제 2장 윈도우 기본 입출력 2

2022년 1학기 윈도우 프로그래밍

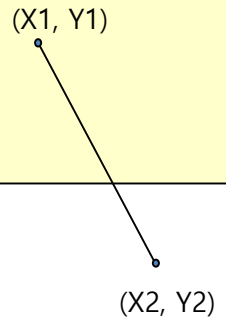
8. 직선, 원, 사각형, 다각형 그리기

- 직선 그리기
- 원 그리기
- 사각형 그리기
- 다각형 그리기
- 선 속성 바꾸기
- 면 색 바꾸기

직선 그리기

- 직선의 시작점으로 이동하기 함수

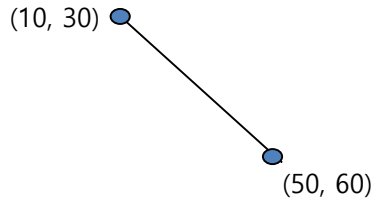
BOOL MoveToEx (HDC hdc, int X1, int Y1, LPPOINT lpPoint);
- hdc: DC 핸들
- X1, Y1: 직선의 시작점 (x와 y 좌표값)
- lpPoint: 이전의 좌표, 사용 안함



- 직선의 시작점에서 끝점까지 직선 그리기 함수

BOOL LineTo (HDC hdc, int X2, int Y2);
- hdc: DC 핸들
- X2, Y2: 직선의 끝점

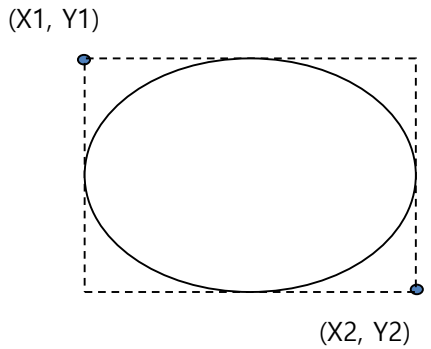
```
case WM_PAINT :
        hdc = BeginPaint (hwnd, &ps) ;
        MoveToEx (hdc, 10, 30, NULL);      // (10, 30) -> (50, 60) 직선 그리기
        LineTo (hdc, 50, 60);
        EndPaint (hwnd, &ps) ;
        return 0 ;
```



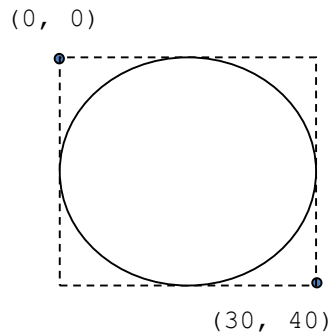
원 그리기

- 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)



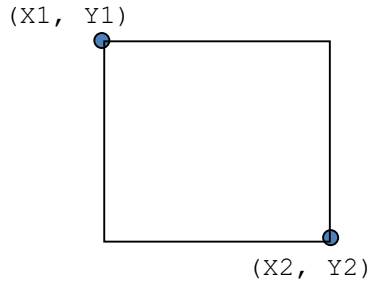

```
case WM_PAINT :
        hdc = BeginPaint (hwnd, &ps) ;
        Ellipse(hdc, 0, 0, 30, 40);          // 박스의 중심점 (15,20)
        EndPaint (hwnd, &ps) ;
        return 0 ;
```



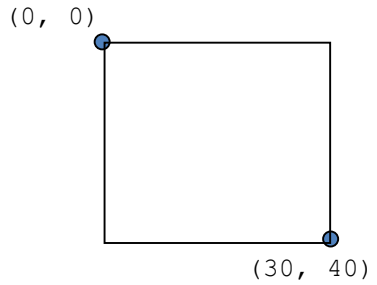
사각형 그리기

- 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)



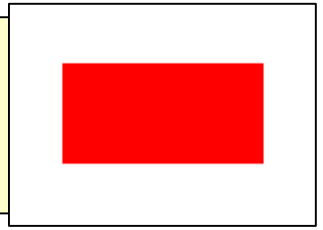

```
case WM_PAINT :
        hdc = BeginPaint (hwnd, &ps) ;
        Rectangle(hdc, 0, 0, 30, 40);      // 원을 둘러싼 사각형의 좌표
        EndPaint (hwnd, &ps) ;
        return 0 ;
```



사각형 그리기

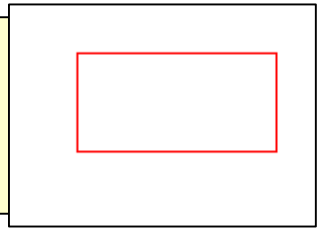
- **사각형 그리기 다른 함수들**
  - 내부가 채워진 사각형 그리기:

int FillRect (HDC hDC, CONST RECT *lprc, HBRUSH hbr)
  - hDC: DC 핸들
  - lprc: 사각형의 좌표값
  - hbr: 내부 색



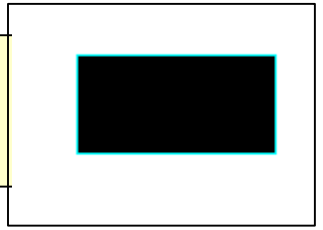
  - 외곽선 사각형 그리기:

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

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

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

실제 사용할 때:
RECT rect = {left좌표, top좌표, right좌표, bottom좌표};

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.

