COMPUTER GRAPHICS
School of Computer Engineering

Suranaree University
of Technology

Lecture 3 Review

Paramate Horkaew

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

Previous Lecture

- Frame Buffer Class
- Simple Graphic Primitive Algorithms
  - Line Drawing Algorithms (Direct v.s. Integer Arithmetic)
  - Circle Drawing Algorithms
  - Rasterization of Arbitrary Curves
  - Polynomial Curve and Spline Drawing Algorithms
- Filled-Area Primitives
  - Polygon Filling
  - Flood-Fill Algorithm
  - Inside-Outside Tests
- Output Primitive Attributes
- Picture Approximation using Halftone

Homework

กำหนดให้จุดปลายทั้งสองจุดของเส้นตรงเป็น (20, 10) และ (30, 16) จง

1) คำนวณความชันของเส้นตรง
2) หาสมการของเส้นตรง
3) ใช้ DDA Algorithm ในการหาพิกัดจุดลำดับที่ k
4) หาค่าตัวแปรตัดสินใจที่ตำแหน่งเริ่มต้น (p_0)
5) ใช้ Bresenham's Algorithm ในการหาค่าตัวแปรตัดสินใจ (p_k) และพิกัดจุดลำดับที่ k

กำหนดให้ k = 0 ถึง 9 แจกจางค่าผลลัพธ์ที่ได้ในตาราง โดยใช้ตารางของผลจาก ข้อ 3 และ ข้อ 5 แยกกัน

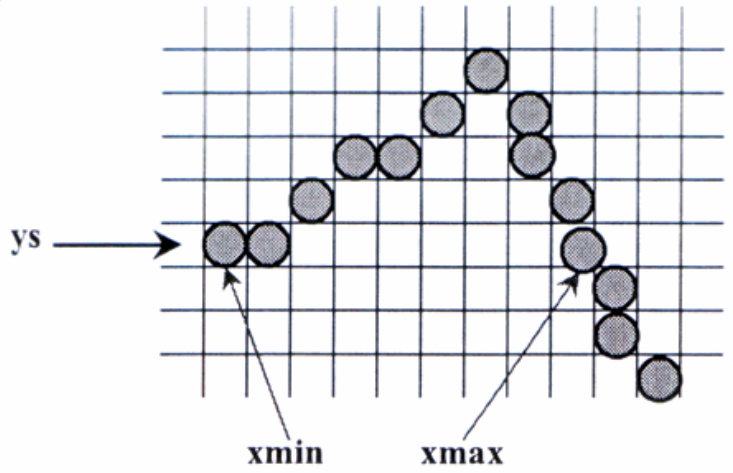
(เฉลยการบ้าน)

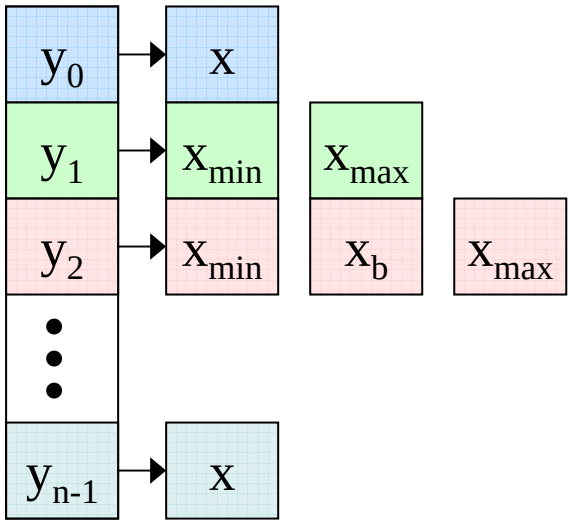
Filled Area Primitives

Simple Polygon Filling Algorithm

วิธีการที่ง่ายที่สุด ของการระบายรูปหลายเหลี่ยม คือ Line Scan Algorithm

1. วาดรูป Polygon เชื่อมต่อจุดโดยใช้ เทคนิค การวาดเส้นตรง
2. จัดเก็บจุดที่ประกอบเป็นขอบของ Polygon (หรือ Edge Pixels)
3. เรียงจุดดังกล่าว ใน Array ตามลำดับพิกัดในแนวตั้ง (y) จากน้อยไปมาก
4. สำหรับพิกัด y แต่ละค่า ถ้ามี จุดเดียวแสดงว่าเป็นจุดยอดบนสุด หรือ ล่างสุด ให้ระบายสีจุดนั้น
5. มิฉะนั้น ระบายสีระหว่างจุดที่มี x มากและ น้อยที่สุด ดังรูป



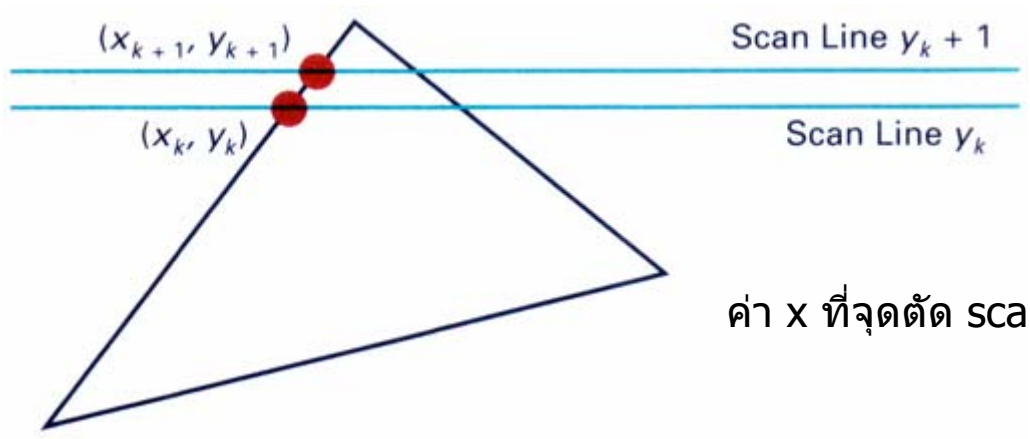


Exceptions? (Self Review Please)

Coherent Polygon Processing

เราสามารถใช้ประโยชน์ ความเกี่ยวเนื่องกัน (coherence) ขององค์ประกอบ polygon สำหรับ scan line ที่ติดกัน มาใช้เพิ่มประสิทธิภาพในการประมวลผลได้

ตัวอย่างเช่น ในกรณีนี้ สังเกตว่า ความชัน m ของเส้นขอบ ของ polygon (edge) มีค่าคงที่ ระหว่าง scan line ที่ติดกัน (ดังรูป) ซึ่งสามารถเขียนความสัมพันธ์ได้ว่า



$$m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$$

since $y_{k+1} - y_k = 1$

ค่า x ที่จุดตัด scan line ถัดไป $x_{k+1} = x_k + \frac{1}{m}$

เขียนในรูปของสมการเส้นขอบ $x_{k+1} = x_k + \frac{\Delta x}{\Delta y}$

An Efficient Polygon Algorithm

จากสมการหาพิกัด x ของจุดตัดระหว่าง scan line ที่ k กับ polygon

$$x_{k+1} = x_k + \frac{\Delta x}{\Delta y}$$

Increment **y** (k^{th} scan line), determine whether the next **x** should be increased

เราสามารถหาคำตอบแบบวนซ้ำ โดยใช้การคำนวณแบบ **จำนวนเต็ม** ได้โดย

- กำหนดให้ counter เริ่มต้นเป็น 0
- ทุกครั้งที่เลื่อนไปพิจารณา scan line ถัดไป k = k + 1 ให้เพิ่มค่า counter ไป Δx
- ถ้า counter ที่ได้มีค่ามากกว่า Δy เราจะเพิ่มค่าจุดตัด x ไป 1 (นั่นคือ พจน์เศษส่วนข้างหลังมีค่า หลังจาก **ตัดเศษ** เป็นจำนวนเต็มเท่ากับ 1) แล้วลดค่า counter ลง Δy
- วนซ้ำขั้นตอนที่ 2 และ 3 สำหรับ scan line ถัดไปสำหรับเส้นขอบ polygon ที่พิจารณา

Precise Integer Arithmetic

เราสามารถหาคำตอบที่แม่นตรงยิ่งขึ้นโดยใช้การ **ปัดเศษ** แทนการ **ตัดเศษ**

- กำหนดให้ counter เริ่มต้นเป็น 0
- ทุกครั้งที่เลื่อนไปพิจารณา scan line ถัดไป k = k + 1 ให้เพิ่มค่า counter ไป $2\Delta x = (\Delta x + \Delta x)$
- ถ้า counter ที่ได้มีค่ามากกว่า Δy เราจะเพิ่มค่าจุดตัด x ไป 1 (นั่นคือ พจน์เศษส่วนข้างหลังมีค่า หลังจากตัดเศษ เป็นจำนวนเต็มเท่ากับ 1) แล้วลดค่า counter ลง $2\Delta y = (\Delta y + \Delta y)$
- วนซ้ำขั้นตอนที่ 2 และ 3 สำหรับ scan line ถัดไปสำหรับเส้นขอบ polygon ที่พิจารณา

ขั้นตอนวิธีดังกล่าว เทียบได้กับ การเปรียบเทียบค่าที่เพิ่มขึ้น Δx กับ $\Delta y/2$ (มากกว่า 0.5 ปัดขึ้น) ดังนั้นถ้า m=7/3 ค่า counter สำหรับ k แรกๆ จะเป็นดังนี้ 0, 6, 12 (ลดลงเป็น 12-2*7 = -2), 4, 10 (ลดลงเป็น -4), ... ซึ่งจะได้ว่าค่า x จะเป็น $x_0 + 0$, $x_0 + 0$, $x_0 + 1$, $x_0 + 1$, $x_0 + 2$, ...

