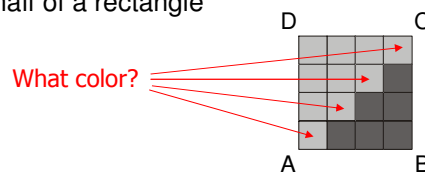
©2006 Wiley & Sons

4



Device Coordinate System (cont'd)

- To fill a rectangle of size $w \times h$, use
 - `g.fillRect(x, y,w,h)`
- `g.drawRect(x, y,w,h)` draws a slightly bigger rectangle than by `g.fillRect(x, y,w,h)`
- Filling problem: discrete nature
 - e.g. to fill a triangle half of a rectangle



©2006 Wiley & Sons

5



Logical Coordinate System

- To put origin at the left-bottom corner as in math:
 - $y' = \text{maxY} - y$ (x -axis is unchanged)

Coordinate System	Convention	Data Type	Value Domain	Positive y -axis
Logical	Lower-case letters	float	Continuous	Upward
Device	Upper-case letters	integer	Discrete	Downward

©2006 Wiley & Sons

6

Converting Between Logical and Device Coordinates

- Rounding:
 - `Int iX(float x){return Math.round(x);} // L -> D`
 - `Float fx(int x){return (float)x;} // D -> L`
 - E.g. $iX(2.8) = 3$, $fx(3) = 3.0$
- Truncating:
 - `Int iX(float x){return (int)x;} // L -> D`
 - `Float fx(int x){return (float)x+0.5;} // D -> L`
 - E.g. $iX(2.8) = 2$, $fx(2) = 2.5$
- For both above, $|x - fx(iX(x))| \leq 0.5$
 - Max lost precision is 0.5
- Rounding will be used throughout this book

©2006 Wiley & Sons

7

An Important Principle in Conversion

Step $i+1$ computation is performed on FP result of Step i :

```

float  →  float  →  float  →  float
 ↓       ↓       ↓       ↓
int     int     int     int
  
```

Rather than (accumulating rounding off errors):

```

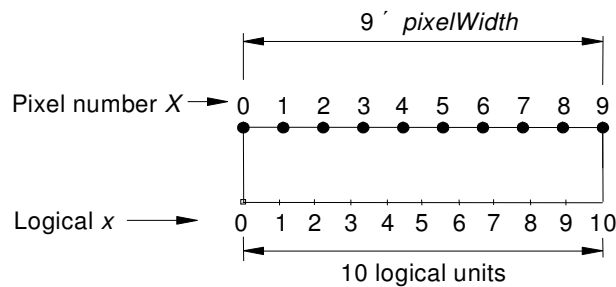
float  ↗      float  ↗      float  ↗      float
 ↓       ↓       ↓       ↓
int     int     int     int
  
```

©2006 Wiley & Sons

8

Mapping From Logical to Device Coordinates

- Can we use `int iX(float x){return Math.round(x);}` to map from logical coordinates 0.0 – 10.0 (real) to device coordinates 0 – 9 (integer)?



©2006 Wiley & Sons

9

Mapping From Logical to Device Coordinates (cont'd)

- So pixel width in terms of logical coordinates is $10/9 = 1.11$.
- Enhanced method is
 - `int iX(float x){return Math.round(x/pixelWidth);}`
- In this example, 9 = number of pixels (10) – 1
 - 10 is width of logical interval, i.e. $0 \leq x \leq rWidth$
- It works similarly with vertical (y) coordinates (rHeight).

©2006 Wiley & Sons

10



Anisotropic/Isotropic Mapping

- Anisotropic mapping: $\text{pixelWidth} \neq \text{pixelHeight}$
 - Unsuitable for shapes like squares and circles
- Isotropic mapping: $\text{pixelWidth} = \text{pixelHeight}$ (pixelSize)
 - `int iX(float){return Math.round(x/pixelWidth);}`
- Usually we want the origin of the logical coordinates to be in the center, so
 - $-0.5 \text{ rWidth} \leq x < 0.5 \text{ rWidth}$
 - $-0.5 \text{ rHeight} \leq y < 0.5 \text{ rHeight}$
- Mapped to device coordinates $0 - \text{maxX}$ and $0 - \text{maxY}$.

