Suranaree University
of Technology

COMPUTER GRAPHICS
School of Computer Engineering

Lecture 4 Review

Paramate Horkaew

School of Computer Engineering, Institute of Engineering
Suranaree University of Technology

Previous Lecture

- Color and Grayscale Levels
- Polygon Filling Styles
  - Pattern Tilling
  - Pattern Blending: Multiple Transparent Layers
- Anti-aliasing
  - Super-sampling Technique
  - Filtering Techniques
  - Anti-aliasing of Areas
- 2D Geometric Transformation
  - Basic Transformations
  - Matrix Representation and Homogeneous Coordinates
  - Composite Transformations
  - Other Transformations, *e.g.* Affine
  - Raster Methods for Transformations

Suranaree University of Technology

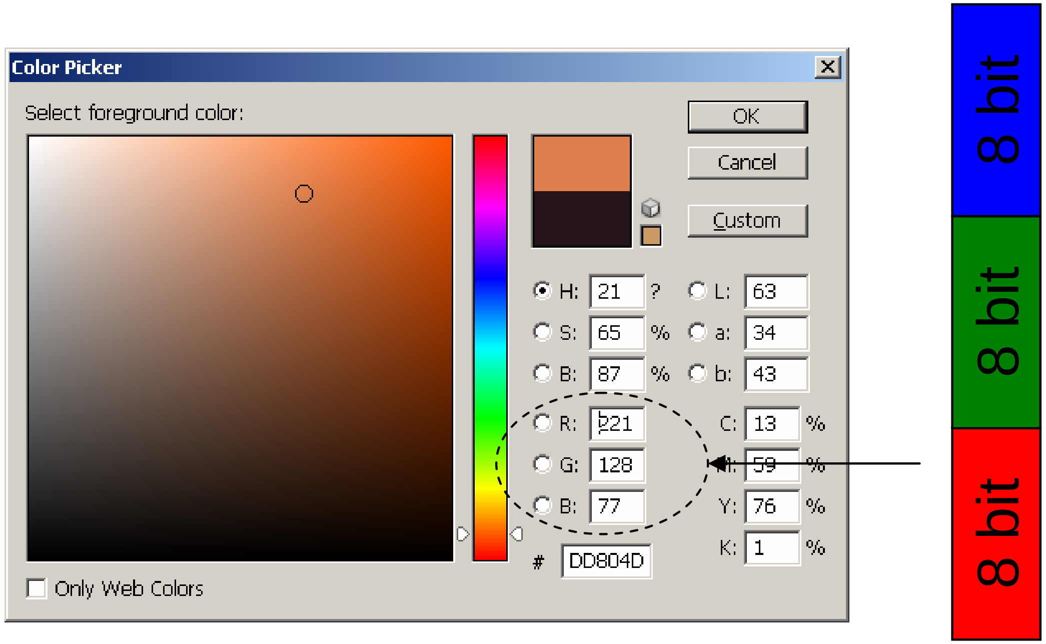
Color Representations

ในที่นี้ สีที่แสดงบนอุปกรณ์แสดงผล จะถูก เข้ารหัส (numerically coded) ด้วยตัวเลขจำนวนเต็มที่มากกว่า 0 ซึ่งสำหรับจอ CRT ตัวเลขเหล่านี้จะเป็นตัวกำหนดระดับความเข้ม ของลำแสง electron

จำนวนของสี ขึ้นอยู่กับขนาดของแต่ละ จุดภาพใน Frame Buffer (หรือ Pixel Register) ซึ่งสามารถกำหนดได้ 2 วิธี ได้แก่ Direct และ Table Storage

TABLE 4-1
THE EIGHT COLOR CODES FOR A THREE-BIT PER PIXEL FRAME BUFFER

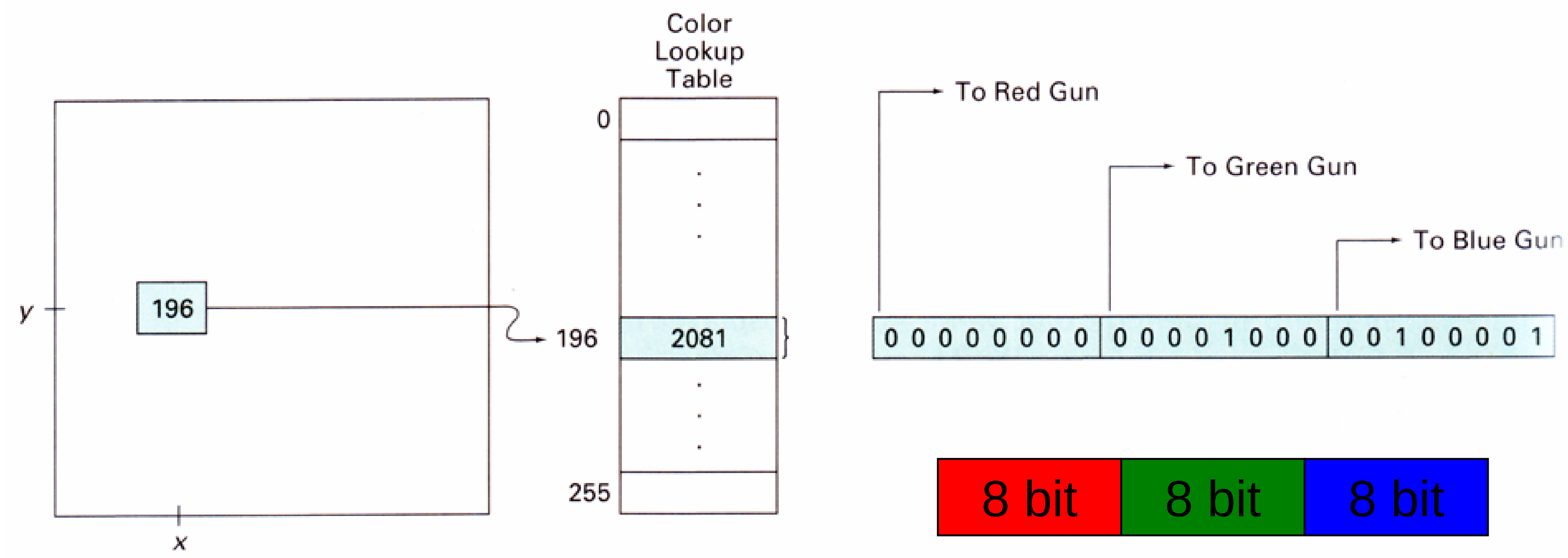
Color Code	*Stored Color Values in Frame Buffer* RED	GREEN	BLUE	*Displayed Color*
0	0	0	0	Black
1	0	0	1	Blue
2	0	1	0	Green
3	0	1	1	Cyan
4	1	0	0	Red
5	1	0	1	Magenta
6	1	1	0	Yellow
7	1	1	1	White



Color Lookup Tables (LUT)

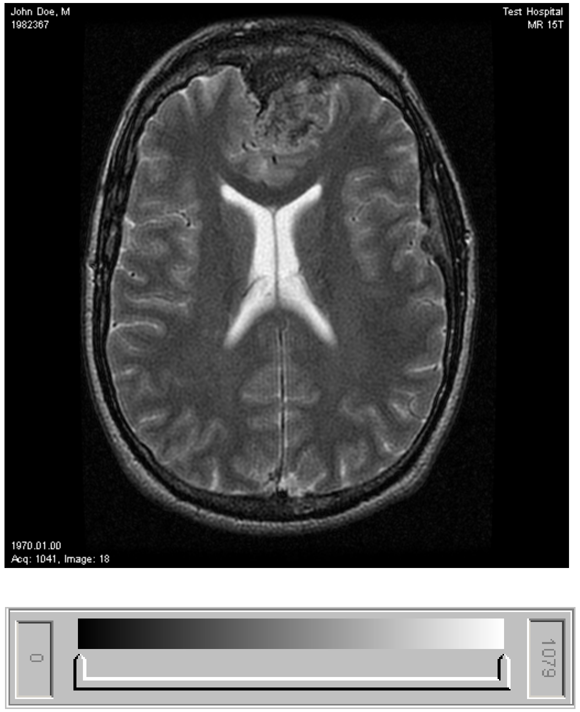
ในการกำหนดสีแบบ Direct จะใช้หน่วยความจำขนาดใหญ่ หากต้องการแสดงภาพสี ขนาด 1024x1024 แบบ Full Color (24 bpp, 8 bpc) จะใช้หน่วยความจำขนาด 3 Megabytes เพื่อจัดเก็บ Frame Buffer

อีกทางเลือกหนึ่งที่ประหยัดกว่าคือ ให้แต่ละจุดภาพใน Frame Buffer เก็บเฉพาะดัชนี ซึ่งชี้ไปยังตารางสี ดังรูป



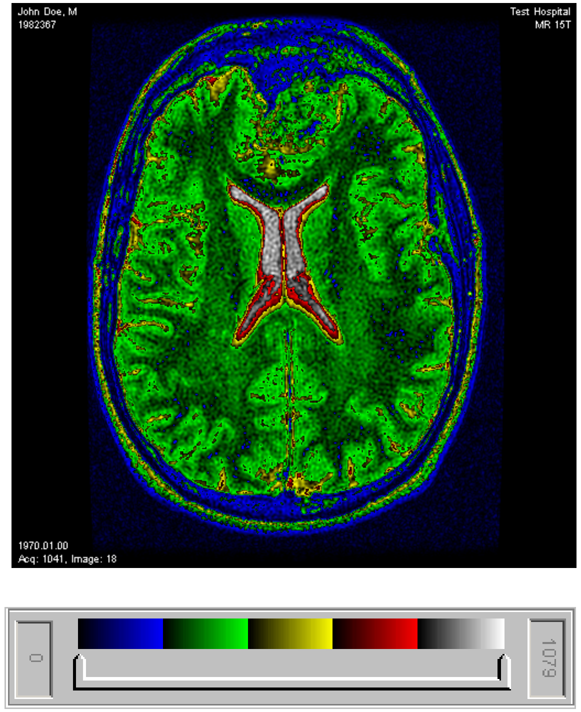
Color LUT and Grayscale

โปรแกรมประยุกต์ ได้นำเทคนิคการแสดงผลภาพกราฟิกแบบสี ด้วยการเปิดตาราง ตัวอย่างเช่น เพื่อการวินิจฉัยทางการแพทย์ (Osiris 4.18) ดังรูป



Grayscale

Flow LUT

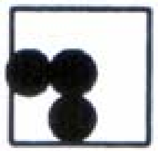


5-Ramp LUT

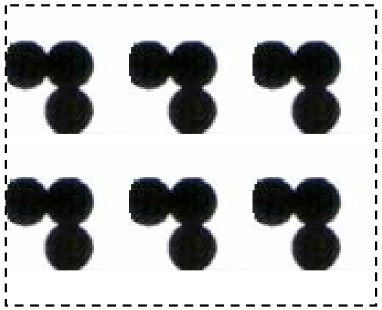
Polygon Filling Styles

เราสามารถระบาย Polygon ใดๆ ด้วยรูปแบบ (Pattern) ชนิดต่างๆ ได้ นอกจากการระบายด้วยสีทึบ โดยดัดแปลง Filling Algorithm ดังต่อไปนี้

สำหรับแต่ละจุด ที่ต้องการระบายสี ให้แทนที่ด้วยรูปแบบที่กำหนด (ในรูปของ mask – array 2 มิติ) การเลื่อนไปตามแกน x และ y ให้เลื่อนไปเท่ากับขนาดความกว้างและความยาว ของ mask นั้นๆ



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



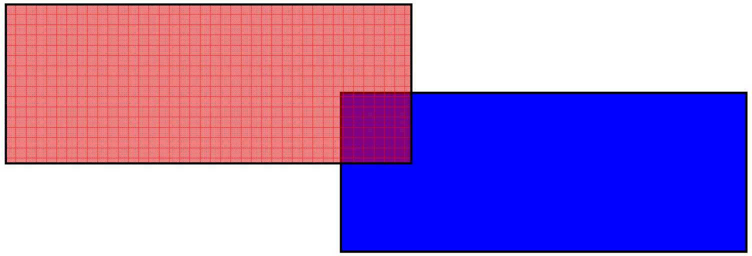
$$y+3 \rightarrow y$$
$$x+3 \rightarrow x$$

