

# 제 3장 제어 메시지 처리하기

2022년 1학기 윈도우 프로그래밍

# 학습 목표

- 학습 목표

- 자동으로 움직이는 형상을 타이머를 이용해 윈도우에 표현할 수 있다.
- 마우스에서 발생한 메시지를 이용할 수 있다.

- 내용

- 타이머 메시지 처리하기
- 마우스 메시지 처리하기
- 래스터 연산

# 1. 윈도우의 크기 및 위치 변경 시 발생 메시지

- **WM\_SIZE**: 윈도우의 크기가 변경되면 발생하는 메시지

- wParam: 메시지가 발생한 이유
  - SIZE\_MAXHIDE: 다른 윈도우가 최대화 되어 이 윈도우가 가려졌음
  - SIZE\_MAXIMIZED: 최대화
  - SIZE\_MAXSHOW: 다른 윈도우가 원래 크기로 복구되어 이 윈도우가 나타났음
  - SIZE\_MINIMIZED: 최소화
  - SIZE\_RESTORED: 크기가 변경
- lParam: 윈도우 높이와 폭 저장
  - HIWORD(lParam): 윈도우의 높이
  - LOWORD(lParam): 윈도우의 폭

• **HIWORD()**: 32bit 데이터에서 상위 16bit 데이터를 구하기 위한 매크로 함수  
• **LOWORD()**: 32bit 데이터에서 하위 16bit 데이터를 구하기 위한 매크로 함수

- **WM\_MOVE**: 윈도우 위치가 변경되면 발생하는 메시지

- wParam: 사용되지 않음
- lParam: 변경된 윈도우의 x와 y의 좌표값
  - HIWORD(lParam): 윈도우의 새로운 y 좌표
  - LOWORD(lParam): 윈도우의 새로운 x 좌표

## 2. 타이머 메시지

- 타이머 메시지
  - 응용 프로그램에 키보드나 마우스 버튼의 입력이 아니라 다른 신호를 주기적으로 주고 싶을 때
  - 메시지 이름: **WM\_TIMER**
- 특정 함수로 타이머를 설치했을 경우, 지정한 시간 간격으로 **WM\_TIMER** 메시지가 반복적으로 큐에 넣어진다.
  - 타이머 설정 함수: `SetTimer ()`;
  - **WM\_TIMER** 메시지 발생 시:
    - **wParam**: 타이머의 ID
    - **lParam**: 타이머 콜백 함수 이름 (콜백함수를 사용하지 않으면 **NULL**)
  - 한 개 이상의 타이머를 설정할 수 있다.
  - 여러 개의 타이머가 설치되어 있을 경우:
    - 각각의 타이머는 정해진 시간 간격으로 이 메시지를 큐에 저장
    - 어떤 타이머에 의해 이 메시지가 발생했는지 **wParam** 값 (타이머 id)으로 조사한다.
- **WM\_TIMER** 메시지는 다른 메시지들에 비해 **우선순위가 낮게 설정**
  - 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 지정한다.
    - 타이머 콜백 함수를 지정했을 경우는 이 메시지부를 수행하는 것이 아니라, 프로그래머가 만든 함수(타이머 콜백 함수)를 OS가 자동으로 주기적으로 호출해 준다.

# 타이머 메시지

- 타이머 설정함수
  - 타이머를 설정/해제하는 함수

**WORD SetTimer** (HWND hWnd, UINT\_PTR nIDEvent, UINT uElapse, TIMERPROC lpTimerFunc);

- hWnd : 윈도우 핸들
- nIDEvent : 타이머 ID, 여러 개의 타이머를 구분하기 위한 정수
- uElapse : 시간간격 milisec 단위(1000분의 1초)
- lpTimerFunc : 시간간격 마다 수행할 함수
  - NULL이라고 쓰면 WndProc() 함수가 타이머메시지를 처리)

**BOOL KillTimer** (HWND hWnd, UINT\_PTR uIDEvent);

- hWnd: 윈도우 핸들
- uIDEvent: 삭제할 타이머의 ID

- 타이머 콜백 함수
  - SetTimer 함수의 마지막 인자로 설정되는 타이머 콜백 함수

**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

# 타이머 메시지

- WM\_TIMER 메시지 처리방법

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

    switch (iMsg) // 메시지 번호
    {
        case WM_CREATE:
            SetTimer (hwnd, 1, 60, NULL);           //--- 1번 아이디를 가진 타이머: 0.06초 간격
            SetTimer (hwnd, 2, 100, NULL);          //--- 2번 아이디를 가진 타이머: 0.1초 간격
            break;

        case WM_TIMER:
            switch (wParam) {
                case 1:                               //--- 1번 아이디 타이머: 0.06초 간격으로 실행
                    Timer1Count++;
                    break;
                case 2:                               //--- 2번 아이디 타이머: 0.1초 간격으로 실행
                    Timer2Count++;
                    break;
            }
            InvalidateRect (hwnd, NULL, TRUE);
            break;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps);
            if ( Timer1Count % 2 == 0 )
                TextOut (hdc, Timer1Count*10, 0, L"Timer1 Count", 12);
            if ( Timer2Count % 2 == 0 )
                TextOut (hdc, Timer2Count*10, 100, L"Timer2 Count", 12);
            EndPaint (hwnd, &ps);
            break;
    }

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

# 타이머 메시지

- 타이머 콜백 함수 이용 방법

```
LRESULT CALLBACK WndProc (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch ( uMsg ){
        case WM_CREATE:
            SetTimer (hWnd, 1, 500, TimerProc);    // 1번 아이디의 타이머가 0.5초 마다 TimerProc 타이머 함수 실행
            break;
    }
    return 0;
}

void CALLBACK TimerProc (HWND hWnd, UINT uMsg, UINT idEvent, DWORD dwTime )    //1번 아이디 타이머 함수
{
    HDC hdc;
    HBRUSH MyBrush, OldBrush;
    RECT rect;

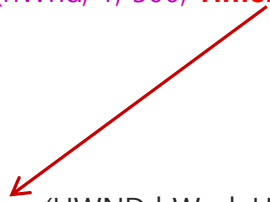
    hdc = GetDC(hWnd);
    GetClientRect(hWnd, &rect);

    MyBrush = CreateSolidBrush(RGB(rand()%255, rand()%255, rand()%255) );
    OldBrush = (HBRUSH)SelectObject(hdc, MyBrush);

    Ellipse(hdc, rand()%(rect.right), rand()%(rect.bottom), rand()%(rect.right), rand()%(rect.bottom) );

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

    ReleaseDC(hWnd, hdc);
}
```



