

제 2장 윈도우 기본 입출력 3

2023년 1학기 윈도우 프로그래밍

8. 직선, 원, 사각형, 다각형 그리기

- 점 그리기
- 직선, 원, 사각형, 다각형 그리기
- 도형 속성 변경하기

점 그리기

- 원도우에 점 그리기

COLORREF SetPixel (HDC hdc, int nXpos, int nYpos, COLORREF color);

- hdc: DC 핸들
- nXpos, nYpos: 설정할 점의 x, y 좌표
- color: 점을 그리는데 사용 할 색상

BOOL SetPixelV (HDC hdc, int nxPos, int nYpos, COLORREF color);

- hdc: DC 핸들
- nXpos, nYpos: 설정할 점의 x, y 좌표
- color: 점을 그리는데 사용 할 색상

- 지정한 좌표에서 픽셀의 색상값 가져오기

COLORREF GetPixel (HDC hdc, int nXPos, int nYpos);

- hdc: DC 핸들
- nXpos, nYpos: 설정할 점의 x, y 좌표
- 리턴 값: 픽셀의 색상

직선 그리기

- 직선의 시작점으로 이동하기 함수

```
BOOL MoveToEx (HDC hdc, int X1, int Y1, LPPOINT lpPoint );
```

- hdc: DC 핸들
- X1, Y1: 직선의 시작점 (x와 y 좌표값)
- lpPoint: 이전의 좌표, 사용 안함

(X1, Y1)

(X2, Y2)

- 직선의 시작점에서 끝점까지 직선 그리기 함수

```
BOOL LineTo (HDC hdc, int X2, int Y2 );
```

- hdc: DC 핸들
- X2, Y2: 직선의 끝점

```
case WM_PAINT :
```

```
hdc = BeginPaint (hwnd, &ps) ;
```

```
MoveToEx (hdc, 10, 30, NULL);    // (10, 30) -> (50, 60) 직선 그리기
```

```
LineTo (hdc, 50, 60);
```

```
EndPaint (hwnd, &ps) ;
```

```
return 0 ;
```

(10, 30)

(50, 60)

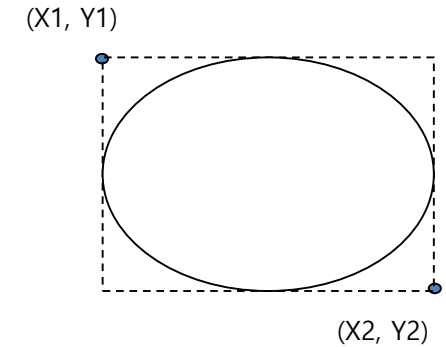
```
typedef struct tagPOINT {  
    LONG x;  
    LONG y;  
} POINT;
```

원 그리기

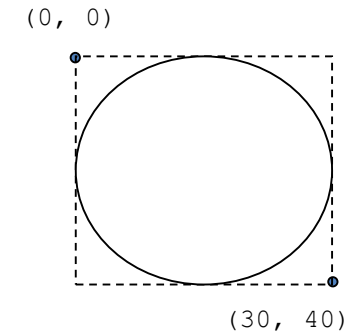
- 두 점의 좌표를 기준으로 만들어진 가상의 사각형에 내접하는 원을 그리는 함수

```
BOOL Ellipse (HDC hdc, int X1, int Y1, int X2, int Y2 );
```

- X1, Y1: 좌측 상단 좌표값 (x와 y 최소값)
- X2, Y2: 우측 하단 좌표값 (x와 y 최대값)



```
case WM_PAINT :  
    hdc = BeginPaint (hwnd, &ps) ;  
    Ellipse(hdc, 0, 0, 30, 40);           // 박스의 중심점 (15,20)  
    EndPaint (hwnd, &ps) ;  
    return 0 ;
```

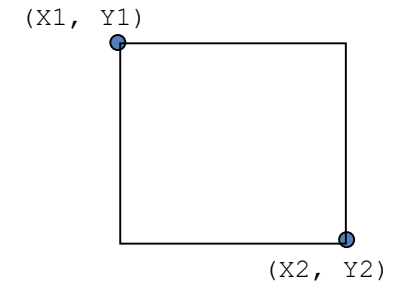


사각형 그리기

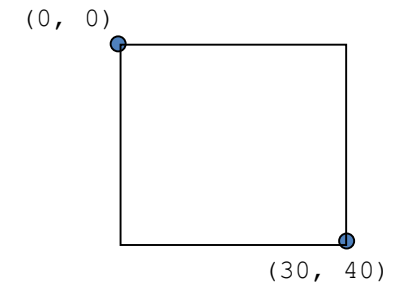
- 두 점의 좌표를 기준으로 수평수직 사각형을 그림

```
BOOL Rectangle (HDC hdc, int X1, int Y1, int X2, int Y2 );
```

- X1, Y1: 좌측 상단 좌표값 (x와 y 최소값)
- X2, Y2: 우측 하단 좌표값 (x와 y 최대값)



```
case WM_PAINT :  
    hdc = BeginPaint (hwnd, &ps) ;  
    Rectangle(hdc, 0, 0, 30, 40);    // 원을 둘러싼 사각형의 좌표  
    EndPaint (hwnd, &ps) ;  
    return 0 ;
```



사각형 그리기

- 사각형 그리기 다른 함수들
 - 내부가 채워진 사각형 그리기:

```
int FillRect (HDC hdc, CONST RECT *lprc, HBRUSH hbr)
```

