Image Processing
- Color Conversion, MFC

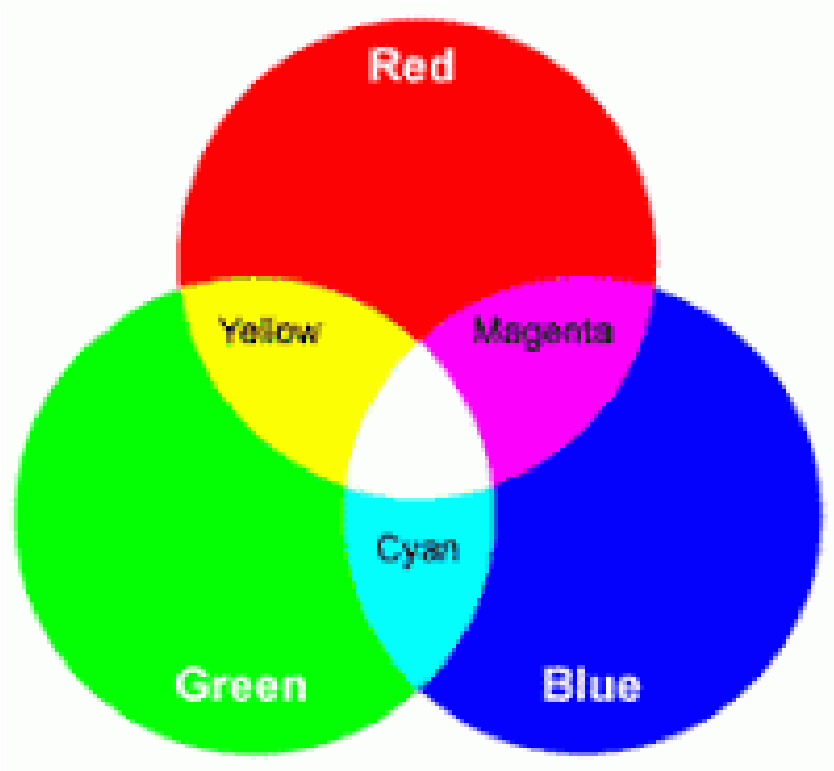
Contents

- RGB & YUV
- YUV ↔ RGB Conversion
- Access Pixel Data
- Sum & Subtract Image
- Dialog Based MFC
- Programming

RGB & YUV

RGB & YUV

- **RGB (Red, Green, Blue)**
  - 빛의 3원색인 빨강, 초록, 파랑의 합성어
  - 3가지의 색을 조합하여 다양한 색을 표현함



RGB & YUV

- **RGB (Red, Green, Blue)**
  - 디지털 공간에서의 RGB를 이용한 색 표현
    - R, G, B에 Bit를 할당, 각각의 픽셀이 가지고 있는 색상 성분 값을 저장

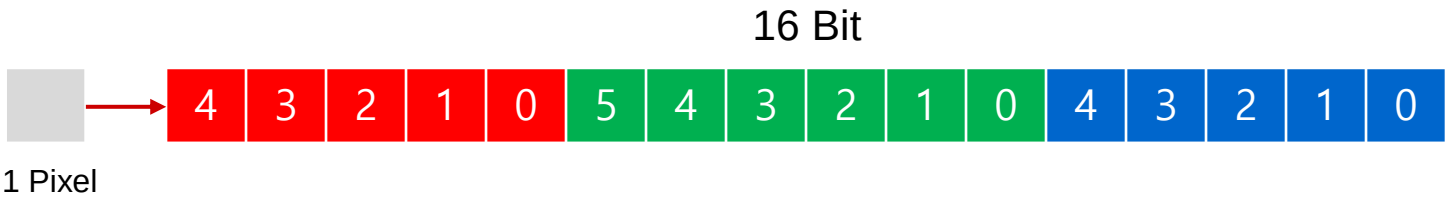
(Height, Width) = (99,254)	
R	189
G	203
B	43

RGB & YUV

RGB (Red, Green, Blue)

- 디지털 공간에서의 RGB를 이용한 색 표현
  - 색에 할당되는 비트에 따라 표현 가능한 색상의 가짓수가 변화

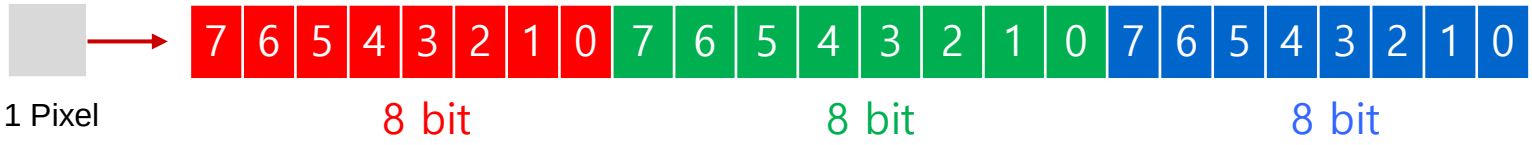
- RGB565 (High Color)
  - Red와 Blue에 5bit, Green에 6Bit를 할당
  - 총 16Bit의 저장공간을 이용하여 $2^{16} = 65{,}535$ 가지의 색상을 표현
  - 가시광선 영역에서 Green이 차지하는 비중이 높아 1Bit를 더 할당



RGB & YUV

KWANGWOON UNIVERSITY

- **RGB (Red, Green, Blue)**
  - RGB888 (True Color)
    - 총 24Bit의 저장공간을 이용, $2^{24} = 16{,}777{,}216$ 가지의 색상을 표현
    - SRGB 라고도 하며, 국제 전기 표준 회의의 표준으로 등재
    - 8 Bit의 투명 값(Alpha)을 추가하여 32Bit Color로 사용하기도 함

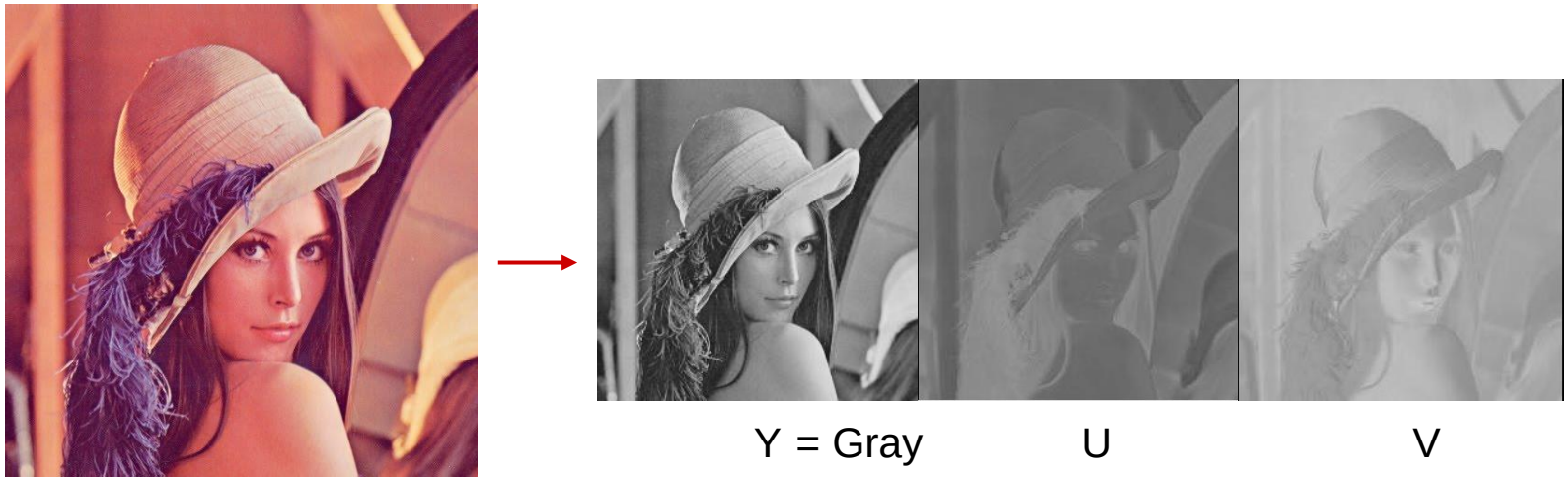


RGB & YUV

YUV (YCbCr)

