Subject: Japanese IME test code
Posted by mobilehunter on Mon, 31 Mar 2008 07:58:17 GMT
View Forum Message <> Reply to Message

Hope Mirek and other can check these code

```
case WM_IME_STARTCOMPOSITION:
 {
  HWND hwnd =GetHWND();
  POINT pt;
  GetCaretPos(&pt);// I don't know the properway to get this in UPP way.
  HIMC hIMC = ImmGetContext(hwnd);
  COMPOSITIONFORM cf;

/*These lines up to us to make ime font match to current font, or just the font size.
  FontInfo fi = Draw::GetStdFont().Info();
  LOGFONT lf;
  HFONT hfont = fi.GetHFONT(); //i'm sorry to add this function manually to Draw.h, since i don't
know how to get HFONT
  GetObject(hfont, sizeof(LOGFONT), &lf);
  ImmSetCompositionFont(hIMC, &lf);

  cf.dwStyle = CFS_POINT;
  cf.ptCurrentPos.x = pt.x;
  cf.ptCurrentPos.y = pt.y;
  ImmSetCompositionWindow(hIMC, &cf);
  ImmReleaseContext(hwnd, hIMC);
 }
break;
```

Above code will make IME window position start at current caret position.

Please fix above codes, since i don't have deep understanding of UPP codes.

Test with GUI17a sample and UWord

Subject: Re: Japanese IME test code
Posted by mirek on Sun, 06 Apr 2008 06:14:58 GMT
View Forum Message <> Reply to Message

mobilehunter wrote on Mon, 31 March 2008 03:58Hope Mirek and other can check these code

```
case WM_IME_STARTCOMPOSITION:
 {
  HWND hwnd =GetHWND();
  POINT pt;
```

```
  GetCaretPos(&pt);// I don't know the properway to get this in UPP way.
  HIMC hIMC = ImmGetContext(hwnd);
  COMPOSITIONFORM cf;

/*These lines up to us to make ime font match to current font, or just the font size.
  FontInfo fi = Draw::GetStdFont().Info();
  LOGFONT lf;
  HFONT hfont = fi.GetHFONT(); //i'm sorry to add this function manually to Draw.h, since i don't
know how to get HFONT
  GetObject(hfont, sizeof(LOGFONT), &lf);
  ImmSetCompositionFont(hIMC, &lf);

  cf.dwStyle = CFS_POINT;
  cf.ptCurrentPos.x = pt.x;
  cf.ptCurrentPos.y = pt.y;
  ImmSetCompositionWindow(hIMC, &cf);
  ImmReleaseContext(hwnd, hIMC);
 }
break;
```

Above code will make IME window position start at current caret position.

Please fix above codes, since i don't have deep understanding of UPP codes.

Test with GUI17a sample and UWord

Looks OK to me, but I cannot check it

If this gets approval from any other CJK user, I will paste it in gladly

Mirek

Subject: Re: Japanese IME test code
Posted by mobilehunter on Mon, 07 Apr 2008 01:10:54 GMT
View Forum Message <> Reply to Message

Hi Mirek,
Thanks for the reply.

I change the code for displaying the IME window to controls such as RichEdit and EditField, since as my understanding those controls know their font and caret position better. And add a virtual function DisplayIMEWindow() to those controls.

So the implementation inside Win32Proc.cpp:

case WM_IME_STARTCOMPOSITION:

```
{
  Ctrl*f=GetFocusChild();
  if(f)
     f->DisplayIMEWindow();
}
break;
```

And below are the implementation inside EditField control

```
void EditField::DisplayIMEWindow()
{
 HWND hwnd =this->GetParent()->GetHWND();
 POINT pt;
 FontInfo fi = font.Info();

 GetCaretPos(&pt);
 HIMC hIMC = ImmGetContext(hwnd);
 COMPOSITIONFORM cf;

 LOGFONT lf;
 HFONT hfont = fi.GetHFONT();
 GetObject(hfont, sizeof(LOGFONT), &lf);
 ImmSetCompositionFont(hIMC, &lf);

 cf.dwStyle = CFS_POINT;
 cf.ptCurrentPos.x = pt.x;
 cf.ptCurrentPos.y = pt.y;
 ImmSetCompositionWindow(hIMC, &cf);
 ImmReleaseContext(hwnd, hIMC);
}
```