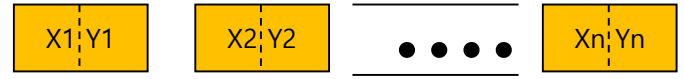
BOOL PtInRect (CONST RECT *lprc, POINT pt);
- 특정 좌표 pt가 lprc 영역 안에 있는지 검사한다.

다각형 그리기

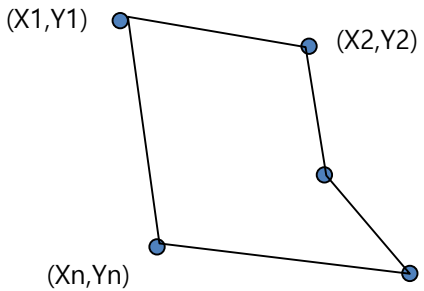
- 연속되는 여러 점의 좌표를 직선으로 연결하여 다각형을 그림

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

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



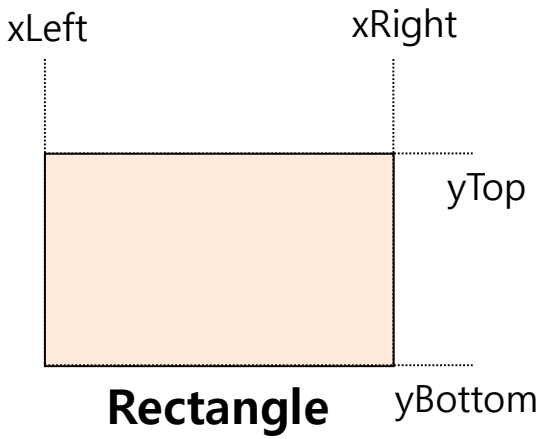

```
typedef struct tagPOINT {
        LONG x;
        LONG y;
} POINT;
```