- 빛의 밝기를 나타내는 휘도(Y, Luminance)와 색차 신호(U & V, Chrominance)를 이용해 색을 표현
- 흑색 인프라 환경에서 컬러를 추가적으로 표현하기 위해 고안됨
- 사람의 눈은 밝기에 민감하므로 U, V 성분의 데이터를 줄여 RGB에 비해 적은 데이터로도 비슷한 화질을 표현할 수 있음
- Sub-Sampling 방식에 따라 여러 Format으로 나뉨
  - ex) YUV444, YUV422, YUV420

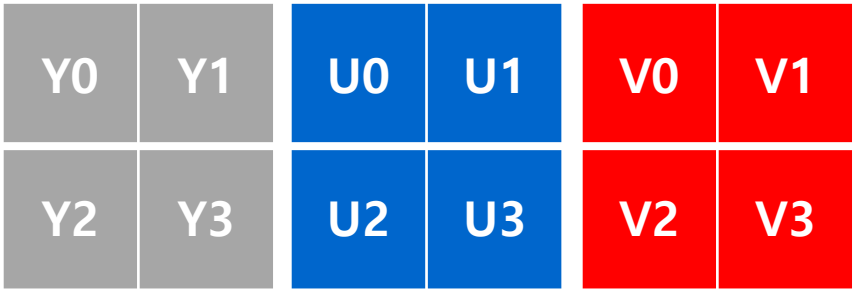
- **영상의 흑백 변환 : YUV에서 Y 성분의 값만을 추출함**



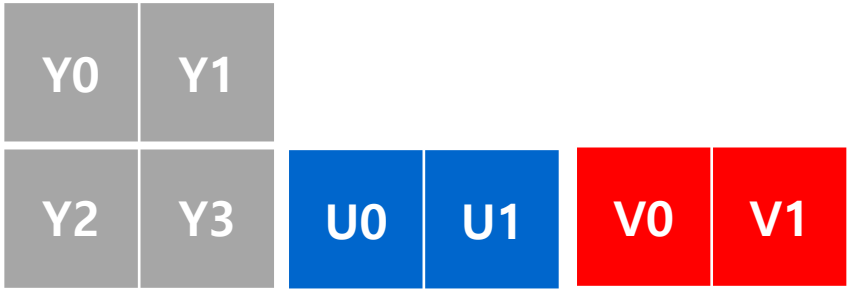
YUV444

RGB & YUV

- YUV (YCbCr)

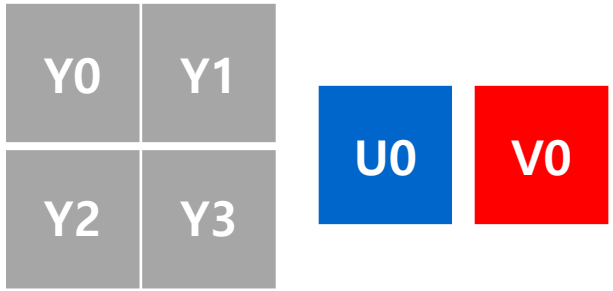


YUV444



YUV422

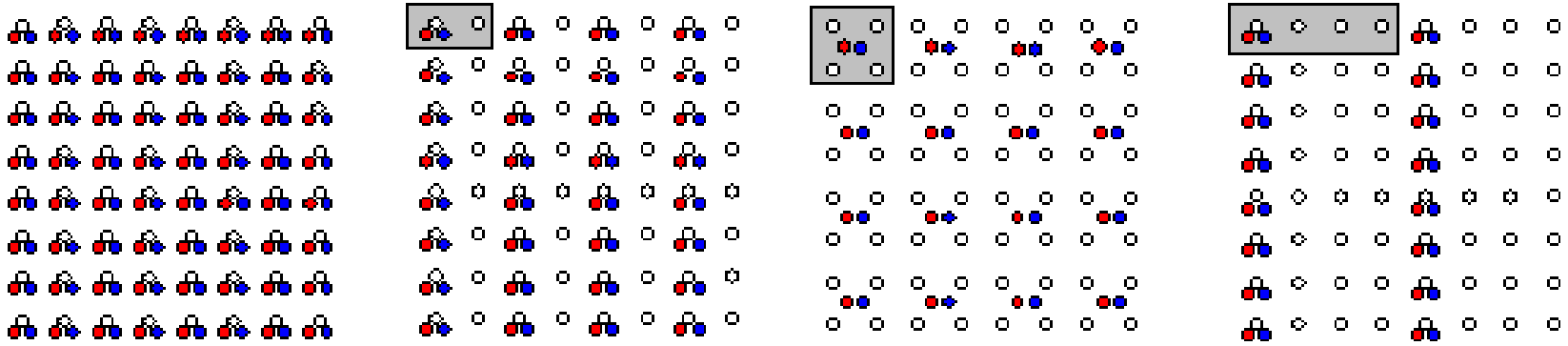
YUV411



YUV420

RGB & YUV

- YUV (YCbCr)



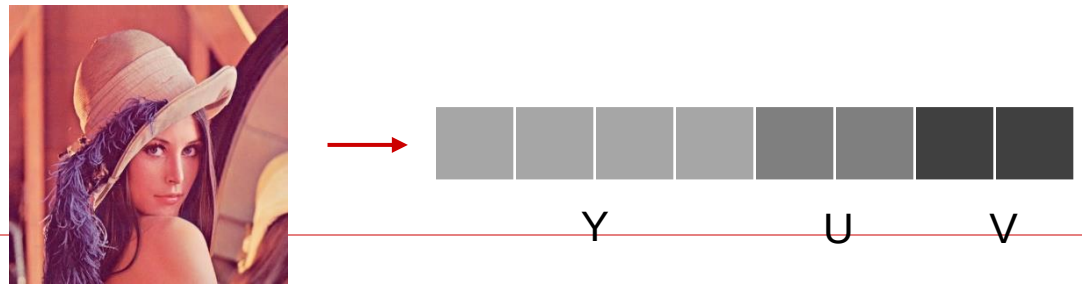
- Y-Signal
- C_R, C_B Signale

RGB & YUV

File Format

- YUV - *.yuv (Raw Data)
  - 확장자 : .yuv or .raw
  - **영상의 픽셀 값**만으로 파일이 구성되어 있음
  - 영상을 분석하기 위해서는 영상의 정보들을 반드시 알아야 함
    - 영상의 폭, 높이, Color Format (YUV444 or YUV420 or …)
  - Bitstream의 구성
    - YUV444

    - YUV422



RGB & YUV

YUV (YCbCr)

- Y 성분(Gray) 만 존재하는 영상(RAW File) 로드 및 데이터 저장

```
#include "opencv2₩opencv.hpp"

#define HEIGHT 256
#define WIDTH 256

using namespace cv;
using namespace std;

void main()
{
    unsigned char **Y;

    FILE *fp_in, *fp_out;

    fopen_s(&fp_in, "lena256.raw", "rb");

    Y = (unsigned char**)malloc(sizeof(unsigned char*) * HEIGHT);

    for (int h = 0; h < HEIGHT; h++)
    {
        Y[h] = (unsigned char*)malloc(sizeof(unsigned char) * WIDTH);

        fread(Y[h], sizeof(unsigned char), WIDTH, fp_in);
    }


}
```

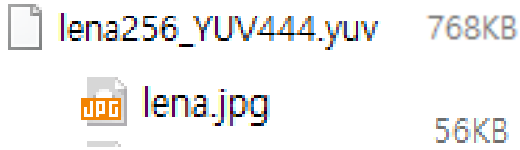
2차원 배열 Memory Allocation

Bitstream을 지정한 buffer에 저장

RGB & YUV

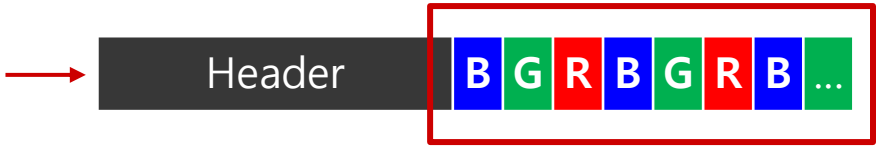
File Format

- RGB - **JPG**
  - 확장자 : .jpg or jpeg
  - 원본 영상(Raw Data)을 **손실 압축**
    - Image Compression (chapter 10)
  - 파일의 Header에 영상의 정보 및 압축 방식이 저장되어 있음