# 예) 타이머 메시지

- 타이머를 이용하여 시계 구현하기


LRESULT CALLBACK **WndProc** (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)

```
{
    HDC hdc;
    PAINTSTRUCT ps;
    SYSTEMTIME curTime;
    static TCHAR str[100];

    switch ( uMsg ){
        case WM_CREATE:
            SetTimer (hWnd, 1, 1000, NULL);           // 1번 아이디의 타이머가 1초 마다 WM_TIMER 메시지 발생
            break;
        case WM_TIMER:
            GetLocalTime (&curTime);
            wsprintf (str, TEXT("current time: %d: %d: %d"), curTime.wHour, curTime.wMinute, curTime.wSecond);
            InvalidateRect (hWnd, NULL, TRUE);
            break;
        case WM_PAINT:
            hdc = BeginPaint (hWnd, &ps);
            TextOut (hdc, 0, 0, str, lstrlen(str));
            EndPaint (hWnd, &ps);
            break;
        case WM_DESTROY:
            PostQuitMessage (0);
            break;
    }
    return 0;
}
```

```
void GetLocalTime(
    [out] LPSYSTEMTIME lpSystemTime
);
```

```
typedef struct _SYSTEMTIME {
    WORD wYear;
    WORD wMonth;
    WORD wDayOfWeek;
    WORD wDay;
    WORD wHour;
    WORD wMinute;
    WORD wSecond;
    WORD wMilliseconds;
} SYSTEMTIME, *PSYSTEMTIME, *LPSYSTEMTIME;
```

 Window Programming Lab

**current time : 16:7:28**



## 예) 타이머 메시지

- 오른쪽 화살표 키를 누르면 타이머가 설정되고, 왼쪽에 있던 원이 오른쪽으로 이동한다. 원이 오른쪽 가장자리에 도착하면 반대편 가장자리로 이동해서 다시 오른쪽으로 이동한다.

LRESULT CALLBACK **WndProc** (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)

{

```
static int x=0, y;  
static RECT rectView;
```

```
switch (iMsg)
```

```
{
```

```
case WM_CREATE:
```

```
    GetClientRect (hwnd, &rectView);
```

```
    x = 20;    y = 20;
```

```
break;
```

```
case WM_PAINT:
```

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

```
    Ellipse (hdc, x-20, y-20, x+20, y+20);
```

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

```
break;
```

```
case WM_KEYDOWN:
```

```
    if (wParam == VK_RIGHT)
```

```
        SetTimer (hwnd, 1, 70, NULL);
```

```
break;
```

```
case WM_TIMER:
```

```
    x += 40;
```

```
    if (x + 20 > rectView.right)
```

```
        x = 0;
```

```
    InvalidateRect (hwnd, NULL, TRUE);
```

```
break;
```

```
case WM_DESTROY :
```

```
    KillTimer (hwnd, 1);
```

```
    PostQuitMessage (0) ;
```

```
break ;
```

```
}
```