หมายเหตุ

1. เราสามารถกำหนดจุดเริ่มต้นของ mask ที่มุมบนสุดของจอภาพ หรือมุมบนสุดของ polygon ก็ได้
2. หากบางส่วนของ mask ตกออกนอกขอบ polygon ก็ให้ละเว้นส่วนนั้น ($mask_i$ AND $pixel_i$)

Soft/Tint Pattern Filling

สำหรับภาพที่แสดงผลด้วยหลายระดับ เราสามารถผสม (Blend) สี (หรือระดับความเทา) ระหว่าง polygon (ตั้งแต่ 1 ชิ้นขึ้นไป) และหรือ background ได้ โดยพิจารณา ให้วัตถุนั้นๆ เสมือนว่าเป็นวัตถุโปร่งใส (Transparent Object)



$$\mathbf{P} = t\mathbf{F} + (1-t)\mathbf{B}$$

$$t = \frac{P_{r,g,b} - B_{r,g,b}}{F_{r,g,b} - B_{r,g,b}}$$

กำหนดให้

- Background (หรือสีพื้นเดิมบน Frame Buffer) นิยามด้วย vector **B** ของ องค์ประกอบ RGB
- Foreground ของวัตถุ ที่นำมาซ้อน นิยามด้วย vector **F**
- สีผสม **P** นิยามด้วยความสัมพันธ์

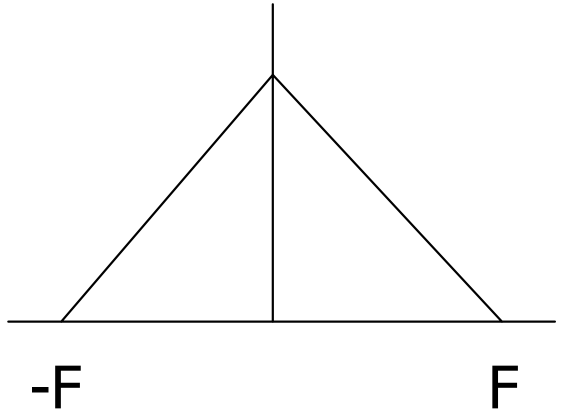
โดยที่ t เป็นจำนวนจริงมีค่าระหว่าง 0 ถึง 1 กำหนดความโปร่งใส ของวัตถุ

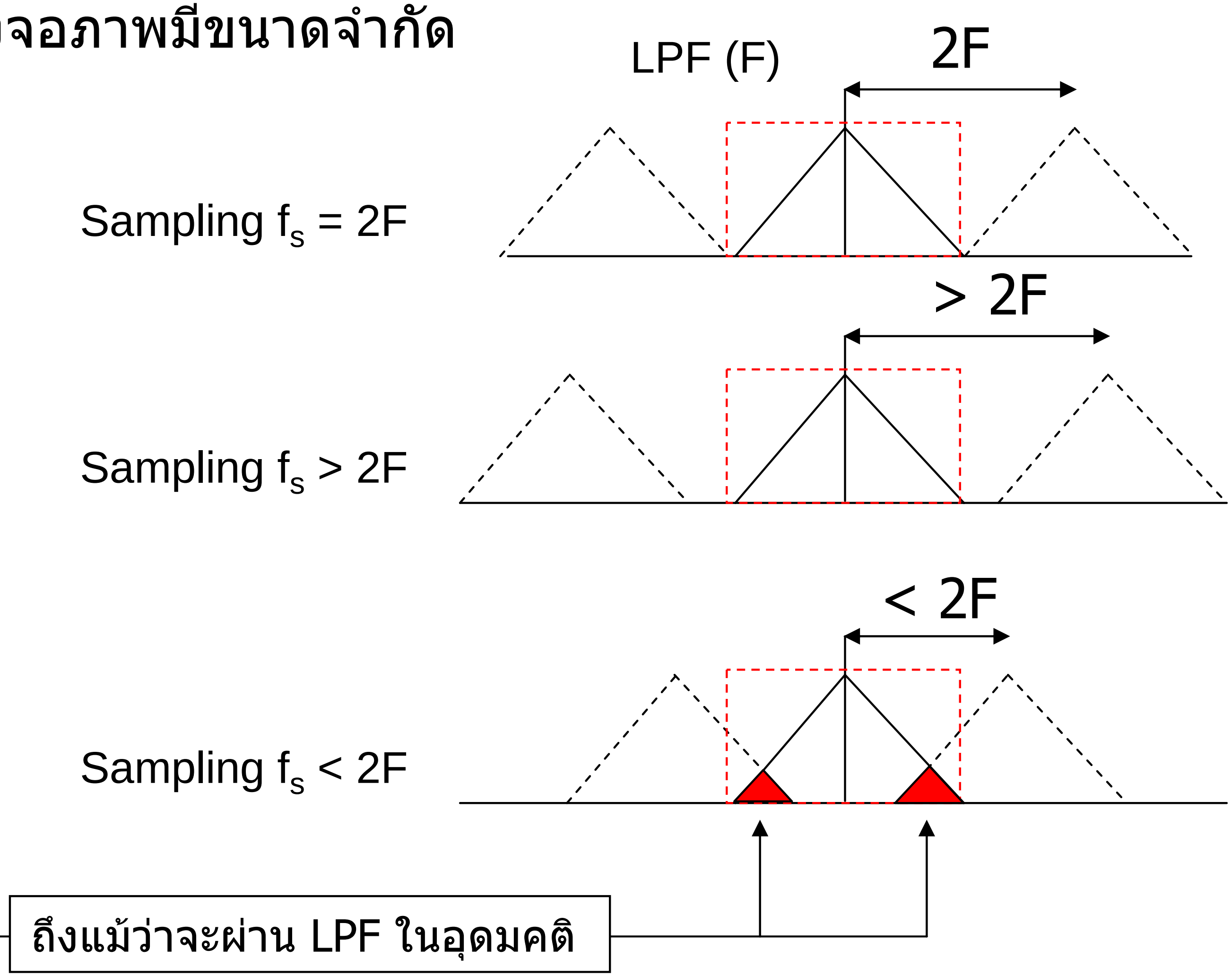
ค่า t น้อยวัตถุโปร่ง สี P ประกอบด้วย สี B ด้วยอัตราส่วนมากกว่าสี F

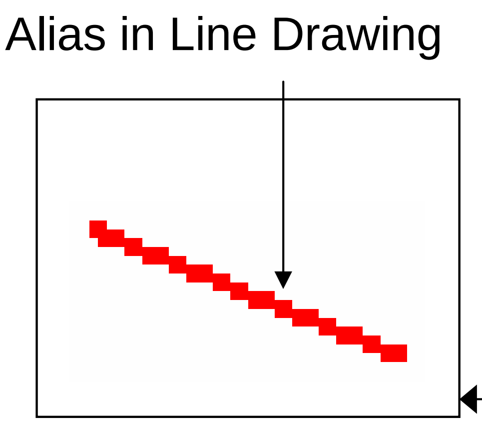
ถ้าวัตถุวางซ้อนกันหลายชั้น ? ($P^i \rightarrow B^{i+1}$)

Signal Alias in Graphics

จากการวิเคราะห์ Fourier สัญญาณเงาเกิดจาก อัตราสุ่มสัญญาณมีค่าน้อยกว่าสองเท่าของแถบความถี่ของสัญญาณที่จะสุ่ม ในทาง Graphics จะเกิดขึ้นเสมอ เนื่องจาก รายละเอียดของจอภาพมีขนาดจำกัด







Anti-aliasing

การปรับปรุงคุณภาพของ การแสดงผลที่ผิดเพี้ยน อันเนื่องมาจาก Signal Alias เรียกว่า Anti-aliasing

มีวิธีที่นิยมกัน 2 วิธี ได้แก่

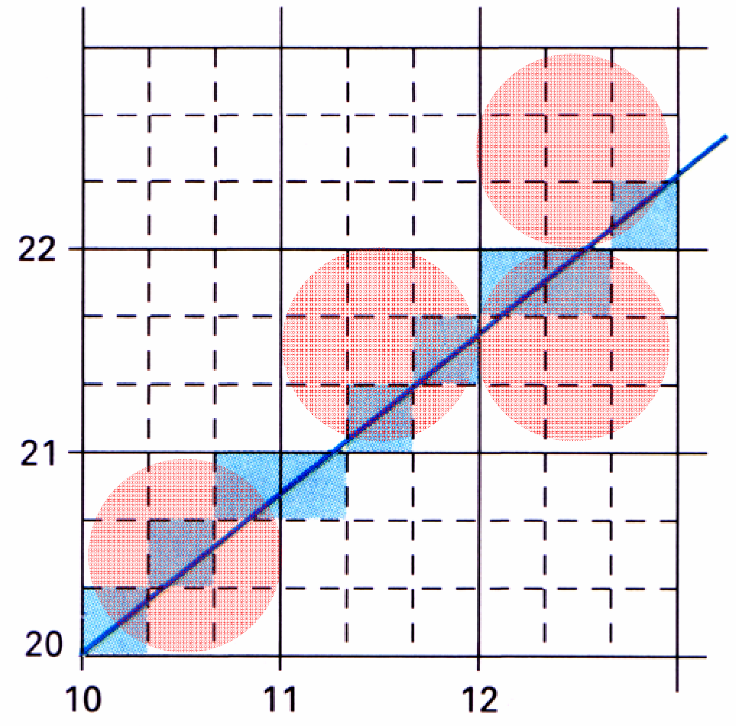
✓ **Super-sampling (Post Filtering)**
ใช้สำหรับอุปกรณ์แสดงผลที่สามารถ แสดงจุดภาพได้หลายระดับความสว่าง โดยทำการสุ่มเทียมแต่ละจุดภาพให้ละเอียดขึ้น แล้วปรับค่าความเข้มของจุดภาพให้เหมาะสม จนเสมือนว่าภาพที่ได้ปรากฏต่อผู้สังเกตเรียบขึ้น (non-linear filter)

✓ **Direct Filtering**
คือการกรองสัญญาณที่ผิดเพี้ยนด้วย LPF ที่ความถี่ต่ำผ่านน้อย (มีอัตราการลดทอนมากขึ้น ที่ความถี่สูง) เพื่อให้ส่วนที่เกิดปัญหา (Overlapping Area) ส่งผลต่อภาพน้อยลง ซึ่งจะทำให้เส้นที่ปรากฏต่อผู้สังเกตเรียบขึ้น (linear filter)

Super Sampling Technique

วิธีนี้มี 3 ขั้นตอนคือ

1) จุดภาพจริงแต่ละจุดจะถูกแบ่งออกเป็นจุดภาพย่อยๆ (sub-pixels) เสมือน
2) นับจำนวน **จุดภาพย่อย** ที่ซ้อนทับกับองค์ประกอบเรขาคณิตที่กำหนด
3) กำหนดระดับความเข้มของจุดภาพจริง แปรผันตรงกับจำนวนจุดย่อยที่นับได้



ตัวอย่างนี้ แสดงการแบ่งจุดภาพจริง ออกเป็น 3x3 จุดภาพย่อย

สังเกตว่า กรณีนี้ สำหรับเส้นตรง ใดๆ จะผ่านจุดภาพย่อย ไม่เกิน 3 จุดภาพต่อ 1 จุดภาพจริง ดังนั้นระดับความเข้มที่เป็นไปได้จึงเท่ากับ 3 + 1 (จุดว่าง) = 4 ระดับ