- 파일의 구성

lena.jpg



YUV ↔ RGB Conversion

YUV ↔ RGB Conversion

KWANGWOON UNIVERSITY

RGB → YUV 변환

- ITU-R Recommendation BT. 601
- $Y = 0.299R + 0.587G + 0.114B$
- $U = 0.564(B - Y) = -0.169R - 0.331G + 0.499B$
- $V = 0.713(R - Y) = 0.499R - 0.418G - 0.081B$

YUV -> RGB 변환

- $R = Y + 1.402V$
- $G = Y - 0.344U - 0.714V$
- $B = Y + 1.772U$

YUV ↔ RGB Conversion

KWANGWOON UNIVERSITY

RGB → YCbCr 변환

- ITU-R Recommendation BT. 601
- $Y = 0.299R + 0.587G + 0.114B$
- $U = 0.564(B - Y) = -0.169R - 0.331G + 0.499B + \mathbf{128}$
- $V = 0.713(R - Y) = 0.499R - 0.418G - 0.081B + \mathbf{128}$

YCbCr -> RGB 변환

- $R = Y + 1.402(V - \mathbf{128})$
- $G = Y - 0.344(U - \mathbf{128}) - 0.714(V - \mathbf{128})$
- $B = Y + 1.772(U - \mathbf{128})$

Access Pixel Data

Access Pixel Data

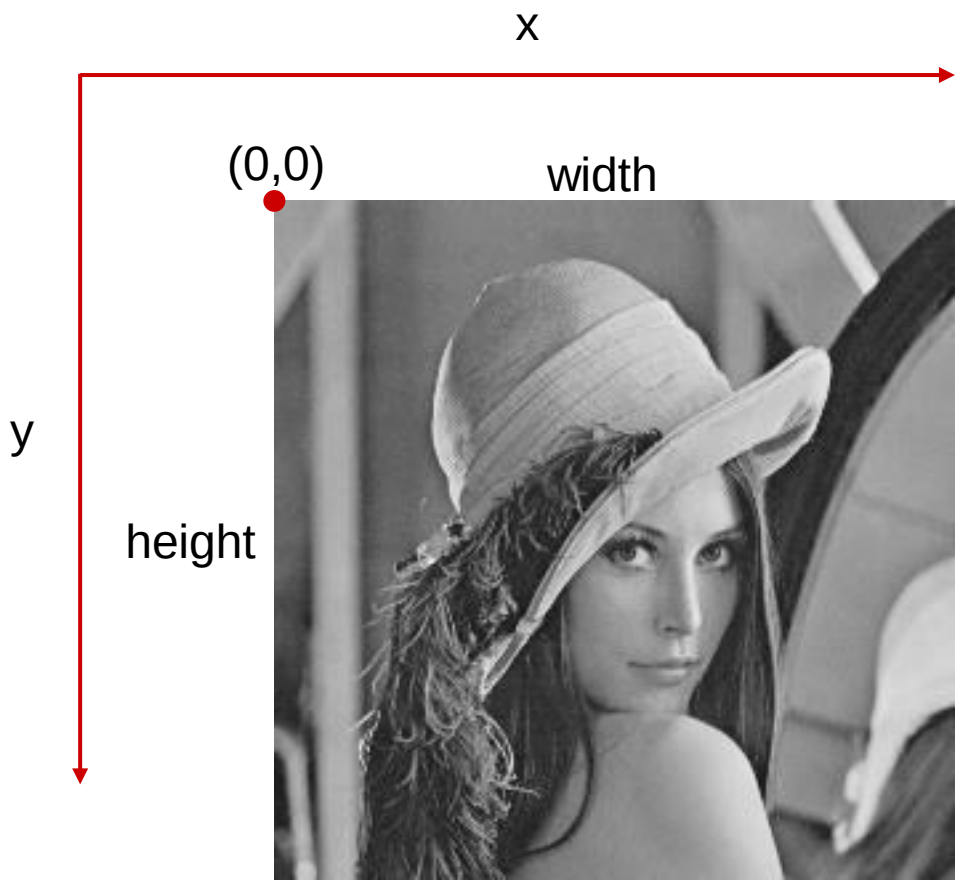
KWANGWOON UNIVERSITY

- **In YUV(YCbCr)**
  - FILE* 구조체를 통하여 영상 파일을 Bitstream의 형태로 읽어들임
  - 배열에 데이터 저장 후 인덱스를 통하여 접근

- **In OpenCV**
  - Mat
    - 메서드를 통한 접근 : Mat.at<_Tp>(int, int) method
      - Template method
      - Point, Vec, uchar, int, double 등의 형태로 저장되어 있는 픽셀값을 불러온다.
    - 직접 접근 : char* Mat.data
      - Pixel Data가 실제로 저장되어 있는 1D Array
      - 영상의 왼쪽 최상단의 Pixel의 Data부터 차례대로 저장되어 있음
      - channel (RGB) 의 경우 B, G, R의 순서로 Data가 저장되어 있음
      - ex) Gray 영상에서 Pixel Data에 접근
        - Img**(y, x)** 에 있는 데이터에 접근
        - Img.data**(y * width + x)**

Access Pixel Data

- 영상 좌표계



Access Pixel Data

KWANGWOON UNIVERSITY

In YUV(YCbCr)

- 예제 코드
  - (200, 100) 에 있는 Pixel 값에 접근

```
unsigned char **Y;

Y = (unsigned char**)malloc(sizeof(unsigned char*) * HEIGHT);

for (int h = 0; h < HEIGHT; h++)
{
    Y[h] = (unsigned char*)malloc(sizeof(unsigned char) * WIDTH);

unsigned char pixel = Y[100][200];
```

In OpenCV

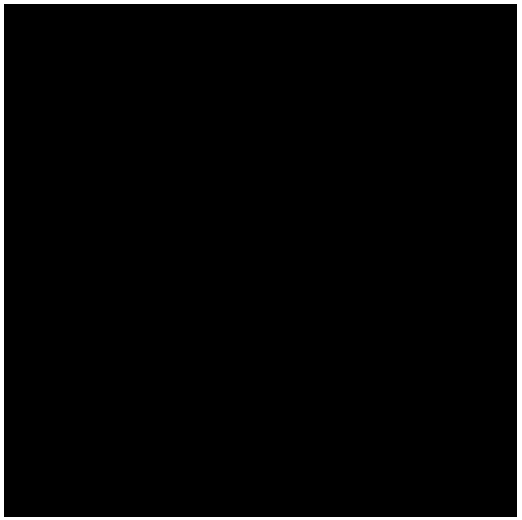
- 예제 코드
  - (200, 100) 에 있는 Pixel 값에 접근

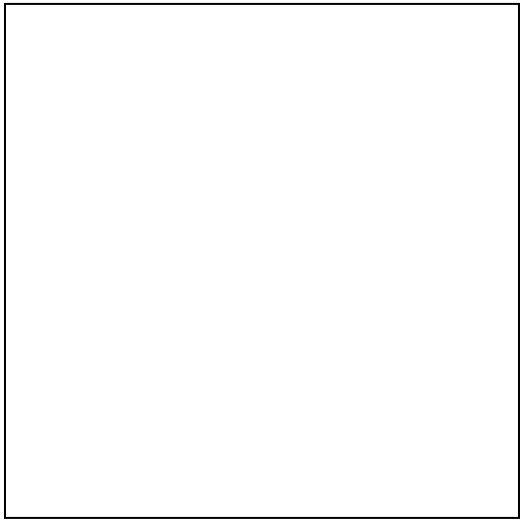
```
Mat Inp = imread("lena.jpg");

uchar pixel = Inp.at<uchar>(100, 200);
```

