



COMPUTER GRAPHICS
School of Computer Engineering

Suranaree University
of Technology

Lecture 0 Introduction

Paramate Horkaew

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



Course Outline

- Graphics Systems
- Raster Algorithms
- Projection, Transformation and Animation
- 2-Dimensional Viewing Algorithms
- Texture and Anti-aliasing

- 3-Dimensional Representation and Viewing
- Illumination Models, Surface Rendering and Colors
- Ray Tracing, Radiosity and CSG
- Geometric Warping and Morphing
- Non-photorealistic Rendering Algorithms



Assessments

- Assignments/Homework 15%
- Random QUIZ 15%
- Midterm Exam 30%
- Final Exam 40%

Official Languages: Pseudo Code or C/C++

Optional Tutorial by Appointment: *Graphics Programming*

Operating System : Microsoft Windows

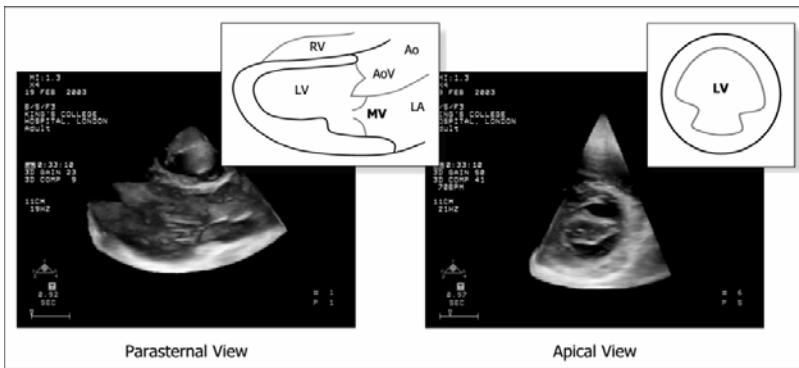
C/C++ framework : [Watcom C/C++](#), Borland C/C++, Microsoft C/C++

OpenGL (on Java or C/C++)



Graphics Applications

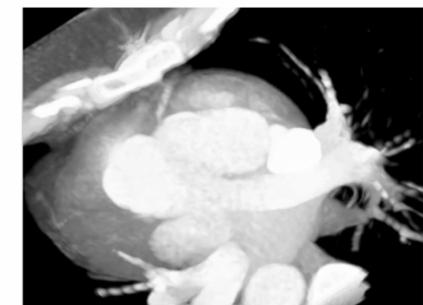
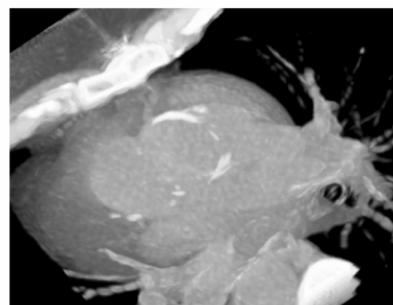
Conventional uses of Medical Imaging



a cross-section slice

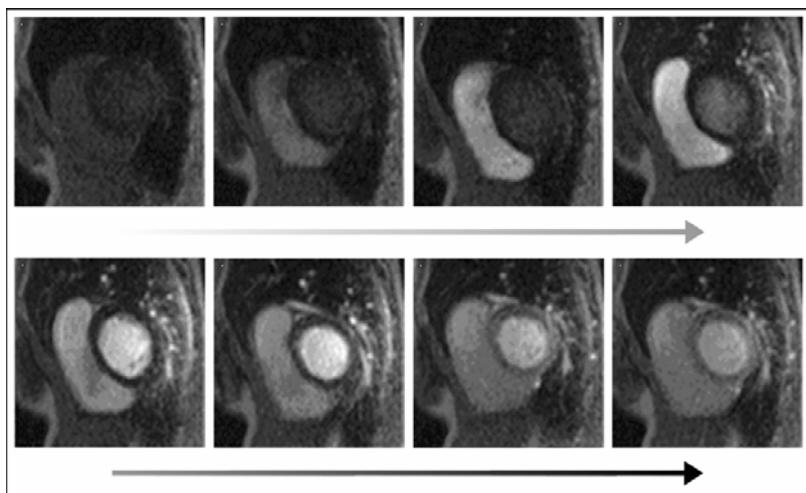


MIP reconstruction



normal EBCT scan

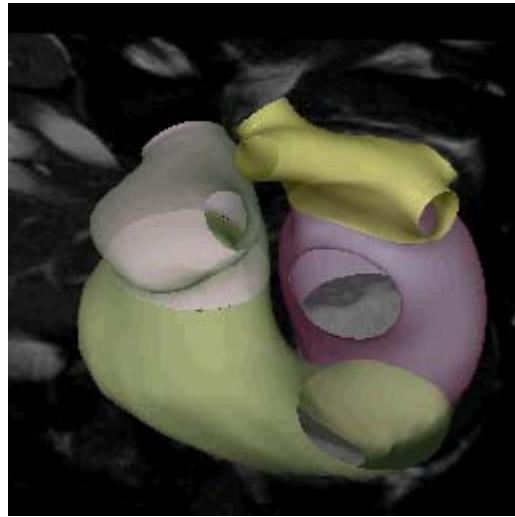
contrast enhanced scan



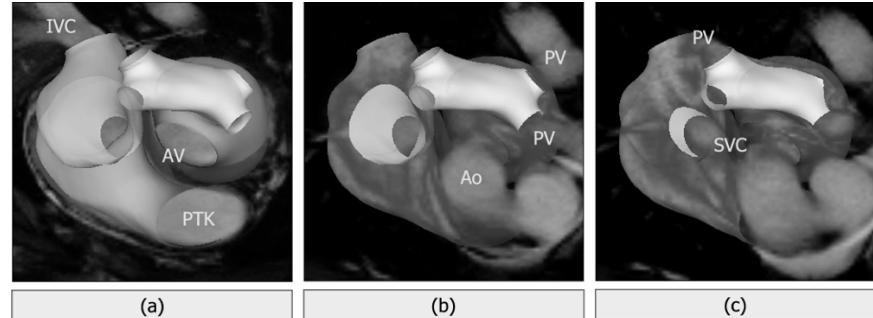
Diagnosis was made primarily on imaging information



