Subject: Using Pen with U++
Posted by Tom1 on Wed, 03 Mar 2021 15:13:12 GMT
View Forum Message <> Reply to Message

Hi,

I just got a Wacom Intuos S (pen tablet). I'm planning on supporting using these devices with my software. Using the pen in place of the mouse works mostly as expected.

However, I would like to read the pen pressure in MouseMove(), (and possibly tilt/rotation some day too):

```
POINTER_INFO pi;
POINTER_PEN_INFO ppi;

UINT32 pressure=0;

if(GetPointerInfo(GET_POINTERID_WPARAM(wParam), &pi)) {
  if(pi.pointerType == PT_PEN) {
    UINT32 ppid = pi.pointerId;
    if(GetPointerPenInfo(ppid, &ppi)) pressure=ppi.pressure;
  }
}
```

,but I can't, since I cannot access wParam.

Would it be much trouble to add support inside CtrlCore for getting pen pressure for current MouseMove()?

Unfortunately GetPointerInfo() and GetPointerPenInfo() are only available in Windows 8 and later, so some conditioning is required.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Wed, 03 Mar 2021 16:06:05 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 03 March 2021 16:13Hi,

I just got a Wacom Intuos S (pen tablet). I'm planning on supporting using these devices with my software. Using the pen in place of the mouse works mostly as expected.

However, I would like to read the pen pressure in MouseMove(), (and possibly tilt/rotation some

day too):

```
POINTER_INFO pi;
POINTER_PEN_INFO ppi;

UINT32 pressure=0;

if(GetPointerInfo(GET_POINTERID_WPARAM(wParam), &pi)) {
  if(pi.pointerType == PT_PEN) {
    UINT32 ppid = pi.pointerId;
    if(GetPointerPenInfo(ppid, &ppi)) pressure=ppi.pressure;
  }
}
```

,but I can't, since I cannot access wParam.

Would it be much trouble to add support inside CtrlCore for getting pen pressure for current MouseMove()?

Unfortunately GetPointerInfo() and GetPointerPenInfo() are only available in Windows 8 and later, so some conditioning is required.

Best regards,

Tom

Sure and you can code around Windows 8 stuff (we do it here and there already, you just need to load the function pointer in the code).

Going to search for cheap tablet :)

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 03 Mar 2021 19:40:47 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thanks, sounds great!

I was too frugal to get one with tilt and/or rotation as I don't really need them at this time. So I can only test the pressure part here.

I wonder if Gdk support is straight forward...

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Tue, 09 Mar 2021 11:04:26 GMT
View Forum Message <> Reply to Message

Should be implemented in windows. This works on my computer:

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

using namespace Upp;

struct MyApp : TopWindow {
 Point pos;

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

 virtual void LeftDown(Point p, dword)
 {
  if(IsPointerPen())
   drawing.Add().Add(MakeTuple(GetPenPressure(), p));
  Refresh();
 }

 virtual void MouseMove(Point p, dword keyflags) {
  pos = p;
  if(IsPointerPen() && drawing.GetCount())
   drawing.Top().Add(MakeTuple(GetPenPressure(), p));
  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, Black());
    }
```

```
  int fcy = GetStdFontCy();
  int y = 10;
  auto Text = [&] (const String& text) {
   w.DrawText(10, y, text);
   y += fcy;
  };
  Text(AsString(pos));
  Text(String() << "Pen: " << IsPointerPen());
  Text(String() << "Pressure: " << GetPenPressure());
  Text(String() << "Rotation: " << GetPenRotation());
  Text(String() << "Tilt: " << GetPenTilt());
  Text(String() << "Barrel: " << IsPenBarrelPressed());
  Text(String() << "Inverted: " << IsPenInverted());
  Text(String() << "Eraser: " << IsPenInverted());
  Refresh();
 }
};

GUI_APP_MAIN
{
 MyApp().Run();
}
```

Cheap tablet I have bought does not have tilt/rotation support so I cannot test it, but the code is quite straightforward so there is a good chance it will work - it is good that you have that HW in yours :)

These changes should not break Win7 compatibility, once again would be nice if you tested that.

Subject: Re: Using Pen with U++
Posted by Tom1 on Tue, 09 Mar 2021 12:10:31 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thanks for getting this going! Unfortunately, my hardware returns false for IsPointerPen(). Based on RDUMPs I tried, it never even visits this code in Ctrl::WindowProc() :

```
 case WM_POINTERDOWN:
 case WM_POINTERUPDATE:
 case WM_POINTERUP:
  ...
```

I will try to dig in further to figure out what's going on...

As for Windows 7, I have none left... I hope someone around here still does.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Tue, 09 Mar 2021 12:27:29 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 09 March 2021 13:10Hi Mirek,

Thanks for getting this going! Unfortunately, my hardware returns false for IsPointerPen(). Based on RDUMPs I tried, it never even visits this code in Ctrl::WindowProc() :
 case WM_POINTERDOWN:
 case WM_POINTERUPDATE:
 case WM_POINTERUP:
  ...

I will try to dig in further to figure out what's going on...

As for Windows 7, I have none left... I hope someone around here still does.

Best regards,

Tom

Do you have proper drivers installed? I have found it is not quite easy to get the pen working...

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Tue, 09 Mar 2021 12:41:10 GMT
View Forum Message <> Reply to Message

Hi,

Yes, drivers are the current WACOM drivers. The pen does show pressure on Wacom Tablet Properties -control app.

Now I found that I had to enable 'Use Windows Ink' checkbox on Wacom Tablet Properties / Mapping page. Otherwise there will be no WM_POINTER... messages at all. However, this is unfortunate, since this causes drawing to start with a couple of centimeters long straight line

before I can draw any curves. Anyway, this gets the IsPointerPen() true!

I also found that:

```
if(ppi.penMask & PEN_MASK_ROTATION)
 pen_pressure = ppi.rotation * M_2PI / 360;
```

should be

```
if(ppi.penMask & PEN_MASK_ROTATION)
 pen_rotation = ppi.rotation * M_2PI / 360;
```

Then the thickness works based on pressure.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Tue, 09 Mar 2021 12:47:47 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 09 March 2021 13:41Hi,

Yes, drivers are the current WACOM drivers. The pen does show pressure on Wacom Tablet Properties -control app.

Now I found that I had to enable 'Use Windows Ink' checkbox on Wacom Tablet Properties / Mapping page. Otherwise there will be no WM_POINTER... messages at all. However, this is unfortunate, since this causes drawing to start with a couple of centimeters long straight line before I can draw any curves. Anyway, this gets the IsPointerPen() true!

I also found that:

```
if(ppi.penMask & PEN_MASK_ROTATION)
 pen_pressure = ppi.rotation * M_2PI / 360;
```

should be

```
if(ppi.penMask & PEN_MASK_ROTATION)
 pen_rotation = ppi.rotation * M_2PI / 360;
```

Then the thickness works based on pressure.

Best regards,

Tom

Thanks, thats a typo, fixed.

Subject: Re: Using Pen with U++
Posted by Tom1 on Tue, 09 Mar 2021 12:59:08 GMT
View Forum Message <> Reply to Message

Mirek, did you get a Wacom or something else?

When I start drawing with the pen (and using 'Use Windows Ink' mode), Windows first draws a circle around the pointer and then only after I have moved about two centimeters, it starts drawing... with the first two centimeters of straight line and only thereafter it accurately follows the pen.

I do not know, but this feels like a Windows Ink behavior. Anyway, it makes freehand drawing impossible. I need to figure out how to get rid of this pre-processing of pen movements.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Tue, 09 Mar 2021 13:18:56 GMT
View Forum Message <> Reply to Message

Hi,

I can now see that even when using 'Use Windows Ink' -mode, Microsoft's own 'Snip and Sketch' tool can draw tiny features without any 2 cm straight lines starters. There must be something that can be used to turn off that pre-processing programmatically.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Tue, 09 Mar 2021 13:37:26 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 09 March 2021 13:59Mirek, did you get a Wacom or something else?

When I start drawing with the pen (and using 'Use Windows Ink' mode), Windows first draws a circle around the pointer and then only after I have moved about two centimeters, it starts drawing... with the first two centimeters of straight line and only thereafter it accurately follows the pen.

I do not know, but this feels like a Windows Ink behavior. Anyway, it makes freehand drawing impossible. I need to figure out how to get rid of this pre-processing of pen movements.

Best regards,

Tom

XP-PEN. Knowing that you have Wacom, wanted to diversify...

Subject: Re: Using Pen with U++
Posted by mirek on Tue, 09 Mar 2021 13:39:05 GMT
View Forum Message <> Reply to Message

Are you testing with my code? Because I can only speak about that one and with xp-pen, it works fine with my example...

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Tue, 09 Mar 2021 14:07:06 GMT
View Forum Message <> Reply to Message

Diversifying input devices sounds a good idea. Especially now that they indeed work differently.

And yes, I'm using your testing code, since diversifying there does not sound too smart...

-

Anyway, now I found that I cannot get any WM_MOUSEMOVE from the first 2 cm on screen when using Wacom with Use Windows Ink enabled. However, all the moves right from start end up in WM_POINTERUPDATE. Now this means that we need to add the following code to WM_POINTERUPDATE for pen and disable it in WM_MOUSEMOVE for pen:

```
DoMouse(MOUSEMOVE, Point(lParam));
DoCursorShape();
```

However, there's a slight problem here. The lParam coordinates are for the screen not window, so

it needs some translation. I don't know yet how to do it...

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Tue, 09 Mar 2021 14:59:54 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 09 March 2021 15:07Diversifying input devices sounds a good idea. Especially now that they indeed work differently.

And yes, I'm using your testing code, since diversifying there does not sound too smart...

-

Anyway, now I found that I cannot get any WM_MOUSEMOVE from the first 2 cm on screen when using Wacom with Use Windows Ink enabled. However, all the moves right from start end up in WM_POINTERUPDATE. Now this means that we need to add the following code to WM_POINTERUPDATE for pen and disable it in WM_MOUSEMOVE for pen:

Disable part seems quite hard to do...

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Tue, 09 Mar 2021 15:56:58 GMT
View Forum Message <> Reply to Message

How about this approach using EnableMouseInPointer() as follows?

```
LRESULT Ctrl::WindowProc(UINT message, WPARAM wParam, LPARAM lParam) {
 GuiLock __;
 eventid++;
// LLOG("Ctrl::WindowProc(" << message << ") in " << ::Name(this) << ", focus " << (void
*)::GetFocus());
 Ptr<Ctrl> _this = this;
 HWND hwnd = GetHWND();

 ONCELOCK {
  EnableMouseInPointer(true); // #1 - enable mouse in WM_POINTER* messages
 };

 switch(message) {
```

```
 case WM_POINTERDOWN:
 case WM_POINTERUPDATE:
 case WM_POINTERUP:
  {

   POINT p = Point(lParam);
   ScreenToClient(hwnd, &p);

   pen = false;
   pen_pressure = pen_rotation = Null;
   pen_tilt = Null;
   pen_eraser = false;
   pen_barrel = false;
   pen_inverted = false;

   static BOOL (WINAPI *GetPointerType)(UINT32 pointerId, POINTER_INPUT_TYPE
*pointerType);
   static BOOL (WINAPI *GetPointerInfo)(UINT32 pointerId, POINTER_INFO *pointerInfo);
   static BOOL (WINAPI *GetPointerPenInfo)(UINT32 pointerId, POINTER_PEN_INFO *penInfo);
   static BOOL (WINAPI *GetPointerTouchInfo)(UINT32 pointerId, POINTER_TOUCH_INFO
*touchInfo);

   ONCELOCK {
    DllFn(GetPointerType, "User32.dll", "GetPointerType");
    DllFn(GetPointerInfo, "User32.dll", "GetPointerInfo");
    DllFn(GetPointerPenInfo, "User32.dll", "GetPointerPenInfo");
    DllFn(GetPointerTouchInfo, "User32.dll", "GetPointerTouchInfo");
   };

   POINTER_INPUT_TYPE pointerType;


   UINT32 pointerId = GET_POINTERID_WPARAM(wParam);
   if(GetPointerType && GetPointerPenInfo && GetPointerType(pointerId, &pointerType)) {
    switch(pointerType){
     case PT_PEN:{
      POINTER_PEN_INFO ppi;
      if(GetPointerPenInfo(pointerId, &ppi)) {
       pen = true;
       if(ppi.penFlags & PEN_FLAG_BARREL)
        pen_barrel = true;
       if(ppi.penFlags & PEN_FLAG_INVERTED)
        pen_inverted = true;
       if(ppi.penFlags & PEN_FLAG_ERASER)
        pen_eraser = true;
       if(ppi.penMask & PEN_MASK_PRESSURE)
        pen_pressure = ppi.pressure / 1024.0;
       if(ppi.penMask & PEN_MASK_ROTATION)
```

```
        pen_rotation = ppi.rotation * M_2PI / 360;
       if(ppi.penMask & PEN_MASK_TILT_X)
        pen_tilt.x = ppi.tiltX * M_2PI / 360;
       if(ppi.penMask & PEN_MASK_TILT_Y)
        pen_tilt.y = ppi.tiltY * M_2PI / 360;

      }
      break;
     }
     case PT_TOUCH:{
      POINTER_TOUCH_INFO pti;
      if(GetPointerTouchInfo(pointerId, &pti)) {
       // Add something touch specific here some day maybe...
      }
      break;
     }
     default:{
      POINTER_INFO pi;
      if(GetPointerInfo(pointerId, &pi)) {
      }
      break;
     }
    }

    if(_this) switch(message){
     case WM_POINTERDOWN:
      ClickActivateWnd();
      DoMouse(LEFTDOWN, Point(p), 0);
      PostInput();
      break;
     case WM_POINTERUP:
      DoMouse(LEFTUP, Point(p), 0);
      PostInput();
      break;
     case WM_POINTERUPDATE:
      DoMouse(MOUSEMOVE, Point(p));
      DoCursorShape();
      break;
    }

   }
  }
  break;
 case WM_POINTERLEAVE:
  pen = false;
  break;
```

Further more, this requires disabling WM_LEFTUP, WM_LEFTDOWN and WM_MOUSEMOVE as they now come in as WM_POINTER* messages.

IMPORTANT WARNING: I have not replicated their function completely in WM_POINTER above as I do not quite understand all the finer details there. Also, I'm worried about breaking everything, so I'll just throw this in 'as is' for you to consider...

Oh and yes, this fixes the 2 cm straight line start issue... :)

Best regards,

Tom

EDIT: EnableMouseInPointer() is available on Windows8 and above only, so it needs to be loaded in a similar way as the rest of the Pointer -functions.

Subject: Re: Using Pen with U++
Posted by mirek on Tue, 09 Mar 2021 16:48:14 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 09 March 2021 16:56How about this approach using EnableMouseInPointer() as follows?

```
LRESULT Ctrl::WindowProc(UINT message, WPARAM wParam, LPARAM lParam) {
 GuiLock __;
 eventid++;
// LLOG("Ctrl::WindowProc(" << message << ") in " << ::Name(this) << ", focus " << (void *)::GetFocus());
 Ptr<Ctrl> _this = this;
 HWND hwnd = GetHWND();

 ONCELOCK {
  EnableMouseInPointer(true); // #1 - enable mouse in WM_POINTER* messages
 };

 switch(message) {
 case WM_POINTERDOWN:
 case WM_POINTERUPDATE:
 case WM_POINTERUP:
  {

   POINT p = Point(lParam);
   ScreenToClient(hwnd, &p);

   pen = false;
   pen_pressure = pen_rotation = Null;
   pen_tilt = Null;
   pen_eraser = false;
```

```
   pen_barrel = false;
   pen_inverted = false;

   static BOOL (WINAPI *GetPointerType)(UINT32 pointerId, POINTER_INPUT_TYPE
*pointerType);
   static BOOL (WINAPI *GetPointerInfo)(UINT32 pointerId, POINTER_INFO *pointerInfo);
   static BOOL (WINAPI *GetPointerPenInfo)(UINT32 pointerId, POINTER_PEN_INFO *penInfo);
   static BOOL (WINAPI *GetPointerTouchInfo)(UINT32 pointerId, POINTER_TOUCH_INFO
*touchInfo);

   ONCELOCK {
    DllFn(GetPointerType, "User32.dll", "GetPointerType");
    DllFn(GetPointerInfo, "User32.dll", "GetPointerInfo");
    DllFn(GetPointerPenInfo, "User32.dll", "GetPointerPenInfo");
    DllFn(GetPointerTouchInfo, "User32.dll", "GetPointerTouchInfo");
   };

   POINTER_INPUT_TYPE pointerType;


   UINT32 pointerId = GET_POINTERID_WPARAM(wParam);
   if(GetPointerType && GetPointerPenInfo && GetPointerType(pointerId, &pointerType)) {
    switch(pointerType){
     case PT_PEN:{
      POINTER_PEN_INFO ppi;
      if(GetPointerPenInfo(pointerId, &ppi)) {
       pen = true;
       if(ppi.penFlags & PEN_FLAG_BARREL)
        pen_barrel = true;
       if(ppi.penFlags & PEN_FLAG_INVERTED)
        pen_inverted = true;
       if(ppi.penFlags & PEN_FLAG_ERASER)
        pen_eraser = true;
       if(ppi.penMask & PEN_MASK_PRESSURE)
        pen_pressure = ppi.pressure / 1024.0;
       if(ppi.penMask & PEN_MASK_ROTATION)
        pen_rotation = ppi.rotation * M_2PI / 360;
       if(ppi.penMask & PEN_MASK_TILT_X)
        pen_tilt.x = ppi.tiltX * M_2PI / 360;
       if(ppi.penMask & PEN_MASK_TILT_Y)
        pen_tilt.y = ppi.tiltY * M_2PI / 360;

