



WINAPI -CHAPTER 5-

SOULSEEK

목차

1. StockObject
2. Pen
3. Brush
4. OldObject 와 Bitmap

1. STOCK OBJECT

STOCKOBJECT

- **Windows**가 기본적으로 제공하는 GDI 오브젝트
- 만들거나 삭제하는 과정없이 사용 할 수 있다.

HGDIOBJ GetStockObject(int fnObject);

- 이미 준비되어 있는 스톡 오브젝트를 얻어온다

fnObject	설명	fnObject	설명
BLACK_BRUSH	검정색 브러시	BLACK_PEN	검정색 펜
GRAY_BRUSH	회색 브러시	WHITE_PEN	흰색 펜
NULL_BRUSH	투명 브러시	NULL_PEN	투명 펜
WHITE_BRUSH	흰색 브러시	DC_PEN	특정 색상 펜 SetDCPenColor() 사용
DKGRAY_BRUSH	짙은 회색 브러시	ANSI_FIXED_FONT	고정폭 폰트
LTGRAY_BRUSH	옅은 회색 브러시	ANSI_VAR_FONT	가변폭 폰트
DC_BRUSH	특정 색상이 들어가는 브러시 SetDCBrushColor() 사용	DEFAULT_PALETTE	시스템 팔레트

HGDIOBJ SelectObject(HDC hdc, HGDIOBJ hgdiobj);

- 이전에 사용했던 오브젝트 핸들을 선택해 준다
- **1번째 인자 : DC** 핸들
- **2번째 인자 : 오브젝트** 핸들

STOCKOBJECT

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPARAM lParam)  
{  
    HDC hdc;  
    PAINTSTRUCT ps;  
    HBRUSH MyBrush, OldBrush;  
  
    switch (iMessage)  
    {  
        case WM_PAINT:  
            hdc = BeginPaint(hWnd, &ps);  
  
            MyBrush = (HBRUSH)GetStockObject(GRAY_BRUSH);  
            OldBrush = (HBRUSH)SelectObject(hdc, MyBrush);  
            Rectangle(hdc, 50, 50, 300, 200);  
            SelectObject(hdc, OldBrush);  
  
            EndPaint(hWnd, &ps);  
            return 0;  
        case WM_DESTROY:  
            PostQuitMessage(0);  
            return 0;  
    }  
    return(DefWindowProc(hWnd, iMessage, wParam, lParam));  
}
```

2. PEN

PEN

HPEN CreatePen(int fnPenStyle, int nWidth, COLORREF crColor);

- 1번째 인수 : 그려진 선의 모양을 지정한다 - **PS_SOLID**가 일반적인 직선
- 2번째 인수 : 선의 굵기를 지정한다.
- 3번째 인수 : **COLORREF**형의 컬러 값 **RGB(255, 255, 255);**

BOOL DeleteObject(HGDIOBJ hObject);

- **GDI**오브젝트는 생성하고 사용한 후 삭제해야 한다.

LRESULT CALLBACK WndProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPARAM lParam)

```
{
    HDC hdc;
    PAINTSTRUCT ps;
    HPEN MyPen, OldPen;

    switch (iMessage)
    {
        case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);

            MyPen = CreatePen(PS_SOLID, 5, RGB(0, 0, 255));
            OldPen = (HPEN)SelectObject(hdc, MyPen);
            Rectangle(hdc, 50, 50, 300, 200);
            SelectObject(hdc, OldPen);
            DeleteObject(MyPen);

