쉽게 배우는 MFC 윈도우 프로그래밍

Visual C++ 2019

2015·2017 버전 사용 가능

5장. 마우스와 키보드

목차

마우스 기초

마우스 처리

- 윈도우 운영체제는 마우스와 관련된 모든 변화를 메시지 형태로 프로그램에 전달한다.

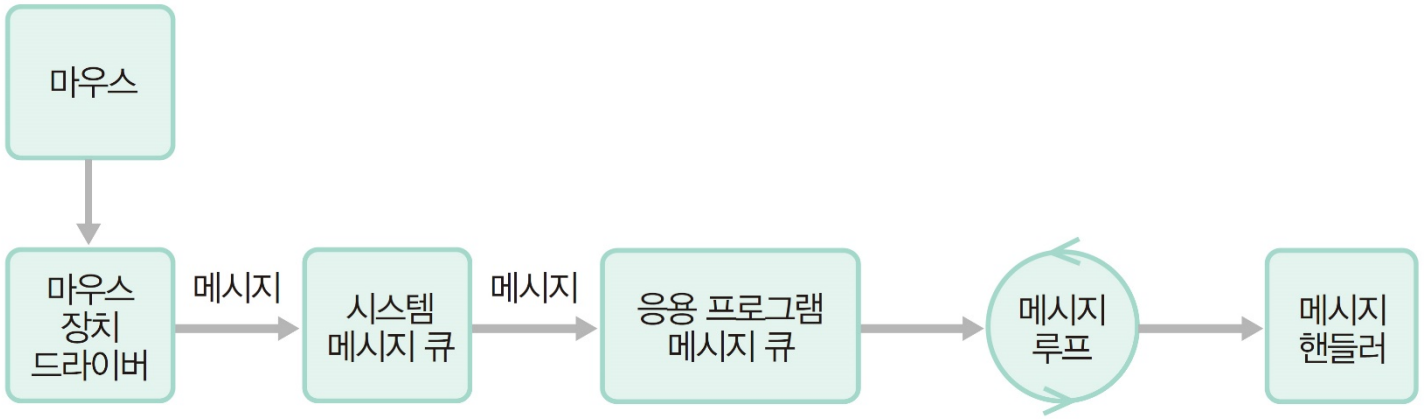


그림 5-1 윈도우의 마우스 메시지 처리

마우스 기초

마우스 메시지 전달

- 마우스 메시지는 원칙적으로 마우스 커서 밑에 있는 윈도우가 받는다.

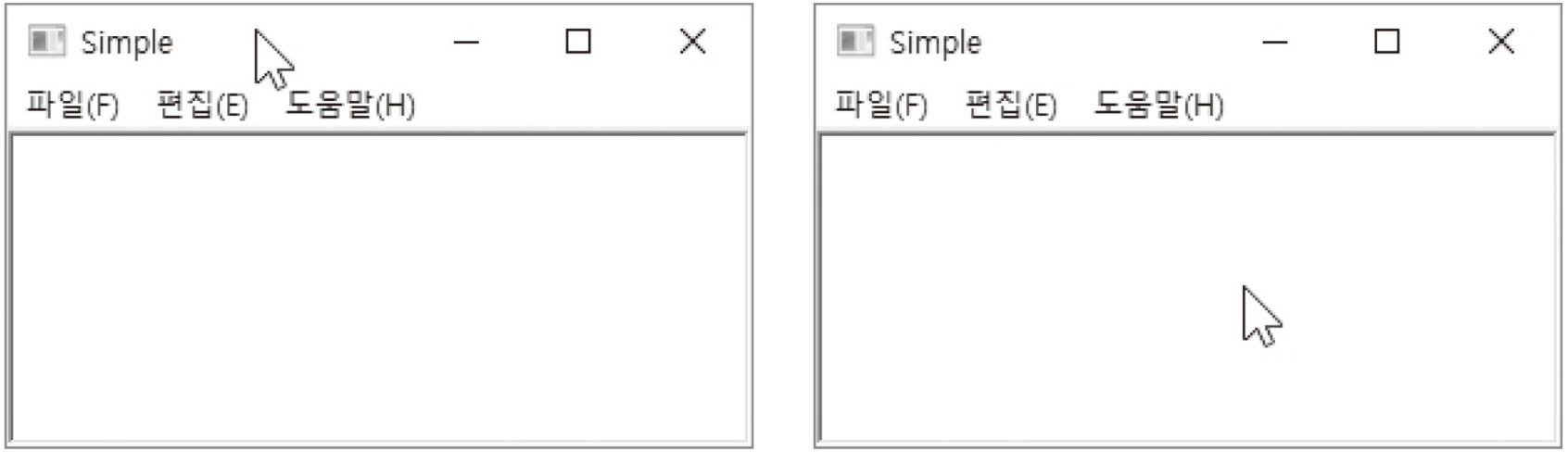


그림 5-2 마우스 커서와 윈도우 메시지

클라이언트 영역 마우스 메시지

표 5-1 클라이언트 영역 마우스 메시지

메시지	발생 시점
WM_LBUTTONDOWN	마우스 왼쪽 버튼을 누를 때
WM_LBUTTONUP	마우스 왼쪽 버튼을 뗄 때
WM_LBUTTONDBLCLK	마우스 왼쪽 버튼을 더블 클릭할 때
WM_MBUTTONDOWN	마우스 가운데 버튼을 누를 때
WM_MBUTTONUP	마우스 가운데 버튼을 뗄 때
WM_MBUTTONDBLCLK	마우스 가운데 버튼을 더블 클릭할 때
WM_RBUTTONDOWN	마우스 오른쪽 버튼을 누를 때
WM_RBUTTONUP	마우스 오른쪽 버튼을 뗄 때
WM_RBUTTONDBLCLK	마우스 오른쪽 버튼을 더블 클릭할 때
WM_MOUSEMOVE	마우스를 움직일 때

클라이언트 영역 마우스 메시지

- 메시지 발생순서 비교

그림 5-3 마우스 왼쪽 버튼 두 번 클릭과 더블 클릭 메시지 발생 순서

클라이언트 영역 마우스 메시지

표 5-3 클라이언트 영역 마우스 메시지 핸들러

메시지	메시지 맵 매크로	메시지 핸들러
WM_LBUTTONDOWN	ON_WM_LBUTTONDOWN()	OnLButtonDown()
WM_LBUTTONUP	ON_WM_LBUTTONUP()	OnLButtonUp()
WM_LBUTTONDBLCLK	ON_WM_LBUTTONDBLCLK()	OnLButtonDblClk()
WM_MBUTTONDOWN	ON_WM_MBUTTONDOWN()	OnMButtonDown()
WM_MBUTTONUP	ON_WM_MBUTTONUP()	OnMButtonUp()
WM_MBUTTONDBLCLK	ON_WM_MBUTTONDBLCLK()	OnMButtonDblClk()
WM_RBUTTONDOWN	ON_WM_RBUTTONDOWN()	OnRButtonDown()
WM_RBUTTONUP	ON_WM_RBUTTONUP()	OnRButtonUp()
WM_RBUTTONDBLCLK	ON_WM_RBUTTONDBLCLK()	OnRButtonDblClk()
WM_MOUSEMOVE	ON_WM_MOUSEMOVE()	OnMouseMove()

클라이언트 영역 마우스 메시지

- 메시지 핸들러 형태

```
afx_msg void On*(UINT nFlags, CPoint point);
```

  - afx_msg : 내부적으로 공백으로 처리되며 이 함수가 메시지 핸들러임을 나타냄

클라이언트 영역 마우스 메시지

메시지 핸들러 형태

```
afx_msg void On*(UINT nFlags, CPoint point);
```

- nFlags : 메시지가 생성될 당시의 키보드나 마우스 버튼의 상태를 나타냄

표 5-3 nFlags 관련 비트 마스크

비트 마스크	의미
MK_CONTROL	Ctrl 누름
MK_SHIFT	Shift 누름
MK_LBUTTON	마우스 왼쪽 버튼 누름
MK_MBUTTON	마우스 가운데 버튼 누름
MK_RBUTTON	마우스 오른쪽 버튼 누름