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

```
}
```

//--- 메시지 종류

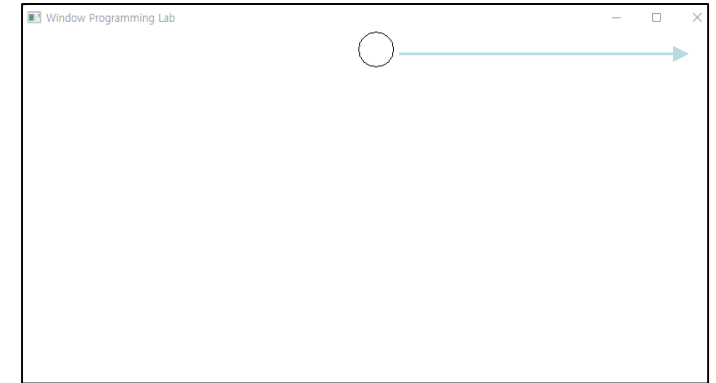
//--- 오른쪽 키를 누를 때

//--- 타이머 설정

//--- 타이머 메시지: 설정된 시간마다 생성

//--- 오른쪽 가장자리에 도착하면 왼쪽 가장자리로 이동

//--- 윈도우 종료 시 타이머도 종료



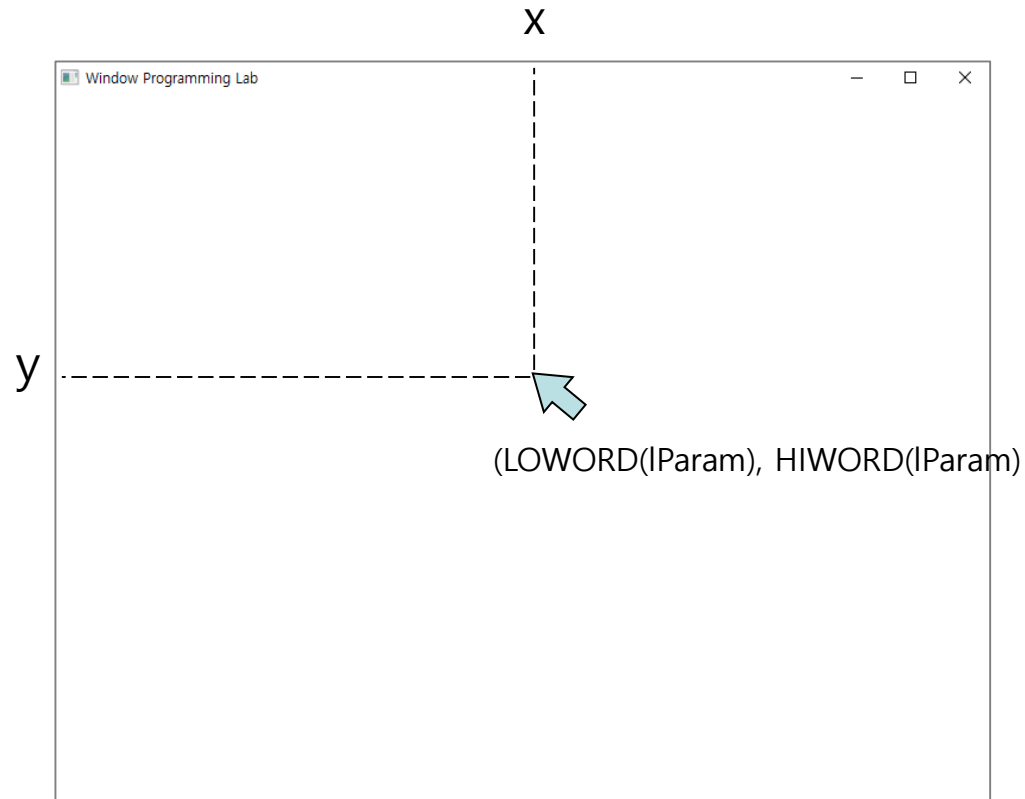
# 3. 마우스 메시지

- 좌우에 버튼이 2개 있는 마우스의 이벤트

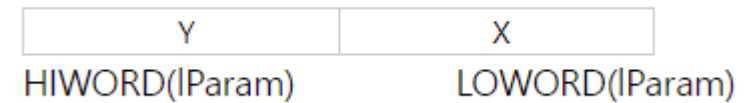
메시지	내용	윈도우 프로시저 인수값
WM_LBUTTONDOWN	왼쪽 마우스 버튼을 눌렀을 때 발생하는 메시지	<p><b>wParam:</b> 키보드와 다른 마우스 버튼의 현재 상태를 알린다. MK_CONTROL: ctrl 키가 눌러져 있다. MK_LBUTTON: 마우스 왼쪽 버튼이 눌러져있다. MK_RBUTTON:마우스 오른쪽 버튼이 눌러져있다. MK_MBUTTON: 마우스 중간 버튼이 눌러져있다. MK_SHIFT: shift 키가 눌러져 있다.</p> <p><b>lParam:</b> x와 y 좌표값이 저장 <b>HIWORD (lParam): y 값</b> <b>LOWORD (lParam): x 값</b></p>
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	
WM_LBUTTONDOWNBLCLK / WM_RBUTTONDOWNBLCLK	버튼 더블 클릭 눌렀을 때 발생하는 메시지 윈도우 클래스가 반드시 <b>CS_DBLCLKS</b> 스타일을 가져야 한다	

# 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



## 예) 마우스로 원 선택하기

- (50, 50) 위치에 반지름이 40인 원을 그리고, 마우스를 원 내부에 클릭하면 테두리에 사각형을 그린다.

```
#include <math.h>
```

```
#define BSIZE 40 //--- 반지름
```

```
//--- (x1, y1)과 (x2, y2)간의 길이
```

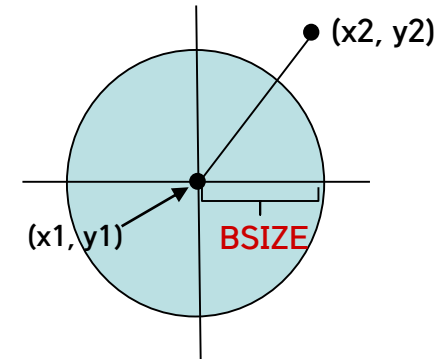
```
float LengthPts (int x1, int y1, int x2, int y2)
```

```
{  
    return (sqrt((x2-x1)*(x2-x1) +(y2-y1)*(y2-y1)));  
}
```

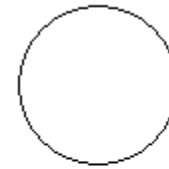
```
//--- (x1, y1)과 (x2, y2)의 길이가 반지름보다 짧으면 true, 아니면 false
```

```
BOOL InCircle (int x1, int y1, int x2, int y2)
```

```
{  
    if (LengthPts (x1, y1, x2, y2) < BSIZE) //--- BSIZE: 반지름  
        return TRUE;  
    else  
        return FALSE;  
}
```



Window Programming Lab



## 예) 마우스로 원 선택하기(계속)

LRESULT CALLBACK **WndProc** (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)

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

    switch (iMsg)
    {
        case WM_CREATE :
            x = 50;    y = 50;
            Selection = FALSE;           //--- 원이 선택되었나, FALSE : 아직 안되었음
            break;
        case WM_PAINT :
            hdc = BeginPaint (hwnd, &ps) ;
            if (Selection)               //--- 만약 원이 선택되었다면, 4각형을 그린다. 아니면 원만 그린다.
                Rectangle(hdc, x-BSIZE, y-BSIZE, x+BSIZE, y+BSIZE);
            Ellipse(hdc, x-BSIZE, y-BSIZE, x+BSIZE, y+BSIZE);
            EndPaint (hwnd, &ps) ;
            break;
        case WM_LBUTTONDOWN :           //--- 왼쪽 버튼 누르면
            mx = LOWORD(lParam);         //--- 마우스 좌표값: (mx, my)
            my = HIWORD(lParam);
            if (InCircle (x, y, mx, my)) //--- 원의 중심점(x, y)와 마우스 좌표 비교
                Selection = TRUE;        //--- 원 안에 있으면 true
            InvalidateRect (hwnd, NULL, TRUE);
            break;
        case WM_LBUTTONUP :           //--- 왼쪽 버튼을 놓으면
            Selection = FALSE;
            InvalidateRect (hwnd, NULL, TRUE);
            break;
        case WM_DESTROY:
            PostQuitMessage (0);
            break;
    }
    return (DefWindowProc (hwnd, iMsg, wParam, lParam));
}
```

## 예) 마우스 드래그로 원 이동하기

- 왼쪽 마우스 버튼으로 원을 선택한 후 마우스를 드래그하면 원과 사각형이 마우스의 위치로 이동된다.

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    static int x, y;
    static BOOL Selection;
    int mx, my;

    switch (iMsg)
    {
        case WM_LBUTTONDOWN :
            mx = LOWORD(lParam);
            my = HIWORD(lParam);
            if (InCircle(x, y, mx, my))
                Selection = TRUE;                //--- mx, my : 마우스 좌표
            InvalidateRect (hwnd, NULL, TRUE);
            break;

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

        case WM_MOUSEMOVE:
            mx = LOWORD(lParam);
            my = HIWORD(lParam);
            if (Selection)                //--- 원이 선택된 상태로 움직이면
            {
                x = mx;        y = my;
                InvalidateRect (hwnd, NULL, TRUE);    //--- 원과 사각형 그리기
            }
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
    }
    return (DefWindowProc (hwnd, iMsg, wParam, lParam));
}
```