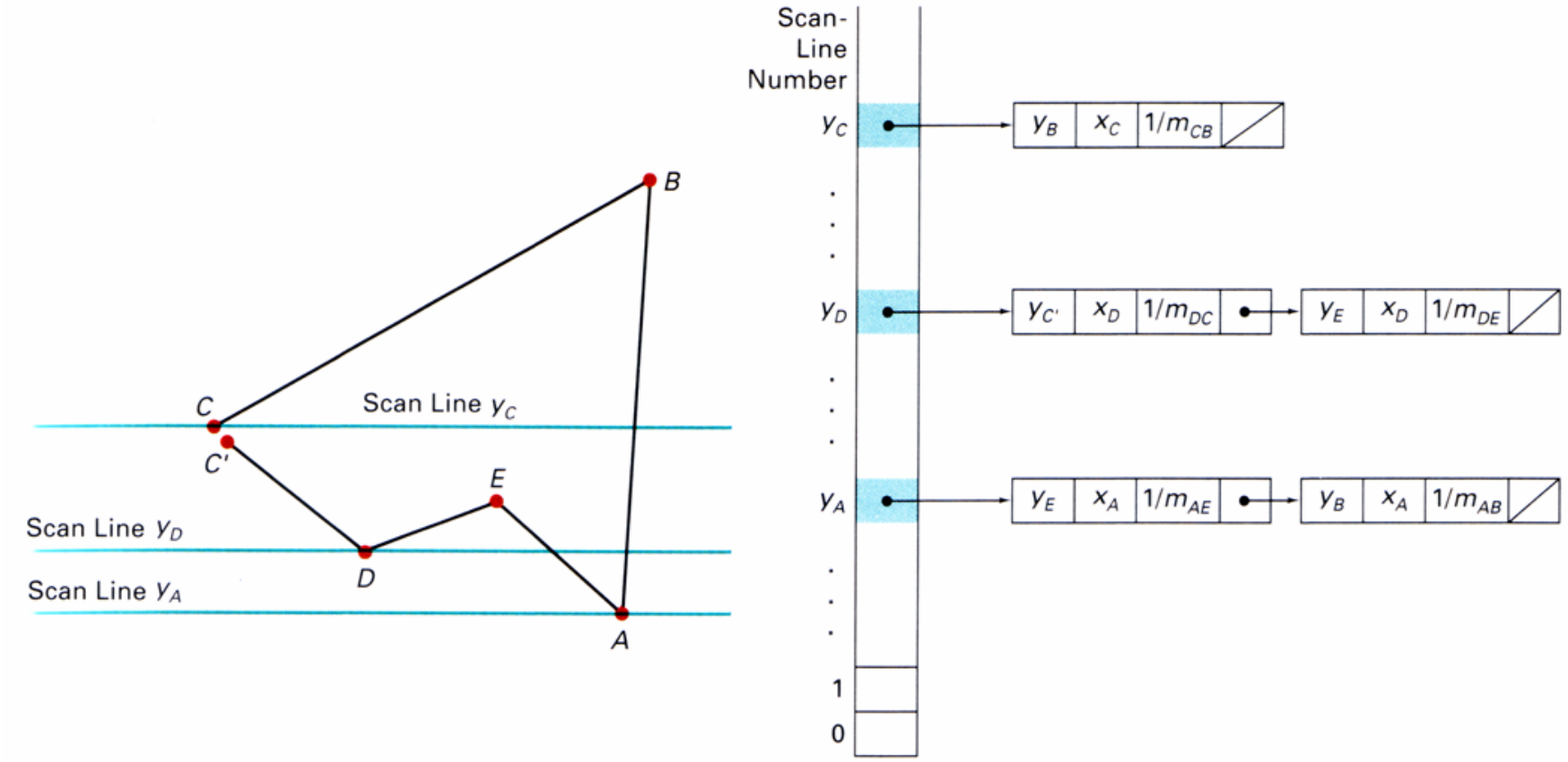
Implementation of the Algorithm

- เดินไปตามขอบ (edge) ของ polygon แต่ละเส้นในทิศทางทวนเข็ม (หรือตามเข็ม) นาฬิกา
- ทำการ shorten edge ในกรณีที่จุดยอด (vertex) ตัดผ่าน scan line พอดีเพื่อแยก แยะชนิดของจุดแบ่งเส้น scan line
- จัดเก็บแต่ละ edge ไว้ในตาราง edge โดยเรียงตามลำดับ พิกัด y ที่น้อยที่สุด (จุดยอดล่างสุดของ edge)
- ในแต่ละช่อง ของตาราง ใส่สมการของ edge ได้แก่ ค่า y ที่มากที่สุด (จุดบนสุด), จุดตัด x ของจุดล่างสุด และค่า 1/m
- สำหรับแต่ละ scan line เรียงสมการตาม ลำดับ ของค่าจุดตัด x จากซ้ายไปขวา

จากโครงสร้างข้อมูลดังกล่าว ไล่ลำดับ scan line จากจุดล่างสุดของ polygon ไปจุดบนสุด เพื่อสร้าง active edge list ซึ่งนำไปหาค่า x ของจุดตัด เพื่อนำไปสู่ขั้นตอน scan line polygon filling algorithm ต่อไป

Implementation Diagram

แผนผังแสดง โครงสร้างข้อมูล ของ integer arithmetic ของ scan line algorithm ซึ่งประกอบด้วย sorted edge table และ active edge list



Nonzero Winding Number Rules

การทดสอบว่าจุดที่กำหนดอยู่ภายในหรืออยู่ภายนอก เส้นโค้งปิด (closed curve หรือ contour) ซึ่งอาจจะเป็นแบบ self intersection หรือไม่ก็ได้

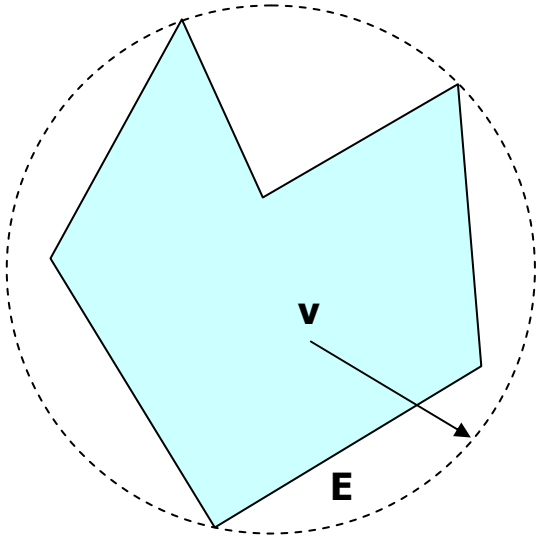
Algorithm

- จากจุดที่ต้องการทดสอบ ลาก vector **v** ให้ตัดผ่าน polygon แต่ไม่ผ่านจุดยอดมุมใดๆ ไปยังระยะอนันต์ กำหนดค่า winding counter เป็น 0
- หาผลคูณ cross product (ในระนาบ x, y) ระหว่าง vector **v** กับ edge **E** ที่ตัดผ่าน
- ผลลัพธ์ที่ได้จะเป็น องค์ประกอบ z ตั้งฉากกับระนาบ x, y พิจารณาได้ 2 กรณี
    - มีค่าเป็นบวก ให้นับค่า counter **เพิ่มขึ้น** 1
    - มีค่าเป็นลบ ให้นับค่า counter **ลดลง** 1
- ถ้าผลลัพธ์สุดท้ายหลังจากพิจารณาทุก edge แล้วจำนวน counter ไม่เท่ากับ 0 จุดที่ทดสอบเป็นจุดภายใน มิฉะนั้น จุดดังกล่าวเป็นจุดภายนอก

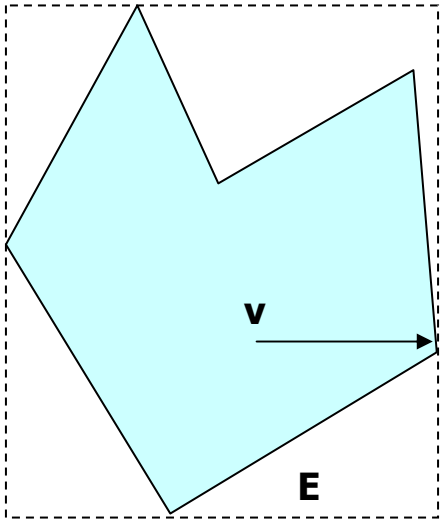
Implementation Technique

การพิจารณาว่า vector ที่นิยามโดยสมการเส้นตรงสองเส้น ตัดกันหรือไม่ ที่จุดใด นั้น สามารถทำได้โดย http://astronomy.swin.edu.au/~pbourke/geometry

