제 2장 윈도우 기본 입출력 3

2024년 1학기 윈도우 프로그래밍

8. 직선, 원, 사각형, 다각형 그리기

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

점 그리기

- **윈도우에 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

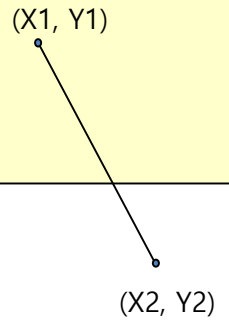
- **지정한 좌표에서 픽셀의 색상값 가져오기**

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

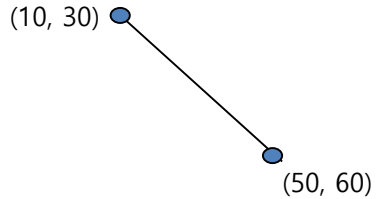
```
BOOL MoveToEx (HDC hDC, int X1, int Y1, LPPOINT lpPoint );
        - hDC: DC 핸들
        - X1, Y1: 직선의 시작점 (x와 y 좌표값)
        - lpPoint: 이전의 좌표, 사용 안함
```



- 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);
        LineTo (hDC, 50, 60);          //-- (10, 30) → (50, 60) 직선 그리기
        EndPaint (hWnd, &ps) ;
        return 0 ;
```

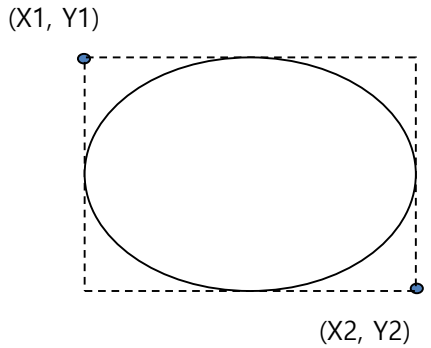



```
typedef struct tagPOINT {
        LONG x;
        LONG y;
} POINT;
```

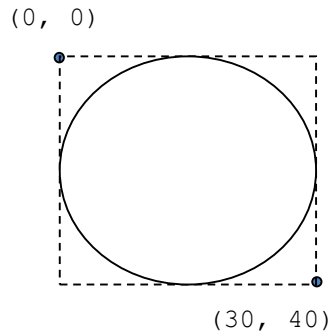
원 그리기

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

BOOL Ellipse (HDC hDC, int X1, int Y1, int X2, int Y2);
- hDC: DC 핸들
- X1, Y1: 좌측 상단 좌표값 (x와 y 최소값)
- X2, Y2: 우측 하단 좌표값 (x와 y 최대값)



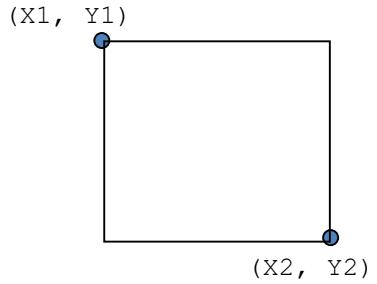

```
case WM_PAINT :
        hDC = BeginPaint (hWnd, &ps) ;
        Ellipse(hDC, 0, 0, 30, 40);          //-- 박스의 중심점 (15,20)
        EndPaint (hWnd, &ps) ;
        return 0 ;
```



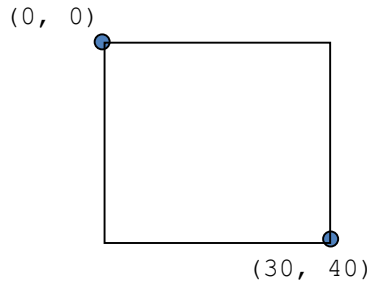
사각형 그리기

- 두 점의 좌표를 기준으로 수평수직 사각형을 그림

BOOL Rectangle (HDC hDC, int X1, int Y1, int X2, int Y2);
- hDC: DC 핸들
- X1, Y1: 좌측 상단 좌표값 (x와 y 최소값)
- X2, Y2: 우측 하단 좌표값 (x와 y 최대값)



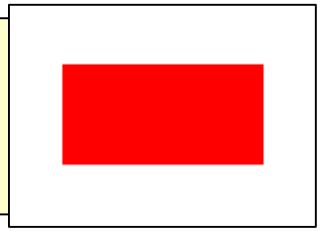