- hdc: DC 핸들
- lprc: 사각형의 좌표값
- hbr: 내부 색



- 외곽선 사각형 그리기:

```
int FrameRect (HDC hdc, CONST RECT *lprc, HBRUSH hbr);
```

- hdc: DC 핸들
- lprc: 사각형의 좌표값
- hbr: 외곽선 색



- 반전 사각형 그리기:

```
int InvertRect (HDC hdc, CONST RECT *lprc);
```

- hdc: DC 핸들
- lprc: 사각형의 좌표값



- 그리기 이전의 픽셀의 색과 반전된 사각형 그리기

```
typedef struct _RECT {  
    LONG left;  
    LONG top;  
    LONG right;  
    LONG bottom;  
} RECT;
```

실제 사용할 때:

RECT rect = {left좌표, top좌표, right좌표, bottom좌표};

FrameRect (hdc, &rect, hBrush);
InvertRect (hdc, &rect);

사각형 그리기

- 몇가지 알아두면 편리한 사각형 관련 함수들

BOOL OffsetRect (LPRECT lprc, int dx, int dy);

- 주어진 Rect를 dx, dy만큼 이동한다.

BOOL InflateRect (LPRECT lprc, int dx, int dy);

- 주어진 Rect를 dx, dy만큼 늘이거나 줄인다.

BOOL IntersectRect (LPRECT lprcDest, CONST RECT *lprcSrc1, CONST RECT lprcSrc2);

- 두 RECT (lprcSrc1, lprcSrc2)가 교차되었는지 검사한다.
- lprcDest: 교차된 RECT 부분의 좌표값이 저장된다.

BOOL UnionRect (LPRECT lprcDest, CONST RECT *lprcSrc1, CONST RECT *lprcSrc2)

- 두 RECT (lprcSrc1, lprcSrc2) 를 union 시킨다.
- lprcDest: 두 사각형을 합한 사각형의 좌표값이 저장된다.

BOOL PtInRect (CONST RECT *lprc, POINT pt);

- 특정 좌표 pt가 lprc 영역 안에 있는지 검사한다.

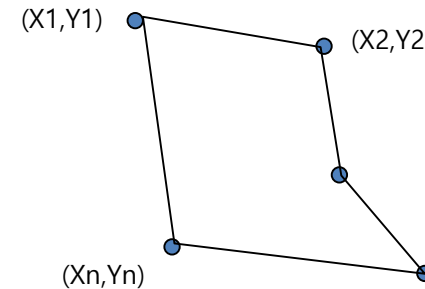
다각형 그리기

- 연속되는 여러 점의 좌표를 직선으로 연결하여 다각형을 그림

```
BOOL Polygon (HDC hdc, CONST POINT *lppt, int cPoints );
```

- hdc: DC 핸들
- lppt: 다각형 꼭지점의 좌표 값이 저장된 리스트
- cPoints: 꼭지점의 개수

```
typedef struct tagPOINT {  
    LONG x;  
    LONG y;  
} POINT;
```

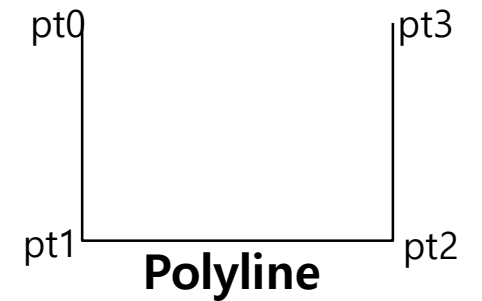
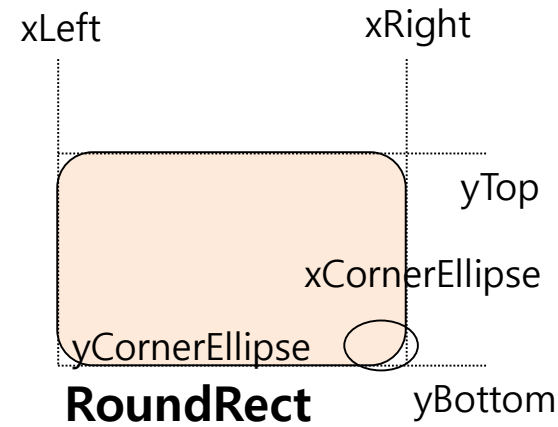
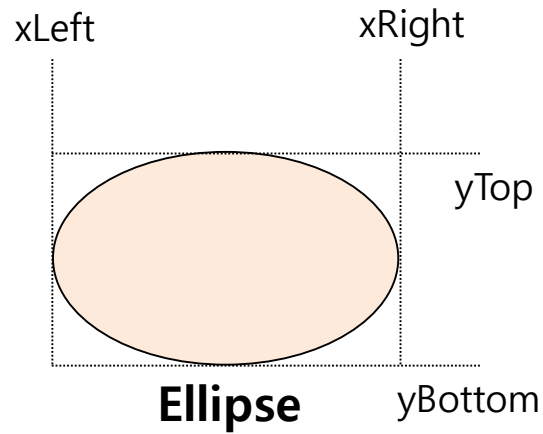
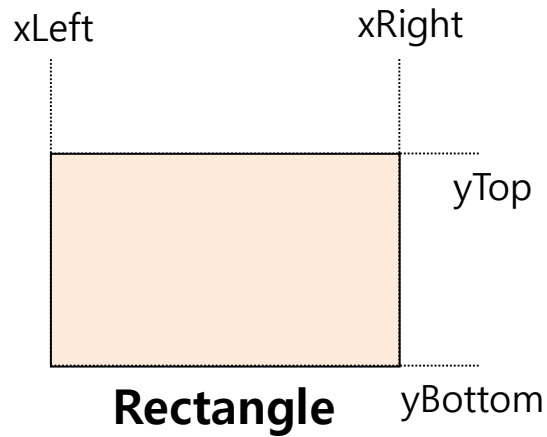


```
POINT point[10] = {{10,20}, {100,30}, {500,200}, {600, 300}, {200, 300}};
```

```
case WM_PAINT :  
    hdc = BeginPaint (hwnd, &ps) ;  
    Polygon(hdc, point, 5);    // 5각형  
    EndPaint (hwnd, &ps) ;  
    return 0 ;
```