      }
      break;
     }
     case PT_TOUCH:{
      POINTER_TOUCH_INFO pti;
      if(GetPointerTouchInfo(pointerId, &pti)) {
```

```
      // Add something touch specific here some day maybe...
     }
     break;
    }
    default:{
     POINTER_INFO pi;
     if(GetPointerInfo(pointerId, &pi)) {
     }
     break;
    }
   }

   if(_this) switch(message){
    case WM_POINTERDOWN:
     ClickActivateWnd();
     DoMouse(LEFTDOWN, Point(p), 0);
     PostInput();
     break;
    case WM_POINTERUP:
     DoMouse(LEFTUP, Point(p), 0);
     PostInput();
     break;
    case WM_POINTERUPDATE:
     DoMouse(MOUSEMOVE, Point(p));
     DoCursorShape();
     break;
   }

  }
 }
 break;
case WM_POINTERLEAVE:
 pen = false;
 break;
```

Further more, this requires disabling WM_LEFTUP, WM_LEFTDOWN and WM_MOUSEMOVE as they now come in as WM_POINTER* messages.

IMPORTANT WARNING: I have not replicated their function completely in WM_POINTER above as I do not quite understand all the finer details there. Also, I'm worried about breaking everything, so I'll just throw this in 'as is' for you to consider...

Oh and yes, this fixes the 2 cm straight line start issue... :)

Best regards,

Tom

EDIT: EnableMouseInPointer() is available on Windows8 and above only, so it needs to be loaded in a similar way as the rest of the Pointer -functions.

Looks good. Will try tommorow.

Subject: Re: Using Pen with U++
Posted by Tom1 on Tue, 09 Mar 2021 17:14:18 GMT
View Forum Message <> Reply to Message

OK, good! But please try this one instead, as it takes into account the Win8 thing and also adds conditional blocking of WM_MOUSEMOVE, WM_LBUTTONUP, WM_LBUTTONDOWN:

```
LRESULT Ctrl::WindowProc(UINT message, WPARAM wParam, LPARAM lParam) {
 GuiLock __;
 eventid++;
// LLOG("Ctrl::WindowProc(" << message << ") in " << ::Name(this) << ", focus " << (void
*)::GetFocus());
 Ptr<Ctrl> _this = this;
 HWND hwnd = GetHWND();

 static bool disableOldWMs=false; // When true, blocks out original WM_MOUSEMOVE,
WM_LBUTTONUP, WM_LBUTTONDOWN


 switch(message) {
 case WM_POINTERDOWN:
 case WM_POINTERUPDATE:
 case WM_POINTERUP:
  {

   POINT p = Point(lParam);
   ScreenToClient(hwnd, &p);

   pen = false;
   pen_pressure = pen_rotation = Null;
   pen_tilt = Null;
   pen_eraser = false;
   pen_barrel = false;
   pen_inverted = false;

   static BOOL (WINAPI *EnableMouseInPointer)(WINBOOL fEnable);
   static BOOL (WINAPI *GetPointerType)(UINT32 pointerId, POINTER_INPUT_TYPE
*pointerType);
   static BOOL (WINAPI *GetPointerInfo)(UINT32 pointerId, POINTER_INFO *pointerInfo);
   static BOOL (WINAPI *GetPointerPenInfo)(UINT32 pointerId, POINTER_PEN_INFO *penInfo);
   static BOOL (WINAPI *GetPointerTouchInfo)(UINT32 pointerId, POINTER_TOUCH_INFO
```

```
*touchInfo);

   ONCELOCK {
    DllFn(EnableMouseInPointer, "User32.dll", "EnableMouseInPointer");
    if(EnableMouseInPointer && EnableMouseInPointer(true)) disableOldWMs=true; // Switching
over to WM_POINTER* functions for mouse

    DllFn(GetPointerType, "User32.dll", "GetPointerType");
    DllFn(GetPointerInfo, "User32.dll", "GetPointerInfo");
    DllFn(GetPointerPenInfo, "User32.dll", "GetPointerPenInfo");
    DllFn(GetPointerTouchInfo, "User32.dll", "GetPointerTouchInfo");
   };

   POINTER_INPUT_TYPE pointerType;


   UINT32 pointerId = GET_POINTERID_WPARAM(wParam);
   if(GetPointerType && GetPointerPenInfo && GetPointerType(pointerId, &pointerType)) {
    switch(pointerType){
     case PT_PEN:{
      POINTER_PEN_INFO ppi;
      if(GetPointerPenInfo(pointerId, &ppi)) {
       pen = true;
       if(ppi.penFlags & PEN_FLAG_BARREL)
        pen_barrel = true;
       if(ppi.penFlags & PEN_FLAG_INVERTED)
        pen_inverted = true;
       if(ppi.penFlags & PEN_FLAG_ERASER)
        pen_eraser = true;
       if(ppi.penMask & PEN_MASK_PRESSURE)
        pen_pressure = ppi.pressure / 1024.0;
       if(ppi.penMask & PEN_MASK_ROTATION)
        pen_rotation = ppi.rotation * M_2PI / 360;
       if(ppi.penMask & PEN_MASK_TILT_X)
        pen_tilt.x = ppi.tiltX * M_2PI / 360;
       if(ppi.penMask & PEN_MASK_TILT_Y)
        pen_tilt.y = ppi.tiltY * M_2PI / 360;

      }
      break;
     }
     case PT_TOUCH:{
      POINTER_TOUCH_INFO pti;
      if(GetPointerTouchInfo(pointerId, &pti)) {
       // Add something touch specific here some day maybe...
      }
      break;
     }
```

```
     default:{
      POINTER_INFO pi;
      if(GetPointerInfo(pointerId, &pi)) {
      }
      break;
     }
    }

    if(_this) switch(message){
     case WM_POINTERDOWN:
      ClickActivateWnd();
      DoMouse(LEFTDOWN, Point(p), 0);
      PostInput();
      break;
     case WM_POINTERUP:
      DoMouse(LEFTUP, Point(p), 0);
      PostInput();
      break;
     case WM_POINTERUPDATE:
      DoMouse(MOUSEMOVE, Point(p));
      DoCursorShape();
      break;
    }

   }
  }
  break;
 case WM_POINTERLEAVE:
  pen = false;
  break;
...

 case WM_LBUTTONDOWN:
  if(disableOldWMs) break;
...

 case WM_LBUTTONUP:
  if(disableOldWMs) break;
...

 case WM_MOUSEMOVE:
  if(disableOldWMs) break;
```

Without native POINTER sources, this one should use the original code all the way, I hope...

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Thu, 11 Mar 2021 10:21:57 GMT
View Forum Message <> Reply to Message

So I have some bad news, even if somewhat funny news:

With your code, it starts drawing those nasty initial lines with my XP-PEN... :(:d

Subject: Re: Using Pen with U++
Posted by Tom1 on Thu, 11 Mar 2021 10:53:42 GMT
View Forum Message <> Reply to Message

:lol:

OK, if I add this to the beginning of WindowProc():

```
 switch(message) {
  case WM_POINTERDOWN:
   RLOG("WM_POINTERDOWN");
   break;
  case WM_POINTERUPDATE:
   RLOG("WM_POINTERUPDATE");
   break;
  case WM_POINTERUP:
   RLOG("WM_POINTERUP");
   break;
  case WM_MOUSEMOVE:
   RLOG("WM_MOUSEMOVE");
   break;
  case WM_LBUTTONUP:
   RLOG("WM_LBUTTONUP");
   break;
  case WM_LBUTTONDOWN:
   RLOG("WM_LBUTTONDOWN");
   break;
 }
```

I will get this sequence for drawing one line:

...
WM_POINTERUPDATE
WM_MOUSEMOVE

WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERDOWN
WM_POINTERUPDATE
WM_POINTERUPDATE
...
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_LBUTTONDOWN
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
...
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUP
WM_MOUSEMOVE
WM_LBUTTONUP
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
...

How about you?

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Thu, 11 Mar 2021 13:16:17 GMT
View Forum Message <> Reply to Message

Added some log to the testing code too.

```
 virtual void LeftDown(Point p, dword)
 {
  if(IsPointerPen()) {
   DLOG("Start line");
   drawing.Add().Add(MakeTuple(GetPenPressure(), p));
  }
  Refresh();
 }

 virtual void MouseMove(Point p, dword keyflags) {
  pos = p;
  if(IsPointerPen() && drawing.GetCount()) {
   DLOG("Drawing line, pressure: " << GetPenPressure());
   drawing.Top().Add(MakeTuple(GetPenPressure(), p));
  }
  Refresh();
 }
```

```
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
```

WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERDOWN
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_LBUTTONDOWN
Start line
WM_MOUSEMOVE
Drawing line, pressure: 0.109375
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.12890625
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.1318359375
WM_POINTERUPDATE
WM_MOUSEMOVE

Drawing line, pressure: 0.1337890625
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.1435546875
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.14453125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.150390625
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.1689453125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.1767578125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.205078125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.220703125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2421875
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.24609375
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2578125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2607421875
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2607421875
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2607421875
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2578125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2578125
WM_POINTERUPDATE
WM_MOUSEMOVE

Drawing line, pressure: 0.2548828125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.25390625
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2548828125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2548828125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.267578125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2666015625
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2666015625
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2685546875
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.267578125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.267578125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.267578125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.26953125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2705078125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.2705078125
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.271484375
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.1015625
WM_POINTERUPDATE
WM_POINTERUP

WM_LBUTTONUP
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_MOUSEMOVE
WM_MOUSEMOVE
WM_MOUSEMOVE
WM_MOUSEMOVE
WM_MOUSEMOVE
WM_MOUSEMOVE

Looking at it, maybe the problem is actually in testing code, at least partially? (drawing line

continues after mouseup).

Also note that I am not getting any other WM_POINTER message than UPDATE....

P.S.: Why RLOG? Do you want to forget it in the code? :)

Subject: Re: Using Pen with U++
Posted by Tom1 on Thu, 11 Mar 2021 13:41:44 GMT
View Forum Message <> Reply to Message

Actually you do get those WM_POINTERDOWN and WM_POINTERUP. From your log:
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERDOWN <<< HERE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_LBUTTONDOWN
Start line
WM_MOUSEMOVE
Drawing line, pressure: 0.109375
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.12890625

...

WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.271484375
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0.1015625
WM_POINTERUPDATE
WM_POINTERUP <<< HERE
WM_LBUTTONUP
WM_POINTERUPDATE
WM_MOUSEMOVE
Drawing line, pressure: 0
WM_POINTERUPDATE
WM_MOUSEMOVE

Drawing line, pressure: 0
WM_POINTERUPDATE
WM_MOUSEMOVE

What I find strange is that in the end, where you (I guess) take over with mouse, you do not get WM_POINTERUPDATEs. This suggests that the EnableMouseInPointer() is not working for you. Also, all your drawing takes place after WM_MOUSEMOVE and not WM_POINTERUPDATE, for some reason...

It is no wonder the drawing continues at zero width after mouse up, since there is no code for LeftUp to stop it.

Well, my RLOGs... I just use DEBUG mode only after getting in deep trouble... :)

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Thu, 11 Mar 2021 13:54:33 GMT
View Forum Message <> Reply to Message

OK, here's a fix for the testing code:

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

using namespace Upp;

struct MyApp : TopWindow {
 Point pos;

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

 bool pendown;

 MyApp(){
  pendown=false;
 }

 virtual void LeftUp(Point p, dword)
 {
  if(IsPointerPen()) {
   DLOG("End line");
   pendown=false;
  }
  Refresh();
 }
```

```
virtual void LeftDown(Point p, dword)
{
 if(IsPointerPen()) {
  DLOG("Start line");
  pendown=true;
  drawing.Add().Add(MakeTuple(GetPenPressure(), p));
 }
 Refresh();
}

virtual void MouseMove(Point p, dword keyflags) {
 pos = p;
 if(IsPointerPen() && drawing.GetCount() && pendown) {
  DLOG("Drawing line, pressure: " << GetPenPressure());
  drawing.Top().Add(MakeTuple(GetPenPressure(), p));
 }
 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, Black());
   }

 int fcy = GetStdFontCy();
 int y = 10;
 auto Text = [&] (const String& text) {
  w.DrawText(10, y, text);
  y += fcy;
 };
 Text(AsString(pos));
 Text(String() << "Pen: " << IsPointerPen());
 Text(String() << "Pressure: " << GetPenPressure());
 Text(String() << "Rotation: " << GetPenRotation());
 Text(String() << "Tilt: " << GetPenTilt());
 Text(String() << "Barrel: " << IsPenBarrelPressed());
 Text(String() << "Inverted: " << IsPenInverted());
 Text(String() << "Eraser: " << IsPenEraserPressed()); // FIXED to IsPenEraserPressed()
```

```
  //Refresh(); // REMOVED
 }
};

GUI_APP_MAIN
{
 MyApp().Run();
}
```

And results for single line draw:

```
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERDOWN
Start line
WM_POINTERUPDATE
Drawing line, pressure: 0.107421875
WM_POINTERUPDATE
Drawing line, pressure: 0.154296875
WM_POINTERUPDATE
Drawing line, pressure: 0.205078125
WM_POINTERUPDATE
Drawing line, pressure: 0.2255859375
WM_POINTERUPDATE
Drawing line, pressure: 0.2255859375
WM_POINTERUPDATE
Drawing line, pressure: 0.2255859375
WM_POINTERUPDATE
Drawing line, pressure: 0.228515625
WM_POINTERUPDATE
Drawing line, pressure: 0.2451171875
WM_POINTERUPDATE
Drawing line, pressure: 0.2451171875
WM_MOUSEMOVE
WM_LBUTTONDOWN
WM_MOUSEMOVE
WM_POINTERUPDATE
Drawing line, pressure: 0.244140625
WM_MOUSEMOVE
WM_POINTERUPDATE
Drawing line, pressure: 0.263671875
WM_MOUSEMOVE
WM_POINTERUPDATE
Drawing line, pressure: 0.2802734375
```

WM_MOUSEMOVE
WM_POINTERUPDATE
Drawing line, pressure: 0.29296875
WM_MOUSEMOVE
WM_POINTERUPDATE
Drawing line, pressure: 0.3056640625
WM_MOUSEMOVE
WM_POINTERUPDATE
Drawing line, pressure: 0.3193359375

...

Drawing line, pressure: 0.4169921875
WM_MOUSEMOVE
WM_POINTERUPDATE
Drawing line, pressure: 0.4052734375
WM_MOUSEMOVE
WM_POINTERUPDATE
Drawing line, pressure: 0.37890625
WM_MOUSEMOVE
WM_POINTERUPDATE
Drawing line, pressure: 0.365234375
WM_MOUSEMOVE
WM_POINTERUPDATE
Drawing line, pressure: 0.3359375
WM_MOUSEMOVE
WM_POINTERUPDATE
Drawing line, pressure: 0.0625
WM_MOUSEMOVE
WM_POINTERUP
End line
WM_LBUTTONUP
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE
WM_POINTERUPDATE
WM_MOUSEMOVE

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Thu, 11 Mar 2021 15:26:26 GMT
View Forum Message <> Reply to Message

Hi,

This change fixes some tails when drawing fast. (This actually brings in your original 'ignoreclick' code and more.)

```
 static bool disableOldWMs=false; // When true, blocks out original WM_MOUSEMOVE, WM_LBUTTONUP, WM_LBUTTONDOWN

 switch(message) {
 case WM_POINTERDOWN:
 case WM_POINTERUPDATE:
 case WM_POINTERUP:
  {

   POINT p = Point((LONG)lParam);
   ScreenToClient(hwnd, &p);

   pen = false;
   pen_pressure = pen_rotation = Null;
   pen_tilt = Null;
   pen_eraser = false;
   pen_barrel = false;
   pen_inverted = false;

   static BOOL (WINAPI *EnableMouseInPointer)(BOOL fEnable);
   static BOOL (WINAPI *GetPointerType)(UINT32 pointerId, POINTER_INPUT_TYPE *pointerType);
   static BOOL (WINAPI *GetPointerInfo)(UINT32 pointerId, POINTER_INFO *pointerInfo);
   static BOOL (WINAPI *GetPointerPenInfo)(UINT32 pointerId, POINTER_PEN_INFO *penInfo);
   static BOOL (WINAPI *GetPointerTouchInfo)(UINT32 pointerId, POINTER_TOUCH_INFO *touchInfo);

   ONCELOCK {
    DllFn(EnableMouseInPointer, "User32.dll", "EnableMouseInPointer");
    if(EnableMouseInPointer && EnableMouseInPointer(true)) disableOldWMs=true; // Switching over to WM_POINTER* functions for mouse

    DllFn(GetPointerType, "User32.dll", "GetPointerType");
```

```
 DllFn(GetPointerInfo, "User32.dll", "GetPointerInfo");
 DllFn(GetPointerPenInfo, "User32.dll", "GetPointerPenInfo");
 DllFn(GetPointerTouchInfo, "User32.dll", "GetPointerTouchInfo");
};

POINTER_INPUT_TYPE pointerType;


UINT32 pointerId = GET_POINTERID_WPARAM(wParam);
if(GetPointerType && GetPointerType(pointerId, &pointerType)) {
 switch(pointerType){
  case PT_PEN:{
   POINTER_PEN_INFO ppi;
   if(GetPointerPenInfo && GetPointerPenInfo(pointerId, &ppi)) {
    pen = true;
    if(ppi.penFlags & PEN_FLAG_BARREL)
     pen_barrel = true;
    if(ppi.penFlags & PEN_FLAG_INVERTED)
     pen_inverted = true;
    if(ppi.penFlags & PEN_FLAG_ERASER)
     pen_eraser = true;
    if(ppi.penMask & PEN_MASK_PRESSURE)
     pen_pressure = ppi.pressure / 1024.0;
    if(ppi.penMask & PEN_MASK_ROTATION)
     pen_rotation = ppi.rotation * M_2PI / 360;
    if(ppi.penMask & PEN_MASK_TILT_X)
     pen_tilt.x = ppi.tiltX * M_2PI / 360;
    if(ppi.penMask & PEN_MASK_TILT_Y)
     pen_tilt.y = ppi.tiltY * M_2PI / 360;

   }
   break;
  }
  case PT_TOUCH:{
   POINTER_TOUCH_INFO pti;
   if(GetPointerTouchInfo && GetPointerTouchInfo(pointerId, &pti)) {
    // Add something touch specific here some day maybe...
   }
   break;
  }
  default:{
   POINTER_INFO pi;
   if(GetPointerInfo && GetPointerInfo(pointerId, &pi)) {
   }
   break;
  }
 }
```

```
    switch(message){
     case WM_POINTERDOWN:
      ClickActivateWnd();
      if(ignoreclick) return 0L;
      DoMouse(LEFTDOWN, Point(p), 0);
      if(_this) PostInput();
      break;
     case WM_POINTERUP:
      if(ignoreclick) EndIgnore();
      else DoMouse(LEFTUP, Point(p), 0);
      if(_this) PostInput();
      break;
     case WM_POINTERUPDATE:
      if(ignoreclick) {
       EndIgnore();
       return 0L;
      }
      if(_this) DoMouse(MOUSEMOVE, Point(p));
      DoCursorShape();
      break;

    }
    return 0L;
   }
  }
  break;
 case WM_POINTERLEAVE:
  pen = false;
  break;
```

The 'if(disableOldWMs) break;' are still needed in WM_MOUSEMOVE, WM_LBUTTONUP, WM_LBUTTONDOWN.