Access Pixel Data

- **Programming #1**
  - 256 x 256 크기의 영상 만들기
    - 검은색 영상
    - 흰색 영상

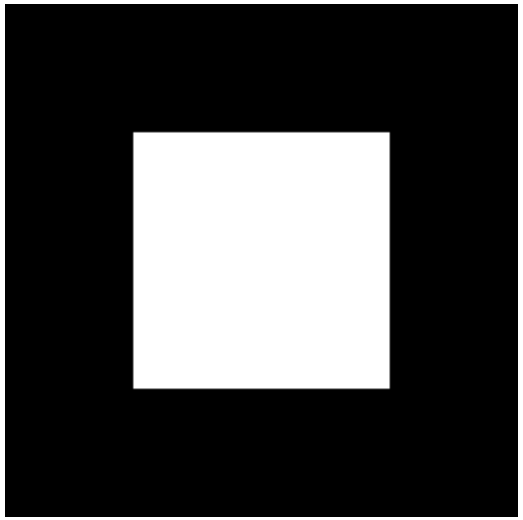


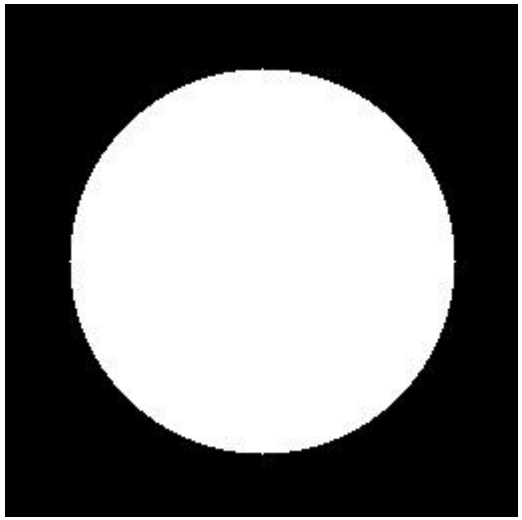


Access Pixel Data

KWANGWOON UNIVERSITY

- **Programming #2**
  - 256 x 256 크기의 영상 만들기
    - 검은색 영상에 하얀색 정사각형 그리기
    - 검은색 영상에 하얀색 원 그리기





Sum & Subtract Image

Sum & Subtract Image

Sum & Subtract Image

- 두 이미지의 같은 픽셀 주소에 있는 값을 더하거나 뺌
- 픽셀 값에서 고정된 값을 더하거나 빼어 영상의 밝기를 조절할 수 있음

Y(Gray Scale) Image

Dark (-70)

Bright (+70)

Sum & Subtract Image

KWANGWOON UNIVERSITY

Sum & Subtract Image

- 두 이미지의 같은 픽셀 주소에 있는 값을 더하거나 뺌
- 픽셀 값에서 고정된 값을 더하거나 빼어 영상의 밝기를 조절할 수 있음

Y(Gray Scale) Image

Reversed Image

Dialog Based MFC

MFC

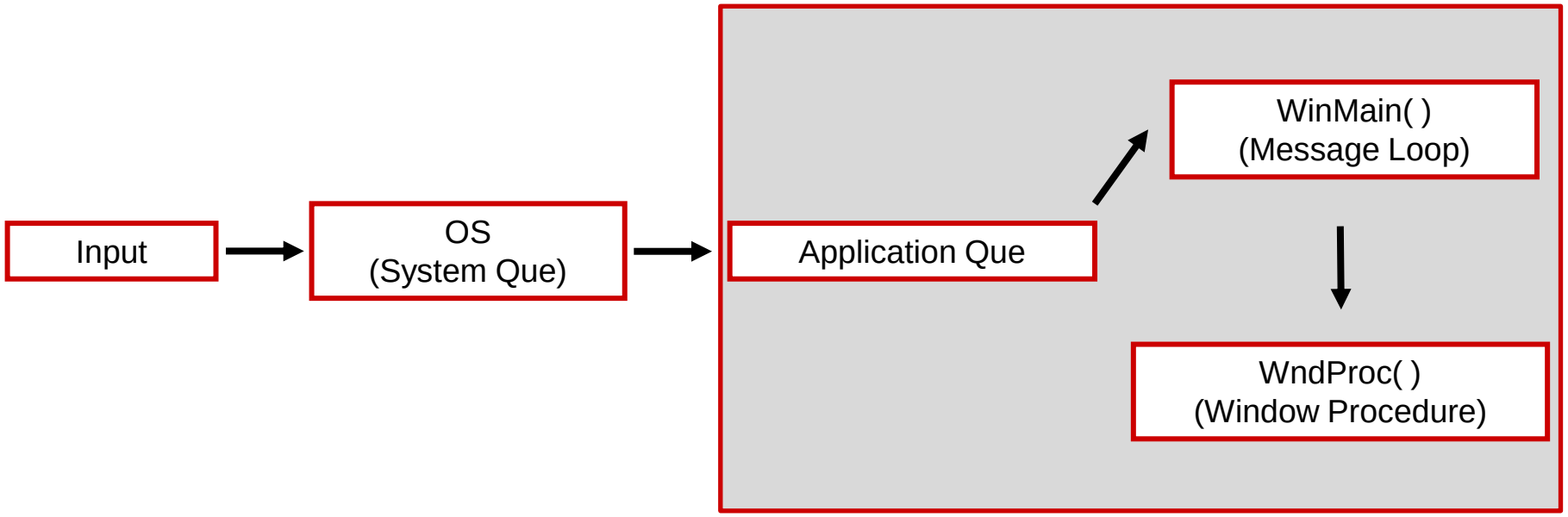
- **MFC (Microsoft Foundation Classes)**
  - Win32 API를 각 기능 별로 클래스화 한 것
  - 창, 메뉴, 대화상자 등의 관리와 기본 입출력 작업 등에 필요한 코드 제공
  - 간단한 데스크톱 응용 프로그램부터 복잡한 사용자 인터페이스 개발에 유용

  - Handle
    - 어떤 대상에 붙여진 Label과 같은 것으로, 대상을 식별하는데 주로 사용됨
    - 사용자가 조작할 타겟(윈도우, 컨트롤 등)을 관리할 수 있음
    - 운영체제가 발급, 사용자는 할당된 핸들을 사용

MFC

MFC (Microsoft Foundation Classes)

