

# 제3장 제어 메시지 처리하기

---

2014년 윈도우 프로그래밍

- **학습목표**

- 타이머를 이용해 자동으로 움직이는 형상을 원도우에 표현할 수 있다.
- 마우스에서 발생한 메시지를 이용할 수 있다.
- 래스터 연산을 이용해 그림의 일부만 삭제할 수 있다.

- **내용**

- 타이머 메시지
- 마우스 메시지
- 래스터 연산

# 윈도우의 크기를 측정

- 클라이언트 영역의 크기 측정

```
BOOL GetClientRect(
    HWND hwnd,
    LPRECT lpRect
);
```

- hwnd: 측정하기 원하는 윈도우의 핸들
  - lpRect: RECT 구조의 공간의 주소
- 
- WM\_SIZE**: 윈도우의 크기가 변경하면 발생하는 메시지
    - HIWORD(IParam): 윈도우의 높이
    - LOWORD(IParam): 윈도우의 폭
      - HIWORD: 32bit 데이터에서 상위 16bit 데이터를 구하기 위한 매크로 함수
      - LOWORD: 32bit 데이터에서 하위 16bit 데이터를 구하기 위한 매크로 함수

## 2절. 타이머 메시지

- 매 10초마다 알람을 하고 싶다면 어떻게 할까?
  - 10초마다 특정메시지를 발생시키고 그 메시지 처리부에서 알람 기능을 구현하면 된다.
- **SetTimer()**함수로 타이머를 설치했을 경우 지정한 시간 간격으로 이 메시지가 반복적으로 큐에 붙여진다.
- 다수의 타이머가 설치되어 있을 경우 각각의 타이머는 정해진 시간 간격으로 이 메시지를 큐에 저장하며 WM\_TIMER에서는 **wParam**값으로 어떤 타이머에 의해 이 메시지가 발생했는지 조사한다.

## 2절. 타이머 메시지

- 이 메시지는 다른 메시지들에 비해 **우선순위가 낮게 설정되어** 있기 때문에 먼저 처리해야 할 메시지가 있을 경우 곧바로 윈도우 프로시저로 보내지지 않을 수 있다. 따라서 정확한 시간에 이 메시지가 전달되지 않는 경우도 있으므로 정확도를 요하는 작업에는 이 메시지를 사용하지 않는 것이 좋다.

- 정확도를 요하는 작업에는 타이머 콜백 함수를 지정**한다. 타이머 콜백 함수를 지정했을 경우는 이 메시지부를 수행하는 것이 아니라, 프로그래머가 만든 함수(타이머 콜백 함수)를 OS가 자동으로 주기적으로 호출해 준다.

- WM\_TIMER 메시지에서

- wParam**에는 **타이머의 ID**가 전달된다. 이 ID는 SetTimer()함수의 두번째 인자로 지정한 값으로 여러 개의 타이머를 구분하기 위한 것

- lParam**에는 **타이머 콜백 함수**를 사용할 경우 콜백 함수명

## 2절. 타이머 메시지

- 타이머를 설치하면 지정한 시간간격으로 WM\_TIMER메시지 유형이 발생하며, wParam에는 타이머 ID가, lParam에는 타이머 콜백함수명이 전달된다.
- 타이머 설정함수  
 WORD SetTimer (HWND hWnd, UINT\_PTR nIDEvent,  
                   UINT uElapse, TIMERPROC lpTimerFunc);
- 매개변수
  - hWnd : 윈도우 핸들
  - nIDEvent : 타이머 ID, 여러 개의 타이머를 구분하기 위한 정수
  - uElapse : 시간간격 milisec(1000분의 1초)
  - lpTimerFunc : 시간간격 마다 수행할 함수(NULL이라고 쓰면 WndProc()가 타이머메시지를 처리)

# 타이머

## • 처리방법

```

LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    static int Timer1Count=0, Timer2Count=0;

    switch (iMsg) // 메시지 번호
    {
        case WM_CREATE:
            SetTimer (hwnd, 1, 70, NULL);           // 1번 아이디를 가진 타이머: 0.07초 간격
            SetTimer (hwnd, 2, 100, NULL);          // 2번 아이디를 가진 타이머: 0.1초 간격
            break;

        case WM_TIMER:
            switch(wParam) {
                case 1: // 0.07초 간격으로 실행
                    Timer1Count++;
                    break;
                case 2: // 0.1초 간격으로 실행
                    Timer2Count++;
                    break;
            }
            }

    return DefWindowProc (hwnd, iMsg, wParam, lParam);
    // CASE에서 정의되지 않은 메시지는 커널이 처리하도록 메시지 전달
}

```


# 타이머

- 타이머 콜백 함수 이용

```

LRESULT CALLBACK WndProc (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch ( uMsg ){
        case WM_CREATE:
            SetTimer (hWnd, 1, 500, TimerProc);
            break;
    }
    return 0;
}

```



```

void CALLBACK TimerProc (HWND hWnd, UINT uMsg, UINT idEvent, DWORD dwTime)    //0.5초 마다 실행
{
    HDC hdc;
    hdc = GetDC(hWnd);
    GetClientRect(hWnd, &rect);

    MyBrush = CreateSolidBrush( RGB(rand()%255, rand()%255, rand()%255) );
    MyPen = CreatePen(PS_SOLID, rand()%5, RGB(rand()%255, rand()%255, rand()%255) );
    OldBrush = (HBRUSH)SelectObject(hdc, MyBrush);
    OldPen = (HPEN)SelectObject(hdc, MyPen);
    Ellipse(hdc, rand()%(rect.right), rand()%(rect.bottom),
            rand()%(rect.right), rand()%(rect.bottom) );

    SelectObject(hdc, OldBrush);
    SelectObject(hdc, OldPen);
    DeleteObject(MyBrush);
    DeleteObject(MyPen);

    ReleaseDC(hWnd, hdc);
}