## 예) 더블 클릭

- 왼쪽 마우스 버튼 더블 클릭하면 도형을 그리거나 지우거나 한다.

```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpszCmdParam, int nCmdShow)
{
    HWND hWnd;
    MSG Message;
    WNDCLASSEX WndClass;
    g_hInst = hInstance;

    WndClass.cbSize = sizeof(WndClass);
    WndClass.style = CS_HREDRAW | CS_VREDRAW | CS_DBLCLKS;
    WndClass.lpfnWndProc = (WNDPROC)WndProc;
    WndClass.cbClsExtra = 0;
    WndClass.cbWndExtra = 0;
    WndClass.hInstance = hInstance;
    WndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
    WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
    WndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
    WndClass.lpszMenuName = NULL;
    WndClass.lpszClassName = lpszClass; // lpszClass;
    WndClass.hIconSm = LoadIcon(NULL, IDI_APPLICATION);
    RegisterClassEx(&WndClass);

    ...
}
```

## 예) 더블 클릭

LRESULT CALLBACK **WndProc** (HWND hwnd, UINT iMessage, WPARAM wParam, LPARAM lParam)

```
{
    PAINTSTRUCT ps;
    HDC hdc;
    HPEN hPen, oldPen;
    static int draw = 1;

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

            if (draw) {
                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;

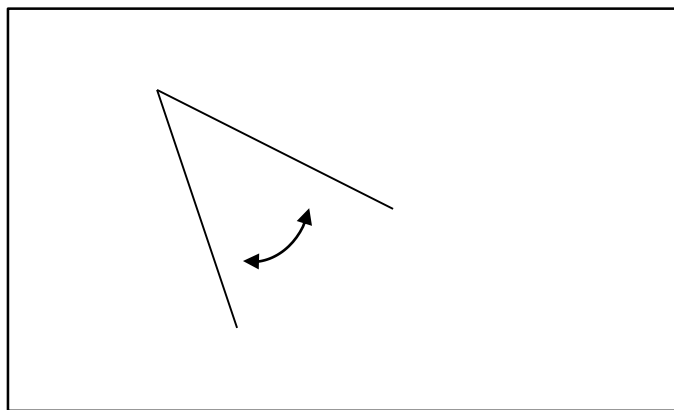
        case WM_LBUTTONDOWN:
            draw++;
            draw %= 2;
            InvalidateRect (hwnd, NULL, TRUE);
            break;

        case WM_DESTROY:
            PostQuitMessage (0);
            break;
    }
    return (DefWindowProc(hwnd, iMessage, wParam, lParam));
}
```



## 4. 래스터 연산

- 객체 움직이기 위해 다시 그리기
  - 화면 전체를 다시 그리기 한다.
  - 빠른 움직임을 위해 윈도우 전체를 삭제하고 다시 그리는 것은 좋지 않다.
- 래스터 연산
  - 윈도우의 배경색과 그리는 색을 연산한 결과 색상으로 그림
  - AND, OR, XOR 등 비트간의 이진 연산과 NOT 연산의 조합으로 지정됨
  - 그리기 연산은 래스터 디바이스에만 적용되며 벡터 디바이스에는 적용되지 않음
  - 래스터 연산 함수 사용
    - 움직이거나 동적으로 표현되는 상태인 경우: 재 출력할 필요 없이 해당 메시지에서 GetDC()로 즉각 출력한다.
    - 선을 그릴 때 마우스를 드래그하면, 마우스 이동 메시지에서 이전의 선을 지우고 새로운 선을 그려야 한다.
  - Raster Operation (Bitwise Boolean 연산)
    - Raster: 이미지를 점들의 패턴으로 표현하는 방식 (cf. Vector)



# 래스터 연산

- 래스터 연산 설정 함수

**int SetROP2 (HDC hdc, int fnDrawMode);**

- 두 픽셀 사이에 bit 연산을 수행하도록 mix 모드를 설정할 수 있는 기능
- HDC hdc: 디바이스 컨텍스트 핸들
- int fnDrawMode: 그리기 모드 (아래 표 참고)

그리기 모드	의미
R2_BLACK	픽셀은 항상 0(검정색)이 된다
R2_COPYPEN	픽셀은 사용된 펜의 색상으로 칠해진다 (디폴트 값)
R2_MASKNOTPEN	펜의 색상을 반전시켜 배경과 AND 연산한다
R2_MASKPEN	펜의 색상과 배경을 AND 시킨다
R2_MASKPENNOT	펜의 색상과 배경을 반전시켜 AND 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 연산

- 그리기 모드에서

- R2\_XORPEN;**

- 바탕색과 그리는 색 사이의 XOR 연산을 수행
    - XOR 연산 : 두 개의 비트가 다를 때만 true(1), 같으면 false(0)

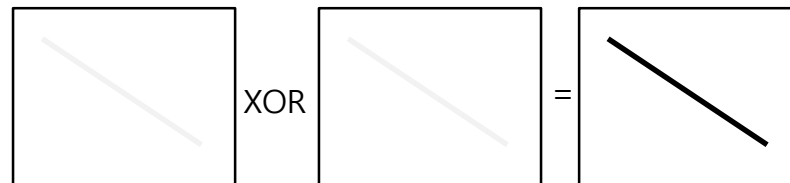
- 사용 예)

- SetROP2 (hdc, R2\_XORPEN); //--- 펜의 색과 배경색을 XOR 연산
  - //--- 펜: (0, 0, 0), 배경색 (255, 255, 255) → 화면에 그려지는 펜 색: (255, 255, 255)
  - //--- 펜: (255, 255, 255), 배경색 (255, 255, 255) → 화면에 그려지는 펜 색: (0, 0, 0)

	검은색	RGB(0,0,0)	= 00000000 00000000 00000000
XOR	흰 색	RGB(255,255,255)	= 11111111 11111111 11111111
<hr/>			
	흰 색	RGB(255,255,255)	= 11111111 11111111 11111111



