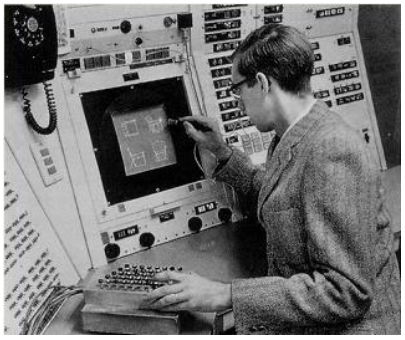
Intro

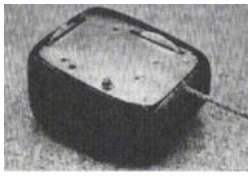
Thursday, July 22, 2010
9:24 AM

The Whirlwind computer

The Whirlwind computer's Memory Units

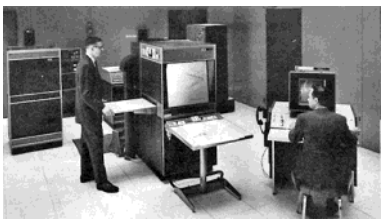


Sketchpad system on the console of the TX-2



mouse

The General Motors DAC system



The DAC-1 System at GM Research Labs, 1965

IBM model number 5150

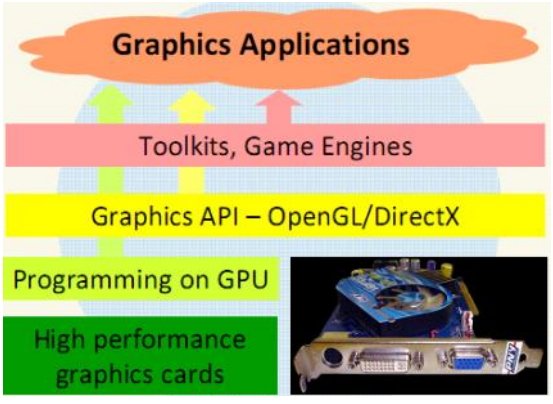
first Apple Macintosh

Hardware

1950	The Whirlwind Computer developed cathode ray tube (CRT)
1963	the Sketchpad drawing system & interaction techniques that used the keyboard and light pen
1963	the mouse
Mid 1960s	The General Motors DAC system (automobile design) and the Itek Digitek system (lens design)
Early 1980s	•Personal computers with built-in raster graphics displays – the Apple Macintosh and IBM PC and its clones •Graphics-based user interfaces became popular
Today	High performance graphic devices: LCD displays, graphic cards, GPU, multi-core technologies, game consoles, advance input devices, etc.

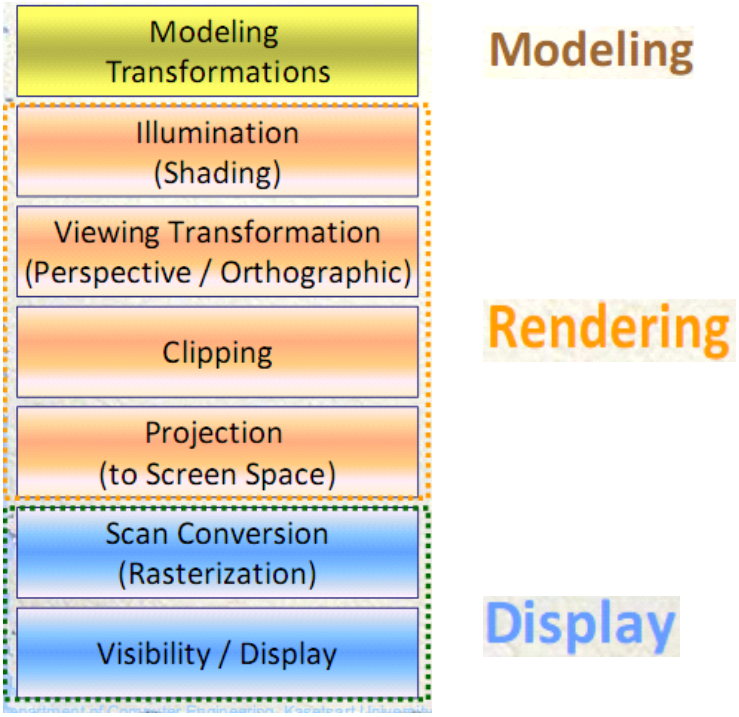
Software

Early	Device-dependent packages supplied by manufacturers
1977	3D Core Graphics System
1985	Graphical Kernel System
Meanwhile and later	Industry graphics standards - Adobe's PostScript - Silicon Graphics' OpenGL - the MIT-led X-Consortium's X Window System and its client-server protocol extensions for 3D graphics
Today	- Graphic APIs (OpenGL, DirectX) - Graphic engines for multi-platforms - Visualization toolkits - X3D for webs, COLLADA for games,



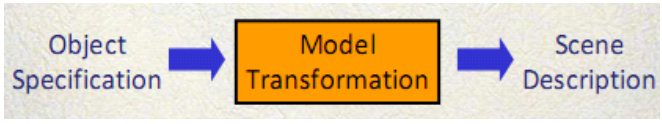
Background

Wednesday, July 14, 2010
5:15 PM



Object Modeling

- Shape
- Materials
- Color
- Shininess
- Transparency
- translucent
- Texture



Rendering

- นำแสงเข้ามาในฉาก
- มุมมองไหนผลเป็นยังไง

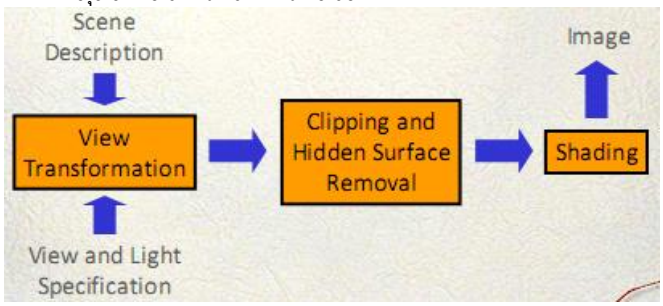


Image Display

- pipeline is handled by firmware and hardware
- past, all of the processing was handled by software
- Pixel เก็บใน mem บางส่วน

Model Transformation

Clipping

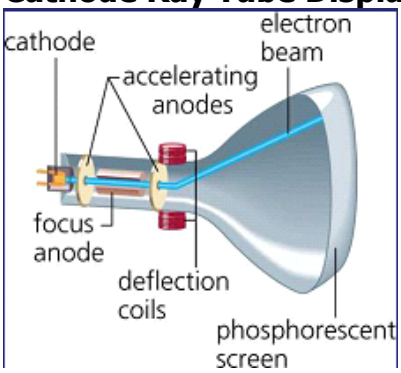
- ตัดบางส่วนทิ้งไป
  - มีบางส่วนมองไม่เห็นจึงตัดทิ้งเพื่อลดขนาด
  - ทำให้คำนวนให้ไวขึ้น

Shading

- กินเวลามากสุด
- นำข้อมูลแสงมาประกอบการคำนวนว่า pixel ควรจะเป็นสีอะไร

Advances in Graphics Technology

Cathode Ray Tube Displays



accelerating anode
- เร่ง e^-

focusing anode
- ทำให้ลำแสงเล็กลง

steering coils
- เบี่ยงเบนอิเล็กตรอน

$E \propto mc^2$

Vector Displays

- ชื่ออื่นๆ
  - Stroke displays
  - Line drawing displays
  - Calligraphic displays
- **"random scan"**
- 1/100 microsec
- Cycle refresh 30 ครั้ง/วิ(30Hz) เพื่อป้องกันภาพ flicker