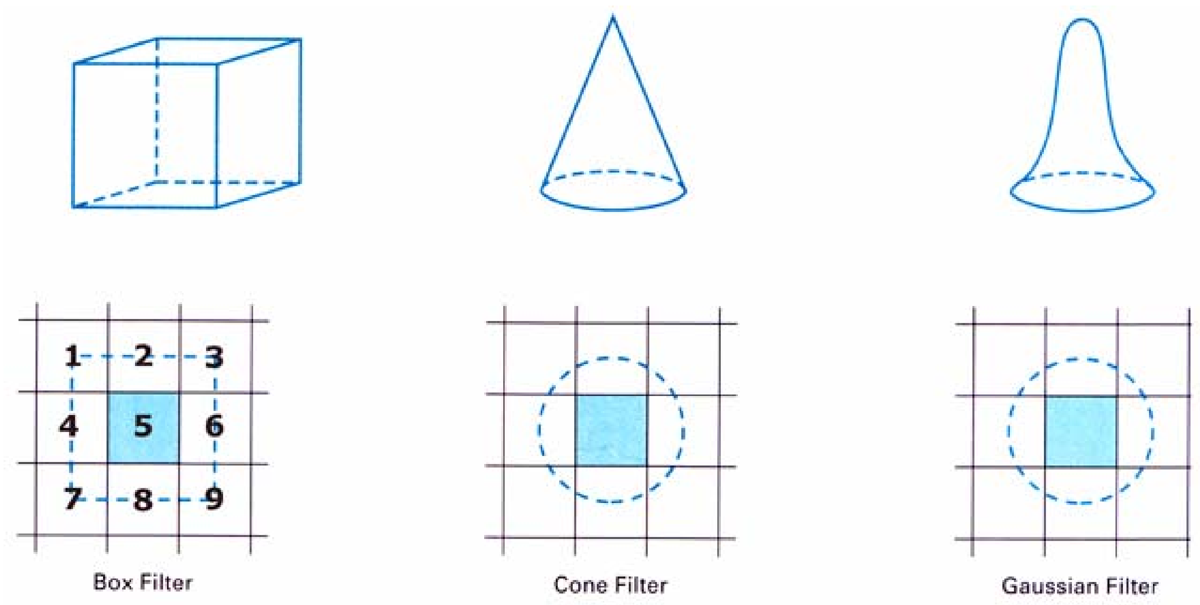
จุดที่ (10, 20) = 3, (11, 21) = (12, 21) = 2, (11, 20) = (12, 22) = 1, จุดอื่นๆ = 0

Bresenham's Line Drawing บนจุดภาพจริง (สีแดง) และจุดภาพย่อย (น้ำเงิน)

Direct Filtering

เป็นวิธีที่แม่นตรงมากขึ้น (แต่ใช้การคำนวณมากกว่า) โดยมีหลักการว่าให้ความสำคัญกับจุดภาพย่อยตรงกลางมากกว่าจุดภาพย่อยตามขอบ

1) จุดภาพจริงแต่ละจุดจะถูกแบ่งออกเป็นจุดภาพย่อยๆ (sub-pixels) เสมือน
2) สำหรับแต่ละจุดภาพจริง หาค่า ผลบวกถ่วงน้ำหนัก ระหว่างจุดภาพย่อยๆ กับ Filter Kernel หรือ (Mask) ที่นิยมกันคือ Binomial Mask
3) กำหนดระดับความเข้มของจุดภาพจริง แปรผันตรงกับผลบวกถ่วงน้ำหนัก



$$V = \sum_i \left(P_i \cdot M_i\right)$$

V ค่าความเข้มผลลัพธ์
P สถานะของจุดภาพย่อย (*1* ซ้อนทับ, *0* ไม่ซ้อน)
M ค่าน้ำหนักของ Kernel ณ ตำแหน่งจุดภาพย่อย ($\Sigma M = 1$)

2D Geometric Transformation

หัวข้อต่อไปนี้จะกล่าวถึงการ ปรับ-แปลง การแสดงผลองค์ประกอบเรขาคณิต ซึ่งเรียกว่า Transformation ซึ่งมีที่ใช้งานดังต่อไปนี้

- โปรแกรมประยุกต์ประเภทการออกแบบ (Design Applications) จะจัดวางรูปแบบของกลุ่มวัตถุ โดยนิยาม การหมุน (Rotation) ขนาด (Size) และตำแหน่งสัมพัทธ์ (หรือสัมบูรณ์) ขององค์ประกอบต่างๆ ที่กำหนด
- การสร้างภาพเคลื่อนไหว สามารถทำได้โดยเลื่อน กล้อง (เสมือน) หรือ วัตถุในแนวเส้นทางของการเคลื่อนไหว

การเปลี่ยนแปลงดังกล่าว ได้แก่ การเปลี่ยนมุมการหมุน ขนาด และ ตำแหน่ง เรียกรวมกันว่า Geometric Transformation (การแปลงทางเรขาคณิต) ซึ่งนิยามได้ว่า คือ *การเปลี่ยนปริภูมิที่ใช้นิยามวัตถุ*

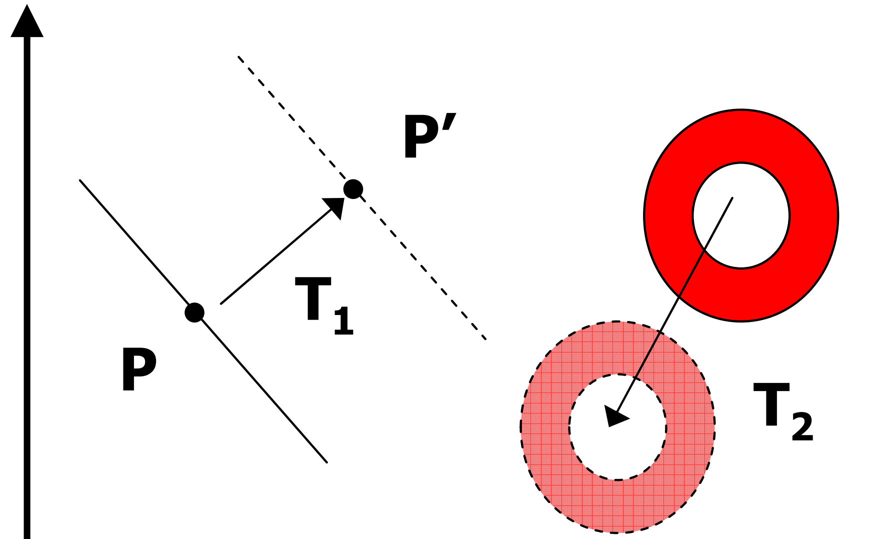
Geometric Transformation พื้นฐานได้แก่ การเลื่อน (Translation) การหมุน (Rotation) และ การย่อ/ขยาย (Scaling) ซึ่งจะได้อธิบายต่อไป

Translation

การเลื่อน (Translation) สามารถ นิยามได้ว่าคือการ **เปลี่ยนตำแหน่ง ของวัตถุไปในแนวเส้นตรง** จากตำแหน่งหนึ่ง ในปริภูมิ ไปยังอีกตำแหน่งหนึ่ง

เราสามารถเลื่อน จุดใดๆ ในสองมิติ ได้โดยการบวก ระยะการเลื่อน (Translation Distances) ในรูปของเวกเตอร์ในแต่ละแกน (t_x *และ* t_y) จากจุดเดิม (x, y) ไปยังจุดใหม่ (x′, y′) ดังนี้

$$x' = x + t_x, \quad y' = y + t_y$$



หรือในรูป Matrix

$$\mathbf{P}' = \mathbf{P} + \mathbf{T}$$

$$\mathbf{P}' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix}, \mathbf{T} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

ทุกๆ จุดบนวัตถุจะเลื่อนไปด้วยปริมาณเท่ากัน

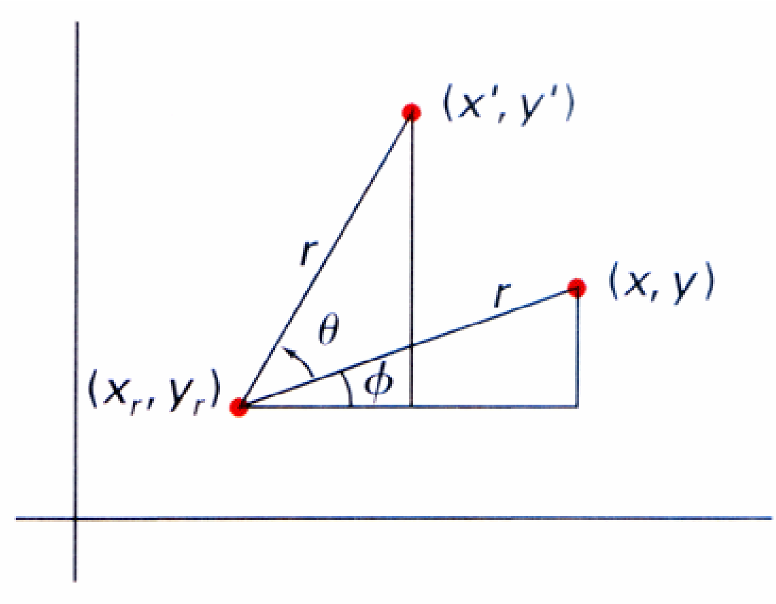
Rotation

การหมุนวัตถุ กรณีที่ pivot point เป็นจุดใดๆ บนระนาบ (x, y) ทำได้โดย

- เลื่อนจุดที่ต้องการหมุนไปยังจุดกำเนิดก่อน $\mathbf{T_1} = [-x_r, -y_r]^T$
- ทำการหมุนโดยใช้ความสัมพันธ์การหมุน ซึ่งมีจุดศูนย์กลางที่จุดกำเนิด
- เลื่อนจุดที่หมุนเรียบร้อยแล้วมาที่ตำแหน่ง pivot point $\mathbf{T_2} = [+x_r, +y_r]^T$

ซึ่งสามารถแสดงได้ด้วยสมการ

$$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta$$
$$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta$$



หรือในรูป Matrix

$$\mathbf{P}' = \mathbf{T}_r + \mathbf{R}\cdot(\mathbf{P} - \mathbf{T}_r)$$

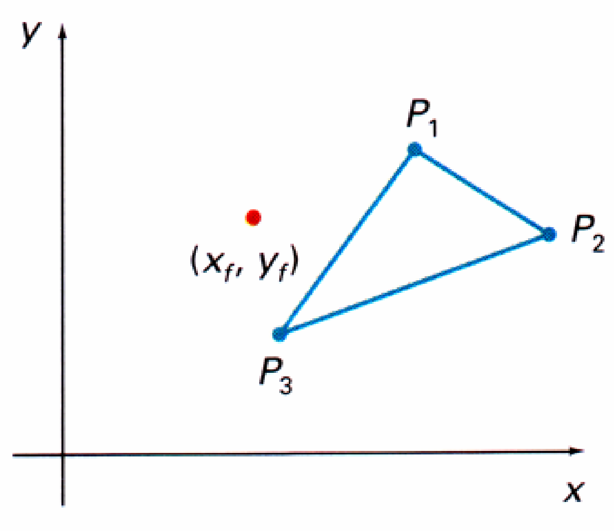
$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$\mathbf{T}_r = \begin{bmatrix} x_r \\ y_r \end{bmatrix}$$

Scaling

การย่อ/ขยาย แบบสัมพัทธ์ คือการย่อ/ขยายวัตถุ โดยกำหนดจุดอ้างอิง ซึ่งเป็นจุดคงที่ (ไม่มีการเปลี่ยนแปลง) ทั้งก่อนและหลังการ ย่อ/ขยาย เราสามารถเลือกจุดอ้างอิง (x_f, y_f) ได้อิสระซึ่งอาจจะเป็น จุดยอดมุมของวัตถุ จุดศูนย์กลางมวลของวัตถุ หรือ จุดอื่นใดก็ได้ ขั้นตอนการทำคล้ายกับ การหมุน แบบมี pivot point

- เลื่อนวัตถุ โดยให้จุดอ้างอิงไปอยู่ที่จุดกำเนิดก่อน $\mathbf{T_1} = [-x_f, -y_f]^T$
- ทำการย่อ/ขยายตามปรกติ
- เลื่อนวัตถุที่ได้ ให้จุดอ้างอิงกลับมาที่เดิม $\mathbf{T_2} = [+x_f, +y_f]^T$



$$x' = x_f + s_x(x - x_f), \quad y' = y_f + s_y(y - y_f)$$

หรือจัดพจน์ใหม่จะได้

$$x' = s_x x + (1 - s_x)x_f, \quad y' = s_y y + (1 - s_y)y_f$$

Homogeneous Coordinates

การทำ Transformation ด้วยสมการข้างต้น หลายครั้ง จะทำให้เกิดข้อผิดพลาดสะสม สำหรับ Integer Arithmetic ดังนั้นจึงจำเป็นต้อง รวมพจน์ $\mathbf{M}_1$ และ $\mathbf{M}_2$ เข้าด้วยกัน