```



# 타이머

- 타이머 콜백 함수

- SetTimer 함수의 마지막 인자로 설정

```
VOID CALLBACK TimerProc(  
    HWND hwnd,  
    UINT uMsg,  
    UINT_PTR idEvent,  
    DWORD dwTime  
);
```

- *hWnd*: 타이머를 소유한 윈도우 핸들
- *uMsg*: WM\_TIMER 메시지
- *idEvent*: 타이머 id
- *dwTime*: 윈도우가 실행된 후의 경과시간

## 3-3 원 자동으로 이동하기

```

case WM_KEYDOWN:
    if (wParam == VK_RIGHT) // 오른쪽 키를 누를 때
        SetTimer(hwnd, 1, 70, NULL); // 타이머 설정
    break;

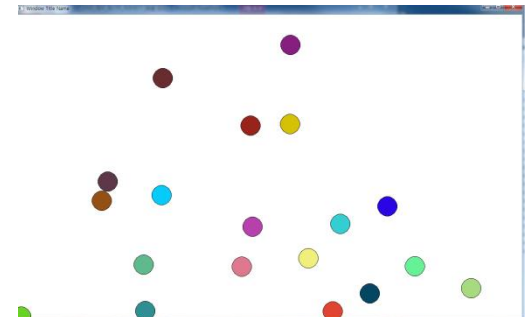
case WM_TIMER: // 시간이 경과하면 메시지 자동 생성
    x += 40;
    if (x + 20 > rectView.right) x -= 40;
    InvalidateRect (hwnd, NULL, FALSE);
    break;

case WM_DESTROY :
    KillTimer (hwnd, 1); // 윈도우 종료시 타이머도 종료
    PostQuitMessage (0) ;
    return 0 ;
}

```

## 연습문제 3-1

- 제목
  - 타이머에 따라 도형 그리기
- 내용
  - 타이머를 설정한다.
  - 타이머가 불리면 도형이 그려진다.
    - 도형의 모양은 원 또는 사각형
    - 도형의 크기는 임의의 크기로 설정한다. 도형 내부의 색은 임의로 바뀐다.
    - 도형은 위치는 임의의 위치에 그려진다.
    - 도형의 이동은 임의의 방향으로 진행된다. 가장자리에 도달하면 없어진다.
  - 도형이 3의 배수의 개수만큼 그려질 때마다 타이머의 시간이 바뀐다.
    - 1, 2, 3개: 1초
    - 4, 5, 6개: 0.5초
    - 7, 8, 9개: 0.3초
    - 10, 11, 12개: 0.5초
    - 13, 14, 15개: 1초
    - 16, 17, 18개: 0.5초
    - 19, 20, 21개: 0.3초
    - 22, 23, 24개: 0.5초
    - ...



## 연습문제 3-2

- 제목

- 애벌레 이동 프로그램

- 내용

- 윈도우 화면에 애벌레 역할을 할 두 개의 원을 나타낸다. *애벌레는 기본적으로 왼쪽에서 오른쪽으로, 오른쪽에서 왼쪽으로 계속 이동하고 있다.*
  - **모양:** 원으로 표시하고 머리와 꼬리 원은 다른 색으로 설정한다.  
**방향:** 애벌레는 왼쪽에서 오른쪽으로 이동하고 있고, 키보드 방향 키보드 입력에 따라 좌우상하로 방향을 바꿔 계속 이동한다. 가장자리에 도달하면 반대 방향으로 방향을 바꿔 계속 이동한다.
  - **속도:** ‘+’ 를 입력하면 속도가 점점 빨라지고, ‘-’ 를 입력하면 속도가 점점 느려진다.
  - **점프:** 특정 키를 누르면 그 자리에서 점프하도록 한다.
  - 타이머 함수를 사용한다.