	흰 색	RGB(255,255,255)	= 11111111 11111111 11111111
XOR	흰 색	RGB(255,255,255)	= 11111111 11111111 11111111
<hr/>			
	검은색	RGB(0,0,0)	= 00000000 00000000 00000000



# 고무줄 효과가 있는 직선그리기

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

```
{
    HDC hdc;
    static int startX, startY, oldX, oldY;
    static BOOL Drag;
    int endX, endY;

    switch (iMsg)
    {
        case WM_CREATE :
            startX = oldX = 0;          startY = oldY = 0;          //--- 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);
        SelectObject(hdc, (HPEN)GetStockObject(WHITE_PEN));

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

        MoveToEx(hdc, startX, startY, NULL);
        LineTo(hdc, oldX, oldY);

        MoveToEx(hdc, startX, startY, NULL);
        LineTo(hdc, endX, endY);

        oldX = endX; oldY = endY;
    }
    ReleaseDC(hwnd, hdc);
    break;

case WM_DESTROY:
    PostQuitMessage(0);
    break;

}

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

//--- 흰 바탕  
//--- 펜의 XOR 연산  
//--- 흰 펜  
//--- 흰 바탕 XOR 흰 펜 = 검은색  
//--- 검정 바탕 XOR 흰 펜 = 흰 색

//--- 지우기 : 검정 바탕 XOR 흰 펜 = 흰 선

//--- 그리기 : 흰 바탕 XOR 흰 펜 = 검은 선 → 화면의 결과 선

//--- 현 지점을 이전 지점으로 설정

# 고무줄 효과가 있는 원 그리기

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

```
{
    HDC hdc;
    PAINTSTRUCT ps;
    static int startX, startY, oldX, oldY;
    static BOOL Drag;
    int endX, endY;

    switch (iMsg)
    {
    case WM_CREATE:
        startX = oldX = 0;  startY = oldY = 0;
        Drag = FALSE;
        break;
    case WM_PAINT:
        hdc = BeginPaint(hwnd, &ps);
        Ellipse(hdc, startX, startY, oldX, oldY);
        EndPaint(hwnd, &ps);
        break;
    case WM_LBUTTONDOWN :
        oldX = startX = LOWORD(lParam);
        oldY = startY = HIWORD(lParam);
        Drag = TRUE;
        break;
    case WM_LBUTTONUP :
        Drag = FALSE;
        break;
    }
```