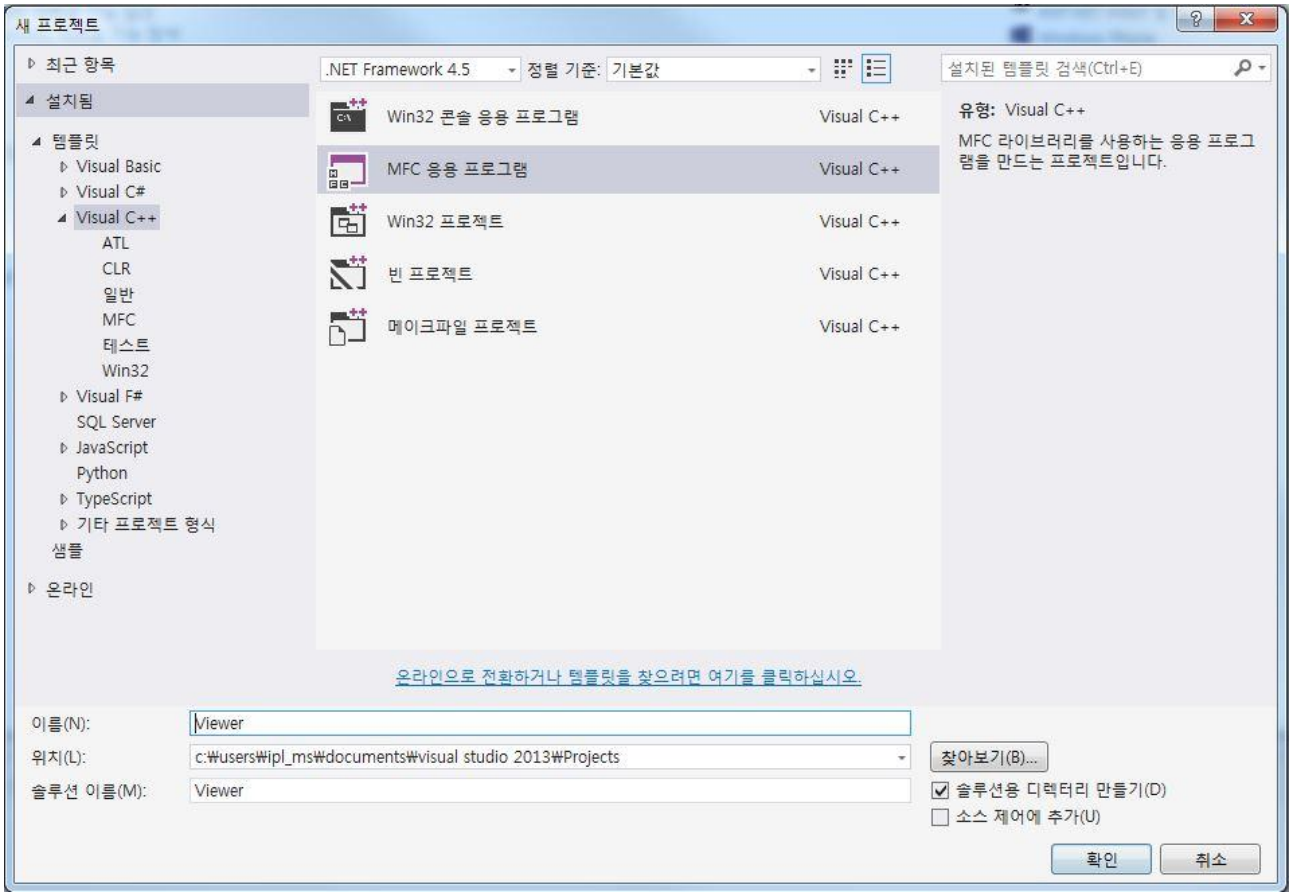
- Message Loop
  - 입력으로 인한 임의의 이벤트가 발생했을 때 즉각적으로 반응하여 처리
  - 이 때, 발생된 이벤트를 Message Loop 에서 감지, 해당 메시지를 윈도우 Procedure 함수로 전달



편집기 프로그램 (응용 프로그램)

