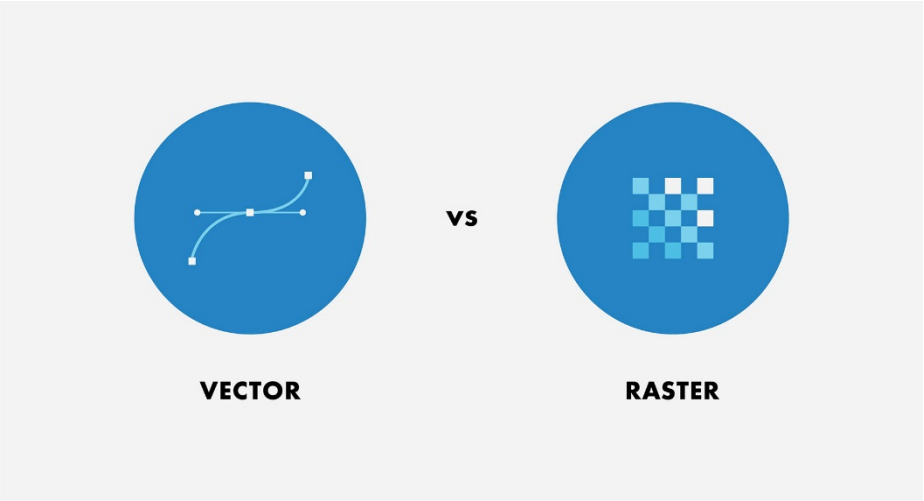
Computer Graphics: Raster Graphics

Dept. of Game Software
Yejin Kim

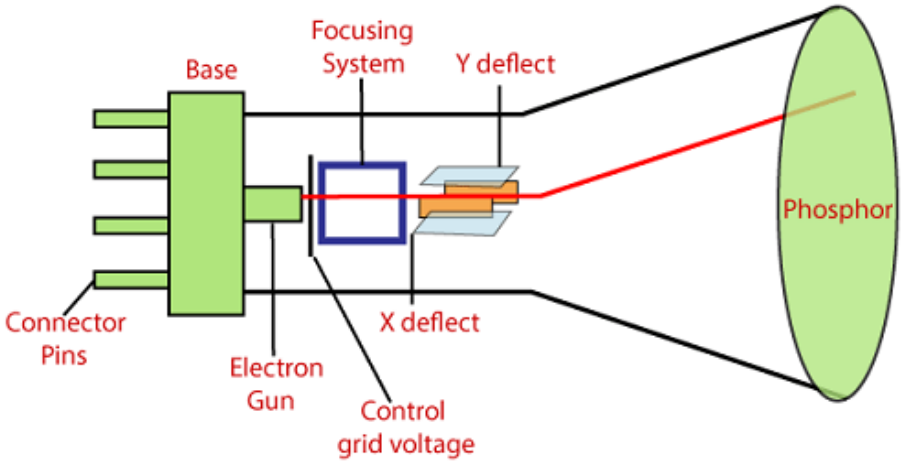
Overview

- Video Display Devices
- Raster Graphics
- Color Models
- 3D Mesh Representation
- Windows Programming
- Tutorials

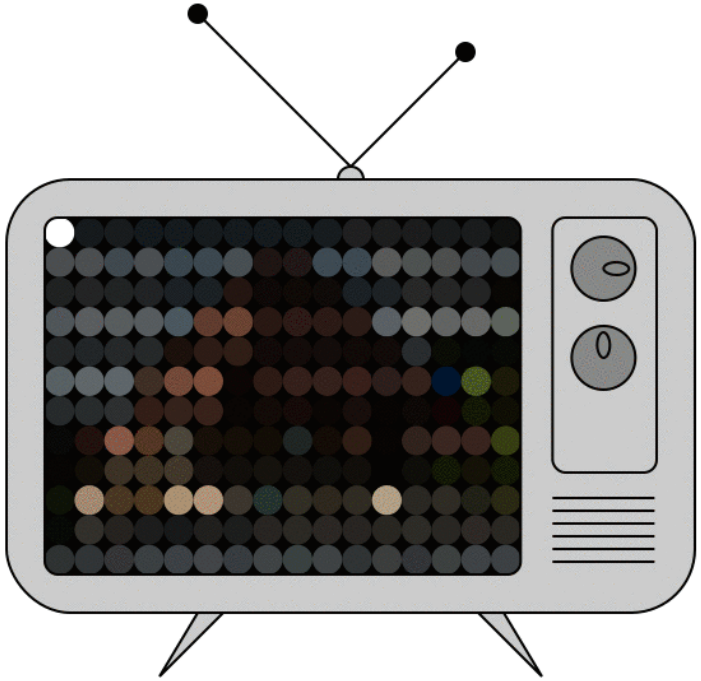


Video Display Devices

- Cathode-ray tube (CRT)
  - A vacuum tube containing one or more electron guns, which emit electron beams that are manipulated to display images on a phosphorescent screen



Cathode Ray Tube (CRT)



CRT interlacing

Video Display Devices

- Liquid-crystal display (LCD)
  - A flat-panel display that uses the light-modulating properties of liquid crystals (controlled by electric power) combined with polarizers and color filters (RGB)
  - Produces images in **pixels** from a backlight

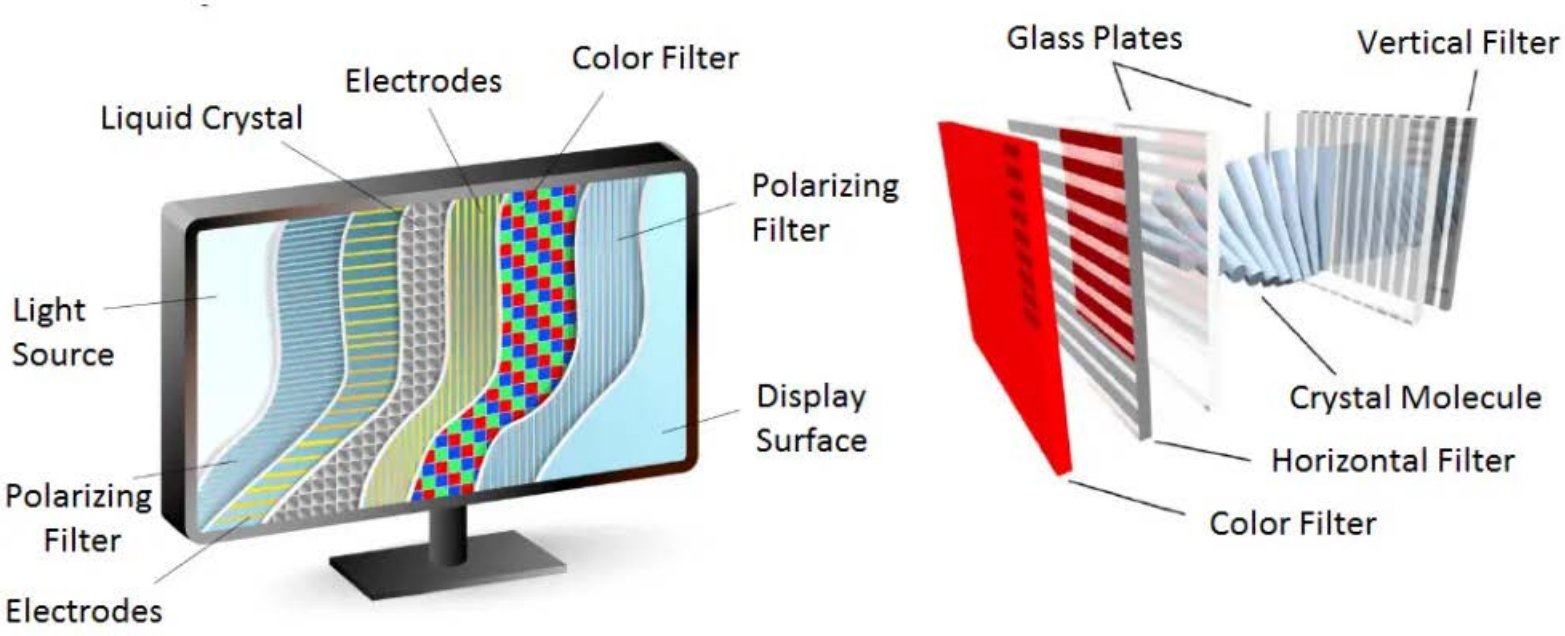
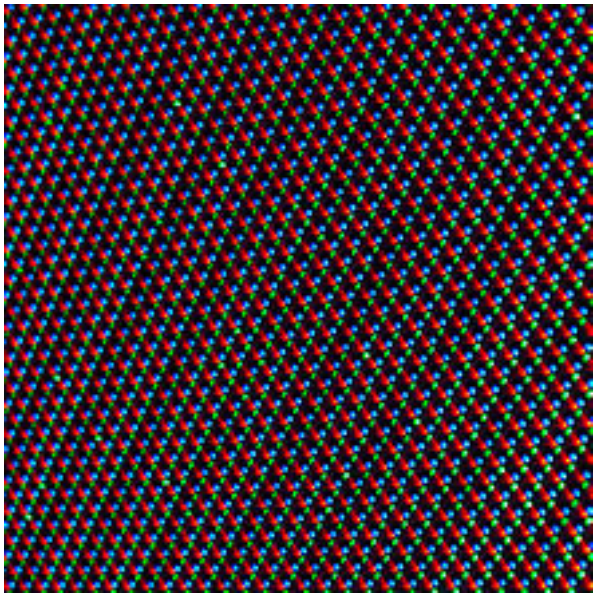


Fig: Liquid Crystal Display (LCD)

Video Display Devices

- Light-emitting diodes (LED)
  - A flat panel display that uses an array of LEDs in pixels for a video display