©2006 Wiley & Sons

11



Chapter 2 Applied Geometry

- Vectors
- Dot (inner) Product
- Vector (cross) Product
- Determinants
- Orientation of 3 points
- Polygon Shapes
- Point-in-Polygon Test
- Point and Line Relationships
- Triangulation of Polygons

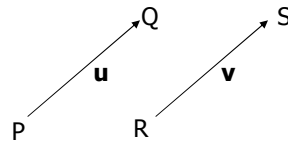
©2006 Wiley & Sons

12



Vectors

- Vector = length + direction of a line segment
 - Useful representation of real-world measurements, e.g. velocity.
 - Not to be confused with Java vectors
 - Length of $\mathbf{u} = |\mathbf{u}|$
 - $-\mathbf{u}$ has the same length but opposite direction



$$\mathbf{u} = \mathbf{PQ} = \mathbf{v} = \mathbf{RS}$$

A translation gives the same vector

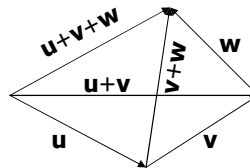
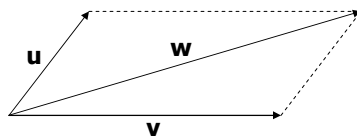
©2006 Wiley & Sons

13



Vector Addition

- $\mathbf{w} = \mathbf{u} + \mathbf{v}$ is the diagonal of the parallelogram formed by \mathbf{u} and \mathbf{v}



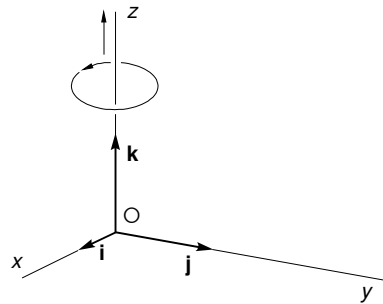
©2006 Wiley & Sons

14



Multiplying a vector with a real number

- C being a real number, the length of vector $c\mathbf{u}$ is $|c||\mathbf{u}|$
- A vector of unit length is called a *unit vector*
- Right-handed coordinate systems for 3D



©2006 Wiley & Sons

15



Vector (cont'd)

- Linear combination of \mathbf{i} , \mathbf{j} and \mathbf{k} :
 - $\mathbf{v} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} = \mathbf{OP}$
 - x , y , and z are coordinates of P and called *elements* or *components* of \mathbf{v} , or simply
 - $\mathbf{v} = [x \ y \ z]$

©2006 Wiley & Sons

16



Dot Product (Inner Product)

- Dot product of \mathbf{u} and \mathbf{v} , i.e. $\mathbf{u} \cdot \mathbf{v}$, is a real number

$$\mathbf{u} \cdot \mathbf{v} = \begin{cases} |\mathbf{u}| |\mathbf{v}| \cos \theta & \text{if } \mathbf{u} \neq 0 \text{ and } \mathbf{v} \neq 0 \\ 0 & \text{if } \mathbf{u} = 0 \text{ or } \mathbf{v} = 0 \end{cases}$$

(θ is the angle between \mathbf{u} and \mathbf{v})

- For unit vectors \mathbf{i} , \mathbf{j} and \mathbf{k} :
 - $\mathbf{i} \cdot \mathbf{i} = \mathbf{j} \cdot \mathbf{j} = \mathbf{k} \cdot \mathbf{k} = 1$
 - $\mathbf{i} \cdot \mathbf{j} = \mathbf{i} \cdot \mathbf{k} = \mathbf{j} \cdot \mathbf{k} = \mathbf{k} \cdot \mathbf{j} = \mathbf{j} \cdot \mathbf{i} = \mathbf{k} \cdot \mathbf{i} = 0$

©2006 Wiley & Sons

17



Dot Product (cont'd)

- Since $\mathbf{u} \cdot \mathbf{u} = |\mathbf{u}|^2$, $|\mathbf{u}| = \sqrt{|\mathbf{u} \cdot \mathbf{u}|}$
- Properties of dot product:
 - $c(\mathbf{u} \cdot \mathbf{v}) = c\mathbf{u} \cdot \mathbf{v}$
 - $(c\mathbf{u} + k\mathbf{v}) \cdot \mathbf{w} = c\mathbf{u} \cdot \mathbf{w} + k\mathbf{v} \cdot \mathbf{w}$
 - $\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{u}$
 - $\mathbf{u} \cdot \mathbf{u} = 0$ only if $\mathbf{u} = 0$
- Dot product of $\mathbf{u} = [u_x \ u_y \ u_z]$ and $\mathbf{v} = [v_x \ v_y \ v_z]$ is
 - $\mathbf{u} \cdot \mathbf{v} = u_x v_x + u_y v_y + u_z v_z$