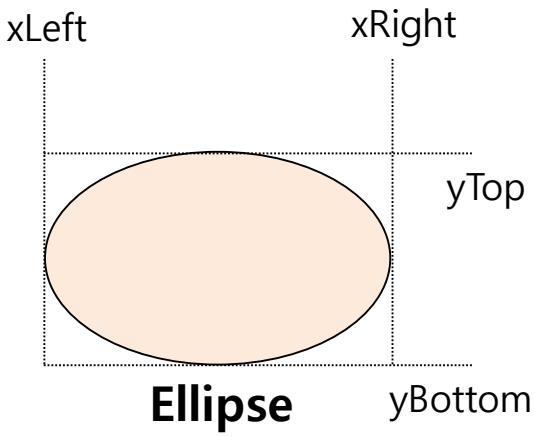


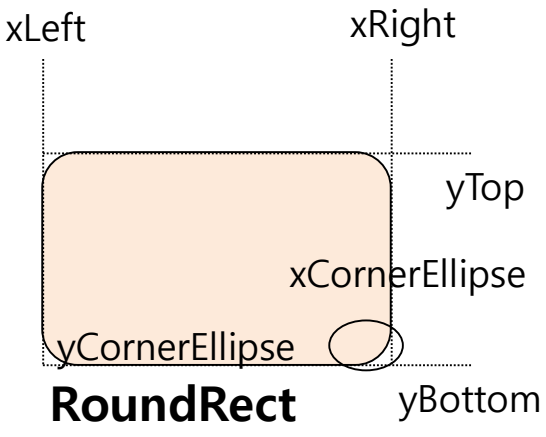

```
POINT point[10] = {{10,20}, {100,30}, {500,200}, {600, 300}, {200, 300}};

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

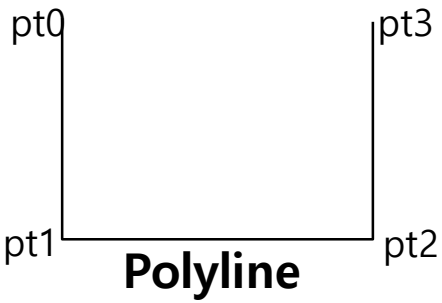
도형 그리기

- **다양한 도형 그리기**
  - BOOL Ellipse (HDC hdc, int X1, int Y1, int X2, int Y2);
  - BOOL Rectangle (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);
  - COLORREF SetPixel (HDC hdc, int nXPos, int nYpos, COLORREF color);



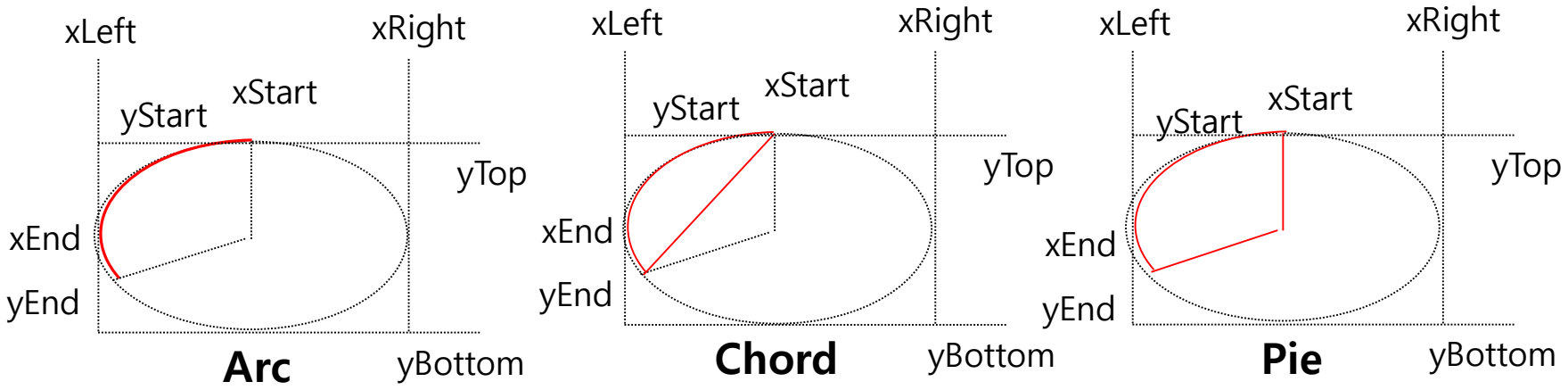






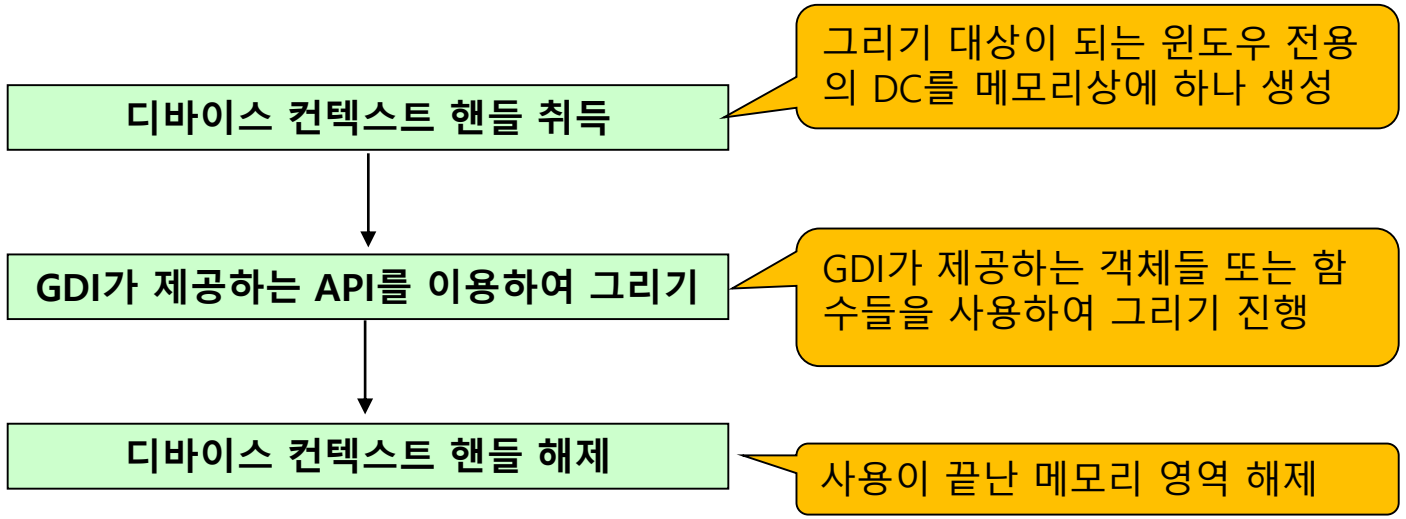
도형 그리기

- **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 nY
  - BOOL Pie (HDC hdc, int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, int nXRadial1, int nYRadial1, int nXRadial2, int nYRadial2);



GDI 객체들

- **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 객체들

GDI 오브젝트	의미	내용	핸들타입	디폴트 값
펜	선을 그을 때	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	어떤 영역의 내부를 채울 때 사용 채우기 색, 채우기 패턴 등을 지정 디폴트는 흰색	HBRUSH	흰색
폰트	문자 출력 글꼴	문자를 출력할 대 사용하며 색, 모양, 크기 등을 지정 디폴트는 시스템 폰트	HFONT	시스템 글꼴
비트맵	비트맵 이미지	비트맵 그림 파일에 관한 옵션	HBITMAP	X
팔레트	팔레트	화면에 출력할 수 있는 색에 제한을 받을 경우, 실제로 화면에 출력할 색의 수 등을 지정	HPALETTE	X