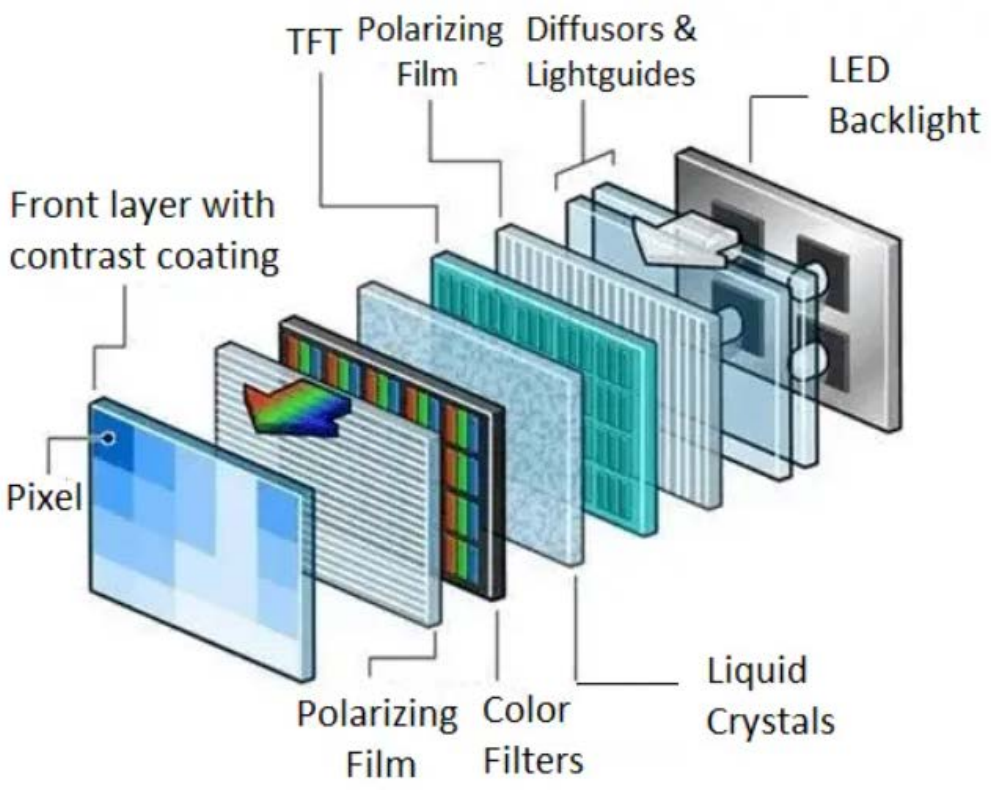
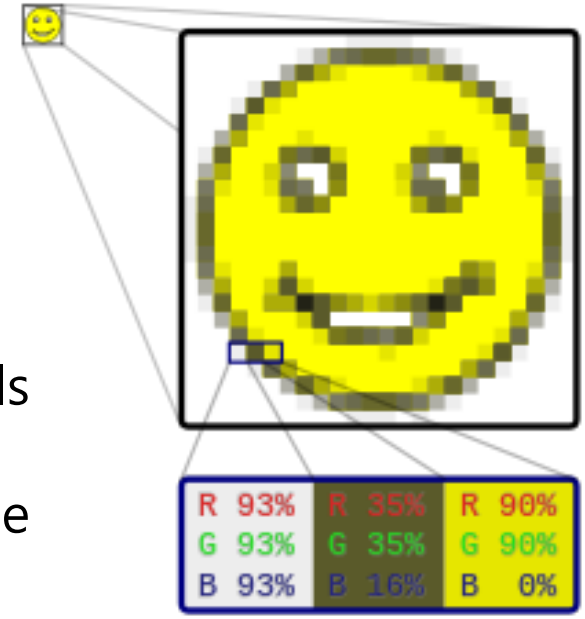


Fig: Light Emitting Diode (LED)

Raster Graphics

- Raster graphics
  - Represents two-dimensional picture as a grid of square pixels, viewable via a computer display

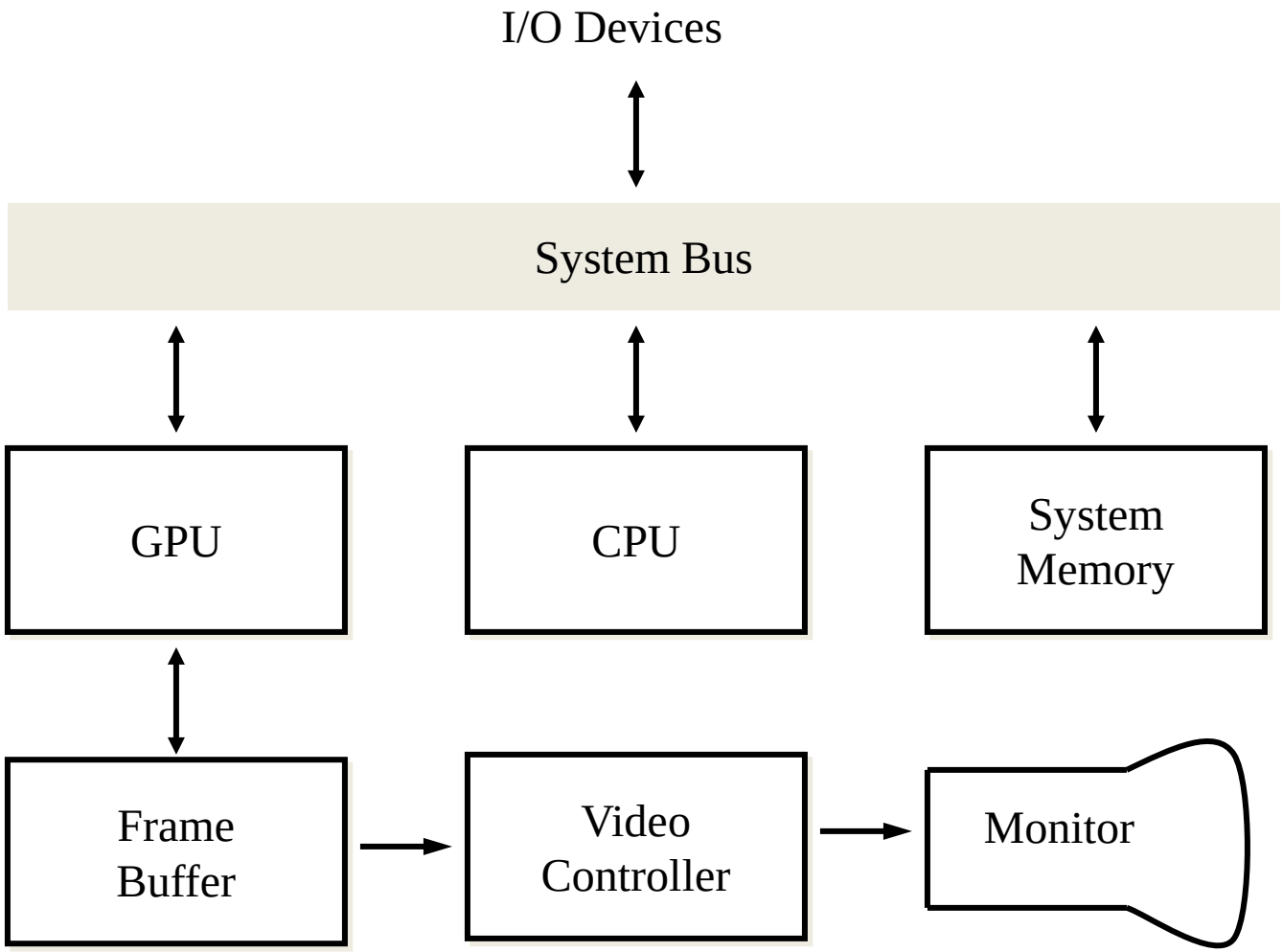
- Rasterization
  - The task of converting a shape into a raster image (as a bitmap file format)
  - May refer to either the conversion of models into raster files, or the conversion of 2D rendering primitives such as polygons or line segments into a rasterized format



Raster image

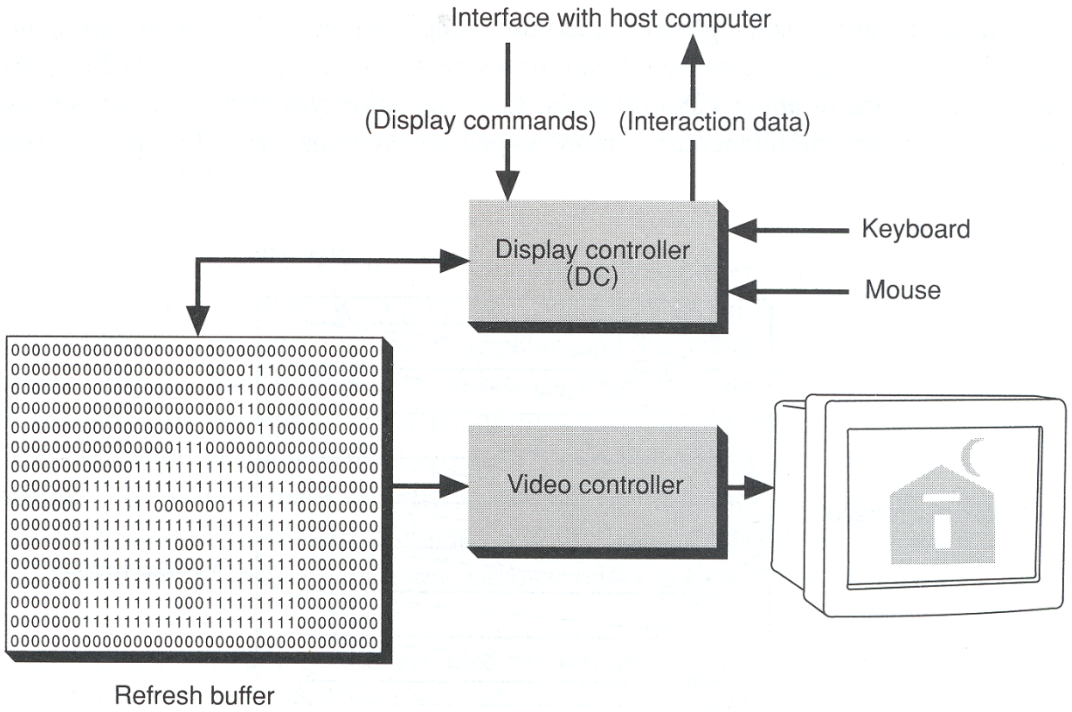
Raster Graphics

- Raster graphics systems



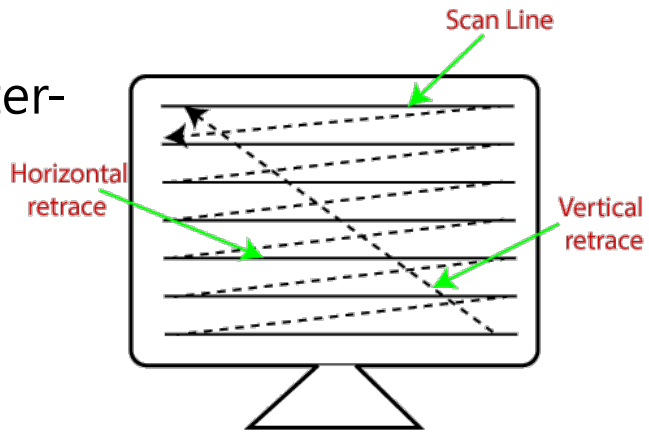
Raster Graphics

- Frame buffer
  - A memory buffer containing data (as a bitmap) which represents color values for every pixel to be shown on the display
  - The total amount of memory required for the frame buffer depends on the resolution of the output signal and the color depth
    - 24-bit for true color (8-bit for RGB) + 8-bit for alpha value (transparency)