도형 그리기

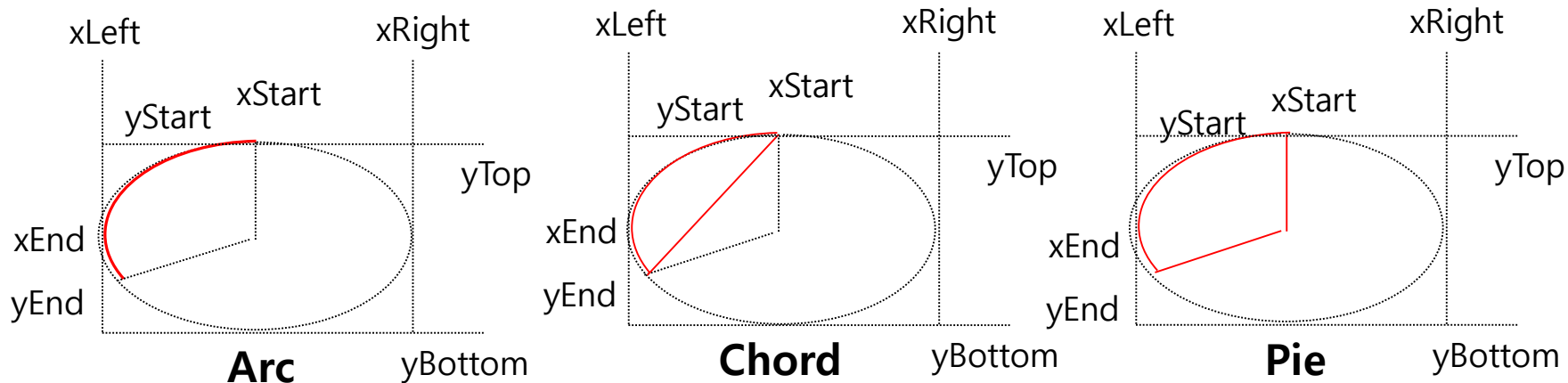
- 다양한 도형 그리기
 - BOOL **Rectangle** (HDC hdc, int X1, int Y1, int X2, int Y2);
 - BOOL **Ellipse** (HDC hdc, int X1, int Y1, int X2, int Y2);
 - BOOL **RoundRect** (HDC hdc, int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, int nWidth, int nHeight);
 - BOOL **Polyline** (HDC hdc, CONST POINT *lppt, int cPoints);



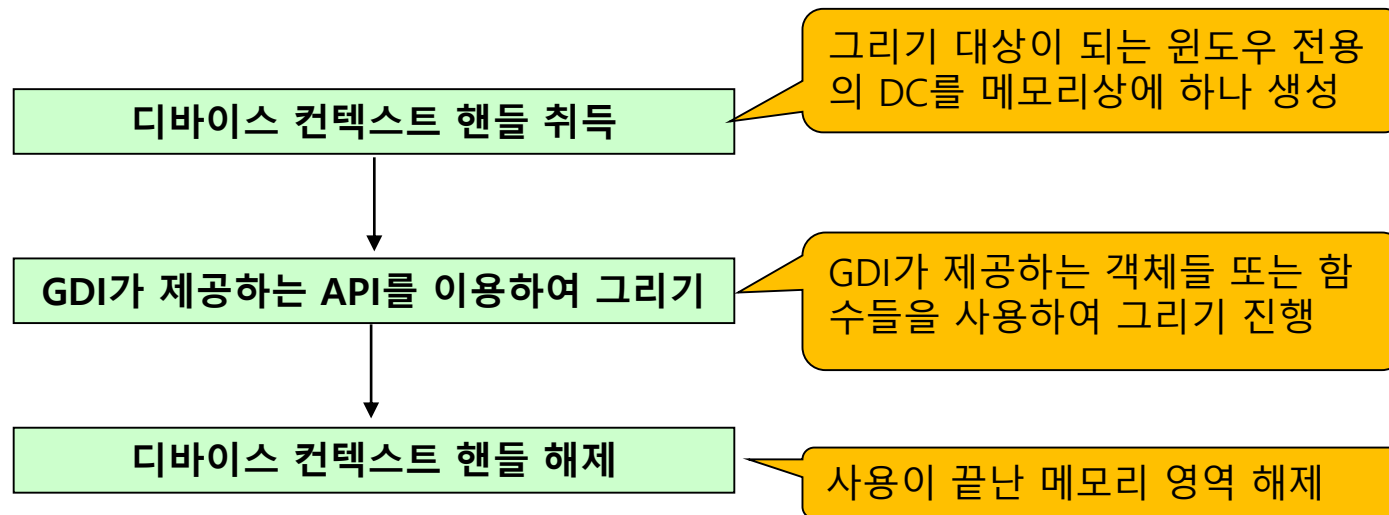
도형 그리기

- 다양한 도형 그리기

- `BOOL Arc (HDC hdc, int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, int nXStartArc, int nYStartArc, int nXEndArc, int nYEndArc);`
- `BOOL Chord (HDC hdc, int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, int nXRadial1, int nYRadial1,`
- `BOOL Pie (HDC hdc, int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, int nXRadial1, int nYRadial1, int nXRadial2, int nYRadial2);`



