Mouse, Keyboard Message

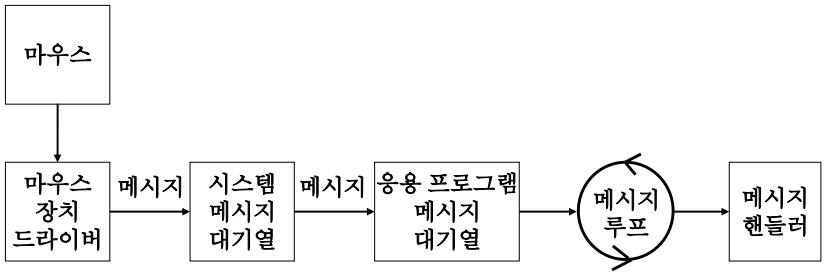
HCI Programming 2 (321190)
2007년 가을학기
10/15/2007
박경신

Overview

- Mouse Message
- Keyboard Message

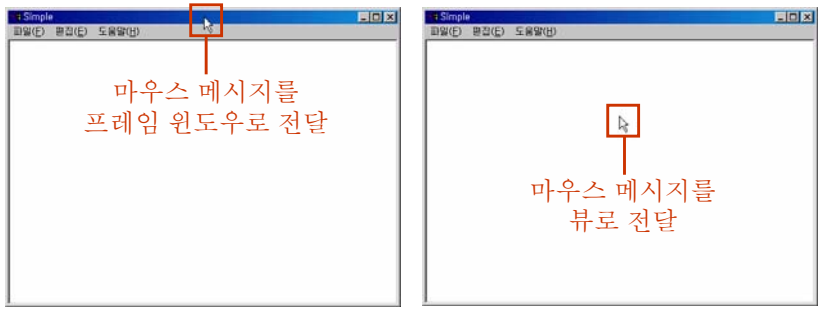
마우스 기초

- 마우스 처리
  - 윈도우 운영체제는 마우스와 관련된 모든 변화를 메시지 형태로 프로그램에게 전달한다.



마우스 기초

- 마우스 메시지 전달
  - 마우스 메시지는 마우스 커서 밑에 있는 윈도우가 받는다.



클라이언트 영역 마우스 메시지

- 클라이언트 영역 마우스 메시지

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

클라이언트 영역 마우스 메시지

- 클라이언트 영역 마우스 메시지 핸들러

메시지	메시지맵 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_MBUTTONDBLCL	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) ;
```

  - nFlags
    - 메시지가 생성될 때의 키보드나 마우스 버튼의 상태를 나타내는 비트 매스크
  - point
    - 메시지가 생성될 때의 마우스 커서 위치 (클라이언트 좌표)

비트 매스크	의미
MK_CONTROL	Ctrl 키가 눌렸을 때
MK_SHIFT	Shift 키가 눌렸을 때
MK_LBUTTON	마우스 왼쪽 버튼이 눌렸을 때
MK_MBUTTON	마우스 가운데 버튼이 눌렸을 때
MK_RBUTTON	마우스 오른쪽 버튼이 눌렸을 때

클라이언트 영역 마우스 메시지

- 예 - 마우스 버튼 클릭 시 키보드 사용 여부를 확인하기 위해 nFlags 확인

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
   if(nFlags & MK_SHIFT) {
      m_strOutText = "Shift키를 누름";  // 만약 Shift 키가 눌렸다면 ...
   }

   CWnd ::OnLButtonDown(nFlags, point);
}
```

클라이언트 영역 마우스 메시지

- 예 - 마우스 이동 시 드래그 처리를 위해 nFlags 확인
  - nFlags & MK_LBUTTON 의 값이 0 이 아니면 마우스 메시지와 함께 왼쪽 마우스 버튼이 눌려졌다는 의미

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
   if ((nFlags & MK_LBUTTON) | (nFlags & MK_RBUTTON))
     m_strOutText = "마우스를 드래그하고 있습니다";
   else
     m_strOutText = "마우스를 이동하였습니다";