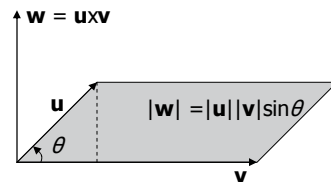
©2006 Wiley & Sons

18



Vector Product (Cross Product)

- Vector product of \mathbf{u} and \mathbf{v} , i.e. $\mathbf{u} \times \mathbf{v}$, is a vector \mathbf{w}
- $\mathbf{w} = 0$ if $\mathbf{u} = c\mathbf{v}$, otherwise $|\mathbf{w}| = |\mathbf{u}||\mathbf{v}|\sin\theta$
(θ is the angle between \mathbf{u} and \mathbf{v})
- $|\mathbf{w}|$ is the area size of the parallelogram formed by \mathbf{u} and \mathbf{v}
- Direction: right-handed screw rule



©2006 Wiley & Sons

19



Vector Product (cont'ed)

- Properties:
 - $(k\mathbf{u}) \times \mathbf{v} = k(\mathbf{u} \times \mathbf{v})$
 - $\mathbf{u} \times (\mathbf{v} + \mathbf{w}) = \mathbf{u} \times \mathbf{v} + \mathbf{u} \times \mathbf{w}$
 - $\mathbf{u} \times \mathbf{v} = -\mathbf{v} \times \mathbf{u}$
 - $\mathbf{i} \times \mathbf{i} = \mathbf{j} \times \mathbf{j} = \mathbf{k} \times \mathbf{k} = 0$
 - $\mathbf{i} \times \mathbf{j} = \mathbf{k}$ $\mathbf{j} \times \mathbf{i} = -\mathbf{k}$
 - $\mathbf{j} \times \mathbf{k} = \mathbf{i}$ $\mathbf{k} \times \mathbf{j} = -\mathbf{i}$
 - $\mathbf{k} \times \mathbf{i} = \mathbf{j}$ $\mathbf{i} \times \mathbf{k} = -\mathbf{j}$

©2006 Wiley & Sons

20



Vector Product (cont'd)

- $\mathbf{u} \times \mathbf{v} = (u_1\mathbf{i} + u_2\mathbf{j} + u_3\mathbf{k}) \times (v_1\mathbf{i} + v_2\mathbf{j} + v_3\mathbf{k})$
 $= u_1v_1(\mathbf{i} \times \mathbf{i}) \dots\dots$

$$\mathbf{u} \times \mathbf{v} = \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix} \mathbf{i} + \begin{vmatrix} u_3 & u_1 \\ v_3 & v_1 \end{vmatrix} \mathbf{j} + \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix} \mathbf{k}$$

$$\mathbf{u} \times \mathbf{v} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix}$$

©2006 Wiley & Sons

21



Determinants

- To solve two linear equations

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$$

we multiply the 1st equation by b_2 , 2nd by $-b_1$, and then add them up to cancel y (similarly cancel x), obtain (if $(a_1b_2 - a_2b_1) \neq 0$):

$$x = \frac{b_2c_1 - b_1c_2}{a_1b_2 - a_2b_1} \qquad y = \frac{a_1c_2 - a_2c_1}{a_1b_2 - a_2b_1}$$

©2006 Wiley & Sons

22



Determinants (cont'd)

- We can write

$$x = \frac{D_1}{D} \quad y = \frac{D_2}{D} \quad (D \neq 0)$$

$$D = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} \quad D_1 = \begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix} \quad D_2 = \begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}$$



Determinants (cont'd)

- Properties

$$(1) \quad \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} = \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix}$$

$$(2) \quad \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} = - \begin{vmatrix} a_1 & b_1 & c_1 \\ a_3 & b_3 & c_3 \\ a_2 & b_2 & c_2 \end{vmatrix}$$

$$(3) \quad \begin{vmatrix} ca_1 & b_1 \\ ca_2 & b_2 \end{vmatrix} = c \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix}$$



Determinants (cont'd)

■ Properties

$$(4) \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 + ka_1 & b_3 + kb_1 & c_3 + kc_1 \end{vmatrix} = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}$$