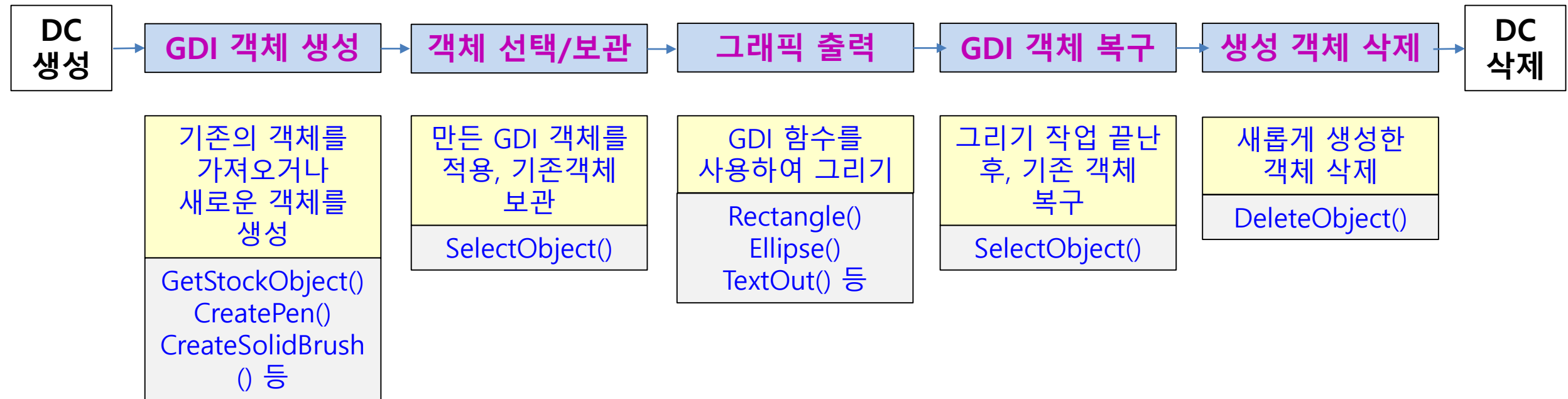
- **GDI (Graphic Device Interface)**
 - 화면, 프린터 등의 모든 출력장치를 제어하는 윈도우 모듈
 - GDI 오브젝트: 그림을 그리는 데 필요한 도구 (펜, 브러쉬 등)
 - GDI를 사용한 그리기 순서



- 윈도우에서 기본적으로 제공하는 GDI객체인 스톡 객체는 따로 생성할 필요가 없어서 편리하지만 다양하게 그래픽을 하기에는 부족
 - 다양한 출력을 위해서는 GDI객체를 만들어서 사용
 - GDI객체를 만드는 함수는 앞의 "Create"로 시작하는 함수

GDI 오브젝트	의미	내용	핸들타입	디폴트 값
펜	선을 그을 때	<ul style="list-style-type: none">• 선을 그리거나 영역의 경계선을 그릴 때 사용• 선의 색, 두께, 형태 등을 지정• 디폴트는 검은색 1픽셀 실선	HPEN	검정색의 가는 실선
브러시	면을 채울 때	<ul style="list-style-type: none">• 어떤 영역의 내부를 채울 때 사용• 채우기 색, 채우기 패턴 등을 지정• 디폴트는 흰색	HBRUSH	흰색
폰트	문자 출력 글꼴	<ul style="list-style-type: none">• 문자를 출력할 때 사용하며 색, 모양, 크기 등을 지정• 디폴트는 시스템 폰트	HFONT	시스템 글꼴
비트맵	비트맵 이미지	<ul style="list-style-type: none">• 비트맵 그림 파일에 관한 옵션	HBITMAP	X
팔레트	팔레트	<ul style="list-style-type: none">• 화면에 출력할 수 있는 색에 제한을 받을 경우, 실제로 화면에 출력할 색의 수 등을 지정	HPALETTE	X
리전	화면상의 영역	<ul style="list-style-type: none">• 임의의 도형을 그리는 것과 관련된 옵션을 설정	HRGN	X

GDI 객체를 사용하는 방법







- GDI객체를 만드는 함수
 - 펜:
 - HPEN CreatePen (int fnPenStyle, int nWidth, COLORREF crColor);
 - 브러시:
 - HBRUSH CreateSolidBrush (COLORREF crColor);
 - HBRUSH CreateHatchBrush (int iHatch, COLORREF crColor);
 - 폰트(글꼴):
 - HFONT CreateFont (int cHeight, int cWidth, int cEscapement, int cOrientation, int cWeight, DWORD bItalic, DWORD bUnderline, DWORD bStrikeOut, DWORD iCharSet, DWORD iOutPrecision, DWORD iClipPrecision, DWORD iQuality, DWORD iPitchAndFamily, LPCSTR pszFaceName);
 - HFONT CreateFontIndirect (const LOGFONT *lpLf);
 - 비트맵:
 - HBITMAP CreateBitmap (int nWidth, int nHeight, UINT nPlanes, UINT nBitCount, const VOID *lpBits);
 - HBITMAP CreateCompatibleBitmap (HDC hdc, int cx, int cy);

펜 : 선 다루기

- 선의 굵기 정보와 색상정보를 가지는 펜 핸들을 생성 함수

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

- fnPenStyle: 펜 스타일
- nWidth: 펜의 굵기로 단위는 픽셀
- crColor: 색상을 표현하기 위해 COLORREF 값을 제공하며, RGB() 함수로 만들

PS_SOLID	
PS_DASH	
PS_DOT	
PS_DASHDOT	
PS_DASHDOTDOT	

- 그림을 그릴 화면인 DC에 펜 핸들을 등록 함수

HGDIOBJ **SelectObject** (HDC hDC, HGDIOBJ hgdiobj);


```
HPEN SelectObject (HDC hdc, HPEN pen);
```