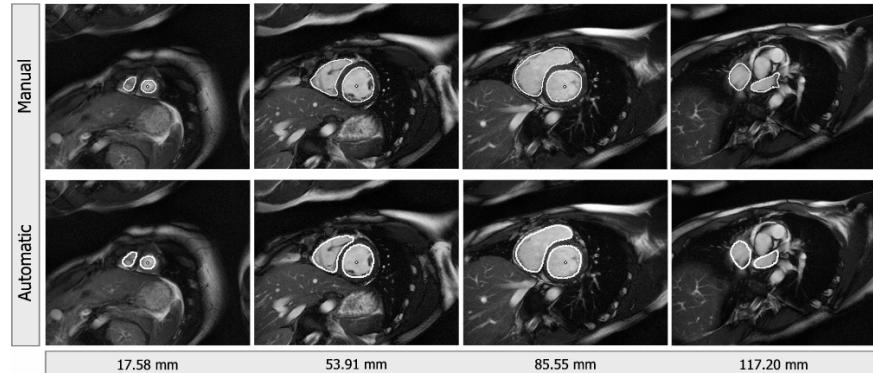
3D Cardiac Tracking



The segmented surface from a selected subject



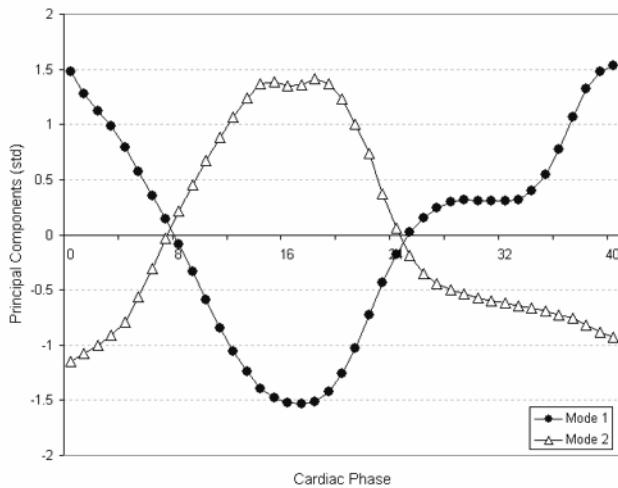
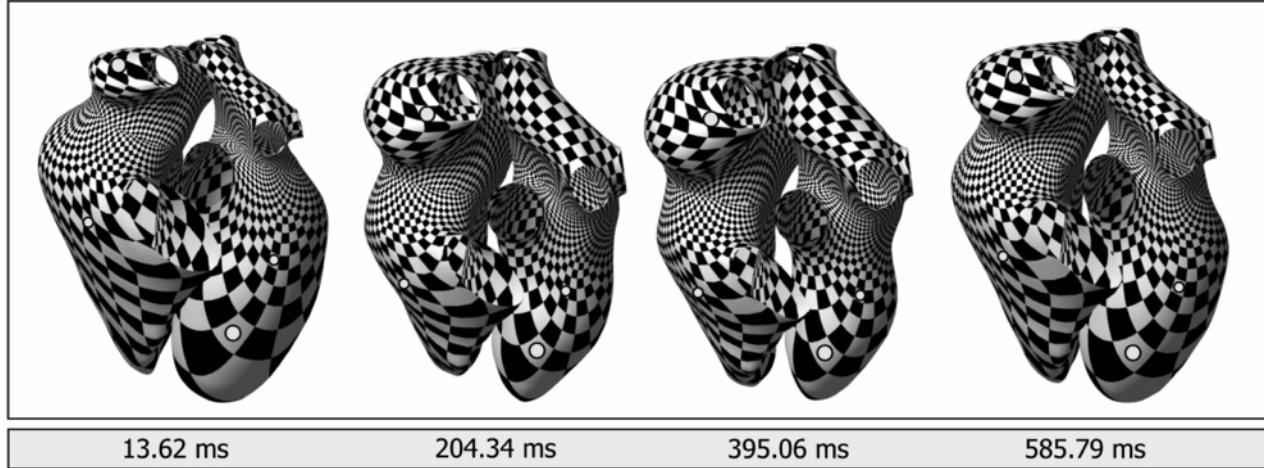
The segmented four-chamber heart superimposed with anatomical landmarks on three short axis images



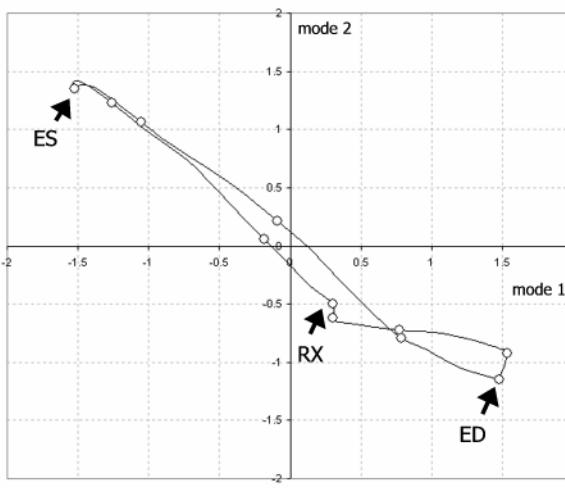
An example of the automatic segmentation result at end diastole



Application to Morphological Analysis



(a) principal components



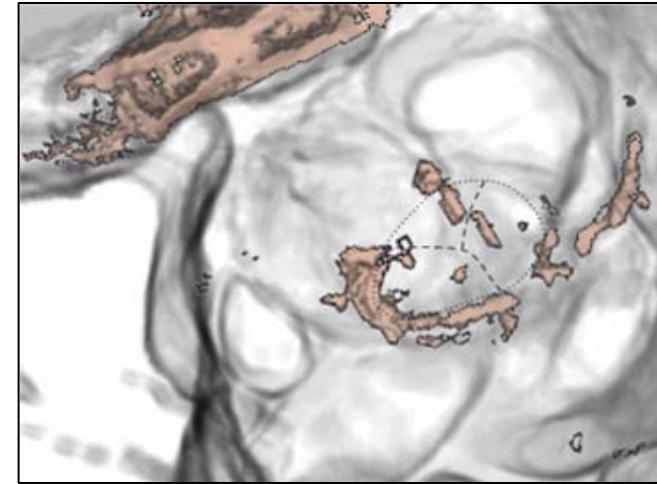
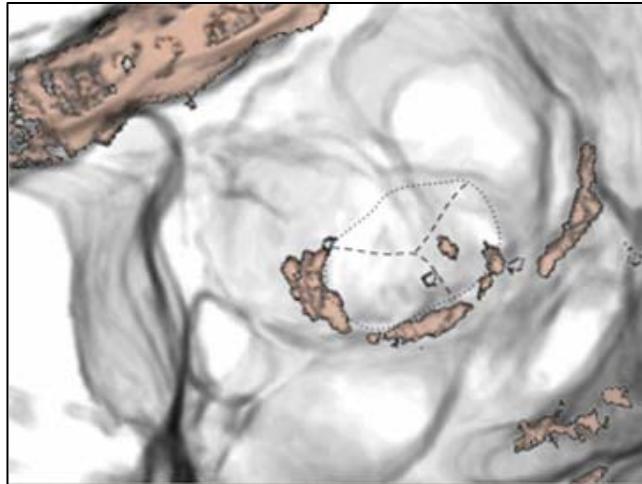
(b) components trajectory

Characterisation of cardiac motion showing the standardised first two principal components (a) and the corresponding trajectory of the cardiac motion, during the cardiac cycle (b)