클라이언트 영역 마우스 메시지

- 메시지 핸들러 형태

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
   if (nFlags & MK_SHIFT){ // [Shift] 키가 눌렸다면
   ...
   }
}
```

클라이언트 영역 마우스 메시지

메시지 핸들러 형태

```
afx_msg void On*(UINT nFlags, CPoint point);
```

- nFlags : 메시지가 생성될 당시의 키보드나 마우스 버튼의 상태를 나타냄

표 5-3 nFlags 관련 비트 마스크

비트 마스크	의미
MK_CONTROL	Ctrl 누름
MK_SHIFT	Shift 누름
MK_LBUTTON	마우스 왼쪽 버튼 누름
MK_MBUTTON	마우스 가운데 버튼 누름
MK_RBUTTON	마우스 오른쪽 버튼 누름

클라이언트 영역 마우스 메시지

- 메시지 핸들러 형태

```
afx_msg void On*(UINT nFlags, CPoint point);
```

  - point : 메시지가 생성될 당시의 마우스 커서 위치를 나타내며, 클라이언트 좌표(클라이언트 영역의 좌상단을 기준으로 한 픽셀 좌표)로 되어 있음

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
   CClientDC dc(this);
   dc.SetMapMode(MM_LOMETRIC); // 매핑 모드를 변경한다.
   CPoint pt = point; // point 객체를 복사한다.
   dc.DPtoLP(&pt); // 장치 좌표를 논리 좌표로 변환한다.
   dc.Rectangle(pt.x - 100, pt.y + 100, pt.x + 100, pt.y - 100);
}
```

[실습 5-1] 클라이언트 영역 마우스 메시지 처리하기

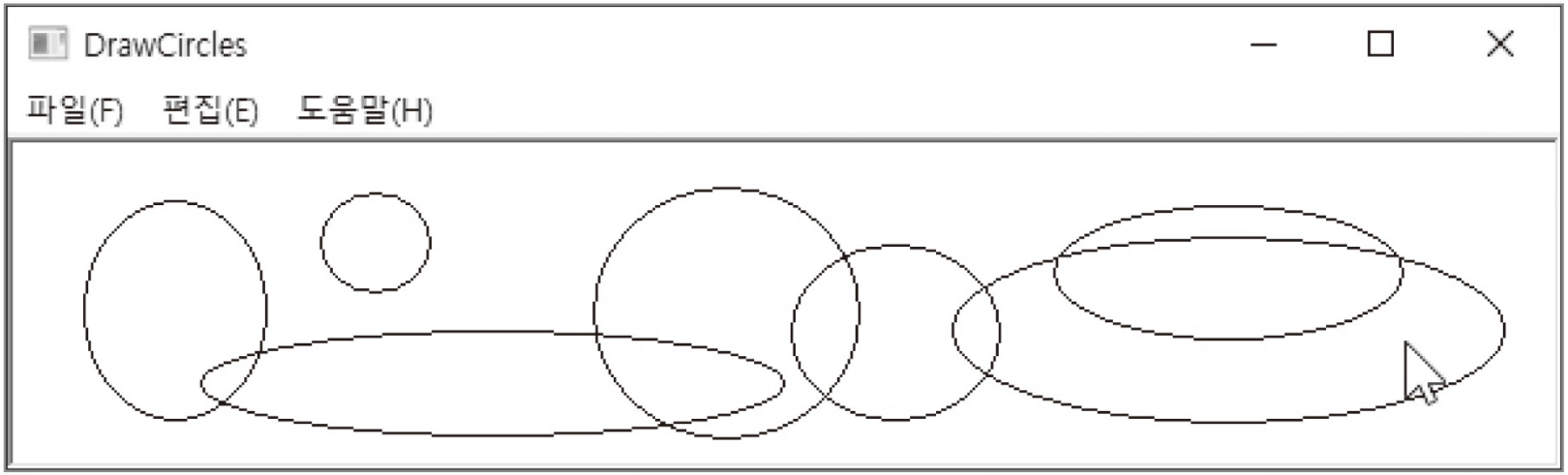


그림 5-4 실행 결과

[실습 5-1] 클라이언트 영역 마우스 메시지 처리하기

```
class CChildView : public CWnd
{
// 생성입니다.
public:
   CChildView();
// 특성입니다.
public:
   BOOL m_bDrawMode; // 그리기 작업이 진행 중임을 나타낸다.
   int m_x1, m_y1, m_x2, m_y2; // 타원에 외접하는 직사각형의 좌상단/우하단 좌표
```

[실습 5-1] 클라이언트 영역 마우스 메시지 처리하기

```
CChildView::CChildView()
{
   m_bDrawMode = FALSE;
}
```

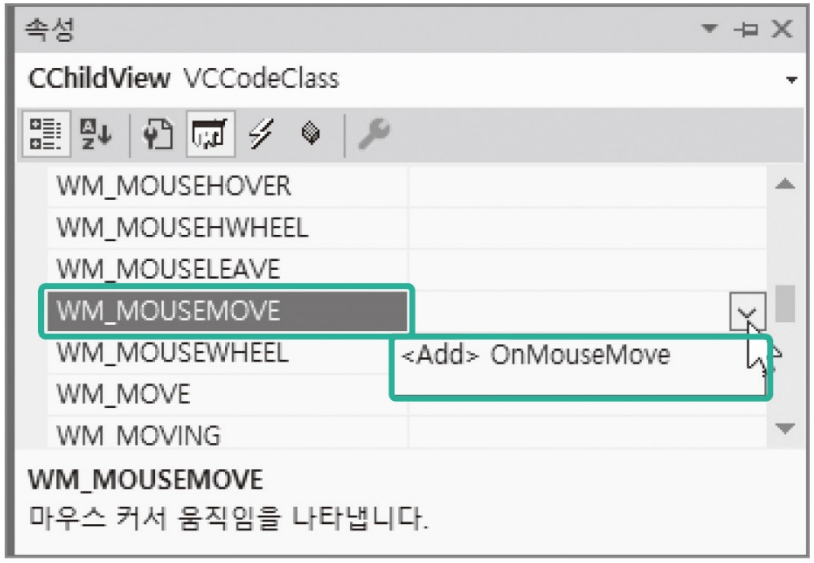


그림 5-5 마우스 메시지 핸들러 추가

[실습 5-1] 클라이언트 영역 마우스 메시지 처리하기

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
   // 그리기 모드를 시작한다.
   m_bDrawMode = TRUE;
   // 좌표를 저장한다.
   m_x1 = m_x2 = point.x;
   m_y1 = m_y2 = point.y;
}
void CChildView::OnMouseMove(UINT nFlags, CPoint point)
{
   // 그리기 모드이면 타원을 지우고 그리기를 반복한다.
   if(m_bDrawMode){
      CClientDC dc(this);
      dc.SelectStockObject(NULL_BRUSH);
      // 이전에 그린 타원을 지운다.
      dc.SetROP2(R2_NOT);
      dc.Ellipse(m_x1, m_y1, m_x2, m_y2);
```

[실습 5-1] 클라이언트 영역 마우스 메시지 처리하기