Does this work for you?

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 12 Mar 2021 08:37:10 GMT
View Forum Message <> Reply to Message

It seems to work, however I feel pretty uneasy that it is handling just LBUTTON messages and leaves the rest to "old" WMs. What about right button and double-clicks? Documentation is pretty

sparse there....

If I understand you right, buttons are not a problem with Wacom, just mousemove messages. Maybe we could be more conservative, leave old messages, issue WM_MOUSEMOVE on POINTERUPDATE and solve the problem by ignoring duplicit messages (store screen hwnd and position, only issue mousemove if it changes...)?

Mirek

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 12 Mar 2021 08:52:44 GMT
View Forum Message <> Reply to Message

Or, maybe, just forget about MouseMove and add a new virtual method? E.g.

virtual Pen(Point pos, double pressure, double rotation, Pointf tilt, dword flags);

which would simply be updated on WM_POINTERUPDATE....

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 12 Mar 2021 09:18:58 GMT
View Forum Message <> Reply to Message

BTW, what do you get with

DDUMPHEX(GetMessageExtraInfo());

It looks like that 0xff515704 indicates pen for "old" WM messages, which could be useful in some context...

https://stackoverflow.com/questions/29857587/detect-if-wm-mo usemove-is-caused-by-touch-pen

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 12 Mar 2021 09:26:20 GMT
View Forum Message <> Reply to Message

Hi,

WM_LBUTTONDOWN comes in late with Wacom, so it needs to be caught with WM_POINTERDOWN. WM_MOUSEMOVEs are also missing in the beginning of draw, so WM_POINTERUPDATE needs to replace WM_MOUSEMOVE there.

The only way (I can see) to properly handle each pointing device when more than one exist, is taking the full control of moves and left button to WM_POINTER* messages.

I think that we should aim for the Pen to work in place of a mouse as logically and cleanly as possible without any changes to the final application. I'm thinking of people using pen with U++ applications without specific pen support. Their experience should not suffer.

Only when pressure/tilt/rotation are beneficial for the application, then additional calls should be placed to acquire their values.

If I understand it correctly, the RBUTTON and DOUBLECLICKS are not part of WM_POINTER* message family. Anyway, at least RBUTTON works through the old messages. I have not tried doubleclicking yet... I'll do it after I get back to my tablet later in the afternoon.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 12 Mar 2021 09:28:48 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 12 March 2021 11:18BTW, what do you get with

DDUMPHEX(GetMessageExtraInfo());

It looks like that 0xff515704 indicates pen for "old" WM messages, which could be useful in some context...

https://stackoverflow.com/questions/29857587/detect-if-wm-mo
usemove-is-caused-by-touch-pen

Mirek

I'll need to check this when I get to my tablet...

You're moving fast! Just cannot keep up writing my replies... :)

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 12 Mar 2021 10:56:27 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 12 March 2021 11:28mirek wrote on Fri, 12 March 2021 11:18BTW, what do you get with

DDUMPHEX(GetMessageExtraInfo());

It looks like that 0xff515704 indicates pen for "old" WM messages, which could be useful in some context...

https://stackoverflow.com/questions/29857587/detect-if-wm-mousemove-is-caused-by-touch-pen

Mirek

I'll need to check this when I get to my tablet...

You're moving fast! Just cannot keep up writing my replies... :)

Best regards,

Tom

It's GetMessageExtraInfo() = 0x0 all the way for WM_POINTER*, WM_MOUSEMOVE, WM_LBUTTONUP, WM_LBUTTONDOWN...

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 12 Mar 2021 11:35:44 GMT
View Forum Message <> Reply to Message

Hi,

It seems, when the pen is moving fast, Windows filters out part of the intermediate pen positions. Therefore, processing GetPointerPenInfoHistory() improves drawing quality especially at higher pen speeds:

```
 static bool disableOldWMs=false; // When true, blocks out original WM_MOUSEMOVE, WM_LBUTTONUP, WM_LBUTTONDOWN

 switch(message) {
 case WM_POINTERDOWN:
 case WM_POINTERUPDATE:
 case WM_POINTERUP:
  {
```

```
  POINT p = Point((LONG)lParam);
  ScreenToClient(hwnd, &p);

  pen = false;
  pen_pressure = pen_rotation = Null;
  pen_tilt = Null;
  pen_eraser = false;
  pen_barrel = false;
  pen_inverted = false;

  static BOOL (WINAPI *EnableMouseInPointer)(BOOL fEnable);
  static BOOL (WINAPI *GetPointerType)(UINT32 pointerId, POINTER_INPUT_TYPE
*pointerType);
  static BOOL (WINAPI *GetPointerInfo)(UINT32 pointerId, POINTER_INFO *pointerInfo);
  static BOOL (WINAPI *GetPointerPenInfo)(UINT32 pointerId, POINTER_PEN_INFO *penInfo);
  static BOOL (WINAPI *GetPointerTouchInfo)(UINT32 pointerId, POINTER_TOUCH_INFO
*touchInfo);
  static BOOL (WINAPI *GetPointerPenInfoHistory)(UINT32 pointerId, UINT32 *entriesCount,
POINTER_PEN_INFO *penInfo);

  ONCELOCK {
   DllFn(EnableMouseInPointer, "User32.dll", "EnableMouseInPointer");
   if(EnableMouseInPointer && EnableMouseInPointer(true)) disableOldWMs=true; // Switching
over to WM_POINTER* functions for mouse

   DllFn(GetPointerType, "User32.dll", "GetPointerType");
   DllFn(GetPointerInfo, "User32.dll", "GetPointerInfo");
   DllFn(GetPointerPenInfo, "User32.dll", "GetPointerPenInfo");
   DllFn(GetPointerTouchInfo, "User32.dll", "GetPointerTouchInfo");

   DllFn(GetPointerPenInfoHistory, "User32.dll", "GetPointerPenInfoHistory");
  };

  POINTER_INPUT_TYPE pointerType;


  UINT32 pointerId = GET_POINTERID_WPARAM(wParam);
  if(GetPointerType && GetPointerType(pointerId, &pointerType)) {
   switch(pointerType){
    case PT_PEN:{

     UINT32 hc=256;
     POINTER_PEN_INFO ppit[hc];
     if(message==WM_POINTERUPDATE && GetPointerPenInfoHistory &&
GetPointerPenInfoHistory(pointerId, &hc, ppit)) {
      for(int i=hc-1;i>=0;i--){
       pen = true;
       if(ppit[i].penFlags & PEN_FLAG_BARREL)
```

```
   pen_barrel = true;
  if(ppit[i].penFlags & PEN_FLAG_INVERTED)
   pen_inverted = true;
  if(ppit[i].penFlags & PEN_FLAG_ERASER)
   pen_eraser = true;
  if(ppit[i].penMask & PEN_MASK_PRESSURE)
   pen_pressure = ppit[i].pressure / 1024.0;
  if(ppit[i].penMask & PEN_MASK_ROTATION)
   pen_rotation = ppit[i].rotation * M_2PI / 360;
  if(ppit[i].penMask & PEN_MASK_TILT_X)
   pen_tilt.x = ppit[i].tiltX * M_2PI / 360;
  if(ppit[i].penMask & PEN_MASK_TILT_Y)
   pen_tilt.y = ppit[i].tiltY * M_2PI / 360;

  POINT hp = ppit[i].pointerInfo.ptPixelLocation;
  ScreenToClient(hwnd, &hp);

  if(ignoreclick) {
   EndIgnore();
   return 0L;
  }
  if(_this) DoMouse(MOUSEMOVE, Point(hp));
  DoCursorShape();

 }
 return 0L;
}

POINTER_PEN_INFO ppi;
if(GetPointerPenInfo && GetPointerPenInfo(pointerId, &ppi)) {
 if(ppi.pointerInfo.historyCount){
 }

 pen = true;
 if(ppi.penFlags & PEN_FLAG_BARREL)
  pen_barrel = true;
 if(ppi.penFlags & PEN_FLAG_INVERTED)
  pen_inverted = true;
 if(ppi.penFlags & PEN_FLAG_ERASER)
  pen_eraser = true;
 if(ppi.penMask & PEN_MASK_PRESSURE)
  pen_pressure = ppi.pressure / 1024.0;
 if(ppi.penMask & PEN_MASK_ROTATION)
  pen_rotation = ppi.rotation * M_2PI / 360;
 if(ppi.penMask & PEN_MASK_TILT_X)
  pen_tilt.x = ppi.tiltX * M_2PI / 360;
 if(ppi.penMask & PEN_MASK_TILT_Y)
  pen_tilt.y = ppi.tiltY * M_2PI / 360;
```

```
      }
      break;
     }
     case PT_TOUCH:{
      POINTER_TOUCH_INFO pti;
      if(GetPointerTouchInfo && GetPointerTouchInfo(pointerId, &pti)) {
       // Add something touch specific here some day maybe...
      }
      break;
     }
/*     default:{
      POINTER_INFO pi;
      if(GetPointerInfo && GetPointerInfo(pointerId, &pi)) {
      }
      break;
     }
*/    }

    switch(message){
     case WM_POINTERDOWN:
      ClickActivateWnd();
      if(ignoreclick) return 0L;
      DoMouse(LEFTDOWN, Point(p), 0);
      if(_this) PostInput();
      break;
     case WM_POINTERUP:
      if(ignoreclick) EndIgnore();
      else DoMouse(LEFTUP, Point(p), 0);
      if(_this) PostInput();
      break;
     case WM_POINTERUPDATE:
      if(ignoreclick) {
       EndIgnore();
       return 0L;
      }
      if(_this) DoMouse(MOUSEMOVE, Point(p));
      DoCursorShape();
      break;

    }
    return 0L;
   }
  }
  break;
 case WM_POINTERLEAVE:
  pen = false;
  break;
...
```

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 12 Mar 2021 16:50:24 GMT
View Forum Message <> Reply to Message

OK, now I checked LeftDouble(), RightDouble(), RightUp()... and NONE of them works with the above code.

Need to think more...

Best regards,

Tom

EDIT: I'm starting to believe that if WM_POINTER* is used for mouse, it needs processing WM_POINTERUP and WM_POINTERDOWN in a way that checks IS_POINTER_FIRSTBUTTON_WPARAM(wParam), IS_POINTER_SECONDBUTTON_WPARAM(wParam) and so on to detect WM_LBUTTONUP/DOWN and WM_RBUTTONUP/DOWN and others correctly. Seems a bit complex to me. And still DOUBLECLICKs are a mystery.

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 12 Mar 2021 17:32:16 GMT
View Forum Message <> Reply to Message

Well, that is about what I expected.

The Pen virtual method seems more and more favorable solution.

Can you please test DDUMPHEX(GetMessageExtraInfo()); ? It would be really helpful in new API, which would use normal mouse, but put K_PEN into flags of Mouse messages... (because we WILL need to distinguish these).

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 12 Mar 2021 18:04:49 GMT
View Forum Message <> Reply to Message

Hi,

GetMessageExtraInfo() returned always zero... :(

(Reported this earlier.)

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 12 Mar 2021 19:27:02 GMT
View Forum Message <> Reply to Message

Mirek,

Can you test this:

```
 switch(message) {
 case WM_POINTERDOWN:
 case WM_POINTERUPDATE:
 case WM_POINTERUP:
  {

   POINT p = Point((LONG)lParam);
   ScreenToClient(hwnd, &p);

   pen = false;
   pen_pressure = pen_rotation = Null;
   pen_tilt = Null;
   pen_eraser = false;
   pen_barrel = false;
   pen_inverted = false;

   static BOOL (WINAPI *EnableMouseInPointer)(BOOL fEnable);
   static BOOL (WINAPI *GetPointerType)(UINT32 pointerId, POINTER_INPUT_TYPE
*pointerType);
   static BOOL (WINAPI *GetPointerInfo)(UINT32 pointerId, POINTER_INFO *pointerInfo);
   static BOOL (WINAPI *GetPointerPenInfo)(UINT32 pointerId, POINTER_PEN_INFO *penInfo);
   static BOOL (WINAPI *GetPointerTouchInfo)(UINT32 pointerId, POINTER_TOUCH_INFO
*touchInfo);
   static BOOL (WINAPI *GetPointerPenInfoHistory)(UINT32 pointerId, UINT32 *entriesCount,
POINTER_PEN_INFO *penInfo);

   ONCELOCK {
    DllFn(GetPointerType, "User32.dll", "GetPointerType");
    DllFn(GetPointerInfo, "User32.dll", "GetPointerInfo");
```

```
    DllFn(GetPointerPenInfo, "User32.dll", "GetPointerPenInfo");
    DllFn(GetPointerTouchInfo, "User32.dll", "GetPointerTouchInfo");

    DllFn(GetPointerPenInfoHistory, "User32.dll", "GetPointerPenInfoHistory");
   };

   POINTER_INPUT_TYPE pointerType;


   UINT32 pointerId = GET_POINTERID_WPARAM(wParam);
   if(GetPointerType && GetPointerType(pointerId, &pointerType)) {
    switch(pointerType){
     case PT_PEN:{

      UINT32 hc=256;
      POINTER_PEN_INFO ppit[hc];
      if(message==WM_POINTERUPDATE && GetPointerPenInfoHistory &&
GetPointerPenInfoHistory(pointerId, &hc, ppit)) {
       for(int i=hc-1;i>=0;i--){
        pen = true;
        if(ppit[i].penFlags & PEN_FLAG_BARREL)
         pen_barrel = true;
        if(ppit[i].penFlags & PEN_FLAG_INVERTED)
         pen_inverted = true;
        if(ppit[i].penFlags & PEN_FLAG_ERASER)
         pen_eraser = true;
        if(ppit[i].penMask & PEN_MASK_PRESSURE)
         pen_pressure = ppit[i].pressure / 1024.0;
        if(ppit[i].penMask & PEN_MASK_ROTATION)
         pen_rotation = ppit[i].rotation * M_2PI / 360;
        if(ppit[i].penMask & PEN_MASK_TILT_X)
         pen_tilt.x = ppit[i].tiltX * M_2PI / 360;
        if(ppit[i].penMask & PEN_MASK_TILT_Y)
         pen_tilt.y = ppit[i].tiltY * M_2PI / 360;

        POINT hp = ppit[i].pointerInfo.ptPixelLocation;
        ScreenToClient(hwnd, &hp);

        if(ignoreclick) {
         EndIgnore();
         return 0L;
        }
        if(_this) DoMouse(MOUSEMOVE, Point(hp));
        DoCursorShape();

       }
       return 0L;
      }
```

```
  POINTER_PEN_INFO ppi;
  if(GetPointerPenInfo && GetPointerPenInfo(pointerId, &ppi)) {
   pen = true;
   if(ppi.penFlags & PEN_FLAG_BARREL)
    pen_barrel = true;
   if(ppi.penFlags & PEN_FLAG_INVERTED)
    pen_inverted = true;
   if(ppi.penFlags & PEN_FLAG_ERASER)
    pen_eraser = true;
   if(ppi.penMask & PEN_MASK_PRESSURE)
    pen_pressure = ppi.pressure / 1024.0;
   if(ppi.penMask & PEN_MASK_ROTATION)
    pen_rotation = ppi.rotation * M_2PI / 360;
   if(ppi.penMask & PEN_MASK_TILT_X)
    pen_tilt.x = ppi.tiltX * M_2PI / 360;
   if(ppi.penMask & PEN_MASK_TILT_Y)
    pen_tilt.y = ppi.tiltY * M_2PI / 360;
  }
  break;
 }
 case PT_TOUCH:{
  POINTER_TOUCH_INFO pti;
  if(GetPointerTouchInfo && GetPointerTouchInfo(pointerId, &pti)) {
   // Add something touch specific here some day maybe...
  }
  break;
 }
}

switch(message){
 case WM_POINTERDOWN:
  ClickActivateWnd();
  if(ignoreclick) return 0L;
  DoMouse(LEFTDOWN, Point(p), 0);
  if(_this) PostInput();
  break;
 case WM_POINTERUP:
  if(ignoreclick) EndIgnore();
  else DoMouse(LEFTUP, Point(p), 0);
  if(_this) PostInput();
  break;
 case WM_POINTERUPDATE:
  if(ignoreclick) {
   EndIgnore();
   return 0L;
  }
  if(_this) DoMouse(MOUSEMOVE, Point(p));
```

```
      DoCursorShape();
      break;

    }
    return 0L;
   }
  }
  break;
 case WM_POINTERLEAVE:
  pen = false;
  break;
```

No blocking of original WM_MOUSEMOVE, etc. required at all.