หลักการของวิธีนี้ คือจัด Geometric Transformation ในรูปของ การคูณกันของ Matrix ซึ่งทำได้โดยขยาย พจน์ที่เป็น Matrix ขนาด 2x2 เป็น 3x3 และ 2x1 เป็น 3x1 ตามลำดับ ซึ่งทำได้โดย จัดพิกัด Cartesian (x, y) ในรูปของ Homogeneous Coordinates (x_h, y_h, h)

$$x = \frac{x_h}{h} \qquad y = \frac{y_h}{h}$$

ดังนั้นพิกัด Homogeneous อาจเขียนได้ใหม่เป็น $(x \bullet h, y \bullet h, h)$

โดยทั่วไป เราสามารถเลือกค่า h เป็นจำนวนจริงบวกใดๆ แต่เพื่อความสะดวกมักจะเลือกให้ h = 1 ซึ่งจะได้พิกัด Homogeneous (x, y, 1)

Homogeneous Transformations

โดยการใช้พิกัด Homogeneous (*ตัวดำเนินการ บนตัวแปรที่ตำแหน่ง a จะให้ผลเดียวกันกับตัวแปรที่ตำแหน่ง b*) เราสามารถจัดรูป Transformation ใหม่ได้ดังนี้

Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

สังเกต

การแปลงกลับ (Inverse Transformation) สามารถทำได้โดย เพียงหา Inverse ของ Matrix ที่เกี่ยวข้องนั่นเอง

Homework (1)

จงพิสูจน์ และ แสดงว่า Inverse ของ Transformation (Translation, Rotation, และ Scaling) มีความหมายในทาง Graphics ซึ่งสอดคล้องกับความเป็นจริง

ตัวอย่าง เราสามารถใช้หลักการ Inverse Matrix พิสูจน์ได้ว่า

Inverse Translation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

ซึ่งหมายความว่า การแปลงจุดที่เลื่อนไปกลับไปยังจุดเดิม ทำได้โดยเลื่อนจุดนั้นในทิศทางตรงกันข้าม

คำแนะนำ

ใช้วิธีการหา Inverse ของ Matrix แก้สมการ หาค่า (x, y, 1) ในรูปของ (x′, y′, 1) แล้วตีความหมายของผลลัพธ์ที่ได้

Composite Transformations

เนื่องจาก Geometric Transformation อยู่ในรูปของ Matrix กระบวนการต่างๆ สามารถนำมาเกี่ยวโยง กันได้ด้วยวิธี Composite Transformation Matrix ซึ่งเรียกรวมกันว่า **Concatenation** หรือ **Composition**

Composite Translations

$$\mathbf{P}' = \mathbf{T}_2 \cdot (\mathbf{T}_1 \cdot \mathbf{P}) = (\mathbf{T}_2 \cdot \mathbf{T}_1) \cdot \mathbf{P}$$

Composite Rotations

$$\mathbf{P}' = \mathbf{R}_2 \cdot (\mathbf{R}_1 \cdot \mathbf{P}) = (\mathbf{R}_2 \cdot \mathbf{R}_1) \cdot \mathbf{P}$$

Composite Scaling

$$\mathbf{P}' = \mathbf{S}_2 \cdot (\mathbf{S}_1 \cdot \mathbf{P}) = (\mathbf{S}_2 \cdot \mathbf{S}_1) \cdot \mathbf{P}$$

Composite Transformations

สำหรับการหมุน และ การย่อขยายก็ทำการเชื่อมโยงได้ในทำนองเดียวกัน

Composite Rotations

การเลื่อนวัตถุ 2 ครั้ง ด้วย R_1 (θ_1) และ R_2 (θ_2) แสดงได้โดยใช้กฎการจัดหมู่ของ Matrix

$$\mathbf{P}' = \mathbf{R}_2 \cdot (\mathbf{R}_1 \cdot \mathbf{P}) = (\mathbf{R}_2 \cdot \mathbf{R}_1) \cdot \mathbf{P}$$

การบ้าน (2) พิสูจน์ว่า

$\mathbf{R}_2$ (θ_2) • $\mathbf{R}_1$ (θ_1) = $\mathbf{R}$ ($\theta_2+\theta_1$)

Composite Scaling

การย่อ/ขยาย วัตถุ 2 ครั้ง ด้วย S_1 (s_{x1}, s_{y1}) และ S_2 (s_{x2}, s_{y2}) แสดงได้โดย

$$\begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x2} \cdot s_{x1} & 0 & 0 \\ 0 & s_{y2} \cdot s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Suranaree University
of Technology

COMPUTER GRAPHICS
School of Computer Engineering

มหาวิทยาลัยเทคโนโลยีสุรนารี

Lecture 5

Geometric Transformations (Part II) and Scene Analysis

Paramate Horkaew

School of Computer Engineering, Institute of Engineering
Suranaree University of Technology

Lecture Outline

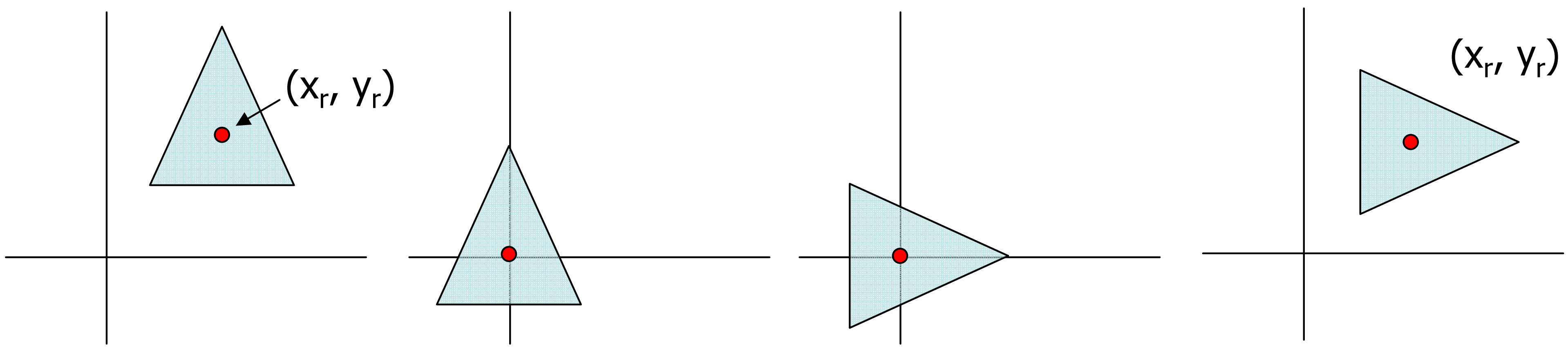
- 2D Geometric Transformations
  - Basic Transformations
  - Matrix Representation and Homogeneous Coordinates
  - General Composite Transformations
  - Reflection and Shear Transformation
  - Coordinate Transformations
  - Affine Transformation
  - Raster Methods for Transformation
- 2-Dimensional Viewing
  - The viewing Pipeline
  - Viewing Coordinate Reference Frame
  - Window to Viewport Coordinate Transformation
  - Practical Viewing Example

General Pivot Rotations

โดยการใช้ Matrix Composition เราสามารถทำการหมุนวัตถุรอบจุดอ้างอิง (Pivot Point) ใดๆ ได้ด้วยขั้นตอนต่อไปนี้

- เลื่อนวัตถุ เพื่อให้ Pivot Point (x_r, y_r) เลื่อนตามไปอยู่ที่จุดกำเนิด
- หมุนวัตถุรอบจุดกำเนิด
- เลื่อนวัตถุ อีกครั้ง แต่ครั้งนี้ให้ Pivot Point กลับมาอยู่ที่จุดเดิม

ขั้นตอนดังกล่าวแสดงได้ ตามลำดับ ดังรูป



General Pivot Rotations

Matrix สำหรับ Composite Transformation แบบนี้สามารถหาได้โดยการนำ Transformation ต่างๆ มาเขียนเรียงกัน

$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta)+y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta)-x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

หรือในรูปของฟังก์ชัน $\mathbf{T}(x_r, y_r)\cdot\mathbf{R}(\theta)\cdot\mathbf{T}(-x_r, -y_r) = \mathbf{R}(x_r, y_r, \theta)$

General Fixed-Point Scaling

ในทำนองเดียวกันกับการหา Matrix โดยใช้สมการการ ย่อ/ขยาย และ การเลื่อนวัตถุ เราสามารถสร้าง Matrix รวม สำหรับ การ ย่อ/ขยาย รอบจุดอ้างอิงได้ โดยขั้นตอนต่อไปนี้

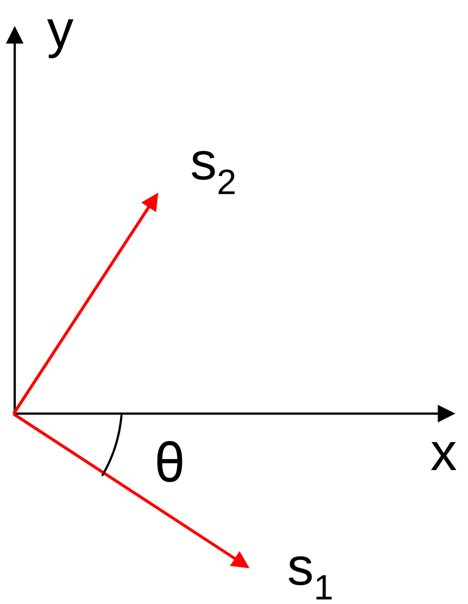
- เลื่อนวัตถุ เพื่อให้จุดอ้างอิงคงที่ (x_r, y_r) เลื่อนตามไปอยู่ที่จุดกำเนิด
- ย่อ/ขยายวัตถุ ตามปรกติ อ้างอิงกับจุดกำเนิด
- เลื่อนวัตถุ อีกครั้ง แต่ครั้งนี้ให้จุดอ้างอิงคงที่ กลับมาอยู่ที่จุดเดิม

$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & x_r(1-s_x) \\ 0 & s_y & y_r(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

หรือในรูปของฟังก์ชัน $\mathbf{T}(x_r, y_r) \cdot \mathbf{S}(s_x, s_y) \cdot \mathbf{T}(-x_r, -y_r) = S(x_r, y_r, s_x, s_y)$

General Scaling Direction

โดยนิยามแล้วค่าตัวแปร s_x และ s_y คือสัมประสิทธิ์การย่อ/ขยายในแนวแกน x และ y ตามลำดับ อย่างไรก็ดี ด้วยเทคนิค Matrix Composition เราสามารถย่อ/ขยาย วัตถุ อ้างอิงกับแกนใดๆ ได้โดย 1) หมุนวัตถุให้ แกนอ้างอิงที่ต้องการซ้อนทับกับ แกนปกติของระบบพิกัด 2) ทำการย่อขยายตามปกติ และ 3) หมุนวัตถุกลับในทิศทางตรงข้ามกับข้อ 1)



$$\mathbf{R}^{-1}(\theta)\cdot\mathbf{S}(s_1, s_2)\cdot\mathbf{R}(\theta)$$

$$= \begin{bmatrix} s_1\cos^2\theta + s_2\sin^2\theta & (s_2 - s_1)\cos\theta\sin\theta & 0 \\ (s_2 - s_1)\cos\theta\sin\theta & s_1\sin^2\theta + s_2\cos^2\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

การบ้าน จงพิสูจน์ Matrix Composition วิธีนี้

Concatenating Properties

การดำเนินการแบบ Matrix เป็นแบบจัดหมู่ได้ (คุณสมบัติ Associative) ดังนั้น สำหรับ Matrix $\mathbf{A} \bullet \mathbf{B} \bullet \mathbf{C} = (\mathbf{A} \bullet \mathbf{B}) \bullet \mathbf{C} = \mathbf{A} \bullet (\mathbf{B} \bullet \mathbf{C})$