리전	화면상의 영역	임의의 도형을 그리는 것과 관련된 옵션을 설정	HRGN	X

객체를 사용하는 방법

순서	내용	사용 함수
1. **DC 생성**:	우선 BeginPaint()나 GetDC() 등의 함수를 이용하여 DC를 생성	BeginPaint(), GetDC() 등
2. **GDI 객체 생성**:	GetStockObject()로 스톡 객체나 Create로 시작하는 함수로 GDI객체를 생성	GetStockObject(), CreatePen(), CreateSolidBrush () 등
3. **객체 선택 및 보관**:	만든 GDI객체를 SelectObject()함수로 선택하고, 이 함수의 리턴값인 이전의 GDI객체를 그리기 작업을 다한 후 DC를 원상 복구할 목적으로 보관	SelectObject()
4. **설정**:	SetBkColor()함수로 배경색을 설정하거나 Set으로 시작하는 함수들을 이용하여 다양한 설정	SetBkColor() 등
5. **그래픽 출력**:	GDI함수를 이용하여 그리기 작업	Rectangle(), TextOut() 등
6. **GDI 객체 복구**:	그리기 작업이 다 끝난 후에는 보관해놓았던 이전 GDI객체를 SelectObject() 함수로 선택하여 이전 DC상태로 복구	SelectObject()
7. **생성 객체 삭제**:	생성한 GDI객체를 DeleteObject()함수로 삭제	DeleteObject()
8. **DC 소멸**:	생성했던 DC를 EndPaint()나 ReleaseDC()함수로 소멸	EndPaint(), ReleaseDC() 등

GDI 객체 생성

- **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 만듦

PS_SOLID	————————
PS_DASH	– – – – – – – – – – – – –
PS_DOT	----------
PS_DASHDOT	_._._._._._._._._._
PS_DASHDOTDOT	_.._.._.._.._.._.._

- **그림을 그릴 화면인 DC에 펜 핸들을 등록 함수**

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

- **그림 그리기를 마친 후 생성된 펜 핸들은 삭제 함수**

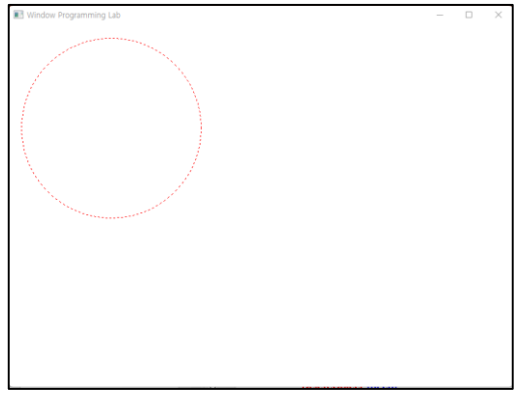