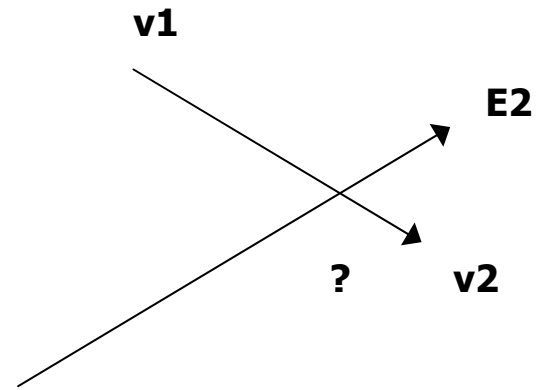
Techniques เนื่องจากค่าอนันต์สำหรับสร้าง vector **v** ไม่มีอยู่จริง แต่สามารถสร้างค่าสมมติได้โดยลากเส้นจากจุดที่กำหนด ไปยังจุดขอบสุดของ polygon



Bounding Circle

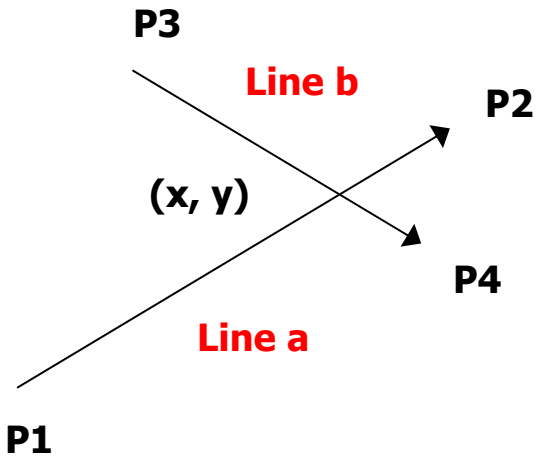


Bounding Box



$$\begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} = \begin{bmatrix} (\mathbf{v}_2 - \mathbf{v}_1)_x \\ (\mathbf{v}_2 - \mathbf{v}_1)_y \\ (\mathbf{v}_2 - \mathbf{v}_1)_z \end{bmatrix} \times \begin{bmatrix} (\mathbf{E}_2 - \mathbf{E}_1)_x \\ (\mathbf{E}_2 - \mathbf{E}_1)_y \\ (\mathbf{E}_2 - \mathbf{E}_1)_z \end{bmatrix}$$

Intersection of Two Vectors



Line Equations

$$\mathbf{P}_a = \mathbf{P}_1 + u_a(\mathbf{P}_2 - \mathbf{P}_1)$$
$$\mathbf{P}_b = \mathbf{P}_3 + u_b(\mathbf{P}_4 - \mathbf{P}_3)$$

ที่จุดตัด $\mathbf{P}_a = \mathbf{P}_b = (x, y)$

$$x_1 + u_a(x_2 - x_1) = x_3 + u_b(x_4 - x_3)$$
$$y_1 + u_a(y_2 - y_1) = y_3 + u_b(y_4 - y_3)$$

จัดข้างสมการเพื่อหาค่า u_a และ u_b

$$u_a = \frac{(x_4 - x_3)(y_1 - y_3) - (y_4 - y_3)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)}$$

$$u_b = \frac{(x_2 - x_1)(y_1 - y_3) - (y_2 - y_1)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)}$$

เส้นตรงสองเส้นตัดกัน ก็ต่อเมื่อ u_a และ u_b มีค่าอยู่ระหว่าง 0 และ 1

หาจุดตัดโดยแทนค่า u ไปในสมการแรก

Fill of Curved Boundary Areas

Algorithm ประเภท scan line สำหรับ บริเวณที่มีขอบเขตเป็นเส้นโค้ง จะมีความซับซ้อน มากกว่า บริเวณที่เป็น polygon (ยกเว้นในกรณีพิเศษที่ บริเวณเป็นส่วนหนึ่ง ของภาคตัดกรวย เช่น วงกลม หรือ วงรี)

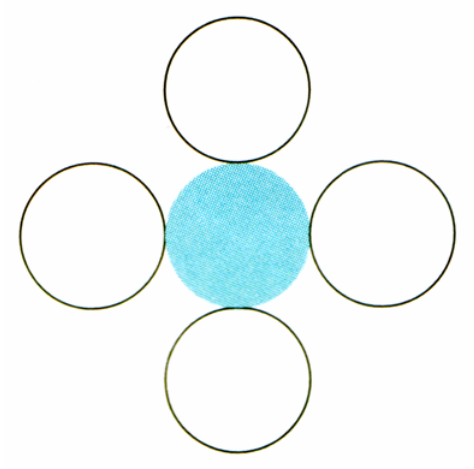
Boundary-Fill Algorithm

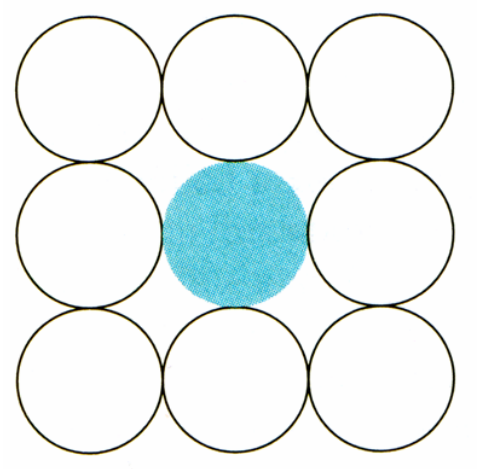
เป็นขั้นตอนวิธีที่สามารถสร้างได้ง่าย และใช้กันมาก ในระบบกราฟิกแบบโต้ตอบ (Interactive Graphics) ซึ่งต้องการ input จากผู้ใช้ คือจุดเริ่มต้นภายในบริเวณ แล้วระบายสี เริ่มจากจุดที่กำหนด แล้วแผ่กระจายออกไป จนกระทั่งไปสิ้นสุดที่ **ขอบ** ของบริเวณ

Flood-Fill Algorithm

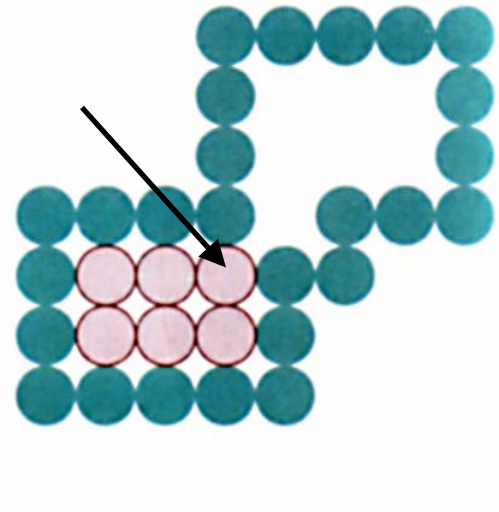
คล้ายกับวิธี Boundary Fill แต่ว่าเงื่อนไขคือ เปลี่ยนสี จุดภาพที่ กำหนดว่าเป็น สีภายในบริเวณ ให้เป็นสีที่ต้องการระบาย

Testing Neighboring Pixels



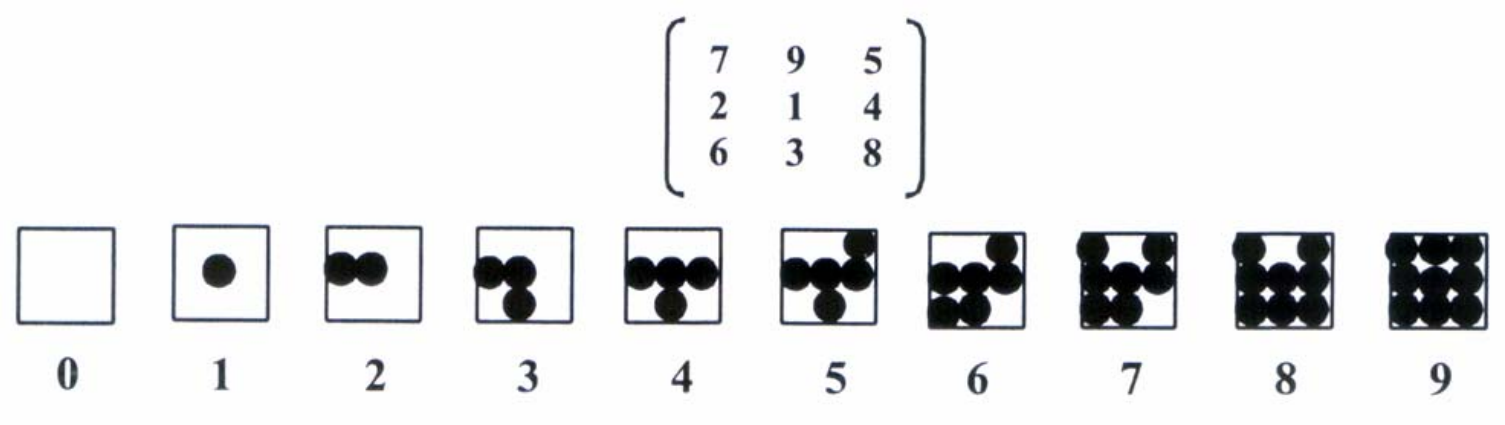


ภาพด้านซ้ายคือ แผนผังแบบ 4-connected test ในขณะที่แผนผังด้านขวามือ คือรูปแบบ 8-connected test ซึ่งรวมเอาจุดทแยงมุมทั้งสี่ไว้



ถ้าจุดปัจจุบันที่ทำการทดสอบคือจุดที่ศรชี้ การทดสอบแบบ 4-connect จะหยุดที่กรอบด้านล่างในขณะที่การทดสอบแบบ 8-connect จะแผ่ไปถึงกรอบด้านบนด้วย