Raster Graphics

- (Vertical) Refresh rate
  - The number of times per second that a raster-based display device displays a new image
  - A flickering rate for CRT displays
  - A updating rate for LCD/LED displays (60/144/165/240 Hz)
  - Dynamic refresh rate: FreeSync, G-Sync



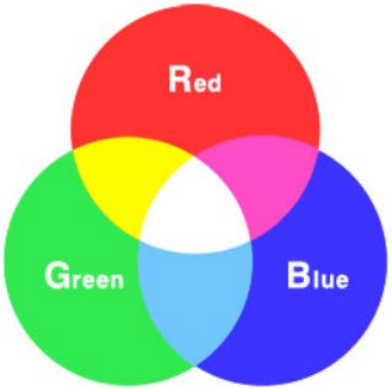
CRT sweeping

- Frame rate (frames per second)
  - Describes how many images are stored or generated every second by the device driving the display
  - 24/30/60/120 FPS


12 FPS

Color Models

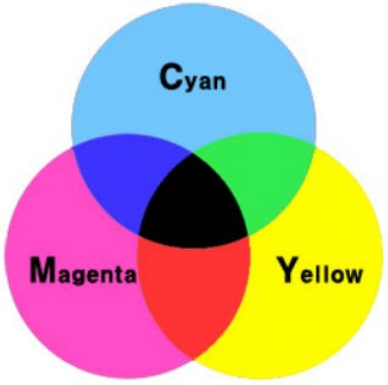
- RGB color model
  - An additive color model combining the RGB colors of light to a black background
    - Wavelength of red + green → wavelength of yellow
    - Idle for video display using light



RGB

Additive color with RGB

- CMYK color model
  - A subtractive color model partially or entirely masking colors on a white background
    - The ink subtract the RGB colors from white light
    - Imperfect black by combining CMY colors → adding a black (key) color (CMYK)
    - Idle for color printing

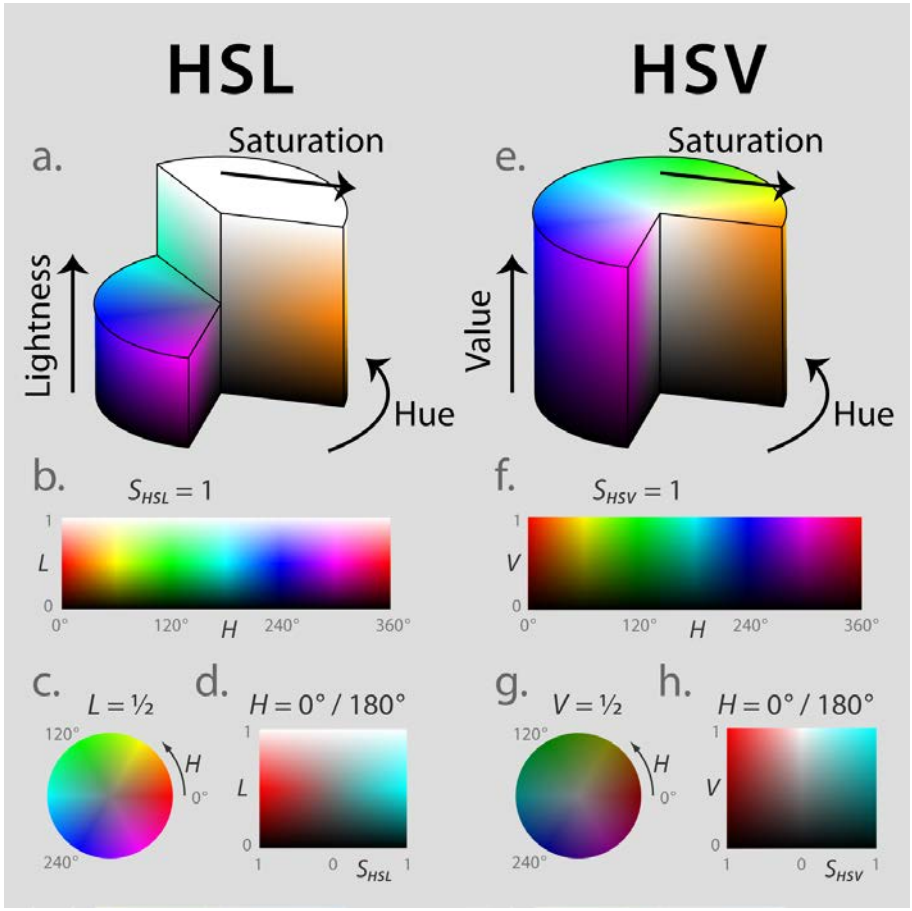


CMYK

Subtractive color with CMY

Color Models

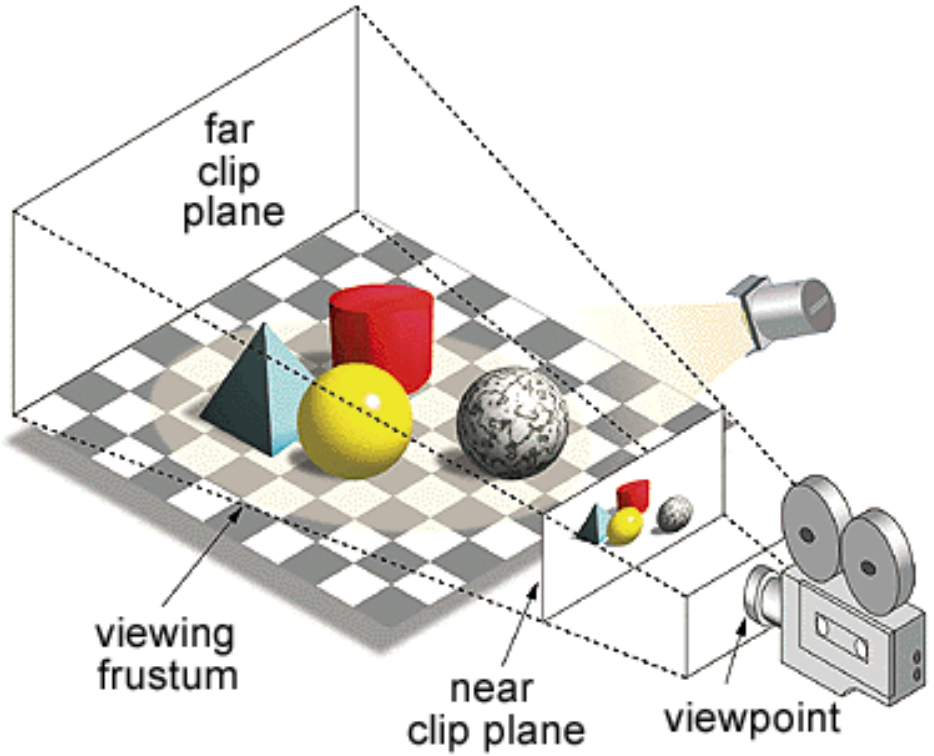
- Color perception usually involves three quantities
  - **Hue**: Distinguishes between colors like red, green, blue, etc
  - **Saturation**: How far the color is from a gray of equal intensity
  - **Lightness**: The perceived intensity of a reflecting object
  - Sometimes lightness is called **brightness** if the object is emitting light instead of reflecting it



Hue, Saturation, Light (HSL) &
Hue, Saturation, Value (Brightness (HSV/HSB)

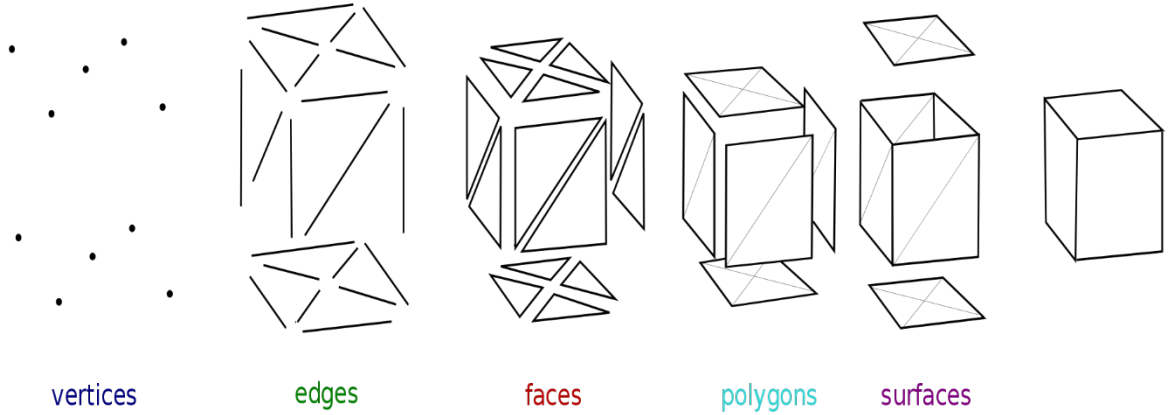
3D Mesh Representation

- 3D virtual scene: Depth-buffered triangle rasterization
  - Virtual camera and light sources
  - Visual properties: materials, textures, etc.
  - → Solving the rendering equation (shading equation)

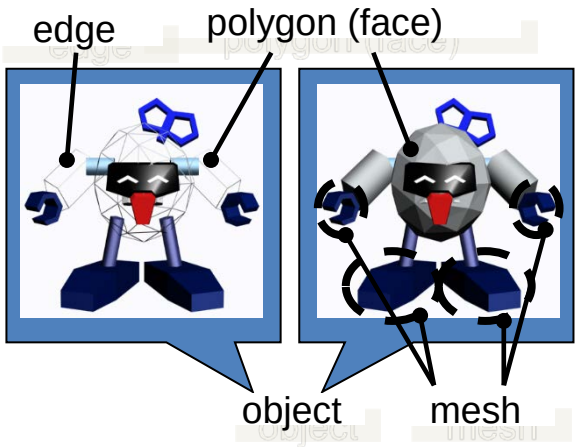


3D Mesh Representation

- Polygon
  - A collection of vertices and edges: triangles, quadrilaterals(quads)
  - HW support for rendering: 3 or 4-sided faces

