Subject: Re: Using Pen with U++
Posted by Tom1 on Sun, 21 Mar 2021 11:29:36 GMT
View Forum Message <> Reply to Message

Hi,

Updated the testing code for completeness:

```
#include <CtrlLib/CtrlLib.h>

using namespace Upp;

#define LLOG(x)  DLOG(x)

struct MyApp : TopWindow {
 Point pos;

 Point lTriple;
 Point rTriple;
 Point lDouble;
 Point rDouble;
 Point lHold;
 Point rHold;

 bool lTriplePen;
 bool rTriplePen;
 bool lDoublePen;
 bool rDoublePen;
 bool lHoldPen;
 bool rHoldPen;

 Vector<Vector<Tuple<double, Pointf>>> drawing;
 Vector<Vector<Tuple<double, Pointf>>> rdrawing;

 bool rdown;
 bool ldown;

 Point p1,p2; // Tracker test variables
 Rect trect;

 MyApp(){
  rdown=false;
  ldown=false;
  lTriple=Null;
  rTriple=Null;
  lDouble=Null;
  rDouble=Null;
  lHold=Null;
```

```
 rHold=Null;

 lTriplePen=false;
 rTriplePen=false;
 lDoublePen=false;
 rDoublePen=false;
 lHoldPen=false;
 rHoldPen=false;

 trect=Null;
 p1=p2=Null;

 EnablePenSupport();
}

virtual void RightDown(Point p, dword){
 LLOG((IsPointerPen()?"PenRightDown":"MouseRightDown"));
 rdown=true;
 rdrawing.Add().Add(MakeTuple(IsPointerPen()?GetPenPressure():0.1, p));
 Refresh();
}

virtual void RightHold(Point p, dword){
 LLOG((IsPointerPen()?"PenRightHold":"MouseRightHold"));
 rHold=p;
 rHoldPen=IsPointerPen();
 Refresh();
}

virtual void RightDrag(Point p, dword){
 LLOG((IsPointerPen()?"PenRightDrag":"MouseRightDrag"));
}

virtual void RightDouble(Point p, dword){
 LLOG((IsPointerPen()?"PenRightDouble":"MouseRightDouble"));
 rDouble=p;
 rDoublePen=IsPointerPen();
 Refresh();
}

virtual void RightTriple(Point p, dword){
 LLOG((IsPointerPen()?"PenRightTriple":"MouseRightTriple"));
 rTriple=p;
 rTriplePen=IsPointerPen();
 Refresh();
}

virtual void RightUp(Point p, dword){
```

```
 LLOG((IsPointerPen()?"PenRightUp":"MouseRightUp"));
 rdown=false;
 Refresh();
}

virtual void LeftDown(Point p, dword keyflags){
 if(keyflags&K_SHIFT){
  RectTracker tracker(*this);
  tracker.sync= [=](Rect r) {  };
  tracker.MinSize(Size(-100000,-100000));
  trect=tracker.Track(Rect(p,p));
 }
 else if(keyflags&K_CTRL){
  RectTracker tracker(*this);
  p1=p;
  p2=tracker.TrackLine(p.x,p.y);
 }
 else{
  LLOG((IsPointerPen()?"PenLeftDown":"MouseLeftDown"));
  ldown=true;
  drawing.Add().Add(MakeTuple(IsPointerPen()?GetPenPressure():0.1, p));
 }
 Refresh();
}

virtual void LeftHold(Point p, dword){
 LLOG((IsPointerPen()?"PenLeftHold":"MouseLeftHold"));
 lHold=p;
 lHoldPen=IsPointerPen();
 Refresh();
}

virtual void LeftDrag(Point p, dword){
 LLOG((IsPointerPen()?"PenLeftDrag":"MouseLeftDrag"));
}

virtual void LeftDouble(Point p, dword){
 LLOG((IsPointerPen()?"PenLeftDouble":"MouseLeftDouble"));
 lDouble=p;
 lDoublePen=IsPointerPen();
 Refresh();
}

virtual void LeftTriple(Point p, dword){
 LLOG((IsPointerPen()?"PenLeftTriple":"MouseLeftTriple"));
 lTriple=p;
 lTriplePen=IsPointerPen();
 Refresh();
```

```
}