```
        // 새로운 타원을 그린다.
        dc.SetROP2(R2_NOT);
        m_x2 = point.x;
        m_y2 = point.y;
        dc.Ellipse(m_x1, m_y1, m_x2, m_y2);
    }
}
void CChildView::OnLButtonUp(UINT nFlags, CPoint point)
{
    CClientDC dc(this);
    dc.SelectStockObject(NULL_BRUSH);
    // 최종적인 타원을 그린다.
    dc.SetROP2(R2_COPYPEN);
    m_x2 = point.x;
    m_y2 = point.y;
    dc.Ellipse(m_x1, m_y1, m_x2, m_y2);
    // 그리기 모드를 끝낸다.
    m_bDrawMode = FALSE;
}
```

클라이언트 영역 마우스 메시지

■ 마우스 캡처

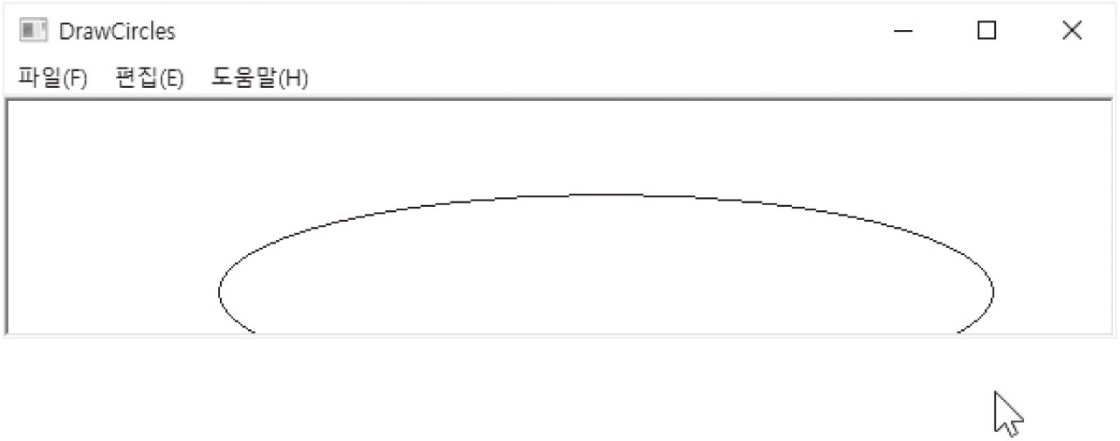


그림 5-6 클라이언트 영역 밖으로 확장된 타원

표 5-4 마우스 캡처 관련 함수

API 함수	MFC 함수	의미
SetCapture()	CWnd::SetCapture()	마우스 캡처를 시작한다.
ReleaseCapture()	없음	마우스 캡처를 해제한다.
GetCapture()	CWnd::GetCapture()	어느 윈도우가 현재 마우스를 캡처하고 있는지 알아낸다.

[실습 5-2] 마우스 캡처 활용하기

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
   // 마우스 캡처를 시작한다(MFC 함수 사용).
   SetCapture();
   // 그리기 모드를 시작한다.
   m_bDrawMode = TRUE;
   ...
}
...
void CChildView::OnLButtonUp(UINT nFlags, CPoint point)
{
   ...
   // 그리기 모드를 끝낸다.
   m_bDrawMode = FALSE;
   Chapter 05 마우스와 키보드 235
   // 마우스 캡처를 해제한다(API 함수 사용).
   ::ReleaseCapture();
}
```

비클라이언트 영역 마우스 메시지

■ 비클라이언트 영역 마우스 메시지

표 5-5 비클라이언트 영역 마우스 메시지

메시지	발생 시점
WM_NCLBUTTONDOWN	왼쪽 버튼을 누를 때
WM_NCLBUTTONUP	왼쪽 버튼을 뗄 때
WM_NCLBUTTONDBLCLK	왼쪽 버튼을 더블 클릭할 때
WM_NCMBUTTONDOWN	가운데 버튼을 누를 때
WM_NCMBUTTONUP	가운데 버튼을 뗄 때
WM_NCMBUTTONDBLCLK	가운데 버튼을 더블 클릭할 때
WM_NCRBUTTONDOWN	오른쪽 버튼을 누를 때
WM_NCRBUTTONUP	오른쪽 버튼을 뗄 때
WM_NCRBUTTONDBLCLK	오른쪽 버튼을 더블 클릭할 때
WM_NCMOUSEMOVE	마우스를 움직일 때

비클라이언트 영역 마우스 메시지

■ 비클라이언트 영역 마우스 메시지 핸들러

표 5-6 비클라이언트 영역 마우스 메시지 핸들러

메시지	메시지 맵 매크로	메시지 핸들러
WM_NCLBUTTONDOWN	ON_WM_NCLBUTTONDOWN()	OnNcLButtonDown()
WM_NCLBUTTONUP	ON_WM_NCLBUTTONUP()	OnNcLButtonUp()
WM_NCLBUTTONDBLCLK	ON_WM_NCLBUTTONDBLCLK()	OnNcLButtonDblClk()
WM_NCMBUTTONDOWN	ON_WM_NCMBUTTONDOWN()	OnNcMButtonDown()
WM_NCMBUTTONUP	ON_WM_NCMBUTTONUP()	OnNcMButtonUp()
WM_NCMBUTTONDBLCLK	ON_WM_NCMBUTTONDBLCLK()	OnNcMButtonDblClk()
WM_NCRBUTTONDOWN	ON_WM_NCRBUTTONDOWN()	OnNcRButtonDown()
WM_NCRBUTTONUP	ON_WM_NCRBUTTONUP()	OnNcRButtonUp()
WM_NCRBUTTONDBLCLK	ON_WM_NCRBUTTONDBLCLK()	OnNcRButtonDblClk()
WM_NCMOUSEMOVE	ON_WM_NCMOUSEMOVE()	OnNcMouseMove()

비클라이언트 영역 마우스 메시지

메시지 핸들러 형태

```
afx_msg void OnNc*(UINT nHitTest, CPoint point);
```

- nHitTest : 메시지가 생성될 당시의 마우스 커서 위치를 나타내는 상숫값
- point : 메시지가 생성될 당시의 마우스 커서 좌표를 나타냄

표 5-7 nHitTest

상수값	의미
HTCAPTION	타이틀바
HTCLIENT	클라이언트 영역
HTCLOSE	종료 버튼
HTHSCROLL	가로 스크롤바
HTMENU	메뉴
HTMAXBUTTON 또는 HTZOOM	최대화 버튼
HTMINBUTTON 또는 HTREDUCE	최소화 버튼
HTSYSMENU	시스템 메뉴
HTVSCROLL	세로 스크롤바