   CWnd ::OnLButtonDown(nFlags, point);
}
```


클라이언트 영역 마우스 메시지

- 예 – MM_LOMETRIC 매핑 모드에서 마우스 왼쪽 버튼을 눌렀을 때 한 변의 길이가 2cm인 정사각형을 그림

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
   CClientDC dc(this);
   dc.SetMapMode(MM_LOMETRIC);    // 매핑 모드를 바꾼다
   CPoint pt = point;             // point 객체를 복사한다
   dc.DPtoLP(&pt);                // 장치좌표를 논리좌표로 변환
   dc.Rectangle(pt.x-100, pt.y+100, pt.x+100, pt.y-100);

   CWnd ::OnLButtonDown(nFlags, point);
}
```


마우스 캡쳐

- 용도
  - 마우스 캡처를 하면 마우스 커서의 위치에 관계없이 마우스 메시지를 받을 수 있다.
- 관련 함수

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

비 클라이언트 영역 마우스 메시지

- 비 클라이언트 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

비 클라이언트 영역 마우스 메시지

- 비 클라이언트 영역 마우스 메시지 핸들러

메시지	메시지맵 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()	OnNcMButtonDblCl
WM_NCRBUTTONDOWN	ON_WM_NCRBUTTONDOWN()	OnNcRButtonDown
WM_NCRBUTTONUP	ON_WM_NCRBUTTONUP()	OnNcRButtonUp
WM_NCRBUTTONDBLCLK	ON_WM_NCRBUTTONDBLCLK()	OnNcRButtonDblClk
WM_NCMOUSEMOVE	ON_WM_NCMOUSEMOVE()	OnNcMouseMove

비 클라이언트 영역 마우스 메시지

- 메시지 핸들러 형태

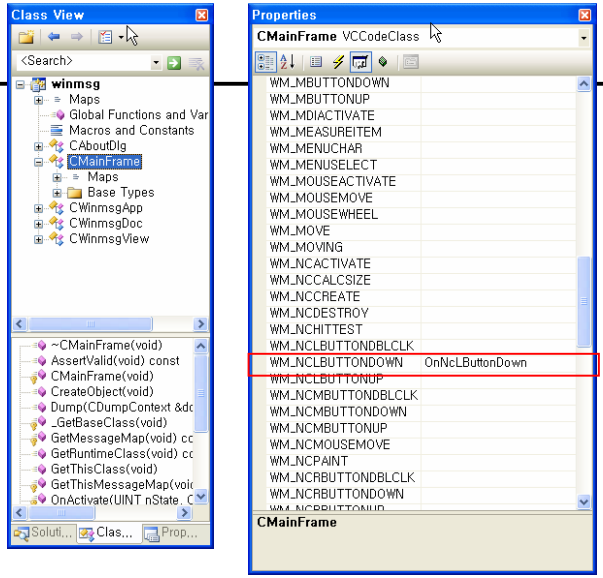
```
afx_msg void OnNc* (UINT nHitTest, CPoint point) ;
```

  - nHitTest
    - 메시지가 생성될 때의 마우스 커서 위치를 나타내는 상수값 ⇨ 다음 페이지 표 참조
  - point
    - 메시지가 생성될 때의 마우스 커서 위치(스크린 좌표)
      - 클라이언트 좌표로 변환하려면 CWnd::ScreenToClient() 함수를 사용

비 클라이언트 영역 마우스 메시지

- nHitTest – 비클라이언트 영역 마우스 위치

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



MainFrm.cpp

```
void CMainFrame::OnNcLButtonDown(UINT nHitTest, CPoint point)
{
	// TODO: Add your message handler code here and/or call default
	if (nHitTest == HTCAPTION)
		AfxMessageBox("제목표시줄을클릭하였습니다.");
	if (nHitTest == HTMINBUTTON)
		AfxMessageBox("최소화할수없습니다.");
	else
		CFrameWnd::OnNcLButtonDown(nHitTest, point);
}
```


마우스 정보와 커서 관리

- 마우스 정보

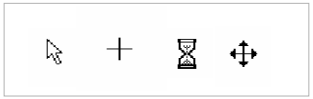
```
int GetSystemMetrics (int nIndex) ;
```

  - 마우스 관련 nIndex 값

nIndex	의미
SM_CMOUSEBUTTONS	마우스 버튼의 개수를 리턴하며 마우스가 설치되지 않은 경우에는 0을 리턴한다.
SM_MOUSEPRESENT	마우스의 설치 여부를 TRUE 또는 FALSE로 리턴한다.
SM_SWAPBUTTON	왼쪽과 오른쪽 버튼의 의미가 바뀌었으면 TRUE를 리턴한다.
SM_MOUSEWHEELPRESENT	휠(Wheel) 마우스이면 TRUE를 리턴한다.