            EndPaint(hWnd, &ps);
            return 0;
        case WM_DESTROY:
            PostQuitMessage(0);
            return 0;
    }
    return(DefWindowProc(hWnd, iMessage, wParam, lParam));
}
```

3. BRUSH

BRUSH

HBRUSH CreateSolidBrush(COLORREF crColor);

- 지정한 색깔로 채워주는 **Brush**를 생성.

HBRUSH CreateHatchBrush(int fnStyle, COLORREF crColor);

- 지정한 스타일과 색깔로 채워주는 **Brushf**를 생성

값	설명
HS_BDIAGONAL	좌하향 줄무늬
HS_CROSS	바둑판 모양
HS_DIAGCROSS	좌하향 및 우하향 줄무늬
HS_FDIAGONAL	우하향 줄무늬
HS_HORIZONTAL	수평선
HS_VERTICAL	수직선

BRUSH

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPARAM lParam)
{
    HDC hdc;
    PAINTSTRUCT ps;
    HBRUSH MyBrush, OldBrush;

    switch (iMessage)
    {
        case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);

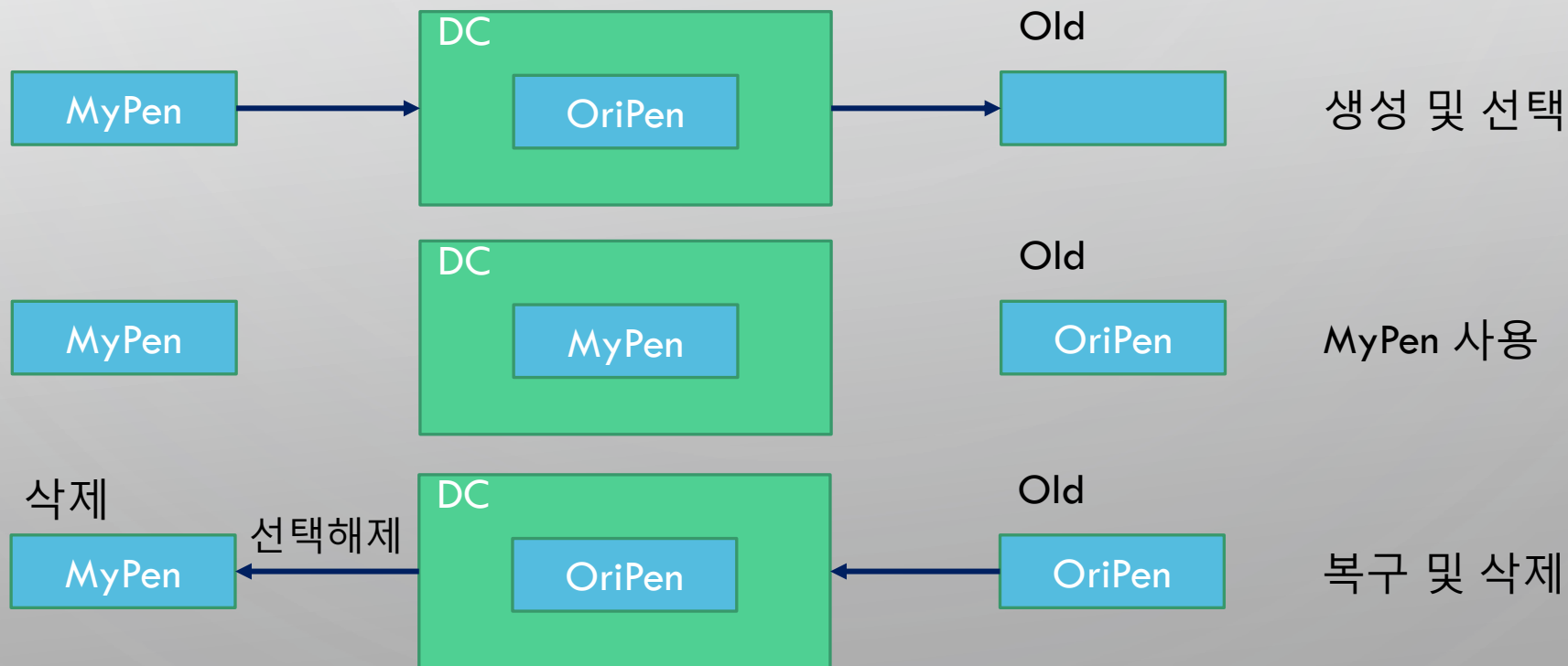
            MyBrush = CreateSolidBrush(RGB(0, 0, 255));
            //MyBrush = CreateHatchBrush(HS_BDIAGONAL, RGB(0, 0, 255));
            OldBrush = (HBRUSH)SelectObject(hdc, MyBrush);
            Rectangle(hdc, 50, 50, 300, 200);
            SelectObject(hdc, OldBrush);
            DeleteObject(MyBrush);

            EndPaint(hWnd, &ps);
            return 0;
        case WM_DESTROY:
            PostQuitMessage(0);
            return 0;
    }
    return(DefWindowProc(hWnd, iMessage, wParam, lParam));
}
```