```
case WM_MOUSEMOVE:
    hdc = GetDC(hwnd);
    if (Drag)
    {
        SetROP2(hdc, R2_XORPEN);
        SelectObject(hdc, (HPEN)GetStockObject(WHITE_PEN));
        SelectObject(hdc, (HBRUSH)GetStockObject(BLACK_BRUSH));
        endX = LOWORD(lParam);
        endY = HIWORD(lParam);
        Ellipse(hdc, startX, startY, oldX, oldY);  // 지우기
        Ellipse(hdc, startX, startY, endX, endY);  // 그리기
        oldX = endX; oldY = endY;
    }
    ReleaseDC(hwnd, hdc);
    break;
case WM_DESTROY:
    PostQuitMessage(0);
    break;
}
return(DefWindowProc(hwnd, iMsg, wParam, lParam));
}
```

## 실습 3-1

- 움직이는 원에 꼬리 달기
  - 화면에 40x40의 보드가 그려진다.
  - **화면의 한쪽 코너에 주인공원이** 있고, 시작 명령어에 따라 열 또는 행에 맞춰 지그재그로 보드의 칸에 맞춰 자동 이동한다.
  - **보드의 랜덤한 위치에서 특정 시간마다 꼬리원들이** 나타난다. 꼬리원들은
    - **이동 방향 1:** 좌우상하 방향 중 한 방향으로 이동, 보드의 가장자리에 도착하면 방향을 바꿔서 이동한다.
    - **이동 방향 2:** 또는 네모를 그리며 돈다.
    - **이동 방향 3:** 제자리에 그대로 있다.
    - 꼬리원끼리 부딪치면 두 개가 붙어서 이동한다.
  - **주인공원과 꼬리원이 만나면 꼬리원은 주인공 원의 뒤에 꼬리로 붙는다.**
  - **(원 대신 다른 도형을 사용해도 무관함)**

- 움직이는 원에 꼬리 달기

- 마우스 명령: 왼쪽 마우스 버튼

- 왼쪽 마우스 버튼을 빈 보드에 누르면:** 마우스가 클릭된 방향으로 원이 이동 방향을 바꾸고 다시 지그재그로 이동한다.
    - 왼쪽 마우스 버튼으로 주인공 원의 내부를 클릭하면:** 원의 이동 속도가 잠시 빨라지고, 다시 원래 속도로 이동한다.
    - 왼쪽 마우스 버튼으로 꼬리원을 클릭하면:** 꼬리원은 주인공 원에서 분리되어 보드의 칸에 남겨지며 이동한다 (새롭게 생긴 꼬리원같이 앞 페이지의 이동방법 3개 중 한 개 방법으로 이동).
      - 중간의 꼬리를 클릭하면 클릭된 꼬리 뒤에 있는 꼬리들도 다 분리된 후 이동한다 (앞 페이지의 이동방법 2개 중 한 개 방법으로 이동).

- 마우스 명령: 오른쪽 마우스 버튼

- 오른쪽 마우스 버튼을 빈 보드에 클릭하면:** 그 자리에 장애물이 생기고, 원은 장애물을 만나면 방향을 바꾼다.

- 키보드 명령어:

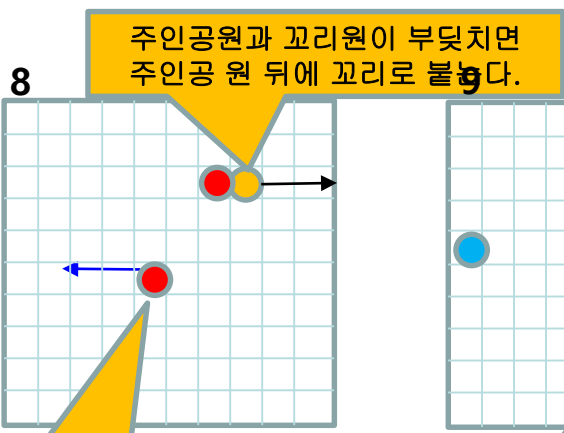
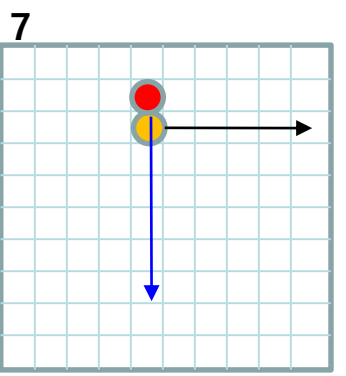
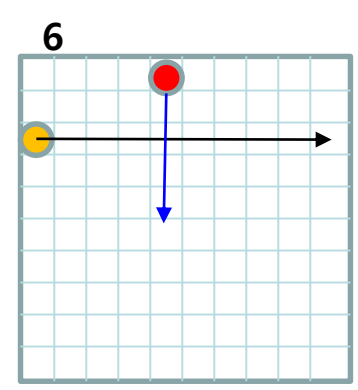
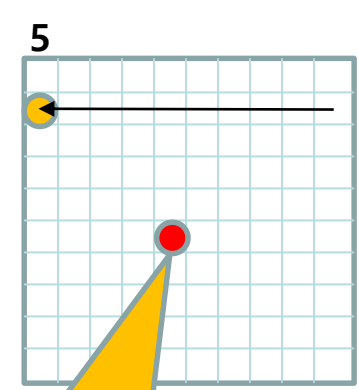
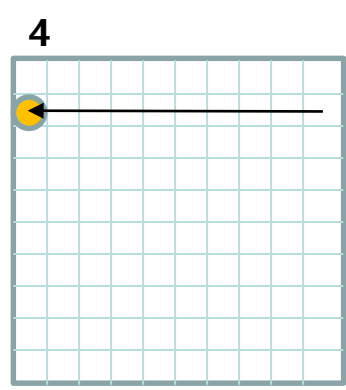
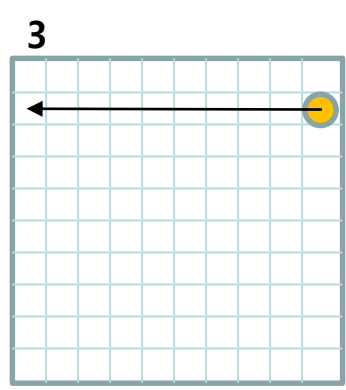
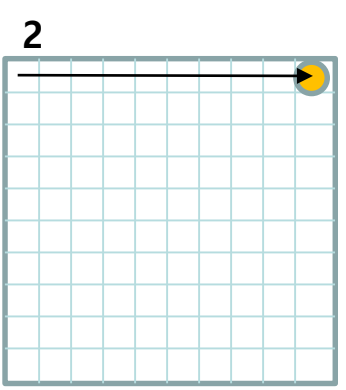
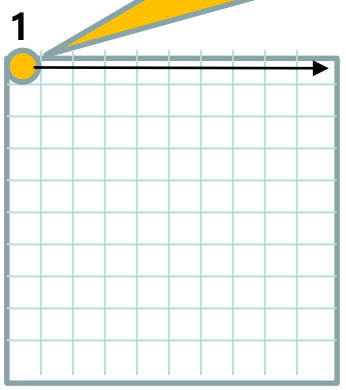
- s/S:** 원 움직임 시작
    - ↑/←/→/↓:** 좌우상하 키보드를 입력하면 주인공 원의 방향이 좌우상하로 바뀌고 이동한다. 가장자리에 도달하면 지그재그 이동한다.
    - +/-:** '+'를 입력하면 주인공원의 속도가 점점 빨라지고, '-'를 입력하면 속도가 점점 느려진다.
    - j/J:** 주인공원과 그 꼬리들은 그 자리에서 (또는 이동하면서) 이동방향에 수직방향으로 점프하도록 한다.
    - t/T:** 주인공 원과 그 꼬리들이 유턴을 한다. (예, 좌로 이동하고 있고 t를 입력하면, 좌 → 하 → 우로 방향을 바꾼다. 아래로 이동하고 있고 t를 입력하면, 하 → 우 → 상 이런 식으로 유턴을 한다.)
    - q/Q:** 프로그램이 종료한다.



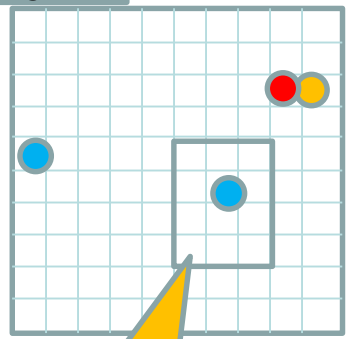
# 실습 3-1

기본

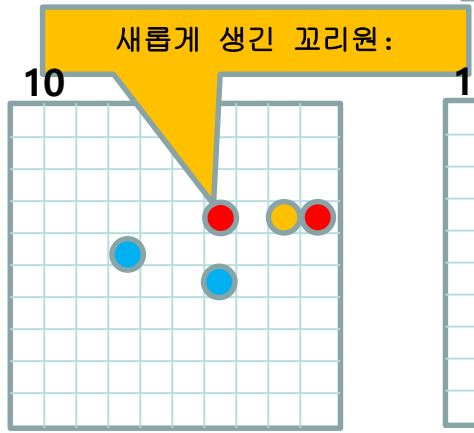
주인공 원: 좌아래우아래 방향  
지그재그로 이동



주인공원과 꼬리원이 부딪치면  
주인공 원 뒤에 꼬리로 붙는다.



새로운 꼬리 원: 중앙에서  
생기고 랜덤방향으로 이동  
(이 경우는 왼쪽으로 이동)

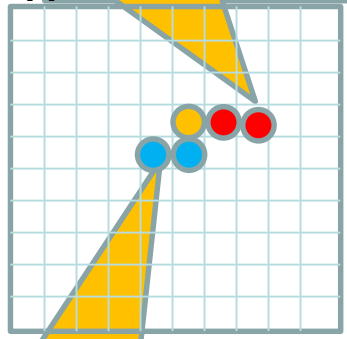


새롭게 생긴 꼬리원:

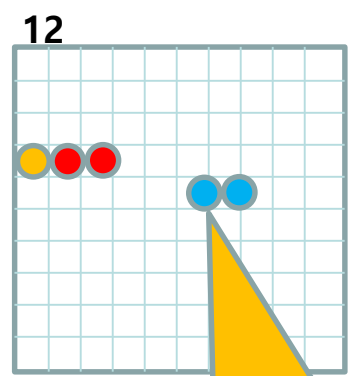
새롭게 생긴 꼬리원: 사각형  
경로로 이동한다. (이동방법  
2)

꼬리 원: 중앙에서 생기고  
랜덤방향으로 이동 (이 경우는  
위쪽으로 이동)

주인공과 만나 주인공 뒤에  
붙는다



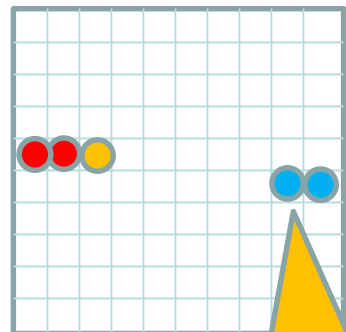
꼬리원끼리 부딪친다.



꼬리원이 붙어서 이동한다.

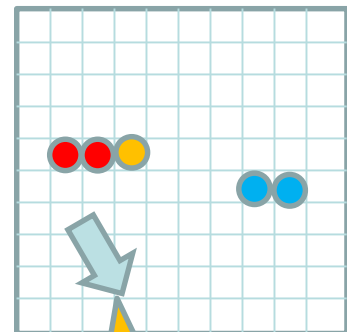
# 실습 3-1

13



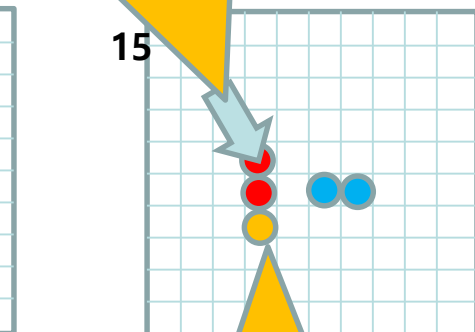
꼬리원들이 붙어서 이동한다.

14



마우스클릭: 주인공 원 아래쪽으로 왼쪽 마우스를 누른다.

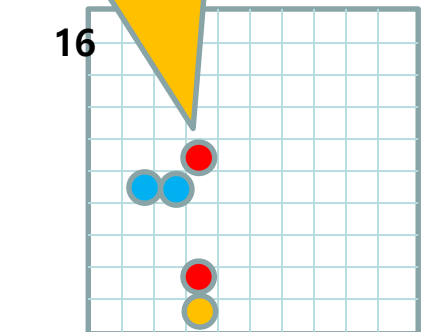
15



마우스클릭: 꼬리원을 클릭한다.

주인공원의 방향이 아래로 바뀐다.

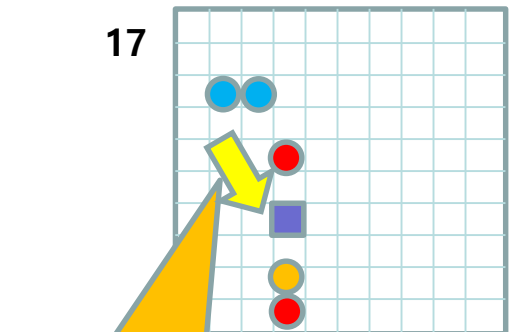
16