```
case WM_PAINT :
        hDC = BeginPaint (hWnd, &ps) ;
        Rectangle(hdc, 0, 0, 30, 40);      //-- 원을 둘러싼 사각형의 좌표
        EndPaint (hWnd, &ps) ;
        return 0 ;
```



사각형 그리기

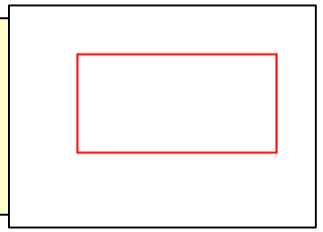
- **사각형 그리기 다른 함수들**
  - 내부가 채워진 사각형 그리기:

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



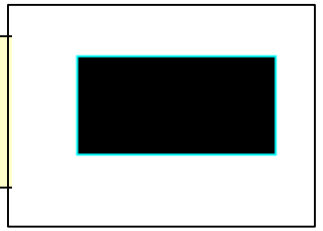
  - 외곽선 사각형 그리기:

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.

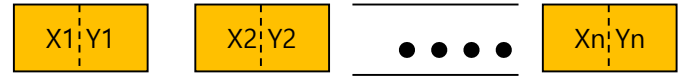
BOOL PtInRect (CONST RECT *lprc, POINT pt);
- 특정 좌표 pt가 lprc 영역 안에 있는지 검사한다.

다각형 그리기

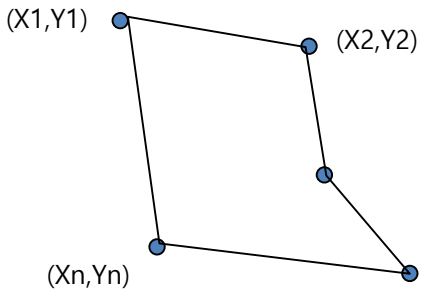
- 연속되는 여러 점의 좌표를 직선으로 연결하여 다각형을 그림

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

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



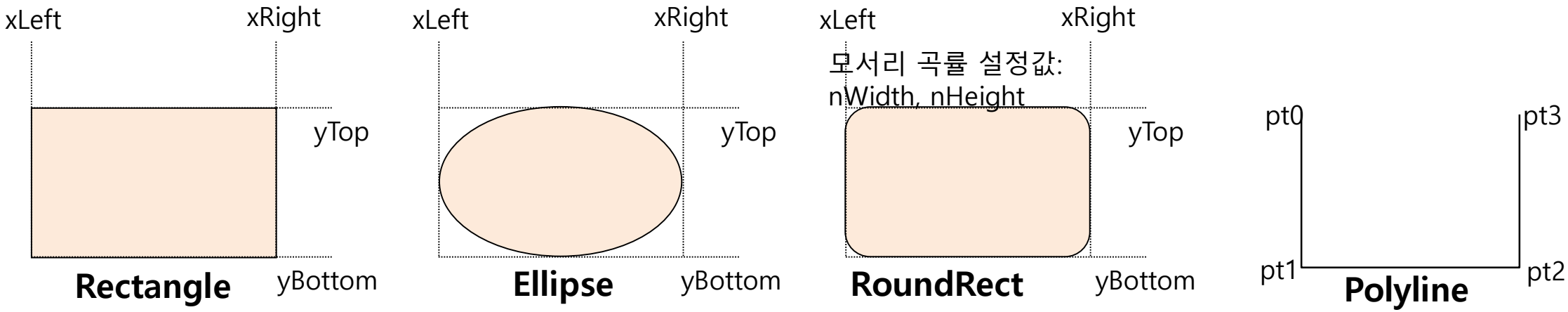