Picture Approximation by Dithering



Original

Dithered Image

COMPUTER GRAPHICS
School of Computer Engineering

Suranaree University
of Technology

Lecture 4 Refined Raster Algorithms and Geometric Transformations (Part I)

Paramate Horkaew

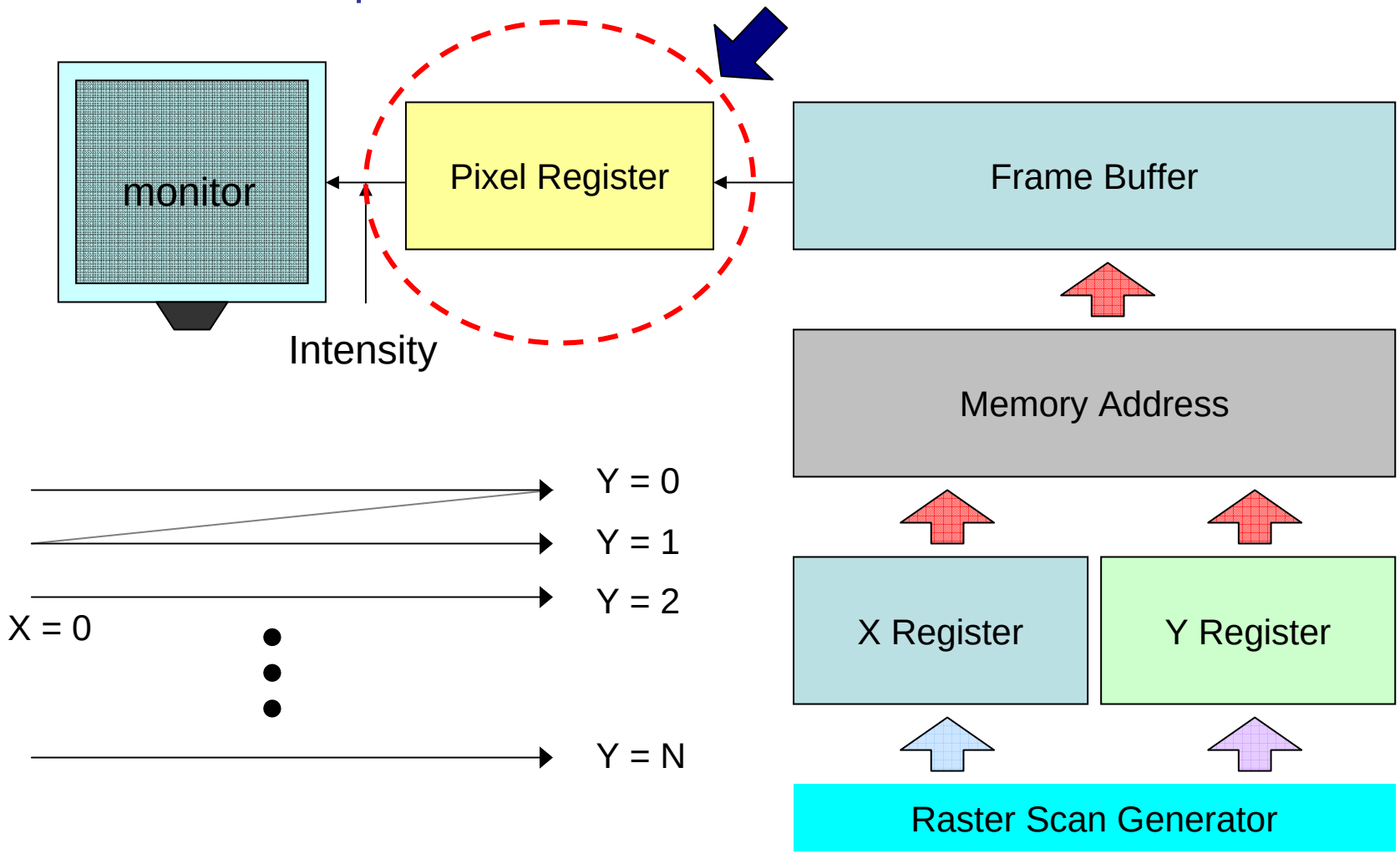
School of Computer Engineering, Institute of Engineering
Suranaree University of Technology

Lecture Outline

- Color and Grayscale Levels
- Polygon Filling Styles
  - Pattern Tilling
  - Pattern Blending: Multiple Transparent Layers
- Text Attributes *(Self Learning)*
- 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

Review of The Raster System

Video Controller Operations



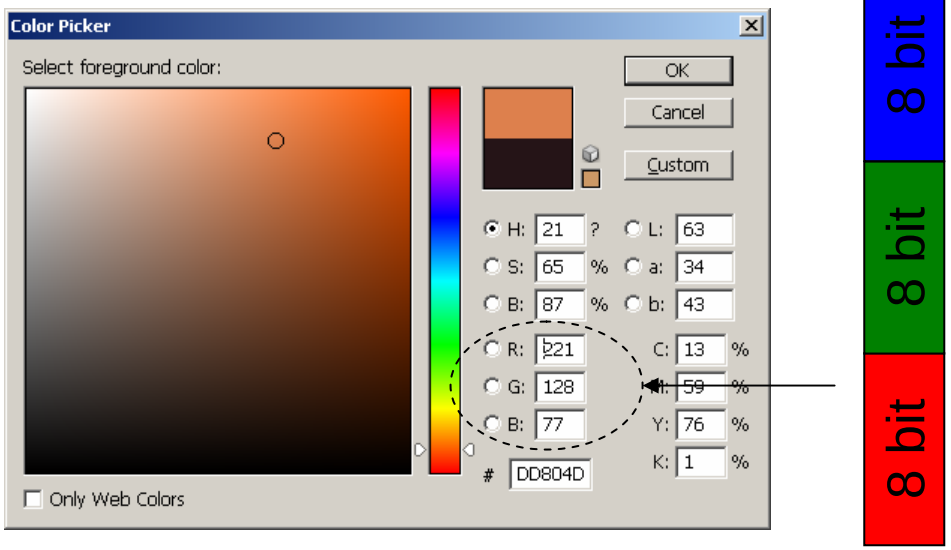
Color Representations

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

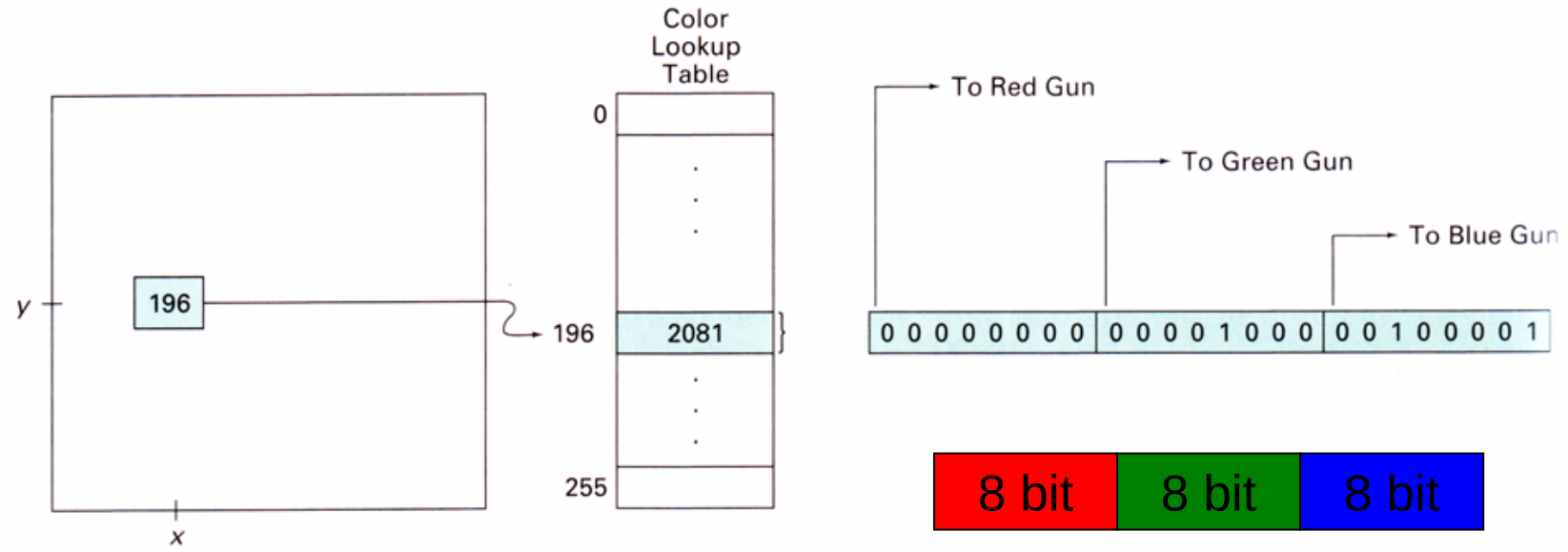
Color	*Stored Color Values in Frame Buffer*			*Displayed Color*
Code	*RED*	*GREEN*	*BLUE*	
0	0	0	0	Black
1	0	0	1	Blue
2	0	1	0	Green
3	0	1	1	Cyan
4	1	0	0	Red
5	1	0	1	Magenta
6	1	1	0	Yellow
7	1	1	1	White