꼬리원이 주인공과 분리되어 이동한다.

주인공 원은 이동방향으로 계속 이동한다.

17



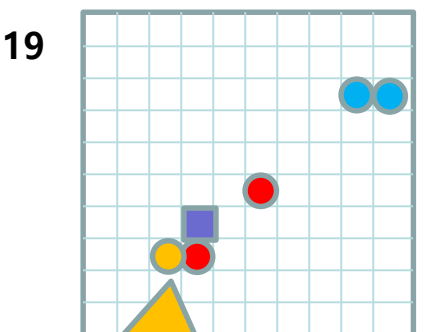
우 마우스 클릭: 장애물이 생긴다.

18



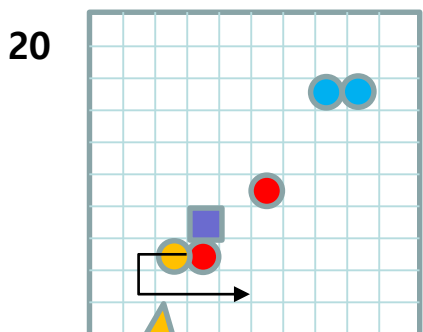
원이 장애물에 부딪친다.

19



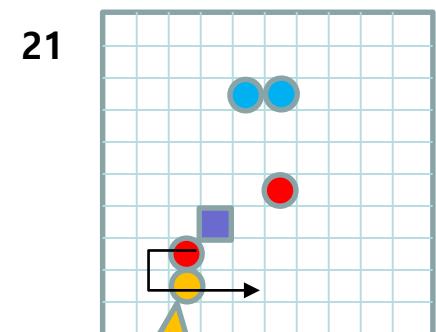
원이 방향을 바꾼다.

20



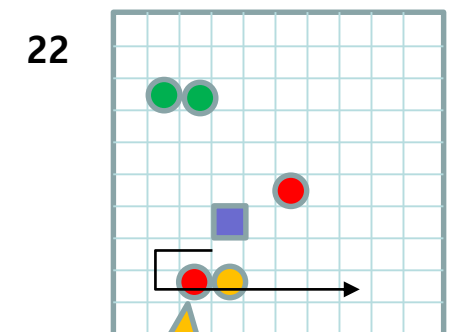
T를 입력하면 화살표 방향으로 방향을 유턴한다.

21



T를 입력하면 화살표 방향으로 방향을 유턴한다.

22

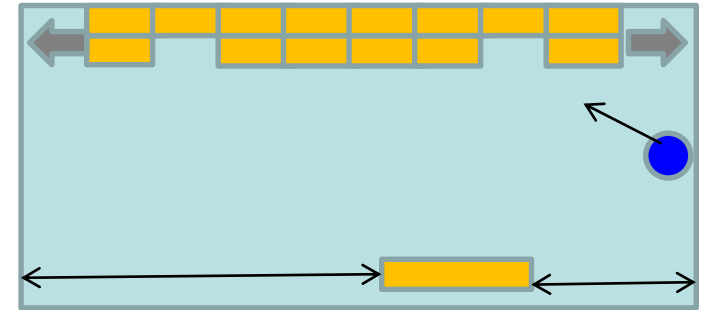
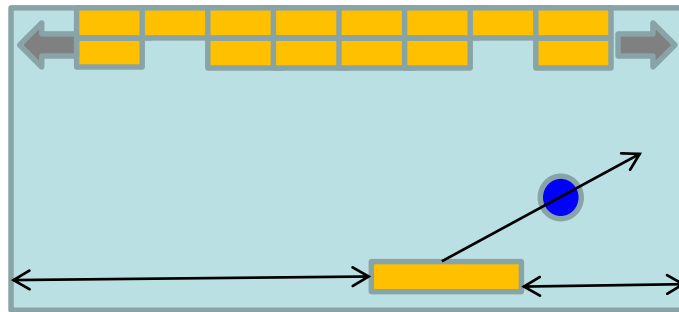
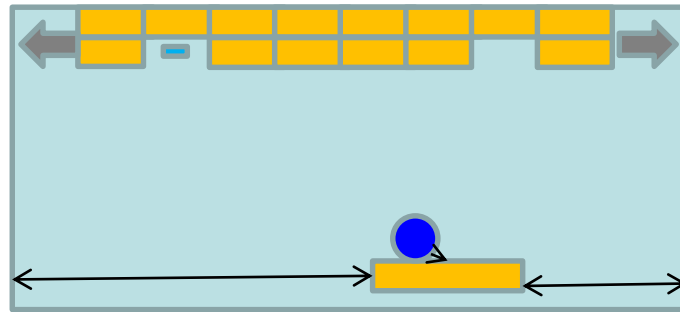
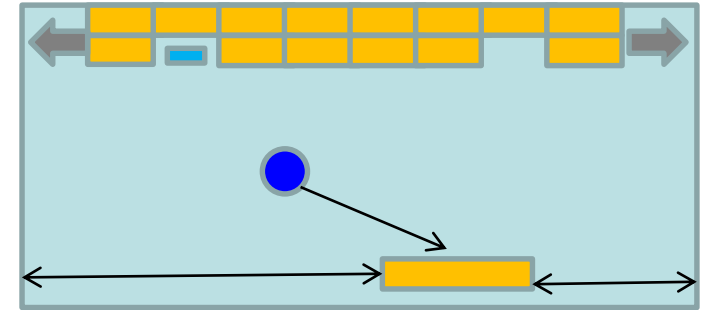
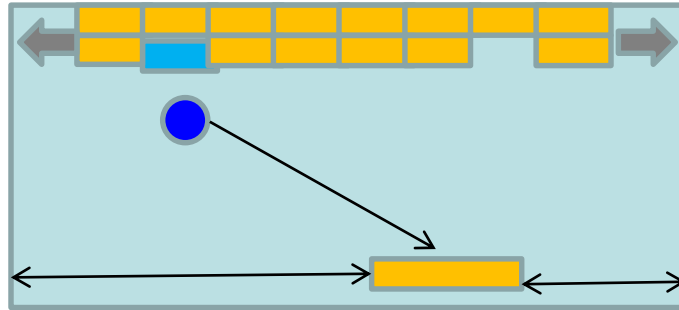
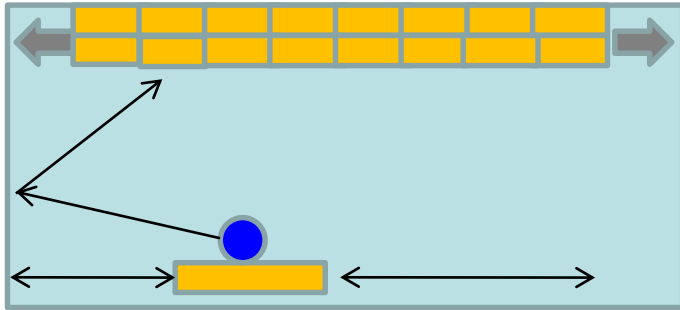


유턴된 방향으로 계속 이동한다.

### • 벽돌 깨기 게임 만들기

- **화면의 상단에 3\*10 개의 벽돌이** 있다. 벽돌들은 화면에 딱 차지않고 양쪽에 공간이 있다.
  - 벽돌들은 시간에 따라 좌우로 왔다갔다한다.
  - 벽돌의 칸과 줄의 숫자는 변경 가능하다.
- **화면의 하단에 bar(바)**가 있고 마우스를 이용하여 바를 움직인다.
  - 바닥의 bar(바)를 마우스로 선택하고 드래그하여 이동한다. (좌우로만 이동)
- 바 위에 공이 있고 시작 명령어를 누르면 공이 튀기기 시작한다. 공은 좌우의 벽과 위의 벽돌에 부딪치면 이동 방향을 바꾼 후 계속 튕긴다.
  - 아래의 바에 부딪히면 위쪽으로 바뀌고 튕긴다.
  - 튕기는 방향은 좌우상하가 아니라 대각선 (비스듬한 방향)으로 설정한다.
  - 아래 쪽에서 바가 공을 놓치면 공은 아래로 사라지고 바 위에 다시 나타나서 다시 튕기기 시작한다.
- **공이 한 벽돌에 한 번 부딪치면 그 벽돌의 색이 바뀌며, 부딪친 벽돌은 크기가 작아지며 화면에서 사라진다.**
  - 부딪치는 횟수는 개발자가 결정한다.
  - 바뀌는 색은 랜덤하게 설정한다.
- **잠시 멈추기 명령어를 누르면 (p 명령어)** 색이 변한 벽돌의 개수와 없어진 개수를 화면에 출력한다.
- **키보드 명령어**
  - **s/S**: 공 튀기기 시작
  - **p/P**: 움직임이 잠시 멈춤/다시 시작
  - **+/-** 입력: 공의 이동 속도가 늘어난다.
  - **n/N**: 게임 리셋
  - **q/Q**: 프로그램 종료

## 실습 3-2



## 3장 학습 내용

- 학습내용
  - 마우스 이벤트 다루기
    - 마우스 누를 때, 떼를 때, 움직일 때 발생 이벤트
  - 타이머 이벤트 다루기
    - 애니메이션 만들기
  - 래스터 연산