For some reason, with this code I do not get those WM_MOUSEMOVE messages for Pen, but only for mouse. Similarly, WM_POINTER... messages only appear for Pen.

Right clicks and double clicks work for mouse as before. So this is closest to a working solution so far.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 13 Mar 2021 11:42:16 GMT
View Forum Message <> Reply to Message

Hi,

OK, now I found the reason why the original code produced both WM_MOUSEMOVEs and WM_POINTERUPDATEs for the pen moving: When case WM_POINTERUPDATE: exits with break; , it causes some emulation to result in WM_MOUSEMOVE. However, when WM_POINTERUPDATE exits with return 0L; , there will not be any WM_MOUSEMOVE and the two get processed separately and correctly.

This must be the reason why the latest code works.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 13 Mar 2021 13:19:13 GMT

View Forum Message <> Reply to Message

Looks good, however, should you return immediately after processing the history?

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 13 Mar 2021 13:39:32 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 13 March 2021 15:19Looks good, however, should you return immediately after processing the history?

I think so, yes. History[0] contains the latest POINTERUPDATE, so everything gets processed this way.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 13 Mar 2021 14:07:33 GMT
View Forum Message <> Reply to Message

OK, done some cleanup/deduplication, I think this might be final. Can you test?

And now important question: Does Wacom work with Linux? :)

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 13 Mar 2021 14:10:14 GMT
View Forum Message <> Reply to Message

... I'm just thinking about the situation, where an application written with U++ has not been specifically designed to support pen: I think that the barrel button should be mapped to the right mouse button in U++ -- as is the default Windows pen behavior, I think. This (and maybe some other things) should give a reasonable pen experience without any special work on the application itself.

The default pen behavior can be seen in:

https://docs.microsoft.com/en-us/previous-versions/windows/d esktop/inputmsg/wm-pointerdown

Look specifically at IS_POINTER_FIRSTBUTTON_WPARAM(wParam) and IS_POINTER_SECONDBUTTON_WPARAM(wParam)...

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 13 Mar 2021 14:20:55 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 13 March 2021 16:07OK, done some cleanup/deduplication, I think this might be final. Can you test?

And now important question: Does Wacom work with Linux? :)

Mirek

Yes, it works just great! Nice cleanup too.

You can remove this too:

static BOOL (WINAPI *EnableMouseInPointer)(BOOL fEnable);

Tracking pen_history is a mystery to me, though... Any reason for that?

Currently my linux is on the virtual machine on top of windows... It does not see pen operations directly. I need to check if I can reconnect the wacom directly to the VM...

Best regards,

Tom

EDIT: A quick search says it should work after installing some X wacom drivers...

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 13 Mar 2021 17:02:37 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 13 March 2021 15:20mirek wrote on Sat, 13 March 2021 16:07OK, done some cleanup/deduplication, I think this might be final. Can you test?

And now important question: Does Wacom work with Linux? :)

Mirek

Yes, it works just great! Nice cleanup too.

You can remove this too:

static BOOL (WINAPI *EnableMouseInPointer)(BOOL fEnable);

Tracking pen_history is a mystery to me, though... Any reason for that?

Well, the reason why it is in history is that your app is not fast enough to process events. So maybe the app should have a chance to know that, right ? (perhaps decide to ignore events that are from the history...)

Mirek

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 13 Mar 2021 18:18:19 GMT
View Forum Message <> Reply to Message

Implemented in Linux/gtk too now. Just bare minimum (pressure, maybe tilt and rotation - could not test), without history yet (which is actually quite similar in linux)...

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 13 Mar 2021 19:54:37 GMT
View Forum Message <> Reply to Message

Hi,

News: Wacom works on Linux Mint out of the box on real hardware. Right from login screen!! :)

On LM, the barrel button generally seems to work like the right mouse button.

State of U++ support for Wacom:

+ Pressure works
? Tilt with non-tilt-supporting hardware reports around 0.007 on both axis
- barrel button is not detected at all

Processing the history on linux seems even more important than with Windows. The drawing output is very coarse.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 13 Mar 2021 20:09:05 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 13 March 2021 19:02Tom1 wrote on Sat, 13 March 2021 15:20
Tracking pen_history is a mystery to me, though... Any reason for that?

Well, the reason why it is in history is that your app is not fast enough to process events. So maybe the app should have a chance to know that, right ? (perhaps decide to ignore events that are from the history...)

Mirek

OK, good point. That may prove useful in the long run...

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 13 Mar 2021 23:05:11 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 13 March 2021 20:54Hi,

News: Wacom works on Linux Mint out of the box on real hardware. Right from login screen!! :)

On LM, the barrel button generally seems to work like the right mouse button.

State of U++ support for Wacom:

+ Pressure works
? Tilt with non-tilt-supporting hardware reports around 0.007 on both axis
- barrel button is not detected at all

Processing the history on linux seems even more important than with Windows. The drawing output is very coarse.

Best regards,

Tom

I can handle the history, maybe disable unsupported tilt, no so sure about barrel...

Subject: Re: Using Pen with U++
Posted by Tom1 on Sun, 14 Mar 2021 11:14:21 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 14 March 2021 01:05
I can handle the history, maybe disable unsupported tilt, no so sure about barrel...

About the barrel on Linux: Now with wacom it goes directly to right mouse button (with pen flag active though). I guess for uniform behavior the two options are now:

1. To catch in Linux RightDown() and RightUp() with IsPointerPen() and set the barrel flag accordingly.

2. Or to catch barrel flag in Windows WM_POINTERDOWN / WM_POINTERUP and call RightDown() / RightUp() accordingly on state changes with correctly set up IsPointerPen() result.

I do not know, which one would be more logical and better in line with general pen experience expectations. Or should we do both?

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 17 Mar 2021 08:59:17 GMT
View Forum Message <> Reply to Message

Hi Mirek,

EDIT: Removed earlier content...

OK, as it turns out, so far I tested with the common testcase drawing stuff on the surface. However, after starting to adapt the pen to the actual application, I found that e.g. Button clicks and double clicks do not work with pen. I have now tried the whole day to figure out what's the problem, but it seems Windows really wants to make this difficult for me.

The point is that in order to have a usable pen experience on the desktop, Windows wants to process all the WM_POINTER* messages using DefWindowProc*() and generate WM_LBUTTONDOWN,WM_LBUTTONUP,WM_MOUSEMOVE,... etc. messages. This is also a prerequisite for WM_LBUTTONDBLCLK to be generated at all.

So, now I'm starting to think that it might be easier to arrange something special to extract the full pen trace while drawing with pen and have the ordinary mouse emulation handled by Windows with DefWindowProc*().

I have nothing that works yet towards this target, but I will still try to figure out something...

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 19 Mar 2021 08:09:43 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Here's an intermediate finding. While working on pen support I accidentally found this code to fix in CtrlMouse.cpp:

```
 if(e == LEFTUP)
  leftmousepos = Null;
 if(e == RIGHTUP)
  rightmousepos = Null;
 if(e == MIDDLEUP)
  rightmousepos = Null;
 if(findarg(e, LEFTUP, RIGHTUP, MIDDLEUP) >= 0)
```

Should be:

```
 if(e == LEFTUP)
  leftmousepos = Null;
 if(e == RIGHTUP)
  rightmousepos = Null;
 if(e == MIDDLEUP)
  middlemousepos = Null;
 if(findarg(e, LEFTUP, RIGHTUP, MIDDLEUP) >= 0)
```

As for the pen, I'm still trying to work around Windows induced issues...

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 19 Mar 2021 11:57:47 GMT
View Forum Message <> Reply to Message

Thanks, applied.

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 19 Mar 2021 15:02:57 GMT
View Forum Message <> Reply to Message

Hi Mirek,

OK, now I'm nearly there with Windows. What's missing is that I cannot get RectTracker to work with Pen. I would really appreciate, if you could take a look at that. There's a new testcase with on-screen instructions below to test the central functions of pen and mouse...

In order to keep everything else working with mouse and a pen emulating a mouse, I have introduced a new flag and function for enabling accurate pen support for Ctrls requiring fine drawing. The reason I had to do this is that I could not retain usable mouse emulation when processing WM_POINTER* messages for fine drawing. (As an example, the buttons would not work with WM_POINTER* generated LEFTDOWN/UPs.)

Anyway, here it is.

Add to "Ctrl{" in CtrlCore.h:

```
 bool   pen_supported;

 void         EnablePenSupport(bool flag=true)  { pen_supported=flag; }
```

Add to "Ctrl::Ctrl() {" in Ctrl.cpp:

```
 pen_supported = false;
```

Changes in "WindowProc()" in Win32Proc.cpp:

```
...
LRESULT Ctrl::WindowProc(UINT message, WPARAM wParam, LPARAM lParam) {
 GuiLock __;
 eventid++;
// LLOG("Ctrl::WindowProc(" << message << ") in " << ::Name(this) << ", focus " << (void
*)::GetFocus());
 Ptr<Ctrl> _this = this;
 HWND hwnd = GetHWND();

 auto DoMouseMove = [&](POINT p) {
  if(ignoreclick)
   EndIgnore();
  else {
   if(_this) DoMouse(MOUSEMOVE, Point(p));
   DoCursorShape();
```

```
 }
};

 static bool left_down=false;
 static bool right_down=false;
 static Point pendownpos=Null;

 switch(message) {
 case WM_POINTERDOWN:
 case WM_POINTERUP:
 case WM_POINTERUPDATE:{
  if(!pen_supported) break; // Go with mouse emulation (default processing) instead

  POINT p = Point((LONG)lParam);
  ScreenToClient(hwnd, &p);

  pen = false;
  pen_pressure = pen_rotation = Null;
  pen_tilt = Null;
  pen_eraser = false;
  pen_barrel = false;
  pen_inverted = false;
  pen_history = false;

  static BOOL (WINAPI *GetPointerType)(UINT32 pointerId, POINTER_INPUT_TYPE
*pointerType);
  static BOOL (WINAPI *GetPointerInfo)(UINT32 pointerId, POINTER_INFO *pointerInfo);
  static BOOL (WINAPI *GetPointerPenInfo)(UINT32 pointerId, POINTER_PEN_INFO *penInfo);
  static BOOL (WINAPI *GetPointerPenInfoHistory)(UINT32 pointerId, UINT32 *entriesCount,
POINTER_PEN_INFO *penInfo);

  ONCELOCK {
   DllFn(GetPointerType, "User32.dll", "GetPointerType");
   DllFn(GetPointerInfo, "User32.dll", "GetPointerInfo");
   DllFn(GetPointerPenInfo, "User32.dll", "GetPointerPenInfo");
   DllFn(GetPointerPenInfoHistory, "User32.dll", "GetPointerPenInfoHistory");
  };

  if(!(GetPointerType && GetPointerInfo && GetPointerPenInfo && GetPointerPenInfoHistory))
   break;

  POINTER_INPUT_TYPE pointerType;

  auto ProcessPenInfo = [&] (POINTER_PEN_INFO& ppi) {
   pen = true;
   if(ppi.penFlags & PEN_FLAG_BARREL)
    pen_barrel = true;
   if(ppi.penFlags & PEN_FLAG_INVERTED)
```

```
     pen_inverted = true;
    if(ppi.penFlags & PEN_FLAG_ERASER)
     pen_eraser = true;
    if(ppi.penMask & PEN_MASK_PRESSURE)
     pen_pressure = ppi.pressure / 1024.0;
    if(ppi.penMask & PEN_MASK_ROTATION)
     pen_rotation = ppi.rotation * M_2PI / 360;
    if(ppi.penMask & PEN_MASK_TILT_X)
     pen_tilt.x = ppi.tiltX * M_2PI / 360;
    if(ppi.penMask & PEN_MASK_TILT_Y)
     pen_tilt.y = ppi.tiltY * M_2PI / 360;
   };

   UINT32 pointerId = GET_POINTERID_WPARAM(wParam);
   if(GetPointerType(pointerId, &pointerType) && pointerType == PT_PEN) {
    UINT32 hc = 256;
    Buffer<POINTER_PEN_INFO> ppit(hc);
    if(message == WM_POINTERUPDATE && (right_down || left_down) &&
GetPointerPenInfoHistory(pointerId, &hc, ppit)) {
     for(int i = hc - 1; i >= 0; i--) {
      ProcessPenInfo(ppit[i]);
      POINT hp = ppit[i].pointerInfo.ptPixelLocation;
      ScreenToClient(hwnd, &hp);
      pen_history = (bool)i;
      DoMouseMove(hp);
     }
     pen_history = false;
    }
    else{
     POINTER_PEN_INFO ppi;
     if(GetPointerPenInfo(pointerId, &ppi)){
      ProcessPenInfo(ppi);
      switch(message) {
       case WM_POINTERDOWN:
        ClickActivateWnd();
        if(ignoreclick) return 0L;
        right_down=pen_barrel;
        left_down=!right_down;
        if(right_down) DoMouse(RIGHTDOWN, pendownpos=Point(p));
        else DoMouse(LEFTDOWN, pendownpos=Point(p), 0);
        if(_this) PostInput();
        break;
      }
     }
    }
   }
  }
  break;
```

```
 case WM_POINTERLEAVE:
  pen = false;
  right_down=false;
  left_down=false;
  break;

...

 case WM_LBUTTONDOWN:
  ClickActivateWnd();
  if(ignoreclick) return 0L;
  if(pendownpos==Point((dword)lParam)) { pendownpos=Null; return 0L; }
  else{
   DoMouse(LEFTDOWN, Point((dword)lParam));
   if(_this) PostInput();
  }
  return 0L;
 case WM_LBUTTONUP:
  if(ignoreclick)
   EndIgnore();
  else{
   if(right_down) DoMouse(RIGHTUP, Point((dword)lParam));
   else DoMouse(LEFTUP, Point((dword)lParam));
  }
  if(_this) PostInput();
  right_down=false;
  left_down=false;
  pendownpos=Null;
  return 0L;
 case WM_LBUTTONDBLCLK:
  ClickActivateWnd();
  if(ignoreclick) return 0L;
  DoMouse(LEFTDOUBLE, Point((dword)lParam));
  if(_this) PostInput();
  return 0L;
 case WM_RBUTTONDOWN:
  if(pendownpos==Point((dword)lParam)) { pendownpos=Null; return 0L; }
  ClickActivateWnd();
  if(ignoreclick) return 0L;
  DoMouse(RIGHTDOWN, Point((dword)lParam));
  if(_this) PostInput();
  return 0L;
 case WM_RBUTTONUP:
  if(ignoreclick)
   EndIgnore();
  else
   DoMouse(RIGHTUP, Point((dword)lParam));
  if(_this) PostInput();
```

```
  right_down=false;
  left_down=false;
  pendownpos=Null;
  return 0L;

...

 case WM_MOUSEMOVE:
  if(left_down || right_down) return 0L;
  LLOG("WM_MOUSEMOVE: ignoreclick = " << ignoreclick);
  DoMouseMove(Point((dword)lParam));
  return 0L;

...
```

And finally the extended testcase:

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

using namespace Upp;

#define LLOG(x)  DLOG(x)

struct MyApp : TopWindow {
 Point pos;

 Point lDouble;
 Point rDouble;
 Point lHold;
 Point rHold;

 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;
 lDouble=Null;
 rDouble=Null;
 lHold=Null;
 rHold=Null;

 lDoublePen=false;
 rDoublePen=false;
 lHoldPen=false;
 rHoldPen=false;

 trect=Null;
 p1=p2=Null;

 EnablePenSupport();
}

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

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

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

virtual void RightUp(Point p, dword){
 LLOG((IsPointerPen()?"PenRightUp":"MouseRightUp"));
 rdown=false;
 Refresh();
}

virtual void LeftHold(Point p, dword){
 LLOG((IsPointerPen()?"PenLeftHold":"MouseLeftHold"));
```

```
 lHold=p;
 lHoldPen=IsPointerPen();
 Refresh();
}

virtual void LeftDouble(Point p, dword){
 LLOG((IsPointerPen()?"PenLeftDouble":"MouseLeftDouble"));
 lDouble=p;
 lDoublePen=IsPointerPen();
 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 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 and DoubleClicks using Mouse with both Left and Right
buttons.");
  Text("2. Test Clicks, Drags and DoubleClicks 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());
 }
};

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

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 20 Mar 2021 10:25:31 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Here are some further findings for the code I posted above. If intermediate MouseMove's are ignored, the following U++ callbacks result for each mouse operation:

CLICK:

LeftDown
LeftUp

DOUBLECLICK:

LeftDown

LeftUp
LeftDouble
LeftUp

TRIPLECLICK:

LeftDown
LeftUp
LeftDouble
LeftUp
LeftTriple
LeftUp

DRAG:

LeftDown
LeftDrag
LeftUp

HOLD:

LeftDown
LeftHold
LeftUp

HOLD+DRAG:

LeftDown
LeftHold
LeftDrag
LeftUp
For Pen we currently get:
CLICK:

LeftDown
LeftUp

DOUBLECLICK:

LeftDown
LeftUp
LeftDown ** Extra
LeftDouble
LeftUp

TRIPLECLICK:

LeftDown

LeftUp
LeftDown ** Extra
LeftDouble
LeftUp
LeftTriple
LeftUp

DRAG:

LeftDown
LeftDrag
LeftUp

HOLD:

LeftDown
!! Missing: LeftHold
LeftRightUp ** !!!