BOOL **DeleteObject** (HPEN pen);

- **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 브러시핸들을 만들어 주는 함수

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

- 그림을 그릴 화면인 디바이스컨텍스트에 브러시핸들을 등록

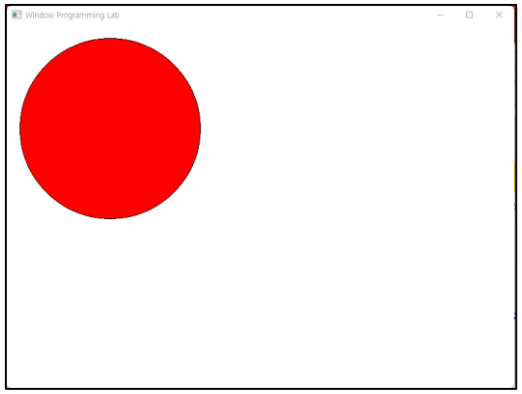
HBRUSH SelectObject (HDC hdc, HBRUSH hBrush);

- 그림 그리기를 마친 후 생성된 브러시 핸들은 삭제

BOOL DeleteObject (HBRUSH hBrush);

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

HGDIOBJ SelectObject (HDC hDC, HGDIOBJ hgdiobj);
- hDC: DC 핸들값
- hgdiobj: GDI의 객체
- 리턴 값은 원래의 오브젝트 값

- 생성한 객체 삭제 함수

BOOL DeleteObject (HGDIOBJ hgdiob);
- GDI오브젝트를 삭제하고 오브젝트가 사용하던 시스템 리소스를 해제
- 객체 생성 함수로 만들어진 GDI 오브젝트는 반드시 삭제해주어야 한다.
- hgdiobj: GDI의 객체

GDI 객체 핸들 구하기 함수들

- **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**

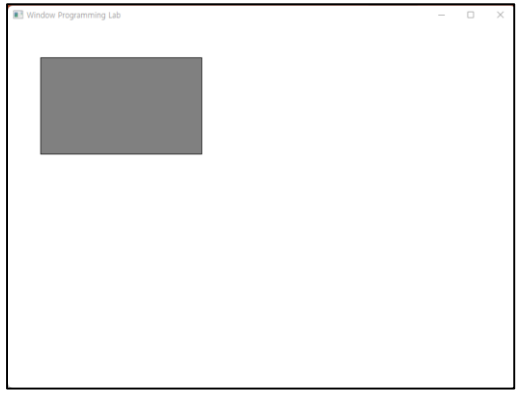
- **클라이언트의 영역을 취득하는 함수**

BOOL GetClientRect (HWND hWnd, LPRECT lprc);
- 클라이언트의 영역을 취득한다.
- hWnd: 윈도우 핸들
- lprc: 윈도우 영역의 좌표값

GDI 객체 핸들하기

- **윈도우가 제공하는 회색 브러시를 사용하여 사각형을 그리는 경우 → 객체를 만들지 않고 윈도우가 제공하는 객체(스톡 객체) 사용**

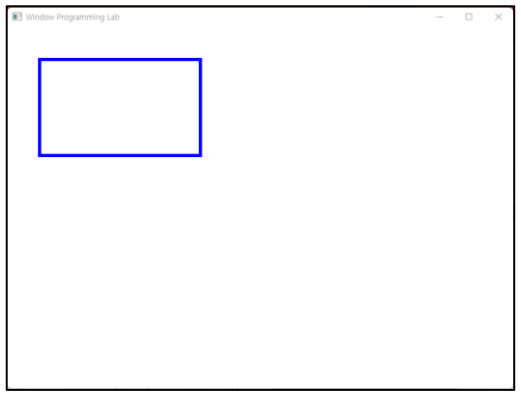
```
LRESULT CALLBACK wndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
	HDC hdc;
	HBRUSH hBrush, oldBrush;