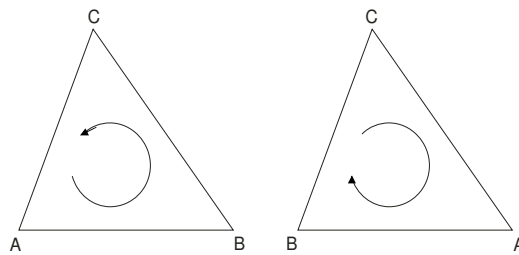
$$(5) \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 3a_1 - 2a_2 & 3b_1 - 2b_2 & 3c_1 - 2c_2 \end{vmatrix} = 0$$

©2006 Wiley & Sons

25



Orientation of 3 Points



Orientation > 0

Orientation < 0

Orientation = 0 when A, B, and C are on a straight line

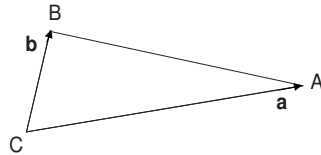
©2006 Wiley & Sons

26



Orientation of 3 Points (cont'd)

- 3D vector representation



$$\mathbf{a} = \text{CA}$$
$$\mathbf{b} = \text{CB}$$

$$\mathbf{a} = a_1\mathbf{i} + a_2\mathbf{j} + 0\mathbf{k}$$
$$\mathbf{b} = b_1\mathbf{i} + b_2\mathbf{j} + 0\mathbf{k}$$

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & 0 \\ b_1 & b_2 & 0 \end{vmatrix} = \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \mathbf{k} = (a_1b_2 - a_2b_1)\mathbf{k}$$

$$a_1b_2 - a_2b_1 \begin{cases} > 0: \text{ thumb pointing to us, ccw} \\ = 0: \text{ on the same line} \\ < 0: \text{ cw} \end{cases}$$

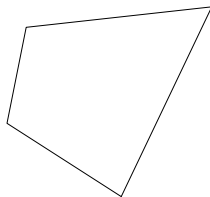
©2006 Wiley & Sons

27



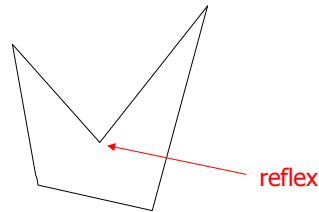
Polygon Shapes

- *Convex vertex*: interior angle $< 180^\circ$
- *Reflex*: interior angle $> 180^\circ$



Convex

(consisting of only convex vertices)



Concave

(at least one reflex)

©2006 Wiley & Sons

28



Determining a Polygon's Orientation

Given a polygon P,

- Find a convex vertex in P by finding a vertex with least x or y coordinate
- Identify the triangle constructed by the found convex vertex and its two neighboring vertices
- Determine the order of the vertex sequence

©2006 Wiley & Sons

29



A Useful Java Method

```
class Tools2D
{ static float area2(Point2D a, Point2D b, Point2D c)
  { return (a.x - c.x) * (b.y - c.y) - (a.y - c.y) * (b.x - c.x);
  }
  ... // return Area(ABC)*2: >0 if ccw; <0 if cw
}
```

Because (if ABC is ccw):

$$2 \text{ Area}(\Delta ABC) = |a \times b| = \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} = a_1 b_2 - a_2 b_1$$

©2006 Wiley & Sons

30



Area of a Polygon

If ABC is ccw and $a_1 = xA - xC$, $a_2 = yA - yC$, $b_1 = xB - xC$, $b_2 = yB - yC$

$$\begin{aligned} 2 \text{ Area}(\triangle ABC) &= |a \times b| = \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} = a_1 b_2 - a_2 b_1 \\ &= (xA - xC)(yB - yC) - (yA - yC)(xB - xC) \\ &= (xAyB - yAxB) + (xB yC - yB xC) + (xCyA - yCx A) \end{aligned}$$

which could be generalized to obtain a polygon area:

$$2 \text{ Area}(P_0 \dots P_{n-1}) = \sum_{i=0}^{n-1} (x_i y_{i+1} - y_i x_{i+1})$$

(p_0 and P_n are the same vertex)

©2006 Wiley & Sons

31



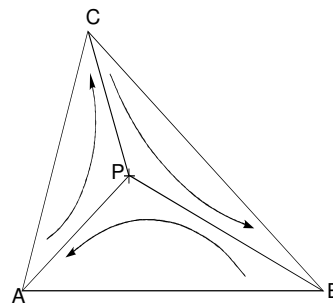
Point-in-Triangle Test

- P is inside a triangle ABC if orientations of ABP, BCP, and CAP are the same as that of ABC.