마우스 정보와 커서 관리

- 커서 (Cursor)
  - 마우스의 위치를 알려주는 이미지
  - 커서모양에 따라 화면의 한 지점을 가리키는 위치(Hot Spot)를 설정
- 커서 선택
  - 표준커서 선택
    - CWinApp::LoadStandardCursor()
    - IDC_ARROW, IDC_CROSS, IDC_WAIT, IDC_SIZEALL,…
  - 리소스의 사용자 정의 커서 선택
    - CWinApp::LoadCursor()
- 커서형태변경 메시지
  - WM_SETCURSOR
  - 윈도우 내에서 커서가 움직일 때마다 윈도우로 보내지는 메시지



마우스 정보와 커서 관리

- 커서변경

  HCURSOR SetCursor (HCURSOR hCursor)
  - hCursor 커서 리소스를 가리키는 핸들값
    - 다음 두 함수의 리턴값을 대입
      - CWinApp::LoadStandardCursor()
      - CWinApp::LoadCursor()
- 커서위치확인

  BOOL GetCursorPos(LPPOINT *lpPoint*)
- 커서위치변경

  BOOL SetCursorPos(int x, int y)
- 커서 위치 제한

  BOOL ClipCursor (CONST RECT *lpRect)
  - lpRect 영역 안에 커서의 움직임이 제한, null이면 제한 해제
    - 커서가 움직일 수 있는 사각형의 범위(스크린 좌표)
    - 커서 움직임의 제한을 없애고자 한다면 lpRect에 NULL값을 사용

마우스 정보와 커서 관리

커서조작함수에서의 위치 값은 모두 스크린 좌표 값의 픽셀단위 사용
스크린 좌표 값 / 클라이언트 영역 좌표 값 변경 함수

CWnd::ScreenToClient
void ScreenToClient(LPPOINT *lpPoint*) const;
void ScreenToClient(LPRECT *lpRect*) const;
CWnd::ClientToScreen
void ClientToScreen(LPPOINT *lpPoint*) const;
void ClientToScreen(LPRECT *lpRect*) const;

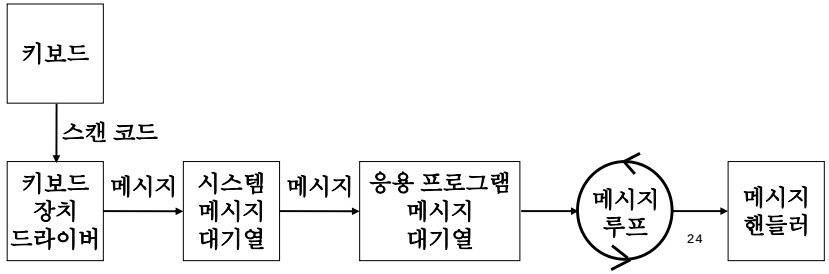

```
//커서 이동 시 발생되는 메시지 처리함수
BOOL CWinmsgMouseView::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message)
{
  // 클라이언트 영역이면 커서 모양을 변경한다.
  if(nHitTest == HTCLIENT){
    CPoint point;
    ::GetCursorPos(&point);    // 커서의 위치를 얻는다(스크린 좌표).
    ScreenToClient(&point);    // 스크린 좌표를 클라이언트 좌표로 변환한다.
   CRect rect1(100, 100, 200, 200); // 기준 사각형1
   CRect rect2(300, 100, 400, 200); // 기준 사각형2
   if(rect1.PtInRect(point))          // 커서가 기준 사각형1 안쪽에 있는지 확인한다.
       ::SetCursor(AfxGetApp()->LoadCursor(IDC_CURSOR1)); // 사용자정의 커서사용
   else if(rect2.PtInRect(point)) // 커서가 기준 사각형2 안쪽에 있는지 확인한다.
       ::SetCursor(AfxGetApp()->LoadCursor(IDC_CURSOR3)); // 사용자정의 커서사용
   else
       ::SetCursor(AfxGetApp()->LoadStandardCursor(IDC_CROSS)); //표준 커서 사용
   return TRUE;
  }
 return CView::OnSetCursor(pWnd, nHitTest, message);
}
```



```
//커서의 위치 제한 / 해제 처리
void CWinmsgMouseView::OnLButtonDblClk(UINT nFlags, CPoint point)
{
        CRect rect;
        GetClientRect(&rect);        //클라이언트 영역 추출
        ClientToScreen( &rect);      //클라이언트 영역을 스크린 영역좌표로 변환

        AfxMessageBox("커서영역을 클라이언트 영역으로 제한합니다");
        ::ClipCursor(&rect);         //클라이언트 영역으로 커서 제한
        CView::OnLButtonDblClk(nFlags, point);
}

void CWinmsgMouseView::OnRButtonDown(UINT nFlags, CPoint point)
{
        AfxMessageBox("커서영역을 제한을 해제합니다");
        ::ClipCursor(NULL);          //커서 영역제한 해제

        CView::OnRButtonDown(nFlags, point);
}
```