And below inside RichEdit control (kbd.cpp):

```
void RichEdit::DisplayIMEWindow()
{
 HWND hwnd = this->GetParent()->GetHWND();
 POINT pt;
 FontInfo fi = formatinfo.Info();

 GetCaretPos(&pt);
 HIMC hIMC = ImmGetContext(hwnd);
 COMPOSITIONFORM cf;
 LOGFONT lf;

 HFONT hfont = fi.GetHFONT();
 ::GetObject(hfont, sizeof(LOGFONT), &lf);
```

```
 ImmSetCompositionFont(hIMC, &lf);

 cf.dwStyle = CFS_POINT;
 cf.ptCurrentPos.x = pt.x;
 cf.ptCurrentPos.y = pt.y;
 ImmSetCompositionWindow(hIMC, &cf);
 ImmReleaseContext(hwnd, hIMC);
}
```

I still have problem with the font size for IME window.
And have problem debugging the codes. The IDE will hang if i press F10 key after a break point.

Subject: Re: Japanese IME test code
Posted by mobilehunter on Thu, 10 Apr 2008 03:27:48 GMT
View Forum Message <> Reply to Message

Another update for RichEdit control, to make the font size of IME window follows RichEdit's font size.

```
void RichEdit::DisplayIMEWindow()
{
  HWND hwnd = this->GetParent()->GetHWND();
  POINT pt;
  COMPOSITIONFORM cf;
  LOGFONT lf;
  HIMC hIMC = ImmGetContext(hwnd);
  GetCaretPos(&pt);

  int zoomHeight = GetZoom() * tabs(formatinfo.GetHeight());
  ImmGetCompositionFont(hIMC,&lf);
  lf.lfHeight = -zoomHeight;
  ImmSetCompositionFont(hIMC, &lf);

  cf.dwStyle = CFS_POINT;
  cf.ptCurrentPos.x = pt.x;
  cf.ptCurrentPos.y = pt.y;

  ImmSetCompositionWindow(hIMC, &cf);
  ImmReleaseContext(hwnd, hIMC);
}
```

And for EditField.

```
void EditField::DisplayIMEWindow()
{
```

```
 HWND hwnd = this->GetParent()->GetHWND();
 POINT pt;
 COMPOSITIONFORM cf;
 LOGFONT lf;
 HIMC hIMC = ImmGetContext(hwnd);
 GetCaretPos(&pt);
 Size sz = GetSize();
 int yy = GetTy();

 ImmGetCompositionFont(hIMC,&lf);
 lf.lfHeight = font.Info().GetHeight()+yy;
 ImmSetCompositionFont(hIMC, &lf);

 cf.dwStyle = CFS_POINT;
 cf.ptCurrentPos.x = pt.x;
 cf.ptCurrentPos.y = pt.y-yy;

 ImmSetCompositionWindow(hIMC, &cf);
 ImmReleaseContext(hwnd, hIMC);
}
```

Subject: Re: Japanese IME test code
Posted by mirek on Tue, 15 Apr 2008 19:17:01 GMT
View Forum Message <> Reply to Message

I am afraid this new spin is not quite as good.

IMO, if you want more correct font, you should rather add some means to get it form the widget, something like "virtual Font GetIMEFont()" method.

Also, suggested implementation are in Win32. At least, they should be #ifdefed in Win32.

Mirek

Subject: Re: Japanese IME test code
Posted by mobilehunter on Mon, 21 Apr 2008 04:33:34 GMT
View Forum Message <> Reply to Message

Changed the codes.
All the functions declared as virtual
RichEdit control:

#ifdef PLATFORM_WIN32
Font RichEdit::GetIMEFont()

```
{
  Font imeFont(formatinfo);
  int zoomHeight = GetZoom() * tabs(formatinfo.GetHeight());

  imeFont.Height(zoomHeight);
  return imeFont;
}

void RichEdit::DisplayIMEWindow()
{
  HWND hwnd = this->GetParent()->GetHWND();
  POINT pt;
  COMPOSITIONFORM cf;
  LOGFONT lf;
  GetCaretPos(&pt);
  Font imeFont = GetIMEFont();

  cf.dwStyle = CFS_POINT;
  cf.ptCurrentPos.x = pt.x;
  cf.ptCurrentPos.y = pt.y;

  HIMC hIMC = ImmGetContext(hwnd);
  ImmGetCompositionFont(hIMC,&lf);
  lf.lfHeight = -imeFont.GetHeight();
  ImmSetCompositionFont(hIMC, &lf);
  ImmSetCompositionWindow(hIMC, &cf);
  ImmReleaseContext(hwnd, hIMC);
}
#endif PLATFORM_WIN32
```

