

제3장 제어 메시지 처리하기

2015년 1학기 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
 - **wParam에는 타이머의 ID**가 전달된다. 이 ID는 SetTimer()함수의 두번째 인자로 지정한 값으로 여러 개의 타이머를 구분하기 위한 것
 - **lParam에는 타이머 콜백 함수**를 사용할 경우 콜백 함수명

2절. 타이머 메시지

- 타이머를 설치하면 지정한 시간간격으로 WM_TIMER메시지 유형이 발생하며, wParam에는 타이머 ID가, lParam에는 타이머 콜백함수 명이 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;
            }
            break;

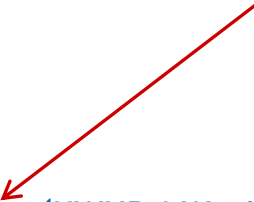
    }

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

타이머

- 타이머 콜백 함수 이용

```
HRESULT 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, TRUE);
    break;

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

연습문제 3-1

© 20000 TBA Name

- 제목

- 애벌레 이동 프로그램



- 내용

- 윈도우 화면에 애벌레 역할을 할 두 개의 원 (다른 색으로)을 나타낸다. 애벌레는 기본적으로 가장자리에 도달할 때 까지 좌우로 이동한다.
 - 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 이동방향에 수직방향으로 점프하도록 한다.
 - 위의 명령어는 처음에 생긴 애벌레에 적용된다.
- 특정 시간이 되면 새로운 애벌레가 (원 1 또는 2개) 나타난다.
 - 새로운 애벌레는 바로 전의 애벌레를 향하여 이동한다. 새로운 애벌레는 자동으로 이동한다.
- 점프 명령은 모든 원에게 적용된다.

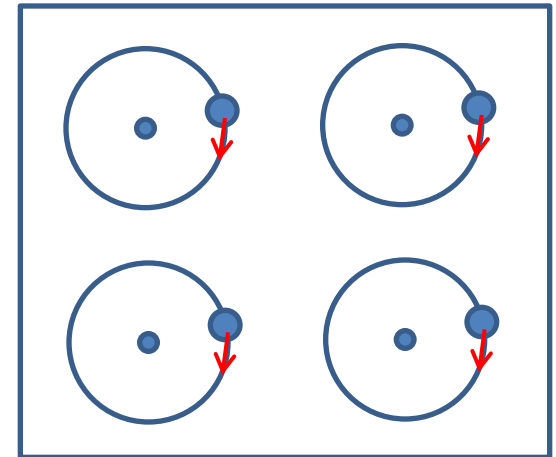
연습문제 3-2

- 제목

- 궤도를 따라 공전하는 원 만들기

- 내용

- 화면을 사등분 하여 각 등분의 중앙에 빨간색 원을 그린다.
 - 각각의 빨간색 원을 중심으로 그 주위를 임의의 반지름을 가진 원의 궤도를 그리고 그 궤도를 따라 다양한 크기와 색의 원이 다양한 타이머에 따라 회전 한다. 회전 방향은 시계방향과 반시계방향이 모두 있다.
 - 다음의 명령어를 실행한다.
 - r: 방향을 거꾸로 이동한다
(시계 -> 반시계, 반시계 -> 시계)
 - t: 원 대신 다른 도형 (예, 사각형)이 이동한다.
 - q: 프로그램이 종료
- 원의 좌표값 구하기
 - `sin()`, `cos()` 함수 사용 (`#include <math.h>`)
 - 위의 `sin`, `cos` 함수는 라디언 값을 인자로 받음



연습문제 3-3

3절. 마우스 메시지

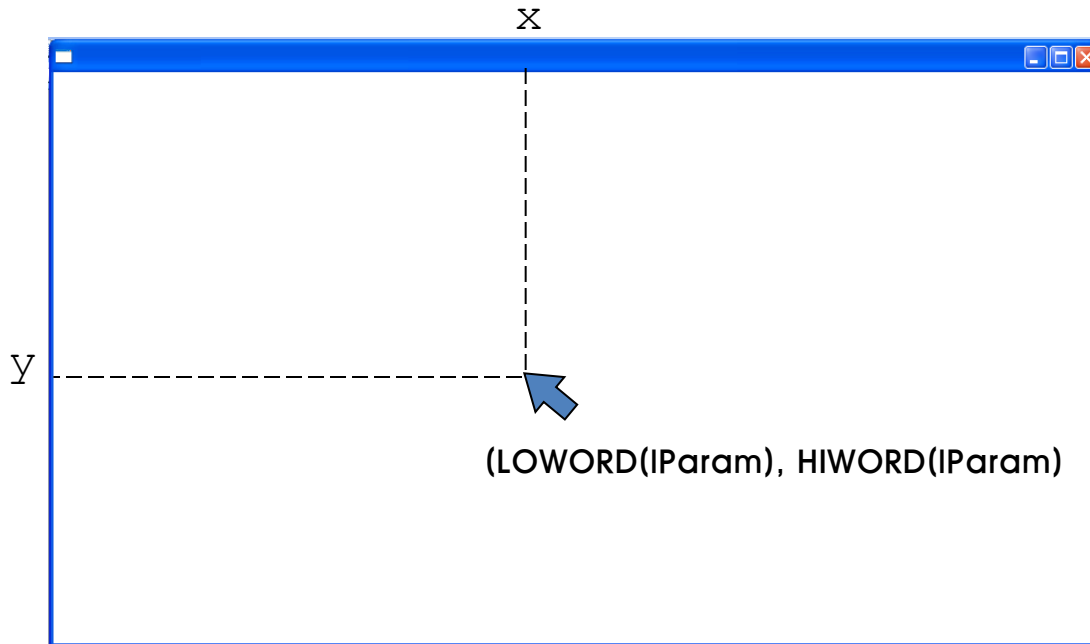
- WM_LBUTTONDOWN
 - 왼쪽 마우스 버튼을 눌렀을 때 발생하는 메시지
- WM_LBUTTONUP
 - 왼쪽 마우스 버튼을 떼었을 때 발생하는 메시지
- WM_RBUTTONDOWN
 - 오른쪽 마우스 버튼을 눌렀을 때 발생하는 메시지
- WM_RBUTTONUP
 - 오른쪽 마우스 버튼을 떼었을 때 발생하는 메시지
- WM_MOUSEMOVE
 - 마우스를 움직일 때 발생하는 메시지

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;

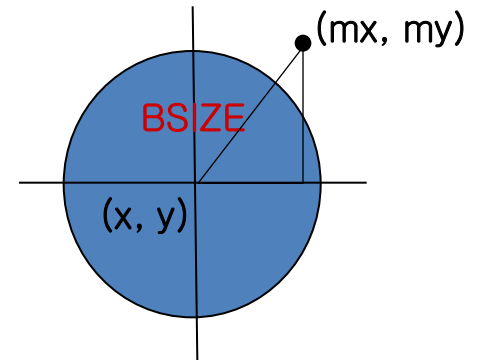
case WM_LBUTTONUP :           // 왼쪽 버튼을 놓으면
    Selection = FALSE;
    InvalidateRgn (hwnd, NULL, TRUE);
    break;
```