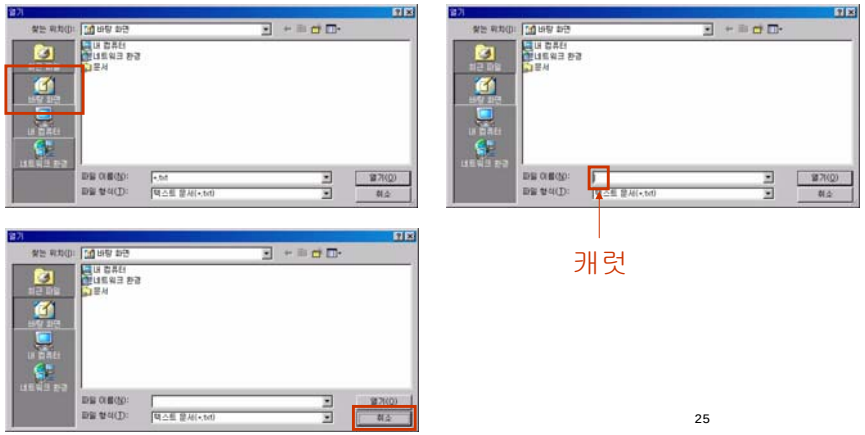

키보드 기초

- 키보드 처리
  - 윈도우 운영체제는 키보드와 관련된 모든 변화를 메시지 형태로 프로그램에게 전달한다.
- 키보드 메시지 전달
  - 키보드 메시지는 키보드 포커스를 가진 윈도우가 받는다.
- 키보드 포커스
  - 활성 윈도우 또는 활성 윈도우의 자식 윈도우가 가지는 일종의 속성



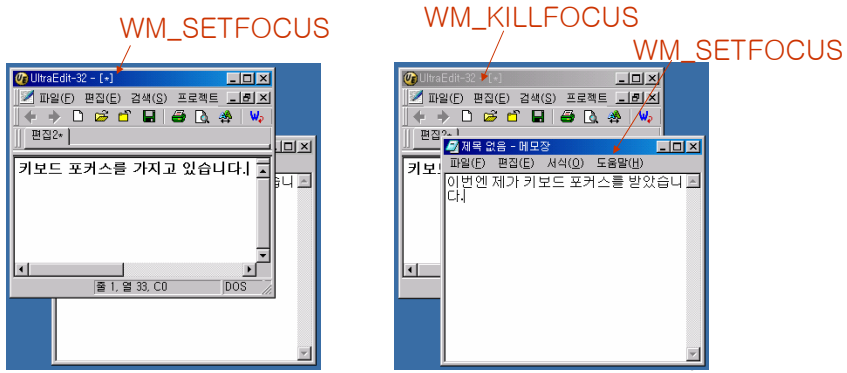
키보드 포커스

- 키보드 포커스 유형



키보드 포커스

- 키보드 포커스 변화
  - WM_SETFOCUS 포커스를 얻은 활성화된 윈도우에게 보내는 메시지
  - WM_KILLFOCUS 비활성화되는 윈도우에게 보내는 메시지