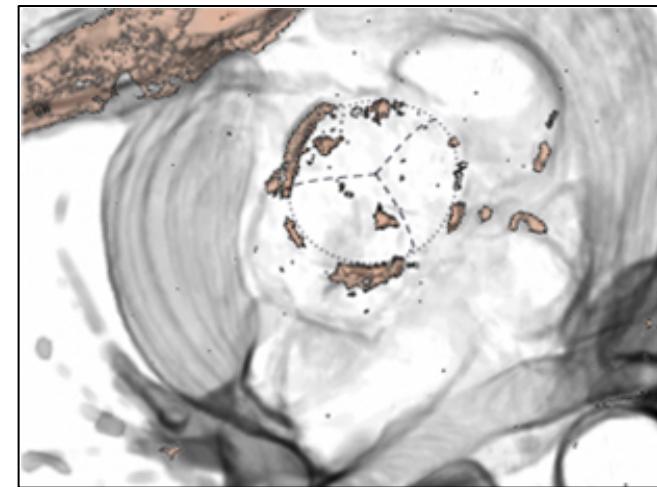
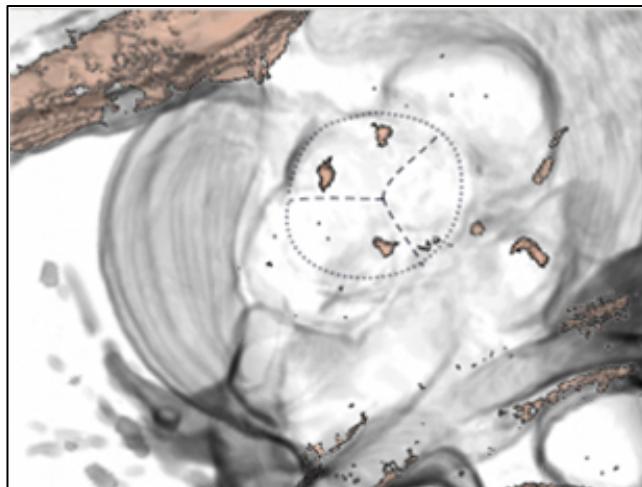


3D Reconstruction of Calcified Valve

Homograft



Freestyle

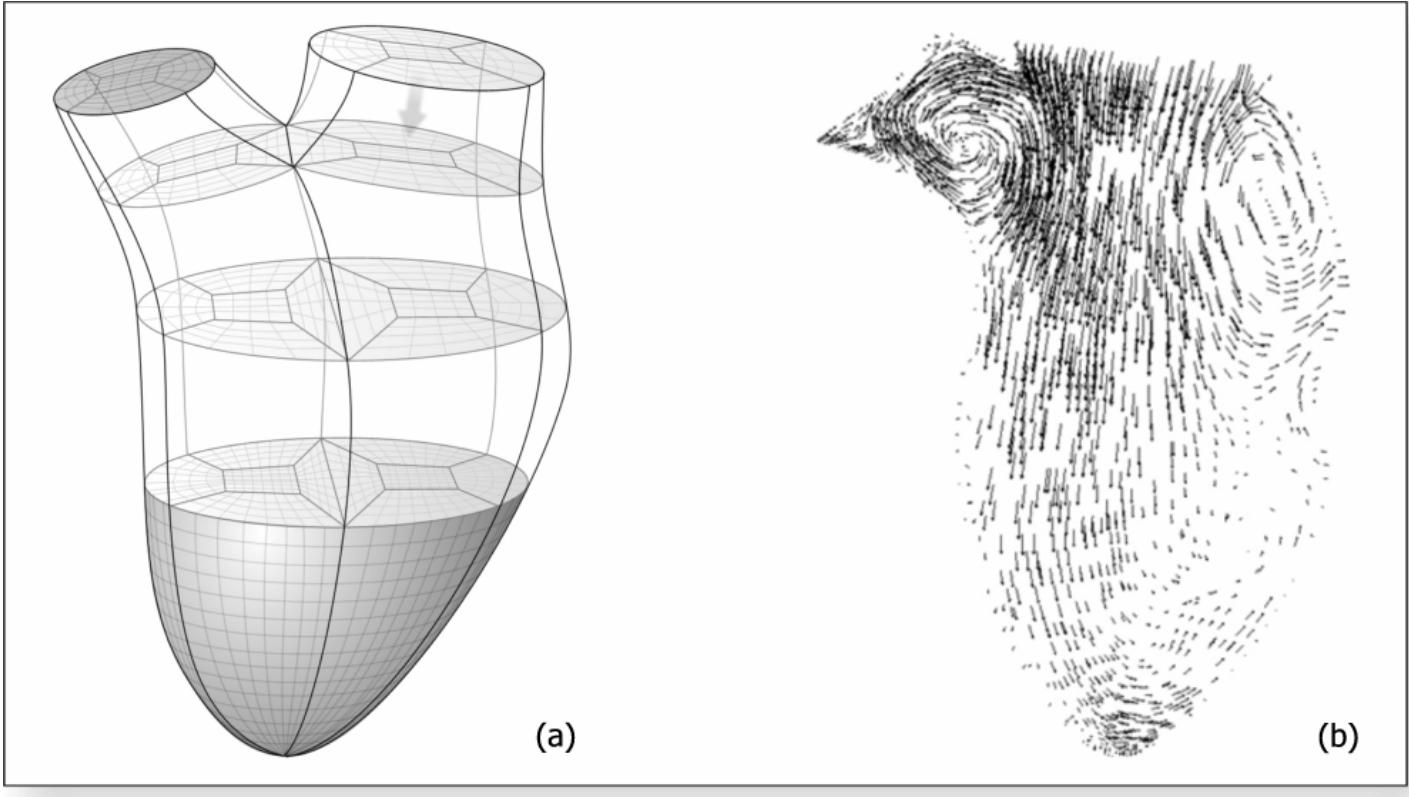


6 months

24 months



A CFD Example



An example of a typical CFD simulation framework, showing of block-structured mesh of a left ventricle (a) and the corresponding simulated flow (b).