- 그림 그리기를 마친 후 생성된 펜 핸들은 삭제 함수

BOOL **DeleteObject** (HGDI OBJ hgdiob);

BOOL DeleteObject (HPEN pen);

- **COLORREF RGB (int Red, int Green, int Blue)**

- Red, Green, Blue에는 0~255 사이의 정수 값을 사용
- COLORREF 는 색상을 표현하는 자료형 (R, G, B 3가지 색으로 표현)

빨간 점선으로 원 그리기

```
LRESULT CALLBACK wndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
```

```
{
```

```
    HDC hdc;
```

```
    HPEN hPen, oldPen;
```

```
    switch (iMsg) {
```

```
        case WM_PAINT :
```

```
            hdc = BeginPaint (hwnd, &ps) ;
```

```
            // DC 얻어오기
```

```
            hPen = CreatePen (PS_DOT, 1, RGB(255,0,0));
```

```
            // GDI: 펜 만들기
```

```
            oldPen = (HPEN) SelectObject (hdc, hPen);
```

```
            // 새로운 펜 선택하기
```

```
            Ellipse(hdc, 20,20, 300,300);
```

```
            // 선택한 펜으로 도형 그리기
```

```
            SelectObject (hdc, oldPen);
```

```
            // 이전의 펜으로 돌아감
```

```
            DeleteObject (hPen);
```

```
            // 만든 펜 객체 삭제하기
```

```
            EndPaint (hwnd, &ps) ;
```

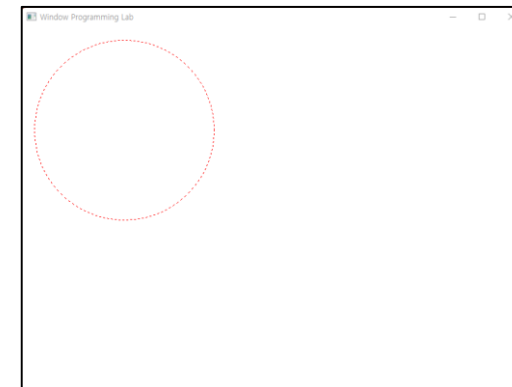
```
            // DC 해제하기
```

```
            break;
```

```
    }
```

```
    return DefWindowProc (hwnd, iMsg, wParam, lParam);
```

```
}
```



면 색상 변경

- 면의 색상정보를 가지는 브러시 핸들을 만들어 주는 함수

```
HBRUSH CreateSolidBrush (COLORREF crColor);  
- crColor: 면의 색상 값
```

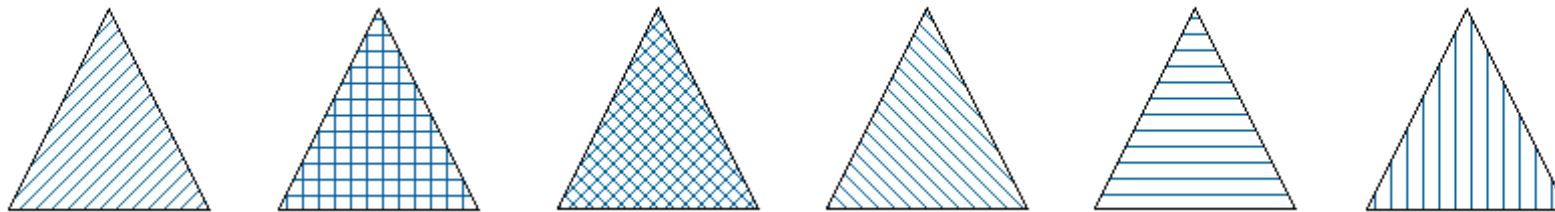
- 그림을 그릴 화면인 디바이스 컨텍스트에 브러시 핸들을 등록
HGDIOBJ **SelectObject** (HDC hdc, HGDIOBJ hgdiobj);
→
HBRUSH **SelectObject** (HDC hdc, HBRUSH hBrush);
- 그림 그리기를 마친 후 생성된 브러시 핸들은 삭제
BOOL **DeleteObject** (HGDIOBJ hgdiobj);
→
BOOL **DeleteObject** (HBRUSH hBrush);

면 색상 변경 (빗살무늬)

- 면에 빗살무늬와 색상정보를 가지는 브러시 핸들을 만들어 주는 함수

HBRUSH CreateHatchBrush (int iHatch, COLORREF crColor);

- iHatch: 브러시 해치 스타일
 - HS_BDIAGONAL: 45도 위쪽 왼쪽에서 오른쪽 해치
 - HS_CROSS: 가로 및 세로 크로스패치
 - HS_DIAGCROSS: 45도 크로스해치
 - HS_FDIAGONAL: 왼쪽에서 오른쪽으로 45도 아래쪽 해치
 - HS_HORIZONTAL: 가로 빗살 무늬
 - HS_VERTICAL: 세로 해치
- crColor: 면의 색상 값



빨간면의 원 그리기

```
LRESULT CALLBACK wndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
```

```
{
    HDC hdc;
    HBRUSH hBrush, oldBrush;

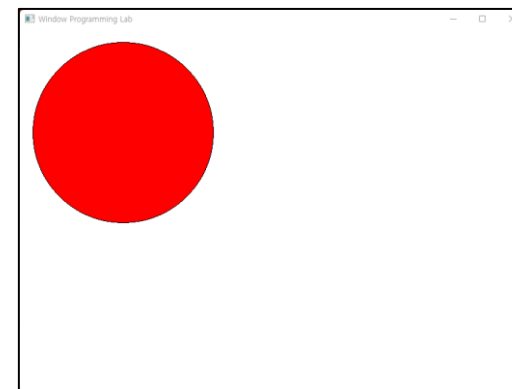
    switch (iMsg) {
        case WM_PAINT :
            hdc = BeginPaint (hwnd, &ps) ;                // DC 얻어오기

            hBrush = CreateSolidBrush (RGB(255,0,0));       // GDI: 브러시 만들기
            oldBrush = (HBRUSH) SelectObject (hdc, hBrush); // 새로운 브러시 선택하기

            Ellipse(hdc, 20,20, 300,300);                  // 선택한 브러시로 도형 그리기

            SelectObject (hdc, oldBrush);                  // 이전의 브러시로 돌아가기
            DeleteObject (hBrush);                          // 만든 브러시 객체 삭제하기

            EndPaint (hwnd, &ps) ;                          // DC 해제하기
            return 0 ;
    }
    return DefWindowProc (hwnd, iMsg, wParam, lParam);
}
```