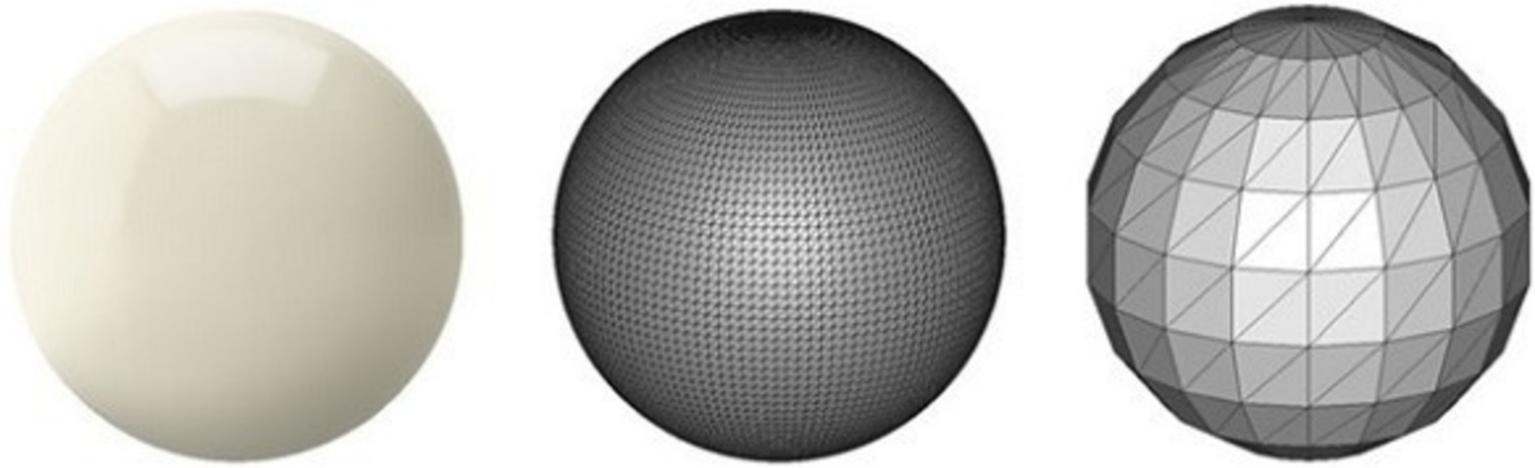


- (Polygonal) Mesh
  - Surface: a collection of polygons
- Object
  - A collection of meshes



3D Mesh Representation

- Surface representations
  - Analytical form (분석적 형태)
  - Collection of patches (패치들의 집합)
  - **Triangle mesh** (삼각 메쉬)

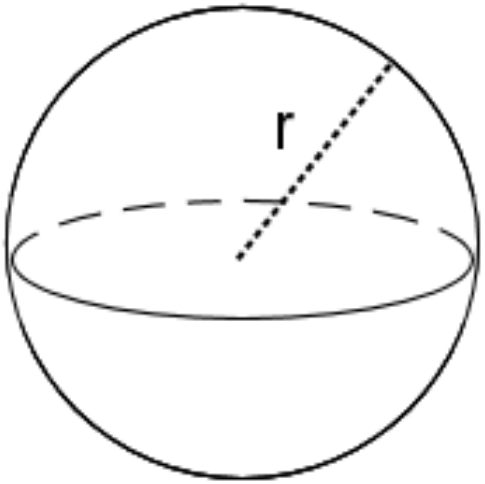


3D Mesh Representation

- Analytical form
  - Parametric surface equation (표면 방정식)

    e.g. Sphere equation:

$$x^2 + y^2 + z^2 = r^2$$

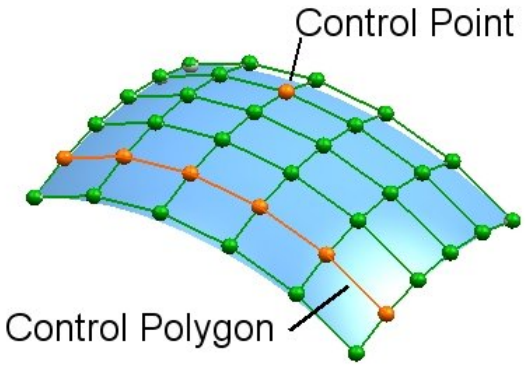


3D Mesh Representation

- Collection of patches
  - Patch: a curved plane composed of a set of rectangles
  - Similar to quilt

    e.g. NURBS, Bezier surfaces, subdivision surface

Quilt

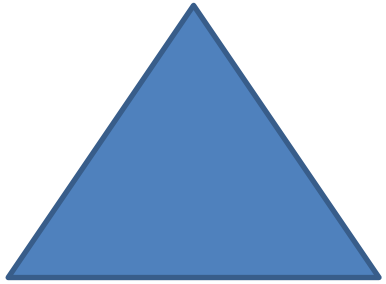


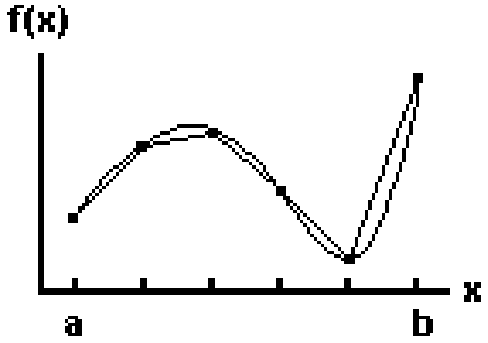
NURBS surface

NURBS modeling

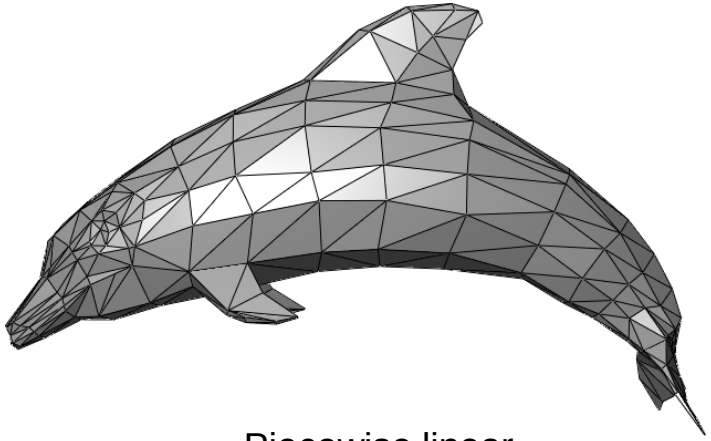
3D Mesh Representation

- Triangle mesh
  - Simplest type of polygon
  - Always planar
  - Still a triangle after transformations
    - e.g. Affine (projective) transformation
  - Hardware acceleration support
  - Piecewise-linear approximation(구분적 선형 근사)
    - used for curved objects





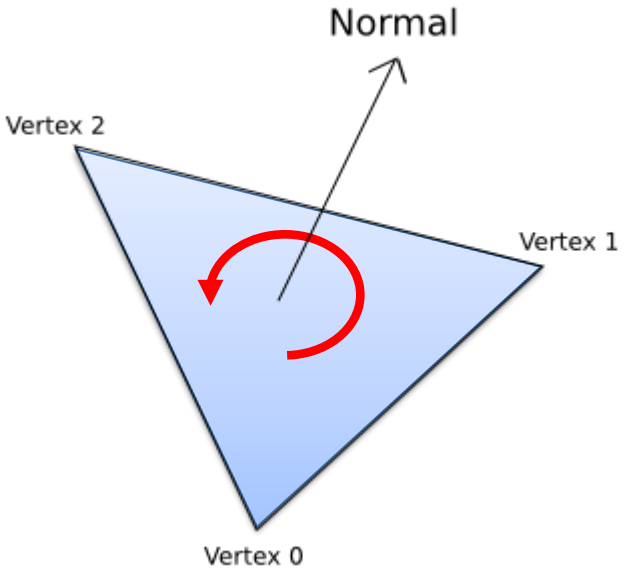
Piecewise linear
approximation to a function



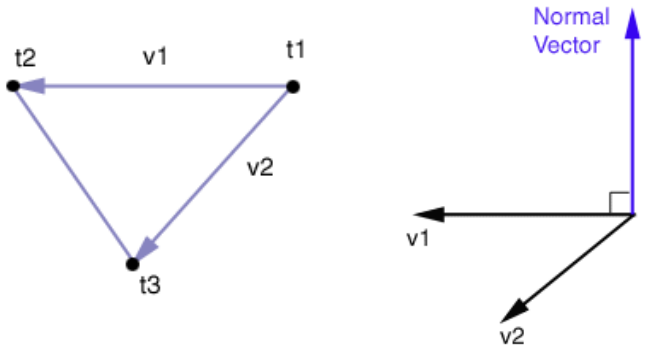
Piecewise linear
approximation to a surface

3D Mesh Representation

- Winding order of triangle mesh
  - Decide a polygon (front or back) side
  - Counterclockwise(CCW) or clockwise(CW)



winding order: CCW



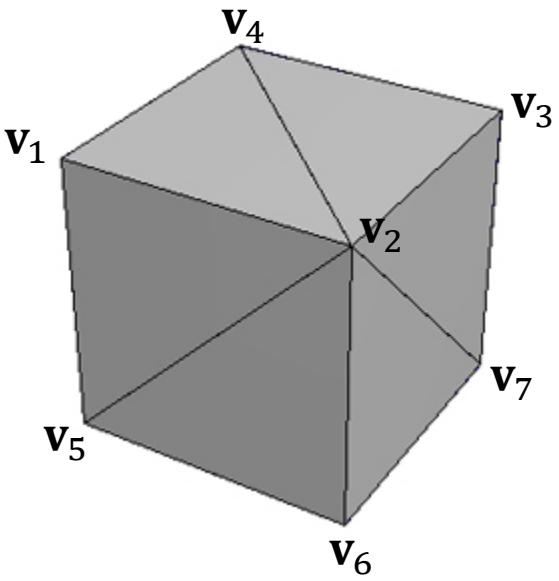
$$\mathbf{n} = \frac{\mathbf{v}_1 \times \mathbf{v}_2}{|\mathbf{v}_1 \times \mathbf{v}_2|}$$

computing normal direction

3D Mesh Representation

- Triangle mesh representation
  - Indexed triangle list

$\mathbf{v}_i$: vertex 3D position



	0	1	2	3	4	5	6	7
vertex list:	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_4$	$\mathbf{v}_5$	$\mathbf{v}_6$	$\mathbf{v}_7$	$\mathbf{v}_8$