## 연습문제 3-3

## 3절. 마우스 메시지

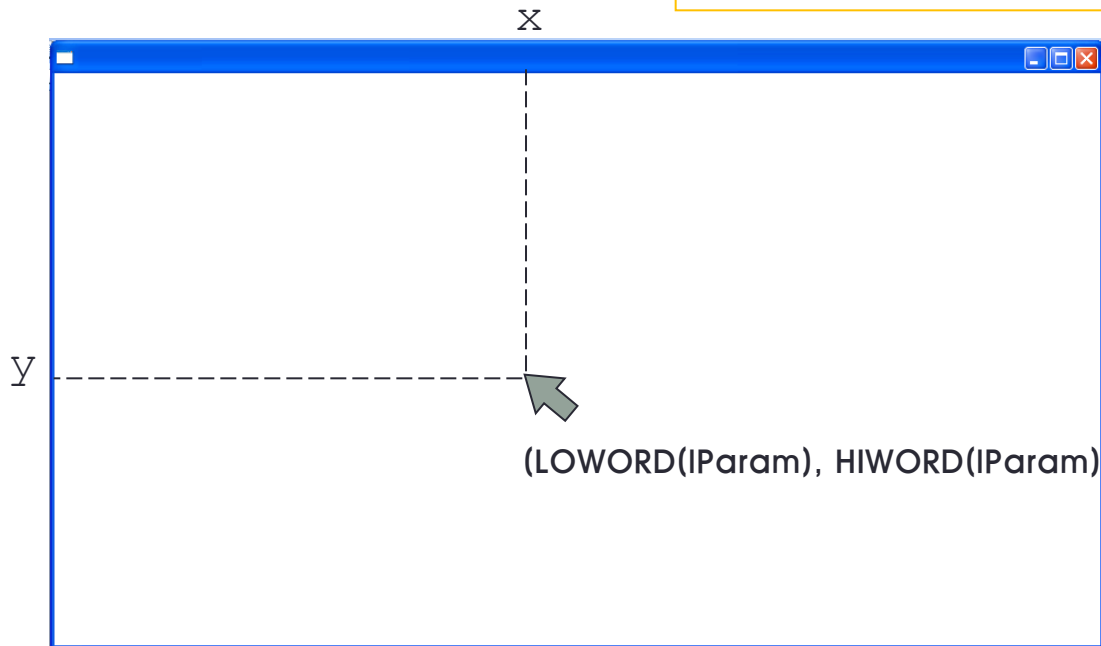
- **WM\_LBUTTONDOWN**
  - 왼쪽 마우스 버튼을 눌렀을 때 발생하는 메시지
- **WM\_LBUTTONUP**
  - 왼쪽 마우스 버튼을 떼었을 때 발생하는 메시지
- **WM\_RBUTTONDOWN**
  - 오른쪽 마우스 버튼을 눌렀을 때 발생하는 메시지
- **WM\_RBUTTONUP**
  - 오른쪽 마우스 버튼을 떼었을 때 발생하는 메시지
- **WM\_MOUSEMOVE**
  - 마우스를 움직일 때 발생하는 메시지

# 마우스 좌표 구하기

- 마우스에 대한 데이터 값은 IParam 에 저장

- `int y = HIWORD (IParam)`
- `int x = LOWORD (IParam)`

- HIWORD: 32bit 데이터에서 상위 16bit 데이터를 구하기 위한 매크로 함수
- LOWORD: 32bit 데이터에서 하위 16bit 데이터를 구하기 위한 매크로 함수



## 3-4 마우스로 원 선택하기

```
static int x, y;
static BOOL Selection;
int mx, my;

switch (iMsg)
{
case WM_CREATE :
    x = 50;      y = 50;
    Selection = FALSE;  // 원이 선택되었나, FALSE : 아직 안되었음
    return 0 ;

case WM_PAINT :
    hdc = BeginPaint (hwnd, &ps) ;
    // 만약 원이 선택되었다면, 4각형을 그린다. 아니면 원만 그린다.
    if (Selection)
        Rectangle(hdc, x-BSIZE, y-BSIZE, x+BSIZE, y+BSIZE);
    Ellipse(hdc, x-BSIZE, y-BSIZE, x+BSIZE, y+BSIZE);
    EndPaint (hwnd, &ps) ;
    return 0 ;
```



## 마우스로 원 선택하기(계속)

```
case WM_LBUTTONDOWN : // 왼쪽 버튼 누르면
    mx = LOWORD(IParam);
    my = HIWORD(IParam);
    if (InCircle(x, y, mx, my)) // 원의 중심점, 마우스 좌표 비교
        Selection = TRUE; // 원 안에 있으면 '참'
    InvalidateRgn(hwnd, NULL, TRUE);
    break;

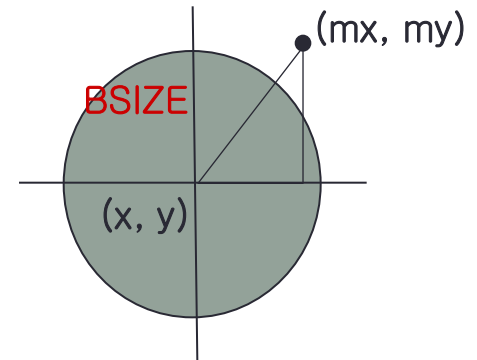
case WM_LBUTTONUP : // 왼쪽 버튼을 놓으면
    Selection = FALSE;
    InvalidateRgn (hwnd, NULL, TRUE);
    break;
```

## 마우스로 원 선택하기(계속)

```
#include <math.h>
#define BSIZE 40    // 반지름

float LengthPts (int x1, int y1, int x2, int y2)
{
    return(sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1)));
}

BOOL InCircle (int x, int y, int mx, int my)
{
    if(LengthPts(x, y, mx, my) < BSIZE) return TRUE;
    else return FALSE;
}
```



