Rapid Application Development
Keystrokes and Mouse Handling

Centering Text in a Window

CWnd GetWindowRect () : - To get the size of client area.

CRect class is used to hold the dimension and size of rectangle and hence the rectangle will be our client area.

View.cpp

```
Void CKeystrokesView :: OnDraw(CDC* pDC)

{

CKeystrokesDoc * pDoc = GetDocument();

ASSERT_VALID(pDoc);

CRect rect;                    //CRect class is used to hold the dimension and size of rectangle

GetWindowRect(&rect);     //GetWindowRect()- to get the size of client area;

Int x= rect.Width()/2;       //The CRect class's Width() and Height() methods are used to gets the

Int y = rect.Height()/2;          client's area width and height.
```

To start centering the text ,find the exact center of the client area and store the location in two variables x and y

Finding size of displayed text string

We use the CDC class GetTextExtent() by just passing the text string to it and that method returns an object of MFC CSize class.

CSize class has two important methods called as cx and cy.

```
CSize size = pDC-> GetTextExtent(pDoc-> StringData);

x -= size.cx/2;

y -= size.cy/2;

pDC-> TextOut(x,y,pDoc-> StringData);

}
```

```
Void CKeystrokesView :: OnDraw(CDC* pDC) // whole onDraw() method
{
CKeystrokesDoc * pDoc  =  GetDocument();
ASSERT_VALID(pDoc);
CRect rect;
GetWindowRect(&rect);
Int x= rect.Width()/2;
Int y = rect.Height()/2;
CSize  size = pDC-> GetTextExtent(pDoc-> StringData);
x -= size.cx/2;
y -= size.cy/2;
pDC-> TextOut(x,y,pDoc-> StringData);
}
```

Adding a caret to a window

The mouse can generate quite a number of events from **WM_LBUTTONDOWN** – i.e. when the user press the left button on the mouse. **WM_MOUSEMOVE** – when the user moves the mouse .When the user clicks a new location it is handled in windows with a caret(called insertion point)

Use AppWizard to create a SDI program named Carets.

Write all the same code written in keystroke progam.

We create a new caret and decide the size of the caret. A caret is usually made the same height as the current character and 1/8 of the width of average character.

To determine the **height** and **width** of characters we use CDC method **GetTextMetrics();**

Measuring TextSizes with Textmetrices

We start by a Boolean variable named CaretCreated in the view object to keep track whether we have created the caret or not.

CaretsView.h

```
class CCaretsView: public CView

{

Protected:

CPoint CaretPosition;

boolean  CaretCreated;

}
```

```
CCaretsView :: CCaretsView()

{CaretCreated = false;//check if this code is written or not}
```

After setting caretcreated to false , we check to see if we've already created the caret or not

Now we decide the size of caret and we get the size from **TEXTMETRIC** structure by calling **GetTextMetrics()**

```
void CCaretsView :: OnDraw( CDC * pDC)

{

    If(!CaretC reated)

   {

        TEXTMETRIC textmetric;

        pDC -> GetTextMetric(& textmetric);

        CreateSolidCaret(Textmetric.tmAveCharWidth/8,textmetric.tmHeight);
```

The Caret method also include ShowCaret(),SetCaretPos(), and HideCaret().We make the caret the same height as our text using textmetric .tmHeight and 1/8 th of the width of average character.we call **CreateSolidCaret()** to actually create the caret.

Setting the Caret's Position

We store the caret's position in a new **CPoint** object named **CaretPosition**.CPoint object named CaretPosition .CPoint class has two data data members x and y which will hold the position of the caret.

CaretPosition.x = CaretPosition.y =0;

Now we select the Caret's position with **SetCaretPos()**- shows the caret's position.**ShowCaret()**- It shows the caret on the sscreen and set the **CaretCreated** boolean flag to true.

SetCaretPos(CaretPosition);

ShowCaret();

CaretCreated = true;

}//end of caret if function

The caret appears on the screen as the blinking function

The next step is to move the caret as the user type text.

pDC-> TextOut(0,0,pDoc-> StringData);

Now we have to determine the end of string where we can place the caret .we do this by **CSize** object named "size" using **GetTextExtent();**

CSize size = pDC-> GetTextExtent(pDoc-> StringData);

To display caret at the end of the text string we first hide it using **HideCaret()**. Next we set x data member of caret position point at the end of text string.

```
HideCaret();

CaretPosition.x=size.cx;

SetCaretPos(CaretPosition);

ShowCaret();

}//ending of OnDraw method()
```

Showing and ending of caret when we lose or gain the focus

When our program looses the focus we get the **WM_KILLFOCUS** msg and the corresponding event handler is **OnKillFocus();**

```
Void CCaretsView:: OnKillFocus(CWnd* pNewWnd)

{

CView :: OnKillFocus(pNewWnd);

HideCaret();

}

Void CCaretView:: OnSetFocus( Cwnd * pOldWnd)

{

CView:: OnSetFocus(pOldWnd);

Sho wCaret();

}
```

Complete OnDraw () method function

```
void CCaretsView :: OnDraw( CDC * pDC)
{ If(!CaretC reated)
{
        TEXTMETRIC textmetric;
        pDC -> GetTextMetric(& textmetric);
        CSize size =  pDC-> GetTextExtent(pDoc-> StringData);
        CreateSolidCaret(Textmetric.tmAveCharWidth/8,textmetric.tmHeight);
        CaretPosition.x = CaretPosition.y =0;
        SetCaretPos(CaretPosition);
         ShowCaret();
        CaretCreated = true;
}//end of caret if function
pDC-> TextOut(0,0,pDoc->  StringData);
CSize size =  pDC-> GetTextExtent(pDoc-> StringData);
HideCaret();
CaretPosition.x=size.cx;
```

```
SetCaretPos(CaretPosition);

ShowCaret();

}//ending of OnDraw method()
```