Raster Displays

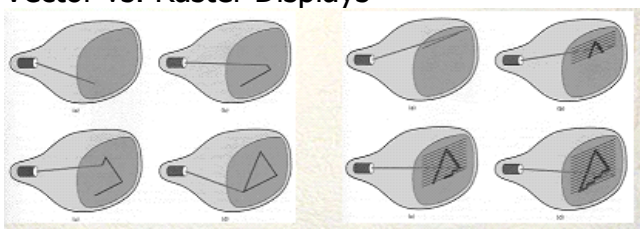
- television technology
- Matrix of pixels
- Bilevel (monochrome)
  - Images display in black/white or black/green
  - 1 bit per pixel – called **"bitmap"**
- Color
  - Multiple-bit-per-pixel = pixmap
- memory space required
  - W x H x N bits

Vector	Raster Displays
☹ ได้ทีละรูป ☹ ถ้าข้อความเยอะๆ แสดงผลไม่ไหว	☺ แสดงผลได้หลายรูปแบบ ☺ ราคาถูก ☹ มีปัญหา aliasing และ pixelization

aliasing = เส้นหยักๆในคอมพิวเตอร์เมื่อส่วนแสดงผลมีความละเอียดน้อยก็จะเห็น
pixelization = รูปเบลอๆ

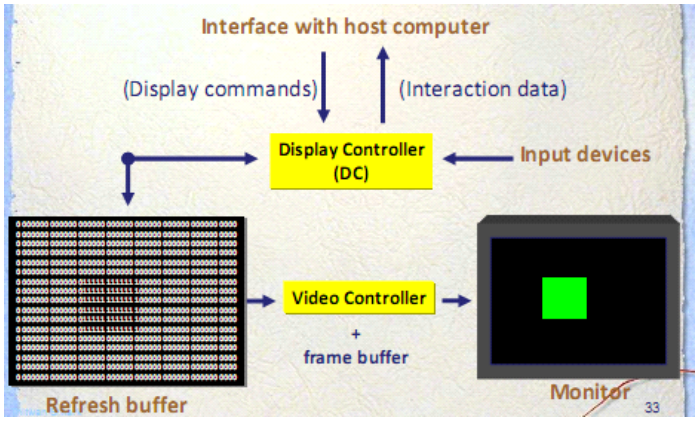
Wednesday, July 14, 2010
6:41 PM

Vector vs. Raster Displays



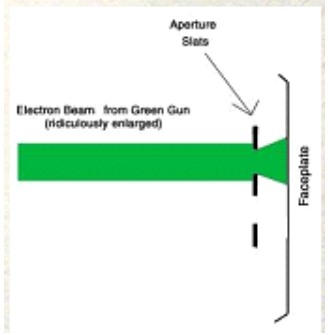
Interlaced Scanning	Progressive Scanning
วาดเส้นตามลำดับ	วาดเส้นคู่ก่อนครบแล้ววาดเส้นคี่

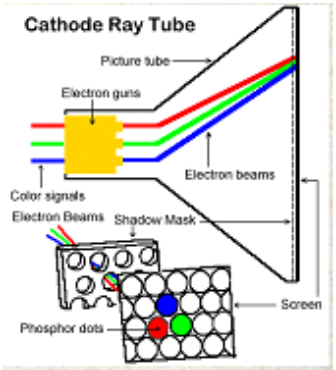
ส่วนมากใช้ progressive เพราะลด flicker ได้ดี
artifacts /comb คือ ภาพไม่ละเอียดบางส่วน



Color CRTs

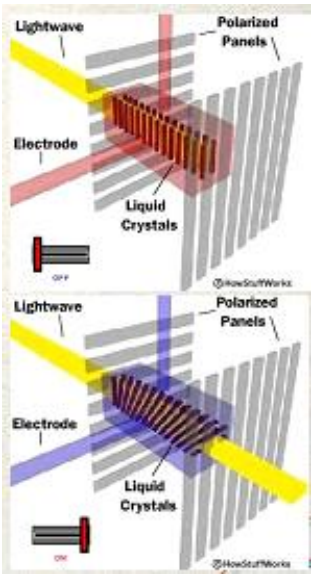
- e^-
- แม่สีคือ RED BLUE GREEN
- ให้จุดอยู่ใกล้กันพอตาคนเราจะแยกไม่ได้
- 1 triad = 1 pixel
- 
- aperture grill
  คือ โลหะแคบ ๆ
  ยาว ๆ เรียงกันตามแนวตั้งโดยเว้นช่องสำหรับให้ลำอีเลคตรอนผ่าน
- **"Trinitron"**





Liquid Crystal Display (LCD)

- Important of liquid crystals is that they are affected by temperature
- untwisted => twisted
- –twisted => untwisted



- **Passive Matrix Displays**
  - rows and columns ต่อกับ integrated circuits
  - voltage ไว้เปลี่ยนสถานะ liquid crystal ที่ Pixel
  - Slow response time (ghost effect)
    - LCD ต้อง refresh image
    - เมาส์ขยับไวๆ
  - Imprecise voltage control (fuzzy and lacking in contrast)
    - limits ความสามารถในการจัดการ 1 pixel ต่อเวลา
- **Active Matrix Displays**
  - thin film transistors (TFT)
    - tiny switching transistors and capacitors which are arranged in a matrix on a glass substrate
    - The amount of voltage supplied to a crystal can be easily controlled.

Dot Pitch	Stripe Pitch
ภาพไม่คม	dot pitch of an aperture-grill monitor

Wednesday, July 14, 2010
7:20 PM

Color LCDs

- Bad pixels (dead pixels)
  - some pixels can stay in a permanently "turned-on" state (this is more common than deal pixels

Reflective

- Malfunctioning pixels can be disturbing

Backlit/Backlight

Light-emitting diode (LED) backlighting

☺ low cost, long life, unaffected by vibration
☹ requires more power especially when the LCD size is large

Electroluminescent panel (ELP) : solid state phenomenon

☺ thin, lightweight, provides even light distribution, various colors
☹ limited life of 3,000 to 5,000 hours to half brightness

Cold Cathode Fluorescent Lamp (CCFL)

☺ longer life time than ELP with minimum of 10,000 to 20,000 hours.
☺ consumes low power, distributes the light evenly across the viewing area
☹ The light output is governed by temperature, in cold conditions the light output by as much as 60%.
☹ The intensity of light cannot be varied.
☹ The reduction of the life expectancy of up to 50% may be expected due to vibration.