อย่างไรก็ดี Matrix ไม่มี คุณสมบัติการสลับที่ (Commutative) ดังนั้น เราต้องระวังว่า ถ้าจะทำการเลื่อนและหมุนวัตถุ ต้องทำในลำดับที่ถูกต้องทั้งทางกายภาย และทางการคำนวณ

มี Matrix เพียงบางคู่เท่านั้น ที่มีคุณสมบัติการสลับที่ นั่นคือ

- การหมุนวัตถุด้วยกัน $\mathbf{R}_1 \bullet \mathbf{R}_2 = \mathbf{R}_2 \bullet \mathbf{R}_1$
- การเลื่อนวัตถุด้วยกัน $\mathbf{T}_1 \bullet \mathbf{T}_2 = \mathbf{T}_2 \bullet \mathbf{T}_1$
- การย่อขยายวัตถุด้วยกัน $\mathbf{S}_1 \bullet \mathbf{S}_2 = \mathbf{S}_2 \bullet \mathbf{S}_1$
- การย่อขยายวัตถุและการหมุนวัตถุ $\mathbf{S}_1 \bullet \mathbf{R}_1 = \mathbf{R}_1 \bullet \mathbf{S}_1$

ลำดับการดำเนินการที่ต่างกันของ Matrix ให้ผลการ Transform ที่ต่างกัน

General Composite Transformation

การดำเนินการ Transformation ใดๆ ซึ่งประกอบด้วย การเลื่อน การหมุน และ การย่อ/ขยาย วัตถุ สามารถสร้างได้โดยการทำ Concatenation ของ Transformation ต่างๆ ซึ่งผลลัพธ์สุดท้าย แสดงในรูปของ Matrix ได้ดังนี้

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} rs_{xx} & rs_{xy} & trs_x \\ rs_{yx} & rs_{yy} & trs_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

rs_{ij} คือ ผลคูณระหว่างพจน์การหมุน และ การย่อ/ขยาย

trs_i คือ พจน์ที่เกี่ยวกับการเลื่อนวัตถุ ประกอบด้วย ระยะทางเลื่อน จุดอ้างอิงการหมุน (pivot point) จุดคงที่การย่อ/ขยาย (fixed point) การหมุน และ การย่อ/ขยาย

$$x' = rs_{xx}x + rs_{xy}y + trs_x$$

$$y' = rs_{yx}x + rs_{yx}y + trs_y$$

การคูณ 4 ครั้ง และ
การบวก 4 ครั้ง

An Example

ถ้าต้องการ 1) ย่อ/ขยายวัตถุ รอบจุดศูนย์กลางมวล (x_c, y_c) แล้ว 2) หมุนวัตถุรอบจุดศูนย์กลางมวล ตามด้วย 3) เลื่อนวัตถุไปเป็นระยะทาง (t_x, t_y) เราสามารถเขียนในรูปของ Composite Matrix ได้ดังนี้

$$\mathbf{T}(t_x, t_y)\cdot\mathbf{R}(x_c, y_c, \theta)\cdot\mathbf{S}(x_c, y_c, s_x, s_y)$$

$$= \begin{bmatrix} s_x\cos\theta & -s_y\sin\theta & x_c(1-s_x\cos\theta)+y_c s_y\sin\theta+t_x \\ s_x\sin\theta & s_y\sin\theta & y_c(1-s_y\cos\theta)-x_c s_x\sin\theta+t_y \\ 0 & 0 & 1 \end{bmatrix}$$

ซึ่งการคำนวณหาพิกัดจุด สุดท้าย ต้องใช้การคูณ 9 ครั้ง การบวก 6 ครั้ง

ดังนั้นเพื่อให้การคำนวณมีประสิทธิภาพ สำหรับระบบ Graphics ซึ่งต้องมีการประมวล Matrix หลายๆ ครั้งสำหรับองค์ประกอบทางเรขาคณิต จึงแนะนำให้สร้าง Matrix Composite ก่อนแล้วจึงนำ Matrix ที่ได้ไปใช้แปลงวัตถุ

Rigid-Body Transformation

Transformation ของวัตถุเกร็ง (บางครั้งเรียก Rigid-Motion Transformation) ประกอบด้วย การเลื่อน และ การหมุน ซึ่งสามารถแสดงได้ด้วย Matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \boxed{\begin{matrix} r_{xx} & r_{xy} \\ r_{yx} & r_{yy} \end{matrix}} & \begin{matrix} tr_x \\ tr_y \end{matrix} \\ \begin{matrix} 0 & 0 \end{matrix} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

โดยมีคุณสมบัติคือ

- ทั้งระยะทาง และ มุม ระหว่างกรอบพิกัด (Coordinate Frames) ไม่เปลี่ยนระหว่างการแปลง
- Matrix ย่อยขนาด 2x2 มุมบนซ้ายเป็น Matrix ตั้งฉาก (Orthogonal Matrix) นั่นคือ unit vector (ขนาดของเวคเตอร์ = 1) ของทั้งสองแถว dot กันได้ 0

$$r_{xx}^2 + r_{xy}^2 = r_{yy}^2 + r_{yx}^2 = 1 \quad and \quad r_{xx}r_{yx} + r_{xy}r_{yy} = 0$$

Unique Property

จากคุณสมบัติข้างต้น พบว่าหากทำการแปลง พิกัดซึ่งแสดงด้วย vector แถวบน และ แถวล่าง ของ sub-Matrix มุมบนซ้าย แล้วจะได้เป็น vector ขนาด 1 หน่วย ตามแนวแกน x และแกน y ตามลำดับ

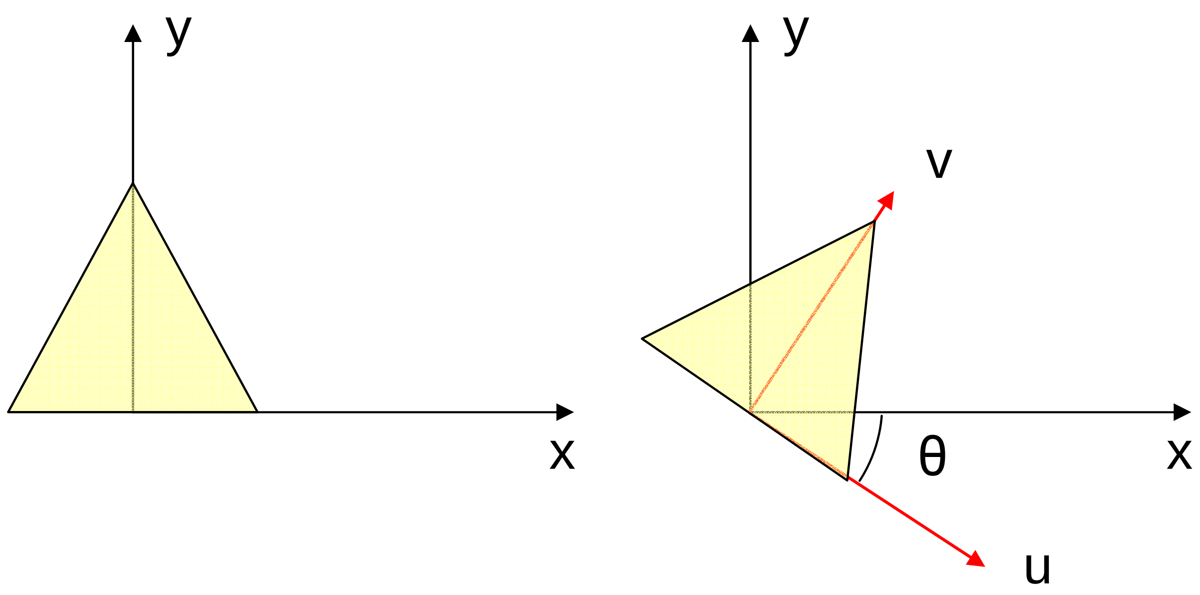
$$\begin{bmatrix} r_{xx} & r_{xy} & 0 \\ r_{yx} & r_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{xx} \\ r_{xy} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} r_{xx} & r_{xy} & 0 \\ r_{yx} & r_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{yx} \\ r_{yy} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

ตัวอย่าง ถ้าหมุนวัตถุรอบจุดกำเนิดด้วยมุม θ

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

A Useful Application

เราสามารถนำคุณสมบัติดังกล่าวไปใช้หา Matrix ของการหมุนวัตถุได้ ถ้าเราไม่ทราบ มุม θ ที่ต้องการหมุน แต่ทราบ แกนเอียงของวัตถุ ดังรูป



รูปสามเหลี่ยมมีแกนเดิมขนานกับ (x, y) ต้องการหมุนวัตถุให้ แกนของสามาเหลี่ยมขนานกับ unit vector u และ v จงหา Matrix และ มุม θ

วิธีทำ

แทนค่า u, v ใน Sub-Matrix

$$\mathbf{R}(\mathbf{u},\mathbf{v}) = \begin{bmatrix} u_x & u_y & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \theta = \cos^{-1} u_x = \sin^{-1}(-u_y)$$

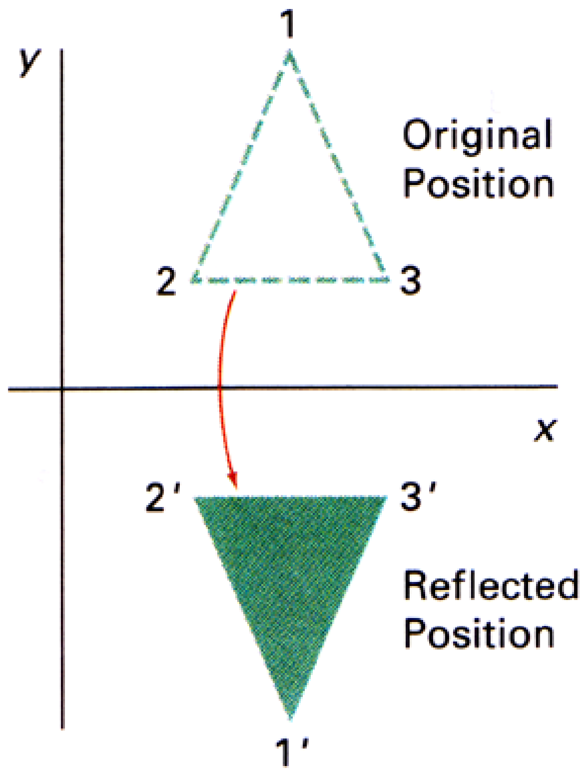
Other Transformations

เนื่องจากการสร้าง Transformation Matrix โดยทั่วไปมัก โดยเฉพาะอย่างยิ่งการหมุนวัตถุ จะต้องคำนวณค่า **ตรีโกณมิติ** ดังนั้นเพื่อให้การคำนวณมีประสิทธิภาพมากขึ้น จึงมีการนิยาม Matrix สำหรับกรณีเฉพาะขึ้นมา เพื่อหลีกเลี่ยงการคำนวณที่ซับซ้อน

Reflection

คือการแปลงวัตถุให้อยู่ในรูปสะท้อนเงากระจก (Mirror Image) เทียบกับเส้นของการสะท้อน ซึ่งในบริบทของ Matrix ทั่วไป คือการหมุนวัตถุ รอบแกนสะท้อนเป็นมุม 180 องศานั่นเอง

เราสามารถเลือกแกนสะท้อน เป็นเส้นตรงใดๆ ในระนาบ xy ก็ได้ จากตัวอย่างเส้นการสะท้อน คือ แกน x (เส้นตรง y = 0)