```
typedef struct tagPOINT {
        LONG x;
        LONG y;
} POINT;
```




```
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 ;
```

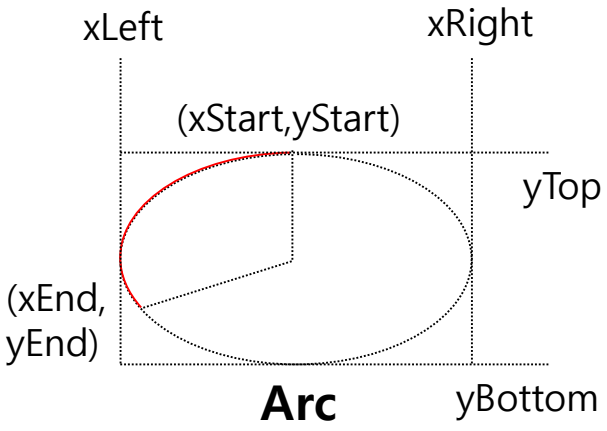
도형 그리기

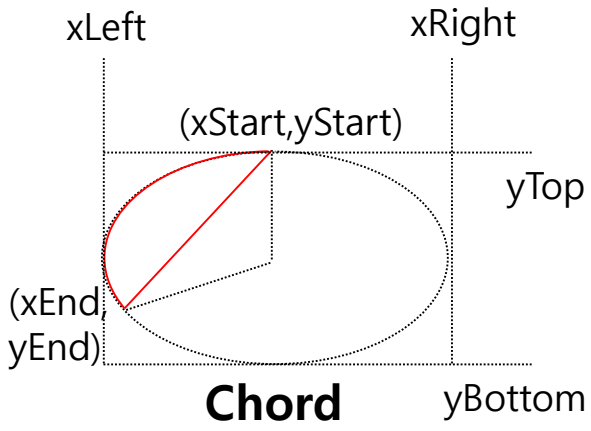
- **다양한 도형 그리기**
  - BOOL Rectangle (HDC hDC, int xLeft, int yTop, int xRight, int yBottom);
  - BOOL RoundRect (HDC hDC, int xLeft, int yTop, int xRight, int yBottom, int nWidth, int nHeight);
  - BOOL Polyline (HDC hDC, CONST POINT *lppt, int cPoints);

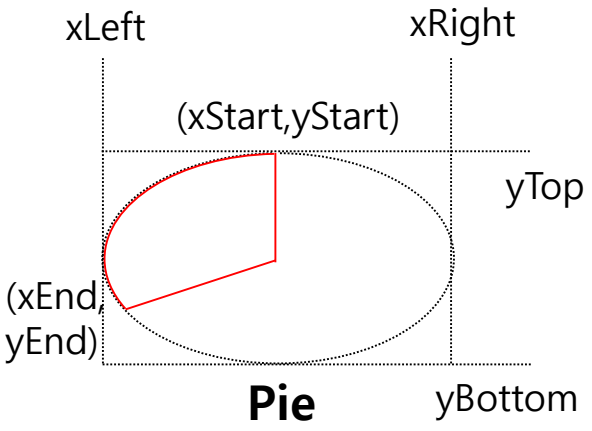


도형 그리기

- **다양한 도형 그리기**
  - BOOL Arc (HDC hDC, int xLeft, int yTop, int xRight, int yBottom, int xStart, int yStart, int xEnd, int yEnd);
  - BOOL Chord (HDC hDC, int xLeft, int yTop, int xRight, int yBottom, int xStart, int yStart, int xEnd, int yEnd);
  - BOOL Pie (HDC hDC, int xLeft, int yTop, int xRight, int yBottom, int xStart, int yStart, int xEnd, int yEnd);
    - 기본 그리기 방향은 시계 반대방향

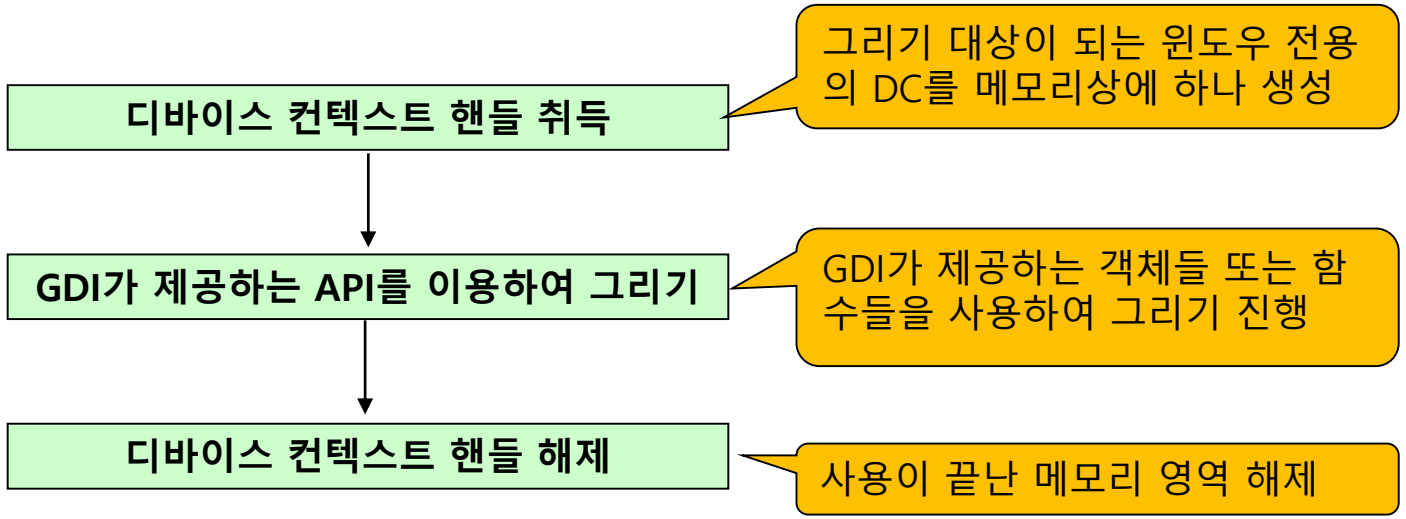