캐럿 (Caret) 관련 함수

- 캐럿 (Caret) 관련 함수

함수 이름	역할
CreateCaret()	비트맵을 이용하여 캐럿을 생성한다.
CreateGrayCaret()	회색 사각형 모양의 캐럿을 생성한다.
CreateSolidCaret()	검정색 사각형 모양의 캐럿을 생성한다.
ShowCaret()	캐럿이 보이도록 한다.
HideCaret()	캐럿을 숨긴다.
GetCaretPos()	캐럿의 위치(클라이언트 좌표)를 얻는다.
SetCaretPos()	캐럿의 위치(클라이언트 좌표)를 변경한다.
::DestroyCaret()	캐럿을 파괴한다.
::GetCaretBlinkTime()	캐럿이 깜박이는 간격을 얻는다.
::SetCaretBlinkTime()	캐럿이 깜박이는 간격을 설정한다.

캐럿

```
// 캐럿(caret) 사용 예
void CChildView::OnSetFocus(CWnd* pOldWnd)
{
    CWnd ::OnSetFocus(pOldWnd);

    CreateSolidCaret(20, 20);          //사각형 모양의 캐럿 생성
    SetCaretPos(CPoint(50, 50));       //캐럿의 위치 설정
    ShowCaret();                       //화면에 캐럿을 보이기
}

void CChildView::OnKillFocus(CWnd* pNewWnd)
{
    CWnd ::OnKillFocus(pNewWnd);

    HideCaret();                       //캐럿 숨기기
    ::DestroyCaret();                  //캐럿 삭제
}
```

키 누름 메시지

- 키 누름 메시지(Keystroke Message)
  - 키보드를 누르거나 떼는 동작에 의해 발생하는 메시지
  - WM_KEYDOWN, WM_KEYUP
- 키 누름 메시지 종류

메시지	의미
WM_KEYDOWN	F10, Alt 이외의 키를 누를 때
WM_KEYUP	F10, Alt를 이외의 키를 뗄 때
WM_SYSKEYDOWN	F10, Alt, Alt + 키 조합을 누를 때
WM_SYSKEYUP	F10, Alt, Alt + 키 조합을 뗄 때

키 누름 메시지

- 키보드 메시지 핸들러 형태

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

  - nChar
    - 키누름 메시지 (WM_KEYDOWN) - 키에 할당된 가상 키 코드값
    - 문자 메시지 (WM_CHAR) – 문자 코드
  - nRepCnt
    - 키를 계속 누르고 있을 경우 1보다 큰 값을 가진다.
  - nFlags
    - 키와 관련된 다양한 정보를 담고 있다(MSDN 참조).

키 누름 메시지

- 가상 키코드
  - 운영체제가 각 키에 할당한 장치독립적인 고유 값 <winuser.h>

가상 키 코드	해당 키	가상 키 코드	해당 키
VK_CANCEL	Ctrl-Break	VK_HOME	Home
VK_BACK	Backspace	VK_LEFT	←
VK_TAB	Tab	VK_UP	↑
VK_RETURN	Enter	VK_RIGHT	→
VK_SHIFT	Shift	VK_DOWN	↓
VK_CONTROL	Ctrl	VK_SNAPSHOT	Print Screen
VK_MENU	Alt	VK_INSERT	Insert
VK_PAUSE	Pause	VK_DELETE	Delete
VK_CAPITAL	Caps Lock	VK_0 - VK_9	0 – 9
VK_ESCAPE	Esc	VK_A - VK_Z	A – Z
VK_SPACE	Spacebar	VK_F1 - VK_F12	F1 - F12
VK_PRIOR	PgUp	VK_NUMLOCK	Num Lock
VK_NEXT	PgDn	VK_SCROLL	Scroll Lock
VK_END	End	VK_SCROLL	Scroll Lock