(Hold missing and RightUp emitted instead of LeftUp call!)

HOLD+DRAG:

LeftDown
!! Missing: LeftHold
LeftDrag
LeftUp

The weird Hold behavior is likely caused by Windows Ink internals.

Anyway, in my opinion we should try to get exact the same callbacks from pen as we get from the mouse.

Best regards,

Tom

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

Subject: Re: Using Pen with U++
Posted by mirek on Sun, 21 Mar 2021 23:39:48 GMT
View Forum Message <> Reply to Message

It is getting complicated to carefuly apply all changes to WindowProc; could you please send the

whole file?

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 22 Mar 2021 07:46:05 GMT
View Forum Message <> Reply to Message

Hi,

Sure, please find attached Win32Proc.cpp.

I wish I could have achieved a better solution. I.e. one with:

- correct callback sequences (same as mouse)
- buttons (and what not) working without flag steered bypass
- working RectTracker

Anyway, Windows beats me on this one.

Best regards,

Tom

EDIT: Updated the Win32Proc.cpp. (Managed to improve callback sequences a little bit. The rest is as is...)

File Attachments
1) Win32Proc.cpp, downloaded 153 times

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 22 Mar 2021 09:35:04 GMT
View Forum Message <> Reply to Message

Hi,

Here's an improved testcase code for easier callback monitoring:

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

using namespace Upp;

#define LLOG(x)  DLOG(x)

bool moved=false;

class PointerEvent{
```

```
public:
 Point p;
 int  row;
 String label;

 PointerEvent(Point p_, String label_, int row_){
  if(moved){
   moved=false;
   LLOG("...");
  }
  p=p_;
  label=label_;
  row=row_;
  LLOG(label << ": " << p);
 }
};

struct MyApp : TopWindow {
 Array<PointerEvent> elist;

 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;

  moved=false;

  trect=Null;
  p1=p2=Null;

  EnablePenSupport();
 }

 virtual void RightDown(Point p, dword){
  elist.Insert(0,PointerEvent(p,IsPointerPen()?"PenRightDown":"MouseRightDown",0));
  rdown=true;
  rdrawing.Add().Add(MakeTuple(IsPointerPen()?GetPenPressure():0.1, p));
  Refresh();
 }
```

```
virtual void RightHold(Point p, dword){
 elist.Insert(0,PointerEvent(p,IsPointerPen()?"PenRightHold":"MouseRightHold",1));
 Refresh();
}

virtual void RightDrag(Point p, dword){
 elist.Insert(0,PointerEvent(p,IsPointerPen()?"PenRightDrag":"MouseRightDrag",-1));
 Refresh();
}

virtual void RightDouble(Point p, dword){
 elist.Insert(0,PointerEvent(p,IsPointerPen()?"PenRightDouble":"MouseRightDouble",-2));
 Refresh();
}

virtual void RightTriple(Point p, dword){
 elist.Insert(0,PointerEvent(p,IsPointerPen()?"PenRightTriple":"MouseRightTriple",-3));
 Refresh();
}

virtual void RightUp(Point p, dword){
 elist.Insert(0,PointerEvent(p,IsPointerPen()?"PenRightUp":"MouseRightUp",-1));
 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{
  elist.Insert(0,PointerEvent(p,IsPointerPen()?"PenLeftDown":"MouseLeftDown",0));
  ldown=true;
  drawing.Add().Add(MakeTuple(IsPointerPen()?GetPenPressure():0.1, p));
 }
 Refresh();
}

virtual void LeftHold(Point p, dword){
 elist.Insert(0,PointerEvent(p,IsPointerPen()?"PenLeftHold":"MouseLeftHold",1));
```

```
 Refresh();
}

virtual void LeftDrag(Point p, dword){
 elist.Insert(0,PointerEvent(p,IsPointerPen()?"PenLeftDrag":"MouseLeftDrag",-1));
 Refresh();
}

virtual void LeftDouble(Point p, dword){
 elist.Insert(0,PointerEvent(p,IsPointerPen()?"PenLeftDouble":"MouseLeftDouble",-2));
 Refresh();
}

virtual void LeftTriple(Point p, dword){
 elist.Insert(0,PointerEvent(p,IsPointerPen()?"PenLeftTriple":"MouseLeftTriple",-3));
 Refresh();
}

virtual void LeftUp(Point p, dword){
 elist.Insert(0,PointerEvent(p,IsPointerPen()?"PenLeftUp":"MouseLeftUp",-1));
 ldown=false;
 Refresh();
}

Vector<String> report;

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

 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(p);
 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]);

  elist.Trim(min(15,elist.GetCount()));
  for(int i=0;i<elist.GetCount();i++) w.DrawText(elist[i].p.x, elist[i].p.y + elist[i].row*GetStdFontCy(),
elist[i].label);
 }
};
```

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

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 22 Mar 2021 15:17:09 GMT
View Forum Message <> Reply to Message

Hi Mirek,

It seems GetMousePos() is not working unless WM_MOUSEMOVE can call DoMouseMove(). It does not help that DoMouseMove() has already been called for WM_POINTERUPDATE at the same coordinates just previously.

Many things are based on GetMousePos() so it has to work. However, it is based on ::GetCursorPos() (on WIN32 API). How to fix this?

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Tue, 23 Mar 2021 12:36:56 GMT
View Forum Message <> Reply to Message

Hi,

Here's the current code that now fixes RectTracker and GetMousePos() with Pen:

Add to "Ctrl{" in CtrlCore.h:

```
 bool   pen_supported;

 void         EnablePenSupport(bool flag=true)  { pen_supported=flag; }
```

Add to "Ctrl::Ctrl() {" in Ctrl.cpp:

```
 pen_supported = false;
```

Changes around Point GetMousePos() in Win32Wnd.cpp:

```
static Point screen_mouse_pos=Null;

void SetMousePos(const Point &p) {
 screen_mouse_pos=p;
}

Point GetMousePos() {
 return screen_mouse_pos;
// Point p;
// return ::GetCursorPos(p) ? p : Null;
// ::GetCursorPos(p);
// return p;
}
```

The rest of the changes are in the attached Win32Proc.cpp.

The last issue (I know of) requiring switchable 'EnablePenSupport()' is the malfunction of buttons.

Best regards,

Tom

File Attachments

1) Win32Proc.cpp, downloaded 118 times

Subject: Re: Using Pen with U++
Posted by mirek on Wed, 24 Mar 2021 10:35:51 GMT
View Forum Message <> Reply to Message

Well, I do not quite like this, it becomes too clumsy.

I am starting to think that we rather need a new virtual method for pen (as result of POINTERUPDATE) that would return true if it would want to avoid further processing of event.

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 24 Mar 2021 10:57:13 GMT
View Forum Message <> Reply to Message

Hi,

Actually, I went that way too, but had to return back to this. The reason is that we need to be able to use all standard Ctrls with Pen equally to Mouse. Otherwise, it becomes tedious to build pen interfaces for each and every standard Ctrl.

Or, do you have something very smart in mind already? (Many times you have, so I just have to ask... :)

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Wed, 24 Mar 2021 12:31:37 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 24 March 2021 11:57Hi,

Actually, I went that way too, but had to return back to this. The reason is that we need to be able to use all standard Ctrls with Pen equally to Mouse. Otherwise, it becomes tedious to build pen interfaces for each and every standard Ctrl.

Or, do you have something very smart in mind already? (Many times you have, so I just have to ask... :)

Best regards,

Tom

You missed 'bool' return parameter... All standard Ctrls would return false, meaning it should be convert to good old WM messages by windows.

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 24 Mar 2021 14:21:09 GMT
View Forum Message <> Reply to Message

OK, I see. I'll give it a try... (will return soon.)

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Wed, 24 Mar 2021 14:56:49 GMT

Tom1 wrote on Wed, 24 March 2021 15:21OK, I see. I'll give it a try... (will return soon.)

Best regards,

Tom

Well, I thought I will do that tonight or tomorrow, but if you want to try, there are some tips and tricks how I planned to do it:

- add another kind of event PEN 0xf0 after TRIPLE constant
- will need to add handling in DoMouse around CtrlCore/CtrlMouse.cpp:149
- we will need to somehow return that bool; DoMouse returns Image, here I planned to use a trick to return Null or Non-null image to signal true-falase (use any image available)
- as for Pen signature, I was thinking about creating some struct type with all the information (pressure, tilt etc) that would be passed as parameter. Perhaps leave Point as separate parameter. So something like

virtual bool Pen(Point p, const PenInfo& f);

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 24 Mar 2021 15:51:09 GMT

Mirek,

I was only thinking about declaring:

```
 virtual bool   PenDown(Point p, dword keyflags){ return false; }
 virtual bool   PenUp(Point p, dword keyflags){ return false; }
 virtual bool   PenMove(Point p, dword keyflags){ return false; }
```

and calling them directly like this

```
    switch(message) {
    case WM_POINTERDOWN:
     ClickActivateWnd();
     if(PenDown(Point(p), GetMouseFlags())) return 0L;
     break;
    case WM_POINTERUP:
     if(PenUp(Point(p), GetMouseFlags())) return 0L;
     break;
    case WM_POINTERUPDATE:
     if(PenMove(Point(p), GetMouseFlags())) return 0L;
     break;
    }
```

break;

...and using the Pen -functions you made earlier.

However, my simple approach above would inevitably have its shortcomings as properly tracking the pen/mouse position (e.g. GetMousePos()) cannot be taken care of this way. RectTracker would likely also be broken in this case.

Despite working several tens of hours on this, my understanding on Ctrl and mouse processing is not sufficient yet to do it the right way. (Despite the several tens of hours I have spent on this during the few last days. Shame on me.)

So, absolutely, if you have time, please do it the right way. (I have a strong feeling that it will eventually save time for both of us.)

In any case, I'll be here ready to test as soon as you post an update.

Best regards,

Tom

EDIT: BTW: How did you think about handling barrel-button and other flags? After playing around with the pen, I have noticed that it is very difficult to click barrel and other buttons on the pen while keeping it steady. So, IMO, barrel should be just a flag to read when pen goes down/up or moves. Just like Shift/Ctrl/Alt, which are used and needed as modifiers similarly for pen and mouse.

Also, I feel PenMove + PenDown + PenUp make it easier to write code for the pen, instead of a single PenUpdate function.

Further on, readily supporting PenHold, PenDouble (and possibly PenTriple) would be useful. Decoding them in application would require passing timestamps too, which is not very convenient.

EDIT2: Actually, I would rather have PEN defined as 3 or 4, i.e. in parallel to LEFT, RIGHT and MIDDLE. This way we can have PEN|DOWN, PEN|UP, PEN|DOUBLE, ... and just PEN for move.

Subject: Re: Using Pen with U++
Posted by mirek on Wed, 24 Mar 2021 17:01:06 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 24 March 2021 16:51Mirek,

I was only thinking about declaring:

```
virtual bool   PenDown(Point p, dword keyflags){ return false; }
virtual bool   PenUp(Point p, dword keyflags){ return false; }
virtual bool   PenMove(Point p, dword keyflags){ return false; }
```

and calling them directly like this

```
switch(message) {
case WM_POINTERDOWN:
 ClickActivateWnd();
 if(PenDown(Point(p), GetMouseFlags())) return 0L;
 break;
case WM_POINTERUP:
 if(PenUp(Point(p), GetMouseFlags())) return 0L;
 break;
case WM_POINTERUPDATE:
 if(PenMove(Point(p), GetMouseFlags())) return 0L;
 break;
}
break;
```

...and using the Pen -functions you made earlier.

This would not work - only top level windows are receiving messages from Win32. DoMouse then distributes them to target widgets (with a ton of logic and coordinate translations involved).

Quote:
EDIT: BTW: How did you think about handling barrel-button and other flags? After playing around with the pen, I have noticed that it is very difficult to click barrel and other buttons on the pen while keeping it steady. So, IMO, barrel should be just a flag to read when pen goes down/up or moves. Just like Shift/Ctrl/Alt, which are used and needed as modifiers similarly for pen and mouse.

As flag in PenInfo. That is the reason I vote for structure passed so that it can be extended.

Quote:
Also, I feel PenMove + PenDown + PenUp make it easier to write code for the pen, instead of a single PenUpdate function.

IDK. I feel like pen actually would be better handled with single method. I guess I have to try.

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 24 Mar 2021 17:22:07 GMT
View Forum Message <> Reply to Message

OK. PenInfo structure sounds good.

Please include 'dword keyflags' in call parameters or PenInfo.

Also, if you end up with a single method like 'PenUpdate()' or whatever, please also include

reason = MOVE/DOWN/UP/HOLD/DOUBLE/... in call parameters or PenInfo.

Thanks and best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Thu, 25 Mar 2021 18:58:21 GMT
View Forum Message <> Reply to Message

First attempt commited.

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

using namespace Upp;

struct MyApp : TopWindow {
 Point pos;

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

 PenInfo pen;

 virtual bool Pen(Point p, const PenInfo& pn, dword keyflags) {
  if(pn.pressure) {
   if((!!pn.pressure == !!pen.pressure) && drawing.GetCount())
    drawing.Top().Add(MakeTuple(pn.pressure, p));
   else
    drawing.Add().Add(MakeTuple(pn.pressure, p));
  }
  pen = pn;
  Refresh();
  return true;
 }

 virtual void Paint(Draw& w0) {
  DrawPainter w(w0, GetSize());
  w.Co();
  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, Black());
   }

  int fcy = GetStdFontCy();
  int y = 10;
  auto Text = [&] (const String& text) {
   w.DrawText(10, y, text);
   y += fcy;
  };
  Text(AsString(pos));
  Text(String() << "Pressure: " << pen.pressure);
  Text(String() << "Rotation: " << pen.rotation);
  Text(String() << "Tilt: " << pen.tilt);
  Text(String() << "Barrel: " << pen.barrel);
  Text(String() << "Inverted: " << pen.inverted);
  Text(String() << "Eraser: " << pen.eraser);
 }
};

GUI_APP_MAIN
{
 MyApp().Run();
}
```

Subject: Re: Using Pen with U++
Posted by Tom1 on Thu, 25 Mar 2021 19:47:20 GMT
View Forum Message <> Reply to Message

Hi Mirek,

And thanks for the first round on the new Pen interface. :)

The standard controls seem to work correctly (as expected) and the new interface is nice and clean.

However, the RectTracker needs attention. Please add the following code to the test case and see what happens with Ctrl+Pen and Shift+Pen:

```
 virtual bool Pen(Point p, const PenInfo& pn, dword keyflags) {
  if(pn.action==PEN_DOWN){
   if(keyflags&K_SHIFT){
    RectTracker tracker(*this);
    tracker.sync= [=](Rect r) {  };
    tracker.MinSize(Size(-100000,-100000));
    tracker.Track(Rect(p,p));
   }
```

```
  else if(keyflags&K_CTRL){
   RectTracker tracker(*this);
   tracker.TrackLine(p.x,p.y);
  }
 }

 if(pn.pressure) {
  if((!!pn.pressure == !!pen.pressure) && drawing.GetCount())
   drawing.Top().Add(MakeTuple(pn.pressure, p));
  else
   drawing.Add().Add(MakeTuple(pn.pressure, p));
 }
 pen = pn;
 Refresh();
 return true;
}
```

Thanks and best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Thu, 25 Mar 2021 21:07:43 GMT
View Forum Message <> Reply to Message

Well, I will try to make it work, however the indended road was something like

```
virtual bool Pen(Point p, const PenInfo& pn, dword keyflags) {
 if(keyflags&K_SHIFT)
  return false;
 if(pn.pressure) {
  if((!!pn.pressure == !!pen.pressure) && drawing.GetCount())
   drawing.Top().Add(MakeTuple(pn.pressure, p));
  else
   drawing.Add().Add(MakeTuple(pn.pressure, p));
 }
 pen = pn;
 Refresh();
 return true;
}

void LeftDown(Point p, dword keyflags) override {
 if(keyflags & K_SHIFT) {
  RectTracker tracker(*this);
  tracker.MinSize(Size(-100000,-100000));
```

```
  tracker.Track(Rect(p,p));
 }
}
```

Subject: Re: Using Pen with U++
Posted by Tom1 on Thu, 25 Mar 2021 21:28:15 GMT
View Forum Message <> Reply to Message

Mirek,

Is pn.action intentionally merely a state flag showing continuously PEN_UP or PEN_DOWN? Or should we reset it along with other PenInfo items in the beginning of processing WM_POINTER* messages?

I would like to have it reset, in order to more easily detect UP/DOWN state changes without a state variable and treat zero action as 'PEN_MOVE'.