And for editfield control:

```
#ifdef PLATFORM_WIN32
Font EditField::GetIMEFont()
{
  Font imeFont(font);
  imeFont.Height(font.Info().GetHeight());
  return imeFont;
}

void EditField::DisplayIMEWindow()
{
  HWND hwnd = this->GetParent()->GetHWND();
  POINT pt;
  COMPOSITIONFORM cf;
  LOGFONT lf;
  GetCaretPos(&pt);
```

```
  int yy = GetTy();
  Font imeFont = GetIMEFont();

  cf.dwStyle = CFS_POINT;
  cf.ptCurrentPos.x = pt.x;
  cf.ptCurrentPos.y = pt.y-yy;//to make the ime window to appear inside editfield

  HIMC hIMC = ImmGetContext(hwnd);
  ImmGetCompositionFont(hIMC,&lf);
  lf.lfHeight = -(imeFont.GetHeight());
  ImmSetCompositionFont(hIMC, &lf);
  ImmSetCompositionWindow(hIMC, &cf);
  ImmReleaseContext(hwnd, hIMC);
}
#endif
```

Subject: Re: Japanese IME test code
Posted by mirek on Wed, 23 Apr 2008 08:02:12 GMT
View Forum Message <> Reply to Message

Ah, sorry, looks like I was not specific enough

DisplayIMEWindow are now the same, are not they? So with GetIMEFont, we can gladly move them into CtrlCore and not be bothered reimplementing it for each individual class.

In the same time, we can implement base Ctrl::GetIMEFont to return StdFont and get this working well for all widgets.

What do you think?

Mirek

Subject: Re: Japanese IME test code
Posted by mobilehunter on Fri, 25 Apr 2008 05:15:58 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 23 April 2008 17:02Ah, sorry, looks like I was not specific enough

DisplayIMEWindow are now the same, are not they? So with GetIMEFont, we can gladly move them into CtrlCore and not be bothered reimplementing it for each individual class.

In the same time, we can implement base Ctrl::GetIMEFont to return StdFont and get this working well for all widgets.

What do you think?

Mirek
DisplayIMEWindow are almost the same, the differences are at font height and coordinate of IME window.

The font height of IME window should follow the font height of it's parent, also should fit nicely inside the parent.

That's why i implemented at RichEdit and EditField ctrl, since i don't know how to get current font height of each control, and to layout nicely in generic way.

Moving to CtrlCore would be the best.

Subject: Re: Japanese IME test code
Posted by mirek on Sat, 26 Apr 2008 08:26:41 GMT
View Forum Message <> Reply to Message

mobilehunter wrote on Fri, 25 April 2008 01:15
That's why i implemented at RichEdit and EditField ctrl, since i don't know how to get current font height of each control, and to layout nicely in generic way.

Well, but if you would introduce

virtual void GetIMEFont();

int Ctrl interface (returning StdFont as default implementation), you could move

void DisplayIMEWindow()

into Ctrl too, placing IME window at caret position, or maybe adding another

virtual Point GetIMEPosition()

returning caret position as default.

In any case, implementing GetIMEFont/GetIMEPosition seems to be more simple (and often OK with default implementation) that implementing DisplayIMEWindow.

Anything I am missing?

Mirek

Subject: Re: Japanese IME test code
Posted by mirek on Sat, 26 Apr 2008 08:27:39 GMT
View Forum Message <> Reply to Message

P.S.: And it solves "#ifdef PLATFORM_WIN32" issue as well...

Mirek

Subject: Re: Japanese IME test code
Posted by mobilehunter on Sat, 26 Apr 2008 12:13:04 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 26 April 2008 17:26

In any case, implementing GetIMEFont/GetIMEPosition seems to be more simple (and often OK with default implementation) that implementing DisplayIMEWindow.

Anything I am missing?

Mirek

No:). I think it's better.
If i'm not wrong, the implementation of invoking IME window will be reside at Win32Proc.cpp or X11Proc.cpp.