Reflection Matrix

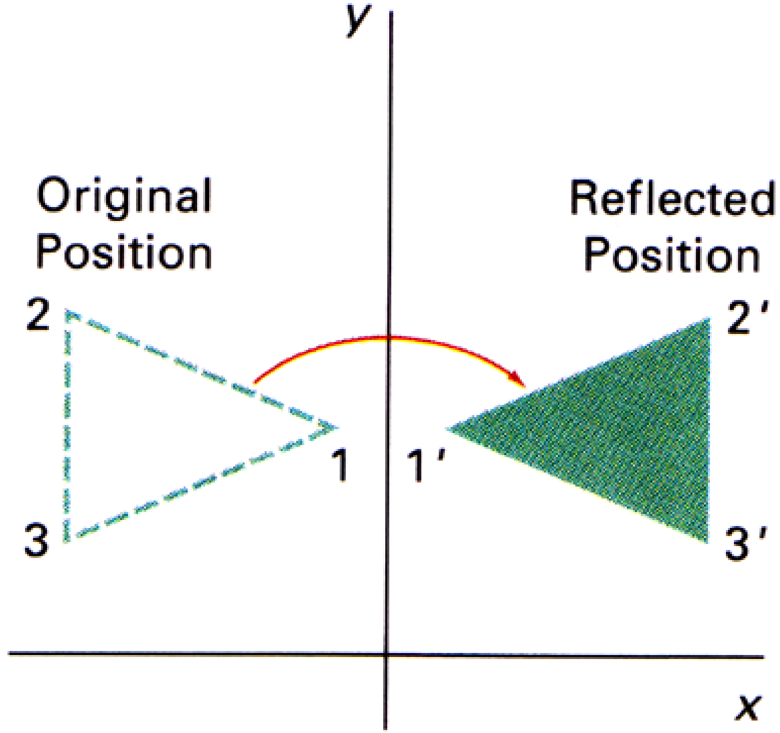
จากตัวอย่างการสะท้อนเทียบกับแกน x สามารถแสดงได้ด้วย Matrix

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

ซึ่งสังเกตได้ว่า Matrix ตัวนี้จะคงค่าพิกัด x เดิมของวัตถุไว้ ในขณะที่ กลับ (Flip) พิกัดค่า y ให้มีค่าเป็นลบ ซึ่งสอดคล้องกับรูปตัวอย่าง

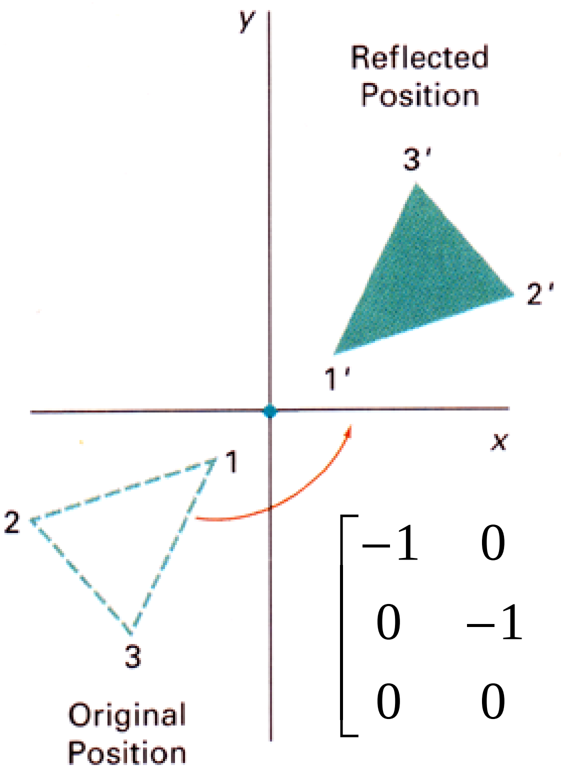
ดังนั้น การสะท้อนเทียบกับแกน y (x = 0) ก็พิจารณาได้ในทำนองเดียวกัน

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



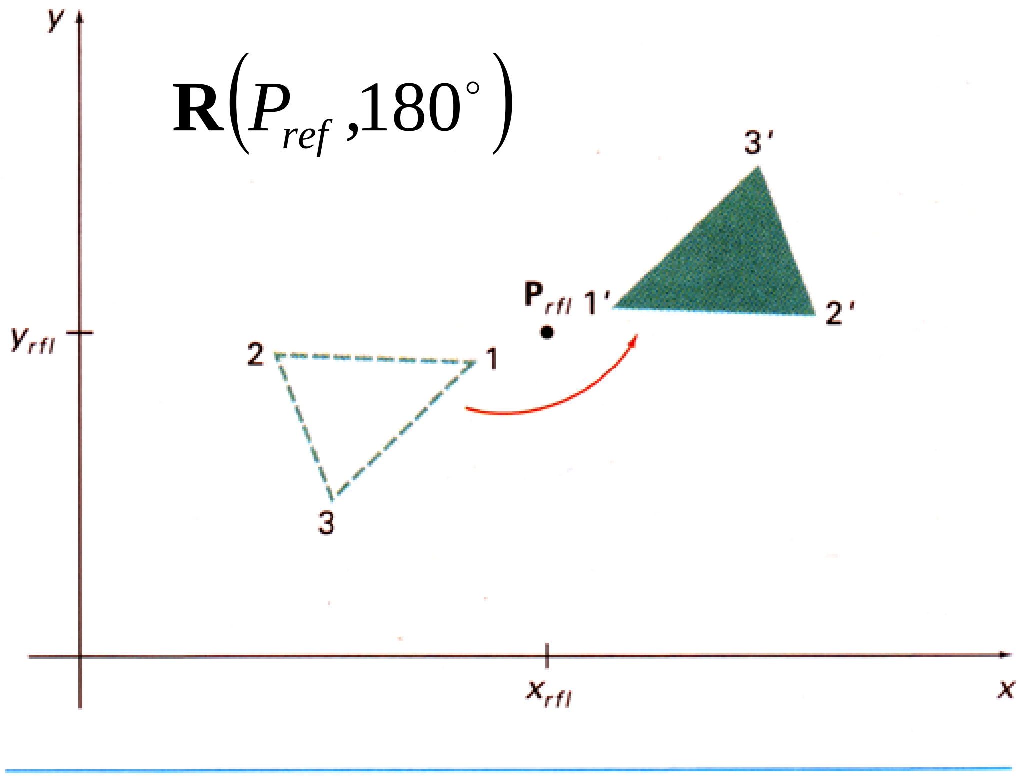
Reflections About A Perpendicular Axis

การสะท้อนรอบแกน z (ตั้งฉากกับระนาบ xy) เรียกว่า การสะท้อนรอบจุดกำเนิด แสดงดังรูปด้านซ้ายมือ ในกรณีทั่วไป การสะท้อนเทียบกับแกนตั้งฉากระนาบ xy รอบจุด **P** ใดๆ สามารถสร้างได้โดยใช้ เทคนิคการหมุนรอบจุดอ้างอิง 180 องศา



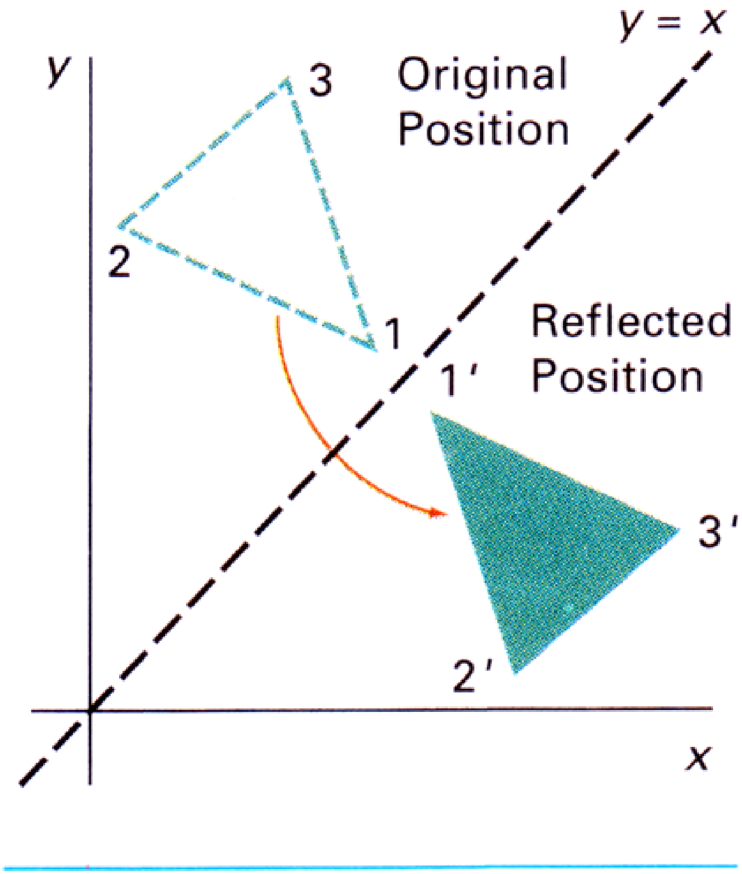
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}\left(P_{ref}, 180^\circ\right)$$

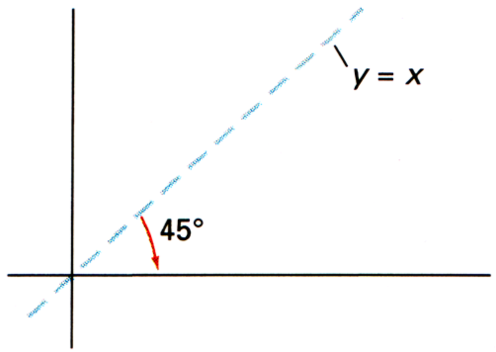


Arbitrary Rotation

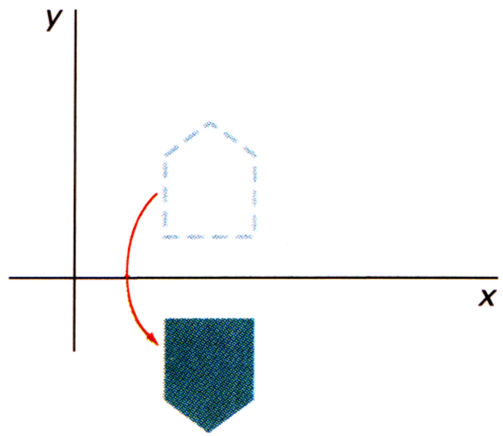
สำหรับการสะท้อนรอบเส้นตรงใดๆ สามารถพิจารณาจากเส้นตรง y = x โดยอาจสร้างได้โดยลำดับของ Transformation Matrix



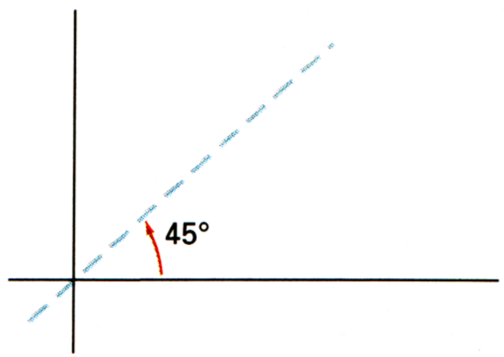
ในกรณีที่เส้นตรง y = ax + b ???