Woven Fiber

☹ It provides uniform backlight using woven fiber optic mesh.
☹ The lifetime is dependent on the type of bulb (Halogen/LEDs) used, and the bulb can be easily replaced.
☹ Somewhat expensive thus it is worth for some application

Incandescent

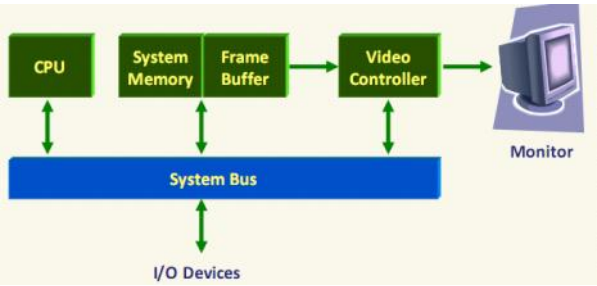
☺ It is only used where cost is a major issue.
☹ Very bright, but not uniform
☹ Generate a large amount of heat
☹ Short life expectance
☹ Consume a lot of power to achieve Brightness
☹ Sensitive to shock and vibration

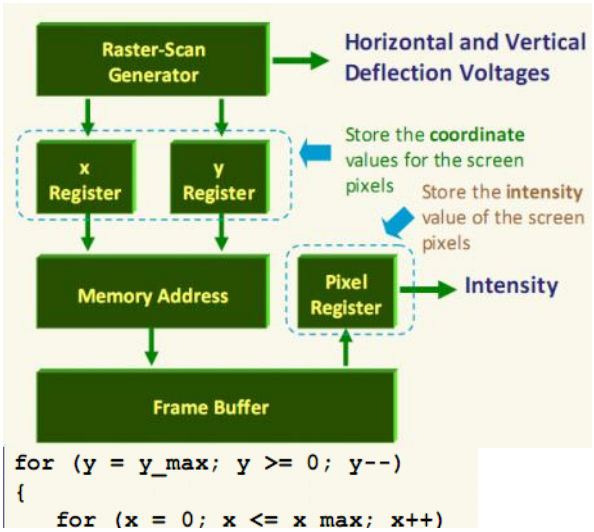
CRTs	LCDs
☺ Inexpensive	☺ Takes up little space on desktop
☺ High refresh rates	☺ Light weight
☺ Color clarity and depth	☺ Energy efficient
☹ Large footprint on desktop	☺ Causing less eye fatigue
☹ Very heavy	☹ Expensive
☹ Use large amounts of energy	☹ Blurry images outside native resolution
☹ Generate excess heat	☹ Motion blur on fast moving images
	☹ Washed out colors

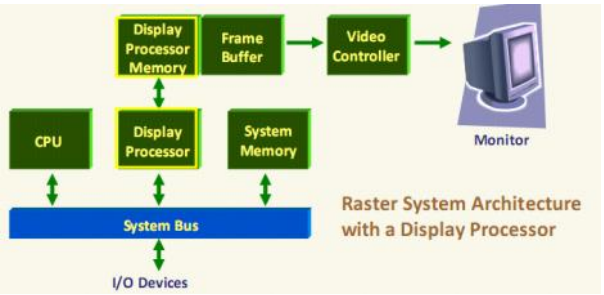
- **CRT** monitors can easily scale to various resolutions. This is referred to as multisync by the industry.
- By adjusting the electron beam in the tube, the screen can easily be adjusted downward to lower resolutions while keeping the picture clarity intact.
- **LCD** screens can actually display only a single given resolution referred to as the native resolution (actual numbers of horizontal and vertical pixels)
- Setting a computer display to a resolution lower than this resolution will have to blend multiple pixels together to produce a similar image but it can result in fuzzy images.

Raster-Scan Systems

- 60 Hz
- retrieve multiple pixel values from the refresh buffer on each pass.
- multiple frame buffers
- display processor = graphics controller =display coprocessor = display-processor memory
- digitizing a picture definition into a set of pixel values for storage in the frame buffer.







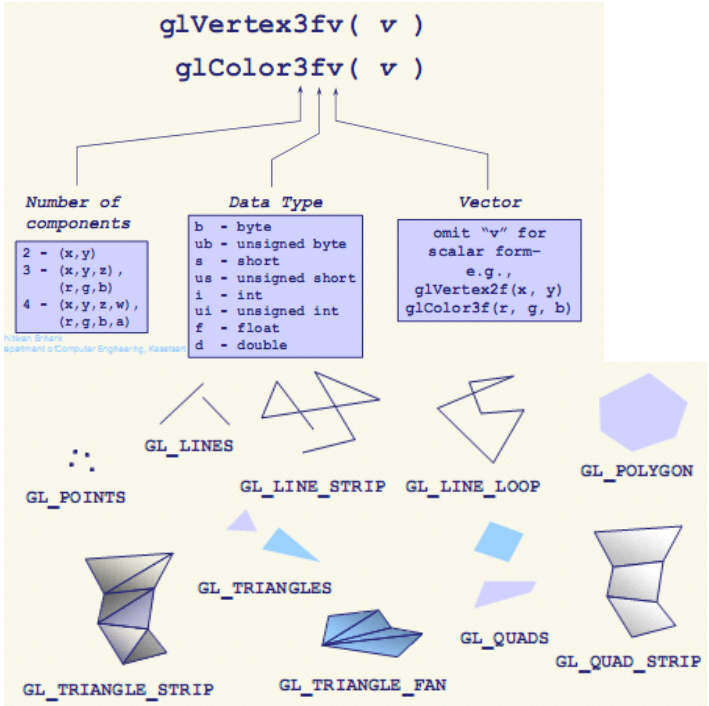
OpenGL + Math

Wednesday, July 14, 2010
8:28 PM

Additional operations of display processors include
1.Generating various line styles (dashed, dotted, or solid)
2.Displaying color areas
3.Applying transformations to the objects in a scene
4.Designed to interface with interactive input devices

Input Devices

- Keyboards
- Button Boxes, and Dials
  - Buttons and switches are used to input predefined functions, and dials are common devices for entering scalar values.
- Mouse Devices
  - Detecting the movement of wheels or rollers
  - Using an optical sensor
- Trackballs and Spaceballs
  - Potentiometers, connected to the ball, measure the amount and direction of rotation.
- Joysticks
- Data Gloves
- Digitizers
- Image Scanners
- Touch Panels



- "gl" = initial capital letters >> glBegin
- "GL" = constants underscore "_" is used as a separator between all component words >> GL_RGB
- pointer to a vector (or an array) of values
  - suffix here is "v"
  - GLfloat color_array[] = {1.0, 0.0, 0.0};
    glColor3fv(color_array);
  - For OpenGL and GLU
    #include <GL/gl.h>
    #include <GL/glu.h>
  - If GLUT is used
    #include <GL/glut.h>

unit vector

$$\mathbf{v}_n = \frac{\mathbf{V}}{\|\mathbf{V}\|}$$

$$\mathbf{P}\cdot\mathbf{Q} = \|\mathbf{P}\|\|\mathbf{Q}\|\cos\alpha$$

$$\text{proj}_{\mathbf{Q}}\,\mathbf{P} = \frac{\mathbf{P}\cdot\mathbf{Q}}{\|\mathbf{Q}\|^2}\mathbf{Q}$$

Dot Product

$$\mathbf{P}\cdot\mathbf{Q} = P_xQ_x + P_yQ_y + P_zQ_z$$

$$\text{perp}_{\mathbf{Q}}\,\mathbf{P} = \mathbf{P} - \text{proj}_{\mathbf{Q}}\,\mathbf{P} = \mathbf{P} - \frac{\mathbf{P}\cdot\mathbf{Q}}{\|\mathbf{Q}\|^2}\mathbf{Q}$$

$$\mathbf{P}^{\mathrm{T}}\cdot\mathbf{Q} = \begin{bmatrix} P_1 & P_2 & \cdots & P_n \end{bmatrix}\begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_n \end{bmatrix}$$

Cross Product

System of Linear Equations
(a) Exchange two rows.
(b) Multiply a row by a nonzero scalar.
(c) Add a multiple of one row to another row.