4. OLD OBJECT와 BITMAP

OLD OBJECT의 의미

GDI 오브젝트들을 생성해서 사용하기만 하고 해제하지 않으면 리소스 영역의 메모리가 모두 소진되어 사용 할 수 없게 된다. 그렇기 때문에 반듯이 삭제를 해주는 것이 원칙이다. 이 과정에서 DC는 지금 사용 중인(Selectobject로 지정된) GDI 오브젝트를 삭제를 해버리면 실제로 DC에 그릴 때 사용하는 오브젝트가 없어지는 것이라 그러지지 않는다. 원칙적으로 GDI 오브젝트들은 DC를 이용할 때 최소한 한 개를 남겨두고 삭제해야 그리기를 수행할 것이다.



BITMAP

HDC CreateCompatibleDC(HDC hdc);

- **DC**와 동일한 정보의 **DC**를 메모리상에 만들어준다.
- 메모리 **DC**를 이용하여 그리기 준비를 하기 위함이다.

HBITMAP LoadBitmap(HINSTANCE hInstance, LPCTSTR lpBitmapName);

- **1**번째 인수 : 인스턴스 핸들
- **2**번째 인수 : 비트맵 리소스 ID

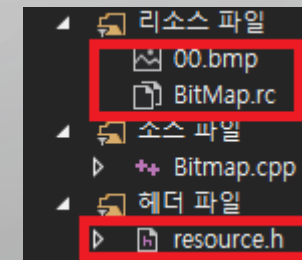
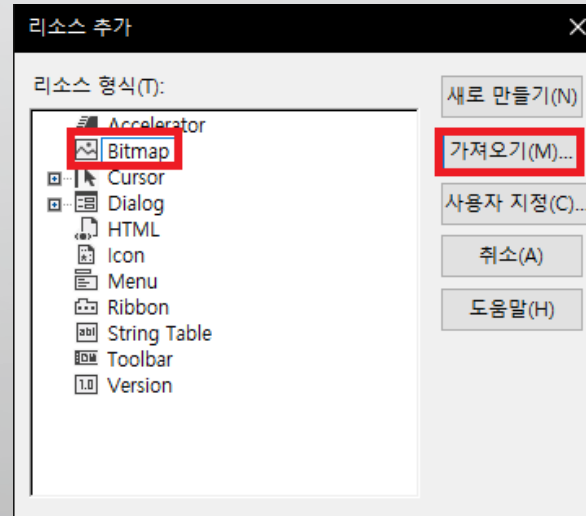
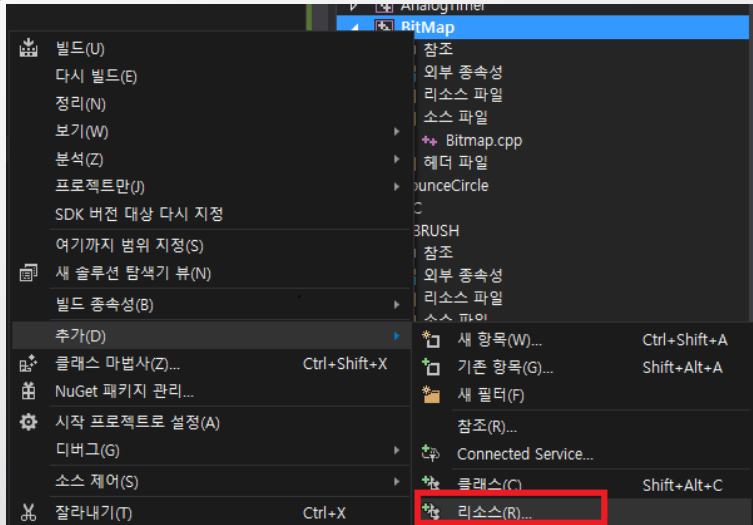
BOOL BitBlt(HDC hdcDest, int nXDest, int nYDest, int nWidth, int nHeight, HDC hdcSrc, int nXSrc, int nYSrc, DWORD dwRop);