## 3-5 마우스 드래그로 원 이동하기

```

case WM_LBUTTONDOWN :
    mx = LOWORD(IParam);
    my = HIWORD(IParam);
    if (InCircle(x, y, mx, my))
        Selection = TRUE;           // mx, my : 마우스 좌표
    InvalidateRgn(hwnd, NULL, TRUE);
    break;

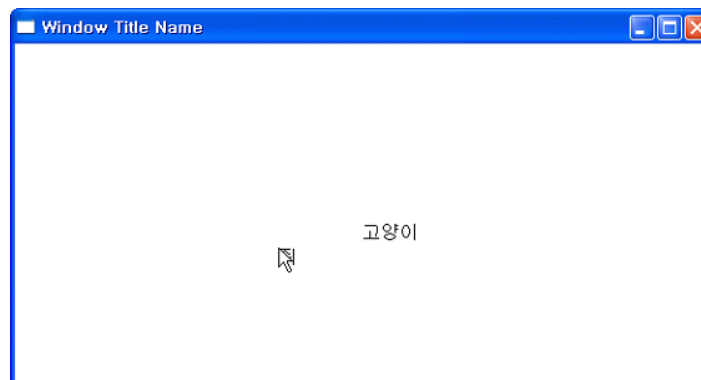
case WM_LBUTTONUP :
    InvalidateRgn(hwnd, NULL, TRUE);
    Selection = FALSE;
    break;

case WM_MOUSEMOVE:
    mx = LOWORD(IParam);
    my = HIWORD(IParam);
    if (Selection)
    {
        x = mx;
        y = my;
        InvalidateRgn(hwnd, NULL, TRUE); // 원과 사격형 그리기
    }
    break;

```

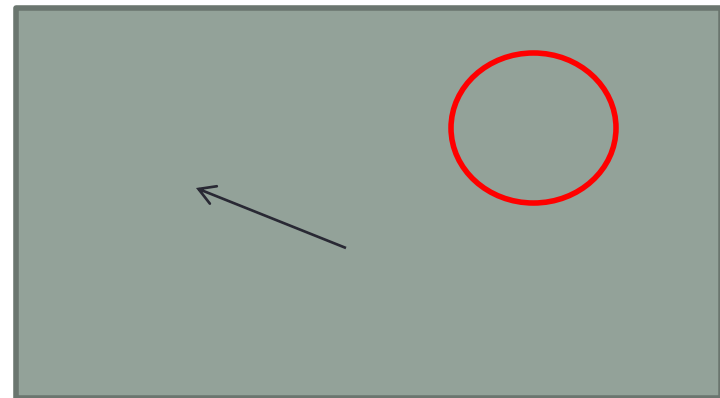
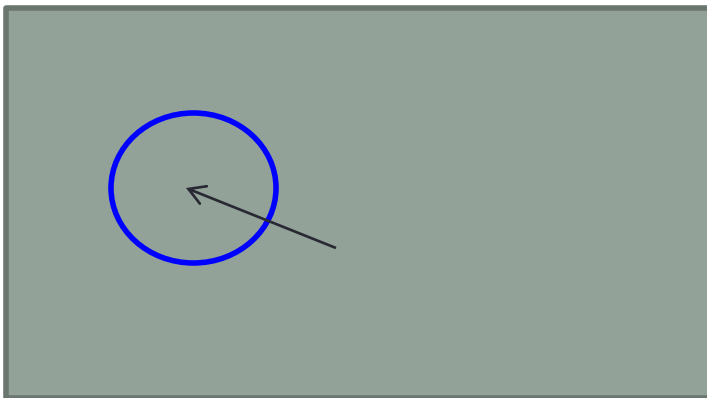
## 연습문제 3-4

- 제목
  - 쥐따라 가는 고양이
- 내용
  - 마우스의 왼쪽 버튼을 누르면 마우스 커서 위치에 “쥐” 라는 말을 출력하고 버튼을 떼면 “쥐” 라는 말이 사라진다.
  - “쥐” 가 화면상에 나타나면 “고양이” 는 “쥐” 를 잡기 위하여 움직이기 시작한다.



## 연습문제 3-5

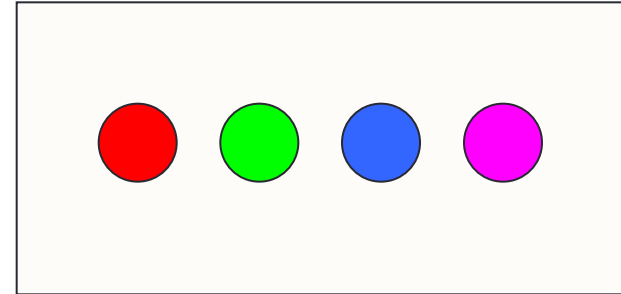
- 제목
  - 마우스 피하는 도형 그리기
- 내용
  - 임의의 위치에 10개의 도형을 그린다. (원, 사각형 또는 다각형)
  - 사용자가 마우스를 클릭하는데, 도형 내부에 클릭하면 선택된 도형은 사라지고 새로운 임의의 위치에 그려진다.
  - 새롭게 그려진 도형의 내부는 새로운 색으로 그려진다. (도형 내부 혹은 도형의 외곽선의 색을 변경)
  - 일정 시간이 지나면 도형들의 위치가 바뀌어져서 그려진다.



## 연습문제 3-6

- 제목

- 신호등 만들기



- 내용

- 화면에 원 4개를 그린다.
- 마우스가 원 내부를 마우스로 클릭하면 원에 색을 넣는다.
  - 첫 번째 원에 빨강색 / 흰색
  - 두 번째 원에 초록색 / 흰색
  - 세 번째 원에 파랑색 / 흰색
  - 네 번째 원에 임의의 색 (random 색) / 흰색
- 마우스가 원 내부를 다시 클릭하면 흰색으로, 다시 클릭하면 각각의 색으로 다시 칠하게 하시오.
  - 빨간색 -> 흰색 -> 빨간색 -> ...
- 키보드를 이용하여 똑같은 실행을 하도록 한다.
  - r/R: 첫 번째 원에 빨강색 / 흰색
  - g/G: 두 번째 원에 초록색 / 흰색
  - b/B: 세 번째 원에 파랑색 / 흰색
  - a/A: 네 번째 원에 임의의 색 (random 색) / 흰색