[실습 5-3] 비클라이언트 영역 마우스 메시지 처리하기

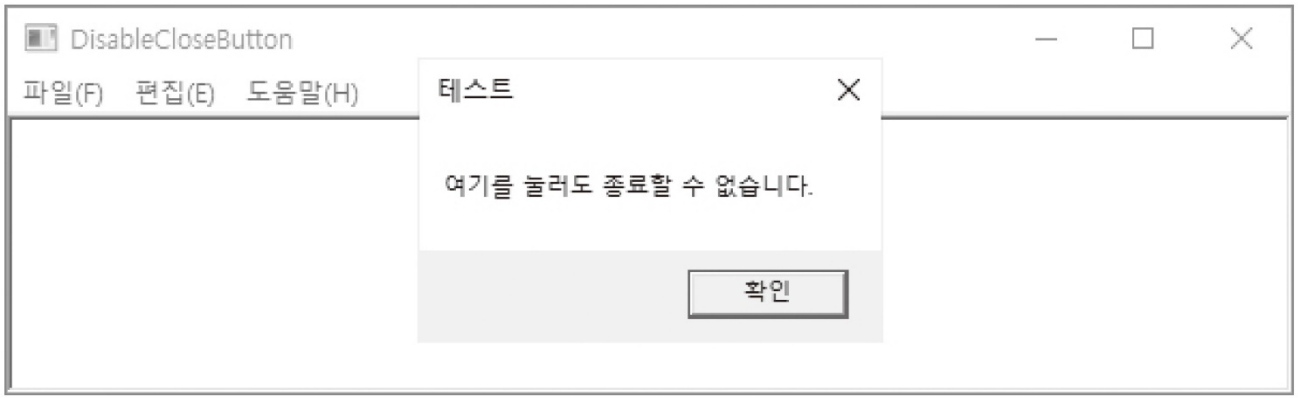


그림 5-7 실행 결과

[실습 5-3] 비클라이언트 영역 마우스 메시지 처리하기

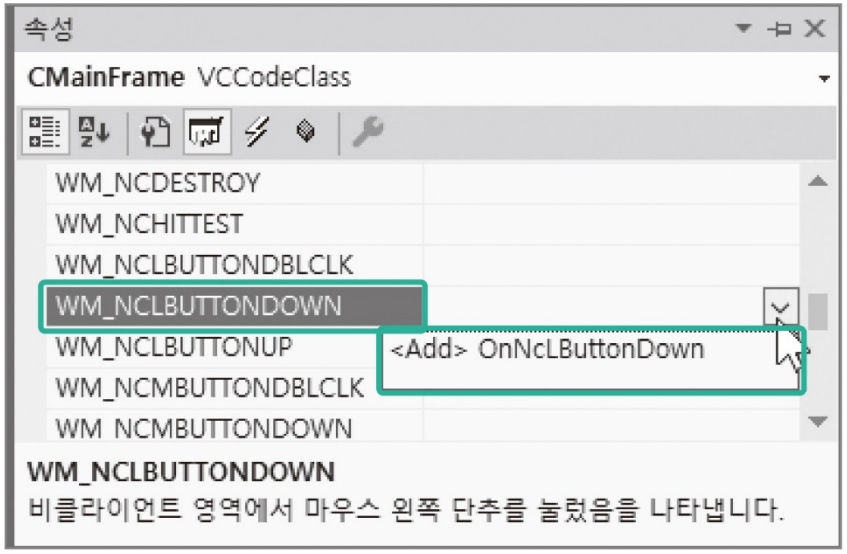


그림 5-8 마우스 메시지 핸들러 추가

[실습 5-3] 비클라이언트 영역 마우스 메시지 처리하기

```
void CMainFrame::OnNcLButtonDown(UINT nHitTest, CPoint point)
{
   // 종료 버튼을 누른 경우에만 특별한 처리를 한다.
   if(nHitTest == HTCLOSE)
      MessageBox(_T("여기를 눌러도 종료할 수 없습니다."), _T("테스트"));
   // 그 밖의 경우에는 운영체제가 자동으로 처리한다.
   else
      CFrameWnd::OnNcLButtonDown(nHitTest, point);
}
```

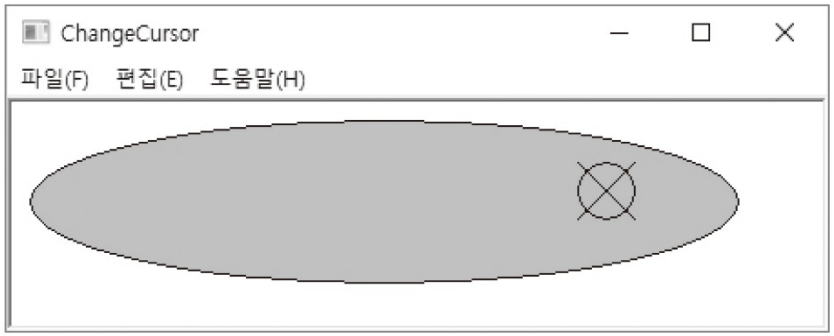
마우스 커서 변경과 위치 추적

마우스 커서 변경

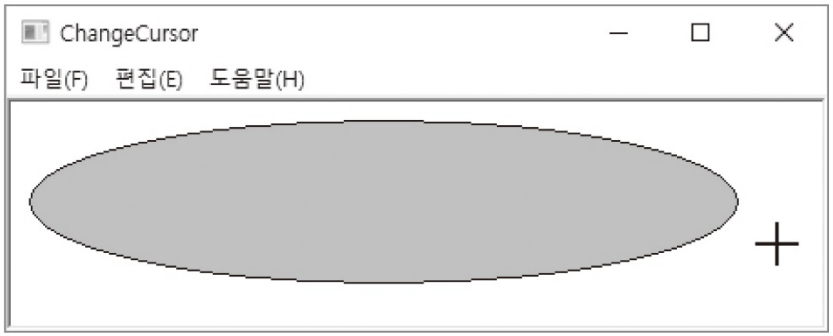
```
HCURSOR SetCursor(HCURSOR hCursor);
```

- hCursor
  - 커서 리소스를 가리키는 핸들값
  - 다음 함수의 리턴값을 대입
    - ✓ CWinApp::LoadStandardCursor()
    - ✓ CWinApp::LoadCursor()