$$\mathbf{Q}\times\mathbf{P} = -(\mathbf{P}\times\mathbf{Q})$$

$$\|\mathbf{P}\times\mathbf{Q}\| = \|\mathbf{P}\|\|\mathbf{Q}\|\sin\alpha$$

$$\mathbf{P}\times\mathbf{Q} = \begin{vmatrix} i & j & k \\ P_x & P_y & P_z \\ Q_x & Q_y & Q_z \end{vmatrix}$$

$$MM^{-1} = M^{-1}M = I.$$

Raster Graphics Algorithms

Wednesday, July 14, 2010
9:07 PM

Rasterization

Geometric primitives

- Points – round vertex location in coordinates
- Lines – endpoints and in between points
- Polygons – filling area bounded by edges

Line Rasterization Requirements

•Transform continuous primitive into discrete samples
•Uniform thickness & brightness
•Continuous appearance
•No gaps
•Accuracy
•Speed

$$y = y1 + \frac{x - x1}{x2 - x1}(y2 - y1) = y1 + m(x - x1)$$

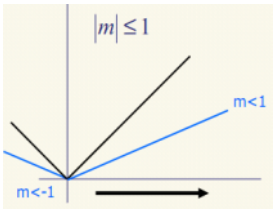
$$y = m \cdot x + b$$

DDA Algorithm

However, it could have some inaccuracy

- round-off error can cause the calculated pixel positions to drift away from true line path for long line segments.

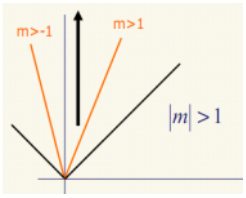
$$\delta y = m \cdot \delta x$$



$$\delta x = 1$$

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

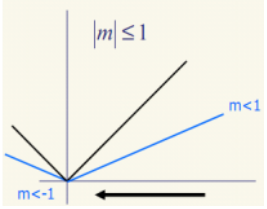
where $x_k < x_{k+1}$
running from left to right



$$\delta y = 1$$

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

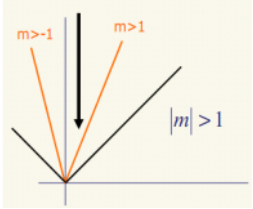
where $y_k < y_{k+1}$
running from bottom to top



$$\delta x = 1$$

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

where $x_k > x_{k+1}$
running from right to left



$$\delta y = 1$$

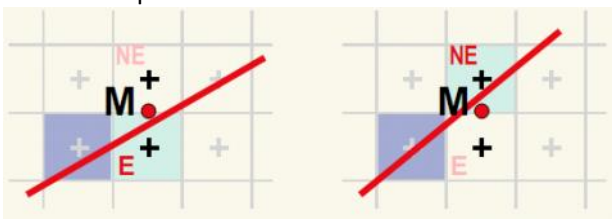
$$x_{k+1} = x_k - \frac{1}{m}$$

where $y_k < y_{k+1}$
running from top to bottom

Midpoint Line Algorithm

- Observation
  - If we're at pixel P(xk, yk), the next pixel must be either E (xk+1, yk) or NE (xk+1, yk+1)

➢ Choose E if segment passes below or through middle point M
➢ Choose NE if segment passes above M Which pixel to choose: E or NE?
➢ The error -- that is, the vertical distance between E and NE, or both pixels and the actual line -- is always less than ½ pixel.



$$y = \frac{dy}{dx}x + B$$

$$d = F\left(x_p + 1, y_p + \frac{1}{2}\right)$$

$$F(x, y) = dy \cdot x - dx \cdot y + B \cdot dx = 0$$

$$F(x, y) = \begin{cases} +, (x, y) \text{ is below the line} \\ 0, (x, y) \text{ is on the line} \\ -, (x, y) \text{ is above the line} \end{cases}$$

เลือก E

$$d_{new} = d_{old} + a$$

เลือก NE

$$d_{new} = d_{old} + a + b$$

$$d_{start} = a + b/2 = dy - dx/2$$

$$d_{start} = 2a + b = 2dy - dx$$

Wednesday, July 14, 2010
9:42 PM

Cycle

$$F(x,y) = x^2 + y^2 - R^2$$

• SE = outside cycle

$d_{old} >= 0$

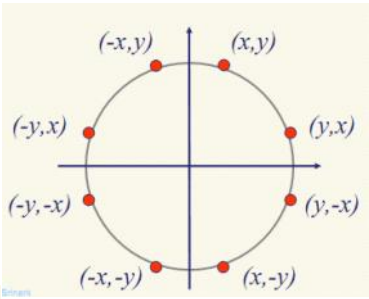
$$d_{new} = d_{old} + (2x_p + 3)$$

•E = inside

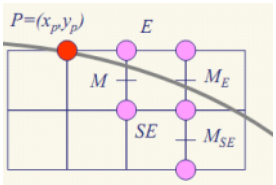
$d_{old} < 0$

$$d_{new} = d_{old} + (2x_p - 2y_p + 5)$$

Eight-Way Symmetry



$$F(x,y) = \begin{cases} +, (x,y) \text{ is outside the circle} \\ 0, (x,y) \text{ is on the circle} \\ -, (x,y) \text{ is inside the circle} \end{cases}$$



$$d_{old} = F(x_p + 1, y_p - \frac{1}{2}) = (x_p + 1)^2 + (y_p - \frac{1}{2})^2 - R^2$$

Midpoint Circle Algorithm

(1) Choose the pixel based on the sign of the variable d computed during the previous iteration.
(2) Update the decision variable d with the ? that corresponds to the choice of pixel.

```
d = 5.0/4 - radius;  ➔  h = 1 - radius;
```

Filling Rectangles

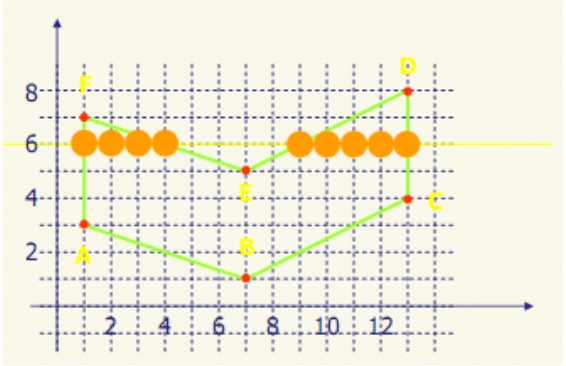
Spans exploit a primitive's spatial coherence:

- the fact that primitives often do not change from pixel to pixel within a span or from scan line to scan line.
- For a solidly shaded primitive, all pixels on a span are set to the same value, which provides span coherence.

Rules:

1.Color only interior pixels
2.Color left and bottom edges

Filling Polygons

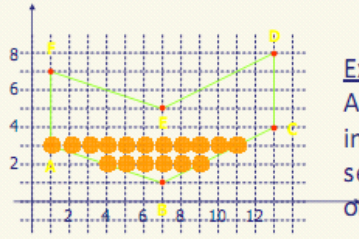


Polygon Filling Algorithm

1. Find the intersections of the scan line with all edges of the polygon.
2. Sort the intersections by increasing x coordinate.
3. Fill in all pixels between pairs of intersections that lie interior to the polygon:
   - Using the odd-parity rule to determine that a point is inside a region.
   - Parity is initially even, and each intersection encountered thus inverts the parity bit-draw when parity is odd, do not draw when it is even.