Then we could do this to decode all kinds of pen events:

```
#define PENMOVE  0
#define PENDOWN  1
#define PENUP  2
#define PENHOLD  3
#define PENDRAG  4
#define PENDOUBLE 5
#define PENTRIPLE 6

bool IsNear(Point p, Point o){
 Point delta(p-o);
 if(abs(delta.x)<5 && abs(delta.y)<5) return true;
 return false;
}

int PenDecoder(Point p, const PenInfo& pn, dword keyflags){
 static int action=0;
 static int downt=0;
 static Point downp=Null;
 static bool doubleClick=true;
 static bool down=false;
 static bool solved=false;
 int now=msecs();
 int deltat=now-downt;


 bool isNear=IsNear(p,downp);
```

```
 switch(pn.action){
  case PEN_DOWN:
   down=true;
   solved=false;
   downp=p;
   downt=now;
   if(isNear){
    if(now-downt < 500){
     if(doubleClick){
      return PENTRIPLE;
     }
     else{
      doubleClick=true;
      return PENDOUBLE;
     }
    }
   }
   doubleClick=false;
   return PENDOWN;

  case PEN_UP:
   down=false;
   return PENUP;

  default:
   if(down && !solved){
    if(isNear){
     if(now-downt > 1000){
      solved=true;
      return PENHOLD;
     }
    }
    else{
     solved=true;
     return PENDRAG;
    }
   }
   return PENMOVE;
 }
}
```

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Thu, 25 Mar 2021 21:33:39 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 25 March 2021 23:07Well, I will try to make it work, however the indended road was something like

```
virtual bool Pen(Point p, const PenInfo& pn, dword keyflags) {
 if(keyflags&K_SHIFT)
  return false;
 if(pn.pressure) {
  if((!!pn.pressure == !!pen.pressure) && drawing.GetCount())
   drawing.Top().Add(MakeTuple(pn.pressure, p));
  else
   drawing.Add().Add(MakeTuple(pn.pressure, p));
 }
 pen = pn;
 Refresh();
 return true;
}

void LeftDown(Point p, dword keyflags) override {
 if(keyflags & K_SHIFT) {
  RectTracker tracker(*this);
  tracker.MinSize(Size(-100000,-100000));
  tracker.Track(Rect(p,p));
 }
}
```

Previous message content cleared.

Issue solved: I will return with a working RectTracker solution in the next message.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Thu, 25 Mar 2021 23:22:29 GMT
View Forum Message <> Reply to Message

OK, here are all the RectTracker changes required:

Add in RectTracker:: in CtrlCore.h:

```
virtual bool  Pen(Point p, const PenInfo& pn, dword keyflags);
```

Add in LocalLoop.cpp:

```
bool RectTracker::Pen(Point p, const PenInfo& pn, dword keyflags){
 switch(pn.action){
  default:
   MouseMove(p, keyflags);
   break;
  case PEN_UP:
   LeftUp(p, keyflags);
   break;
 }
 return true;
}
```

And finally changes in Win32Proc.cpp:

```
...

static bool pendown=false;

bool GetShift()       { return !!(GetKeyStateSafe(VK_SHIFT) & 0x8000); }
bool GetCtrl()        { return !!(GetKeyStateSafe(VK_CONTROL) & 0x8000); }
bool GetAlt()         { return !!(GetKeyStateSafe(VK_MENU) & 0x8000); }
bool GetCapsLock()    { return !!(GetKeyStateSafe(VK_CAPITAL) & 1); }
bool GetMouseLeft()   { return pendown || !!(GetKeyStateSafe(VK_LBUTTON) & 0x8000); }
bool GetMouseRight()  { return !!(GetKeyStateSafe(VK_RBUTTON) & 0x8000); }
bool GetMouseMiddle() { return !!(GetKeyStateSafe(VK_MBUTTON) & 0x8000); }

bool PassWindowsKey(int wParam);

LRESULT Ctrl::WindowProc(UINT message, WPARAM wParam, LPARAM lParam) {
 GuiLock __;
 eventid++;
// LLOG("Ctrl::WindowProc(" << message << ") in " << ::Name(this) << ", focus " << (void
*)::GetFocus());
 Ptr<Ctrl> _this = this;
 HWND hwnd = GetHWND();

 switch(message) {
 case WM_POINTERDOWN:
 case WM_POINTERUPDATE:
 case WM_POINTERUP: {
   pen.action = 0;
   pen.pressure = pen.rotation = Null;
   pen.tilt = Null;
   pen.eraser = pen.barrel = pen.inverted = pen.history = false;

   static BOOL (WINAPI *GetPointerType)(UINT32 pointerId, POINTER_INPUT_TYPE
```

```
*pointerType);
  static BOOL (WINAPI *GetPointerInfo)(UINT32 pointerId, POINTER_INFO *pointerInfo);
  static BOOL (WINAPI *GetPointerPenInfo)(UINT32 pointerId, POINTER_PEN_INFO *penInfo);
  static BOOL (WINAPI *GetPointerPenInfoHistory)(UINT32 pointerId, UINT32 *entriesCount,
POINTER_PEN_INFO *penInfo);

  ONCELOCK {
   DllFn(GetPointerType, "User32.dll", "GetPointerType");
   DllFn(GetPointerInfo, "User32.dll", "GetPointerInfo");
   DllFn(GetPointerPenInfo, "User32.dll", "GetPointerPenInfo");
   DllFn(GetPointerPenInfoHistory, "User32.dll", "GetPointerPenInfoHistory");
  };

  if(!(GetPointerType && GetPointerInfo && GetPointerPenInfo && GetPointerPenInfoHistory))
   break;

  POINTER_INPUT_TYPE pointerType;

  auto ProcessPenInfo = [&] (POINTER_PEN_INFO& ppi) {
   if(ppi.penFlags & PEN_FLAG_BARREL)
    pen.barrel = true;
   if(ppi.penFlags & PEN_FLAG_INVERTED)
    pen.inverted = true;
   if(ppi.penFlags & PEN_FLAG_ERASER)
    pen.eraser = true;
   if(ppi.penMask & PEN_MASK_PRESSURE)
    pen.pressure = ppi.pressure / 1024.0;
   if(ppi.penMask & PEN_MASK_ROTATION)
    pen.rotation = ppi.rotation * M_2PI / 360;
   if(ppi.penMask & PEN_MASK_TILT_X)
    pen.tilt.x = ppi.tiltX * M_2PI / 360;
   if(ppi.penMask & PEN_MASK_TILT_Y)
    pen.tilt.y = ppi.tiltY * M_2PI / 360;
  };

  UINT32 pointerId = GET_POINTERID_WPARAM(wParam);
  if(GetPointerType(pointerId, &pointerType) && pointerType == PT_PEN) {
   UINT32 hc = 256;
   Buffer<POINTER_PEN_INFO> ppit(hc);
   if(message == WM_POINTERUPDATE && GetPointerPenInfoHistory(pointerId, &hc, ppit)) {
    bool processed = false;
    for(int i = hc - 1; i >= 0; i--) {
     ProcessPenInfo(ppit[i]);
     POINT hp = ppit[i].pointerInfo.ptPixelLocation;
     ::SetCursorPos(hp.x,hp.y);
     ::ScreenToClient(hwnd, &hp);
     pen.history = (bool)i;
     processed = !IsNull(DoMouse(PEN, hp, 0));
```

```
     }
     if(processed)
      return 0L;
     else
      break;
    }
    POINTER_PEN_INFO ppi;
    if(GetPointerPenInfo(pointerId, &ppi))
     ProcessPenInfo(ppi);

    POINT p = ppi.pointerInfo.ptPixelLocation;
    ::SetCursorPos(p.x,p.y);
    ::ScreenToClient(hwnd, &p);

    switch(message) {
    case WM_POINTERDOWN:
     pendown=true;
     pen.action = PEN_DOWN;
     ClickActivateWnd();
     break;
    case WM_POINTERUP:
     pendown=false;
     pen.action = PEN_UP;
     break;
    }
    if(!IsNull(DoMouse(PEN, p, 0)))
     return 0L;
    break;
   }
  }
  break;
...
```

The main problem was that ::GetCursorPos() was not working as ::SetCursorPos() was not called from WM_POINTER*. Now it all works and mouse follows pen.

Best regards,

Tom

EDIT: Not quite... calling SetCursorPos() causes incoming WM_MOUSEMOVE messages at some frequency, but the coordinates are obviously old ones causing erratic behavior if both MouseMove and Pen are processed... However, if SetCursorPos() is not called, the mouse position will be different to pen and it will in turn cause flickering RectTracker with intermediate rectangles to mouseposition! :(

Well, I'm done for today. (or yesterday, or whatever.) Anyway, if you have any ideas how to go

around this, please fill in...

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 26 Mar 2021 07:57:50 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Thu, 25 March 2021 22:28Mirek,

Is pn.action intentionally merely a state flag showing continuously PEN_UP or PEN_DOWN? Or should we reset it along with other PenInfo items in the beginning of processing WM_POINTER* messages?

I would like to have it reset, in order to more easily detect UP/DOWN state changes without a state variable and treat zero action as 'PEN_MOVE'.

Then we could do this to decode all kinds of pen events:

```
#define PENMOVE  0
#define PENDOWN  1
#define PENUP  2
#define PENHOLD  3
#define PENDRAG  4
#define PENDOUBLE 5
#define PENTRIPLE 6

bool IsNear(Point p, Point o){
 Point delta(p-o);
 if(abs(delta.x)<5 && abs(delta.y)<5) return true;
 return false;
}

int PenDecoder(Point p, const PenInfo& pn, dword keyflags){
 static int action=0;
 static int downt=0;
 static Point downp=Null;
 static bool doubleClick=true;
 static bool down=false;
 static bool solved=false;
 int now=msecs();
 int deltat=now-downt;


 bool isNear=IsNear(p,downp);

 switch(pn.action){
  case PEN_DOWN:
   down=true;
```

```
   solved=false;
   downp=p;
   downt=now;
   if(isNear){
    if(now-downt < 500){
     if(doubleClick){
      return PENTRIPLE;
     }
     else{
      doubleClick=true;
      return PENDOUBLE;
     }
    }
   }
   doubleClick=false;
   return PENDOWN;

  case PEN_UP:
   down=false;
   return PENUP;

  default:
   if(down && !solved){
    if(isNear){
     if(now-downt > 1000){
      solved=true;
      return PENHOLD;
     }
    }
    else{
     solved=true;
     return PENDRAG;
    }
   }
   return PENMOVE;
 }
}
```

Best regards,

Tom

Would it solve any real problem? I mean, I believe that the application can do all these on its own, if it needs to.

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 26 Mar 2021 09:15:18 GMT
View Forum Message <> Reply to Message

Hi,

Well, I put the 'PenDecoder' here just for reference for anyone wishing to map pen to mouse like callbacks in application. It does not really need to be part of CtrlCore.

However, my point here is that, I would feel more at home, if:
- only WM_POINTERUP would cause pn.action=PEN_UP
- only WM_POINTERDOWN would cause pn.action=PEN_DOWN
- and WM_POINTERUPDATE would always keep pn.action=0; (or PEN_MOVE).

I.e. pn.action would be event type instead of state flag of up/down status.

Or do you have some particular reason to always have pn.action set to either PEN_UP or PEN_DOWN depepending on the last change?

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 26 Mar 2021 11:42:41 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 26 March 2021 10:15

Or do you have some particular reason to always have pn.action set to either PEN_UP or PEN_DOWN depepending on the last change?

Best regards,

Tom

Ah, sorry, I misunderstood the intent. Nope, that was just a bug, forgot to assign 0 in WindowProc...

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 26 Mar 2021 12:39:53 GMT
View Forum Message <> Reply to Message

Good! :)

Maybe you can add a:

#define PEN_MOVE 0
or something alike in CtrlCore.h?

Now about that RectTracker: I'm having hard time with the ::GetCursorPos/::SetCursorPos...

::GetCursorPos knows where mouse is, but in order for it to know where Pen is, we would need to call ::SetCursorPos in WM_POINTER*. But calling it results in a slightly delayed WM_MOUSEMOVE message coming in causing all kinds of annoying stuff happening.

If we do not call ::SetCursorPos in WM_POINTER, the Pen and Mouse will be in two different positions causing other issues in many places because ::GetCursorPos only knows about the mouse, not pen.

Any ideas how to proceed?

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Novo on Fri, 26 Mar 2021 14:05:06 GMT
View Forum Message <> Reply to Message

Linux, Clang:
./umk uppsrc ide CLANG -bus
/home/buildbot/worker/l-upp/build/uppsrc/CtrlCore/GtkEvent.cpp:429:6: error: no viable overloaded '='
	pen = CurrentEvent.pen;
	~~~ ^ ~~~~~~~~~~~~~~~~

Subject: Re: Using Pen with U++
Posted by Novo on Fri, 26 Mar 2021 14:08:55 GMT
View Forum Message <> Reply to Message

Alpine Linux (in addition to previous error):
./umk uppsrc ide CLANG -bus
/home/buildbot/worker/al-upp/build/uppsrc/CtrlCore/GtkEvent.cpp:430:2: error: use of undeclared identifier 'pen_barrel'
	pen_barrel = CurrentEvent.pen_barrel;
	^
/home/buildbot/worker/al-upp/build/uppsrc/CtrlCore/GtkEvent.cpp:431:2: error: use of undeclared identifier 'pen_inverted'
	pen_inverted = CurrentEvent.pen_inverted;
	^
	~~~

/home/buildbot/worker/al-upp/build/uppsrc/CtrlCore/GtkEvent.cpp:432:2: error: use of undeclared identifier 'pen_eraser'
	pen_eraser = CurrentEvent.pen_eraser;
	^
/home/buildbot/worker/al-upp/build/uppsrc/CtrlCore/GtkEvent.cpp:433:2: error: use of undeclared identifier 'pen_pressure'
	pen_pressure = CurrentEvent.pen_pressure;
	^
/home/buildbot/worker/al-upp/build/uppsrc/CtrlCore/GtkEvent.cpp:434:2: error: use of undeclared identifier 'pen_rotation'
	pen_rotation = CurrentEvent.pen_rotation;
	^
/home/buildbot/worker/al-upp/build/uppsrc/CtrlCore/GtkEvent.cpp:435:2: error: use of undeclared identifier 'pen_tilt'
	pen_tilt = CurrentEvent.pen_tilt;
	^

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 26 Mar 2021 14:45:07 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 26 March 2021 13:39Good! :)

Maybe you can add a:
#define PEN_MOVE 0
or something alike in CtrlCore.h?

Is this just MOVE though? I can add PEN_OTHER, but do not see much point. It is UP, DOWN or nothing...

Now about that RectTracker: I'm having hard time with the ::GetCursorPos/::SetCursorPos...

Quote:
If we do not call ::SetCursorPos in WM_POINTER, the Pen and Mouse will be in two different positions causing other issues in many places because ::GetCursorPos only knows about the mouse, not pen.

Any ideas how to proceed?

Well, the whole point of Pen virtual method is that we should not meddle with normal mouse.... Is it so bad to just leave it to LeftDown?

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 26 Mar 2021 15:44:19 GMT
View Forum Message <> Reply to Message

Mirek,

'case PEN_OTHER:' Looks better than 'case 0:' ... :)

Quote:Is it so bad to just leave it to LeftDown?

I'm not quite sure what do you mean by this.. (?) (If it is about double clicks, drags or barrel mappings to right mouse button, I can handle them in the application side.)

--

Anyway, now the good news: I finally got through with RectTracker and Get/SetCursorPos.

The set of changes required follow.

Add to CtrlCore.h:
```
bool GetPenDown(); // << ADD THIS
... AND ...
class RectTracker : public LocalLoop {
public:
 virtual void  LeftUp(Point, dword);
 virtual void  RightUp(Point, dword);
 virtual void  MouseMove(Point p, dword);
 virtual bool  Pen(Point p, const PenInfo& pn, dword keyflags); // << ADD THIS
```

Add "&& !GetPenDown()" at line 595 in CtrlMouse.cpp:
```
...
 if(findarg(e, LEFTUP, RIGHTUP, MIDDLEUP) >= 0)
  KillRepeat();
 Image result = DispatchMouseEvent(e, p, zd);
 if(!GetMouseRight() && !GetMouseMiddle() && !GetMouseLeft() && !GetPenDown())
  ReleaseCtrlCapture();
 return result;
...
```

Add to LocalLoop.cpp:
```
bool RectTracker::Pen(Point p, const PenInfo& pn, dword keyflags){
 switch(pn.action){
  case 0:
   MouseMove(p,keyflags);
   break;
  case PEN_UP:
   EndLoop();
   break;
 }
```

```
 return true;
}
```
Add "&& !GetPenDown()" at line 781 in Win32Wnd.cpp:
```
...
bool Ctrl::ProcessEvent(bool *quit)
{
 ASSERT_(IsMainThread(), "ProcessEvent can only run in the main thread");
 if(!GetMouseLeft() && !GetMouseRight() && !GetMouseMiddle() && !GetPenDown()) // << HERE
  ReleaseCtrlCapture();
...
```

And the rest is in the attached Win32Proc.cpp.

Best regards,

Tom

File Attachments

1) Win32Proc.cpp, downloaded 138 times

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 26 Mar 2021 17:19:49 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 26 March 2021 16:44Mirek,

'case PEN_OTHER:' Looks better than 'case 0:' ... :)

'default:' is fine too :)

Quote:

Quote:Is it so bad to just leave it to LeftDown?

I'm not quite sure what do you mean by this.. (?) (If it is about double clicks, drags or barrel mappings to right mouse button, I can handle them in the application side.)

```
 bool Pen(Point p, const PenInfo& pn, dword keyflags) override {
  if(keyflags & K_SHIFT)
   return false;
  if(pn.pressure) {
   if((!!pn.pressure == !!pen.pressure) && drawing.GetCount())
    drawing.Top().Add(MakeTuple(pn.pressure, p));
```

```
  else
   drawing.Add().Add(MakeTuple(pn.pressure, p));
 }
 pen = pn;
 Refresh();
 return true;
}

void LeftDown(Point p, dword keyflags) override {
 if(keyflags & K_SHIFT) {
  RectTracker tracker(*this);
  tracker.MinSize(Size(-100000,-100000));
  tracker.Track(Rect(p,p));
 }
}
```

I mean, the main reason for separated Pen method interface is that we do not want to add support to any client code. So to start adding just that right away is perhaps not the best idea?