If ABC is ccw (i.e. $\text{Tools2D.area2}(A, B, C) \geq 0$), and if

$$\begin{aligned} &\text{Tools2D.area2}(A, B, P) \geq 0 \ \&\& \\ &\text{Tools2D.area2}(B, C, P) \geq 0 \ \&\& \\ &\text{Tools2D.area2}(C, A, P) \geq 0 \end{aligned}$$

Then P is inside ABC, or on an edge of it.

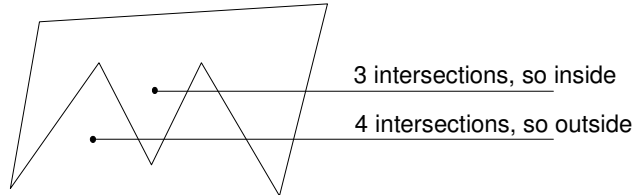


©2006 Wiley & Sons

32



Point-in-Polygon Test



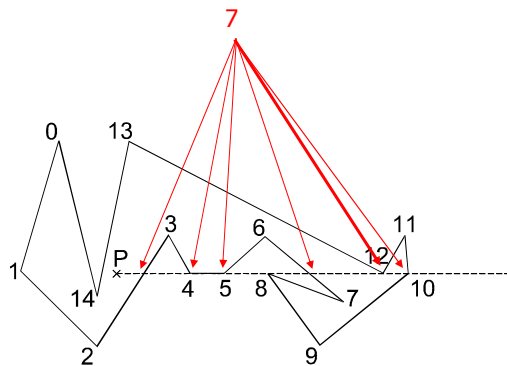
©2006 Wiley & Sons

33



Point-in-Polygon Test

- Special cases
- An edge $(i, i+1)$ is counted if
 - $y_i \leq y_P < y_{i+1}$ or $y_{i+1} \leq y_P < y_i$
(a lower endpoint is counted once for each edge, but upper endpoints are not counted)
- *insidePolygon* (page 42)



©2006 Wiley & Sons

34

- The method can also be used to implement polygon-filling.



Point and Line Relationships

- Testing whether a point P is on a line defined by A and B:
 - if $(\text{Math.abs}(\text{Tools2D.area2}(a, b, p)) < \text{eps})$
 - eps – a small positive number for tolerance
- Testing whether a point P is on a line segment AB:
 - If $x_A \neq x_B$ AND x_P in-between x_A and x_B AND $\text{Math.abs}(\text{Tools2D.area2}(a, b, p)) < \text{eps}$
 - If $x_A == x_B$ AND y_P in-between y_A and y_B AND $\text{Math.abs}(\text{Tools2D.area2}(a, b, p)) < \text{eps}$

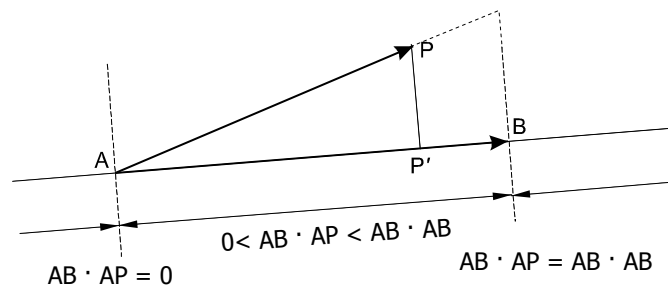
©2006 Wiley & Sons

35



Point and Line Relationships (cont'd)

- Testing whether a point P is projected on a line segment AB :



©2006 Wiley & Sons

36



Triangulation of Polygons

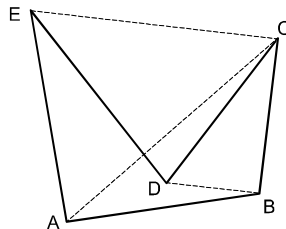
- **Input:** an array of N `Pint2D` objects representing polygon vertices
- **Output:** an array of $N-2$ `Triangle` objects
 1. Traverse polygon vertices in ccw order
 2. Work on every 3 successive vertices A , B and C , if B is a convex vertex, then
 - If ABC does not contain any of other vertices
 - Then cut ABC off the polygon
 3. Continue Step 1 with the remaining polygon