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;
```

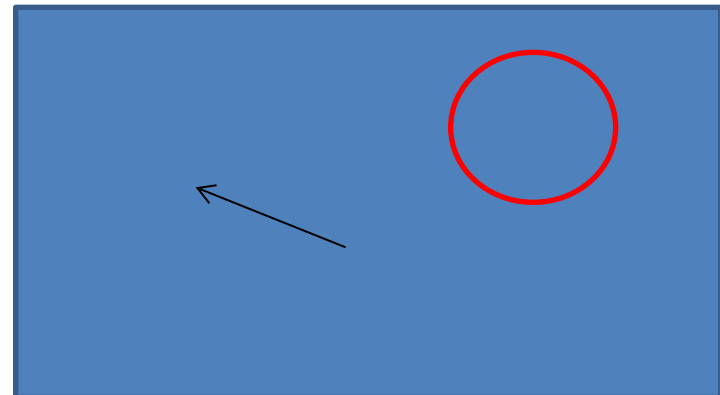
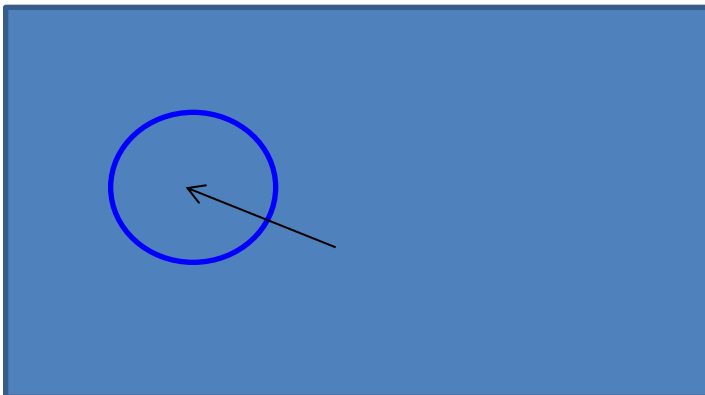
연습문제 3-4

- 제목

- 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.
 - 도형은 점점 작아지면서 사라진다. 완전히 사라지면 새로운 위치에 그려진다.
- 새롭게 그려진 도형의 내부는 새로운 색으로 그려진다. (도형 내부 혹은 도형의 외곽선의 색을 변경)
- 일정 시간이 지나면 도형들의 위치가 바뀌어져서 그려진다.



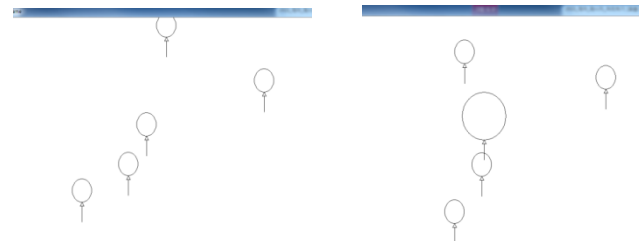
연습문제 3-5

- 제목

- 풍선 바람 넣기 게임 만들기

- 내용

- 화면의 임의의 위치에 10개의 풍선을 그리고, 풍선을 위쪽으로 올라가고 있다.
 - 풍선은 원과 삼각형, 선을 이용하여 구현한다.
 - 풍선의 내부는 랜덤 색으로 칠한다.
 - 위의 가장자리에 도달하면 화면 밖으로 나가고 다시 아래에서 새로운 풍선이 그려진다.
 - 마우스로 풍선 내부를 클릭하고 있으면 풍선의 크기가 커진다.
 - 일정 크기 이상이 되면 풍선이 터진다.
 - 터진 풍선은 납작한 모양이 되어 아래로 내려가고 화면 밖으로 사라진다.
 - 풍선이 터지면 다른 곳에 새로운 풍선이 그려진다.



연습문제 3-6

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)

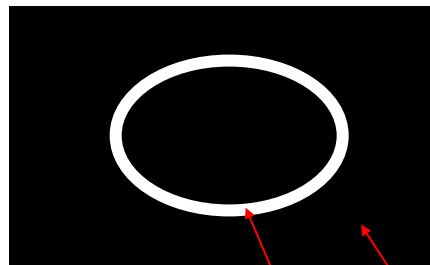
4절. 래스터 연산

- 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

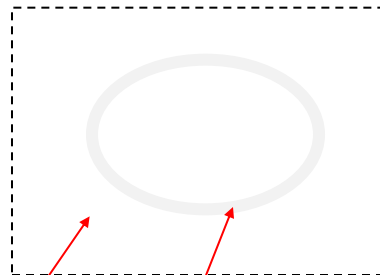
| MIX MODE | 의미 |
|-------------------|-----------------------------|
| R2_BLACK | 픽셀은 항상 0(검정색)이 된다 |
| R2_COPYPEN | 픽셀은 사용된 펜의 색상으로 칠해진다 |
| R2_MASKNOTPEN | 펜의 색상을 반전시켜 배경과 AND 연산한다 |
| R2_MASKPEN | 펜의 색상과 배경을 AND 시킨다 |
| R2_MASKPENNOT | 배경색을 반전시켜 배경과 OR 연산한다 |
| R2_MERGEPEN | 펜의 색상과 배경을 OR 시킨다 |
| 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

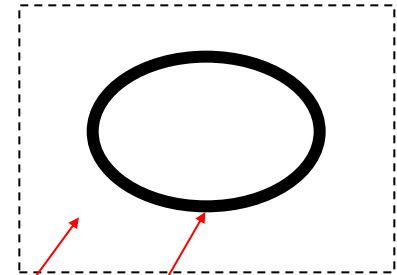
- `SetROP2(hdc,R2_XORPEN);`



XOR



=

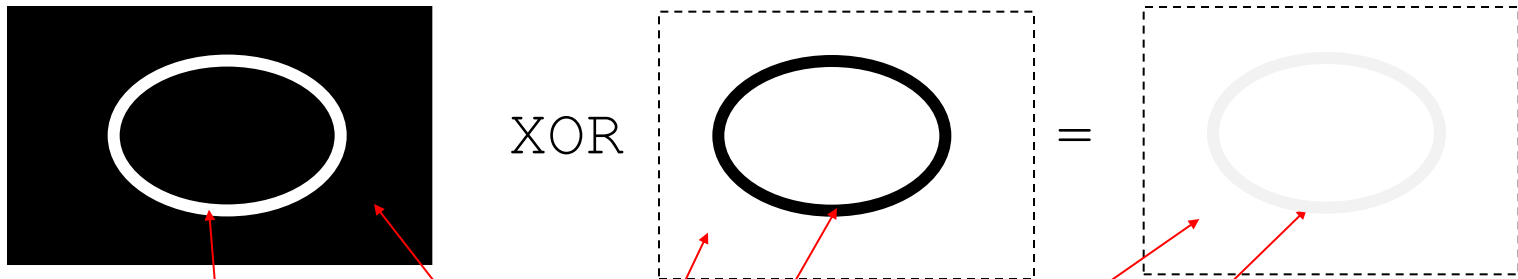


| | | | | | |
|-----------|---------------------|---|----------|----------|----------|
| 바탕 검은색 | RGB (0, 0, 0) | = | 00000000 | 00000000 | 00000000 |
| XOR 바탕 흰색 | RGB (255, 255, 255) | = | 11111111 | 11111111 | 11111111 |
| <hr/> | | | | | |
| 바탕 흰색 | RGB (255, 255, 255) | = | 11111111 | 11111111 | 11111111 |