Color Lookup Tables

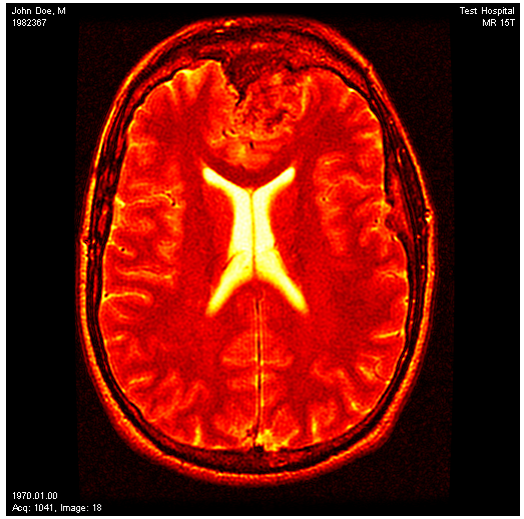
ในการกำหนดสีแบบ Direct จะใช้หน่วยความจำขนาดใหญ่ หากต้องการแสดงภาพสี ขนาด 1024x1024 แบบ Full Color (24 bpp, 8 bpc) จะใช้หน่วยความจำขนาด 3 Megabytes เพื่อจัดเก็บ Frame Buffer

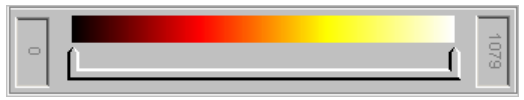
อีกทางเลือกหนึ่งที่ประหยัดกว่าคือ ให้แต่ละจุดภาพใน Frame Buffer เก็บเฉพาะดัชนี ซึ่งชี้ไปยังตารางสี ดังรูป



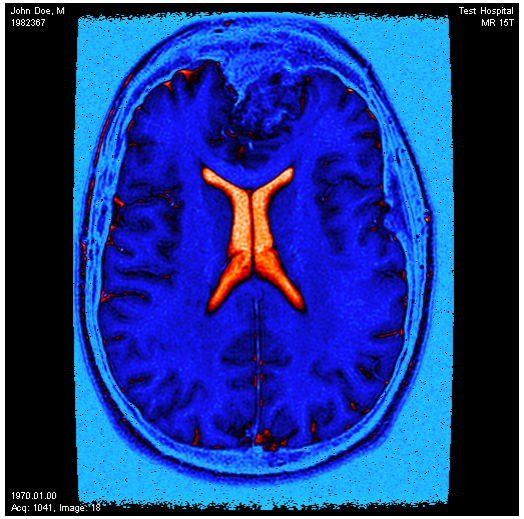
Lookup Tables in Practice

โปรแกรมประยุกต์ ได้นำเทคนิคการแสดงผลภาพกราฟิกแบบสี ด้วยการเปิดตาราง ตัวอย่างเช่น เพื่อการวินิจฉัยทางการแพทย์ (Osiris 4.18) ดังรูป

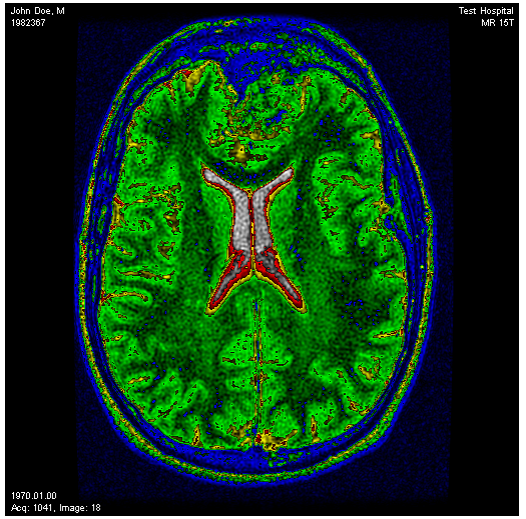


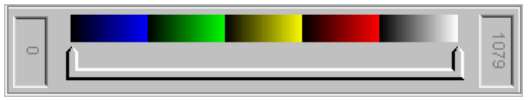


Black Body



Flow





Five Ramps

Grayscale

ระบบการแสดงผลบางชนิด ไม่สามารถ (หรือไม่จำเป็นที่) จะแสดงผลแบบสี และอาจจะเลือกแสดงผล แบบ ระดับความเทา (Shades of Gray) โดยแทนที่แต่ละระดับ ด้วยตัวเลขจำนวนจริงบวก ที่มีพิสัย ตั้งแต่ 0 ถึง 1 ระบบ Raster จะแปลงตัวเลขดังกล่าว ให้เป็นเลขฐานสองที่เหมาะสม เพื่อจะทำการจัดเก็บในหน่วยความจำ Frame Buffer ต่อไป

ระบบแสดงผลแบบ Grayscale มักจองหน่วยความจำสำหรับแต่ละ Pixel ให้มีขนาด 8 บิต (256 ระดับ สำหรับงานทั่วไป), 16 บิต (65536 ระดับ สำหรับงานด้านการแพทย์) หรือ 24 บิตขึ้นไป (สำหรับงานวิจัยวิทยาศาสตร์/ดาราศาสตร์)

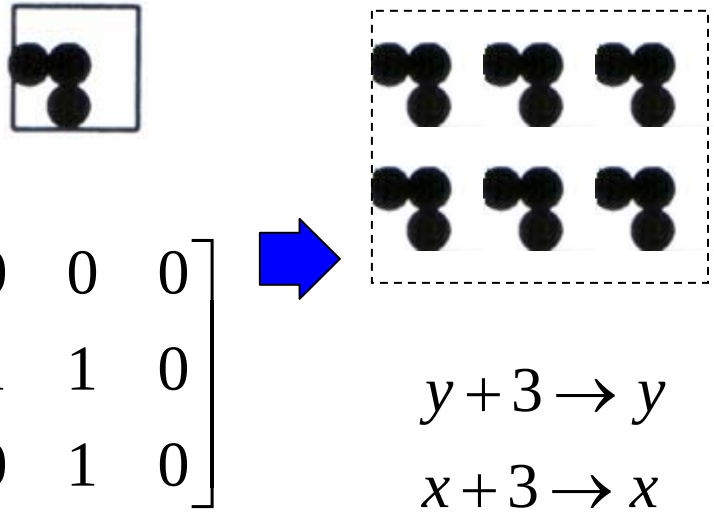
Intensity Codes	*Stored Intensity Values In The Frame Buffer (Binary Code)*		*Displayed Grayscale*
0.0	0	(00)	Black
0.33	1	(01)	Dark gray
0.67	2	(10)	Light gray
1.0	3	(11)	White

$$ci = \lfloor g \times (2^N - 1) \rfloor$$

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



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

$$y+3 \rightarrow y$$

$$x+3 \rightarrow x$$

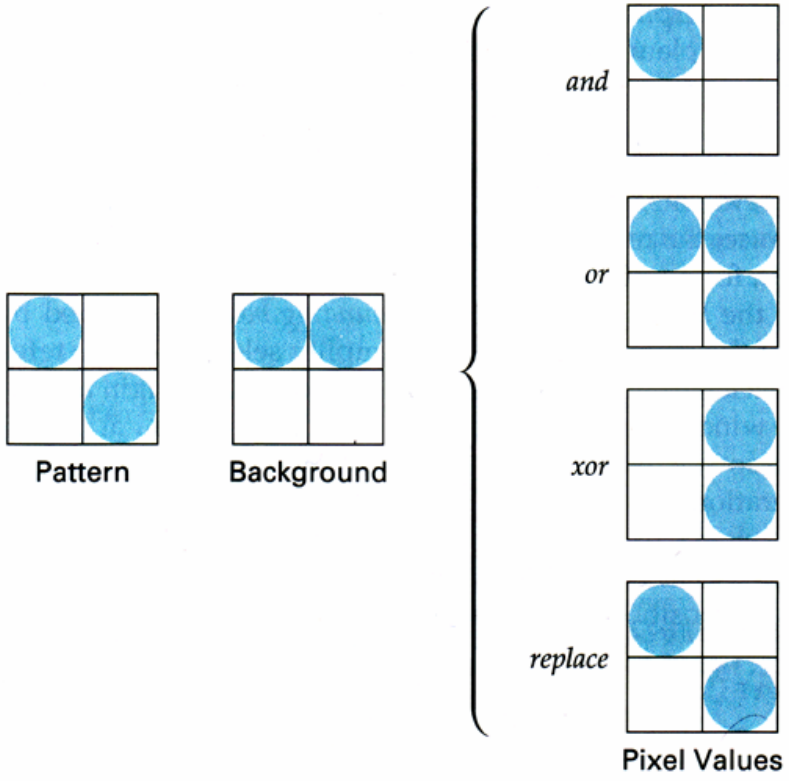
หมายเหตุ

1. เราสามารถกำหนดจุดเริ่มต้นของ mask ที่มุมบนสุดของจอภาพ หรือมุมบนสุดของ polygon ก็ได้

2. หากบางส่วนของ mask ตกออกนอกขอบ polygon ก็ให้ละเว้นส่วนนั้น ($mask_i$ AND $pixel_i$)

A. Binary Pattern Filling

ด้วยหลักการเดียวกัน เราสามารถระบาย polygon บนฉากหลัง (background) ที่ก็เป็น pattern ต่างๆ ได้ โดยใช้ Binary Operations ต่างๆ ได้