Subject: Re: Japanese IME test code
Posted by mirek on Sat, 26 Apr 2008 13:00:55 GMT
View Forum Message <> Reply to Message

Will you implement and test it? (I am a little bit short of IME here

Mirek

Subject: Re: Japanese IME test code
Posted by mobilehunter on Sat, 26 Apr 2008 13:35:12 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 26 April 2008 22:00Will you implement and test it? (I am a little bit short of IME here

Mirek
Sure. I will try at monday.
I don't have windows machine for development at home now.

Subject: Re: Japanese IME test code
Posted by mobilehunter on Sun, 27 Apr 2008 02:05:30 GMT
View Forum Message <> Reply to Message

Here are new implementation.
Tested against EditCtrl only.

at CtrlCore.h:

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

at Ctrl.cpp:

```
//default implementation
Font Ctrl::GetIMEFont()
{
 return StdFont();
}

Point Ctrl::GetIMEPoint()
{
 POINT pt;
 GetCaretPos(&pt);//still not upp-wise :)

 return Point(pt);
}
```

at win32proc.cpp:

```
case WM_IME_STARTCOMPOSITION:
{
 Ctrl*f=GetFocusChild();

 HWND hwnd =GetHWND();
 Point imePt = f->GetIMEPoint();
 Font imeFont = f->GetIMEFont();

 COMPOSITIONFORM cf;
 cf.dwStyle = CFS_POINT;
 cf.ptCurrentPos.x = imePt.x;
 cf.ptCurrentPos.y = imePt.y;

 LOGFONT lf;
 HIMC hIMC = ImmGetContext(hwnd);

 ImmGetCompositionFont(hIMC,&lf);
```

```
 lf.lfHeight = -imeFont.Info().GetHeight();
 ImmSetCompositionFont(hIMC, &lf);

 ImmSetCompositionWindow(hIMC, &cf);
 ImmReleaseContext(hwnd, hIMC);
}
break;
```

Subject: Re: Japanese IME test code
Posted by mirek on Tue, 29 Apr 2008 16:56:09 GMT
View Forum Message <> Reply to Message

Should not GetIMEPoint be rather in view coordinates?

(I suppose you have screen coordinates here).

Mirek

Subject: Re: Japanese IME test code
Posted by mobilehunter on Fri, 02 May 2008 05:39:43 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 30 April 2008 01:56Should not GetIMEPoint be rather in view coordinates?

(I suppose you have screen coordinates here).

Mirek

GetCaret() will return relative coordinate from containing window.

Subject: Re: Japanese IME test code
Posted by mirek on Tue, 06 May 2008 17:49:36 GMT
View Forum Message <> Reply to Message

mobilehunter wrote on Fri, 02 May 2008 01:39luzr wrote on Wed, 30 April 2008 01:56Should not GetIMEPoint be rather in view coordinates?

(I suppose you have screen coordinates here).

Mirek

GetCaret() will return relative coordinate from containing window.

Which is not the same as view of child widget...

(I could fix the code, but I cannot test it.)

Mirek

Subject: Re: Japanese IME test code
Posted by mirek on Tue, 06 May 2008 20:21:23 GMT
View Forum Message <> Reply to Message

I guess this might be helpful here:

```
Rect Ctrl::GetCaret() const
{
 return RectC(caretx, carety, caretcx, caretcy);
}
```

Mirek

Subject: Re: Japanese IME test code
Posted by mobilehunter on Thu, 08 May 2008 06:04:16 GMT
View Forum Message <> Reply to Message

I change my codes to support "on-the-spot" mode of IME for Windows, i also tried playing with XIM for linux, tested againt 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;
 caretcx=1;
 caretcy=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(lParam & GCS_COMPSTR)
  {
        HWND hWnd =GetHWND();
        HIMC hIMC = ImmGetContext(hwnd);