หมุนวัตถุ CW 45 องศา



สะท้อนกับแกน x



หมุนวัตถุ CCW 45 องศา

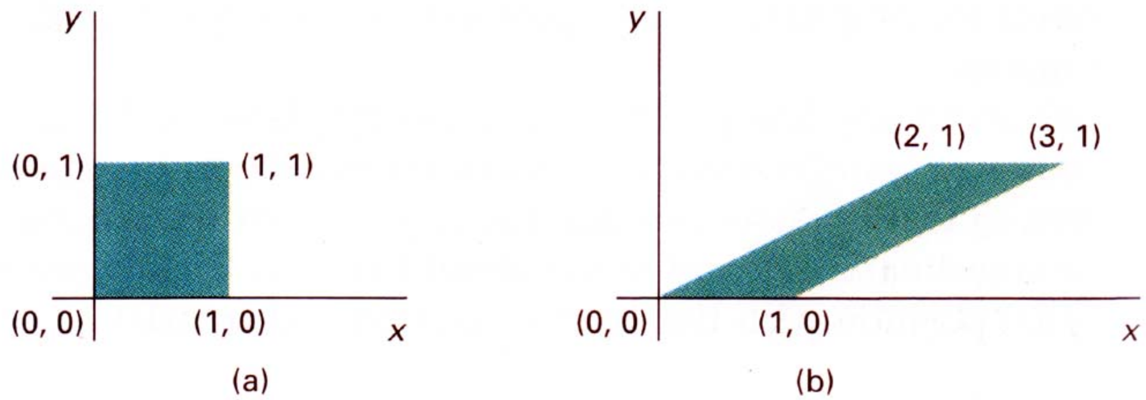
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shear

Shear (เฉือน) คือการแปลงใช้วัตถุมีรูปร่างบิดเบี้ยว เสมือนว่าวัตถุประกอบด้วยชั้นบางๆ ของวัสดุ แล้วได้รับแรงอัดเฉือน **เชิงเส้น** ในแต่ละชั้นของวัสดุ เทียบกับจุด y (หรือ x) = 0 ซึ่งจำแนกได้ 2 แบบ คือ ในแนวแกน x และ ในแนวแกน y

แกน x

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x + sh_x \cdot y \\ y \\ 1 \end{bmatrix}$$



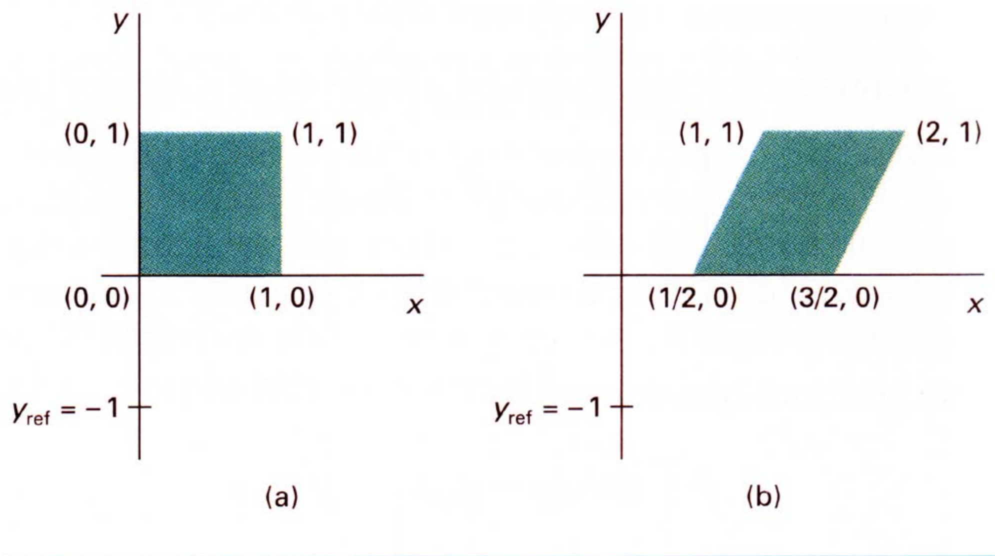
กำหนดสัมประสิทธิ์การเฉือนเท่ากับ 2 เปลี่ยนสี่เหลี่ยมจัตุรัสเป็นสี่เหลี่ยมด้านขนาน

General Shear (X)

เราสามารถกำหนดแรงอัดเฉือน **เชิงเส้น** ในแต่ละชั้นของวัสดุ โดยอ้างอิงจากจุดใดๆ ก็ได้ เพียงแค่เพิ่มค่าคงที่ (Constant Shear/Offset) ในพจน์การเลื่อน

แกน x

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & -sh_x y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x + sh_x \cdot \left(y - y_{ref} \right) \\ y \\ 1 \end{bmatrix}$$



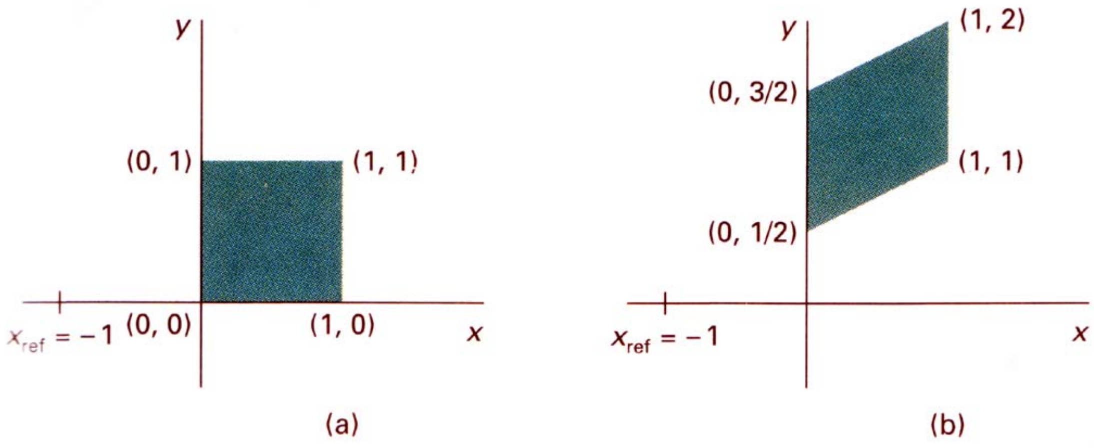
กำหนดสัมประสิทธิ์การเฉือนเท่ากับ 0.5 และค่า y อ้างอิงเท่ากับ -1

General Shear (Y)

สำหรับการเฉือนในแนวแกน y สามารถพิจารณาได้ในทำนองเดียวกัน โดยการเพิ่มแรงเฉือนคงที่เทียบกับจุดอ้างอิงบนแกน x

แกน y

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y x_{ref} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y + sh_y \cdot (x - x_{ref}) \\ 1 \end{bmatrix}$$



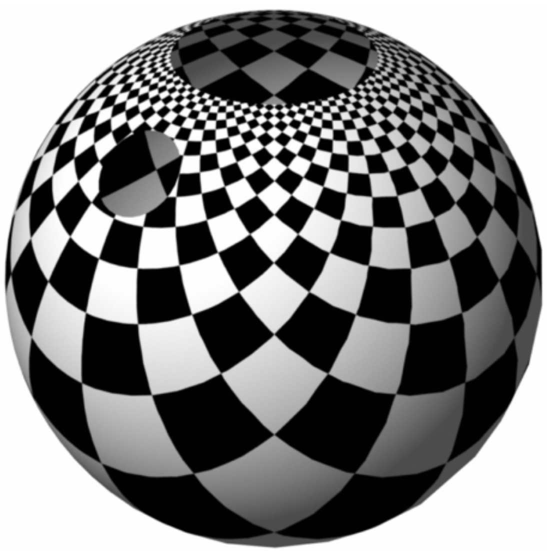
กำหนดสัมประสิทธิ์การเฉือนเท่ากับ 0.5 และค่า x อ้างอิงเท่ากับ -1

Suranaree University
of Technology

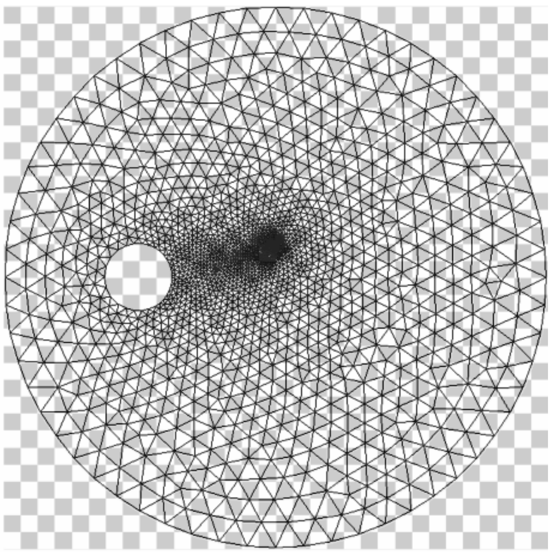
Coordinate Transformations

การทำ Transformation หรือ Parameterization ระหว่างระบบพิกัด แบ่งได้เป็นสองกรณี

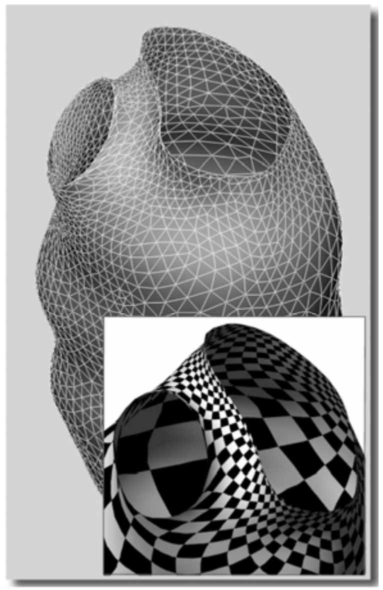
1) ระหว่างพิกัดต่างชนิด เช่น Cartesian กับ Polar Coordinate ซึ่งเหมาะสำหรับ Application เฉพาะกิจ เช่นการหากำลังของสายอากาศในการสื่อสารคลื่นสั้น หรือ ระหว่าง subset ของ Spherical Coordinate กับ Manifold ใดๆ สำหรับการกำหนดลวดลาย หรือคำนวณหาค่าฟังก์ชันอัตโนมัติ บนพื้นผิวสามมิติ



parametric domain



stereographic projection

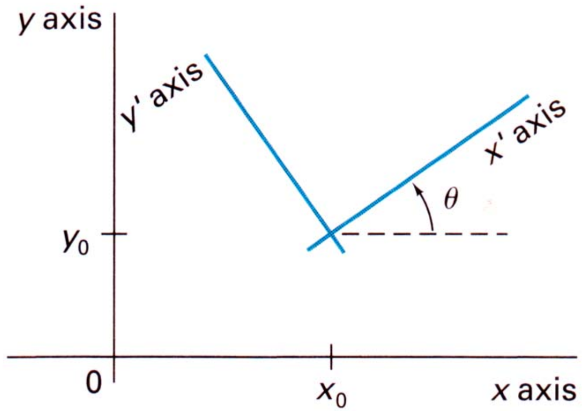


Conformal Map

Coordinate Transformations

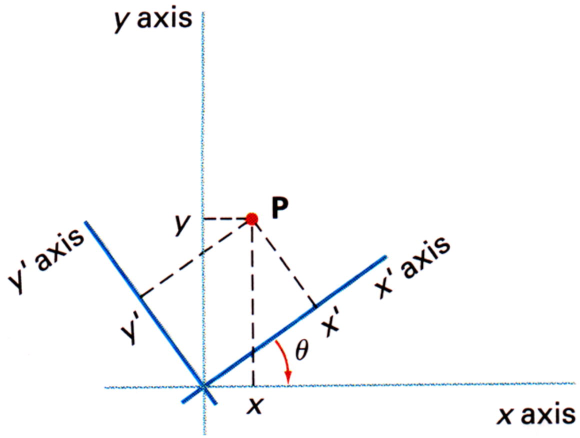
2) ระหว่างพิกัดชนิดเดียวกัน เช่น Cartesian ซึ่งเหมาะสำหรับ Application การออกแบบทางด้านกราฟิก เช่นมีวัตถุต้นแบบที่สร้างจาก local coordinate ต้องการนำมาวางบนฉากหรือ world coordinate ที่ตำแหน่งต่างๆ และมุมมองที่ต่างกันการแปลง coordinate (x′, y′) ไปซ้อนกับ (x, y)

ทำได้โดย เลื่อนจุดกำเนิด (x0, y0) ของ (x′, y′) ไปยังจุดกำเนิด (0, 0) ของ (x, y) ตามด้วยหมุนแกน x′ ไปซ้อนกับแกน x



$$T(-x_0, -y_0) = \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

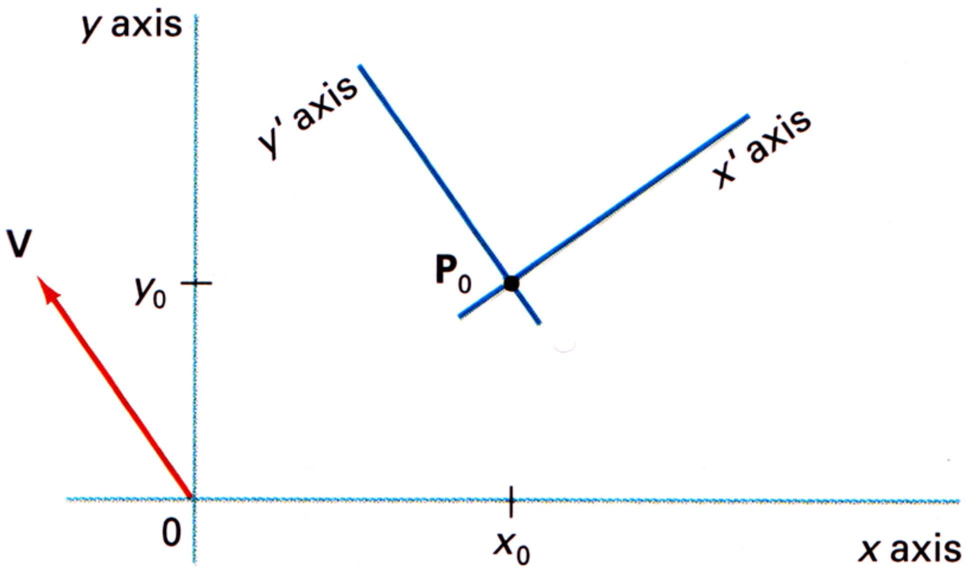
$$R(-\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Suranaree University of Technology