GDI 객체들

- **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

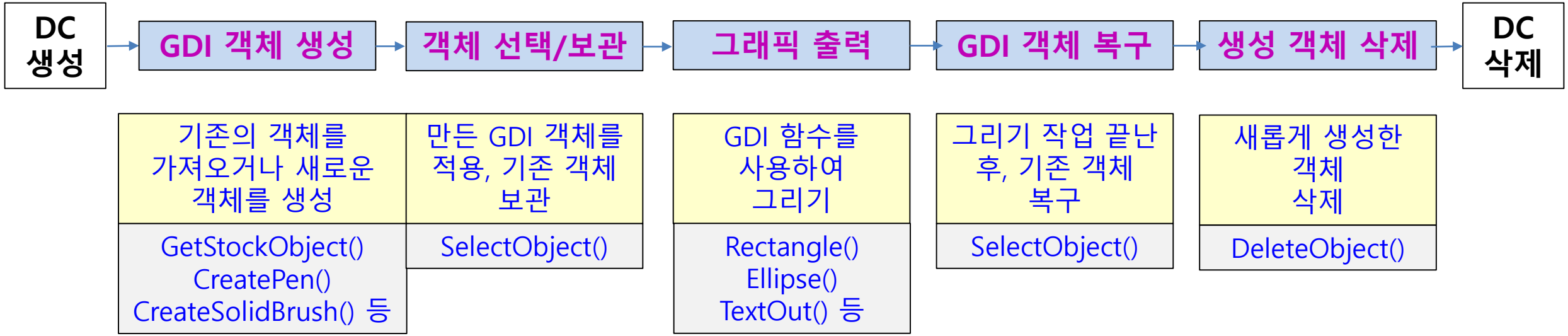


- **윈도우에서 기본적으로 제공하는 GDI객체인 스톡 객체는 따로 생성할 필요가 없어서 편리하지만 다양하게 그래픽을 하기에는 부족**
  - 다양한 출력을 위해서는 GDI객체를 만들어서 사용
  - GDI객체를 만드는 함수는 앞의 "Create"로 시작하는 함수

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

GDI 객체를 사용하는 방법



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**

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

→

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

- **그림 그리기를 마친 후 생성된 펜 핸들은 삭제 함수**

BOOL **DeleteObject** (HGDIOBJ hgdiob);

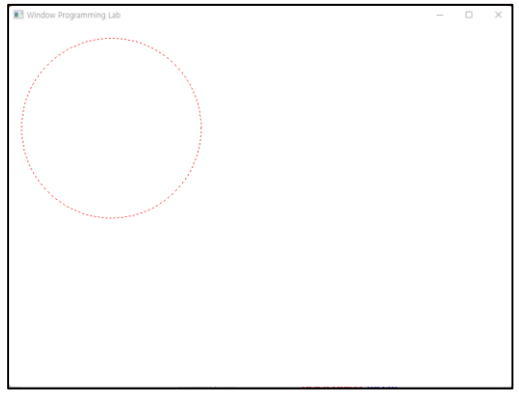
→

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 (Ellipse(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

HGDIOBJ **SelectObject** (HDC hDC, HGDIOBJ hgdiobj);
→
HBRUSH **SelectObject** (HDC hDC, HBRUSH hBrush);

- 그림 그리기를 마친 후 생성된 브러시 핸들은 삭제

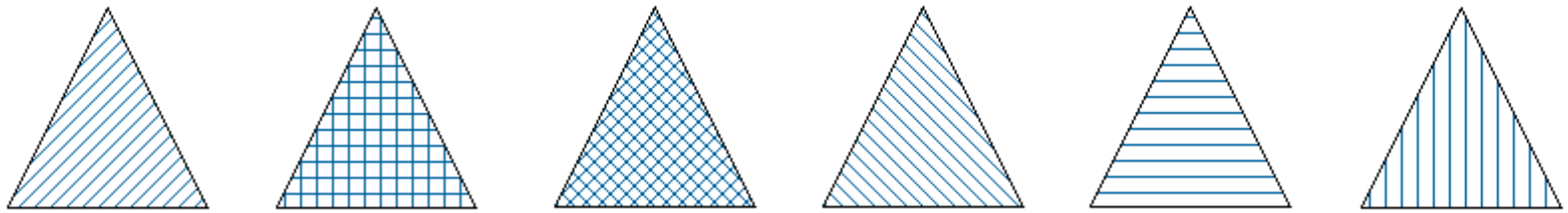
BOOL **DeleteObject** (HGDIOBJ hgdiob);