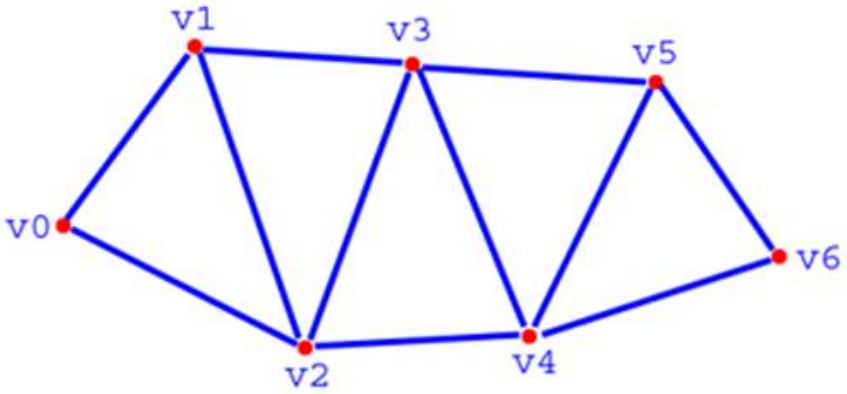
indexed triangle list:	0	1	3	1	2	3	0	4	1	…

3D Mesh Representation

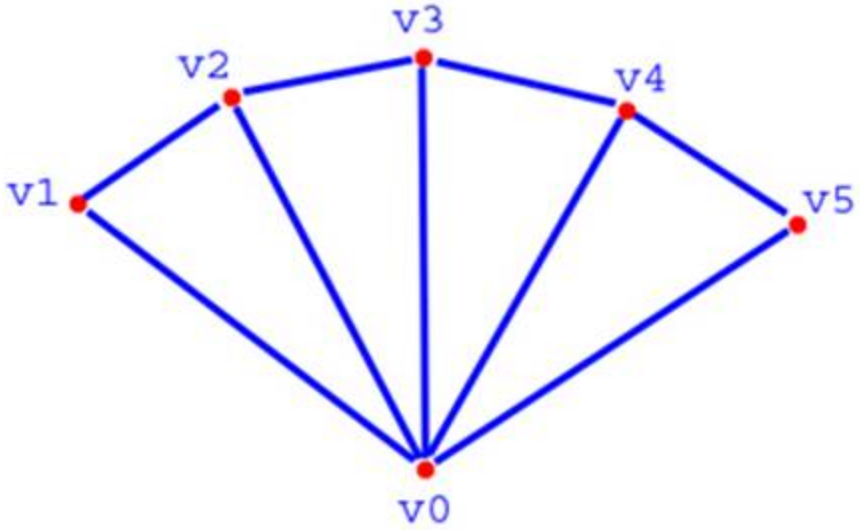
- Triangle mesh representation
  - Triangle strip
  - Triangle fan

Vertex list:

$\mathbf{v}_0$	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_4$	$\mathbf{v}_5$	$\mathbf{v}_6$



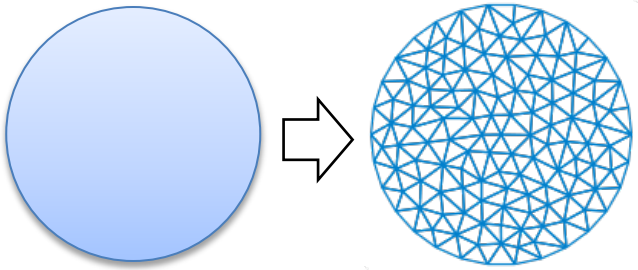
triangle strip



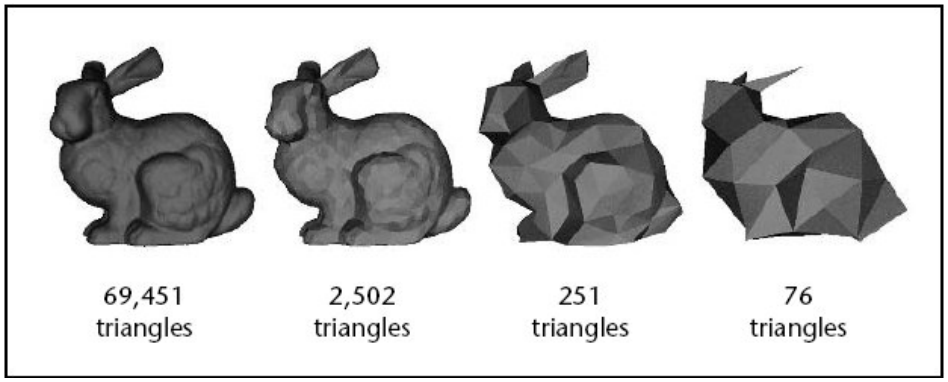
triangle fan

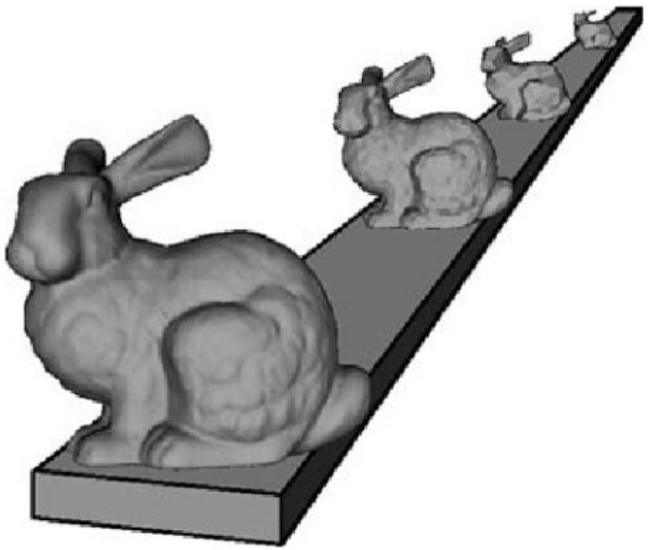
3D Mesh Representation

- Tessellation of triangule mesh
  - Divide a surface into a collections of triangles



e.g. Level-of-detail(LOD)

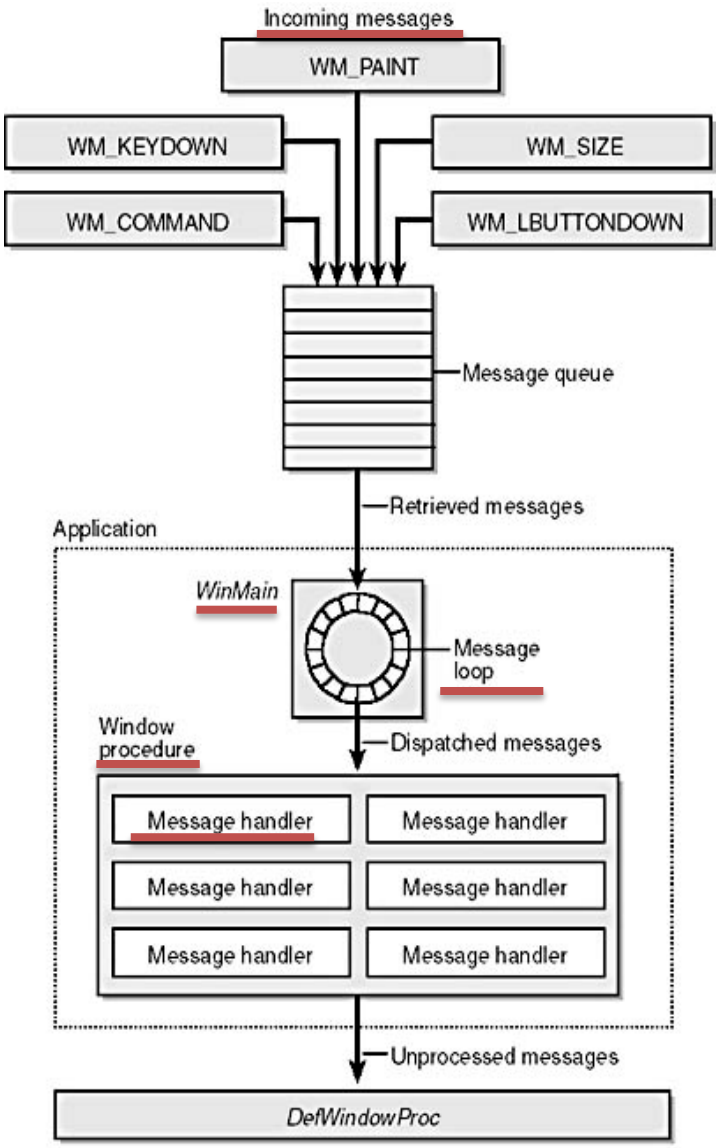




Windows Programming

- Windows (Event-driven) programing
  - **Event**: State change of input device or program inside
  - **Message**: Form of state change to the program
  - **Message handler**: Function handling event

- Win32 program structure
  - Begins with WinMain() function
  - Starts a massage loop in the WinMain() for waiting messages
  - Gets messages from operating system, a user or the program
  - Messages are processed by windows procedure
  - Ends when Quit message is given



Windows Programming

- Win32 Program Structure

```
WinMain(…)                                  ← main function
{
        WNDCLASS …                          ← Define a new program
        CreateWindows (…)                   ← Create a window

        while(GetMessage (…))               ← Message Loop
        {

                DispatchMessage(…)          ← Message Handler
        }                                       (Windows Procedure)
}
```

Windows Programming

- Common Windows Messages

Message	**Sent When**
WM_CHAR	A character is input from the keyboard.
WM_COMMAND	The user selects an item from a menu, or a control sends a notificati on to its parent.
WM_CREATE	A window is created.
WM_DESTROY	A window is destroyed.
WM_LBUTTONDOWN	The left mouse button is pressed.
WM_LBUTTONUP	The left mouse button is released.
WM_MOUSEMOVE	The mouse pointer is moved.
WM_PAINT	A window needs repainting.
WM_QUIT	The application is about to terminate.
WM_SIZE	A window is resized.

Windows Programming

- Application Programming Interface(API)
  - A set of functions for controlling and using operating system
  - Mostly C functions

- Windows(Win32) API
  - Collection of C functions for making windows programming (library)
  - e.g. Functions for

    creating new windows,

    adding a button,

    adding a new menu,

    etc.