[실습 5-4] 마우스 커서 변경하기



(a) 커서가 타원 안쪽에 위치할 때



(b) 커서가 타원 바깥쪽에 위치할 때

그림 5-9 실행 결과

[실습 5-4] 마우스 커서 변경하기

그림 5-10 커서 리소스 추가

[실습 5-4] 마우스 커서 변경하기

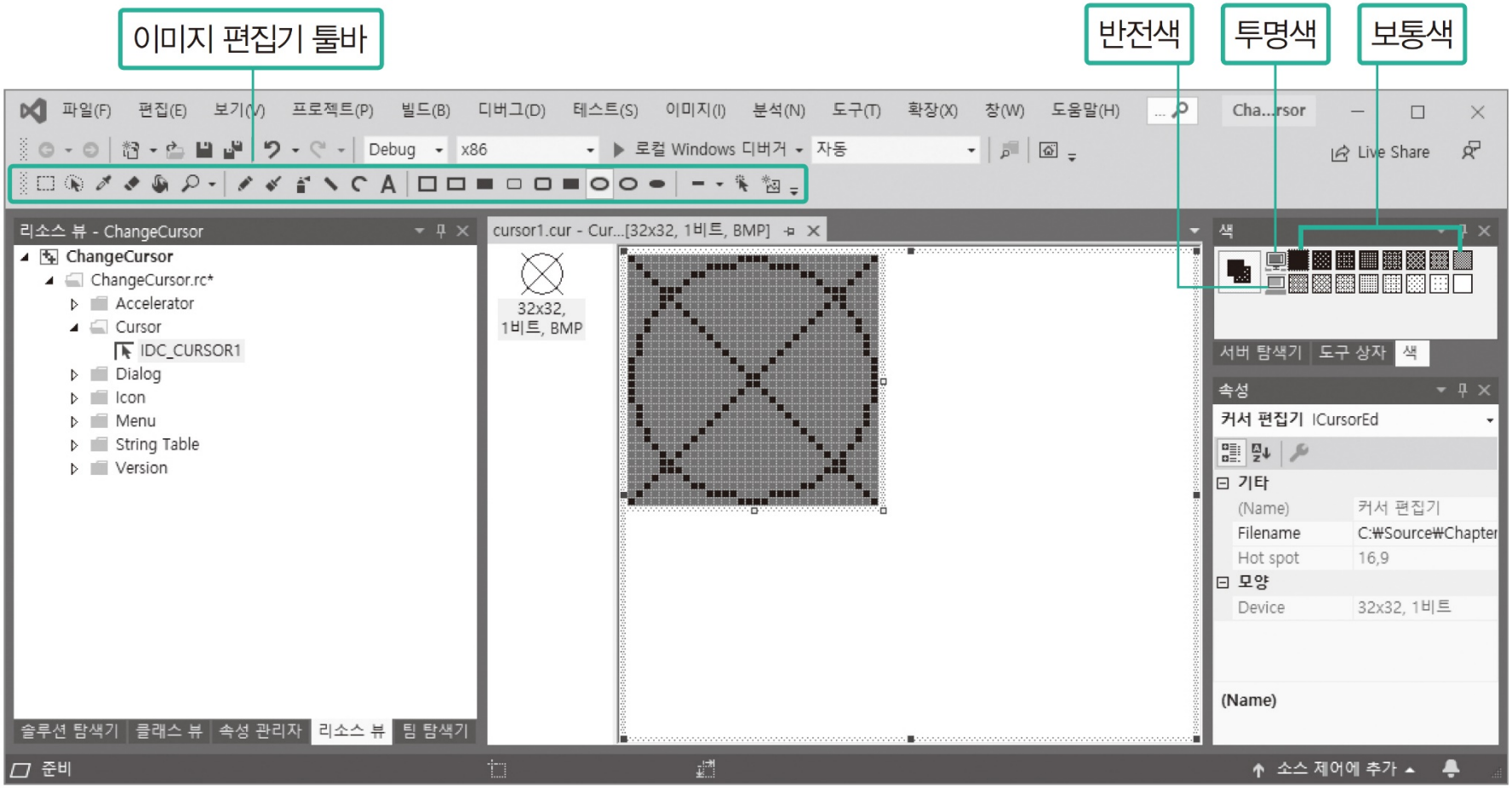


그림 5-11 커서 디자인

[실습 5-4] 마우스 커서 변경하기

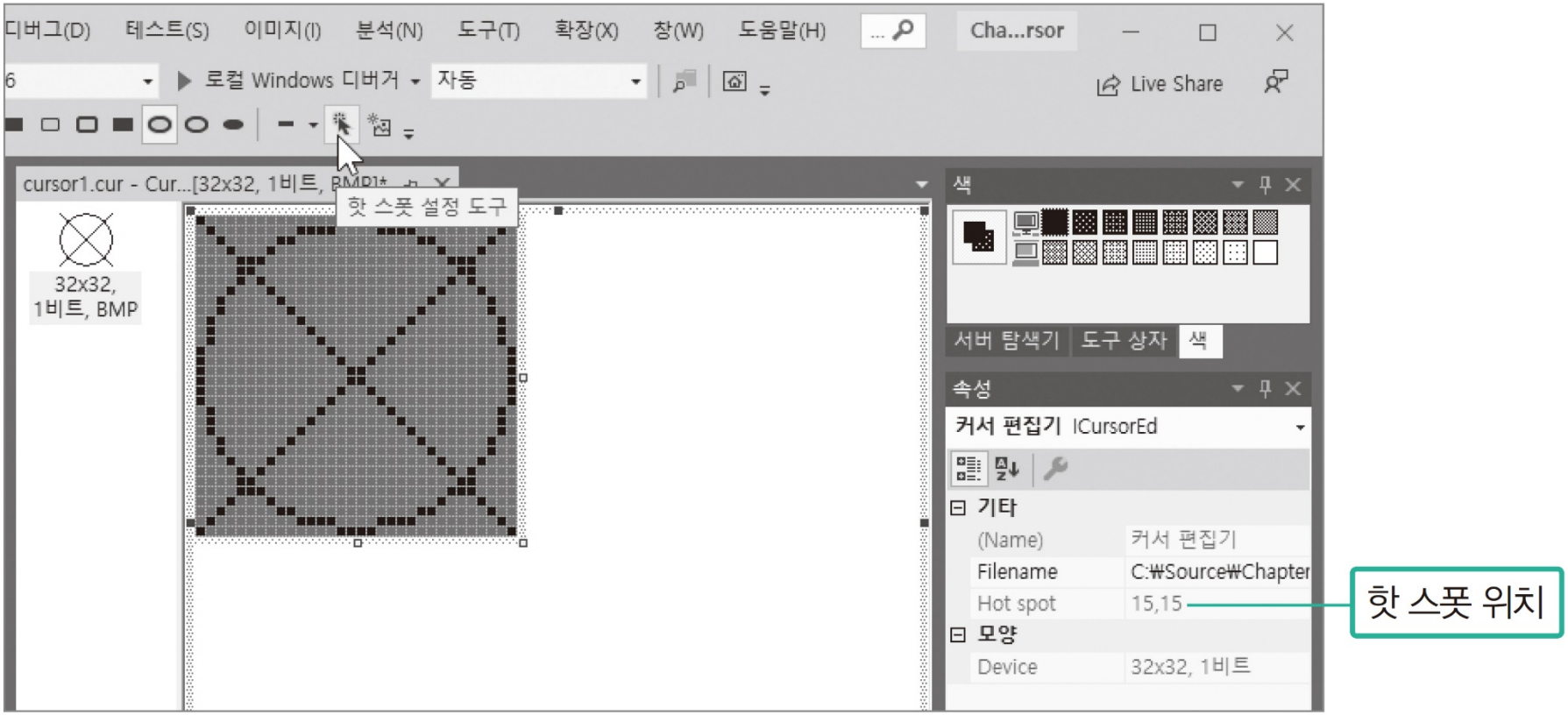


그림 5-12 핫 스폿 설정

[실습 5-4] 마우스 커서 변경하기

```
void CChildView::OnPaint()
{
   CPaintDC dc(this);
   dc.SelectStockObject(LTGRAY_BRUSH);
   dc.Ellipse(10, 10, 400, 100);
}
```

[실습 5-4] 마우스 커서 변경하기

```
BOOL CChildView::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message)
{
   // 클라이언트 영역이면 커서를 변경한다.
   if (nHitTest == HTCLIENT) {
      CPoint point;
      ::GetCursorPos(&point); // 커서 위치(스크린 좌표)를 얻는다.
      ScreenToClient(&point); // 스크린 좌표를 클라이언트 좌표로 변환한다.
      CRgn rgn;
      rgn.CreateEllipticRgn(10, 10, 400, 100); // 타원형 리전을 생성한다.
      if (rgn.PtInRegion(point)) // 커서가 리전 안쪽에 있는지 확인한다.
         // 사용자 정의 커서로 변경한다.
         ::SetCursor(AfxGetApp()->LoadCursor(IDC_CURSOR1));
      else
         // 표준 커서 중 하나로 변경한다.
         ::SetCursor(AfxGetApp()->LoadStandardCursor(IDC_CROSS));
      return TRUE;
   }

   // 클라이언트 영역이 아니면 운영체제가 자동으로 처리한다.
   return CWnd::OnSetCursor(pWnd, nHitTest, message);
}
```

마우스 커서 변경과 위치 추적

- 마우스 커서 위치 추적

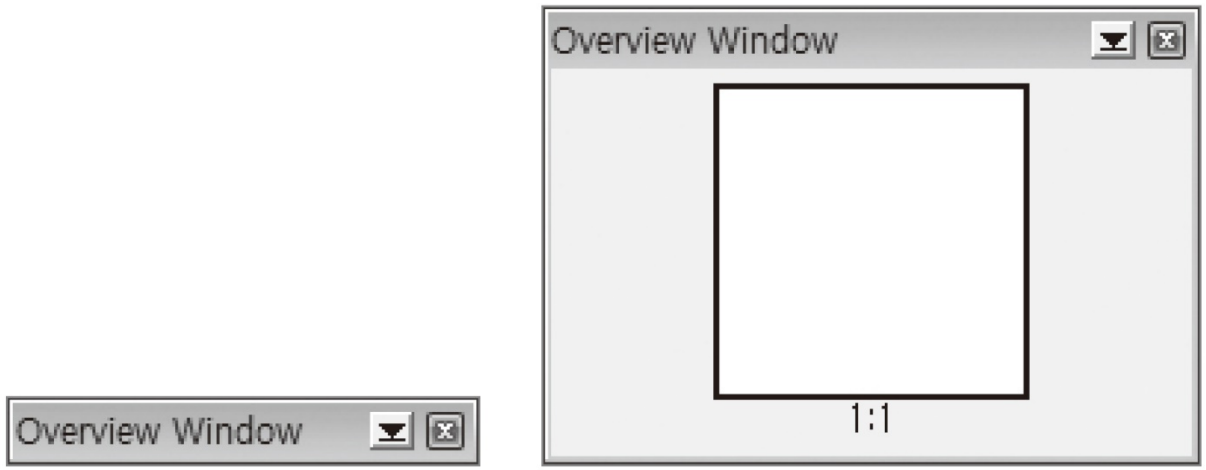


그림 5-13 커서 위치 추적이 필요한 경우

마우스 커서 변경과 위치 추적

- 마우스 커서 위치 추적
  - ::TrackMouseEvent() 함수 원형

```
HCURSOR SetCursor(HCURSOR hCursor);
```