COMPUTER GRAPHICS
School of Computer Engineering

Suranaree University
of Technology

Lecture 1 Graphics System

Paramate Horkaew

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

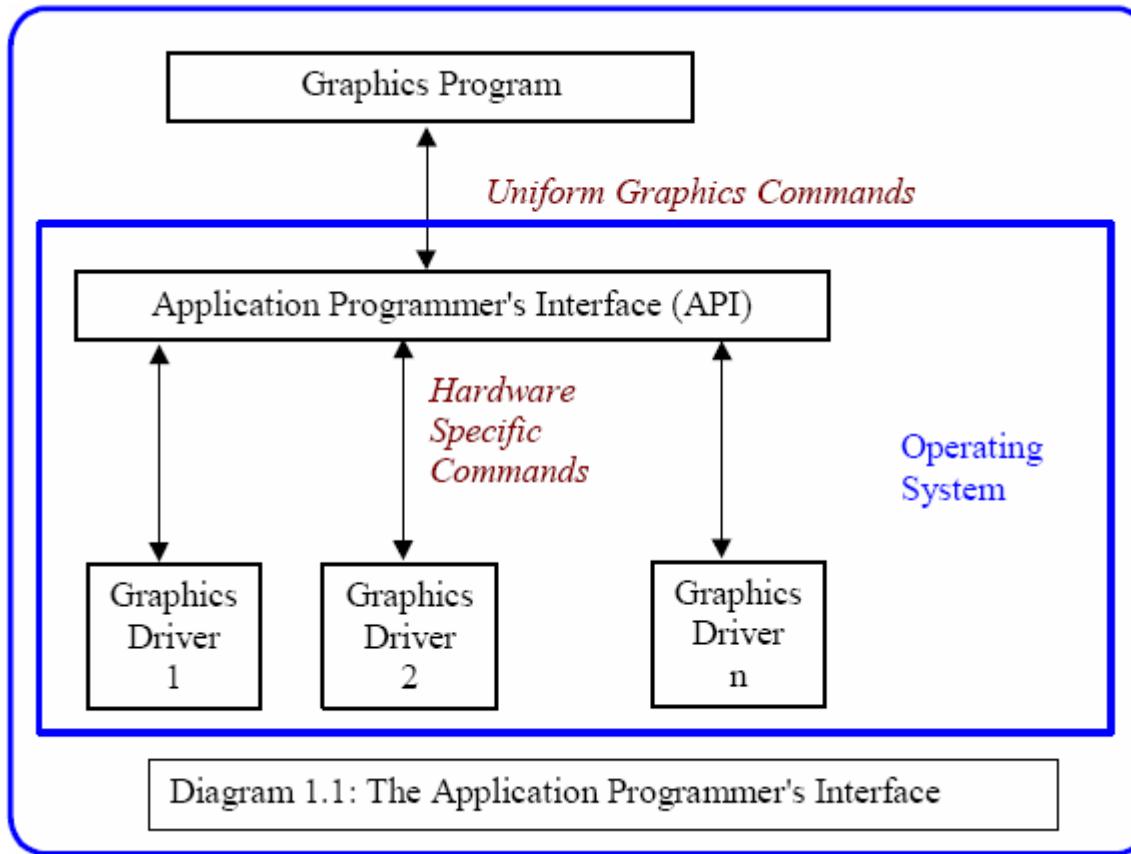


Key Elements of Graphics System

- Processor (Central Processing Unit or Dedicated Hardware)
- Memory
- Frame Buffer
- Output Devices
 - CRT, LCD Monitors
 - 3D Viewing Device
 - Printer
- Input Devices
 - Mouse
 - Keyboards
 - Trackball and Joystick
 - Digitizers and Scanners
- **Algorithms**



Application Programmer's Interface



โปรแกรมกราฟิก ในทางปฏิบัติควรจะ เป็นอิสระ ต่ออุปกรณ์แสดงผลให้มากที่สุด (Device Independent) โดยปกติ OS จะจัด API ให้สำหรับการทำงานพื้นฐาน



Device Dependent or Independent?

Device Dependent

เข้าถึงคำสั่งเฉพาะของอุปกรณ์แสดงผลโดยตรง เหมาะกับงานที่ต้องการความเร็ว สูง เช่น เกมคอมพิวเตอร์ หรือ งานแสดงผลทางอุตสาหกรรม

ข้อเสีย คือ การปรับปรุงระบบทำได้ยาก เสียค่าใช้จ่ายในการพัฒนาสูง

Device Independent

เลือกเฉพาะ คำสั่งที่จำเป็นเพื่อติดต่ออุปกรณ์แสดงผล ให้น้อยที่สุด และกำหนดให้ คำสั่งอื่นๆ อ้างถึงคำสั่งนี้ ซึ่งปกติจะเป็นคำสั่งที่นำข้อมูลใน Frame Buffer ไป แสดงผล ทำให้การศึกษา และ พัฒนาโปรแกรมทำได้ง่าย

ข้อเสีย เสียเวลาในการแปลงชุดคำสั่ง และ สูญเสียความสามารถพิเศษบางอย่าง ของอุปกรณ์แสดงผล เช่น การปรับแต่งค่าในรีจิสเตอร์ เพื่อเข้าถึงหน่วยความจำ



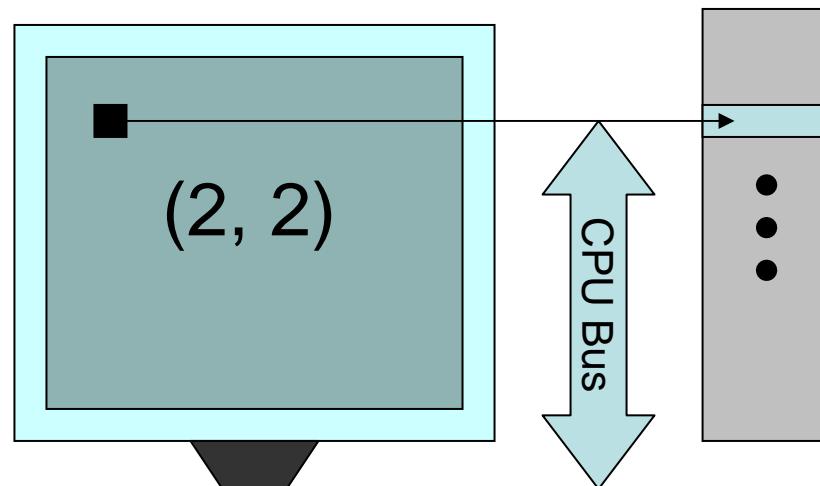
Raster Devices

Raster Devices

คืออุปกรณ์แสดงผล ที่พบรได้โดยทั่วไป ได้แก่ จอภาพ และ เครื่องพิมพ์

ลักษณะจำเพาะคือ แต่ละจุด หรือ **Picture Element (Pixel)** ซึ่งประกอบกันขึ้นเป็นภาพบนอุปกรณ์แสดงผล จะอ้างถึง (Map) หน่วยความจำ แบบ Random Access (RAM) ซึ่งสามารถเข้าถึงได้ โดย CPU หรือ Register ควบคุม