Windows Programming

- Create an empty Windows
  - 새 프로젝트 만들기 → **빈 프로젝트**
  - Project 속성
    - 고급 → 문자 집합 → **유니코드 문자 집합 사용**
    - 링커 → 시스템 → 하위 시스템 → **창(/SUBSYSTEM:WINDOWS)**
  - **Main.cpp** 추가

```
// Main.cpp
#include <windows.h>
LRESULT WINAPI WndProc (HWND, UINT, WPARAM, LPARAM);

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpszCmdLine, int nCmdShow)
{
    WNDCLASS wc;
    HWND hwnd;
    MSG msg;

    wc.style = 0;
    wc.lpfnWndProc = (WNDPROC) WndProc;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.hInstance = hInstance;
    wc.hIcon = LoadIcon (NULL, IDI_WINLOGO);
    wc.hCursor = LoadCursor (NULL, IDC_ARROW);
    wc.hbrBackground = (HBRUSH) (COLOR_WINDOW + 1);
    wc.lpszMenuName = NULL;
    wc.lpszClassName = L"MyWndClass";
// 이어서 계속
```

Windows Programming

- Create an empty Windows

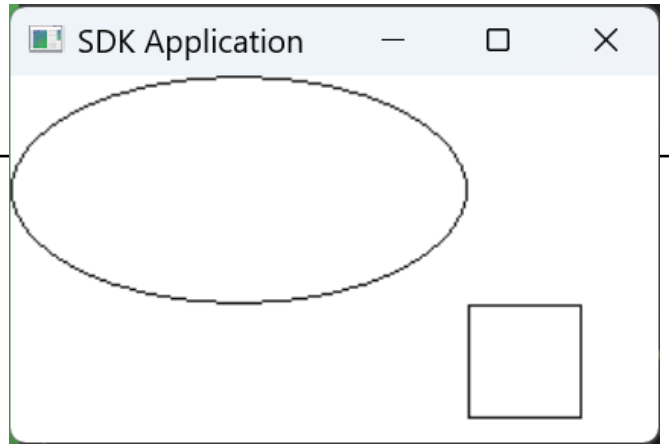
```
// Main.cpp
    RegisterClass (&wc);

    hwnd = CreateWindow (
        L"MyWndClass",      // WNDCLASS name
        L"SDK Application", // Window title
        WS_OVERLAPPEDWINDOW,// Window style
        CW_USEDEFAULT,      // Horizontal position
        CW_USEDEFAULT,      // Vertical position
        300,                // Initial width
        200,                // Initial height
        HWND_DESKTOP,       // Handle of parent window
        NULL,               // Menu handle
        hInstance,          // Application's instance handle
        NULL                // Window-creation data
    );

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

    while (GetMessage (&msg, NULL, 0, 0)) {
        TranslateMessage (&msg);
        DispatchMessage (&msg);
    }
    return (int)msg.wParam;
}
```

Windows Programming

- Create an empty Windows

```
// Main.cpp
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    PAINTSTRUCT ps;                              // a structure need to paint the client area of a window
    HDC hdc;                                     // a device context need for drawing on a window

    switch (message) {

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps);            // start of drawing objects
        Rectangle(hdc, 200, 100, 250, 150);      // draw a rectangle using a start and an end point
        Ellipse (hdc, 0, 0, 200, 100);           // draw an ellipse using a start and an end point
        EndPaint (hwnd, &ps);                    // end of drawing objects

        return 0;

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

- Run the program
  - Check the objects drawn on Windows

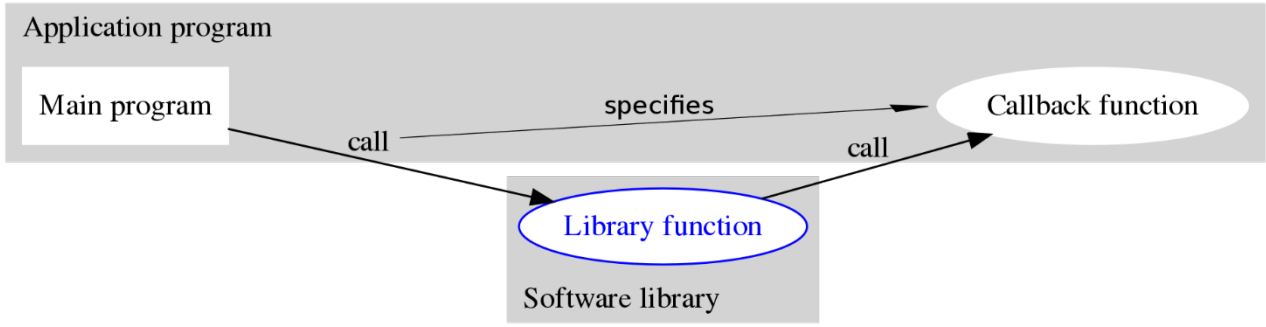


Windows Programming

- Callback function

```
// Main.cpp
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
```

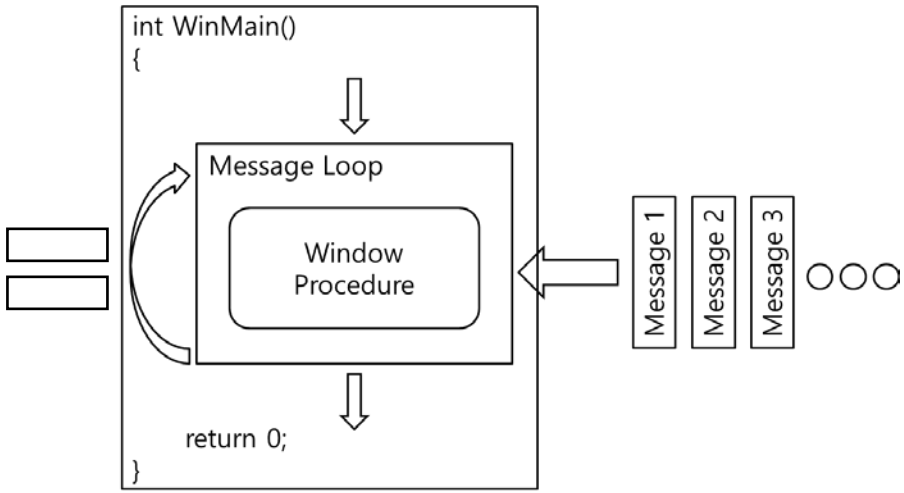
  - Callback function: a function passed as an argument to another piece of code; that code is expected to call back (execute) the callback function as part of its job (event)
    - Synchronous (blocking) callback: immediate execution before a function returns
    - Asynchronous (non-blocking) callback: deferred execution after a function returns
      - e.g. I/O operations, event handling
  - Often implemented in subroutines or function pointers



Windows Programming

- Code looks complex, but same structure

```
// Main.cpp
#include <windows.h>

LRESULT WINAPI WndProc (HWND, UINT, WPARAM, LPARAM);

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE
hPrevInstance, LPSTR lpszCmdLine, int nCmdShow)
{
      < existing codes >
      while (GetMessage (&msg, NULL, 0, 0)) {
              TranslateMessage (&msg);
              DispatchMessage (&msg);
      }
      < existing codes >
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM
wParam, LPARAM lParam)
{
    < existing codes >

    switch (message) {
          case WM_PAINT:
            < existing codes >

          case WM_DESTROY:
            < existing codes >
    }

    return DefWindowProc (hwnd, message, wParam, lParam);
}
```

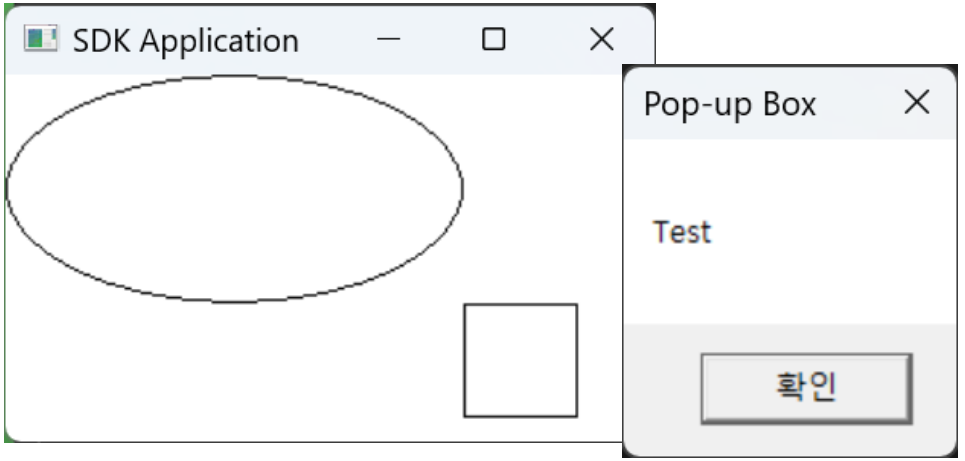


Windows Programming

- Add an event handler for LMB

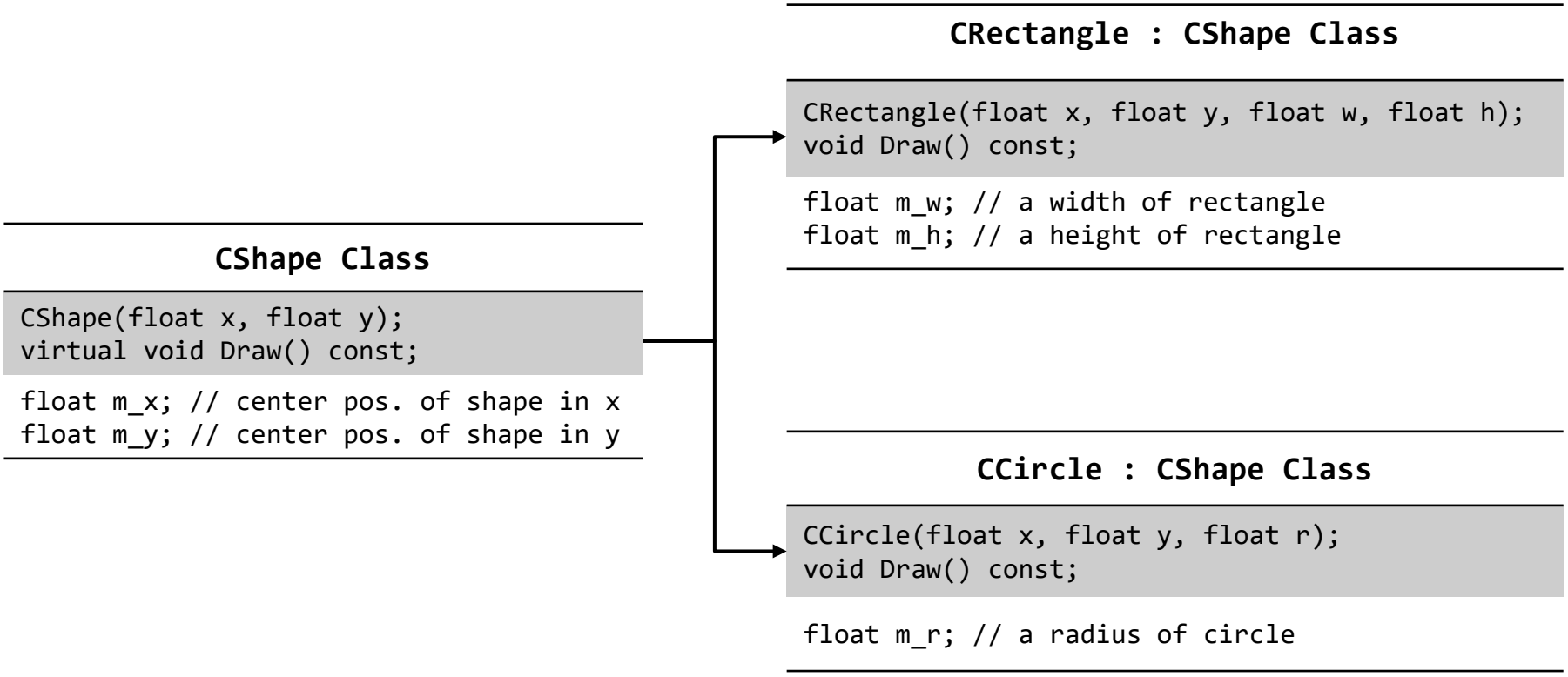
```
// Main.cpp
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
   < exiting codes >