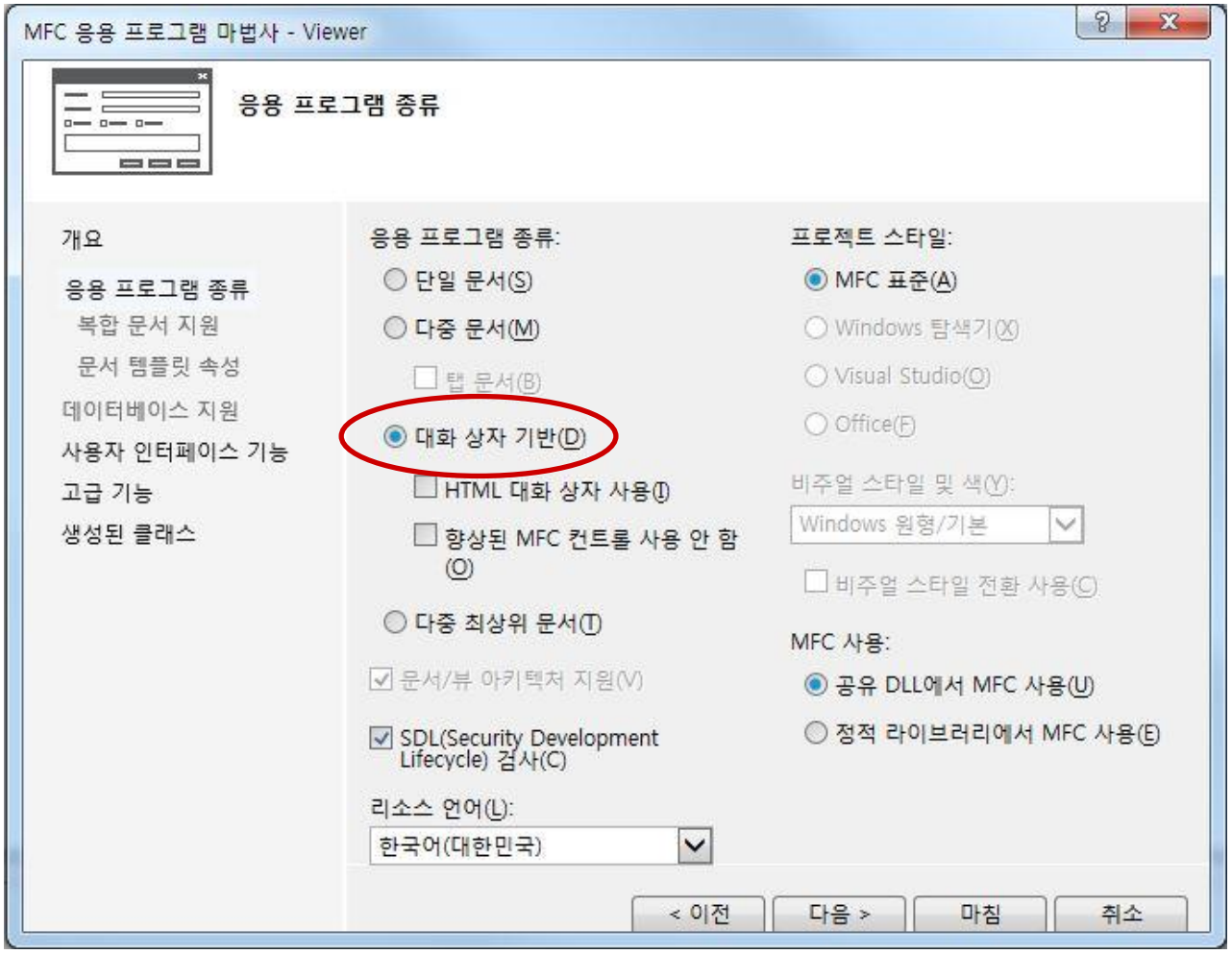
Dialog Based MFC

Dialog Based MFC



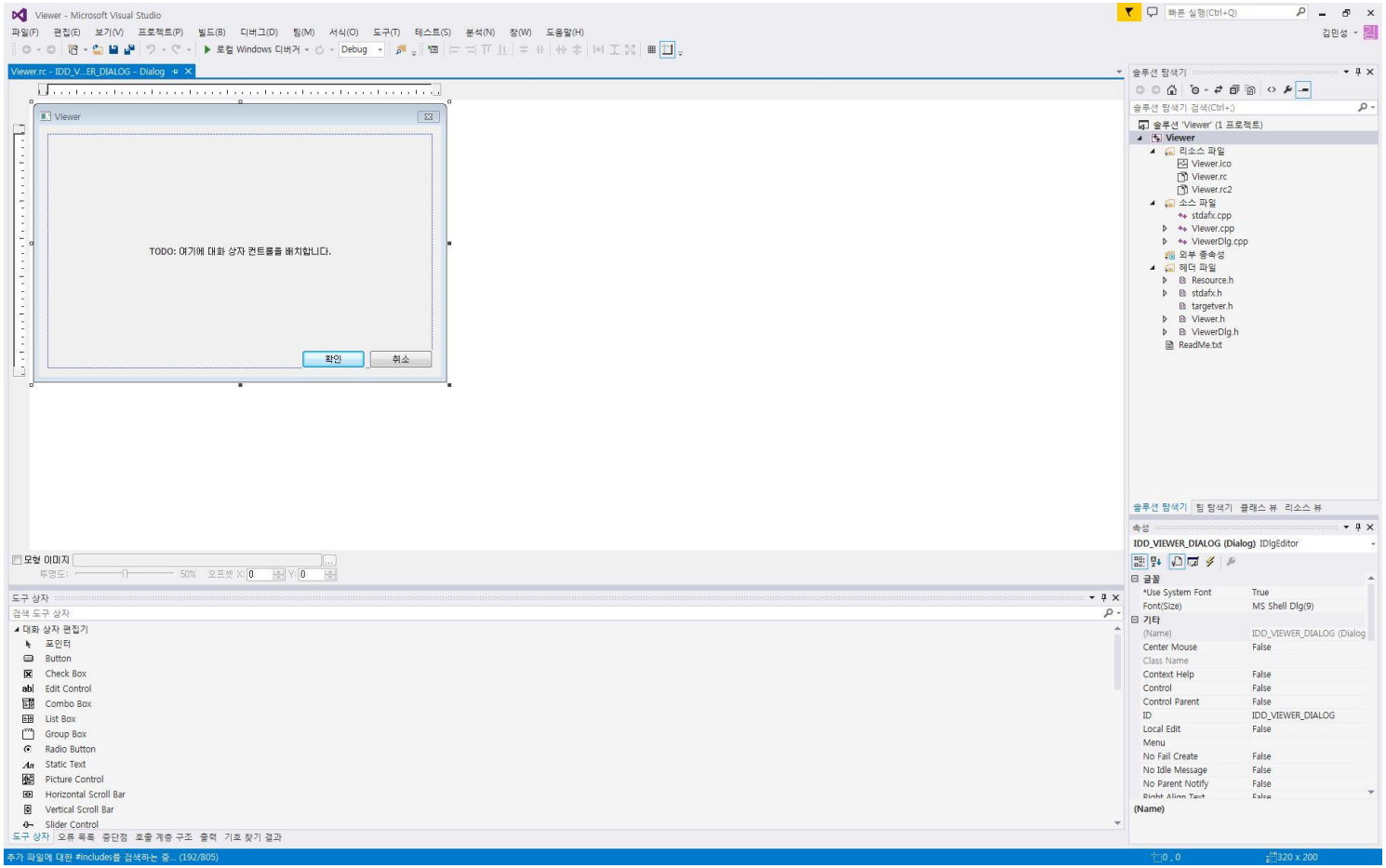
Dialog Based MFC

Dialog Based MFC



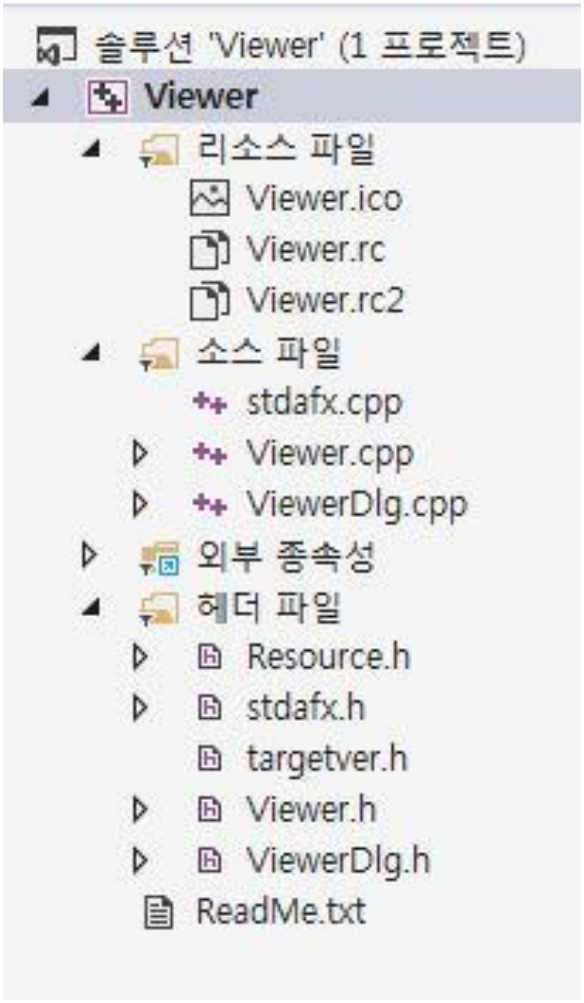
Dialog Based MFC

Dialog Based MFC



Dialog Based MFC

Dialog Based MFC

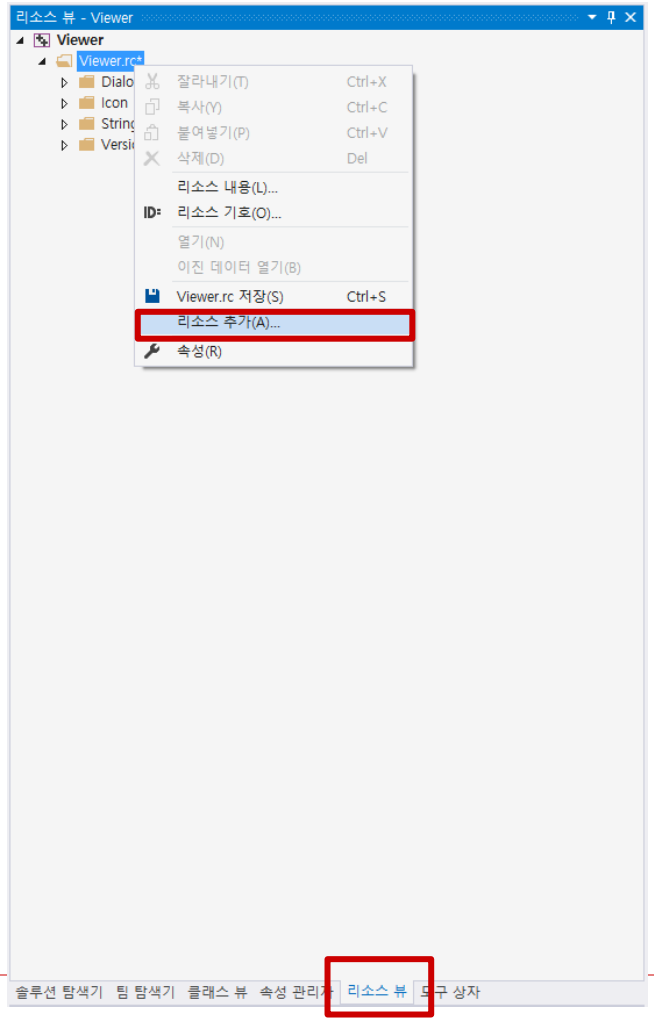


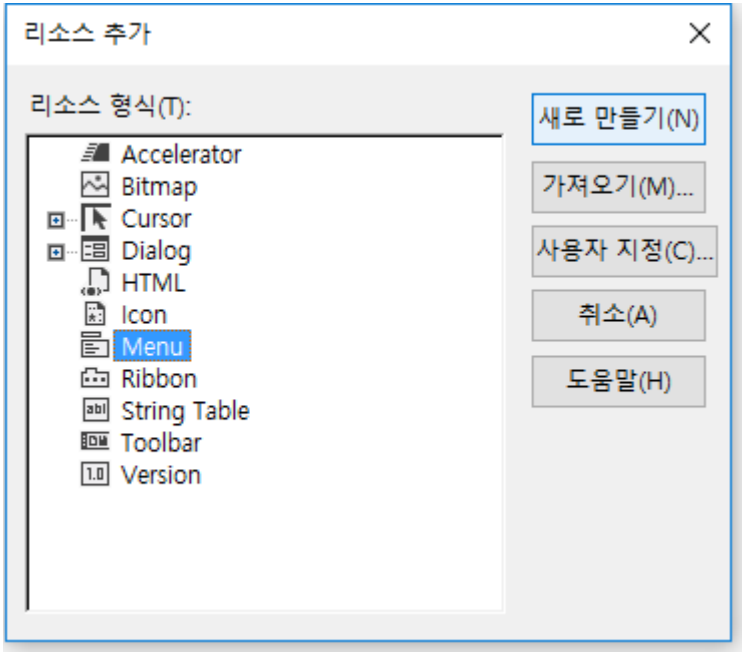
- Viewer.cpp / Viewer.hpp
  - 프로그램 전체를 관리하는 CViewerApp Class가 선언되고 Viewer 프로그램의 바깥 틀에 해당
  - 특별한 경우가 아니라면 별도로 수정하지 않음
- ViewerDlg.cpp / ViewerDlg.hpp
  - 대화 상자 클래스인 CViewerDlg Class가 선언되고 정의됨
  - GUI 화면을 담는 Resource 파일과 연동, 이벤트 발생 시 멤버 함수로 하여금 처리하게 함
- Viewer.rc
  - 대화 상자의 GUI 등 프로젝트의 리소스에 대한 정의가 들어있음
- Resource.h
  - 리소스를 프로젝트의 다른 Header / Source 파일들에서 사용할 수 있게 하는 리소스 ID가 정의
  - 대화상자 편집 시 자동으로 수정되므로 직접 변경할 필요 없음