## 연습문제 3-7

## 4절. 래스터 연산

- WM\_PAINT 나 GetDC() 사용
  - 다시 그리기를 위해서는 WM\_PAINT 메시지를 처리해야 한다.
  - 배경이나 윈도우가 정적인 상태인 경우: WM\_PAINT에서 처리한다.
  - 움직이거나 동적으로 표현되는 상태인 경우: 재 출력할 필요 없이 GetDC()로 즉각 출력한다.
- 선을 그릴 때 마우스를 드래그하면, 이전의 선을 지우고 새로운 선을 그려야 한다.
  - Raster Operation (Bitwise Boolean 연산)
  - Raster: 이미지를 점들의 패턴으로 표현하는 방식 (cf. Vector)



## 4절. 래스터 연산

- 그리기 모드에서
  - R2\_COPYPEN;
    - 펜이나 브러쉬의 default 동작
    - 바탕색은 무시하고, 그리고자 하는 색을 보여 줌
  - R2\_XORPEN;
    - 바탕색과 그리는 색 사이의 XOR 연산을 수행
    - XOR 연산 : 두 개의 비트가 다를 때만 true(1), 같으면 false(0)

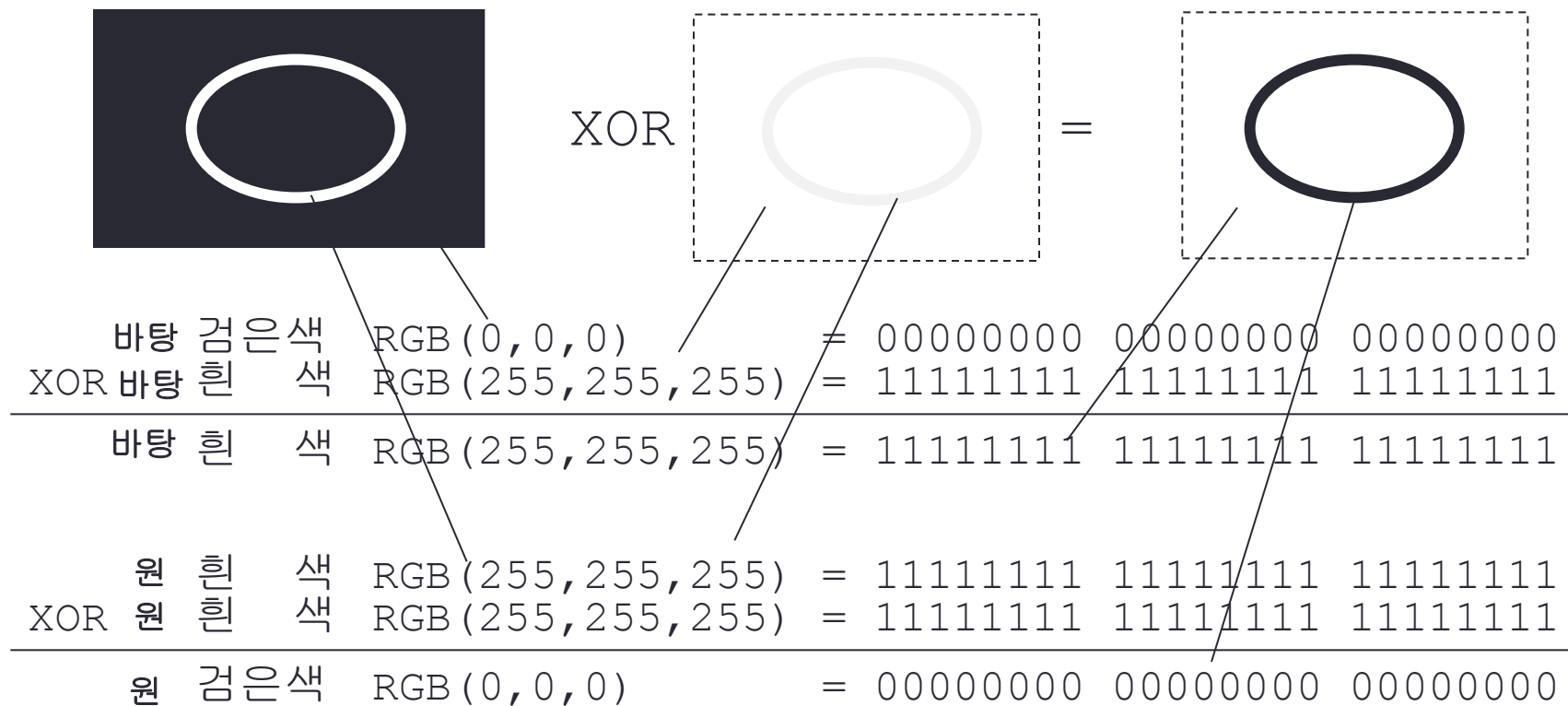
## 4절. 래스터 연산

- `int SetROP2 (HDC hdc, int fnDrawMode);`
  - 두 픽셀 사이에 bit 연산을 수행하도록 mix 모드를 설정할 수 있는 기능
    - `hdc`: 디바이스 컨텍스트 핸들      `fnDrawMode`: 그리기 모드