   switch (message) {
       < exiting codes >

       case WM_LBUTTONDOWN:
             MessageBox(hwnd, L"Test", L"Pop-up Box", MB_OK);
             break;

   }

   return DefWindowProc (hwnd, message, wParam, lParam);
}
```

- Run the program
  - Check the pop-up box when LMB is clicked



Windows Programming

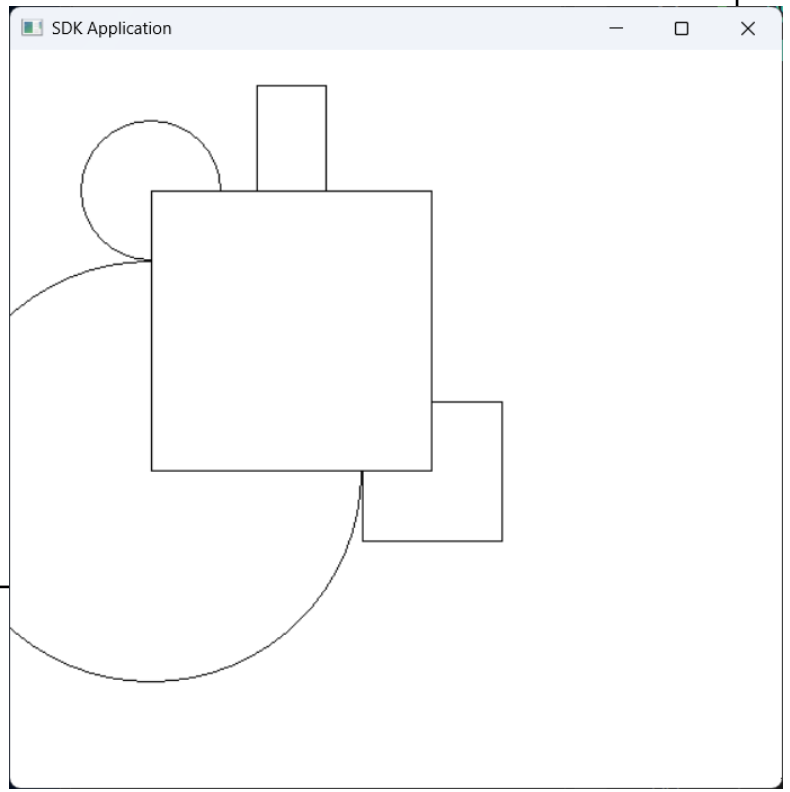
- Creating multiple shape objects
  - Create child objects **inherited** from the parent object as follows:



Windows Programming

- Draw shape objects on Windows
  - Use a single array for the shape objects and polymorphism of C++
    - e.g. Pointing a child object using a pointer of parent class

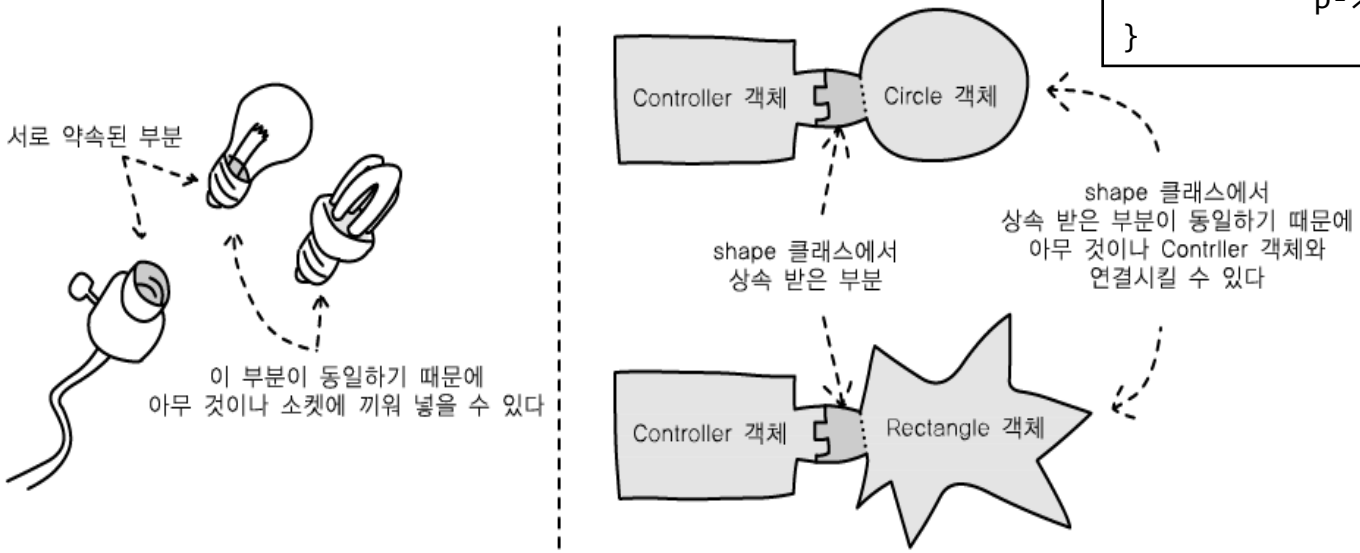
```
// main.cpp
CShape* shapes[5] = {NULL};
shapes[0] = new CCircle(100, 100, 50);
shapes[1] = new CRectangle(300, 300, 100, 100);
shapes[2] = new CRectangle(200, 100, 50, 150);
shapes[3] = new CCircle(100, 300, 150);
shapes[4] = new CRectangle(200, 200, 300, 300);

for (int i = 0; i < 5; ++i)
      shapes[i]->Draw();

for (int i = 0; i < 5; ++i)
{
      delete shapes[i];
      shapes[i] = NULL;
}
```



Windows Programming

- Polymorphism in C++
  - Object type에 관계 없이 같은 method로 다룰 수 있는 능력
    - Member function이 compile시에 결정되지 않고, runtime시 결정됨
      → member functio의 동적인 선택(dynamic binding)
    - Circle이나 Rectangle object들을 Shape object처럼 다룰 수 있음
  - Object 간의 object 간의 연결을 유연하게 해 줌
    - 아래와 같이 새로운 함수를 만들 때 유용