อุปกรณ์แสดงผล

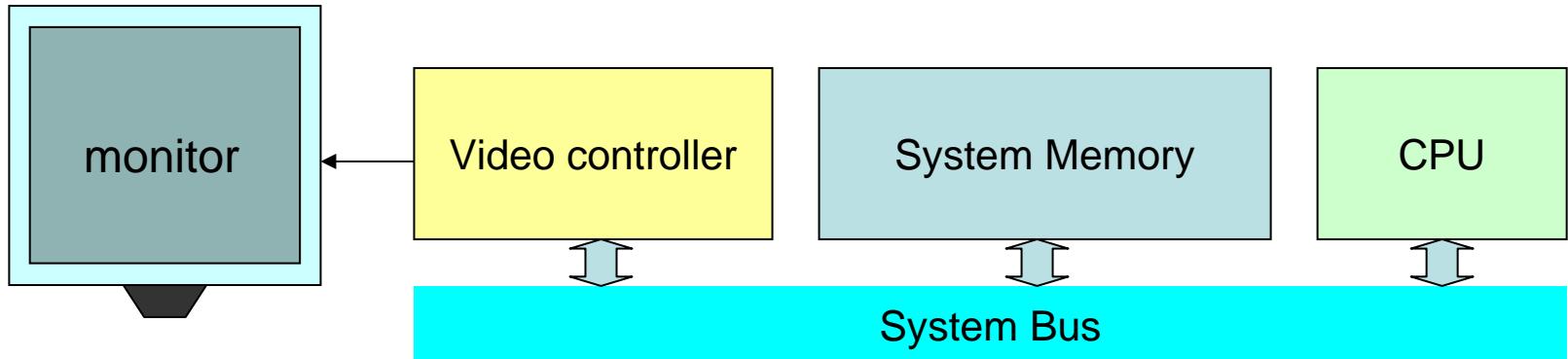


หน่วยความจำบัสเฟอร์

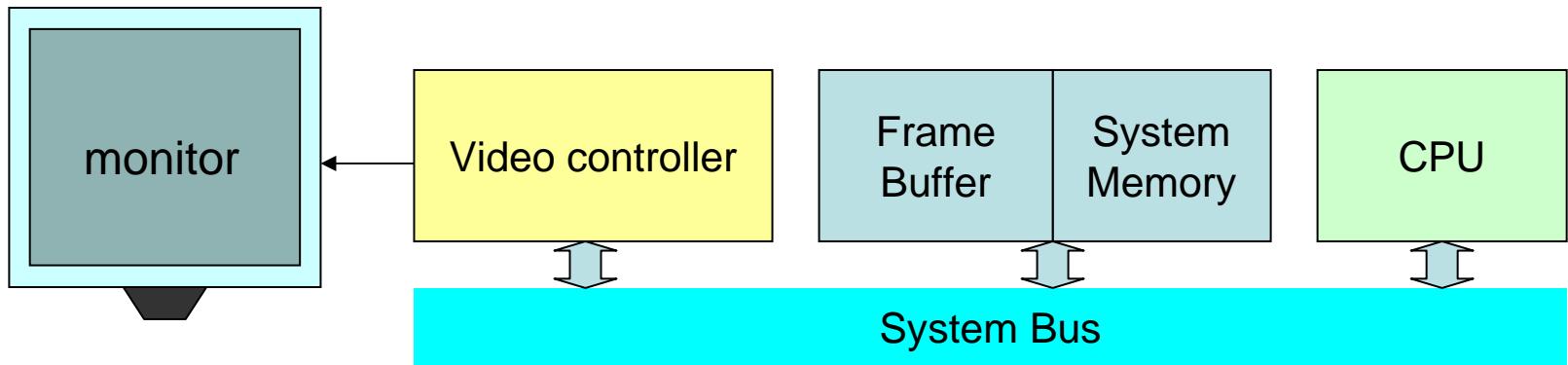


Raster System Architectures

Simple Raster Graphic System



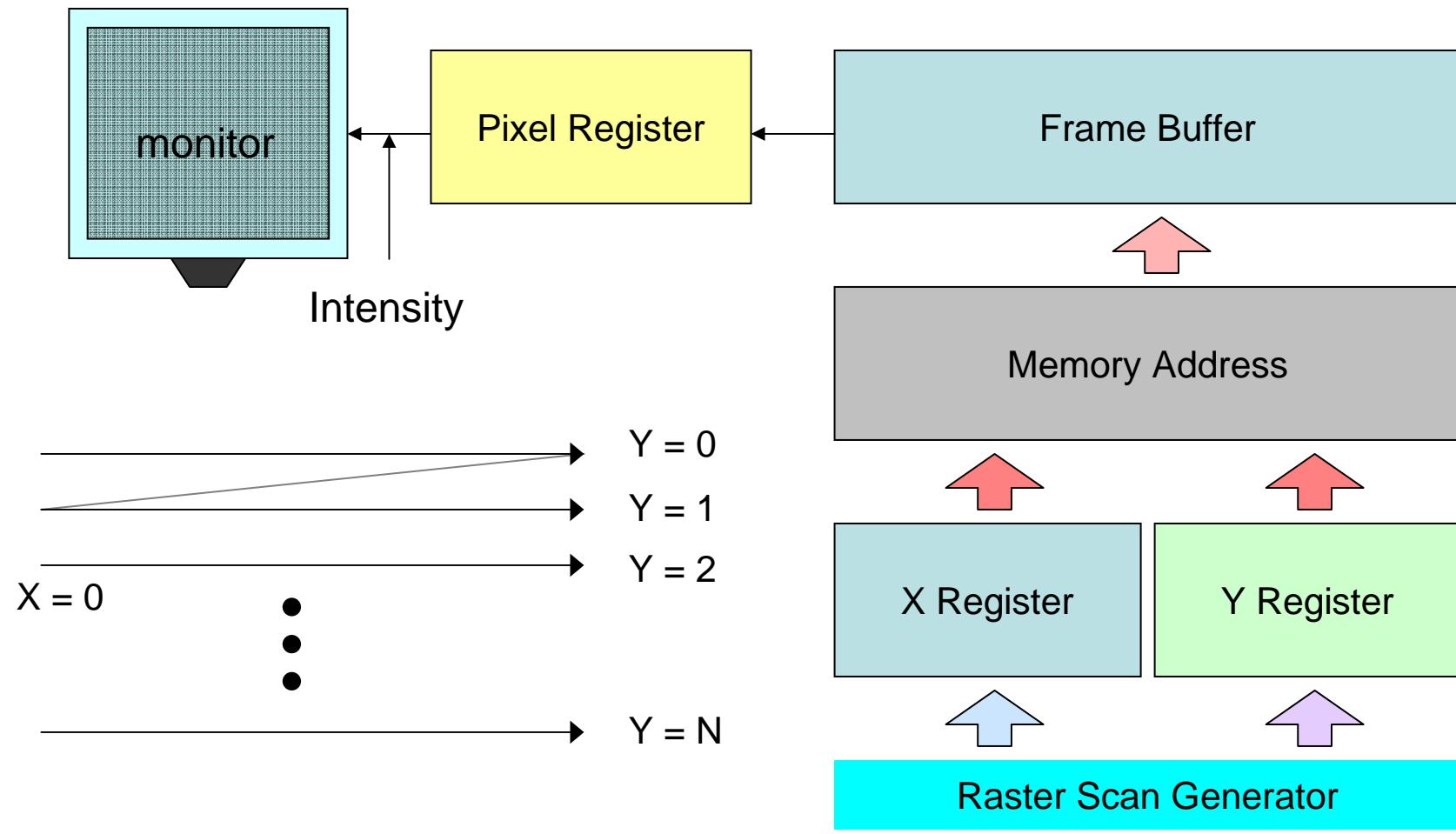
Raster Graphic System with Reserved Frame Buffer