→
BOOL **DeleteObject** (HBRUSH hBrush);

면 색상 변경 (빗살무늬)

- 면에 빗살무늬와 색상정보를 가지는 브러시핸들을 만들어 주는 함수

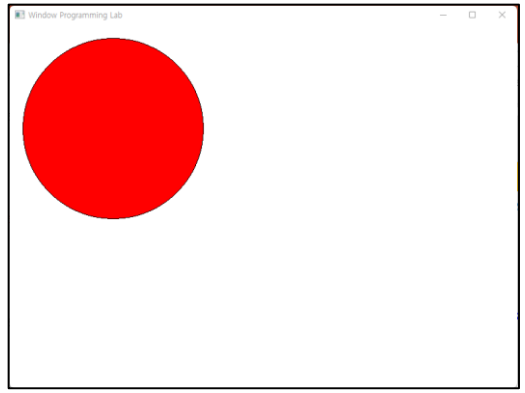
HBRUSH CreateHatchBrush (int iHatch, COLORREF crColor);

- 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 hgdiob);
- 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
  - **브러쉬 속성: BLACK_BRUSH / DKGRAY_BRUSH / DC_BRUSH / GRAY_BRUSH / HOLLOW_BRUSH / LTGRAY_BRUSH / NULL_BRUSH/ WHITE_BRUSH**
    - **DC_BRUSH: 단색 브러쉬, 기본색은 흰색**
    - **HOLLOW_BRUSH: 속이 빈 브러쉬 (NULL_BRUSH)**
  - **펜 속성: BLACK_PEN / DC_PEN / WHITE_PEN / NULL_PEN**
    - **DC_PEN: 단색 펜, 기본색은 검정색**
    - **NULL_PEN: 아무것도 그리지 않는다.**

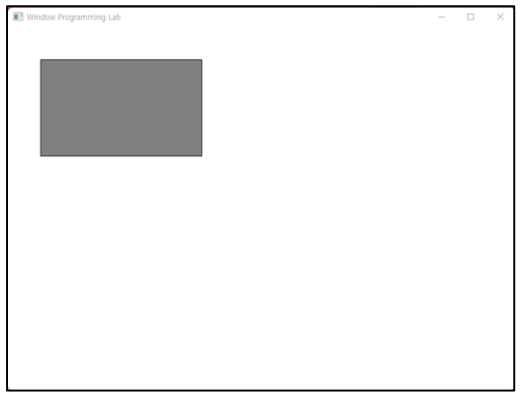
- **클라이언트의 영역을 취득하는 함수**

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

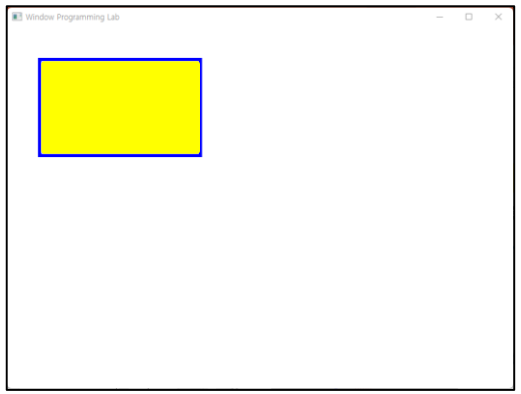
```
LRESULT CALLBACK WndProc (HWND hWnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    HDC hDC;
    HBRUSH hBrush, oldBrush;