GDI 객체 핸들 구하기 함수들

- 생성한 펜, 브러시 및 폰트를 설정하는 함수
 - 지정된 디바이스 컨텍스트로 객체를 선택한다.
 - 새로운 객체는 동일한 형식의 이전 객체를 대체한다.

HGDIOBJ SelectObject (HDC hDC, HGDIOBJ hgdiobj);

- hDC: DC 핸들값
- hgdiobj: GDI의 객체
- 리턴 값은 원래의 오브젝트 값

- 생성한 객체 삭제 함수

BOOL DeleteObject (HGDIOBJ hgdiobj);

- GDI오브젝트를 삭제하고 오브젝트가 사용하던 시스템 리소스를 해제
- 객체 생성 함수로 만들어진 GDI 오브젝트는 반드시 삭제해주어야 한다.
- hgdiobj: GDI의 객체

기존의 GDI 객체 사용하기

- 윈도우가 제공하는 펜 브러시 및 폰트를 취득하는 함수
 - 스톡 오브젝트: 윈도우에서 기본적으로 제공하는 GDI 객체
 - 윈도우가 제공해주므로 따로 생성하지 않고 사용할 수 있으며 해제하지 않아도 됨.

HGDIOBJ GetStockObject (int fnObject);

- 리턴 값은 가져올 스톡 오브젝트 핸들 값
- fnObject: 가져올 스톡 오브젝트의 속성
 - BLACK_BRUSH / DKGRAY_BRUSH / DC_BRUSH / GRAY_BRUSH / HOLLOW_BRUSH / LTGRAY_BRUSH / NULL_BRUSH / WHITE_BRUSH
 - BLACK_PEN / DC_PEN / WHITE_PEN

- 클라이언트의 영역을 취득하는 함수

BOOL GetClientRect (HWND hWnd, LPRECT lprc);

- 클라이언트의 영역을 취득한다.
- hWnd: 윈도우 핸들
- lprc: 윈도우 영역의 좌표값

GDI 객체 핸들하기

- 검정색 테두리를 가지고 회색 내부를 가진 사각형을 그리는 경우 → 펜은 변경하지 않고, 브러시 색상만 변경 → 객체를 만들지 않고 윈도우가 제공하는 브러시 객체 (스톡 객체) 사용

```
LRESULT CALLBACK wndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
```

```
{
```

```
    HDC hdc;
```

```
    HBRUSH hBrush, oldBrush;
```

```
    switch (iMsg) {
```

```
        case WM_PAINT :
```

```
            hdc = BeginPaint (hwnd, &ps) ;
```

```
// DC 얻어오기
```

```
            hBrush = (HBRUSH) GetStockObject (GRAY_BRUSH);
```

```
// 윈도우가 제공하는 객체 가져오기
```

```
            oldBrush = (HBRUSH) SelectObject (hdc, hBrush);
```

```
            Rectangle (hdc, 50, 50, 300, 200);
```

```
            SelectObject (hdc, oldBrush);
```

```
            EndPaint (hwnd, &ps) ;
```

```
// DC 해제하기
```

```
            return 0 ;
```

```
    }
```

```
    return DefWindowProc (hwnd, iMsg, wParam, lParam);
```

```
}
```