Dialog Based MFC

Dialog Based MFC

- 메뉴 바 추가

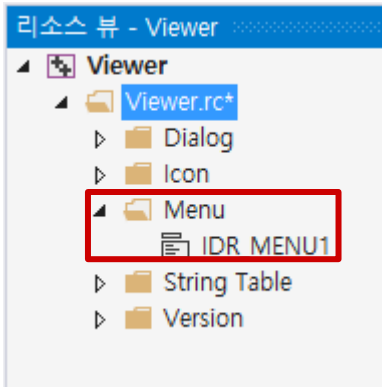


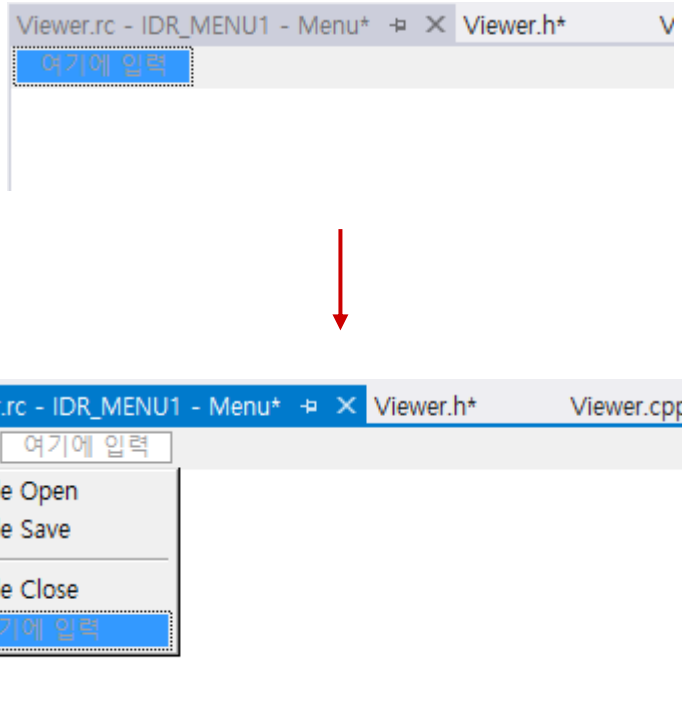


Dialog Based MFC

Dialog Based MFC

- 메뉴 바 추가

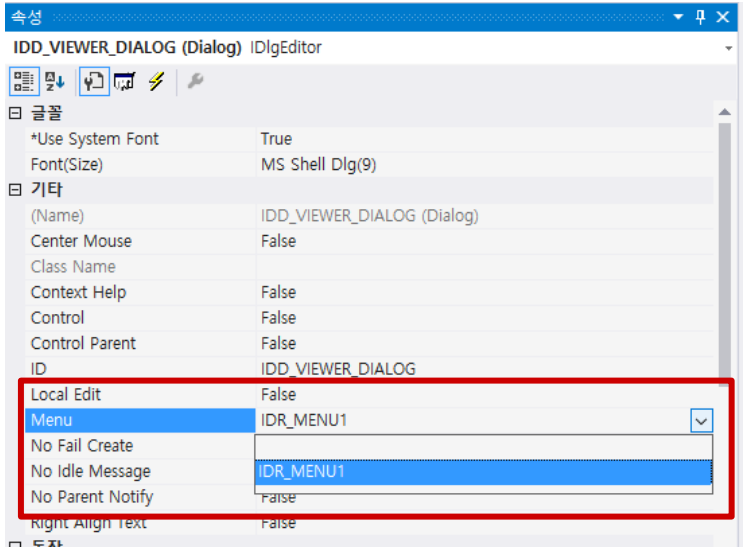


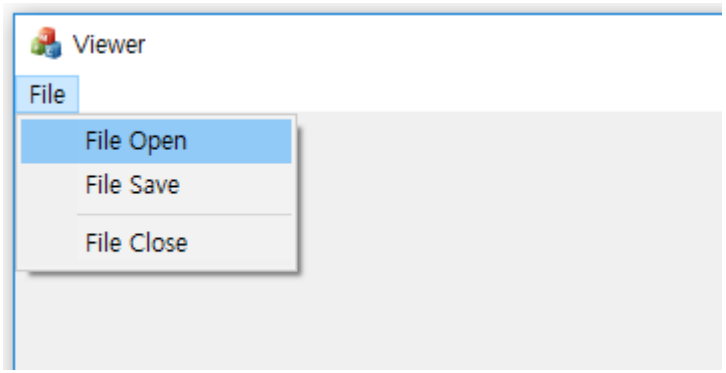


Dialog Based MFC

Dialog Based MFC

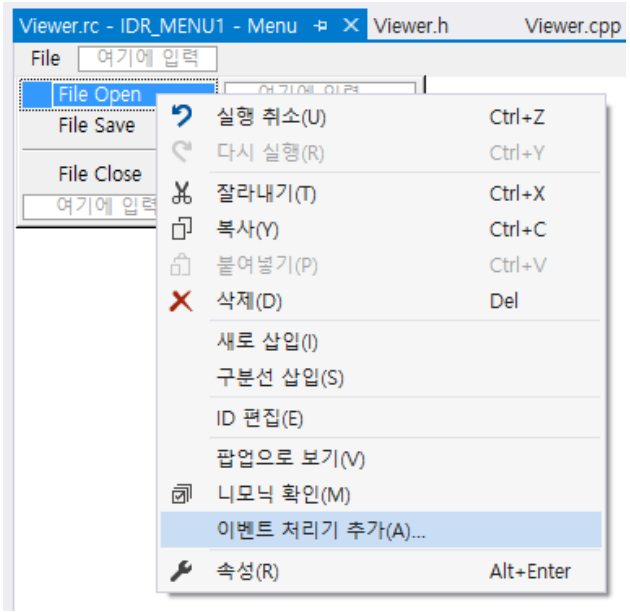
- 메뉴 바 추가

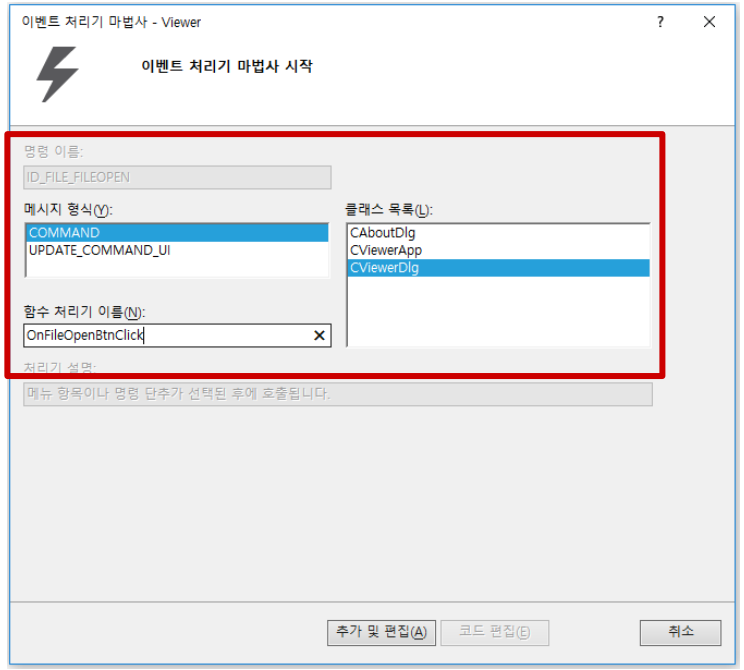




Dialog Based MFC

- **Dialog Based MFC**
  - 메뉴 바 추가
    - 이벤트 처리기 추가






```
void CViewerDlg::OnFileOpenBtnClick()
{
    // TODO: 여기에 명령 처리기 코드를 추가합니다.
}
```