    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);                     // 제자리 돌아가기

            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));         // 새로운 객체 만들기: 펜
                oldPen = (HPEN) SelectObject (hDC, hPen);
                hBrush = CreateSolidBrush (RGB(255, 255, 0));           // 새로운 객체 만들기: 브러쉬
                oldBrush = (HBRUSH) SelectObject (hDC, hBrush);

                Rectangle (hDC, 50, 50, 300, 200);

                SelectObject (hDC, oldPen);                             // 제자리 돌아가기
                SelectObject (hDC, oldBrush);
                DeleteObject (hPen);                                    // 새로운 객체 삭제
                DeleteObject (hBrush);

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



실습 2-7

- **실습 2-5를 이어 캐럿을 이용한 메모 만들기**
  - Caret이 있는 10줄까지 입력 받을 수 있는 메모장을 작성
  - 입력 받을 때 한 줄은 최대 30자 까지 저장 가능
  - 윈도우를 띄우면 좌측 상단에 캐럿이 깜빡이고 있다. 항상 캐럿의 위치에서 문자를 입력한다.
  - 다음의 명령을 수행한다. (이전의 엔터와 백스페이스키는 기본으로 추가하기)
    - 이스케이프키 (esc): 화면이 다 지워지고 캐럿은 맨 윗줄 앞에 있다..
    - 탭키 (tab): 5개의 스페이스가 삽입되고 캐럿도 5개 스페이스 뒤로 이동한다.
    - 화살표 키: 캐럿이 현재 위치에서 문자 기준으로 좌/우/상/하로 이동한다. 캐럿을 이동한 후 문자를 입력하면 그 위치에 입력된다. 문자 덮어쓰기가 디폴트이다. 한 줄은 최대 30자 까지만 입력된다.
    - F1 키: 입력하는 문자가 대문자로 출력된다. 다시 누르면 소문자로 출력된다.
    - Del 키: 현재 캐럿이 놓인 단어가 (스페이스로 구분) 삭제되며 뒤의 문자들이 앞으로 나온다. 캐럿의 위치는 바뀌지 않는다. 마지막 문자였고 그 단어가 삭제되었다면 캐럿은 새로운 마지막 문자의 맨 뒤로 이동한다.
    - Home 키: 캐럿이 그 줄의 맨 앞에 온다. 홈 키로 캐럿을 이동 후 문자를 입력하면 문자가 입력된다. 문자 덮어쓰기가 디폴트이다.
    - End 키: 캐럿이 놓인 그 줄의 맨 뒤로 캐럿이 움직인다. 움직인 위치에 새로운 문자를 넣을 수 있다. 80자가 다 찼다면 캐럿은 다음 줄로 이동하고 문자가 출력된다 (덮어쓰기 혹은 삽입은 선택 가능)
    - Insert 키: 토글키로 insert키를 누르면 현재 캐럿 위치에서 문자가 추가되고, 다시 누르면 문자를 덮어쓴다.

  - 문자는 항상 캐럿 위치에 추가 또는 덮어쓰기로 입력 된다.