Video Controller

Video Controller Operations



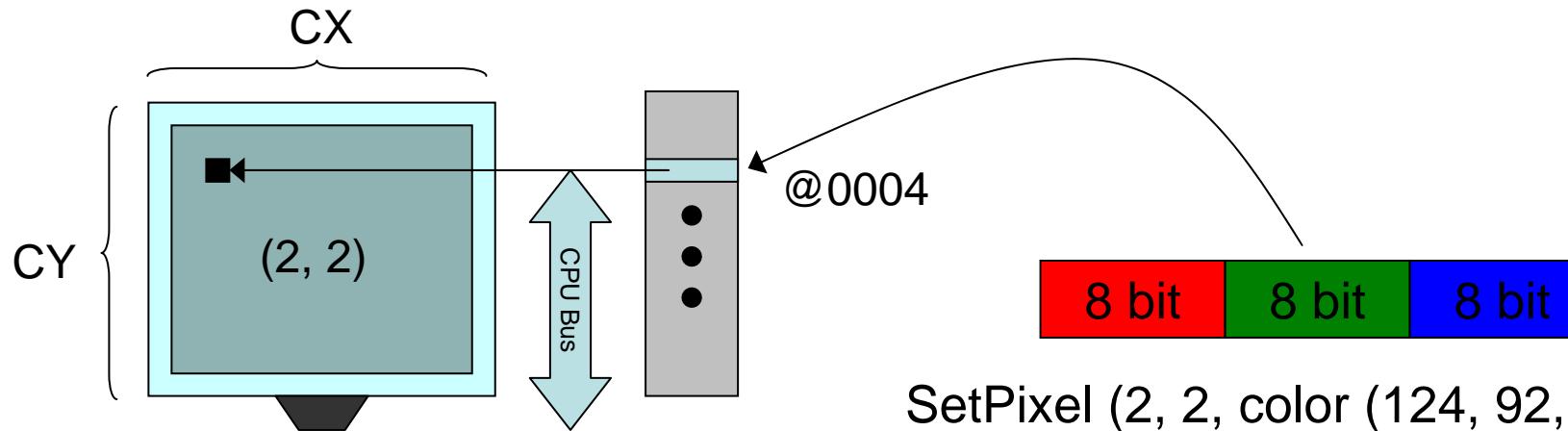


Raster Terminology

Pixel depth คือขนาดแต่ของหน่วยความจำแต่ละตำแหน่ง (บิต) ซึ่งกำหนดพิสัยของความเข้มของจุดภาพนั้น เช่น 1 บิต หมายถึงจุดภาพนั้น ดับ หรือ ส่อง 8 บิต หมายถึง จุดภาพมีความเข้มแตกต่างกัน 256 ระดับ

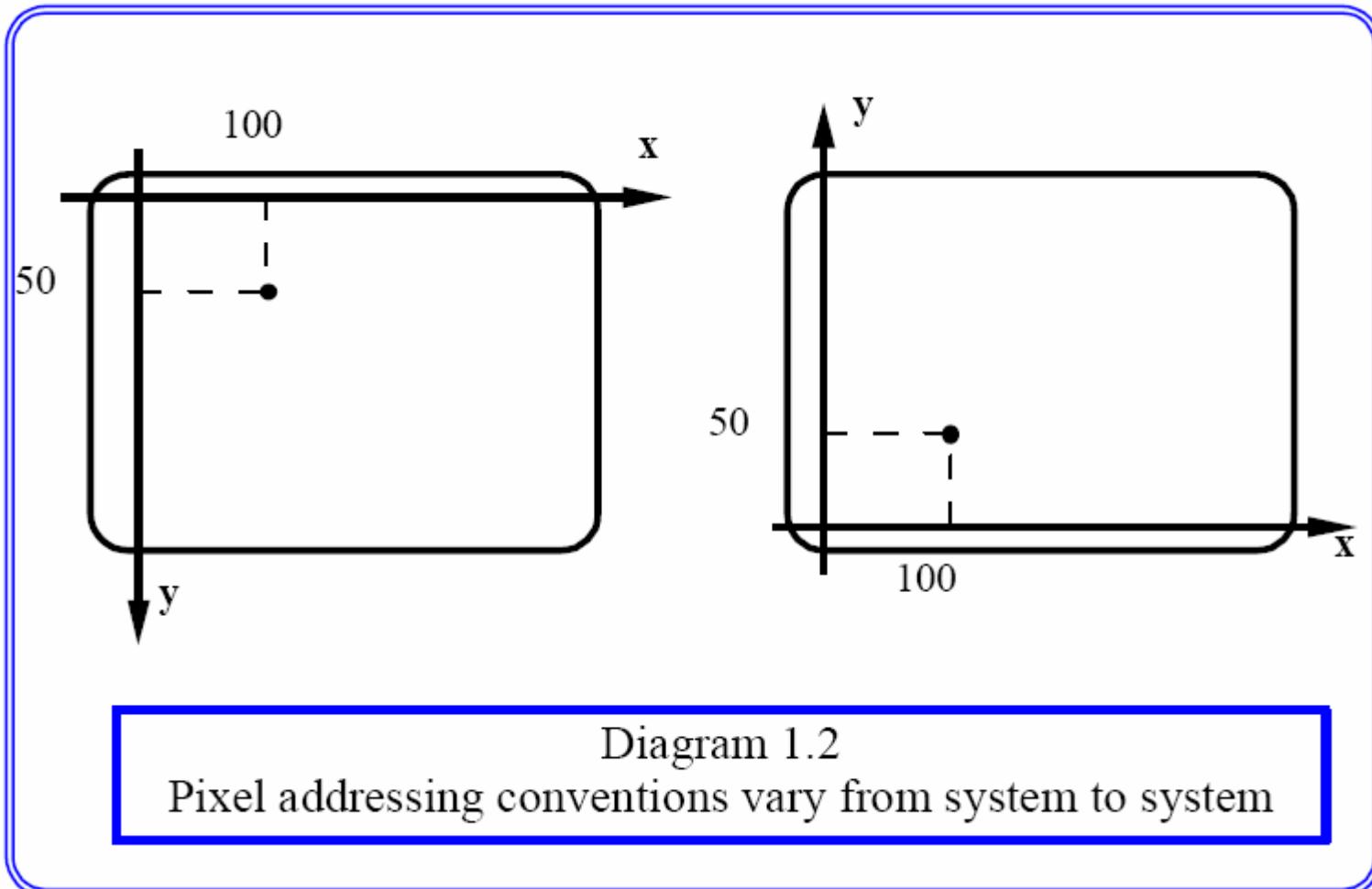
Color คือจำนวนสีพื้นฐาน ที่ประกอบขึ้นเป็นจุดภาพนั้นๆ เช่น จอภาพ CRT มี 3 สี ได้แก่ แดง (R) เขียว (G) น้ำเงิน (B) แต่ละสีมี Pixel depth 8 บิต จะแสดงจุดภาพ ที่มีสีที่สว่างแตกต่างกันได้ 16 ล้านสี หรือ Pixel depth รวม 24 บิต

Resolution คือ จำนวนจุดภาพทั้งหมด ที่ปรากฏบนอุปกรณ์แสดงผล





Addressing Conventions



การเขียนโปรแกรมจำเป็นจะต้องคำนึงถึงการเปลี่ยนแปลง Resolution ด้วย



Coordinate Systems

a) Object or Model or Local Coordinates

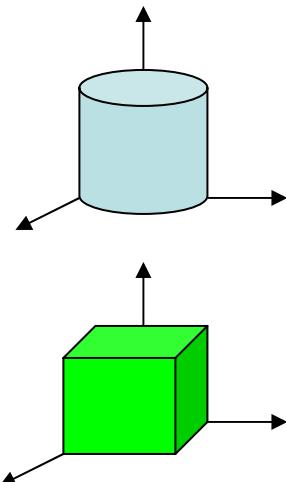
คือระบบปริภูมิที่กำหนดสำหรับการวัดวัตถุแต่ละชิ้น

b) World or Physical Coordinates

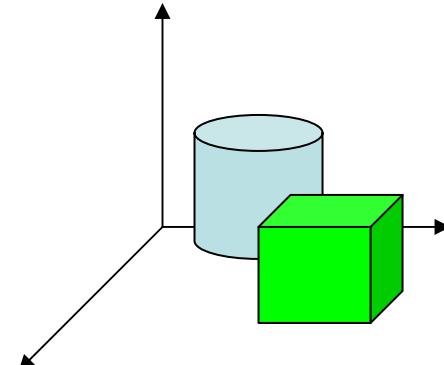
คือระบบปริภูมิที่ เลื่อนวัตถุใดๆ ไปอยู่ในตำแหน่งที่เหมาะสมในจาก

c) Normalized or Device or Screen Coordinates

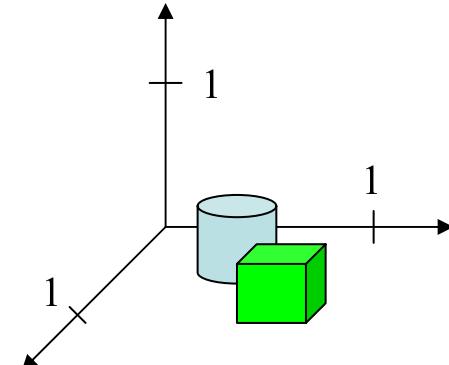
คือระบบปริภูมิที่ใช้สำหรับการวัดวัตถุที่ปรากฏ บนอุปกรณ์แสดงผล



a)



b)