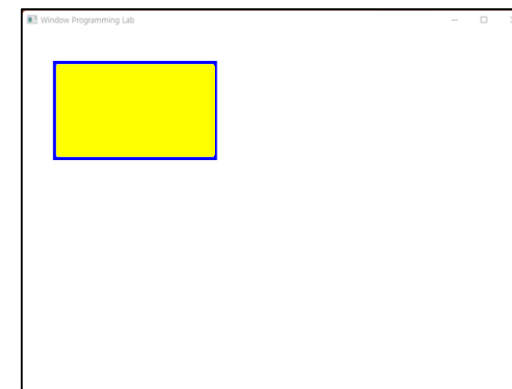


GDI 객체 핸들하기

- 파란색 테두리를 가지고, 노란색 내부를 가진 사각형을 그리는 경우 → 파란색의 펜 객체와 노란색 브러시 객체를 만들어 사용

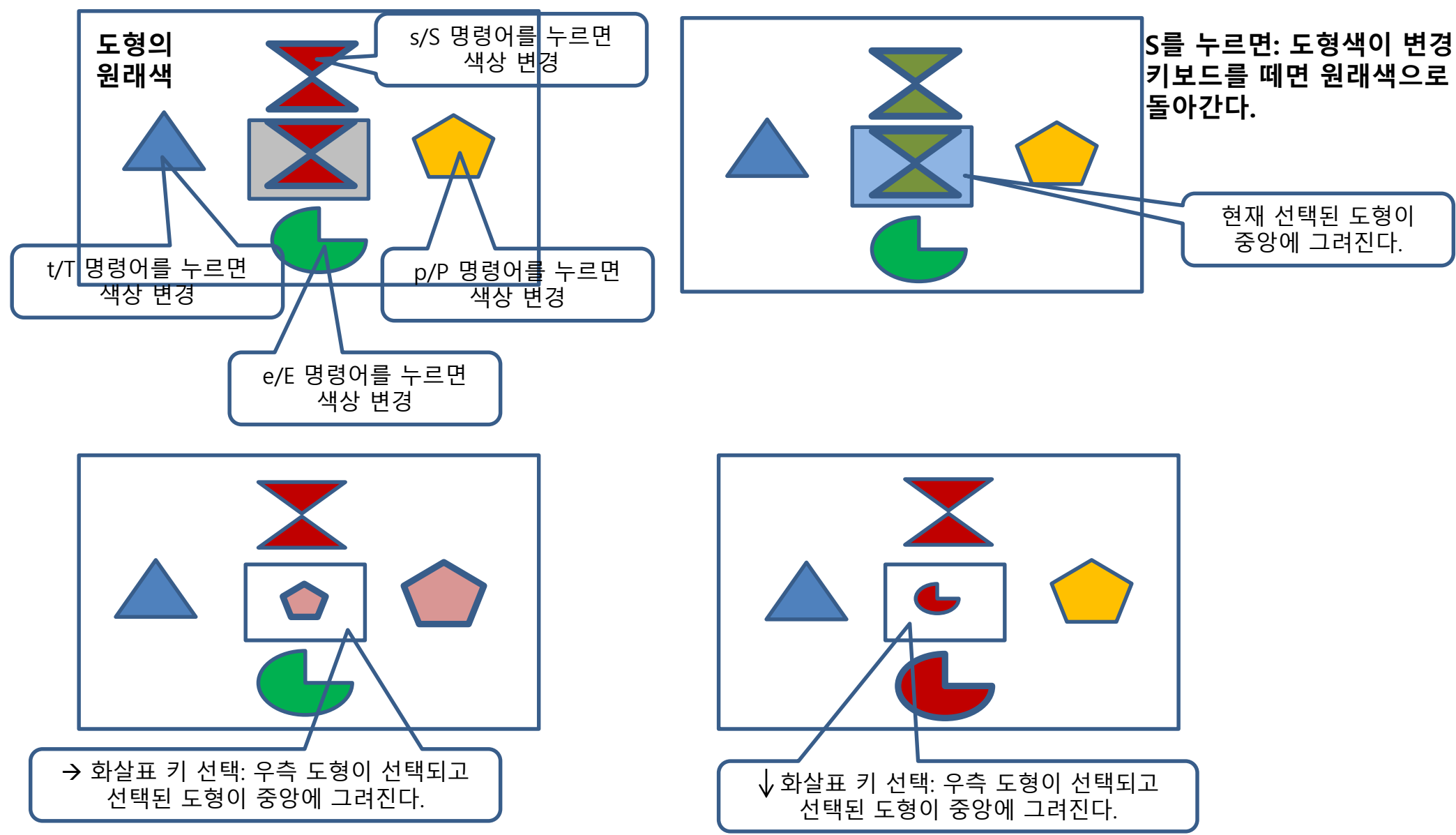
```
LRESULT CALLBACK wndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
```

```
{  
    HDC hdc;  
    HPEN hPen, oldPen;  
    HBRUSH hBrush, oldBrush;  
  
    switch (iMsg) {  
        case WM_PAINT :  
            hdc = BeginPaint (hwnd, &ps) ;  
                                                    // DC 얻어오기  
  
            hPen = CreatePen (PS_SOLID, 5, RGB(0, 0, 255));  
            oldPen = (HPEN) SelectObject (hdc, hPen);  
            hBrush = CreateSolidBrush (RGB(255, 255, 0));  
            oldBrush = (HBRUSH) SelectObject (hdc, hBrush);  
                                                    // 새로운 객체 만들기  
  
            Rectangle (hdc, 50, 50, 300, 200);  
  
            SelectObject (hdc, oldPen);  
            SelectObject (hdc, oldBrush);  
            DeleteObject (hPen);  
            DeleteObject (hBrush);  
  
            EndPaint (hwnd, &ps) ;  
            return 0 ;  
                                                    // DC 해제하기  
        }  
    }  
    return DefWindowProc (hwnd, iMsg, wParam, lParam);  
}
```