  - TRACKMOUSEEVENT 구조체 정의

```
typedef struct tagTRACKMOUSEEVENT {
    DWORD cbSize;
    DWORD dwFlags;
    HWND hwndTrack;
    DWORD dwHoverTime;
} TRACKMOUSEEVENT, *LPTRACKMOUSEEVENT;
```

  - 주의점 : ::TrackMouseEvent() 함수로 요청한 WM_MOUSELEAVE 메시지는 한 번만 발생하므로 필요 시 ::TrackMouseEvent() 함수를 다시 호출해야 함

[실습 5-5] 마우스 커서 위치 추적하기

(a) 마우스 커서가 내부에 있을 때 (b) 마우스 커서가 외부에 있을 때

그림 5-14 실행 결과

[실습 5-5] 마우스 커서 위치 추적하기

```
class CChildView : public CWnd
{
// 생성입니다.
public:
   CChildView();
// 특성입니다.
public:
   BOOL m_bMouseIn;
```

[실습 5-5] 마우스 커서 위치 추적하기

```
CChildView::CChildView()
{
   m_bMouseIn = FALSE;
}
```

[실습 5-5] 마우스 커서 위치 추적하기

```
void CChildView::OnMouseMove(UINT nFlags, CPoint point)
{
   if (m_bMouseIn == FALSE) {
      // 마우스 커서 추적을 요청한다.
      TRACKMOUSEEVENT tme;
      tme.cbSize = sizeof(tme);
      tme.dwFlags = TME_LEAVE;
      tme.hwndTrack = this->m_hWnd;
      tme.dwHoverTime = HOVER_DEFAULT;
      ::TrackMouseEvent(&tme);

      // 메인 윈도우 크기를 300*150으로 변경한다.
      CWnd* pMainWnd = AfxGetMainWnd();
      CRect rect;
      pMainWnd->GetWindowRect(&rect);
      rect.right = rect.left + 300;
      rect.bottom = rect.top + 150;
      pMainWnd->MoveWindow(&rect);

      // 마우스 커서가 클라이언트 영역에 있음을 기억해둔다.
      m_bMouseIn = TRUE;
   }
}
```

[실습 5-5] 마우스 커서 위치 추적하기

```
void CChildView::OnMouseLeave()
{
   // 마우스 커서가 클라이언트 영역 밖에 있음을 기억해둔다.
   m_bMouseIn = FALSE;

   // 메인 윈도우 크기를 200*100으로 변경한다.
   CWnd* pMainWnd = AfxGetMainWnd();
   CRect rect;
   pMainWnd->GetWindowRect(&rect);
   rect.right = rect.left + 200;
   rect.bottom = rect.top + 100;
   pMainWnd->MoveWindow(&rect);
}
```

[실습 5-5] 마우스 커서 위치 추적하기

```
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
   if (!CFrameWnd::PreCreateWindow(cs))
      return FALSE;
   cs.cx = 200;
   cs.cy = 100;
   cs.dwExStyle &= ~WS_EX_CLIENTEDGE;
   cs.lpszClass = AfxRegisterWndClass(0);
   return TRUE;
}
```