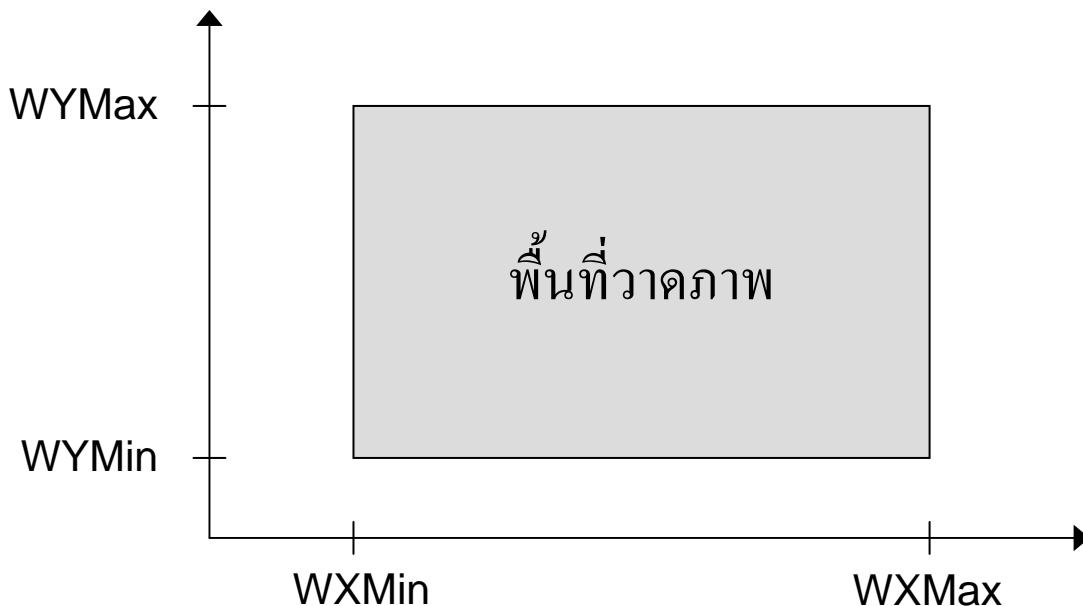
c)



World Coordinate System

การเขียนโปรแกรมแบบ **Device Independent** จะเป็นต้องมีการนิยามระบบปริภูมิ
กายภาพ (World Coordinate System) ซึ่งจะเป็นการอ้างอิงการแสดงผลกับหน่วย
วัดทางกายภาพ ให้เหมาะสมกับการประยุกต์ใช้งานได้แก่ mm, cm หรือ yard

SetWindowWorldCoords (WXMin,WYMin,WXMax,WYMax)





Drawing Graphics Primitives

หลังจากนิยามปริภูมิทางกายภาพแล้ว เราสามารถวาด Graphics Primitive บนบริเวณที่กำหนดได้ ตัวอย่างเช่น

SetPixel (x, y, color);

DrawLine (x1, y1, x2, y2);

DrawCircle (x, y, radius);

DrawPolygon (List of Points (x, y));

DrawText (x1, y1, "Suranaree University of Technology");

นอกจากนี้ เรายังสามารถกำหนดรูปแบบ ของการแสดงผลได้โดย การกำหนดค่าให้กับ Attributes ที่เกี่ยวข้อง เช่น

Line ได้แก่ Style (Solid/Dash หรือ Dot) Thickness (in Pixels) Color

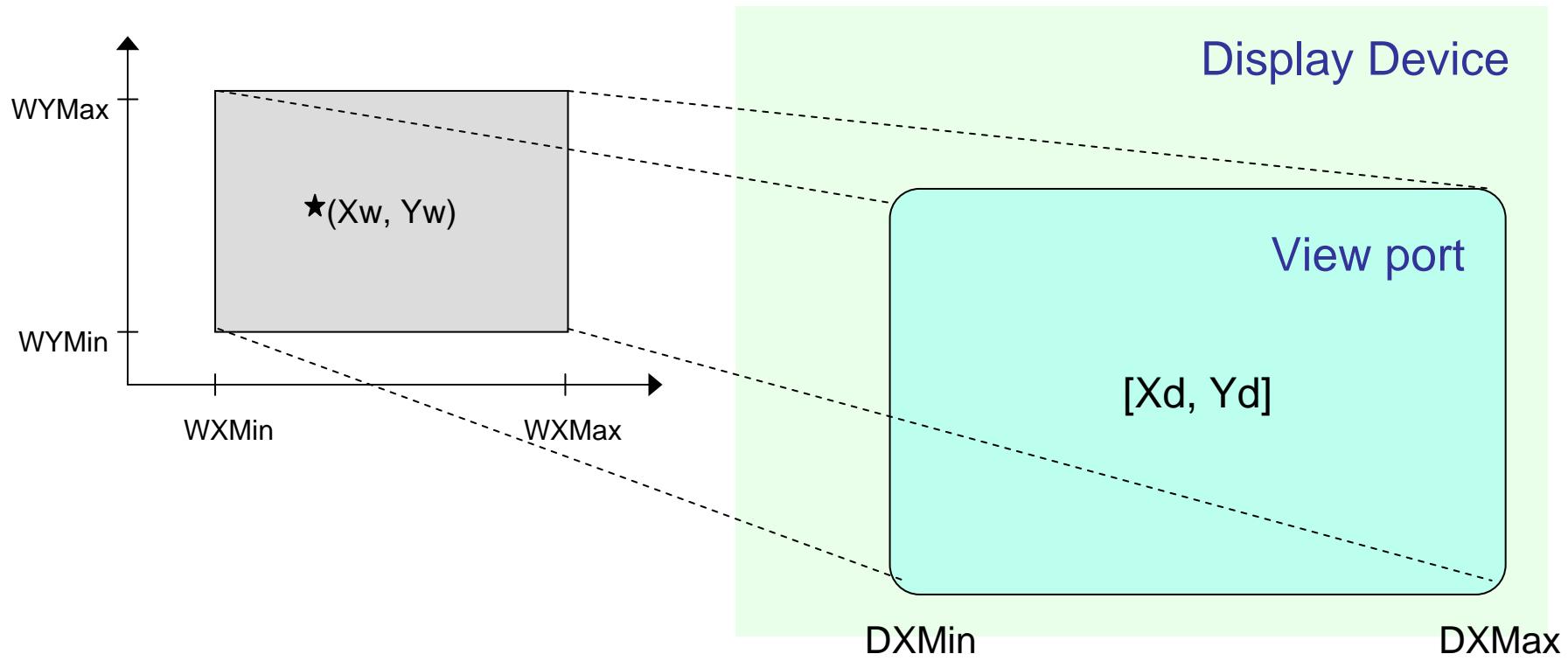
Text ได้แก่ Font, Size Color



Normalization

คือขั้นตอนการแปลง World Coordinate ไปเป็น Normalized หรือ Device Coordinates เพื่อที่จะนำไปแสดงที่อุปกรณ์แสดงผลต่อไป