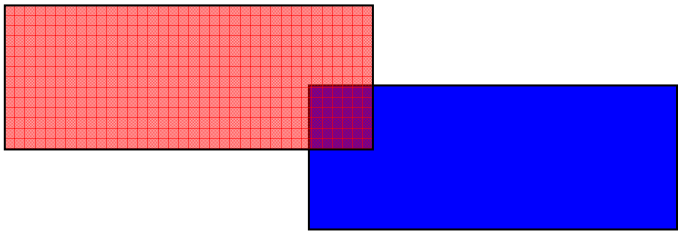
หมายเหตุ

1. วิธีการนี้เหมาะสำหรับระบบแสดงผลแบบ binary คือมีการกำหนดความเข้มเพียงสองระดับ คือ เข้ม (1) กับสว่าง (จุดว่าง) (0)

2. ผลลัพธ์ที่ได้จะทำให้ส่วนที่เป็น background ปรากฏผ่านส่วนที่เป็นช่องว่าง (0) ของ polygon

B. 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

การแสดงผลแบบ Raster นั้น สุ่มตัวอย่าง จุดภาพ (pixels) แบบไม่ต่อเนื่อง ด้วยอัตราที่จำกัด ทำให้ภาพที่ปรากฏผิดเพี้ยนไป ในลักษณะเป็นขั้นบันได ตามรูป

ลักษณะ Aliasing ที่เกิดขึ้น จากการสุ่มตัวอย่าง pixels ด้วยขนาดจำกัด

ทฤษฎีการสุ่มของ Nyquist ระบุว่า ความถี่ของการสุ่มตัวอย่างสัญญาณใดๆ จะต้องมีค่าเป็น 2 เท่าขององค์ประกอบความถี่ ที่สูงที่สุดของสัญญาณนั้น

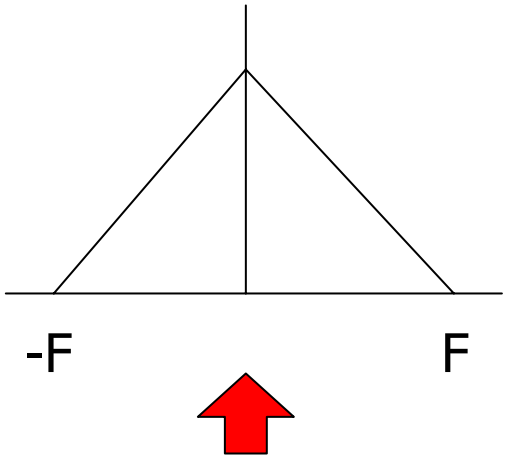
ดังความสัมพันธ์ $f_s \geq 2 f_{\max}$ หรือในรูปของระยะสุ่ม $\Delta x_s \geq \frac{\Delta x_{cycle}}{2}$

ค่าในอุดมคติดังกล่าว f_s และ x_s เรียกว่า Nyquist's Sampling Frequency และ Nyquist's Sampling Interval ตามลำดับ

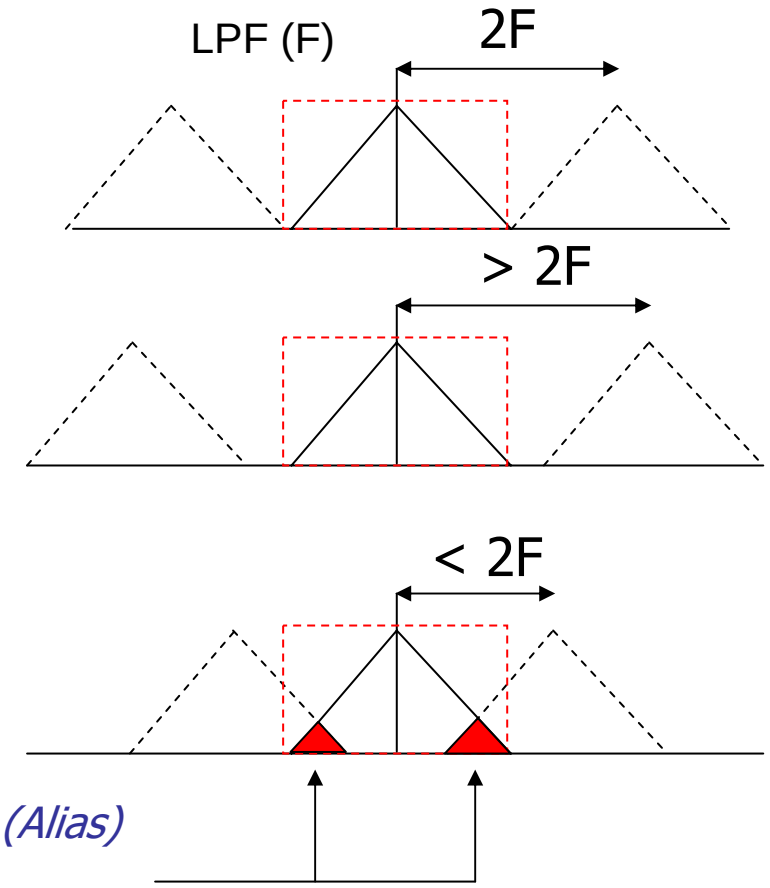
ถ้าความถี่ของจุดภาพต่ำกว่านี้เรียกว่า Under-sampling และ ผลลัพธ์ที่เกิดขึ้นเรียกว่า Aliasing (สัญญาณเงา) ซึ่งเรียกตามภาพที่ปรากฏใน Fourier Domain

Frequency Analysis

การเกิดสัญญาณเงา สามารถอธิบายได้ ด้วยการวิเคราะห์ Fourier ในกรณี ต่างๆ สำหรับสัญญาณสุ่มของสัญญาณใดๆ ดังแสดงในรูป



สัญญาณดั้งเดิม ที่มีความถี่สูงสุด เท่ากับ F



เกิดการรบกวนกันระหว่าง สัญญาณจริง กับสัญญาณเงา (Alias) ทำให้ไม่สามารถกู้สัญญาณเดิมกลับคืนมาได้โดย LPF

Anti-aliasing

การปรับปรุงคุณภาพของ การแสดงผลที่ผิดเพี้ยน อันเนื่องมาจาก Signal Alias เรียกว่า Anti-aliasing ซึ่งแบ่งออกได้เป็น 4 วิธี

- เพิ่มความละเอียดของจอภาพให้มากขึ้น ทว่าในการใช้งาน Graphics เรามักจะพูดถึง องค์ประกอบเรขาคณิตทางอุดมคติ เช่น จุด เส้นตรง และ เส้นโค้ง ซึ่งมีความถี่สูงสุดในทางทฤษฎีเป็นอนันต์ (เมื่อ $\Delta x \rightarrow 0$, $F \rightarrow \infty$) ยิ่งไปกว่านั้น เมื่อจอภาพละเอียดมากขึ้น ต้องใช้หน่วยความจำมากขึ้น และ อัตราการ refresh จอต่อจุดภาพสูงขึ้นอีกด้วย (ขึ้นอยู่กับข้อจำกัดทางเทคนิค)

✓ Super-sampling (Post Filtering) ใช้สำหรับอุปกรณ์แสดงผลที่สามารถแสดงจุดภาพได้หลายระดับความสว่าง โดยทำการสุ่มเทียมแต่ละจุดภาพให้ละเอียดขึ้น แล้วปรับค่าความเข้มของจุดภาพให้เหมาะสม จนเสมือนว่าภาพที่ได้ปรากฏต่อผู้สังเกตเรียบขึ้น (non-linear filter)

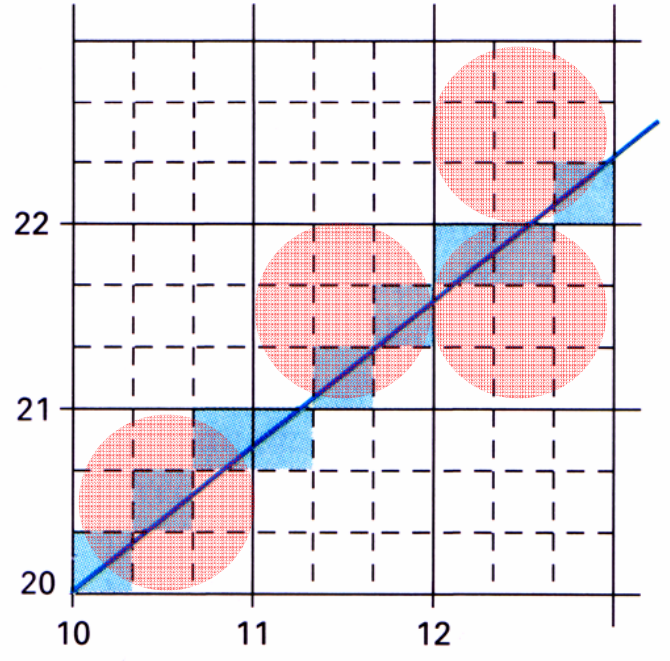
✓ Direct Filtering คือการกรองสัญญาณที่ผิดเพี้ยนด้วย LPF ที่ความถี่ต่ำผ่านน้อย (มีอัตราการลดทอนมากขึ้น ที่ความถี่สูง) เพื่อให้ส่วนที่เกิดปัญหา (Overlapping Area) ส่งผลต่อภาพน้อยลง ซึ่งจะทำให้เส้นที่ปรากฏเรียบขึ้น