키보드 다루기

윈도우의 키보드 메시지 처리

- 윈도우 운영체제는 키보드와 관련된 모든 이벤트를 프로그램에 메시지 형태로 전달

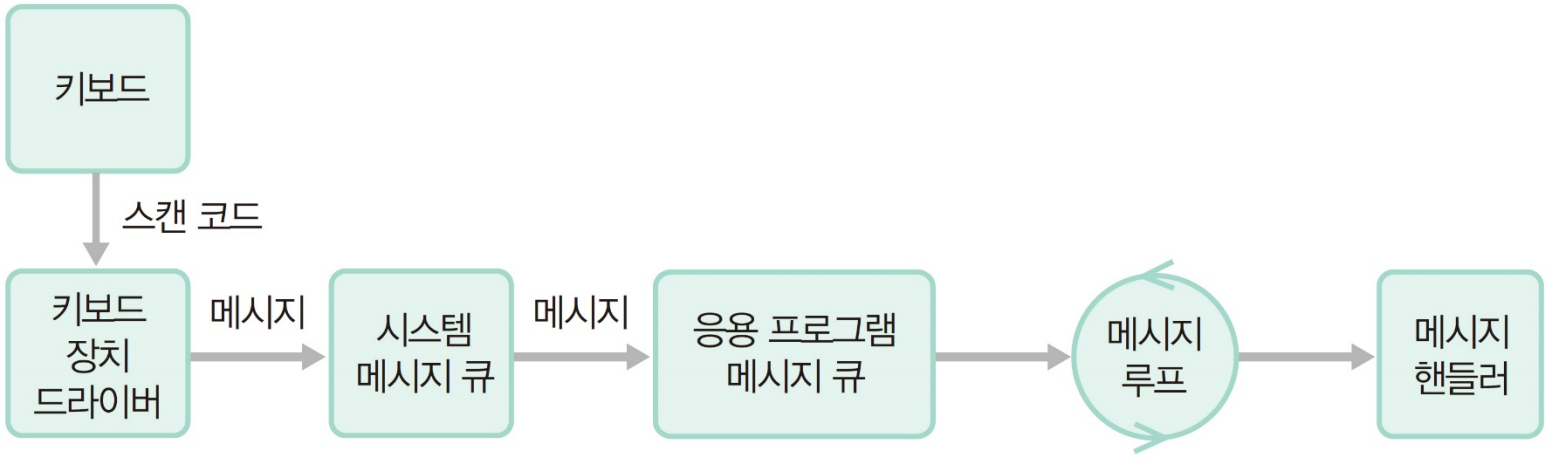


그림 5-15 윈도우의 키보드 메시지 처리

키보드 다루기

키보드 메시지 전달

- 키보드 메시지는 키보드 포커스를 가진 윈도우가 받음

키보드 포커스

- 활성 윈도우 또는 활성 윈도우의 자식 윈도우가 가지는 일종의 속성

그림 5-16 키보드 포커스 표시 유형

키보드 포커스

■ 키보드 포커스 변화

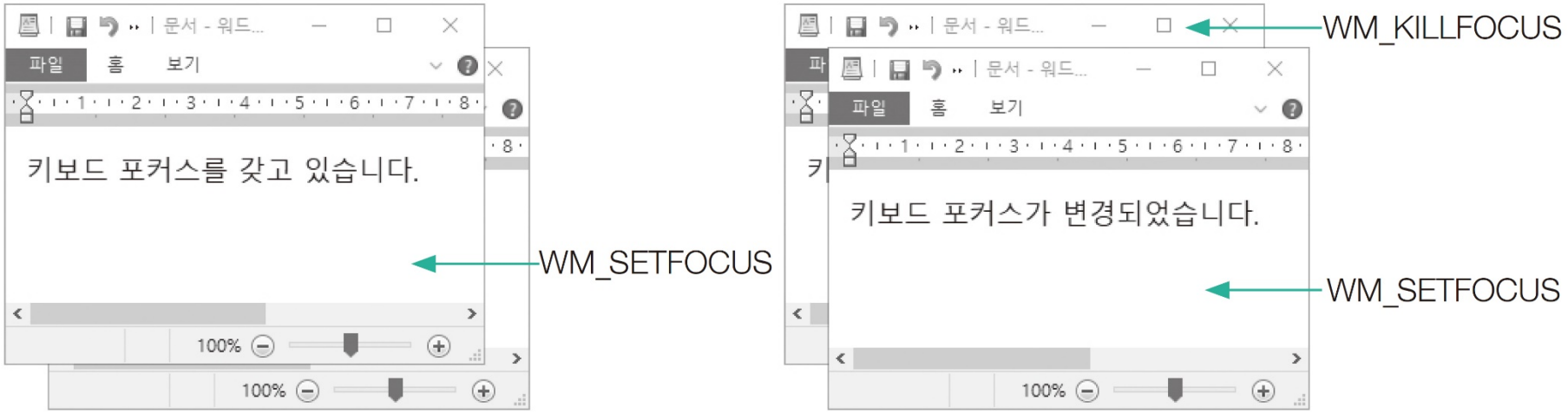


그림 5-17 키보드 포커스 변화

```
void CMainFrame::OnSetFocus(CWnd* /*pOldWnd*/)
{
    // 뷰 창으로 포커스를 이동합니다.
    m_wndView.SetFocus();
}
```

키보드 포커스

■ 캐럿 함수 - MFC

표 5-8 캐럿 함수 – MFC

함수 이름	기능
CreateCaret()	비트맵을 이용하여 캐럿을 생성한다.
CreateGrayCaret()	회색 직사각형 캐럿을 생성한다.
CreateSolidCaret()	검은색 직사각형 캐럿을 생성한다.
ShowCaret()	캐럿을 보인다.
HideCaret()	캐럿을 숨긴다.
GetCaretPos()	캐럿의 위치(클라이언트 좌표)를 얻는다.
SetCaretPos()	캐럿의 위치(클라이언트 좌표)를 설정한다.

키보드 포커스

■ 캐럿 함수 - API

표 5-9 캐럿 함수 – API

함수 이름	기능
::DestroyCaret()	캐럿을 파괴한다.
::GetCaretBlinkTime()	캐럿이 깜박이는 간격을 얻는다.
::SetCaretBlinkTime()	캐럿이 깜박이는 간격을 설정한다.

[실습 5-6] 캐럿 사용하기

그림 5-18 실행 결과

[실습 5-6] 캐럿 사용하기

```
void CChildView::OnSetFocus(CWnd* pOldWnd)
{
   CreateSolidCaret(20, 20); // 캐럿을 생성한다.
   SetCaretPos(CPoint(50, 50)); // 캐럿의 위치를 설정한다.
   ShowCaret( ) ; // 캐럿을 화면에 보인다.
}

void CChildView::OnKillFocus(CWnd* pNewWnd)
{
   HideCaret( ) ; // 캐럿을 숨긴다.
   ::DestroyCaret( ) ; // 캐럿을 파괴한다.
}
```

문자메시지

■ 키 누름 메시지(Keystroke Message)

- 키보드를 누르거나 떼는 동작에 의해 발생하는 메시지

■ 키 누름 메시지 종류

표 5-10 키 누름 메시지

메시지	발생 시점
WM_KEYDOWN	F10, Alt 이외의 키를 누를 때
WM_KEYUP	F10, Alt 이외의 키를 뗄 때
WM_SYSKEYDOWN	F10, Alt, Alt +[다른 키]를 누를 때
WM_SYSKEYUP	F10, Alt, Alt +[다른 키]를 뗄 때

문자 메시지

- 키 누름 메시지 핸들러 형태

```
afx_msg void On*(UINT nChar, UINT nRepCnt, UINT nFlags);
```

  - nChar : 키에 할당된 가상 키 코드값을 가짐
  - nRepCnt : 키를 계속 누르고 있을 경우 1보다 큰 값을 가질 수 있음
  - nFlags : 키와 관련된 부가적인(스캔 코드, 확장 키 여부, …) 정보를 담고 있음

[실습 5-7] 키 누름 메시지 처리하기

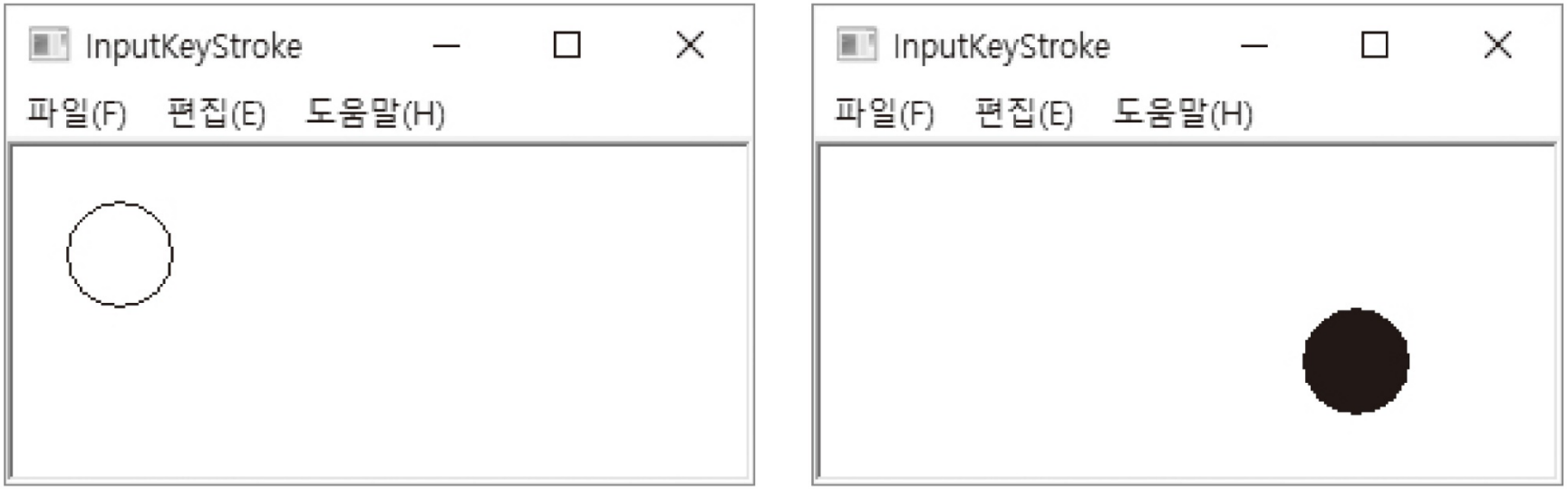


그림 5-19 실행 결과

[실습 5-7] 키 누름 메시지 처리하기

```
class CChildView : public CWnd
{
// 생성입니다.
public:
   CChildView();
// 특성입니다.
public:
   int m_xPos, m_yPos; // 도형의 현재 위치
   int m_xMax, m_yMax; // 클라이언트 영역의 크기
   BOOL m_bFill; // 도형의 내부를 채울지 여부
```

[실습 5-7] 키 누름 메시지 처리하기

```
CChildView::CChildView()
{
   m_xPos = m_yPos = 60; // 임의 값으로 초기화
   m_bFill = FALSE; // 도형 내부를 채우지 않음
}
```

[실습 5-7] 키 누름 메시지 처리하기

```
void CChildView::OnSize(UINT nType, int cx, int cy)
{
   m_xMax = cx;
   m_yMax = cy;
}
```

[실습 5-7] 키 누름 메시지 처리하기

```
void CChildView::OnPaint()
{
   CPaintDC dc(this);
   if(m_bFill == TRUE) dc.SelectStockObject(BLACK_BRUSH);
   dc.Ellipse(m_xPos - 20, m_yPos - 20, m_xPos + 20, m_yPos + 20);
}
```