Dialog Based MFC

KWANGWOON UNIVERSITY

- **Dialog Based MFC**
  - OpenCV를 이용하여 영상 띄우기
    - 헤더에 왼쪽 객체 추가

```
Mat InpImg, OutImg, TempImg;
unsigned char** GrayImg;
unsigned char** U, V;
```

```
void C_ViewerDlg::OnFileOpenBtnClick()
{
    CFileDialog dlg(TRUE);
    if (dlg.DoModal() == IDOK)
    {
        if (dlg.GetFileExt() != "jpg" && dlg.GetFileExt() != "JPG")
        {
            MessageBox("JPG 파일이 아닙니다.");
            return;
        }
        InpImg = imread((const char*)dlg.GetPathName());
        imshow("InpImg", InpImg);
        waitKey(0);
    }
}
```

멀티바이트 문자 집합 사용

Programming

Programming

KWANGWOON UNIVERSITY

- **Lena**
  - Image Processing 의 표준 영상으로 많이 사용
  - 512x512 size JPG

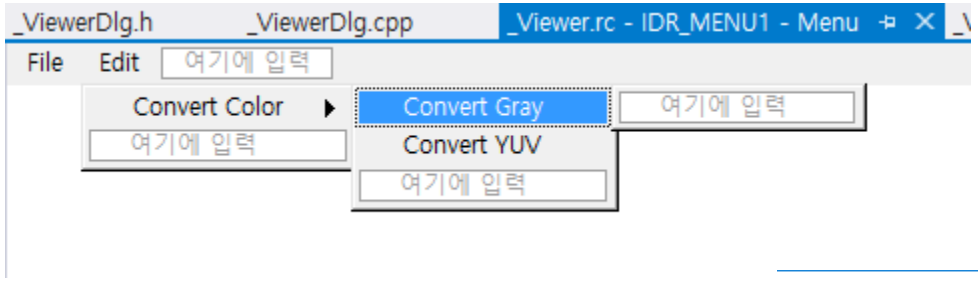
Programming

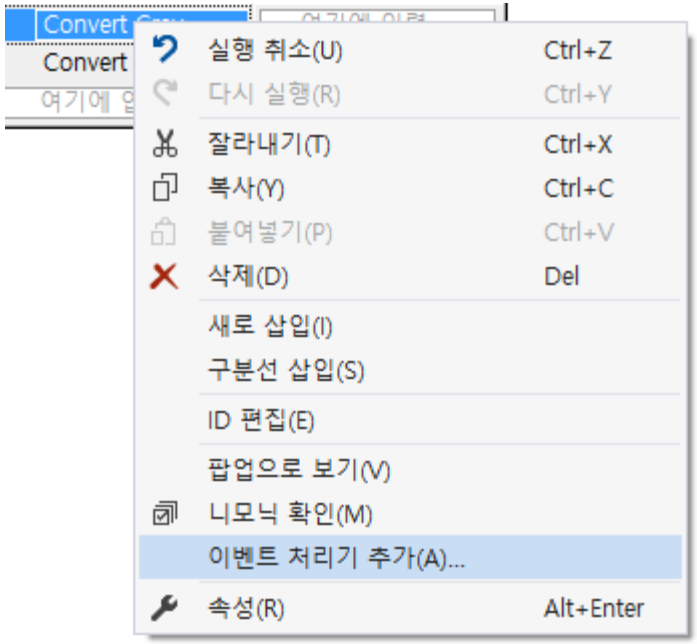
- **Programming #3**
  - Gray Scale Image
    - RGB에서 Y값 계산

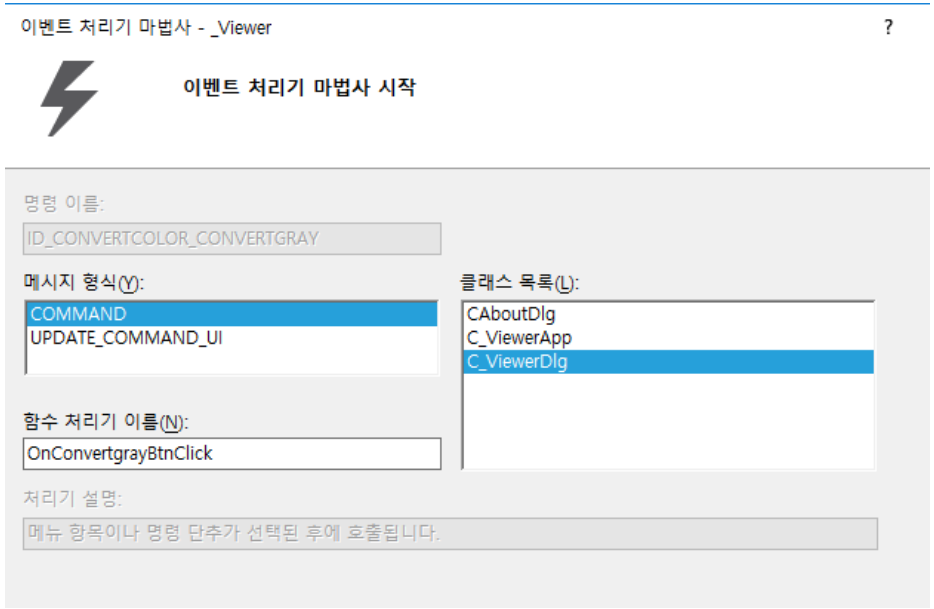
Programming

Programming #3

- 메뉴 바에 새 메뉴 추가 및 이벤트 핸들러 추가







Programming

Programming #3

- 이벤트 핸들러의 빈칸 채우기

```
void C_ViewerDlg::OnConvertgrayBtnClick()
{
    if (InpImg.data != NULL)
    {
        //2D Memory Allocation
        GrayImg = new unsigned char*[InpImg.rows];
        for (int h = 0; h < InpImg.rows; h++) {
            GrayImg[h] = new unsigned char[InpImg.cols];
        }

        // Convert Gray Image



        //

        // 2D Memory Free
        for (int n = 0; n < InpImg.rows; n++)
        {
            delete[] GrayImg[n];
        }
        delete[] GrayImg;

    }
}
```

Programming

KWANGWOON UNIVERSITY

Programming #4

- RGB를 YUV444 포맷으로 변환
- 이벤트 핸들러 추가 후 코드 작성하기

```
void C_ViewerDlg::OnConvertyuvBtnClick()
{
    if (InpImg.data != NULL)
    {
        // 2D Memory Allocation
        GrayImg = new unsigned char*[InpImg.rows];
        U = new unsigned char*[InpImg.rows];
        V = new unsigned char*[InpImg.rows];
        for (int h = 0; h < InpImg.rows; h++)
        {
            GrayImg[h] = new unsigned char[InpImg.cols];
            U[h] = new unsigned char[InpImg.cols];
            V[h] = new unsigned char[InpImg.cols];
        }

        // Convert RGB For YUV444



        //

        // Write YUV444 File

        FILE* fp;
        fopen_s(&fp, "Result.yuv", "wb");

        for (int h = 0; h < InpImg.rows; h++)
        {
            fwrite(GrayImg[h], sizeof(unsigned char), InpImg.cols, fp);
        }
        for (int h = 0; h < InpImg.rows; h++)
        {
            fwrite(U[h], sizeof(unsigned char), InpImg.cols, fp);
        }
        for (int h = 0; h < InpImg.rows; h++)
        {
            fwrite(V[h], sizeof(unsigned char), InpImg.cols, fp);
        }

        fclose(fp);

        // 2D Memory Free
        for (int n = 0; n < InpImg.rows; n++)
        {
            delete[] GrayImg[n];
            delete[] U[n];
            delete[] V[n];
        }
        delete[] GrayImg;
        delete[] U;
        delete[] V;
    }
}
```