virtual void LeftUp(Point p, dword){
 LLOG((IsPointerPen()?"PenLeftUp":"MouseLeftUp"));
 ldown=false;
 Refresh();
}

Vector<String> report;

virtual void MouseMove(Point p, dword keyflags) {
 pos = p;

 LLOG((IsPointerPen()?"PenMove":"MouseMove"));

 if((ldown && drawing.GetCount()) || (rdown && rdrawing.GetCount())) {
  if(rdown){
   if(!IsPenHistoryEvent()) rdrawing.Top().Add(MakeTuple(IsPointerPen()?GetPenPressure():0.1,
p));
  }
  else if(ldown) drawing.Top().Add(MakeTuple(IsPointerPen()?GetPenPressure():0.1, p));
 }

 report.Clear();
 report.Add() << AsString(pos);
 report.Add() << "Pen: " << IsPointerPen();
 report.Add() << "Pressure: " << GetPenPressure();
 report.Add() << "Rotation: " << GetPenRotation();
 report.Add() << "Tilt: " << GetPenTilt();
 report.Add() << "Barrel: " << IsPenBarrelPressed();
 report.Add() << "Inverted: " << IsPenInverted();
 report.Add() << "Eraser: " << IsPenEraserPressed();

 Refresh();
}

virtual void Paint(Draw& w0){
 DrawPainter w(w0, GetSize());
 w.Clear(SColorPaper());

 w.LineCap(LINECAP_ROUND);

 for(const auto& stroke : drawing)
  if(stroke.GetCount())
   for(int i = 0; i < stroke.GetCount() - 1; i++) {
    w.Move(stroke[i].b);
    w.Line(stroke[i + 1].b);
    w.Stroke(DPI(20) * stroke[i].a, LtBlue());
```

```
    }

  for(const auto& stroke : rdrawing)
   if(stroke.GetCount())
    for(int i = 0; i < stroke.GetCount() - 1; i++) {
     w.Move(stroke[i].b);
     w.Line(stroke[i + 1].b);
     w.Stroke(DPI(20) * stroke[i].a, Black());
    }

  if(!IsNull(trect)) w.RectPath(trect).Stroke(2.0,Red());
  if(!IsNull(p2)) w.Move(p1).Line(p2).Stroke(2.0,Green());

  int fcy = GetStdFontCy();
  int y = 10;
  auto Text = [&] (const String& text) {
   w.DrawText(10, y, text);
   y += fcy;
  };

  Text("1. Test Clicks, Holds, Drags, DoubleClicks and TripleClicks using Mouse with both Left and
Right buttons.");
  Text("2. Test Clicks, Holds, Drags, DoubleClicks and TripleClicks using Pen without and with
Barrel button.");
  Text("3. Test RectTracker with Ctrl+LeftDrag and Shift+LeftDrag using Mouse and then Pen.");

  for(int i=0;i<report.GetCount();i++) Text(report[i]);

  if(!IsNull(lHold)) w.DrawText(lHold.x, lHold.y, "LeftHold", StdFont(), lHoldPen?LtBlue():Black());
  if(!IsNull(rHold)) w.DrawText(rHold.x, rHold.y, "RightHold", StdFont(),
rHoldPen?LtBlue():Black());
  if(!IsNull(lDouble)) w.DrawText(lDouble.x, lDouble.y, "LeftDouble", StdFont(),
lDoublePen?LtBlue():Black());
  if(!IsNull(rDouble)) w.DrawText(rDouble.x, rDouble.y, "RightDouble", StdFont(),
rDoublePen?LtBlue():Black());
  if(!IsNull(lTriple)) w.DrawText(lTriple.x, lTriple.y, "LeftTriple", StdFont(),
lTriplePen?LtBlue():Black());
  if(!IsNull(rTriple)) w.DrawText(rTriple.x, rTriple.y, "RightTriple", StdFont(),
rTriplePen?LtBlue():Black());
 }
};

GUI_APP_MAIN
{
 PromptOK("Please tap OK with Pen to verify button operation!");
 MyApp().Run();
}
```

Best regards,

Tom