  - 위의 명령키는 다른 키를 사용해도 무관하고, 라인수나 한 라인의 문자 수는 변경 가능하다.

실습 2-8

- **메모장에 명령어 추가하기**
  - f2 키: 모든 줄에서 앞에 4칸의 _을 넣는다. 다시 누르면 없어진다.
    - 예) This is line 1. → ___This is line 1.
  - f3 키: 모든 줄을 한 줄 아래로 출력한다. 만약 모든 줄이 차 있다면 밀리는 한 줄은 삭제된다.
    - 예) This is line 1.
      This is line 2.

      →

      <빈 라인>
      This is line 1.
      This is line 2.
  - f4 키: 모든 줄의 순서를 뒤바꾼다. (1, 2, 3, 4 → 4, 3, 2, 1)
    - 예) This is line 1. <캐럿>
      This is line 2.

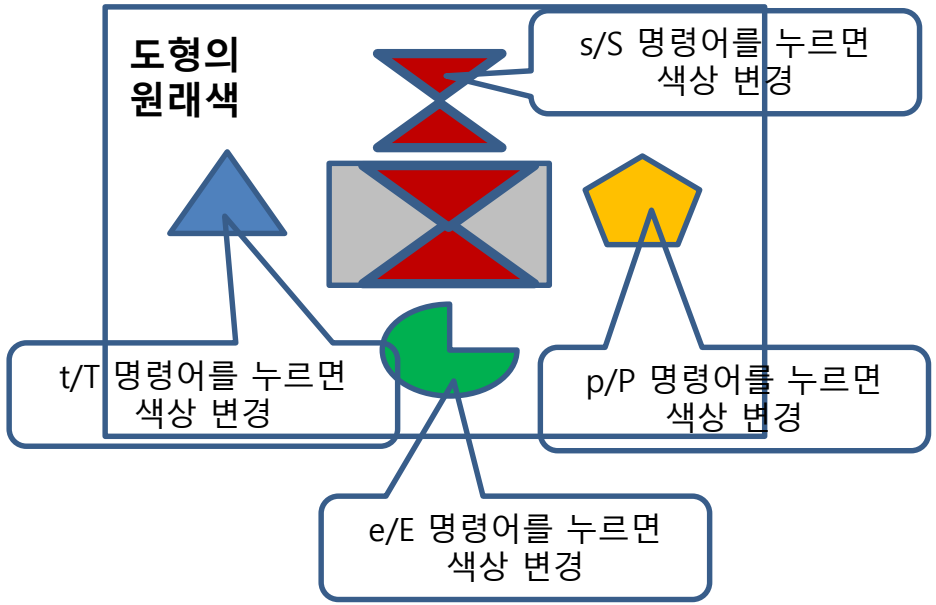
      →

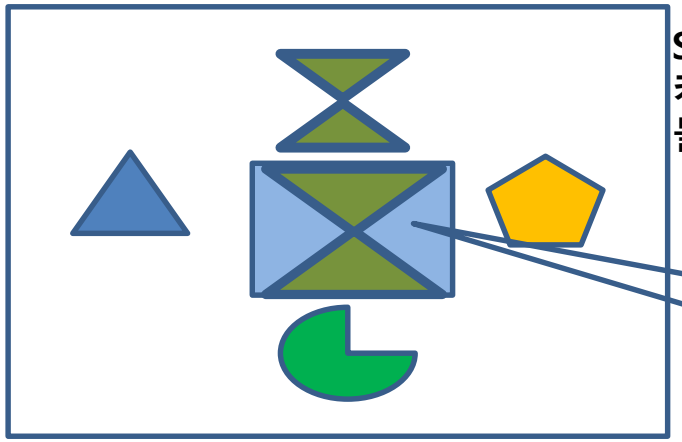
      This is line 2.
      This is line 1.
  - Page-down: 공백으로 구분되는 단어를 괄호가 둘러싸고 대문자로 출력한다. 다시 누르면 괄호가 삭제되고 대문자도 원래문자로 바뀐다..
    - 예) This is sample. → (THIS)(IS)(SAMPLE).
  - Page-up: 공백 문자를 모두 삭제한다. 다시 입력하면 다시 원래 위치에 공백이 생긴다. 다시 누르면 공백이 나타난다.
    - 예) This is sample. → Thisissample.
  - +: 입력되어 있는 모든 문자는 알파벳 순서에서 다음 문자로, 숫자는 다음 숫자로 바꾼다. (a→b, b→c, c→d, d→e … z→a, 1→2, 2→3,… 0→1)
    - 예) This is 1 2 3 4 5 number test. → Uijt jt 1 2 3 4 5 6 ovncfs uftu.
  - -: 입력되어 있는 모든 문자는 알파벳 순서에서 이전 문자로, 숫자도 이전 숫자로 바꾼다.
    - 예) Uijt jt 2 3 4 5 6 ovncfs uftu. → This is 1 2 3 4 5 number test.