©2006 Wiley & Sons

37



Triangulation of Polygons (cont'd)



- ABC cannot be cut since it contains D
- CDE cannot be cut since D is a reflex
- BCD can be cut to generate new polygon $ABDE$
-

©2006 Wiley & Sons

38



Chapter 3 Geometrical Transformations

- Linear Transformations in 2D
 - Rotation
 - Scaling
 - Shearing
- Translation
- Homogeneous Coordinates
- Inverse Transformations
- Composition of 2D Transformations
- Change in Coordinate System
- 3D Transformations
 - Rotation about a Coordinate Axis
 - Rotation about an Arbitrary Axis

©2006 Wiley & Sons

39



Linear Transformations in 2D

$$\begin{cases} x' = 2x \\ y' = x + y \end{cases} \xrightarrow{\text{math}} \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\xrightarrow{\text{This book}} \begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$$

- Rows of 2 x 2 matrix are the images of unit vectors (1, 0) and (0, 1), i.e.

$$\begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$$

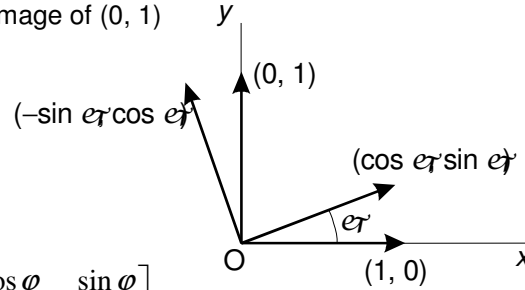
©2006 Wiley & Sons

40



Rotation

- Rotate about O:
 - $(\cos \varphi, \sin \varphi)$ is the image of $(1, 0)$
 - $(-\sin \varphi, \cos \varphi)$ is the image of $(0, 1)$



- Rotate matrix:

$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} \cos \varphi & \sin \varphi \\ -\sin \varphi & \cos \varphi \end{bmatrix}$$

©2006 Wiley & Sons

41



Scaling

- Scale with reference to O:
$$\begin{cases} x' = s_x x \\ y' = s_y y \end{cases}$$

- Scaling matrix:
$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

- Special cases:
 - $s_x = 1, s_y = -1$ reflection about the x -axis
 - $s_x = -1, s_y = 1$ reflection about the y -axis
 - $s_x = s_y = -1$ reflection about O

©2006 Wiley & Sons

42

Shearing

- Along x-axis:

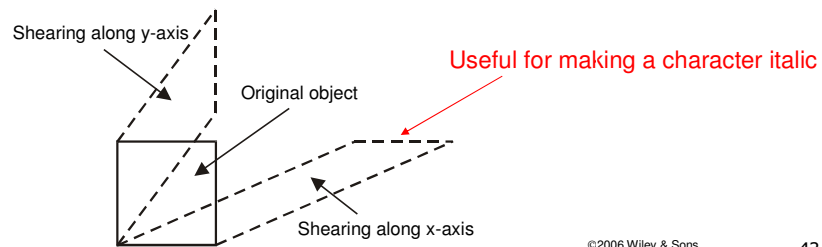
$$\begin{cases} x' = x + ay \\ y' = y \end{cases}$$

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

- Along y-axis:

$$\begin{cases} x' = x \\ y' = bx + y \end{cases}$$

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



Translation

- Shifting all points in a constant distance

$$\begin{cases} x' = x + a \\ y' = y + b \end{cases}$$

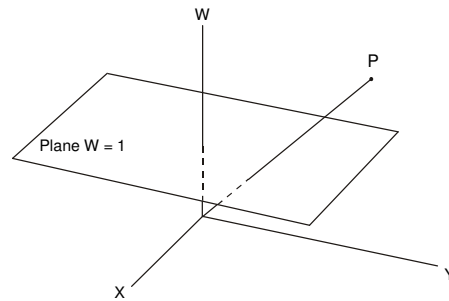
$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} + \begin{bmatrix} a & b \end{bmatrix}$$

- Non-linear since image of origin is (a, b) , rather than $(0, 0)$



Homogeneous Coordinates

- To make all transformations as matrix multiplications in order to combine them, we introduce one more dimension
- Each point has many different homogeneous coordinates
 - E.g. (2, 3, 6) and (4, 6, 12) are the same point – one is multiple of the other



©2006 Wiley & Sons

45



Homogeneous Coordinates (cont'd)