```
// Move the shape object to an origin
void CShape::MoveToOrigin(CShape *p) {
            p->Move(0, 0);
            p->Draw();
}
```

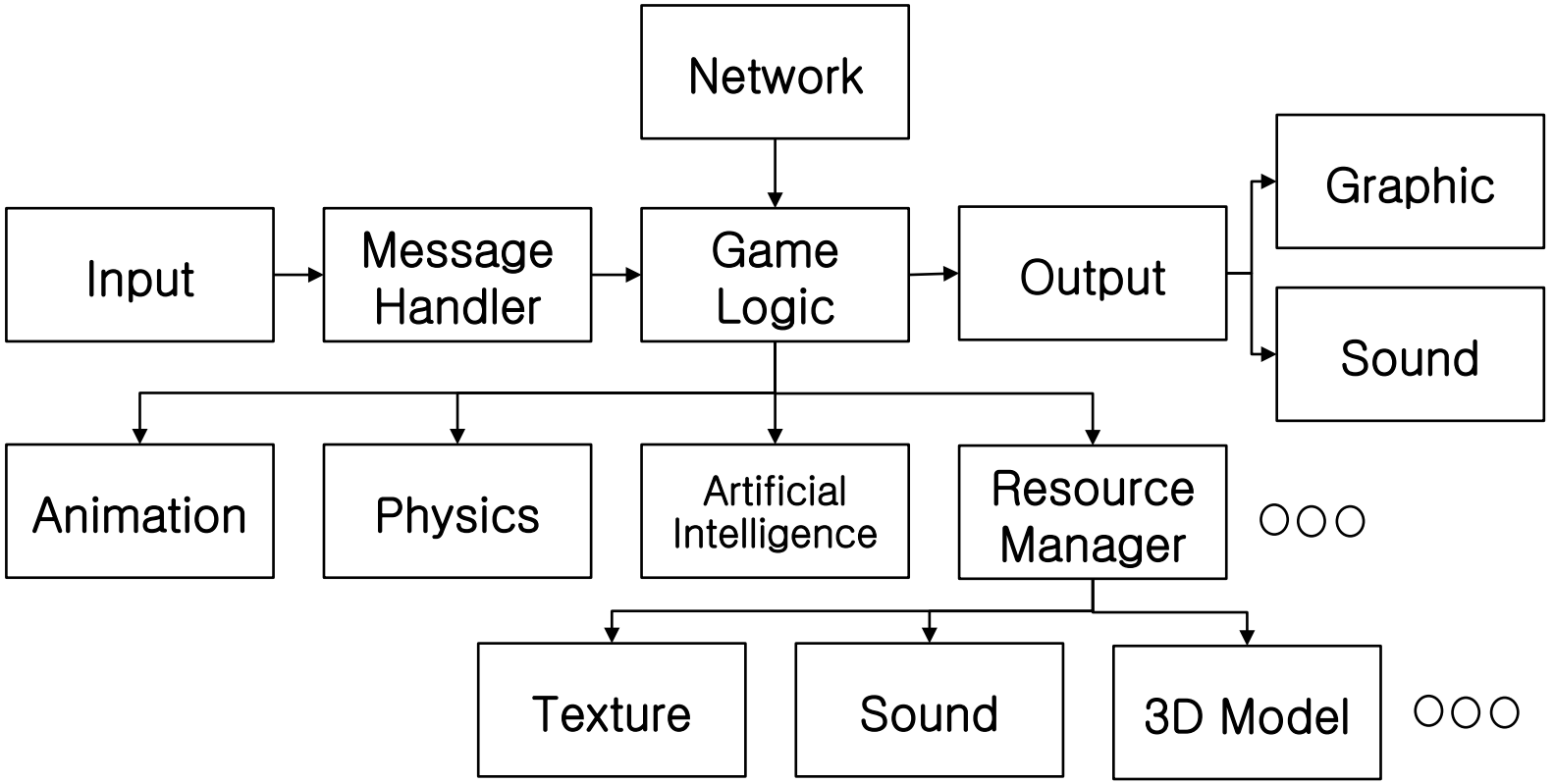


Tutorials

- Framework Setup
- Direct3D Setup

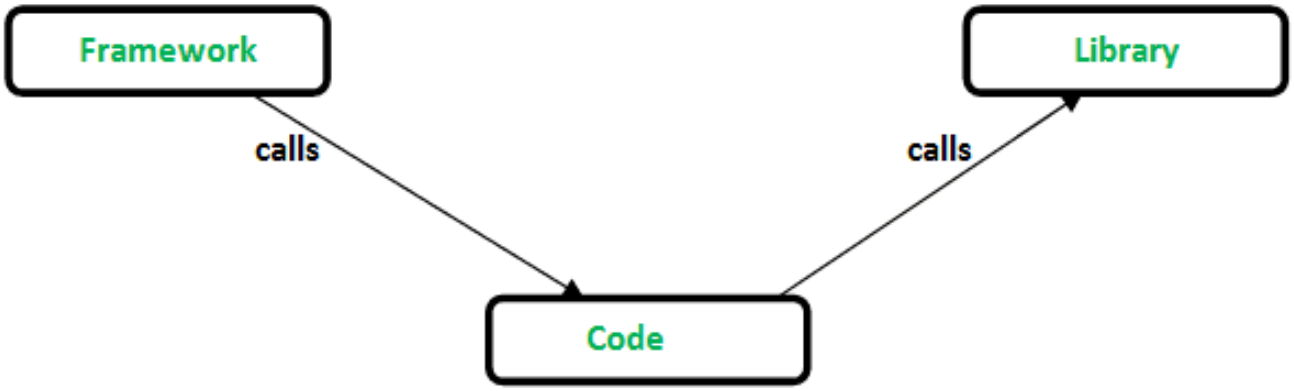
1-1 Framework Setup

- Typical game application structure



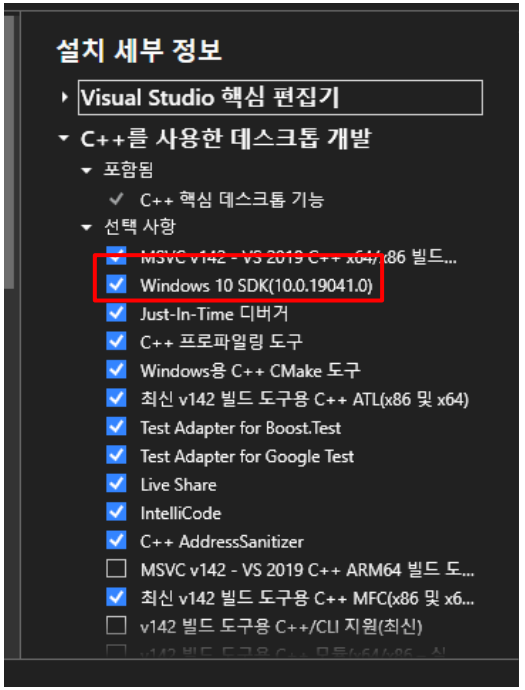
1-1 Framework Setup

- Creating a Framework
  - Code framework
    - Handles the basic Windows functionality and provides an easy way to **expand** the code in an organized and readable manner
    - Keep the framework as **thin** as possible



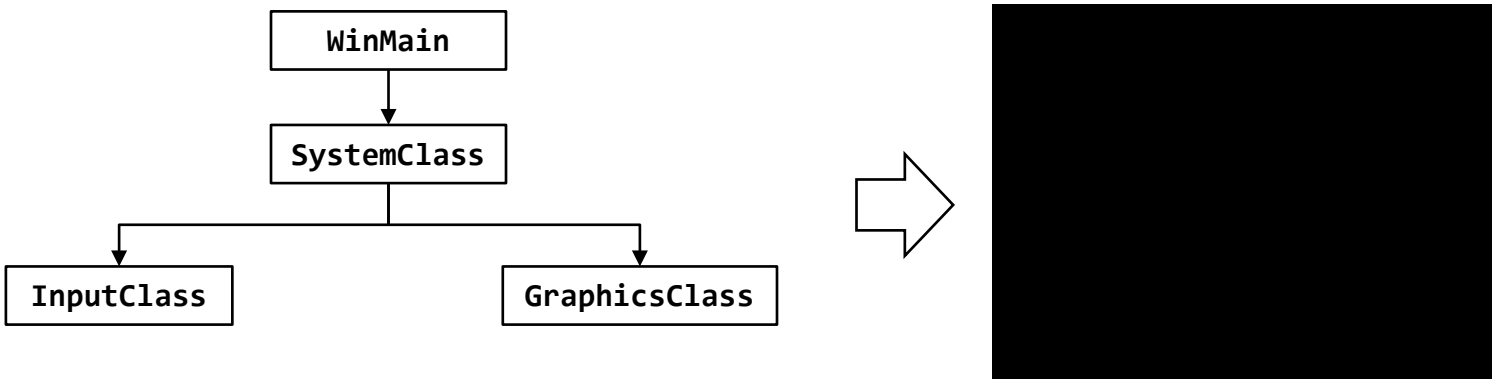
1-1 Framework Setup

- Setting up DirectX 11 with Visual Studio 2022
  - DirectX is now a part of Windows SDK (installed with VS)
    - Install DirectX SDK (June 2010) if old DirectX functions are used
  - Create a Win32 project: 새 프로젝트 만들기 → 빈 프로젝트
    - 속성 → 고급 → 문자 집합 → **유니코드 문자 집합 사용**
    - 속성 → 링커 → 시스템 → 하위 시스템: **창(/SUBSYSTEM:WINDOWS)**
    - 속성 → C/C++ → 고급 → 특정 경고 사용 안 함: **4005**
      - 4005: DirectX macro redefinitions
  - Build with **x86** and **Debug**/**Release** modes



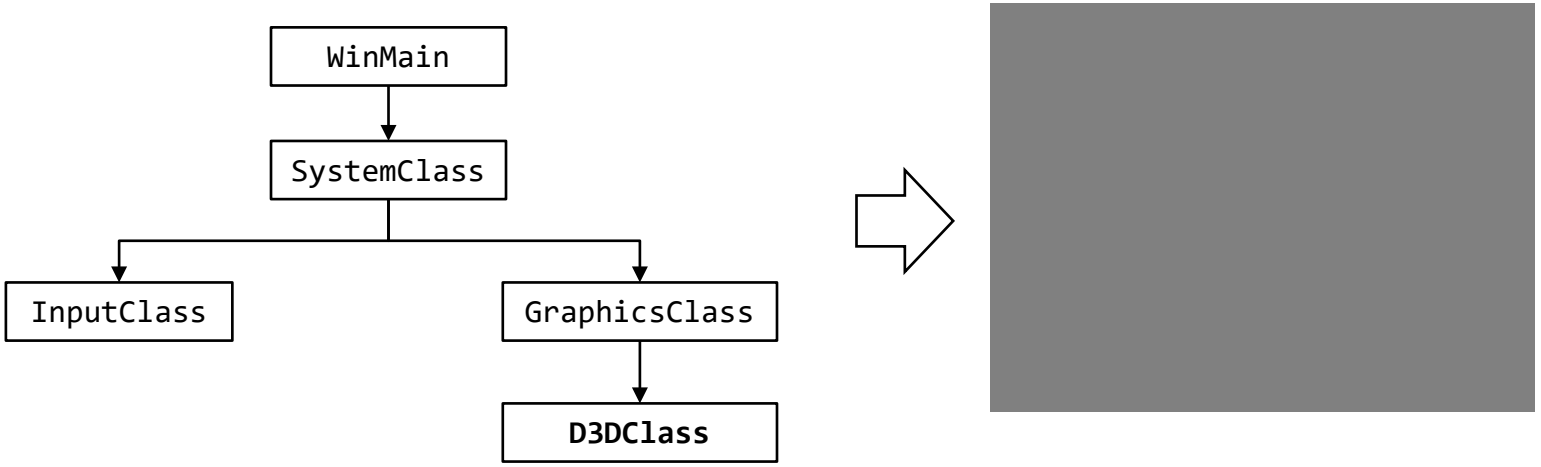
1-1 Framework Setup

- Creating a framework structure
  - WinMain: handle the entry point of the application
  - SystemClass: encapsulate the entire application
  - InputClass: handle user inputs
  - GraphicsClass: initialize and shut down D3DClass object



1-2 Direct3D Setup

- Initalizing Direct3D graphics
  - D3DClass: initialize the DirectX graphics
    - D3DClass::InitializeGraphicsHW()
    - D3DClass::InitializeSwapBuffer()
    - D3DClass::InitializeDepthStencilBuffer()
    - D3DClass::InitializeRasterizer()
    - D3DClass::InitializeViewport()



References

- Wikipedia
  - www.wikipedia.org
- Introduction to DirectX 11
  - www.3dgep.com/introduction-to-directx-11
- Raster Tek
  - www.ratertek.com
- Braynzar Soft
  - www.braynzarsoft.net)
- CS 445: Introduction to Computer Graphics *[Aaron Bloomield]*
  - www.cs.virginia.edu/~asb/teaching/cs445-fall06

Q & A