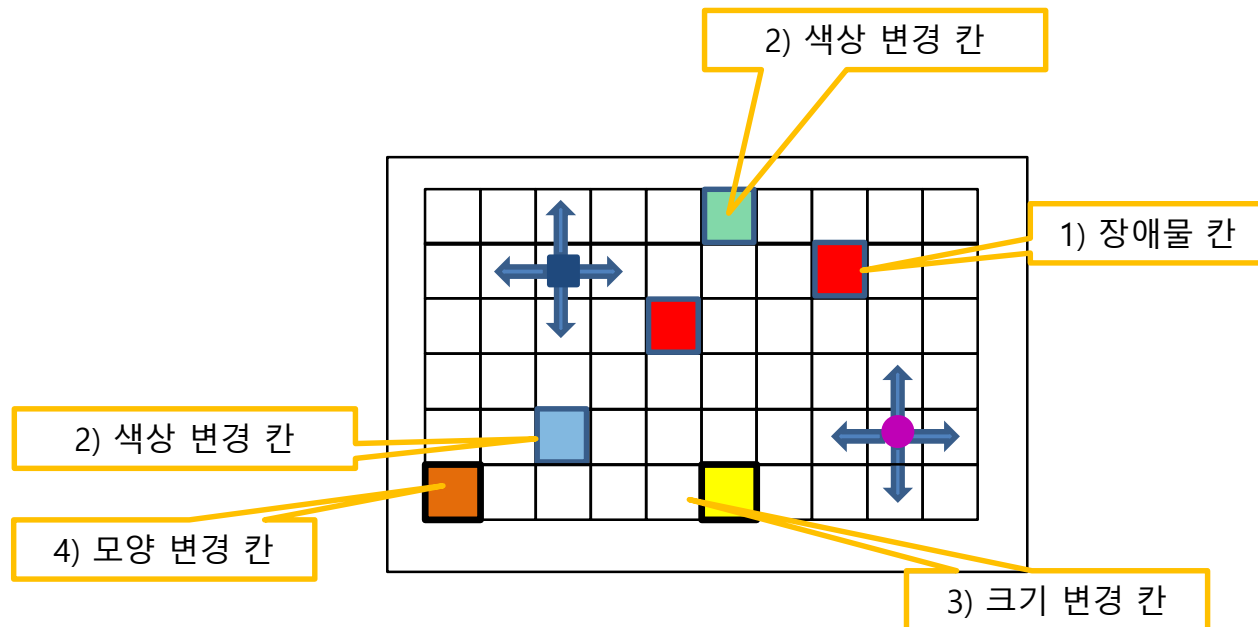
실습 2-9

- 화면에 도형 그리고 색 입히기
 - 윈도우를 띄운다.
 - 윈도우 중앙에 사각형을 그린다.
 - 사각형의 좌우상하에 각각 삼각형, 모래시계, 오각형, 파이 모양을 그린다.
 - 좌우상하 도형 중 한 개의 도형이 랜덤하게 선택되어 있다. 선택된 도형은 중앙의 사각형 안에 그려진다.
 - **도형 선택 및 색상 변경: 아래의 키보드 명령어에 따라 도형이 선택되고, 선택된 도형의 색상을 랜덤하게 바꾸고 중앙의 사각형 안에 해당 도형이 같은 크기로 그려진다.** 이때 키보드를 누르면 색이 변경되고 떼면 다시 원래의 색으로 돌아간다.
 - t/T: 삼각형의 색
 - s/S: 모래시계의 색
 - p/P: 오각형의 색
 - e/E: 파이 모양의 색
 - 색상이 변경되는 도형이 선택된 도형으로, 그 도형은 역시 중앙의 사각형 안에 그려진다.
 - **도형 선택 및 크기 변경: 아래의 키보드 명령어에 따라 도형이 선택되고, 선택된 도형의 크기가 작게 바뀌어 중앙의 사각형 안에 그려진다.** 이때 키보드를 누르면 색상과 크기가 변경되고 떼면 다시 원래의 색상, 크기로 돌아간다.
 - </>/↑/↓: 화살표 키에 따라 좌우상하의 도형이 각각 선택된다.
 - 모든 명령어는 다른 키보드로 바꿀 수 있다.
 - q/Q: 프로그램 종료



• 돌 이동하기

- 화면에 40X40 칸을 그린다.
- 칸의 중간 중간에 네 종류의 다른 색의 칸들이 있다. (최소 20개)
 - 1) 장애물 (빨간색 칸) - 객체들이 통과할 수 없다.
 - 2) 색상 변경 칸 - 돌이 이 칸을 지나면 색이 랜덤하게 바뀐다. 색상 변경 칸은 다양한 색으로 설정한다.
 - 3) 크기 변경 칸 - 돌의 크기가 축소된다. 최소 크기가 되면 그 후에는 돌의 크기가 커진다. 돌의 크기는 바둑판의 칸이 최대 크기이다.
 - 4) 모양 변경 칸 - 돌의 모양이 변경된다. 4종류의 다른 모양으로 바뀔 수 있다. (네모 → 원, 원 → 세모 ...)
- 좌상, 우하 위치에 두 개의 다른 색을 가진 돌을 그린다.
 - 빈 칸의 부분에 돌이 생긴다.
- 2인 플레이로 각각 다른 키보드 명령어를 이용하여 자신의 돌을 칸에 맞추어 좌우상하 이동한다.
 - 가장자리에 도착하면 반대쪽으로 나타난다.
 - 장애물 칸: 지날 수 없다.
 - 색상 변경칸: 색상이 랜덤하게 변한다.
 - 크기 변경칸: 돌의 크기가 변한다.
 - 모양 변경칸: 돌의 모양이 변한다.
- 플레이는 한 명씩 번갈아 하도록 한다.
 - 같은 돌을 두 번 이동하려고 하면 에러 메시지 출력
- 키보드 명령
 - 2인의 돌 이동 키보드 (좌우상하 두 세트)
 - r/R: 새롭게 시작
 - q/Q: 프로그램 종료



- 2장 학습내용
 - GDI (Graphic Device Interface)
 - DC (Device Context)
 - DC를 얻어오는 함수들
 - 메시지:
 - WM_PAINT/WM_CREATE/WM_QUIT
 - 문자 출력함수
 - BOOL TextOut (HDC hdc, int x, int y, LPCTSTR lpString, int nLength);
 - int DrawText (HDC hdc, LPCSTR lpString, int nLength, LPRECT lpRect, UINT Flags);
 - 문자 입력 메시지
 - WM_CHAR, WM_KEYDOWN, WM_KEYUP
 - WM_PAINT 메시지 발생 함수
 - BOOL InvalidateRect (HWND hWnd, const RECT* lpRect, BOOL bErase);
 - BOOL InvalidateRgn (HWND hWnd, HRGN hRgn, BOOL bErase);
 - 캐럿 만들기
 - 도형 그리기
 - 선, 원, 사각형, 다각형
 - GDI 객체 다루기
 - 펜, 브러시 사용하기
 - 스톱 오브젝트 사용하기
 - 객체 생성 함수들, 객체 선택 함수들