	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 색상의 테두리를 가진 사각형을 그리는 경우 → 파란색의 펜 객체를 만들어 사용**

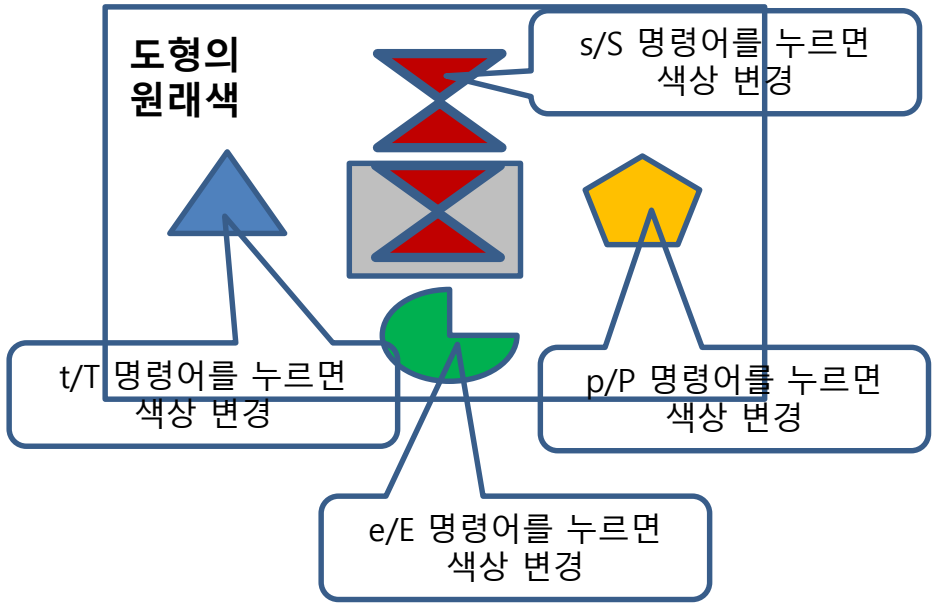
```
LRESULT CALLBACK wndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
      HDC hdc;
            HPEN hPen, oldPen;

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

               hPen = CreatePen (PS_SOLID, 5, RGB(0, 0, 255));   // 새로운 객체 만들기
               oldPen = (HPEN) SelectObject (hdc, hPen);
               Rectangle (hdc, 50, 50, 300, 200);
               SelectObject (hdc, oldPen);
               DeleteObject (hPen);

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

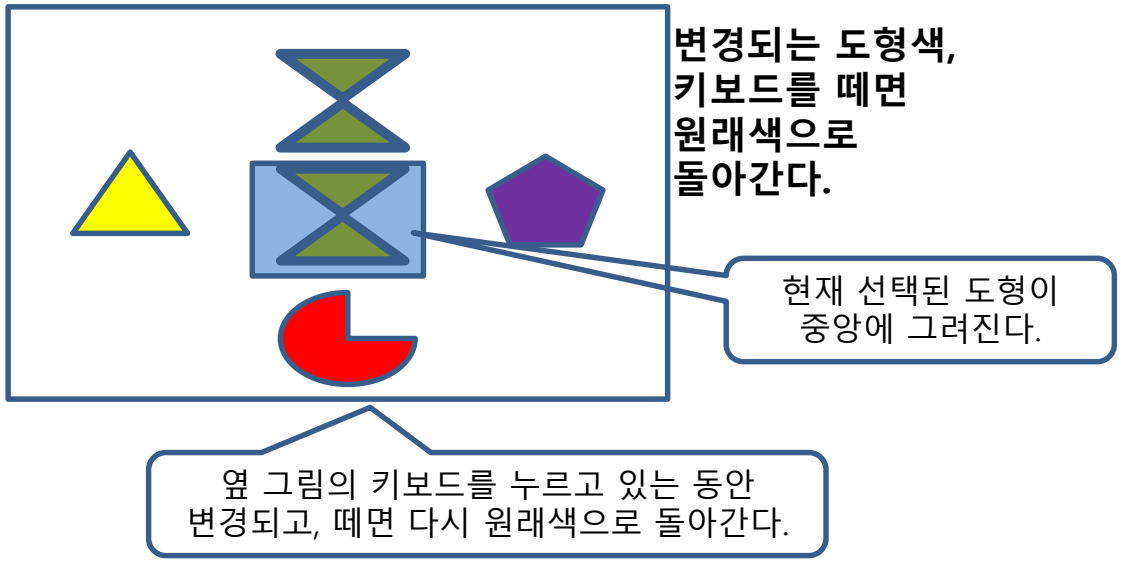


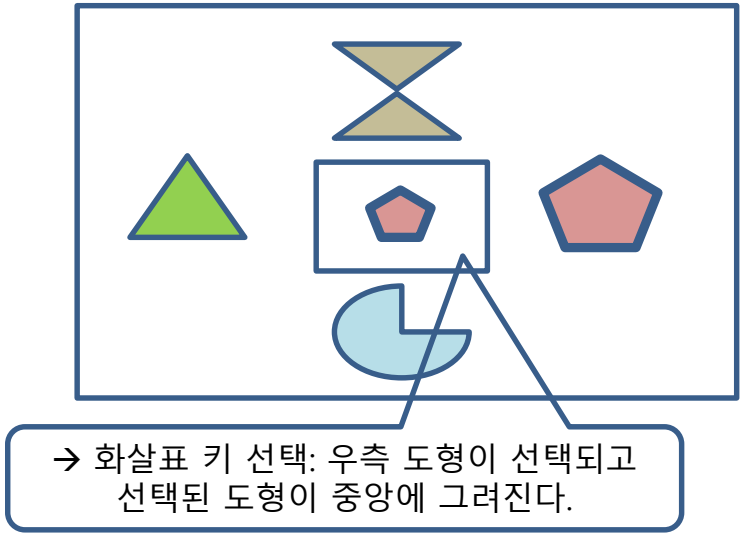
실습 2-8

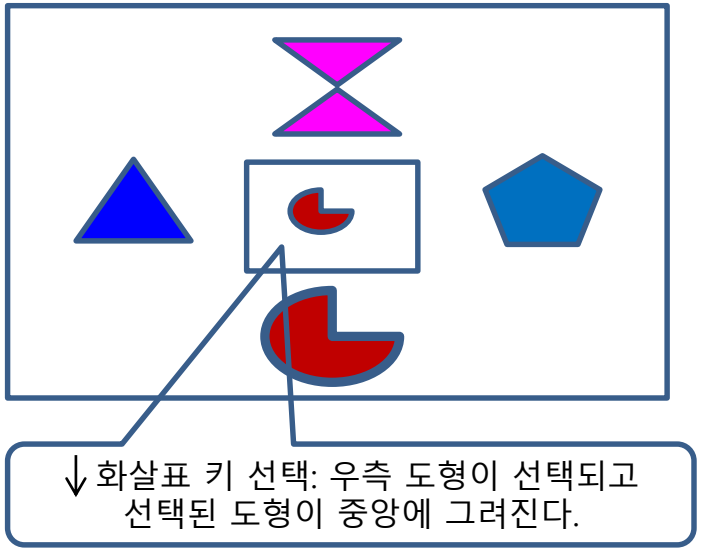
- **화면에 도형 그리고 색 입히기**
  - 윈도우를 띄운다.
  - 윈도우 중앙에 사각형을 그린다.
  - 사각형의 좌우상하에 각각 삼각형, 모래시계, 오각형, 파이 모양을 그린다.
  - 좌우상하 도형 중 한 개의 도형이 랜덤하게 선택되어 있다. 선택된 도형은 중앙의 사각형 안에 그려진다.
  - **키보드 명령어에 따라 각 도형의 색상을 랜덤하게 바꾸도록 한다.** 이때 키보드를 누르면 색이 변경되고 떼면 다시 원래의 색으로 돌아간다.
    - t/T: 삼각형의 색
    - s/S: 모래시계의 색
    - p/P: 오각형의 색
    - e/E: 파이 모양의 색