ขั้นตอนนี้สามารถคำนวณได้โดยใช้หลักสมการเชิงเส้นพื้นฐาน





Normalization

การกำหนดค่าของ View port สามารถทำได้ในทำงานเดียวกัน

SetDisplayCoords (DXMin,DYMin,DXMax,DYMax)

ความสัมพันธ์ระหว่าง World และ Device coordinates เขียนได้เป็น

$$(X_w - W_{Xmin}) / (W_{Xmax} - W_{Xmin}) = (X_d - D_{Xmin}) / (D_{Xmax} - D_{Xmin})$$

จัดข้างสมการแล้วจะได้ว่า

$$X_d = X_w * A + B$$

โดยที่ $A = (D_{Xmax} - D_{Xmin}) / (W_{Xmax} - W_{Xmin})$

$$B = - W_{Xmin} (D_{Xmax} - D_{Xmin}) / (W_{Xmax} - W_{Xmin}) + D_{Xmin}$$



Input Devices

สำหรับ Computer Graphics มีอุปกรณ์รับข้อมูลหลายชนิดได้แก่

Mouse, Joystick, Keypad, Digitization Tablet, Light Pen, etc

ในที่นี้เราจะพิจารณาเฉพาะ Mouse เท่านั้น

Mouse คืออุปกรณ์ที่ส่งข้อมูล อย่างน้อย 3 ชนิดไปยัง Operating System

- ตำแหน่งสัมพัทธ์ที่เลื่อนไปตามแกน x
- ตำแหน่งสัมพัทธ์ที่เลื่อนไปตามแกน y
- สถานะของปุ่ม

หมายเหตุ เครื่องหมาย Mouse ที่ปรากฏบนจอภาพเป็นสิ่งที่สร้างขึ้นโดย software ซึ่งอาศัยข้อมูลดังกล่าวข้างต้น



Event Driven Mouse

Mouse Event จะเกิดขึ้นก็ต่อเมื่อ ข้อมูลได้ข้อมูลหนึ่งใน 3 ชนิดได้เปลี่ยนสถานะ

ในขั้นตอนนี้ Mouse จะส่งสัญญาณ Interrupt ไปยังระบบปฏิบัติการ พร้อมกับ ข้อมูล ของสถานะในปัจจุบัน

หลังจากนั้นระบบปฏิบัติการจะปรับตำแหน่งของเครื่องหมาย Mouse บนอุปกรณ์ แสดงผล พร้อมกับ ส่งข้อมูล Mouse ไปยัง โปรแกรมกราฟิกเพื่อประมวลผลต่อไป

สำหรับระบบปฏิบัติการ **Windows** จะเรียก Mouse Event ว่า Message ซึ่งจะ ส่งไปโปรแกรมประยุกต์ พร้อมกับ Interrupt อีก

Mouse Message ประกอบด้วย **Message ID** ซึ่งระบุการเปลี่ยนแปลงที่เกิดขึ้น เช่น WM_MOUSEMOVE, WM_LBUTTONDOWN และ **IParam** และ **wParam** (Parameter ขนาด 32 และ 16 บิต ตามลำดับ) ซึ่ง โดยปกติ **IParam** จะเก็บค่าพิกัด X และ Y ขณะนั้นไว้ที่ตำแหน่ง 16 บิตล่าง และ 16 บิตบน ตามลำดับ



Event Loop

ในระบบปฏิบัติการ Windows โปรแกรมกราฟิก จะเป็นจะต้องมีการวนลูป (Event Loop) เพื่อรับเหตุการณ์ดังนี้

```
while (GetMessage (&msg, NULL, 0, 0))  
{  
    TranslateMessage (&msg);  
    DispatchMessage (&msg);  
}
```

เมื่อโปรแกรมกราฟิกรับ Mouse Event (Message) ได้จาก DispatchMessage ก็จะส่งไปยังฟังก์ชันประมวลผล Message ต่อไป (Callback Function) ดังนี้

```
LRESULT phkwnddisp2d::WndProc (UINT iMessage, WPARAM wParam, LPARAM lParam)  
{  
    switch (iMessage)  
    {  
        case WM_LBUTTONDOWN : /* Process Mouse Event when the Left Button is pressed */  
            SetCapture (m_hWnd);  
            m_px = LOWORD (lParam);  
            m_py = HIWORD (lParam);  
            break;  
    }  
}
```



Overview of Windows Programming (I)

main () หรือ WinMain ()

```
int PASCAL WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpszCmd, int nCmdShow)
{
    static char          szAppName[] = "Medical Image Registration" ;
    HWND                hwnd;
    MSG                 msg;
    WNDCLASS            wndclass;
    HANDLE              hAccel;

    if (!hPrevInstance)
    {
        wndclass.style           = CS_HREDRAW | CS_VREDRAW ;
        wndclass.lpfnWndProc     = (WNDPROC) WndProc;
        wndclass.cbClsExtra      = 0 ;
        wndclass.cbWndExtra      = 0 ;
        wndclass.hInstance        = hInstance ;
        wndclass.hIcon            = LoadIcon (NULL, IDI_APPLICATION);
        wndclass.hCursor          = LoadCursor (NULL, IDC_ARROW) ;
        wndclass.hbrBackground    = (HBRUSH)(COLOR_APPWORKSPACE + 2);
        wndclass.lpszMenuName     = NULL;
        wndclass.lpszClassName     = szAppName ;

        RegisterClass (&wndclass);
    }

    RegisterChildWindows (hInstance);
```



Overview of Windows Programming (II)

```
hwnd = CreateWindow (szAppName,
"Medical Image Registration 1.0",
WS_OVERLAPPEDWINDOW,
0, 0, 640, 480, NULL, NULL, hInstance, NULL);

ShowWindow (hwnd, nCmdShow) ;
UpdateWindow (hwnd) ;

hAccel      = LoadAccelerators( hInstance, MAKEINTRESOURCE ( PROGRAM_ACCEL ) );
m_hInstance = hInstance;

while (GetMessage (&msg, NULL, 0, 0))
{
    if (!TranslateAccelerator (hwnd, (HACCEL) hAccel, &msg))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
}

UnRegisterChildWindows (hInstance);

return msg.wParam ;
}
```



Callback Function

```
LRESULT FAR PASCAL _export WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    int ret;

    switch (message)
    {
        case WM_CREATE      : HDC      hdc;
                               hdc = GetDC (hwnd);
                               Ellipse (0, 0, 10, 20);
                               ReleaseDC (hwnd, hdc);
                               break;

        case WM_DESTROY     : PostQuitMessage (0);
                               return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

ฟังก์ชัน Callback นี้จะวาดวงรี ซึ่งมีแกนขนาด 10 คูณ 20 จุดภาพ ณ ตำแหน่ง มุมบนซ้ายของจอ (0, 0)