Alternative Method

การแปลงระบบพิกัดอีกวิธีหนึ่ง ทำได้โดย เลื่อนจุดกำเนิด (x0, y0) ของ (x′, y′) ไปยังจุดกำเนิด (0, 0) ของ (x, y) เหมือนวิธีแรก



นิยามเวกเตอร์ **V′** ซึ่งระบุทิศทางในแกน y′ ในทิศทางเป็นบวก และนิยามเวกเตอร์ **U′** ซึ่งระบุทิศทางในแกน x′ ในทิศทางบวก แล้วคำนวณ unit เวกเตอร์ **v′**= **V′**/|**V′**| และ **u′**= **U′**/|**U′**| แทนค่า u = (u_x, u_y) และ v = (v_x, v_y) ใน sub-Matrix มุมบนซ้ายสุด

ผลลัพธ์ที่ได้จะเป็น Rotation Matrix เดียวกัน

$$\mathbf{R}(\mathbf{u},\mathbf{v}) = \begin{bmatrix} u_x & u_y & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Affine Transformation

คือ Transformation ซึ่งเป็น superset ของ การแปลงทั้งหมดที่กล่าวมา ซึ่งแสดงในรูปของ Matrix ได้ดังนี้

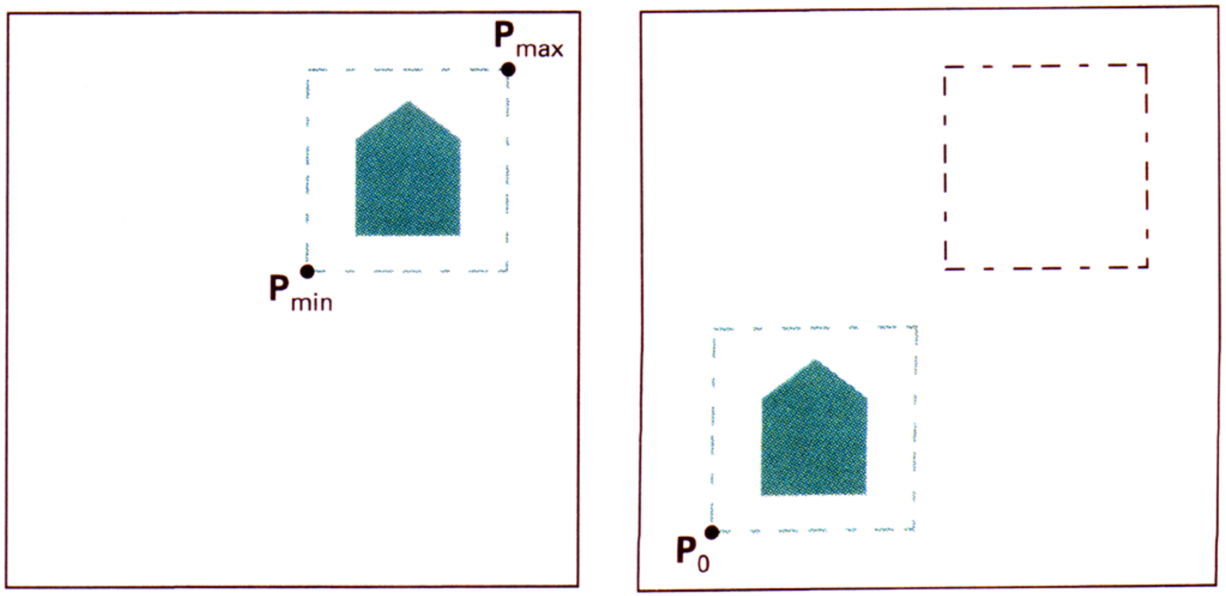
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{xx} & a_{xy} & b_x \\ a_{yx} & a_{yy} & b_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

คุณสมบัติของ Affine Transformation ดังต่อไปนี้

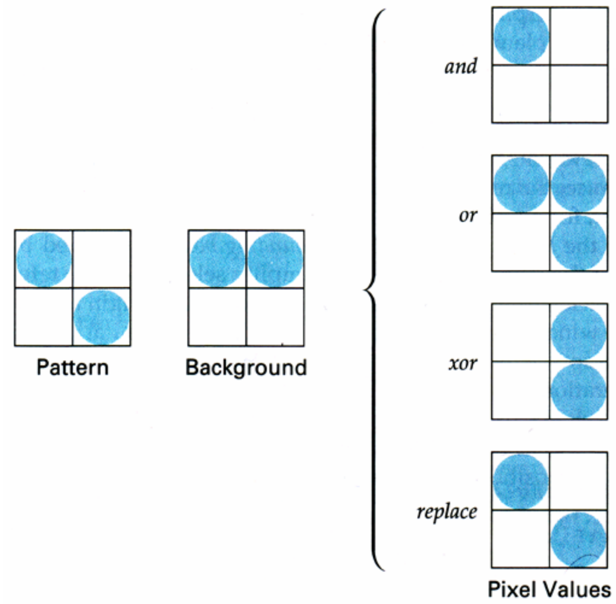
- การแปลงเส้นขนานยังคงเป็นเส้นขนาน
- จุดที่มีขนาดจำกัดแปลงเป็นจุดที่มีขนาดจำกัด
- สามารถอธิบายได้ด้วย การต่อกันของ Matrix ที่ได้จากการ เลื่อน หมุน ย่อ/ขยาย สะท้อน และ เฉือน
- Affine ที่เกิดจาก การเลื่อน หมุน และสะท้อน จะสงวนมุมระหว่างเส้นโค้ง (Conformal) และความยาว (Metrics)

Raster Methods

เนื่องจากอุปกรณ์แสดงผลที่เราพิจารณาเป็นชนิด Raster ซึ่งประกอบด้วยจุดภาพไม่ต่อเนื่อง ดังนั้นบางกรณี เราสามารถเพิ่มประสิทธิภาพของการคำนวณการแปลงวัตถุได้ โดยอาศัยคุณสมบัติของ Frame Buffer



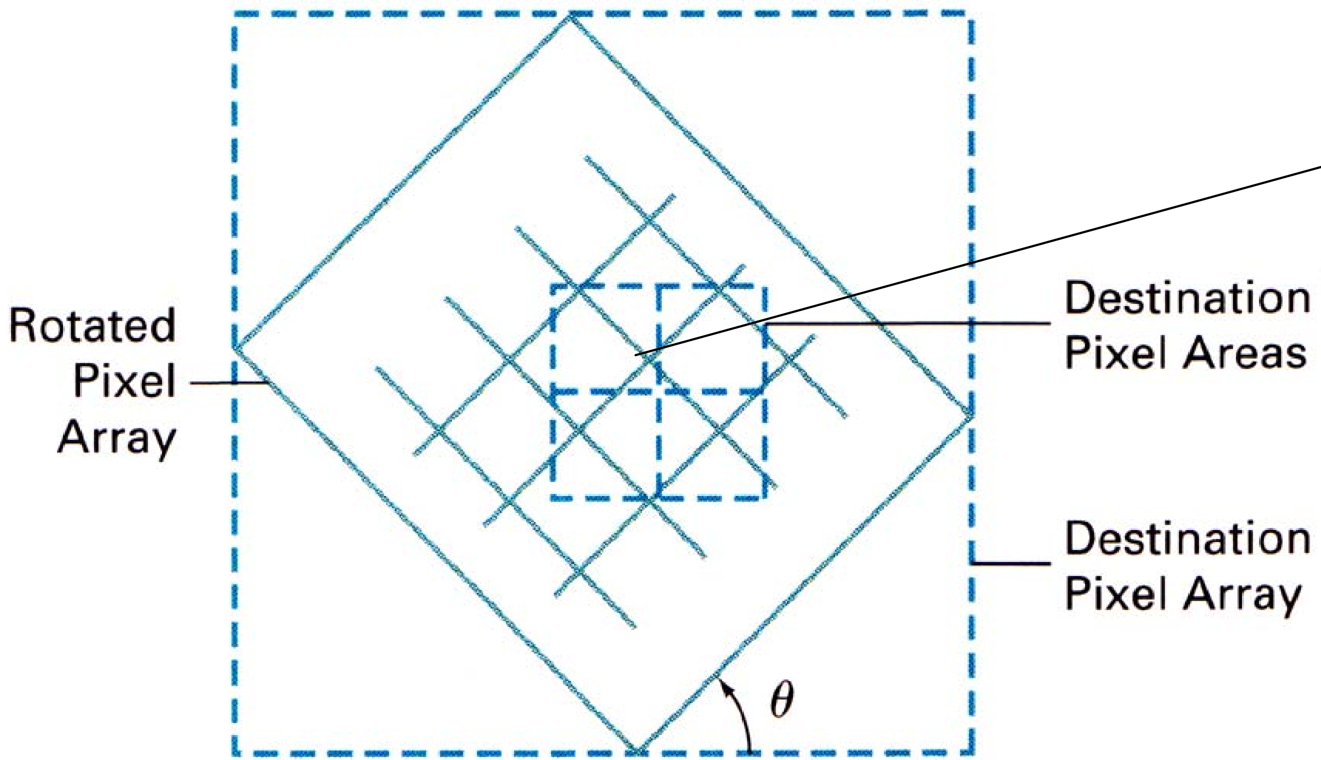
นำ Binary Operator มาใช้เพื่อรวมวัตถุกับพื้นหลัง

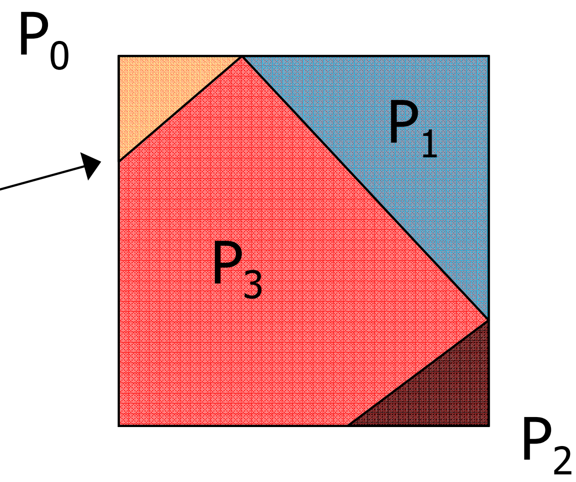


การเลื่อนวัตถุดังรูปใช้ฟังก์ชันของหน่วยความจำ BitBlt (Bit-Block Transfer) ทำให้ทำงานได้เร็วขึ้น

Arbitrary Raster Rotations

สำหรับการหมุนวัตถุ เพื่อหลีกเลี่ยงปัญหา Alias เราจะต้องหาค่าความเข้มของจุดภาพปลายทาง โดยคำนวณจากค่าเฉลี่ยของจุดภาพต้นฉบับ ที่ซ้อนทับจุดที่พิจารณา ถ่วงน้ำหนักกับร้อยละของบริเวณที่ซ้อนทับ





$$I = \sum_k A(P_k) \cdot I(P_k)$$

Conclusions

- 2D Geometric Transformations
  - Basic Transformations
  - Matrix Representation and Homogeneous Coordinates
  - General Composite Transformations
  - Reflection and Shear Transformation
  - Coordinate Transformations
  - Affine Transformation
  - Raster Methods for Transformation
- 2-Dimensional Viewing
  - The viewing Pipeline
  - Viewing Coordinate Reference Frame
  - Window to Viewport Coordinate Transformation
  - Practical Viewing Example