    - 색상이 변경되는 도형이 선택된 도형으로 그 도형은 역시 중앙의 사각형 안에 그려진다.
  - **키보드 명령어에 따라 도형이 선택되고, 선택된 도형의 크기가 작게 바뀌어 중앙의 사각형에 그려진다.**
    - Left/right/up/down: 화살표 키에 따라 좌우상하의 도형이 각각 선택된다.
    - 모든 명령어는 바꿀 수 있다.
  - q/Q: 프로그램 종료

실습 2-8


도형의
원래색
s/S 명령어를 누르면
색상 변경
t/T 명령어를 누르면
색상 변경
p/P 명령어를 누르면
색상 변경
e/E 명령어를 누르면
색상 변경

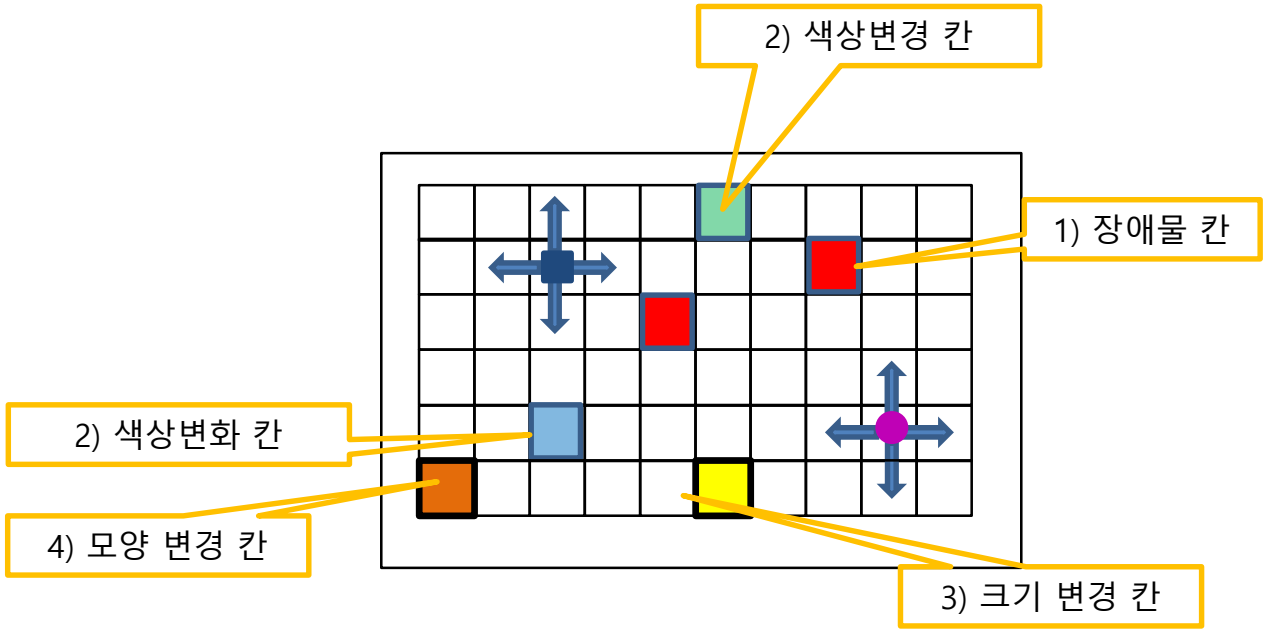

변경되는 도형색,
키보드를 떼면
원래색으로
돌아간다.
현재 선택된 도형이
중앙에 그려진다.
옆 그림의 키보드를 누르고 있는 동안
변경되고, 떼면 다시 원래색으로 돌아간다.


→ 화살표 키 선택: 우측 도형이 선택되고
선택된 도형이 중앙에 그려진다.


↓ 화살표 키 선택: 우측 도형이 선택되고
선택된 도형이 중앙에 그려진다.

실습 2-9

- **돌 이동하기**
  - 화면에 40X40 칸을 그린다.
  - 칸의 중간 중간에 네 종류의 다른 색의 칸들이 있다. (최소 20개)
    1) 빨간색 칸: 장애물 – 객체들이 통과할 수 없다.
    2) 색상 변경 칸 – 돌이 이 칸을 지나면 그 칸의 색으로 바뀐다. 색상변경 칸은 다양한 색으로 설정하고, 돌은 그 칸의 색으로 변경된다.
    3) 크기 변경 칸 – 돌의 크기가 변경된다. (확대 → 축소 → 확대 → 축소 → ...)
    4) 모양 변경 칸 – 돌의 모양이 변경된다. (네모 → 원, 원 → 세모 ...)
  - 좌상, 우하 위치에 두 개의 다른 색을 가진 돌을 그린다.
    - 다른 색으로 표시된 영역 이외의 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.
    - 색상변경칸: 색상이 랜덤하게 변한다.
    - 크기변경칸: 돌의 크기가 변한다.
    - 모양변경칸: 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 학습 내용

- **이번주 학습내용**
  - 도형 그리기