| <hr/> | | | | | |
| XOR 흰색 원 | RGB (255, 255, 255) | = | 11111111 | 11111111 | 11111111 |
| XOR 흰색 원 | RGB (255, 255, 255) | = | 11111111 | 11111111 | 11111111 |
| <hr/> | | | | | |
| 원 검은색 | RGB (0, 0, 0) | = | 00000000 | 00000000 | 00000000 |

검은색 바탕 XOR 흰색 바탕 = 흰색 바탕
 흰색 원 XOR 흰색 원 = 검은색 원

래스터 연산으로 지우기

- `SetROP2(hdc,R2_XORPEN);`



| | | | | | |
|-----------|---------------------|---|----------|----------|----------|
| 바탕 검은색 | RGB (0, 0, 0) | = | 00000000 | 00000000 | 00000000 |
| XOR 바탕 흰색 | RGB (255, 255, 255) | = | 11111111 | 11111111 | 11111111 |
| <hr/> | | | | | |
| 바탕 흰색 | RGB (255, 255, 255) | = | 11111111 | 11111111 | 11111111 |
| <hr/> | | | | | |
| 원 흰색 | RGB (255, 255, 255) | = | 11111111 | 11111111 | 11111111 |
| XOR 원 검은색 | RGB (0, 0, 0) | = | 00000000 | 00000000 | 00000000 |
| <hr/> | | | | | |
| 바탕 흰색 | RGB (255, 255, 255) | = | 11111111 | 11111111 | 11111111 |

흰색 원 XOR 검은색 원 = 흰색 원

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, oldX, oldY);
    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)
    {
        SetROP2(hdc, R2_XORPEN);           // 흰 바탕
        SelectObject(hdc, (HPEN)GetStockObject(WHITE_PEN)); // 펜의 XOR 연산
                                                // 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;  
    break;
```

```
case WM_PAINT :
```

```
    hdc = BeginPaint (hwnd, &ps) ;  
    Ellipse(hdc, startX, startY, oldX, oldY);  
    EndPaint (hwnd, &ps) ;  
    break ;
```

```
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);
        SelectObject(hdc, (HPEN)GetStockObject(WHITE_PEN));
        // 흰 바탕
        // 펜의 XOR 연산
        // 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;
```

