
Subject: Re: Japanese IME test code

Posted by [mirek](#) on Thu, 08 May 2008 07:52:28 GMT

[View Forum Message](#) <> [Reply to Message](#)

mobilehunter wrote on Thu, 08 May 2008 02:04I change my codes to support "on-the-spot" mode of IME for Windows, i also tried playing with XIM for linux, tested against SCIM (this will come later sometime:)).

I still have problem with caret, and many behaviours not supported.

For this mode i tested with EditField control (i have to modify it's Paint function).

Here are my codes:

At CtrlCore.h

//Added for CJK

public:

virtual bool ImeIsInPreEditMode() { return imeIsInPreEditMode;}

virtual Font ImePreEditFont();

virtual Point ImePreEditStartPoint();

virtual void ImePreEditSetText(const WString & text);

virtual int ImePreEditGetTextCx(const wchar *txt, int n, bool password, Font fnt);

virtual void ImePreEditStart();

virtual void ImePreEditEnd();

virtual void ImePreEditSyncCaret();

virtual int ImePreEditPaint(Draw &w,int fcy, Color ink, Color paper, Font fnt);

private:

Font imeFont;

bool imeIsInPreEditMode;

Point imePreEditPoint;

int imePreEditCursor;

WString imePreEditString;

//End of CJK

At Ctrl.cpp

//Added for CJK

Font Ctrl::ImePreEditFont()

{

return StdFont();

}

Point Ctrl::ImePreEditStartPoint()

{

return Point(caretx,carety);

}

```

void Ctrl::ImePreEditSetText(const WString& text)
{
    imePreEditString = text;
    Refresh();
}

void Ctrl::ImePreEditSyncCaret()
{
    int x = ImePreEditGetTextCx(imePreEditString,imePreEditCursor,false,imeFont);
    FontInfo fi = imeFont.Info();
    int ty = (GetSize().cy + 1 - fi.GetHeight()) / 2;

    caretX=imePreEditPoint.x + x + 2 - fi.GetRightSpace('o') + fi.GetLeftSpace('o');
    caretY=ty;
    caretX=1;
    caretY=min(GetSize().cy - 2 * ty, fi.GetHeight());
}

int Ctrl::ImePreEditGetTextCx(const wchar *txt, int n, bool password, Font fnt)
{
    FontInfo fi = fnt.Info();
    if(password)
        return n * fi['*'];
    const wchar *s = txt;
    int x = 0;
    while(n--)
        x += fi[*s++];
    return x;
}

void Ctrl::ImePreEditStart()
{
    imePreEditPoint = ImePreEditStartPoint();
    imeIsInPreEditMode = true;
    imePreEditCursor=0;
    imePreEditString="";
    imeFont = ImePreEditFont();
}

void Ctrl::ImePreEditEnd()
{
    imeIsInPreEditMode = true;
    imePreEditCursor=0;
    imePreEditString="";
    Refresh();
}

int Ctrl::ImePreEditPaint(Draw &w,int fcy, Color ink, Color paper, Font fnt)

```

```

{
int n=imePreEditString.GetLength();
if(n < 0) return 0;
const wchar * imtxt=imePreEditString;
int cx = ImePreEditGetTextCx(imePreEditString,n,false,imeFont);
w.DrawRect(imePreEditPoint.x, 0, cx, fcy, paper);
w.DrawText(imePreEditPoint.x, 0, imtxt, fnt, ink, n);
//draw feedback
w.DrawLine(imePreEditPoint.x,fcy,imePreEditPoint.x+cx,fcy,1);
return cx;
}
//End of CJK

```

At Win32Proc.cpp

```

case WM_IME_STARTCOMPOSITION:
{
    Ctrl*f=GetFocusChild();
    f->ImePreEditStart();

    CANDIDATEFORM cf;
    cf.dwStyle = CFS_CANDIDATEPOS;
    cf.ptCurrentPos.x = f->imePreEditPoint.x;
    cf.ptCurrentPos.y = f->imePreEditPoint.y;

    HIMC hIMC = ImmGetContext(hwnd);
    ImmSetCandidateWindow(hIMC, &cf);
    ImmReleaseContext(hwnd, hIMC);

    return 0L;
}
case WM_IME_ENDCOMPOSITION:
{
    Ctrl*f=GetFocusChild();
    f->ImePreEditEnd();
    return 0L;
}
case WM_IME_COMPOSITION:
{
    Ctrl*f=GetFocusChild();

    if(IParam & GCS_COMPSTR)
    {
        HWND hWnd =GetHWND();
        HIMC hIMC = ImmGetContext(hwnd);

        long dwSize = ImmGetCompositionStringW(hIMC, GCS_COMPSTR, NULL, 0);
    }
}

```

```

wchar temp[1024];
wchar attribute[10];

if(IParam & GCS_COMPATTR)
{
    int attrLen=ImmGetCompositionStringW(hIMC, GCS_COMPATTR, NULL, 0);
    ImmGetCompositionStringW(hIMC, GCS_COMPATTR, attribute, attrLen);
    LOGHEXDUMP(attribute,attrLen);
}

    ImmGetCompositionStringW(hIMC, GCS_COMPSTR, temp, dwSize);
    if(IParam & GCS_CURSORPOS)
    {
        int cursor = ImmGetCompositionString(hIMC, GCS_CURSORPOS, NULL, 0);
        DUMP(cursor);
        if(attribute[0] != ATTR_TARGET_CONVERTED)
            f->imePreEditCursor = cursor/2;
    }
    ImmReleaseContext(hWnd, hIMC);
    f->ImePreEditSetText(WString(temp,dwSize/2));
    if(attribute[0] != ATTR_TARGET_CONVERTED)
    {
        f->ImePreEditSyncCaret();
        SyncCaret();
    }
    return 0L;
}
break;
}

```

And at EditField.cpp

```

if(GetSelection(l, h)) {
    Paints(w, x, fcy, txt, ink, paper, l, password, font);
    Paints(w, x, fcy, txt, enabled ? st->selectedtext : paper,
        enabled ? st->selected : ink, h - l, password, font);
    Paints(w, x, fcy, txt, ink, paper, text.GetLength() - h, password, font);
}
else
{
    if(!ImeIsInPreEditMode())
        Paints(w, x, fcy, txt, ink, paper, text.GetLength(), password, font);
    else
    {
        Paints(w, x, fcy, txt, ink, paper, cursor, password, font);
        x+=ImePreEditPaint(w,fcy,ink,paper,font);
    }
}

```

```
    Paints(w, x, fcy, txt, ink, paper, text.GetLength() - cursor, password, font);  
}  
}
```

Uh, what is "on-the-spot" mode?

The number of new virtual functions is staggering....

BTW, for now I would like to add the mode from previous posts, where we were ok with just

```
virtual Font  GetIMEFont();  
virtual Point GetIMEPoint();
```

The only thing I am missing there is change so that GetIMEPoint returns view-relative Point, not top window relative.

Mirek