문자 메시지

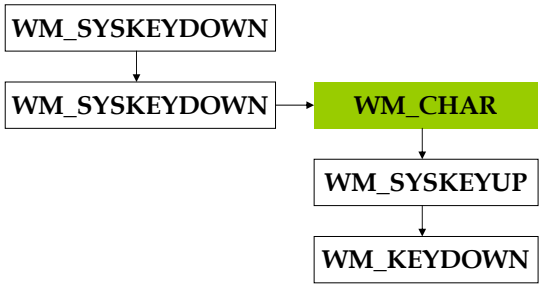
- 문자 메시지 (Character Message) 필요성 - VK_R 키를 누른 경우 ?

문자	가상 키 코드 조합
r	영문 입력 모드에서 VK_R 또는 VK_R + Caps Lock + Shift 키를 누른 경우
R	영문 입력 모드에서 VK_R + Caps Lock 또는 VK_R + Shift 키를 누른 경우
ㄱ	한글 입력 모드에서 VK_R 키를 누른 경우
ㄲ	한글 입력 모드에서 VK_R + Shift 키를 누른 경우

문자 메시지

- 문자 메시지 - VK_R 키를 누른 경우

- 문자 메시지 - Alt + VK_R 키를 누른 경우



문자 메시지

- 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 코드(Character Code) 값을 가진다.
  - nRepCnt
    - 키를 계속 누르고 있을 경우 1보다 큰 값을 가진다.
  - nFlags
    - 키와 관련된 다양한 정보를 담고 있다(MSDN 참조).

문자 메시지

- 키보드 메시지 예
  - 문자열을 입력 받고 이동키(화살표, pageUp/Down, Home, End)를 사용하여 문자열 이동
  - Document 클래스 멤버
    - 입력문자열(Cstring m_strOutText)
  - View 클래스 멤버
    - 출력위치 (Cpoint m_pNow)

```
void CWinmsgKeyView::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags) {
   CWinmsgKeyDoc* pDoc = GetDocument();// 도큐먼트의 포인터 얻음
   pDoc->m_strOutText += nChar;          // 키보드로 입력된 문자를 문자열에 추가
   Invalidate(false);                    // 화면 갱신, 변경된 부분만 다시 그림
   CView::OnChar(nChar, nRepCnt, nFlags);
}
```



```
void CWinmsgKeyView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {
   switch(nChar) {                       // 가상키 코드값
      case VK_LEFT:                      // 왼쪽 화살표키를 누를 때
         m_ptNow.x--;break;              // 왼쪽으로 1 이동
      case VK_RIGHT:                     // 오른쪽 화살표키를 누를 때
         m_ptNow.x++;break;              // 오른쪽으로 1 이동
      case VK_UP:                        // 위쪽 화살표키를 누를 때
         m_ptNow.y--;break;              // 위쪽으로 1 이동
      case VK_DOWN:                      // 아래쪽 화살표키를 누를 때
         m_ptNow.y++;break;              // 아래쪽으로 1 이동
      case VK_PRIOR:                     // Pageup키를 누를 때
         m_ptNow.y -= 50;break;          // 위쪽으로 50 이동
      case VK_NEXT:                      // Pagedown키를 누를 때
         m_ptNow.y += 50;break;          // 아래쪽으로 50 이동
      case VK_HOME:                      // Home키를 누를 때
         m_ptNow = CPoint(0, 0);break;   // 처음위치로 이동
   }
   if(m_ptNow.x < 0) {                   // X좌표가 0보다 작으면
      m_ptNow.x = 0;                     // m_ptNow.x = 0으로 초기화
      AfxMessageBox("왼쪽으로 더 이상 이동할 수 없습니다."); // 메시지 박스 출력
   }
   if(m_ptNow.y < 0) {                   // Y좌표가 0보다 작으면
      m_ptNow.y = 0;                     // m_ptNow.y = 0으로 초기화
      AfxMessageBox("위쪽으로 더 이상 이동할 수 없습니다."); // 메시지 박스 출력
   }
   Invalidate();                         // 화면 갱신
   CView::OnKeyDown(nChar, nRepCnt, nFlags);
}
```