[실습 5-7] 키 누름 메시지 처리하기

```
void CChildView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
   switch (nChar) {
   case VK_LEFT:
      m_xPos -= 20;
      break;
   case VK_RIGHT:
      m_xPos += 20;
      break;
   case VK_UP:
      m_yPos -= 20;
      break;
   case VK_DOWN:
      m_yPos += 20;
      break;
   case VK_SPACE:
      m_bFill = !m_bFill;
   }
   // 20 <= m_xPos <= m_xMax-20
   m_xPos = min(max(20, m_xPos), m_xMax - 20);
   // 20 <= m_yPos <= m_yMax-20
   m_yPos = min(max(20, m_yPos), m_yMax - 20);
   Invalidate();
}
```

문자 메시지

■ 문자 메시지 필요성

표 5-11 R 를 눌렀을 때 발생할 수 있는 문자

문자	가상 키 코드 조합
r	영문 입력 모드에서 R 또는 Caps Lock + Shift + R 을 누름
R	영문 입력 모드에서 Caps Lock + R 또는 Shift + R 을 누름
ㄱ	한글 입력 모드에서 R 을 누름
ㄲ	한글 입력 모드에서 Shift + R 을 누름

문자 메시지

- 문자 메시지 발생 시나리오

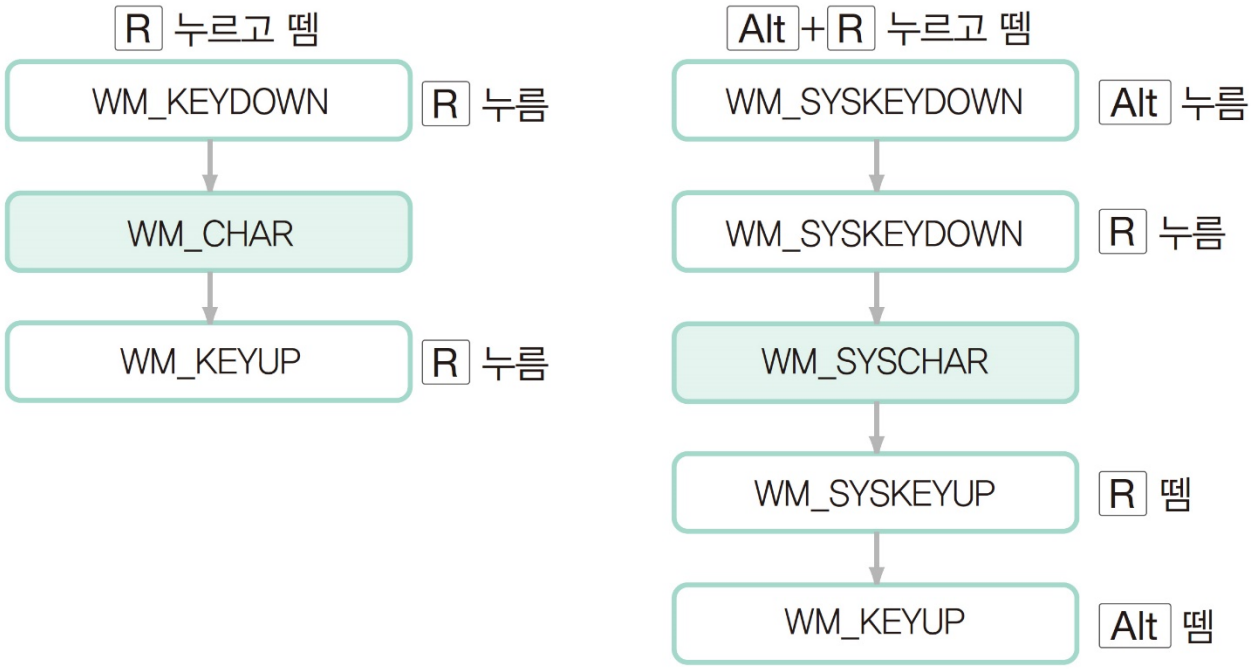


그림 5-20 문자 메시지

문자 메시지

- 문자 메시지 핸들러 형태

```
afx_msg void OnChar(UINT nChar, UINT nRepCnt, UINT nFlags);
afx_msg void OnSysChar(UINT nChar, UINT nRepCnt, UINT nFlags);
```

  - nChar : 키에 해당하는 문자 코드값을 가짐
  - nRepCnt : 키를 계속 누르고 있을 경우 1보다 큰 값을 가질 수 있음
  - nFlags : 키와 관련된 부가적인 정보(스캔 코드, 확장 키 여부, …)를 담고 있음

[실습 5-8] 문자 메시지 처리하기

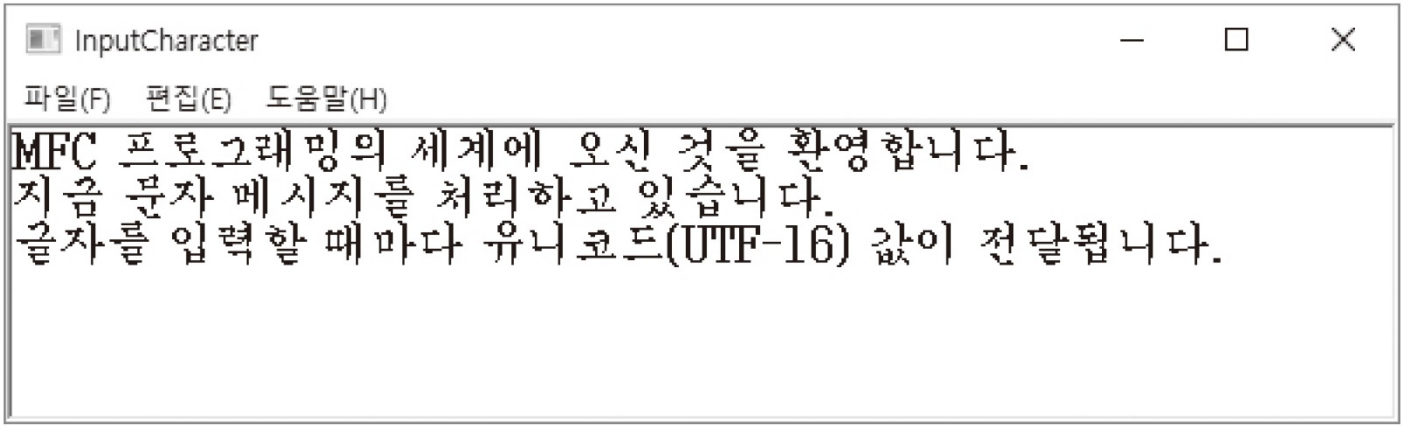


그림 5-21 실행 결과

[실습 5-8] 문자 메시지 처리하기

```
#include <afxtempl.h>


class CChildView : public CWnd
{
// 생성입니다.
public:
   CChildView();
// 특성입니다.
public:
   CArray<TCHAR, TCHAR> m_str;
```

[실습 5-8] 문자 메시지 처리하기

```
void CChildView::OnPaint()
{
   CPaintDC dc(this);

   // 화면 출력용 폰트를 선택한다.
   CFont font;
   font.CreatePointFont(150, _T("궁서"));
   dc.SelectObject(&font);

   // 현재까지 입력된 글자들을 화면에 출력한다.
   CRect rect;
   GetClientRect(&rect);
   dc.DrawText(m_str.GetData(), m_str.GetSize(), &rect, DT_LEFT);
}
```

[실습 5-8] 문자 메시지 처리하기

```
void CChildView::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
   // [Backspace] 키 입력 시 맨 마지막 글자를 삭제한다.
   if (nChar == _T('\b')) {
      if(m_str.GetSize() > 0)
         m_str.RemoveAt(m_str.GetSize()-1);
   }
   // 그 밖의 경우에는 동적 배열에 글자를 추가한다.
   else{
      m_str.Add(nChar);
   }

   // 화면을 갱신한다.
   Invalidate();
}
```