- Translation matrix can then be written as

$$[x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a & b & 1 \end{bmatrix}$$

- All the previous linear transformations can be written as 3 x 3 matrices for them to operate together
 - Rotation:

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

©2006 Wiley & Sons

46



Inverse Transformations

- For rotation matrix R , inverse rotation through $-\alpha$ is

$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} R^{-1}$$

$$\text{where } R^{-1} = \begin{bmatrix} \cos(-\varphi) & \sin(-\varphi) \\ -\sin(-\varphi) & \cos(-\varphi) \end{bmatrix} = \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix}$$

- In general, if matrix A has an inverse A^{-1} , then

$$AA^{-1} = A^{-1}A = I$$

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ called } \textit{identity} \text{ matrix}$$

©2006 Wiley & Sons

47



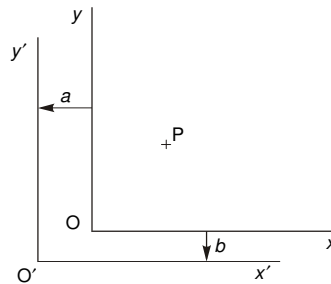
Changing Coordinate System

- An alternative but equivalent way of transformation is to change coordinate system

E.g.

$$x' = x + a$$

$$y' = y + b$$



- Same principle applies to other transformations

©2006 Wiley & Sons

48



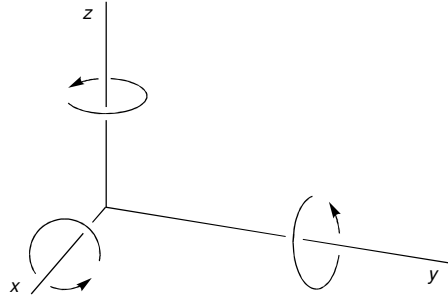
3D Transformations

- Rotation about a coordinate axis

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & \sin \varphi \\ 0 & -\sin \varphi & \cos \varphi \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos \varphi & 0 & -\sin \varphi \\ 0 & 1 & 0 \\ \sin \varphi & 0 & \cos \varphi \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos \varphi & \sin \varphi & 0 \\ -\sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



©2006 Wiley & Sons

49

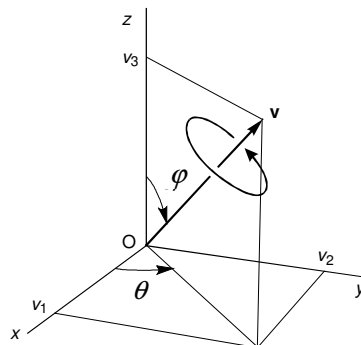


3D Transformations (cont'd)

- Rotation about an arbitrary coordinate axis \mathbf{v}
 1. Rotate about z-axis for φ
 2. Rotate about y-axis for θ

$$R_y^{-1} = \begin{bmatrix} \cos \varphi & 0 & \sin \varphi \\ 0 & 1 & 0 \\ -\sin \varphi & 0 & \cos \varphi \end{bmatrix}$$

$$R_z^{-1} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



©2006 Wiley & Sons

50



3D Transformations (cont'd)

- Rotation about an arbitrary coordinate axis

3. Perform rotation about \mathbf{v} , now same as z-axis

$$R_v = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4. Rotate z-axis back to its original position (inverse of 1 and 2)

$$R_y = \begin{bmatrix} \cos \varphi & 0 & -\sin \varphi \\ 0 & 1 & 0 \\ \sin \varphi & 0 & \cos \varphi \end{bmatrix} \quad R_z = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Combined transformations: $R = R_z^{-1} R_y^{-1} R_v R_y R_z$

©2006 Wiley & Sons

51



3D Transformations (cont'd)

- Rotation about an arbitrary coordinate axis \mathbf{v} from any point $A(a_1, a_2, a_3)$

- Translation involved, so use homogeneous coordinates

1. Translate from A to O
2. Apply R in homo. coordinates, R^*
3. Translate from O back to A

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -a_1 & -a_2 & -a_3 & 1 \end{bmatrix} \quad R^* = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a_1 & a_2 & a_3 & 1 \end{bmatrix}$$

So, generalized rotation matrix $R_{GEN} = T^{-1} R^* T$

(for efficiency, R_{GEN} is calculated beforehand, then many points could be rotated using R_{GEN})

©2006 Wiley & Sons

52