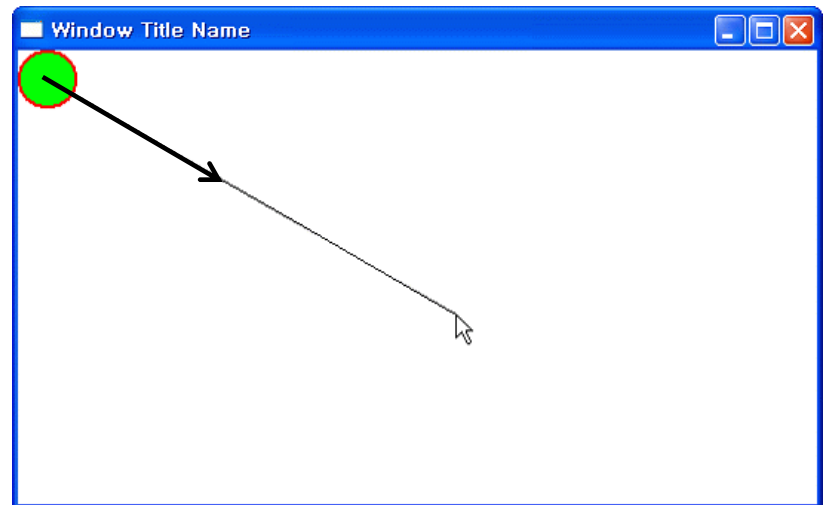
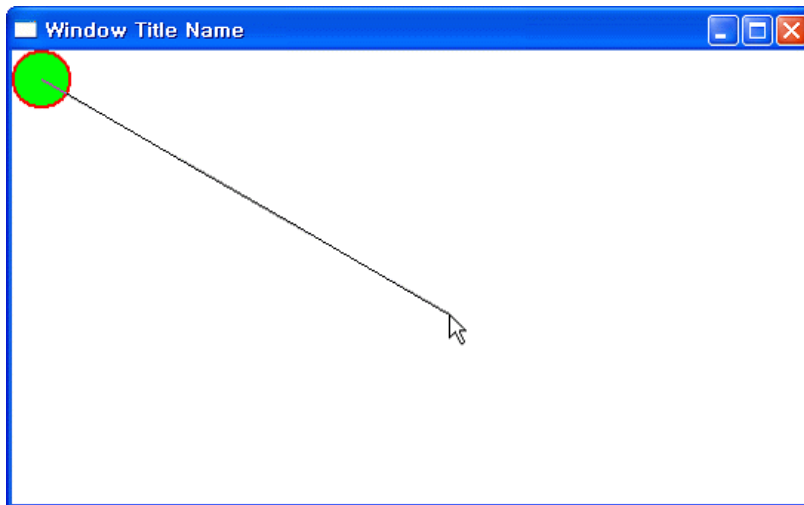
연습문제 3-7