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 26 Mar 2021 17:52:51 GMT
View Forum Message <> Reply to Message

Yes, default: is great (if you also handle both "case PEN_DOWN:" and "case PEN_UP:") Anyway, this is not important now. I can write case 0: :)

I agree 100 %: We do not want to add any pen support to client code.

However, e.g. RectTracker does not work as is with my Wacom tablet. The specific pen support (I just posted) is simply required for it to work. (I'm now under impression that RectTracker works with your tablet perfectly without these changes... ?? Perhaps XP-PEN drivers can avoids Windows Ink and work more mouse-like through the stack.)

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 27 Mar 2021 08:11:56 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 26 March 2021 18:52
However, e.g. RectTracker does not work as is with my Wacom tablet. The specific pen support (I just posted) is simply required for it to work. (I'm now under impression that RectTracker works with your tablet perfectly without these changes... ?? Perhaps XP-PEN drivers can avoids Windows Ink and work more mouse-like through the stack.)

Sorry, I might get lost in the code that we exchange here a lot...

Do you mean that my variant where Pen returns false and RectTracker is called in normal LeftDown does not work with Wacom? (My impression so far was that you are only testing RectTracker in Pen virtual method).

If so, does it mean RectTracker does not work even in pre-Pen U++? (With Wacom in "mouse mode")

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 27 Mar 2021 11:20:46 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 27 March 2021 10:11Tom1 wrote on Fri, 26 March 2021 18:52
However, e.g. RectTracker does not work as is with my Wacom tablet. The specific pen support (I just posted) is simply required for it to work. (I'm now under impression that RectTracker works with your tablet perfectly without these changes... ?? Perhaps XP-PEN drivers can avoids Windows Ink and work more mouse-like through the stack.)

Sorry, I might get lost in the code that we exchange here a lot...

Do you mean that my variant where Pen returns false and RectTracker is called in normal LeftDown does not work with Wacom? (My impression so far was that you are only testing RectTracker in Pen virtual method).

If so, does it mean RectTracker does not work even in pre-Pen U++? (With Wacom in "mouse mode")

Mirek

Hi,

With original pre-Pen U++, RectTracker worked with Wacom at correct coordinates. (It only suffered from the Windows Ink induced 2 cm starting delay, but the rectangle followed pen thereafter.)

With current trunk code and testing with:

```
bool Pen(Point p, const PenInfo& pn, dword keyflags) override {
  if(keyflags & K_SHIFT)
   return false;
  if(pn.pressure) {
   if((!!pn.pressure == !!pen.pressure) && drawing.GetCount())
    drawing.Top().Add(MakeTuple(pn.pressure, p));
   else
    drawing.Add().Add(MakeTuple(pn.pressure, p));
  }
  pen = pn;
  Refresh();
  return true;
 }

 void LeftDown(Point p, dword keyflags) override {
  if(keyflags & K_SHIFT) {
   RectTracker tracker(*this);
   tracker.MinSize(Size(-100000,-100000));
   tracker.Track(Rect(p,p));
  }
 }
```

the tracking rect still of course has the 2 cm starting delay. However, now the rect dragged corner lags behind with a 2 cm constant dx,dy offset. Please see below a photo of a live 'while dragging' situation. So, actually, RectTracker is now worse than before introducing pen support:

Further on, what I really need with RectTracker, is that there is no 2 cm reaction delay. The tracking must start immediately from PEN_DOWN event. My RectTracker inherited sketching code requires immediate reaction for even a single pixel pen drags. This is why I need specific Pen() support in RectTracker.

Best regards,

Tom

File Attachments

1) 20210327_120800.jpg, downloaded 436 times

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 27 Mar 2021 19:23:43 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 27 March 2021 12:20mirek wrote on Sat, 27 March 2021 10:11Tom1 wrote on Fri, 26 March 2021 18:52
However, e.g. RectTracker does not work as is with my Wacom tablet. The specific pen support (I

just posted) is simply required for it to work. (I'm now under impression that RectTracker works with your tablet perfectly without these changes... ?? Perhaps XP-PEN drivers can avoids Windows Ink and work more mouse-like through the stack.)

Sorry, I might get lost in the code that we exchange here a lot...

Do you mean that my variant where Pen returns false and RectTracker is called in normal LeftDown does not work with Wacom? (My impression so far was that you are only testing RectTracker in Pen virtual method).

If so, does it mean RectTracker does not work even in pre-Pen U++? (With Wacom in "mouse mode")

Mirek

Hi,

With original pre-Pen U++, RectTracker worked with Wacom at correct coordinates. (It only suffered from the Windows Ink induced 2 cm starting delay, but the rectangle followed pen thereafter.)

With current trunk code and testing with:

```
bool Pen(Point p, const PenInfo& pn, dword keyflags) override {
  if(keyflags & K_SHIFT)
   return false;
  if(pn.pressure) {
   if((!!pn.pressure == !!pen.pressure) && drawing.GetCount())
    drawing.Top().Add(MakeTuple(pn.pressure, p));
   else
    drawing.Add().Add(MakeTuple(pn.pressure, p));
  }
  pen = pn;
  Refresh();
  return true;
 }

 void LeftDown(Point p, dword keyflags) override {
  if(keyflags & K_SHIFT) {
   RectTracker tracker(*this);
   tracker.MinSize(Size(-100000,-100000));
   tracker.Track(Rect(p,p));
  }
 }
```

the tracking rect still of course has the 2 cm starting delay. However, now the rect dragged corner lags behind with a 2 cm constant dx,dy offset. Please see below a photo of a live 'while dragging' situation. So, actually, RectTracker is now worse than before introducing pen support:

Further on, what I really need with RectTracker, is that there is no 2 cm reaction delay. The tracking must start immediately from PEN_DOWN event. My RectTracker inherited sketching code requires immediate reaction for even a single pixel pen drags. This is why I need specific Pen() support in RectTracker.

Best regards,

Tom

OK, this works just fine with xp-pen...

Widgets in general do work? (Because if it is this way, I would just expect the offset to be everywhere...)

What happens if you remove return 0L in pen processing?

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 27 Mar 2021 19:45:43 GMT
View Forum Message <> Reply to Message

Yes. Widgets mostly work. Some flickering with e.g. Menu, but I think nothing is severely broken. I can do a more thorough check later.

Breaking out from WM_POINTER* instead of return 0L; does not have any effect on this test with RectTracker. Offset remains.

The core problem in my opinion is ::GetCursorPos which does not work with pen. This comes evident with drags.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sun, 28 Mar 2021 11:15:31 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 27 March 2021 20:45Yes. Widgets mostly work. Some flickering with e.g. Menu, but I think nothing is severely broken. I can do a more thorough check later.

Breaking out from WM_POINTER* instead of return 0L; does not have any effect on this test with

RectTracker. Offset remains.

The core problem in my opinion is ::GetCursorPos which does not work with pen. This comes evident with drags.

OK. In the new commit I treat GetCursorPos as "not reliable", overriding it with position got from lParam.

Can you try?

Subject: Re: Using Pen with U++
Posted by Tom1 on Sun, 28 Mar 2021 16:11:38 GMT
View Forum Message <> Reply to Message

Quote:OK. In the new commit I treat GetCursorPos as "not reliable", overriding it with position got from lParam.
Hi Mirek,

That was a correct move. Now RectTracker works correctly after 2 cm Windows Ink threshold is crossed. This is via "Pen() return false; to LeftDown()".

In Pen(), RectTracker does not work at all yet.

I also tested with UWord to see how different widgets work. I could not immediately find anything that does not. Only dragging a top menu item to open a drop down menu causes flickering of the selected item on the drop down menu. If the drop down menu is opened by a pen click, hovering on top of the sub menu works correctly without any flickering.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Mon, 29 Mar 2021 09:10:01 GMT
View Forum Message <> Reply to Message

[quote title=Tom1 wrote on Sun, 28 March 2021 18:11]Quote:Only dragging a top menu item to open a drop down menu causes flickering of the selected item on the drop down menu. If the drop down menu is opened by a pen click, hovering on top of the sub menu works correctly without any flickering.

I do not see anything like that, which probably means I am not testing correctly.

Also, can you test the same thing with "pre-pen" U++?

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 29 Mar 2021 10:33:42 GMT
View Forum Message <> Reply to Message

Mirek,

I did a video clip portraying the menu highlight flickering issue. Sent a PM on that.

There is no such anomaly on pre-pen U++.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Mon, 29 Mar 2021 11:57:40 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 29 March 2021 12:33Mirek,

I did a video clip portraying the menu highlight flickering issue. Sent a PM on that.

There is no such anomaly on pre-pen U++.

Best regards,

Tom

Well, all works fine with xp-pen (again...).

If you deactivate Windows Ink for tablet, what does it do?

Does it mean it now does those weird click animations (extending circle) even for mouse?

Mirek

Subject: Re: Using Pen with U++

Posted by Tom1 on Mon, 29 Mar 2021 12:36:43 GMT
View Forum Message <> Reply to Message

The click animations are never present with mouse.

When using Windows Ink -mode for Wacom, I get those click animations for pen.

When I disable Windows Ink -mode for Wacom, the thing starts to work exactly like mouse:
+ no click animations
+ immediate response with drags (i.e. no annoying 2 cm delay)
- NO Pen() event. Everything comes in as MouseMove, LeftDown, LeftUp... (This likely means there is no WM_POINTER* messages and everything is already translated to WM_MOUSEMOVE, WM_LBUTTONDOWN,... etc.)
- NO pressure detection either due to missing WM_POINTER* -messages.

So, using Windows Ink -mode is a must.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Mon, 29 Mar 2021 13:35:21 GMT
View Forum Message <> Reply to Message

Well, that means that in the movie, you have used pen to open the menu, right?

Does this disbehave with mouse too? (Without animation?)

What happens if you comment out CtrlCore/Win32Proc.cpp:74?

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 29 Mar 2021 13:50:01 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 29 March 2021 16:35Well, that means that in the movie, you have used pen to open the menu, right?

Does this disbehave with mouse too? (Without animation?)

What happens if you comment out CtrlCore/Win32Proc.cpp:74?

Mirek

1. Yes, I used pen throughout the video.

2. With mouse it works perfectly OK. There is no click animation with mouse.

3. Commenting out:
// POINT p;
// if(::GetCursorPos(&p))
  //CurrentMousePos = p;
 Does not change anything.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Mon, 29 Mar 2021 14:01:21 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 29 March 2021 15:50mirek wrote on Mon, 29 March 2021 16:35Well, that means that in the movie, you have used pen to open the menu, right?

Does this disbehave with mouse too? (Without animation?)

What happens if you comment out CtrlCore/Win32Proc.cpp:74?

Mirek

1. Yes, I used pen throughout the video.

2. With mouse it works perfectly OK. There is no click animation with mouse.

3. Commenting out:
// POINT p;
// if(::GetCursorPos(&p))
  //CurrentMousePos = p;
 Does not change anything.

Best regards,

Tom

Pity, I was almost sure it must be related...

OK. The place that says whether the item is highlighted or not is

CtrlLib/MenuItem.cpp:267

Putting some DDUMPs for parts of that complex condition should reveal the source of oscilation.

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 29 Mar 2021 14:20:25 GMT
View Forum Message <> Reply to Message

Logging the changes shows that it's HasFocus() and especially HasMouse() that oscillates:
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 32 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 31 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 31 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true

[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 31 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 32 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 31 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 31 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true

[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 31 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Mon, 29 Mar 2021 14:54:28 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 29 March 2021 16:20Logging the changes shows that it's HasFocus() and especially HasMouse() that oscillates:
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 32 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 31 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true

[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 31 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 31 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 32 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 31 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true

[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 31 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 31 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true

Best regards,

Tom

Frankly, not sure what to do next... :(

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 29 Mar 2021 15:04:24 GMT
View Forum Message <> Reply to Message

Do you have Windows Ink enabled with your XP-PEN? If not, please do. You might then see the click animations and experience the drag lag. Maybe the menu issue shows itself then...(?)

BR,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 29 Mar 2021 15:18:49 GMT
View Forum Message <> Reply to Message

Thinking about it, as hovering with pen has no trouble, it must be an issue with 'pendown' not comparable to left mouse button being down. It might be partly identified as left mouse button being pressed and partly not, giving mixed messages about the state.

This is something that partially prevents RectTracker from being used with/from Pen() interface. (Which is still a main concern for me...)

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 29 Mar 2021 15:29:48 GMT
View Forum Message <> Reply to Message

OK, I logged some MEvent0 coordinates and tracked for mouseCtrl changes. It seems this is flipping mouseCtrl a lot:

[+ 0 ms] WM_POINTERUPDATE
[+ 0 ms] MEvent0 [141, 199]
[+ 0 ms] WM_POINTERUPDATE
[+ 0 ms] MEvent0 [141, 199]
[+ 0 ms] WM_POINTERUPDATE
[+ 0 ms] MEvent0 [141, 199]
[+ 0 ms] HasMouse() = false
[+ 0 ms] MEvent0 [141, 199]
[+ 0 ms] WM_POINTERUPDATE
[+ 0 ms] MEvent0 [141, 199]
[+ 0 ms] WM_POINTERUPDATE
[+ 0 ms] MEvent0 [142, 199]
[+ 0 ms] MEvent0 [142, 199]
[+ 0 ms] MEvent0 [142, 199]
[+ 0 ms] WM_MOUSEMOVE
[+ 0 ms] MEvent0 [139, 21]
[+ 0 ms] mouseCtrl != this
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] MEvent0 [139, 21]
[+ 31 ms] WM_POINTERUPDATE

```
[+     0 ms] MEvent0 [142, 199]
[+     0 ms] mouseCtrl != this
[+     0 ms] HasFocus() = false
[+     0 ms] HasMouse() = false
[+     0 ms] HasMouse() = true
[+     0 ms] HasMouse() = false
[+     0 ms] HasMouse() = true
[+     0 ms] WM_POINTERUPDATE
[+     0 ms] MEvent0 [142, 199]
[+     0 ms] WM_POINTERUPDATE
[+     0 ms] MEvent0 [142, 199]
[+     0 ms] HasMouse() = false
[+     0 ms] MEvent0 [142, 199]
[+    16 ms] WM_POINTERUPDATE
[+     0 ms] MEvent0 [143, 199]
[+     0 ms] MEvent0 [143, 199]
[+     0 ms] MEvent0 [143, 199]
[+     0 ms] WM_MOUSEMOVE
[+     0 ms] MEvent0 [140, 21]
[+     0 ms] mouseCtrl != this
[+     0 ms] HasFocus() = true
[+     0 ms] HasMouse() = true
[+     0 ms] HasFocus() = false
[+     0 ms] HasMouse() = false
[+     0 ms] HasFocus() = true
[+     0 ms] HasMouse() = true
[+     0 ms] MEvent0 [140, 21]
[+    16 ms] WM_POINTERUPDATE
[+     0 ms] MEvent0 [143, 199]
[+     0 ms] mouseCtrl != this
[+     0 ms] HasFocus() = false
[+     0 ms] HasMouse() = false
[+     0 ms] HasMouse() = true
[+     0 ms] HasMouse() = false
[+     0 ms] HasMouse() = true
[+     0 ms] WM_POINTERUPDATE
[+     0 ms] MEvent0 [143, 199]
```

Please note that pen is nearly stationary during this.

Best regards,

Tom

EDIT: Added WM_POINTERUPDATE and WM_MOUSEMOVE markers to the log, so that we can identify that the incompatible coordinates are coming from respective sources.

Subject: Re: Using Pen with U++
Posted by mirek on Mon, 29 Mar 2021 16:17:54 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 29 March 2021 17:04Do you have Windows Ink enabled with your XP-PEN? If not, please do. You might then see the click animations and experience the drag lag. Maybe the menu issue shows itself then...(?)

BR,

Tom

I have it enabled, I see the animation, but no problems.

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 29 Mar 2021 18:41:51 GMT
View Forum Message <> Reply to Message

Mirek,

I'm almost sure this problem is about CtrlCore not entirely knowing about pen down condition. In a way, this resembles the ::GetCursorPos() issue you just fixed, where ::GetCursorPos() was not entirely aware of pen position, but only knew it after default processing. In this case GetMouseLeft() asks Windows for the left button state, but pen down is not the same as left mouse button in this case and is not detected. E.g. mouseCtrl and captureCtrl may be affected by this fact.

Again, I'm not familiar enough with the CtrlCore to make it properly aware about pen down condition. Can you improve the 'pen down awareness' of CtrlCore?

I tried:
static bool pendown=false;
bool GetMouseLeft() { return pendown || (!!(GetKeyStateSafe(VK_LBUTTON) & 0x8000)); }
and updating the flag with WM_POINTERDOWN/WM_POINTERDOWN, but this was not enough.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 29 Mar 2021 20:02:21 GMT
View Forum Message <> Reply to Message

Hi Mirek,

See the switching Ctrls/hwnds for WM_POINTER and WM_MOUSEMOVE messages while on

top of menu:
[+ 0 ms] WM_MOUSEMOVE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xd3009a)
[+ 0 ms] Ctrl::DoMouse() [141, 302]
[+ 0 ms] MEvent0 [138, 19]
[+ 0 ms] mouseCtrl != this
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true
[+ 0 ms] HasMouse() = true
[+ 0 ms] MEvent0 [138, 19]
[+ 15 ms] WM_POINTERUPDATE 5UWord : 0x6b88070 (hwnd 0x7c02b4)
[+ 0 ms] Ctrl::DoMouse() [141, 343]
[+ 0 ms] MEvent0 [141, 197]
[+ 0 ms] mouseCtrl != this
[+ 0 ms] HasFocus() = false
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasMouse() = true
[+ 0 ms] WM_POINTERUPDATE 5UWord : 0x6b88070 (hwnd 0x7c02b4)
[+ 0 ms] Ctrl::DoMouse() [142, 343]
[+ 0 ms] MEvent0 [142, 197]
[+ 0 ms] MEvent0 [142, 197]
[+ 16 ms] MEvent0 [142, 197]
[+ 0 ms] WM_MOUSEMOVE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xd3009a)
[+ 0 ms] Ctrl::DoMouse() [142, 302]
[+ 0 ms] MEvent0 [139, 19]
[+ 0 ms] mouseCtrl != this
[+ 0 ms] HasMouse() = false
[+ 0 ms] HasFocus() = true

Although not shown here, the screen coordinates come in right for both events, but the translation differs in DoMouse() because the events end up in different Ctrls/hwnds. Therefore, also local coordinates mismatch. As you can see, WM_POINTERUPDATE lands in main window, whereas WM_MOUSEMOVE goes to drop down menu just as it should.