> Special cases: shared vertices

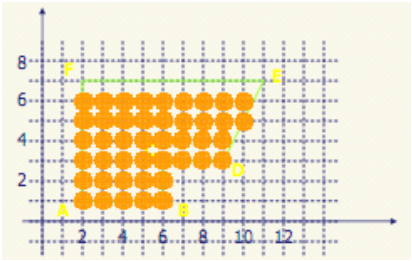
Count the y_{min} vertex of an edge in the parity calculation but not the y_{max} vertex.



Example
At A, we will get two intersections (FA, AB) to set the parity, use only A of FA.

> Special cases: horizontal edges

- They can be handled by not counting their vertices.



> Special cases: slivers

- polygon เล็กเกิดไป
- แก้โดย **"anti-aliasing"**

Aliasing

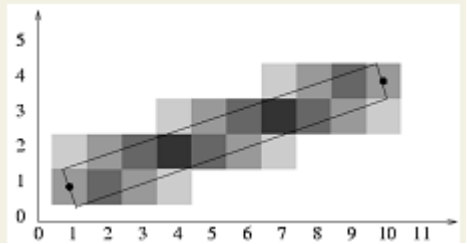
> Undersampling: Low-frequency sampling

$$f_s = 2 f_{max} \qquad \Delta x_s = \frac{\Delta x_{cycle}}{2}$$

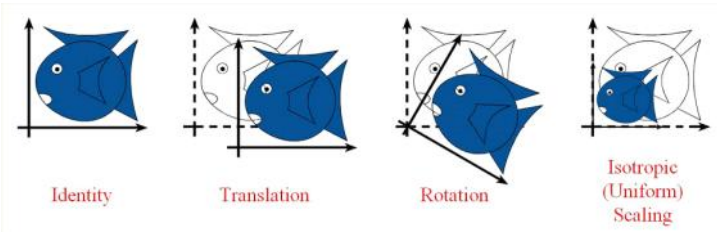
2D Transformations

Wednesday, July 14, 2010
10:16 PM

Antialiasing

1. **Increasing resolution**
   - It is to use a display device with higher resolution!!
2. **Supersampling**
   - Construct a virtual higher resolution display by expanding each pixel into k × k subpixels.
   - Scan convert the desired graphic into the above.
   - Set the intensity of each actual display pixel to a value that depends upon the number active subpixels (n)
   - $\frac{n}{N}F + \left(1 - \frac{n}{N}\right)B$ (n: the maximum number of the k × k subpixels that can be activated; F: foreground color; B: background color)
   - The fraction n/N is called the blend ratio.
   - n = the sum of the weights of the active subpixels
   - $N = \sum_{i,j} w_{ij}$ = the sum of the total weight of each pixel
1	2	1
2	4	2
1	2	1
3. **Unweighted area sampling**
   - A line should not set the intensity of only a single pixel in a column to black,but rather should contribute some amount of intensity to each pixel in the columns whose area it intersects.
   - centered on grid points
     - proportional to the percentage of the pixel's tile it covers.
     - Pixel intensity →Percentage of covering area
   - 

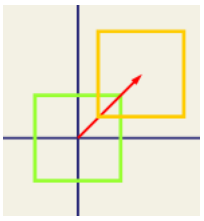
The approximation to the area overlap:
1. Divide the pixel into a finer grid of rectangular subpixels
2. Count the number of subpixels inside the line



2D Translation

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

$$\mathbf{P'} = \begin{bmatrix} x' \\ y' \end{bmatrix}, \mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix}, \mathbf{T} = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

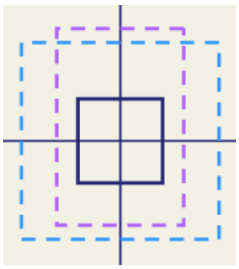


2D Scaling

$$x' = s_x x$$
$$y' = s_y y$$

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

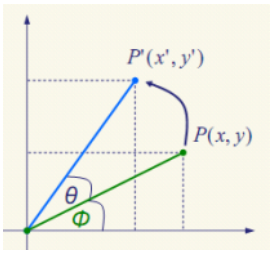
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



2D Rotation

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

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



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

$$\mathbf{P'} = \mathbf{P} + \mathbf{T} \qquad \mathbf{P'} = \mathbf{S} \cdot \mathbf{P} \qquad \mathbf{P'} = \mathbf{R} \cdot \mathbf{P}$$

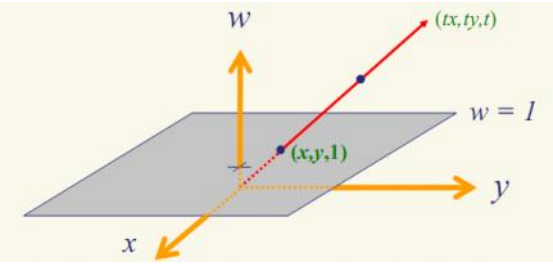
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$$

- If points are expressed in homogeneous coordinates, all three transformations can be treated as multiplications.

Wednesday, July 14, 2010
11:00 PM

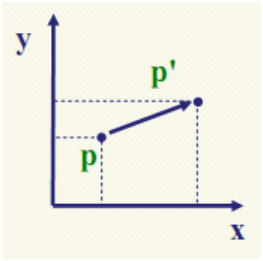
Homogeneous Coordinates

- Two sets of homogeneous coordinates (x,y,W) and(x',y',W') represent the same point if and only if one is a multiple of the other.
- (2,3,6) and (4,6,12)
- If the W coordinate is nonzero, (x, y, W) represent the same point as (x/W, y/W, 1). The number x/Wand y/Ware called the Cartesian coordinates of the homogeneous point.



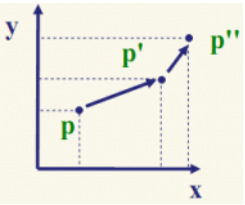
2D Translation

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



$$\mathbf{P'} = \mathbf{T}(d_x, d_y) \cdot \mathbf{P}$$

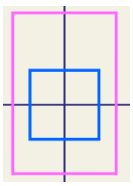
$$\mathbf{P''} = \mathbf{T}(d_{x2}, d_{y2}) \cdot \mathbf{P'} = \mathbf{T}(d_{x2}, d_{y2}) \cdot \mathbf{T}(d_{x1}, d_{y1}) \cdot \mathbf{P}$$



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

$$\mathbf{T}(d_{x2}, d_{y2}) \cdot \mathbf{T}(d_{x1}, d_{y1}) = \mathbf{T}(d_{x1} + d_{x2}, d_{y1} + d_{y2})$$

2D Scaling



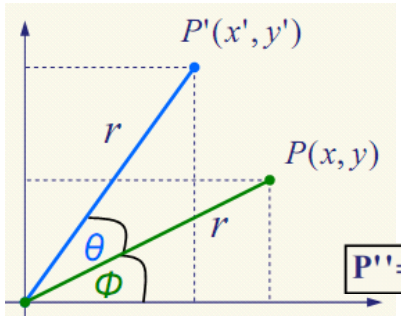
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{P'} = \mathbf{S}(s_x, s_y) \cdot \mathbf{P}$$

$$\mathbf{P''} = \mathbf{S}(s_{x2}, s_{y2}) \cdot \mathbf{P'} = \mathbf{S}(s_{x2}, s_{y2}) \cdot \mathbf{S}(s_{x1}, s_{y1}) \cdot \mathbf{P}$$

