```
//=========================================
// Compiler: Gewoon
// Project:  AtsuiTest
// Bestand:  AtsuiTest.cp
//=========================================

#include "AtsuiTest.h"

int main()
{
    QDGlobals qd;
    InitGraf(&qd.thePort);
    ::GetDateTime(reinterpret_cast<UInt32*>(&qd.randSeed));
    InitWindows();

    Rect r;
    ::SetRect(&r,
              100,
              100,
              400,
              200);
    WindowPtr const w=::NewCWindow(nil,
                                   &r,
                                   "\pATSUI test",
                                   true,
                                   documentProc,
                                   reinterpret_cast<WindowPtr>(-1),
                                   false,
                                   0);
    if (!w)
        {
        return 0;
        }
    ::SetPort(w);

    const UInt32 len=31;
    Ptr const p=::NewPtr(len*sizeof(UniChar));
    if (!w)
        {
        return 0;
        }
    UniCharArrayPtr const t=reinterpret_cast<UniCharArrayPtr>(p);
    t[0]=208;
    t[1]=32;
    t[2]=222;
```

```
    t[3]=32;
    t[4]=240;
    t[5]=32;
    t[6]=254;
    t[7]=32;
    t[8]=257;
    t[9]=32;
    t[10]=259;
    t[11]=32;
    t[12]=261;
    t[13]=32;
    t[14]=267;
    t[15]=32;
    t[16]=269;
    t[17]=32;
    t[18]=273;
    t[19]=32;
    t[20]=285;
    t[21]=32;
    t[22]=295;
    t[23]=32;
    t[24]=305;
    t[25]=32;
    t[26]=337;
    t[27]=32;
    t[28]=537;
    t[29]=32;
    t[30]=8355;
    ConstUniCharArrayPtr const text=
        reinterpret_cast<ConstUniCharArrayPtr>(p);

    ATSUStyle style;
    OSStatus status=::ATSUCreateStyle(&style);
    if (status)
        {
        return 0;
        }
    ATSUAttributeTag tag[]={kATSUQDItalicTag};
    ByteCount valueSize[]={sizeof(bool)};
    bool c=true;
    ATSUAttributeValuePtr value[]={&c};
    status=::ATSUSetAttributes(style,
                               1,
                               tag,
                               valueSize,
```

```
                                value);
    if (status)
        {
        return 0;
        }

    UniCharCount runLengths[]={len};
    ATSUStyle styles[]={style};

    ATSUTextLayout textLayout;
    status=::ATSUCreateTextLayoutWithTextPtr(text,
                                             kATSUFromTextBeginning,
                                             len,
                                             kATSUToTextEnd,
                                             1,
                                             runLengths,
                                             styles,
                                             &textLayout);
    if (status)
        {
        return 0;
        }

    const ATSUTextMeasurement locationX=32<<16;
    const ATSUTextMeasurement locationY=32<<16;
    status=::ATSUDrawText(textLayout,
                          kATSUFromTextBeginning,
                          kATSUToTextEnd,
                          locationX,
                          locationY);
    if (status)
        {
        return 0;
        }

    while (true)
        {
        EventRecord event;
        const bool h=::WaitNextEvent(everyEvent,
                                     &event,
                                     6,
                                     nil);
        if (h)
            {
            if (event.what==mouseDown)
```

```
                {
                break;
                }
            if (event.what==keyDown)
                {
                break;
                }
            }
        else
            {
            status=::ATSUIdle(textLayout);
            if (status)
                {
                return 0;
                }
            }
        }

    status=::ATSUDisposeTextLayout(textLayout);
    if (status)
        {
        return 0;
        }

    status=::ATSUDisposeStyle(style);
    if (status)
        {
        return 0;
        }

    return 0;
}
```