MIX MODE	의미
R2_BLACK	픽셀은 항상 0(검정색)이 된다
R2_COPYPEN	픽셀은 사용된 펜의 색상으로 칠해진다
R2_MASKNOTPEN	펜의 색상을 반전시켜 배경과 AND 연산한다
R2_MASKPEN	펜의 색상과 배경을 AND 시킨다
R2_MASKPENNOT	배경색을 반전시켜 배경과 OR 연산한다
R2_MERGEPEN	펜의 색상과 배경을 OR 시킨다
R2_MERGEPENNOT	배경색을 반전시켜 펜의 색상과 OR 연산한다
R2_NOP	픽셀은 아무런 영향을 받지 않는다
R2_NOT	배경색을 반전시킨다
R2_NOTCOPYPEN	펜의 색상을 반전시켜 칠한다
R2_NOTMASKPEN	R2_MASKPEN의 반전효과
R2_NOTMERGEPEN	R2_MERGEPEN의 반전효과
R2_NOTXORPEN	R2_XORPEN의 반전효과
R2_WHITE	픽셀은 항상 1(흰색)이 된다
R2_XORPEN	펜의 색상과 배경을 XOR 시킨다

# 래스터 연산으로 지우기

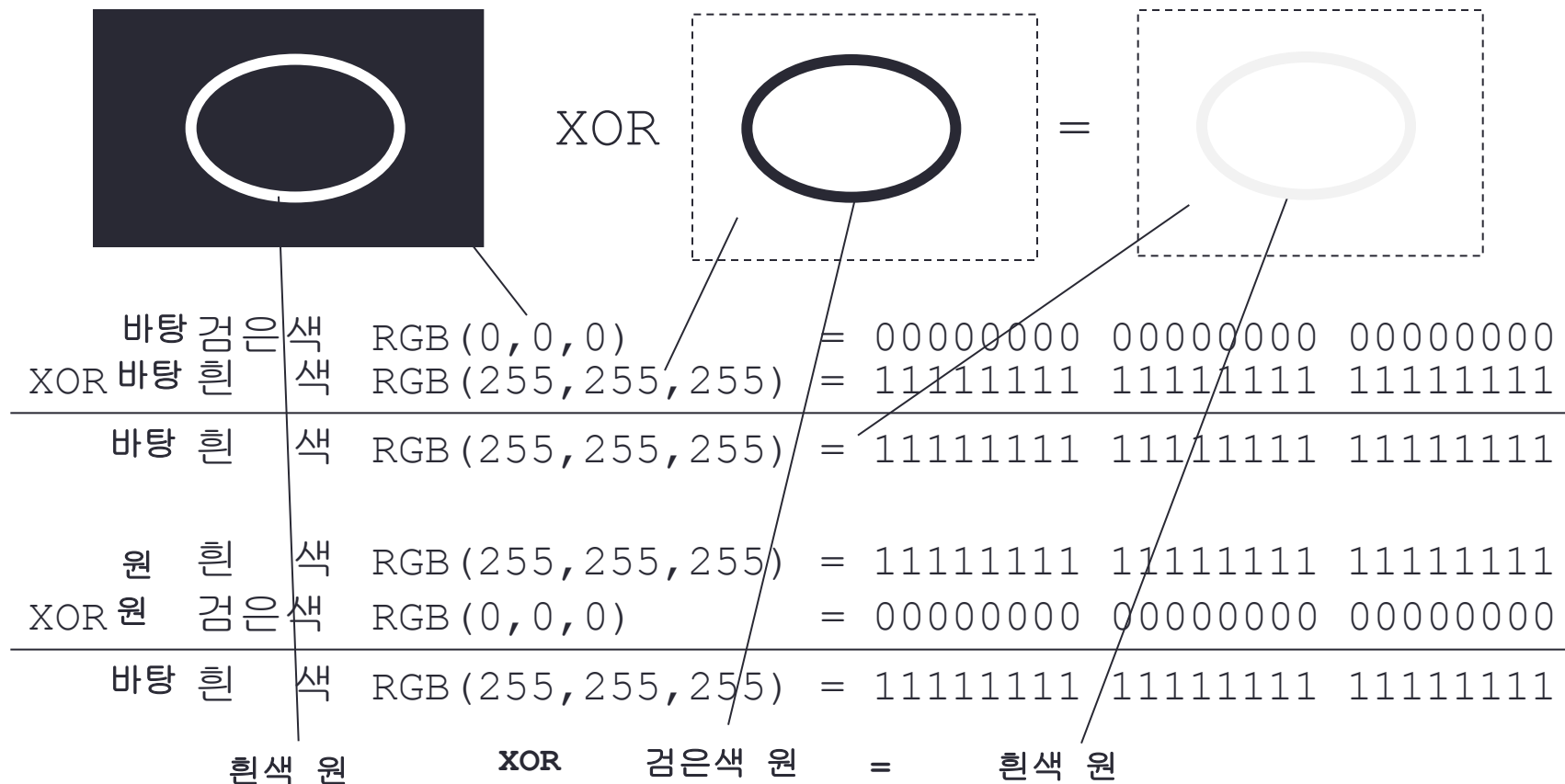
- SetRop2(hdc,R2\_XORPEN);



검은색 버튼 XOR 흰색 버튼 = 흰색 버튼  
 흰색 원 XOR 흰색 원 = 검은색 원

# 래스터 연산으로 지우기

- SetRop2(hdc,R2\_XORPEN);