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/크기 변경: 아래의 키보드 명령어에 따라 도형이 선택되고, 선택된 도형의 색상을 랜덤하게 바꾸고 중앙의 사각형 안에 해당 도형이 커진 크기로 그려진다.** 이때 키보드를 누르면 색이 변경되고 떼면 다시 원래의 색이 된다.
    - 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 위치 변경: 아래의 키보드 명령어에 따라 도형이 시계방향 또는 반시계 방향으로 위치가 바뀌고, 위칸에 있는 도형이 같은 크기로 중앙의 사각형 안에 그려진다.**
    - ←/→: 화살표 키에 따라 도형이 시계방향/반시계방향으로 위치가 바뀐다.각각 선택된다.
    - 모든 명령어는 다른 키보드로 바꿀 수 있다.
  - q/Q: 프로그램 종료

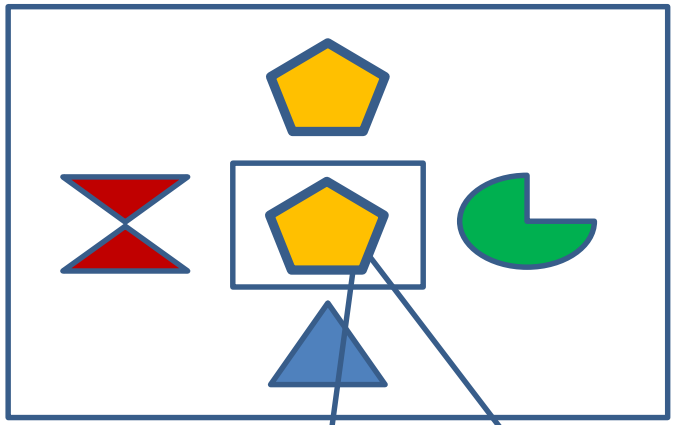
실습 2-9



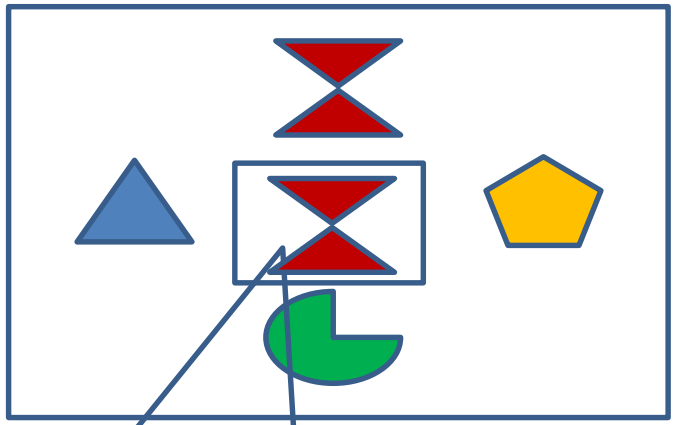


**S를 누르면: 도형색이 변경
키보드를 떼면 원래색으로
돌아간다.**

현재 선택된 도형이
중앙에 커진 크기로
그려진다.



→ 화살표 키 선택: 좌우상하의 도형이
반시계방향으로 위치를 바꾼다. 위칸에
있는 도형이 중앙에 그려진다.



← 화살표 키 선택: 좌우상하의 도형이
시계방향으로 위치를 바꾼다.

2장 학습 내용

- **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
    - 스톡 오브젝트 사용하기
  - 객체 생성 함수들, 객체 선택 함수들