- Pixel Phasing ปรับแต่งลำ electron ให้เบี่ยงเบนในระดับน้อยกว่าจุดภาพ

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

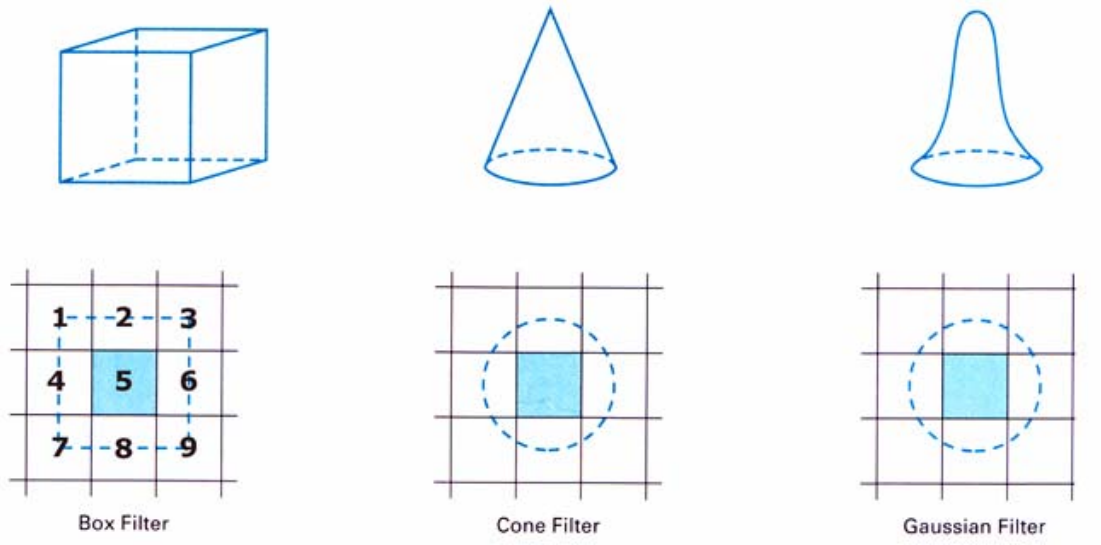
จุดที่ (10, 20) = 3, (11, 21) = (12, 21) = 2, (11, 20) = (12, 22) = 1, จุดอื่นๆ = 0

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)
3) กำหนดระดับความเข้มของจุดภาพจริง แปรผันตรงกับผลบวกถ่วงน้ำหนัก



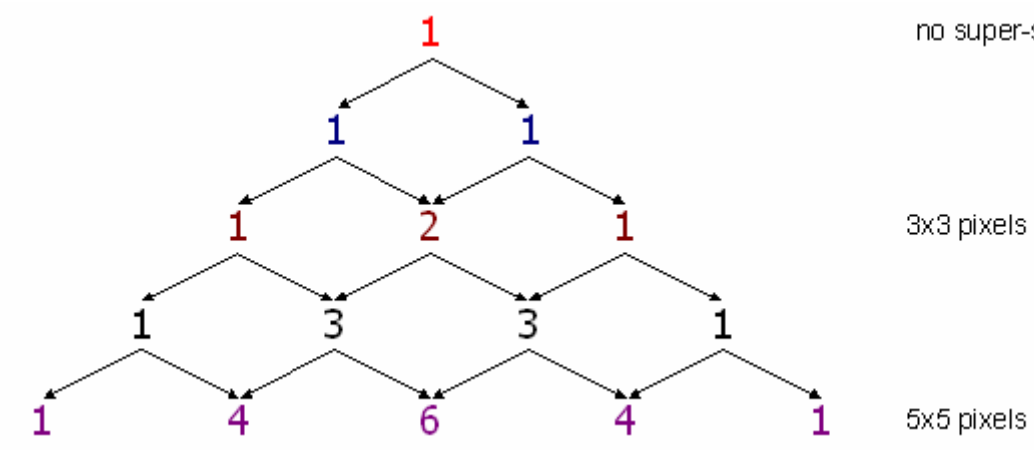
$$V = \sum_i \left(P_i \cdot M_i \right)$$

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

Gaussian Kernel Approximation

เพื่อประหยัดการคำนวณ Kernel ที่มีฟังก์ชันแบบ Gaussian สามารถประมาณได้ด้วย อนุกรม Binomial โดยใช้ Δ Pascal

$$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{x^2}{2\sigma^2}\right)$$



ผลลัพธ์ที่ได้ เป็นค่าถ่วงน้ำหนักใน 1 มิติ vector **K** จากการแบ่งจุดภาพย่อยจำนวนต่างๆ กัน เลือกระดับตามจำนวนจุดภาพที่ต้องการแบ่ง

คำนวณ K^TK

$$\begin{bmatrix}1\\2\\1\end{bmatrix}\begin{bmatrix}1 & 2 & 1\end{bmatrix}=\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$$

หาร Matrix ด้วยผลรวมภายในในที่นี้คือ 1 + 2 + 1 + 2 + 4 + 2 + 1 + 2 + 1 = 16 เพื่อให้ $\Sigma M = 1$

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

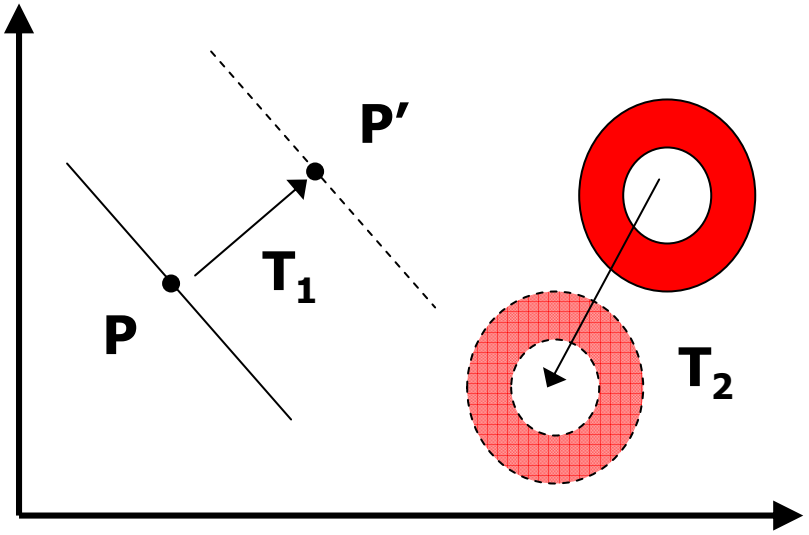
เราสามารถเลื่อน จุดใดๆ ในสองมิติ ได้โดยการบวก ระยะการเลื่อน (Translation Distances) ในรูปของเวกเตอร์ในแต่ละแกน (t_x *และ* t_y) จากจุดเดิม (x, y) ไปยังจุดใหม่ (x′, y′) ดังนี้

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

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

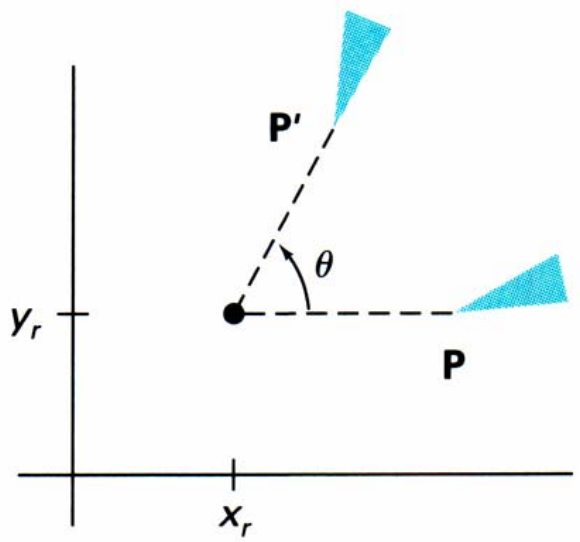
Rotation

การหมุน (Rotation) สามารถ นิยามได้ว่าคือการ **เปลี่ยนตำแหน่ง ของวัตถุไปในแนวเส้นโค้งของวงกลมในระนาบ (x, y)** จากตำแหน่งหนึ่ง ในปริภูมิ ไปยังอีกตำแหน่งหนึ่ง

สำหรับการหมุนวัตถุนั้น เราจะต้องกำหนดตัวแปรสองตัวได้แก่ 1) มุมที่ต้องการหมุน (θ) ซึ่งค่าเป็น + หมายถึงการหมุนทวนเข็มนาฬิกา (CCW) และค่าเป็น – หมายถึงการหมุนตามเข็มนาฬิกา (CW) และ 2) จุดศูนย์กลางการหมุน (rotation point หรือ pivot point)