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

2D Rotation

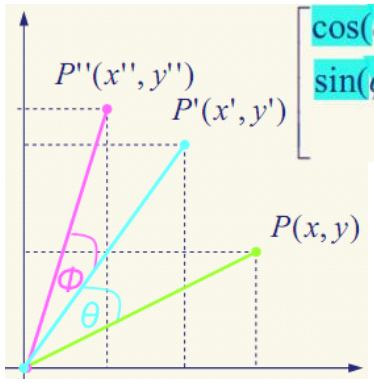


$$\mathbf{P'} = \mathbf{R}(\theta) \cdot \mathbf{P}$$

$$\mathbf{P'} = \mathbf{R}(\theta) \cdot \mathbf{P}$$
$$\mathbf{P''} = \mathbf{R}(\phi) \cdot \mathbf{P'}$$

$$\mathbf{P''} = \mathbf{R}(\phi) \cdot \mathbf{P'} = \mathbf{R}(\phi) \cdot \mathbf{R}(\theta) \cdot \mathbf{P}$$

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

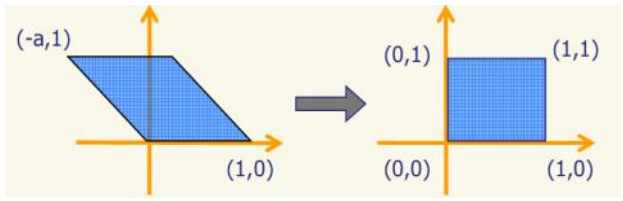


$$\begin{bmatrix} \cos(\phi+\theta) & -\sin(\phi+\theta) & 0 \\ \sin(\phi+\theta) & \cos(\phi+\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shear Transformations

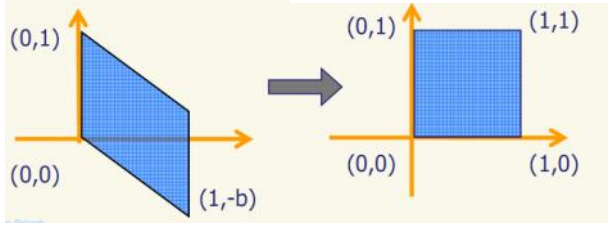
Shear in x

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



Shear in y

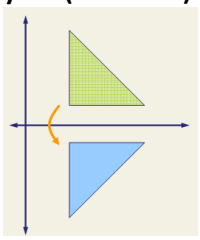
$$\mathbf{SH}_y = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Thursday, July 15, 2010
9:57 AM

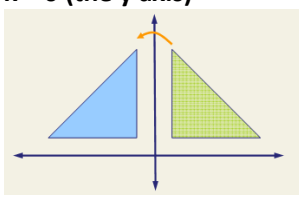
Reflection Transformations

y = 0 (the x axis)



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

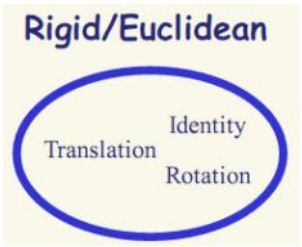
x = 0 (the y axis)

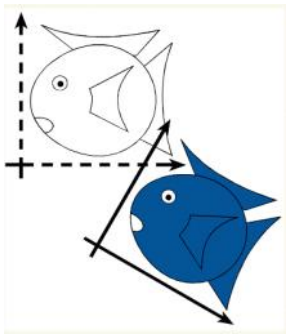


$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Classes of Transformations

Rigid-Body/Euclidean Transforms



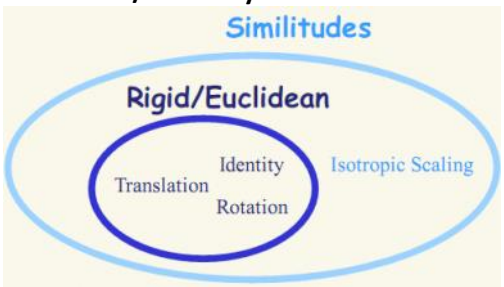


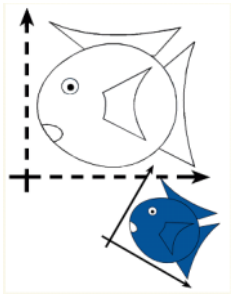
$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

rigid-body transformations

- preserves angles & lengths.

Similitudes/Similarity Transforms

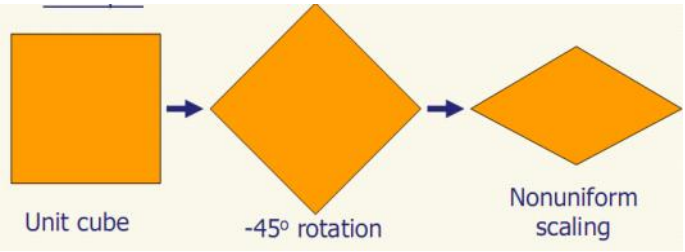




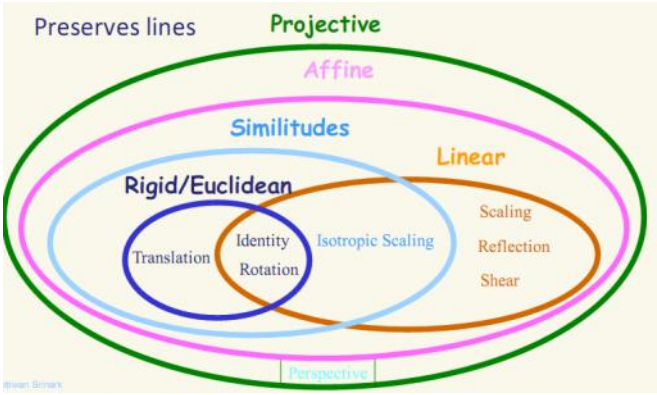
Linear Transformations

between two vector spaces V and W is a map T:V ? ?? ?W such that the
following hold:
1. T(v1+v2) = T(v1)+T(v2) for any vectors v1 and v2 in V, and
2. T(av) = aT(v) for any scalar a.

Affine Transformations



Projective Transformations

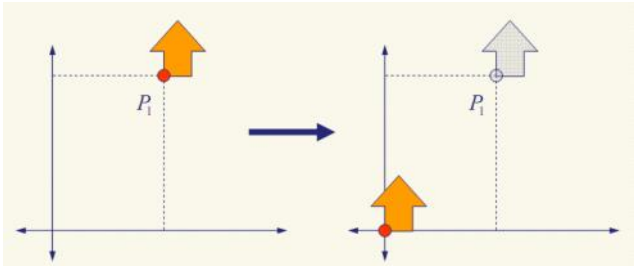


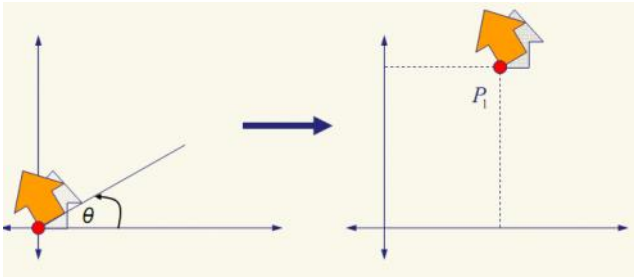
Thursday, July 15, 2010
10:14 AM

Composition of 2D Transformations

Rotation

1. Translate such that P1 is at the origin.
2. Rotate.
3. Translate such that the point at the origin returns to P1.





$$\mathbf{T}(x_1, y_1) \cdot \mathbf{R}(\theta) \cdot \mathbf{T}(-x_1, -y_1)$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & x_1(1-\cos\theta)+y_1\sin\theta \\ \sin\theta & \cos\theta & y_1(1-\cos\theta)-x_1\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

Sale

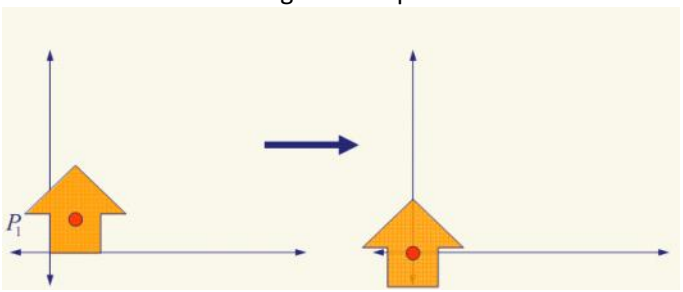
1. Translate such that P1 is at the origin.
2. Scale.
3. Translate back to P1.