## 3-6 고무줄 효과가 있는 직선그리기

```

static int startX, startY, oldX, oldY;
static BOOL Drag;
int endX, endY;
switch (iMsg)
{
case WM_CREATE :
    startX = oldX = 50;    startY = oldY = 50;    // 시작 좌표
    Drag = FALSE;
    return 0 ;
case WM_PAINT :
    hdc = BeginPaint (hwnd, &ps) ;
    MoveToEx(hdc, startX, startY, NULL);    // 이동하고 선으로 연결
    LineTo(hdc, endX, endY);
    EndPaint (hwnd, &ps) ;
    return 0 ;
case WM_LBUTTONDOWN :    // 버튼을 누르면 드래그 동작 시작
    Drag = TRUE;
    break;
case WM_LBUTTONUP :    // 버튼을 놓으면 드래그 종료
    Drag = FALSE;
    break;
}

```

# 고무줄 효과가 있는 직선그리기(계속)

```

case WM_MOUSEMOVE:
    hdc = GetDC(hwnd);
    if (Drag)
    {
        // 흰 바탕
        SetROP2(hdc, R2_XORPEN); // 펜의 XOR 연산
        SelectObject(hdc, (HPEN)GetStockObject(WHITE_PEN)); // 흰 펜
        // 흰 바탕 XOR 흰 펜 = 검은색 펜

        endX = LOWORD(IParam);
        endY = HIWORD(IParam);

        MoveToEx(hdc, startX, startY, NULL);
        LineTo(hdc, oldX, oldY); // 지우기 : 흰 바탕 XOR 검은 펜 = 흰 선

        MoveToEx(hdc, startX, startY, NULL);
        LineTo(hdc, endX, endY); // 그리기 : 흰 바탕 XOR 흰 펜 = 검은 선

        oldX = endX; oldY = endY; // 현 지점을 이전 지점으로 설정
    }
    ReleaseDC(hwnd, hdc);
    break;

```

## 3-7 고무줄 효과가 있는 원그리기

```
static int startX, startY, oldX, oldY;
static BOOL Drag;
int endX, endY;
switch (iMsg)
{
case WM_CREATE :
    startX = oldX = 50;    startY = oldY = 50;    // 시작 좌표
    Drag = FALSE;
    return 0 ;
case WM_PAINT :
    hdc = BeginPaint (hwnd, &ps) ;
    Ellipse(hdc, startX, startY, endX, endY);
    EndPaint (hwnd, &ps) ;
    return 0 ;
case WM_LBUTTONDOWN :    // 버튼을 누르면 드래그 동작 시작
    Drag = TRUE;
    break;
case WM_LBUTTONUP :    // 버튼을 놓으면 드래그 종료
    Drag = FALSE;
    break;
```

# 고무줄 효과가 있는 원 그리기(계속)

```

case WM_MOUSEMOVE:
    hdc = GetDC(hwnd);
    if (Drag)
    {
        SetROP2(hdc, R2_XORPEN);           // 흰 바탕
        SelectObject(hdc, (HPEN)GetStockObject(WHITE_PEN)); // 흰 펜
        // 흰 바탕 XOR 흰 펜 = 검은색 펜

        endX = LOWORD(IParam);
        endY = HIWORD(IParam);

        Ellipse(hdc, startX, startY, oldX, oldY);
        // 지우기 : 흰 바탕 XOR 검은 펜 = 흰 선

        Ellipse(hdc, startX, startY, endX, endY);

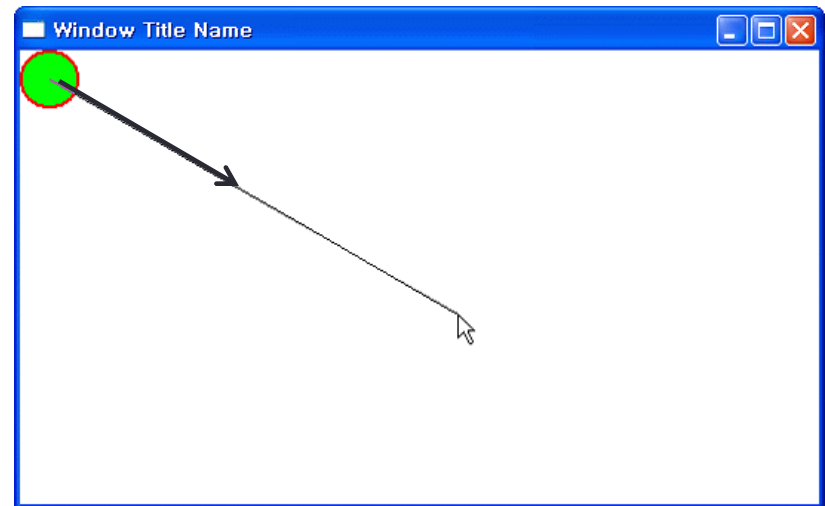
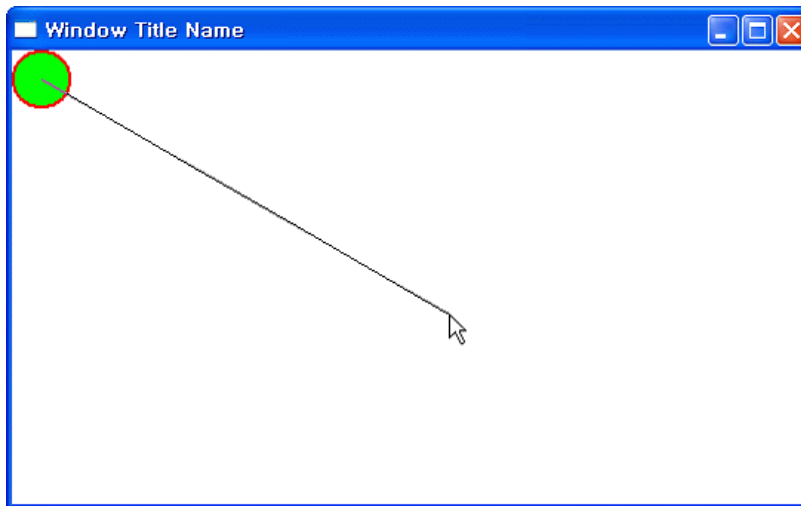
        // 그리기 : 흰 바탕 XOR 흰 펜 = 검은 선
        oldX = endX; oldY = endY; // 현 지점을 이전 지점으로 설정
    }
    ReleaseDC(hwnd, hdc);
    break;