- 메모리 **DC**의 비트맵을 **hdc**에 출력한다. 원형 그대로만 그릴 수 있다.
- **1**번째 인수 : **hdc**
- **2, 3**번째 인수 : **x, y** 좌표
- **4, 5**번째 인수 : 그려질 크기 **width, height**
- **6**번째 인수 : 메모리 **DC**
- **7, 8**번째 인수 : 메모리 **DC**에서 어떤 위치에서 그릴 것인지 설정 **x, y**
- **9**번째 인수 : 정보만큼 어떻게 그릴 것인지 **SRCCOPY** - 그대로 복사 한다.

BOOL StretchBlt(HDC hdcDest, int nXOriginDest, int nYOriginDest, int nWidthDest, int nHeightDest, HDC hdcSrc, int nXOriginSrc, int nYOriginSrc, int nWidthSrc, int nHeightSrc, DWORD dwRop);

- 메모리 **DC**의 비트맵을 **hdc**에 출력한다. 그림의 크기를 변경 할 수 있다.
- **1**번째 인수 : **hdc**
- **2, 3**번째 인수 : **x, y**좌표
- **4, 5**번째 인수 : 그려질 크기 **width, height**
- **6**번째 인수 : 메모리 **DC**
- **7, 8**번째 인수 : 메모리 **DC**에서 어떤 위치에서 시작점이 될지 설정 **x, y**
- **9, 10**번째 인수 : 메모리 **DC**의 어떤 위치에서 끝점이 될지 설정 **width, height**
- **11**번째 인수 : 정보만큼 어떻게 그릴 것인지 **SRCCOPY** - 그대로 복사 한다.