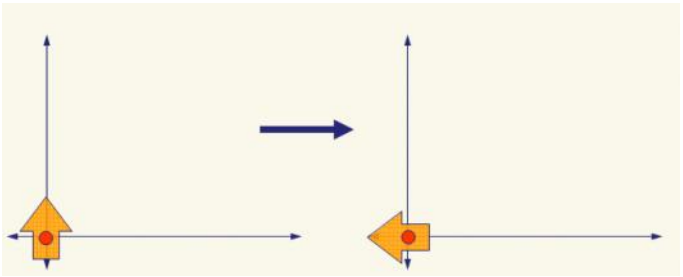
$$\mathbf{T}(x_1, y_1) \cdot \mathbf{S}(s_x, s_y) \cdot \mathbf{T}(-x_1, -y_1)$$

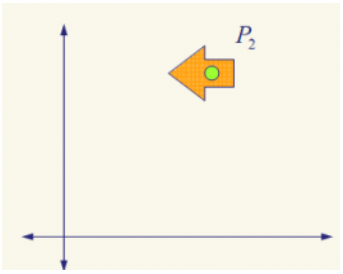
$$\begin{bmatrix} s_x & 0 & x_1(1-s_x) \\ 0 & s_y & y_1(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

Scale+ Rotation

1. Translate such that P1 is at the origin.
2. Scale and rotate.
3. Translate from the origin to the position P2.







$$\mathbf{T}(x_2, y_2) \cdot \mathbf{R}(\theta) \cdot \mathbf{S}(s_x, s_y) \cdot \mathbf{T}(-x_1, -y_1)$$
$$= \begin{bmatrix} 1 & 0 & x_2 \\ 0 & 1 & y_2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformations Between Coordinate Systems

Viewing

Transform object descriptions

World coordinates → Device coordinates

Modeling and Design

Each object has its own local Cartesian reference, it is transformed into position and orientation within the scene.

Local coordinates → Scene coordinates

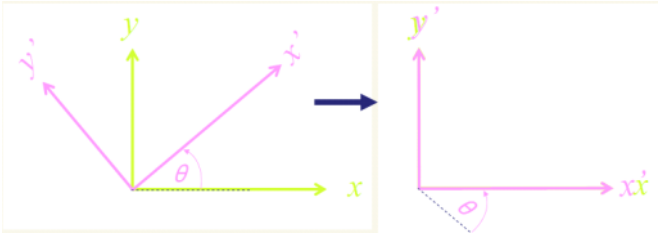
3D Rotations

Thursday, July 15, 2010
10:24 AM

1. Translate so that the origin (x0,y0) of the x'y' system is moved to the origin (0,0) of the xy system.

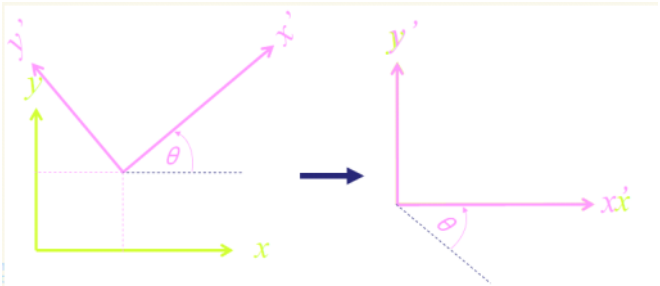
$$\mathbf{T}(-x_0,-y_0) = \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

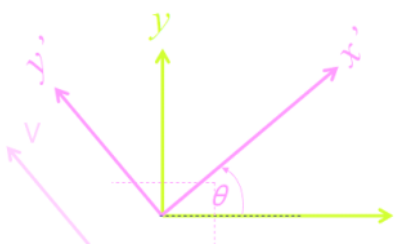
2. Rotate the x' axis onto the x axis.



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

$$\mathbf{M}_{xy,x'y'} = \mathbf{R}(-\theta)\cdot\mathbf{T}(-x_0,-y_0)$$





$$\mathbf{u} = (v_y,-v_x) = (u_x,u_y)$$

$$\mathbf{v} = \frac{\mathbf{V}}{|\mathbf{V}|} = (v_x,v_y)$$

$$\mathbf{R} = \begin{bmatrix} u_x & u_y & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3-D Rotation Matrices

z-axis

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

x-axis

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

y-axis

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

rotated about an axis that is parallel to one of the coordinate axes.

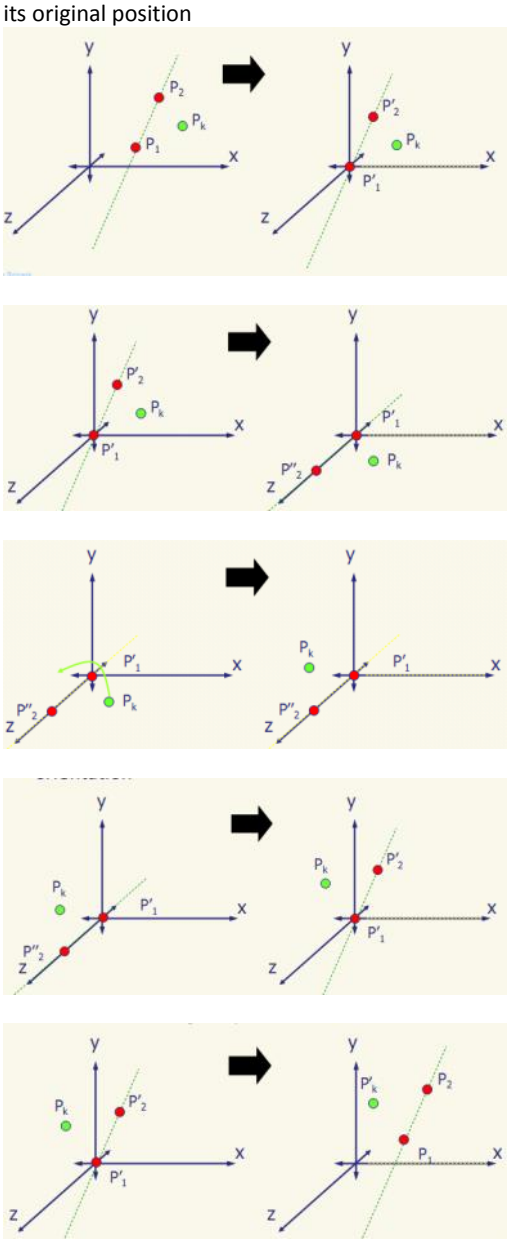
1. Translate the object so that the rotation axis coincides with the parallel coordinate axis.
2. Perform the specified rotation about that axis.
3. Translate the object so that the rotation axis is moved back to its original position.

$$\mathbf{P}' = \mathbf{T}^{-1}\cdot\mathbf{R}_x(\theta)\cdot\mathbf{T}\cdot\mathbf{P}$$

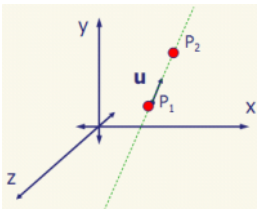
Monday, July 19, 2010
6:30 PM

rotated about an axis that is <u>not</u> parallel to one of the coordinate axes.