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

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

   if(lParam & 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(lParam & 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);
   }
  }
```

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 againt 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;
 caretcx=1;
 caretcy=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(lParam & GCS_COMPSTR)
  {
        HWND hWnd =GetHWND();
        HIMC hIMC = ImmGetContext(hwnd);

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

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

   if(lParam & 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(lParam & 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

Subject: Re: Japanese IME test code
Posted by mobilehunter on Thu, 08 May 2008 08:04:02 GMT
View Forum Message <> Reply to Message

According to these:
http://www.mozilla.org/projects/intl/input-method-spec.html

On-the-spot means:
The composed text is rendered inside the text window by the application, by maintaining a special editing area "between" the text before the insertion point and the text after the insertion point. The composed text looks like text part of the document, however, different stylistic attributes are applied to the text into indicate that it is part of the input method composition string. Different parts of the input method composing string will have different styles applied to them, which indicates that they are in different stages of editing. Once the composition text is finalized by the user, it merges into the original document and is indistinguishable from the surrounding text. The on-the-spot style is also know as inline input on some platforms.

Subject: Re: Japanese IME test code
Posted by mobilehunter on Fri, 09 May 2008 00:16:48 GMT
View Forum Message <> Reply to Message

Hi Mirek,
Sorry about virtual functions. Maybe we can remove some.
Please also consider the X11 implementations

Here my codes for X11:
at CtrlCore.cpp

```
#ifdef PLATFORM_X11
 static int ImePreEditStartCB(XIC xic, XPointer clientData, XPointer callData);
 static void ImePreEditDoneCB(XIC xic, XPointer clientData, XPointer callData);
 static void ImePreEditDrawCB(XIC xic, XPointer clientData, XIMPreeditDrawCallbackStruct
*callData);
 static void ImePreEditCaretCB(XIC xic, XPointer clientData, XIMPreeditCaretCallbackStruct
*callData);
 #endif
```

at Ctrl.cpp

```
#ifdef PLATFORM_X11
int Ctrl::ImePreEditStartCB(XIC xic, XPointer clientData, XPointer callData)
{
 Ctrl *ctrl = (Ctrl*)clientData;

 Ptr<Ctrl> pfocusCtrl = ctrl->GetFocusCtrl();
 if(!pfocusCtrl)
  return -1;

 pfocusCtrl->ImePreEditStart();
 return -1;
}

void Ctrl::ImePreEditDoneCB(XIC xic, XPointer clientData, XPointer callData)
{
 Ctrl *ctrl = (Ctrl*)clientData;

 Ptr<Ctrl> pfocusCtrl = ctrl->GetFocusCtrl();
 if(!pfocusCtrl)
  return;
 pfocusCtrl->ImePreEditEnd();
}

void Ctrl::ImePreEditDrawCB(XIC xic, XPointer clientData, XIMPreeditDrawCallbackStruct
*callData)
{
 Ctrl *ctrl = (Ctrl*)clientData;
 Ptr<Ctrl> pfocusCtrl = ctrl->GetFocusCtrl();

 if(callData)
 {
  int caretPos = callData->caret;
  int changeFirst = callData->chg_first;
  int changeLength = callData->chg_length;

  if(callData->text)
  {
   int length = callData->text->length;

   if(!callData->text->encoding_is_wchar)
   {
    String newString = callData->text->string.multi_byte;
    WString newWString = ToUnicode(newString, newString.GetLength(), CHARSET_UTF8);

    pfocusCtrl->ImePreEditSetText(newWString);
   }
  }
 }
```

```
}

void Ctrl::ImePreEditCaretCB(XIC xic, XPointer clientData, XIMPreeditCaretCallbackStruct
*callData)
{
 Ctrl *ctrl = (Ctrl*)clientData;
 Ptr<Ctrl> pfocusCtrl = ctrl->GetFocusCtrl();

 if(!pfocusCtrl)
  return;

 pfocusCtrl->imePreEditCursor = callData->position;
 pfocusCtrl->ImePreEditSyncCaret();
 pfocusCtrl->SyncCaret();
}
#endif
```

At X11Wnd.cpp

```
/*cw.xic = xim ? XCreateIC(xim,
                    XNInputStyle, XIMPreeditNothing|XIMStatusNothing,
                    XNClientWindow, w,
                    XNFocusWindow, w,
                    NULL)
             : NULL;*/
//Added for CJK
 //Using on the spot way
 XIMCallback     PreEditStartCallback;
 XIMCallback     PreEditDoneCallback;
 XIMCallback     PreEditDrawCallback;
 XIMCallback     PreEditCaretCallback;

 PreEditStartCallback.client_data = (XPointer)this;
 PreEditStartCallback.callback = (XIMProc)ImePreEditStartCB;