BITMAP



BITMAP

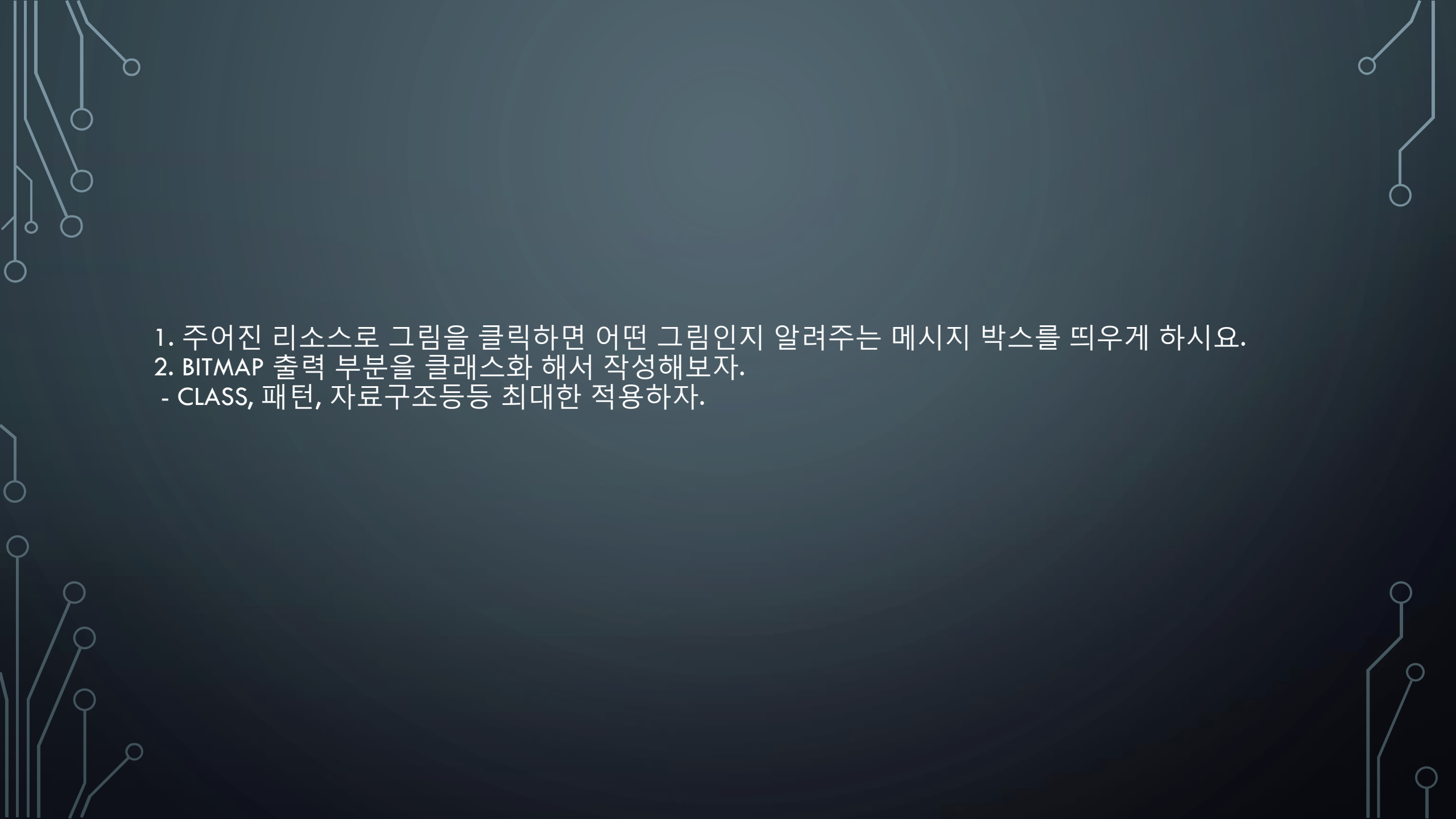
```
#include "resource.h"
```

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPARAM lParam)
{
    HDC hdc, MemDC;
    PAINTSTRUCT ps;
    HBITMAP myBitmap, oldBitmap;

    switch (iMessage)
    {
        case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);

            MemDC = CreateCompatibleDC(hdc);
            myBitmap = LoadBitmap(g_hInst, MAKEINTRESOURCE(IDB_BITMAP1));
            oldBitmap = (HBITMAP)SelectObject(MemDC, myBitmap);
            BitBlt(hdc, 100, 100, 145, 235, MemDC, 0, 0, SRCCOPY);
            StretchBlt(hdc, 300, 100, 290, 470, MemDC, 0, 0, 145, 235, SRCCOPY);
            SelectObject(MemDC, oldBitmap);
            DeleteObject(myBitmap);
            DeleteDC(MemDC);

            EndPaint(hWnd, &ps);
            return 0;
        case WM_DESTROY:
            PostQuitMessage(0);
            return 0;
    }
    return(DefWindowProc(hWnd, iMessage, wParam, lParam));
}
```

- 
- The background is a dark blue gradient. In the corners, there are decorative white line art elements resembling circuit boards or neural networks, with lines and small circles.
1. 주어진 리소스로 그림을 클릭하면 어떤 그림인지 알려주는 메시지 박스를 띄우게 하세요.
 2. BITMAP 출력 부분을 클래스화 해서 작성해보자.
 - CLASS, 패턴, 자료구조등등 최대한 적용하자.