1. Translate the object so that the rotation axis passes through the coordinate origin
2. Rotate the object so that the axis of rotation coincides with one of the coordinate axes
3. Perform the specified rotation about that coordinate axis
4. Apply inverse rotation axis back to its original orientation
5. Apply inverse translation to bring the rotation axis back to its original position



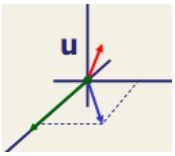
rotation axis



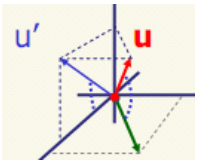
$$\mathbf{V} = \mathbf{P}_2 - \mathbf{P}_1 = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$$

$$\mathbf{u} = \frac{\mathbf{V}}{|\mathbf{V}|} = (a,b,c)$$

1. Set up the translation matrix that repositions the rotation axis so that it passes through the origin.
2. Put the rotation axis onto the z axis by using the coordinate-axis rotations:



2.1 Rotate about the x axis to get u into the xz plane.



$$d = \sqrt{b^2 + c^2}$$

u′ = (0,b,c)
u_z = (0,0,1)

$$\cos\alpha = \frac{\mathbf{u}' \cdot \mathbf{u}_z}{|\mathbf{u}'||\mathbf{u}_z|} = \frac{c}{d}$$

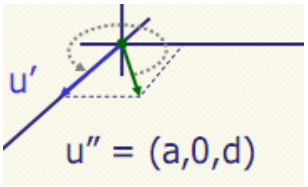
$$\mathbf{u}' \times \mathbf{u}_z = \mathbf{u}_x \cdot b$$

$$d \sin\alpha = b$$
$$\sin\alpha = \frac{b}{d}$$
$$|\mathbf{u}'| = d, |\mathbf{u}_z| = 1$$

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c}{d} & -\frac{b}{d} & 0 \\ 0 & \frac{b}{d} & \frac{c}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.2 Rotate about the y axis to swing u around to the z axis.

$$\mathbf{u}' \times \mathbf{u}_z = \mathbf{u}_x |\mathbf{u}'||\mathbf{u}_z| \sin\alpha = \mathbf{u}_x d \sin\alpha$$



$$\cos\beta = \frac{\mathbf{u}'' \cdot \mathbf{u}_z}{|\mathbf{u}''||\mathbf{u}_z|} = d$$

$$|\mathbf{u}''| = 1, |\mathbf{u}_z| = 1$$

$$\sin\beta = -a$$

$$\mathbf{u}'' \times \mathbf{u}_z = -\mathbf{u}_y \cdot a$$

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

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

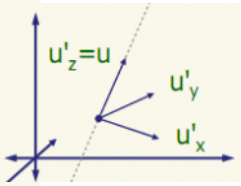
$$\mathbf{R}(\theta) = \mathbf{T}^{-1} \cdot \mathbf{R}_x^{-1}(\alpha) \cdot \mathbf{R}_y^{-1}(\beta) \cdot \mathbf{R}_z(\theta) \cdot \mathbf{R}_y(\beta) \cdot \mathbf{R}_x(\alpha) \cdot \mathbf{T}$$

Monday, July 19, 2010
6:49 PM

$$\mathbf{R}_{xy} = \mathbf{R}_y(\beta) \cdot \mathbf{R}_x(\alpha)$$

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

is orthogonal



$$\mathbf{u}'_z = \mathbf{u}$$

$$\mathbf{u}'_y = \frac{\mathbf{u} \times \mathbf{u}_x}{|\mathbf{u} \times \mathbf{u}_x|}$$

$$\mathbf{u}'_x = \mathbf{u}'_y \times \mathbf{u}'_z$$

$$\mathbf{u}'_x = (u'_{x1}, u'_{x2}, u'_{x3})$$

$$\mathbf{u}'_y = (u'_{y1}, u'_{y2}, u'_{y3})$$

$$\mathbf{u}'_z = (u'_{z1}, u'_{z2}, u'_{z3})$$

$$\mathbf{R} = \mathbf{R}_y(\beta) \cdot \mathbf{R}_x(\alpha) = \begin{bmatrix} u'_{x1} & u'_{x2} & u'_{x3} & 0 \\ u'_{y1} & u'_{y2} & u'_{y3} & 0 \\ u'_{z1} & u'_{z2} & u'_{z3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Reflections

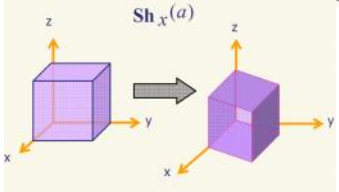
$$\mathbf{M}_{z\_reflect} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Shears in x-Direction

$$\mathbf{Sh}_x(a) = \begin{bmatrix} 1 & a & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x+ay \\ y \\ z \\ 1 \end{bmatrix}$$

$$\mathbf{Sh}_x(b) = \begin{bmatrix} 1 & 0 & b & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x+bz \\ y \\ z \\ 1 \end{bmatrix}$$

$\mathbf{Sh}_x(a)$

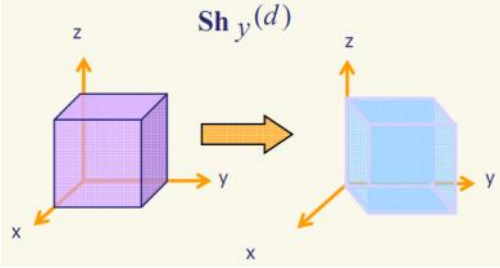


3D Shears in Y-Direction

$$\mathbf{Sh}_y(c) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & c & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y+cz \\ z \\ 1 \end{bmatrix}$$

$$\mathbf{Sh}_y(d) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ d & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ dx+y \\ z \\ 1 \end{bmatrix}$$

$\mathbf{Sh}_y(d)$



3D Shears in z-Direction

$$\mathbf{Sh}_z(e) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ e & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ ex+z \\ 1 \end{bmatrix}$$

$$\mathbf{Sh}_z(f) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & f & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ fy+z \\ 1 \end{bmatrix}$$

Between Coordinate Systems

1. Translate such that the x'y'z' origin is positioned at the xyz origin)

$$T(-x_0, -y_0, -z_0)$$

2. Use the unit axis vectors to form the coordinate axis rotation matrix

$$\mathbf{R} = \begin{bmatrix} u'_{x1} & u'_{x2} & u'_{x3} & 0 \\ u'_{y1} & u'_{y2} & u'_{y3} & 0 \\ u'_{z1} & u'_{z2} & u'_{z3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where u'_x, u'_y, u'_z are unit vectors along x', y', z', resp.