 PreEditDoneCallback.client_data = (XPointer)this;
 PreEditDoneCallback.callback = (XIMProc)ImePreEditDoneCB;

 PreEditDrawCallback.client_data = (XPointer)this;
 PreEditDrawCallback.callback = (XIMProc)ImePreEditDrawCB;

 PreEditCaretCallback.client_data = (XPointer)this;
 PreEditCaretCallback.callback = (XIMProc)ImePreEditCaretCB;

 XVaNestedList preEditAttribute;
 XPoint imePosition;
 XFontSet fs;
```

```
XIMStyle overSpot=XIMPreeditPosition | XIMStatusNothing;
XIMStyle onSpot=XIMPreeditCallbacks | XIMStatusNothing;
XIMStyle uppStyle = onSpot;
char                **missing_list;
  int                missing_count;
  char                *def_string;
fs = XCreateFontSet(Xdisplay, "-*-*-*-R-Normal--14-130-75-75-*-*", &missing_list,
                    &missing_count, &def_string);

imePosition.x=imePosition.y=0;
preEditAttribute = XVaCreateNestedList(NULL,
                        XNSpotLocation, &imePosition,
                        XNFontSet, fs,
                        XNPreeditStartCallback, &PreEditStartCallback,
                    XNPreeditDoneCallback, &PreEditDoneCallback,
                    XNPreeditDrawCallback, &PreEditDrawCallback,
                    XNPreeditCaretCallback, &PreEditCaretCallback,
                        NULL);

cw.xic = xim ? XCreateIC(xim,
                    XNInputStyle, uppStyle,
                    XNClientWindow, w,
                    XNFocusWindow, w,
                    XNPreeditAttributes, preEditAttribute,
                    NULL)
          : NULL;
XFree(preEditAttribute);
//End of Added for CJK
```

Subject: Re: Japanese IME test code
Posted by mirek on Wed, 14 May 2008 07:13:52 GMT
View Forum Message <> Reply to Message

Well, please.

Quote:
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.

Can we do just that before expanding to other areas?

I mean, can you fix GetIMEPoint from a couple of posts before, so that I can use that code finally in uppsrc for Win32 (at least, for now...).

Mirek

Subject: Re: Japanese IME test code
Posted by mobilehunter on Thu, 15 May 2008 09:57:48 GMT
View Forum Message <> Reply to Message

I tried the GetCaret.

But, still confused with view-relative point, please tell me a little about this.

Does the GetCaret's coordinate already in view-relative point?

File Attachments
1) imecompose.JPG, downloaded 1230 times

Subject: Re: Japanese IME test code
Posted by cbpporter on Thu, 15 May 2008 12:43:31 GMT
View Forum Message <> Reply to Message

Hi!

I tried to give your code a spin, but I can't seem to compile it because I'm missing types like CANDIDATEFORM and constants like CFS_CANDIDATEPOS (and a lot more). Are you using some extra includes?

Anyway if I can get this to compile it looks promising. Some adjustments of positioning still needed if I understand correctly. I never written code with the IME API, but that is a pop-up window so there must be a way to get it reduced in size so that the portion which underlines "ari" doesn't overlap the main window in such an ugly manner.

Subject: Re: Japanese IME test code
Posted by mobilehunter on Fri, 16 May 2008 05:20:33 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Thu, 15 May 2008 21:43Hi!

I tried to give your code a spin, but I can't seem to compile it because I'm missing types like CANDIDATEFORM and constants like CFS_CANDIDATEPOS (and a lot more). Are you using some extra includes?

Anyway if I can get this to compile it looks promising. Some adjustments of positioning still needed if I understand correctly. I never written code with the IME API, but that is a pop-up window so there must be a way to get it reduced in size so that the portion which underlines "ari" doesn't overlap the main window in such an ugly manner.

Just add #include "imm.h" to Win32Proc.cpp

Subject: Re: Japanese IME test code
Posted by mirek on Fri, 16 May 2008 07:14:21 GMT
View Forum Message <> Reply to Message

mobilehunter wrote on Thu, 15 May 2008 05:57I tried the GetCaret.

But, still confused with view-relative point, please tell me a little about this.

Does the GetCaret's coordinate already in view-relative point?

Basically, it does not matter. We can always fix the base GetIMEPoint implementation.

What matters is that GetIMEPoint should return view coordinates...

Mirek