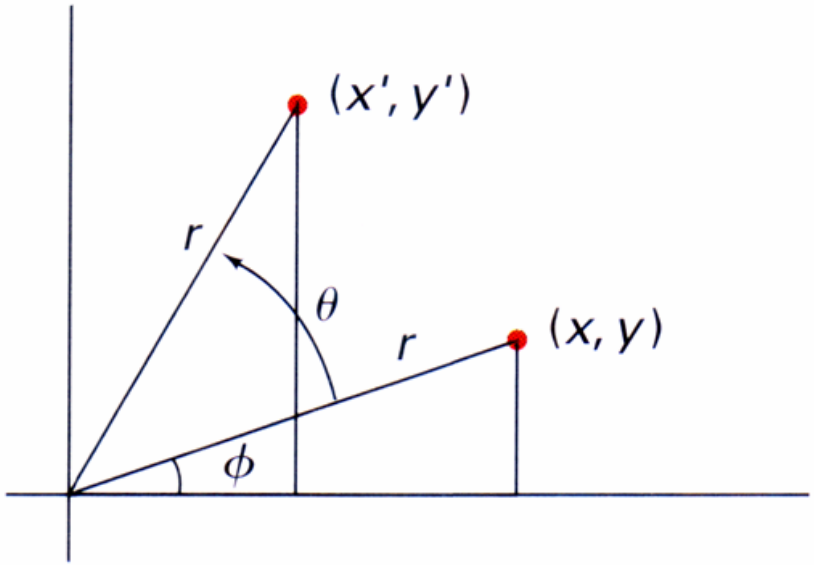
ทั้งนี้แกนการหมุน (z) จะตั้งฉากกับระนาบ (x, y) และมีจุดศูนย์กลางอยู่ที่ pivot point (x_r, y_r)

ดังแสดงในรูป



Rotation Equations

ในที่นี้เราจะพิจารณาการหมุน กรณีที่ pivot point เป็นจุดกำเนิด โดยที่ความสัมพันธ์ระหว่างมุมที่ เวกเตอร์ของจุดเดิม (x, y) และจุดที่ต้องการเปลี่ยนไป (x′, y′) ได้แก่ Φ และ θ ตามลำดับ และ r คือค่ารัศมี (ขนาดของเวกเตอร์ของจุดทั้งสอง) มีค่าคงที่ แสดงดังรูป



ซึ่งเขียนความสัมพันธ์ได้ดังนี้

$$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$$
$$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$$

$\because$ $x = r\cos\phi$ $y = r\sin\phi$ เมื่อแทนค่าจะได้

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

หรือเขียนในรูป Matrix ได้เป็น $\mathbf{P}' = \mathbf{R}\cdot\mathbf{P}$ $\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

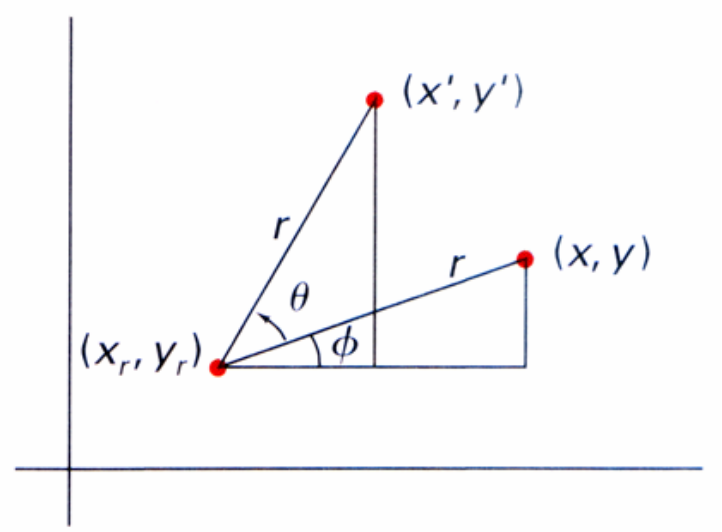
Arbitrary Rotation

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

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

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

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

Transformation Example

การเลื่อน และ การหมุนวัตถุ เรียกรวมกันว่า Rigid Body Transformation ซึ่งหมายถึง การแปลงวัตถุ โดยที่ไม่เกิดการบิดเบี้ยว ($\subset$ conformal mapping)

- จุดทุกจุดบนวัตถุจะ หมุนไปด้วยมุมเท่ากัน (และ/หรือ เลื่อนไปด้วยระยะเท่ากัน)
- การหมุน เส้นตรง (หรือ polygon) ทำได้โดยหมุน จุดปลายเส้นทั้งสอง แล้ววาด เส้นตรง (หรือ polygon) ขึ้นมาจากจุดปลาย (หรือจุดยอดมุม) ใหม่

จงวาดเส้นตรงที่ลากจากจุด (1, 1) เป็นระยะทาง 4 จุดภาพ และมีความชันเท่ากับ 4/3 ซึ่งหมุนรอบจุดกำเนิดไป 45 องศา

หาจุดปลายอีกด้านหนึ่ง $$\begin{bmatrix}1\\1\end{bmatrix}+4\cdot\left\{\frac{1}{\sqrt{3^2+4^2}}\cdot\begin{bmatrix}3\\4\end{bmatrix}\right\}=\begin{bmatrix}1\\1\end{bmatrix}+\frac{4}{5}\cdot\begin{bmatrix}3\\4\end{bmatrix}=\begin{bmatrix}3.4\\4.2\end{bmatrix}$$

จุดปลายใหม่ที่หมุนไป $$\begin{bmatrix}\cos 45^\circ & -\sin 45^\circ\\ \sin 45^\circ & \cos 45^\circ\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} \qquad \begin{bmatrix}\cos 45^\circ & -\sin 45^\circ\\ \sin 45^\circ & \cos 45^\circ\end{bmatrix}\begin{bmatrix}3.4\\4.2\end{bmatrix}$$

หาสมการเส้นตรงจากจุดปลายทั้งสอง เพื่อวาดด้วยวิธี DDA หรือ Bresenham

Scaling

การย่อ/ขยาย (Scaling) สามารถ นิยามได้ว่าคือการ **เปลี่ยนขนาดของวัตถุ**

การ Scaling ของวัตถุทางเรขาคณิตใดๆ ทำได้โดยคูณ พิกัด (x, y) ด้วยสัมประสิทธิ์ซึ่งเป็นจำนวนจริงบวก ในแต่ละแกน (s_x, s_y)

$$x' = s_x x, \quad y' = s_y y$$

หรือเขียนในรูป Matrix

$$\mathbf{P}' = \mathbf{S} \cdot \mathbf{P}$$

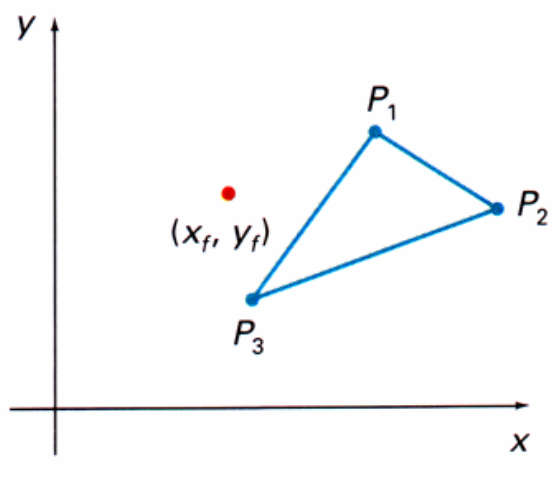
$$\mathbf{P}' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix}, \mathbf{S} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

- ค่าสัมประสิทธิ์มากกว่า 1 จะขยายวัตถุ (น้อยกว่า 1 จะย่อวัตถุ)
- ถ้าค่าสัมประสิทธิ์ที่เท่ากันทั้งสองแกน เรียกว่า uniform scaling มิฉะนั้นเรียกว่า differential scaling

Relative 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

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

Matrix Representations

โปรแกรมกราฟิกทั่วไป มักจะเกี่ยวข้องกับการใช้ Geometric Transformation หลายครั้ง หลายรูปแบบ ตัวอย่างเช่น ในงานสร้างการ์ตูนเคลื่อนไหว จำเป็นต้องหมุน และ เลื่อน วัตถุไปยังตำแหน่งต่างๆ ในฉากทุกๆ เฟรม

ในหัวข้อนี้เราจะพิจารณาการจัด Geometric Transformation ในรูปของ Matrix เพื่อให้การประมวลผล มีประสิทธิภาพมากขึ้น

สังเกตว่า Transformation ที่ผ่านมา จัดได้ในรูปของ $\mathbf{P}' = \mathbf{M}_1 \cdot \mathbf{P} + \mathbf{M}_2$

โดยที่ **P** และ **P′** แทนเวกเตอร์ พิกัดจุดเริ่มต้น และจุดสิ้นสุดตามลำดับ
$\mathbf{M}_1$ เป็น Matrix ขนาด 2x2 ซึ่งเก็บค่าสัมประสิทธิ์การคูณ (Rotation, Scaling)
$\mathbf{M}_2$ เป็น Matrix ขนาด 2x1 ซึ่งเก็บพจน์ที่นำไปบวก (Translation)

ตัวอย่างเช่น
กรณี Translation $\mathbf{M}_1$ จะเป็น Matrix เอกลักษณ์ (Identity Matrix)
กรณี Rotation หรือ Scaling รอบจุดกำเนิด $\mathbf{M}_2$ จะเป็น 0

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 • h, y • h, h)

โดยทั่วไป เราสามารถเลือกค่า h เป็นจำนวนจริงบวกใดๆ แต่เพื่อความสะดวก มักจะเลือกให้ 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

การเลื่อนวัตถุ 2 ครั้ง ด้วย T_1 (t_{x1}, t_{y1}) และ T_2 (t_{x2}, t_{y2}) แสดงได้โดยใช้กฎการจัดหมู่ของ Matrix

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

หรือเขียนในรูป Matrix

$$\begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x1}+t_{x2} \\ 0 & 1 & t_{y1}+t_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

Conclusions

- Color and Grayscale Levels
- Polygon Filling Styles
  - Pattern Tilling
  - Pattern Blending: Multiple Transparent Layers
- Text Attributes *(Self Learning)*
- 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