- 제목

- 직선을 따라 움직이는 원 그리기
- 선택한 원의 중심에서 마우스 커서까지 직선 그리기

- 내용

- 왼쪽 마우스 버튼을 눌러서 드래그 하여 직선을 그린다. (고무줄 효과)
- 마우스 버튼을 놓으면 직선이 완성된다.
- 직선이 완성되면 임의의 위치에 반지름이 20인 원이 그려진다.
- 원 내부를 선택한 후 드래그 하여 직선의 한쪽 끝에 놓으면 원은 직선 위로 이동하고 선의 끝에서 멈춘다.



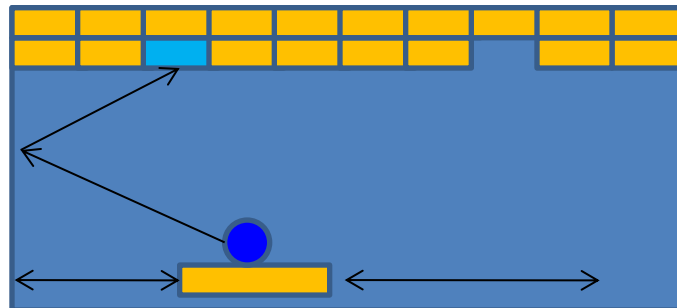
연습문제 3-8

- 제목

- 벽돌 깨기 게임 만들기

- 내용

- 화면의 상단에 2*10 개의 벽돌이 있다.
- 화면의 하단에 바가 있고 마우스를 이용하여 바를 움직인다.
 - 마우스를 누르면 마우스의 x 위치로 이동한다.
- 공이 튀기면서 벽돌에 1번 닿으면 벽돌의 색이 바뀐다.
- 공이 튀기면서 벽돌에 2번 닿으면 벽돌이 없어진다.
- 색이 변한 벽돌의 개수와 없어진 개수를 화면에 출력한다.
- 명령어 입력: 공의 이동 속도가 늘어난다.
- 벽돌이 모두 없어지면 게임이 종료된다.



연습문제 3-9

연습문제 3-10

- 제목