```



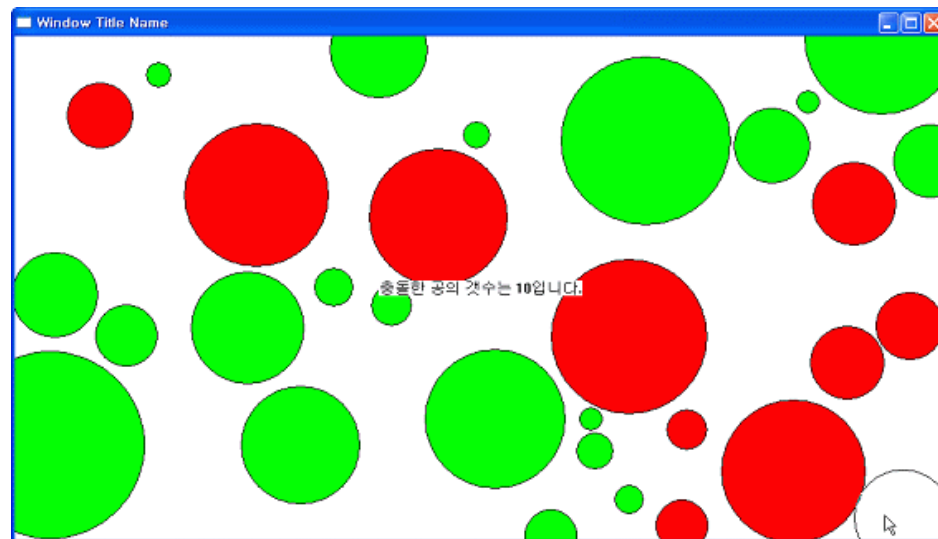
## 연습문제 3-8

- 제목
  - 선택한 원의 중심에서 마우스 커서까지 직선 그리기
- 내용
  - 중심이 (20,20)이고 반지름이 20인 원을 그리시오.
  - 왼쪽 마우스버튼을 눌러서 원이 선택되면 원의 중심부터 드래그 중인 마우스의 커서까지 직선을 그리시오.
  - 마우스버튼에서 손을 놓으면 직선위로 원이 이동하기 시작하고, 선의 끝에서 멈춘다.



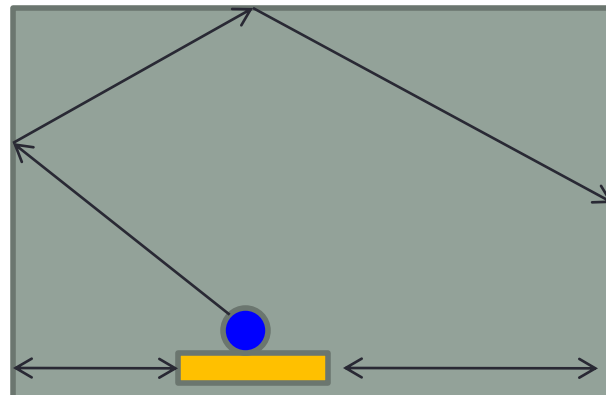
## 연습문제 3-9

- 제목
  - 장애물 피하여 이동하기
- 내용
  - 프로그램이 시작하면 화면에 장애물 역할을 할 크고 작은 원들이 많이 나타나고 주인공 역할을 할 흰색 원이 중심좌표 (0,0), 반지름 50으로 나타난다.
  - 마우스 드래그로 흰색 원을 반대쪽 모서리까지 이동 시키면 기록으로 부딪힌 장애물 숫자를 화면에 출력한다.



## 연습문제 3-10

- 제목
  - 공 튀기기 프로그램 구현하기
- 내용
  - 화면에 바를 그리고 바를 마우스로 이동하는 프로그램 구현
  - 바는 좌우로만 이동 가능하다. 특정 키를 눌러 아래의 두 이동 방법 중 한 방법으로 바가 이동하게 한다. (두 방법 모두 구현)
    - 이동 방법 1 (M/m): 좌측 마우스를 누른 채로 이동하면 바도 그 위치로 이동하고, 마우스 버튼을 놓으면 그 자리에 있다.
    - 이동 방법 2 (S/s): 좌측 마우스를 누른 채로 이동하면 바도 그 위치로 이동하고, 마우스 버튼을 놓으면, 바가 원래의 위치로 다시 이동한다.
  - 특정 키를 누르면 공의 속도가 증가/감소한다.



## 연습문제 3-11

- 제목
- 내용