    - 선, 원, 사각형, 다각형
  - GDI 객체 다루기
    - 펜, 브러시 사용하기
    - 스톡 오브젝트 사용하기
  - 객체 생성 함수들, 객체 선택 함수들

실습 2-10

- **명령에 따라 그림을 그리는 프로그램**
  - 윈도우 화면 아랫단 중앙에 문자열 한 줄 입력 받을 수 있는 글상자 역할을 할 사각형을 배치
  - **입력값은 여섯 개의 숫자로**: **그릴 도형의 종류, 좌표 값 (x1, y1) (x2, y2), 도형 테두리 두께**가 입력된다.
    - 도형의 종류: 1 – 직선, 2 – 삼각형, 3 – 사각형, 4 – 원
  - 최대 10개의 도형을 입력받도록 한다.

  - 예를 들어,
    - 직선일 경우는 [1 10 10 200 150 2] → 직선을 (10, 10)에서 (200, 150)에 두께 2로 그린다.
    - 삼각형일 경우는 [2 0 0 100 200 2] → 삼각형을 (0, 0)과 (100, 200)에 2 두께로 그린다.
    - 사각형일 경우는 [3 0 0 100 200 1] → 사각형을 (0, 0)과 (100, 200) 에 1 두께로 그린다.
    - 원일 경우는 [4 0 0 50 50 3] → 원을 (0, 0)과 (50, 50)에 3 두께로 그린다.
  - **문자 명령어를 넣어서 그린 도형의 두께를 늘리거나 줄이도록 한다.**
    - **+ / -** : 두께를 1 ~ 10 사이로 늘리거나 줄인다. 두께 값이 1보다 작아지면 도형의 크기가 줄어들고, 두께 값이 10보다 커지면 도형의 크기가 늘어난다.
    - **화살표**: 도형의 위치를 좌우상하로 이동
    - **k**: 도형의 테두리 또는 내부의 색깔을 바꾼다.
    - **p**: 바로 전에 그린 도형을 다시 그린다. 즉, 그렸던 도형에 대한 정보들을 저장하도록 한다. (prior)
    - **n**: p 명령어에 의해 이전 도형을 그린 후 n을 입력하면 다음에 그렸던 도형을 다시 그린다. (next)

    - 도형 테두리 또는 내부의 색깔을 랜덤하게 설정한다.
    - 화면에 도형이 없는데 위의 명령어를 입력하면 **에러 메시지를** 출력한다.

  - 키보드 명령어 또는 도형 종류는 변경 가능함