Hope this helps...

Best regards,

Tom

EDIT: Plain hovering on an open drop down menu makes both WM_POINTERUPDATE and WM_MOUSEMOVE land in the drop down menu:
[+ 0 ms] WM_POINTERUPDATE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xb10064)

[+ 0 ms] Ctrl::DoMouse() [124, 302]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] WM_MOUSEMOVE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xb10064)
[+ 0 ms] Ctrl::DoMouse() [124, 302]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] MEvent0 [121, 19]
[+ 16 ms] MEvent0 [121, 19]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] WM_POINTERUPDATE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xb10064)
[+ 0 ms] Ctrl::DoMouse() [124, 302]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] WM_POINTERUPDATE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xb10064)
[+ 0 ms] Ctrl::DoMouse() [123, 302]
[+ 0 ms] MEvent0 [120, 19]
[+ 0 ms] MEvent0 [120, 19]
[+ 0 ms] MEvent0 [120, 19]
[+ 0 ms] WM_MOUSEMOVE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xb10064)
[+ 0 ms] Ctrl::DoMouse() [123, 302]
[+ 0 ms] MEvent0 [120, 19]
[+ 0 ms] MEvent0 [120, 19]
[+ 16 ms] MEvent0 [120, 19]
[+ 0 ms] MEvent0 [120, 19]
[+ 0 ms] WM_POINTERUPDATE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xb10064)
[+ 0 ms] Ctrl::DoMouse() [123, 301]

Drag is the difference...

Subject: Re: Using Pen with U++
Posted by mirek on Tue, 30 Mar 2021 09:21:05 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 29 March 2021 22:02Hi Mirek,

See the switching Ctrls/hwnds for WM_POINTER and WM_MOUSEMOVE messages while on top of menu:

Looks like Wacom sends WM_POINTER to the window that has keyboard focus...

Can you add something like DDUMP(Name(GetFocusCtrl()) before WM_MOUSEMOVE / WM_POINT... ?

That said, the true reason is perhaps that Pen is altering mouseCtrl. That should be fixable...

Mirek

```
[+     0 ms] WM_MOUSEMOVE  N3Upp7MenuBarE  : 0x6a82880 (hwnd 0xd3009a)
[+     0 ms] Ctrl::DoMouse() [141, 302]
[+     0 ms] MEvent0 [138, 19]
[+     0 ms] mouseCtrl != this
[+     0 ms] HasMouse() = false
[+     0 ms] HasFocus() = true
[+     0 ms] HasMouse() = true
[+     0 ms] HasFocus() = false
[+     0 ms] HasMouse() = false
[+     0 ms] HasFocus() = true
[+     0 ms] HasMouse() = true
[+     0 ms] MEvent0 [138, 19]
[+    15 ms] WM_POINTERUPDATE 5UWord  : 0x6b88070 (hwnd 0x7c02b4)
[+     0 ms] Ctrl::DoMouse() [141, 343]
[+     0 ms] MEvent0 [141, 197]
[+     0 ms] mouseCtrl != this
[+     0 ms] HasFocus() = false
[+     0 ms] HasMouse() = false
[+     0 ms] HasMouse() = true
[+     0 ms] HasMouse() = false
[+     0 ms] HasMouse() = true
[+     0 ms] WM_POINTERUPDATE 5UWord  : 0x6b88070 (hwnd 0x7c02b4)
[+     0 ms] Ctrl::DoMouse() [142, 343]
[+     0 ms] MEvent0 [142, 197]
[+     0 ms] MEvent0 [142, 197]
[+    16 ms] MEvent0 [142, 197]
[+     0 ms] WM_MOUSEMOVE  N3Upp7MenuBarE  : 0x6a82880 (hwnd 0xd3009a)
[+     0 ms] Ctrl::DoMouse() [142, 302]
[+     0 ms] MEvent0 [139, 19]
[+     0 ms] mouseCtrl != this
[+     0 ms] HasMouse() = false
[+     0 ms] HasFocus() = true
```

Although not shown here, the screen coordinates come in right for both events, but the translation differs in DoMouse() because the events end up in different Ctrls/hwnds. Therefore, also local coordinates mismatch. As you can see, WM_POINTERUPDATE lands in main window, whereas WM_MOUSEMOVE goes to drop down menu just as it should.

Hope this helps...

Best regards,

Tom

EDIT: Plain hovering on an open drop down menu makes both WM_POINTERUPDATE and

WM_MOUSEMOVE land in the drop down menu:
[+ 0 ms] WM_POINTERUPDATE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xb10064)
[+ 0 ms] Ctrl::DoMouse() [124, 302]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] WM_MOUSEMOVE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xb10064)
[+ 0 ms] Ctrl::DoMouse() [124, 302]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] MEvent0 [121, 19]
[+ 16 ms] MEvent0 [121, 19]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] WM_POINTERUPDATE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xb10064)
[+ 0 ms] Ctrl::DoMouse() [124, 302]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] MEvent0 [121, 19]
[+ 0 ms] WM_POINTERUPDATE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xb10064)
[+ 0 ms] Ctrl::DoMouse() [123, 302]
[+ 0 ms] MEvent0 [120, 19]
[+ 0 ms] MEvent0 [120, 19]
[+ 0 ms] MEvent0 [120, 19]
[+ 0 ms] WM_MOUSEMOVE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xb10064)
[+ 0 ms] Ctrl::DoMouse() [123, 302]
[+ 0 ms] MEvent0 [120, 19]
[+ 0 ms] MEvent0 [120, 19]
[+ 16 ms] MEvent0 [120, 19]
[+ 0 ms] MEvent0 [120, 19]
[+ 0 ms] WM_POINTERUPDATE N3Upp7MenuBarE : 0x6a82880 (hwnd 0xb10064)
[+ 0 ms] Ctrl::DoMouse() [123, 301]

Drag is the difference...[/quote]

Subject: Re: Using Pen with U++
Posted by mirek on Tue, 30 Mar 2021 09:26:26 GMT
View Forum Message <> Reply to Message

Try CtrlCore/CtrlMouse.cpp:320:

```
 if(e != PEN && mouseCtrl != this) {
```

Subject: Re: Using Pen with U++
Posted by Tom1 on Tue, 30 Mar 2021 09:36:29 GMT

mirek wrote on Tue, 30 March 2021 12:26Try CtrlCore/CtrlMouse.cpp:320:

```
 if(e != PEN && mouseCtrl != this) {
```

The focusCtrl is switching between the menubar and the menuitem regardless the above change.
The logs are taken just before DoMouse call in WM_MOUSEMOVE and WM_POINTERUPDATE:

```
[+     0 ms] WM_MOUSEMOVE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+    16 ms] WM_POINTERUPDATE N3Upp8MenuItemE  : 0x6c60c60 (parent N3Upp7BarPaneE)
[+     0 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+    15 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+    16 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+    15 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_MOUSEMOVE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+    16 ms] WM_POINTERUPDATE N3Upp8MenuItemE  : 0x6c60c60 (parent N3Upp7BarPaneE)
[+     0 ms] WM_POINTERUPDATE N3Upp8MenuItemE  : 0x6c60c60 (parent N3Upp7BarPaneE)
[+    16 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_MOUSEMOVE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+    15 ms] WM_POINTERUPDATE N3Upp8MenuItemE  : 0x6c60c60 (parent N3Upp7BarPaneE)
[+     0 ms] WM_POINTERUPDATE N3Upp8MenuItemE  : 0x6c60c60 (parent N3Upp7BarPaneE)
[+    16 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_MOUSEMOVE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+    16 ms] WM_POINTERUPDATE N3Upp8MenuItemE  : 0x6c60c60 (parent N3Upp7BarPaneE)
[+     0 ms] WM_POINTERUPDATE N3Upp8MenuItemE  : 0x6c60c60 (parent N3Upp7BarPaneE)
[+    15 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_MOUSEMOVE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+    16 ms] WM_POINTERUPDATE N3Upp8MenuItemE  : 0x6c60c60 (parent N3Upp7BarPaneE)
[+     0 ms] WM_POINTERUPDATE N3Upp8MenuItemE  : 0x6c60c60 (parent N3Upp7BarPaneE)
[+    15 ms] WM_POINTERUPDATE N3Upp8MenuItemE  : 0x6c60c60 (parent N3Upp7BarPaneE)
[+     0 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_MOUSEMOVE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+    16 ms] WM_POINTERUPDATE N3Upp8MenuItemE  : 0x6c60c60 (parent N3Upp7BarPaneE)
[+     0 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_MOUSEMOVE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+    16 ms] WM_POINTERUPDATE N3Upp8MenuItemE  : 0x6c60c60 (parent N3Upp7BarPaneE)
[+    15 ms] WM_POINTERUPDATE N3Upp8MenuItemE  : 0x6c60c60 (parent N3Upp7BarPaneE)
[+     0 ms] WM_POINTERUPDATE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
[+     0 ms] WM_MOUSEMOVE N3Upp7BarPaneE  : 0x6bc2978 (parent N3Upp7MenuBarE)
```

Best regards,

Tom

EDIT: Fixed content.

Subject: Re: Using Pen with U++
Posted by mirek on Tue, 30 Mar 2021 10:00:18 GMT
View Forum Message <> Reply to Message

OK, I tried to completely separate Pen from Mouse now. Please try the trunk.

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Tue, 30 Mar 2021 10:44:09 GMT
View Forum Message <> Reply to Message

Hi,

Thank you Mirek! Well done! Now the menus work beautifully.

Can you take a look at the attached RectTracker testcase? It includes my Sketcher class which inherits RectTracker. In order to draw short detailed curves with freehand (i.e. Alt+PenDrag), Pen() support is needed in RectTracker. The 2 cm drag threshold with Wacom prevents relying on mouse processing (i.e. // 2. option in comment).

Thanks and best regards,

Tom

File Attachments
1) main.cpp, downloaded 173 times

Subject: Re: Using Pen with U++
Posted by Tom1 on Tue, 30 Mar 2021 13:26:29 GMT
View Forum Message <> Reply to Message

Hi Mirek,

The following changes would add Pen support to RectTracker (and the inherited Sketcher).

Add to RectTracker in CtrlCore.h:

```
public:
 virtual bool  Pen(Point p, const PenInfo &pn, dword keyflags);
 virtual void  DoMouseMove(Point p, dword);

protected:
 int    pointer;
```

Changes in LocalLoop.cpp:

```
RectTracker::RectTracker(Ctrl& master)
{
 pointer=Null; // << ADD THIS IN CONSTRUCTOR

...

bool  RectTracker::Pen(Point p, const PenInfo &pn, dword keyflags){ // ADD Pen() here
 if(IsNull(pointer)) pointer=1;
 if(pointer!=1) return false;

 switch(pn.action){
  case 0:
   DoMouseMove(p, keyflags);
   break;
  case PEN_DOWN:
   LeftDown(p, keyflags);
   break;
  case PEN_UP:
   LeftUp(p, keyflags);
   break;
 }
 return true;
}

void RectTracker::MouseMove(Point mp, dword){ // ADD NEW MouseMove here
 if(IsNull(pointer)) pointer=2;
 if(pointer!=2) return;
 DoMouseMove(mp,0);
}

void RectTracker::DoMouseMove(Point mp, dword) // RENAMED ORIGINAL MouseMove to
DoMouseMove
{

... THIS IS THE ORIGINAL MouseMove() code here ...
```

Changes in Win32Proc.cpp:
static bool pendown=false; // ADD
bool GetMouseLeft() { return pendown || !!(GetKeyStateSafe(VK_LBUTTON) & 0x8000); } // ADD 'pendown ||' here

...

```
   if(message == WM_POINTERUPDATE && GetPointerPenInfoHistory(pointerId, &hc, ppit)) {
    bool processed = false;
    for(int i = hc - 1; i >= 0; i--) {
     ProcessPenInfo(ppit[i]);
     POINT hp = ppit[i].pointerInfo.ptPixelLocation;
     CurrentMousePos =  hp; // << ADD UPDATING HERE
     ScreenToClient(hwnd, &hp);
     pen.history = (bool)i;
     processed = DoPen(hp);
    }
```

...

```
   switch(message) {
   case WM_POINTERDOWN:
    pendown=true; // << ADD set pendown here
    pen.action = PEN_DOWN;
    ClickActivateWnd();
    break;
   case WM_POINTERUP:
    pendown=false; // << ADD set pendown here
    pen.action = PEN_UP;
    break;
   }
```

The updated testcase including modified Sketcher is attached in main2.cpp.

Would this be acceptable, or do you have a more elegant solution?

Best regards,

Tom

File Attachments

1) main2.cpp, downloaded 175 times

Subject: Re: Using Pen with U++
Posted by Novo on Tue, 30 Mar 2021 21:34:50 GMT
View Forum Message <> Reply to Message

Upp build has been broken again. Most likely, this is not related to Pen stuff, but still ...
./umk reference SuggestCtrl CLANG -bus
/home/buildbot/worker/l-upp/build/reference/SuggestCtrl/main.cpp:54:21: error: address of overloaded function 'CharFilterToUpperAscii' is ambiguous
	ctrl.CompareFilter(CharFilterToUpperAscii);
	 ^~~~~~~~~~~~~~~~~~~~~~
/home/buildbot/worker/l-upp/build/reference/SuggestCtrl/main.cpp:5:12: note: candidate function
inline int CharFilterToUpperAscii(int c)
	 ^
/home/buildbot/worker/l-upp/build/uppsrc/Core/String.h:954:5: note: candidate function
int CharFilterToUpperAscii(int c);
    ^
/home/buildbot/worker/l-upp/build/uppsrc/CtrlLib/SuggestCtrl.h:39:35: note: passing argument to parameter 'filter' here
	SuggestCtrl& CompareFilter(int (*filter)(int c)) { compare_filter = filter; return *this; }
	 ^
1 error generated.

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 31 Mar 2021 08:37:33 GMT
View Forum Message <> Reply to Message

Mirek,

Something is preventing delivery of Pen() call to widgets inside a Splitter. Here's a testcase:

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

using namespace Upp;

struct Canvas: Ctrl {
 Point pos;

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

 PenInfo pen;

 Canvas(){
  pos=Null;
  Zero(pen);
 }

 virtual bool Pen(Point p, const PenInfo& pn, dword keyflags) override {
```

```
  pen=pn;

  switch(pn.action){
   case PEN_DOWN:
    drawing.Add().Add(MakeTuple(pn.pressure, p));
    break;
   case 0: // Move
    if(!!pn.pressure && !drawing.IsEmpty()) drawing.Top().Add(MakeTuple(pn.pressure, p));
   case PEN_UP: // Finish
    Refresh();
    break;
  }
  return true;
 }

 virtual void Paint(Draw& w0) override {
  DrawPainter w(w0, GetSize());
  w.Co();
  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, Black());
    }

  int fcy = GetStdFontCy();
  int y = 10;
  auto Text = [&] (const String& text) {
   w.DrawText(10, y, text);
   y += fcy;
  };
  Text(AsString(pos));
  Text(String() << "Pressure: " << pen.pressure);
  Text(String() << "Rotation: " << pen.rotation);
  Text(String() << "Tilt: " << pen.tilt);
  Text(String() << "Barrel: " << pen.barrel);
  Text(String() << "Inverted: " << pen.inverted);
  Text(String() << "Eraser: " << pen.eraser);
 }
};

struct MyApp : TopWindow {
 Canvas c1;
 Canvas c2;
```

```
 Splitter s;

 MyApp(){
  Add(s.Horz(c1,c2));
 }
};

GUI_APP_MAIN { MyApp().Run(); }
```

Best regards,

Tom

EDIT: Fixed testcase crash when drawing across the splitter wall.

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 31 Mar 2021 11:14:28 GMT
View Forum Message <> Reply to Message

Mirek,

How about this change in Ctrl::WindowProc() in Win32Proc.cpp?:

```
   auto DoPen = [&](Point p) {
    GuiLock __;
    eventCtrl = this;
    Ctrl *q = this;
    for(Ctrl *t = q; t; t=q->ChildFromPoint(p)) q = t;
    bool b = q->Pen(p, pen, GetMouseFlags());
    SyncCaret();
    return b;
   };
```

Now the for loop propagates the Pen to last child, and the Pen works in the above testcase.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 31 Mar 2021 11:46:58 GMT
View Forum Message <> Reply to Message

Hi,

Another issue: The mouse cursor in the above testcase does not get updated when using pen. If the cursor crosses the splitter wall, the wall -specific cursor will remain even when pen hovers on top of the drawing canvas. The cursor updating code must be absent in the DoPen function...

EDIT: How about this?

```
  auto DoPen = [&](Point p) {
   GuiLock __;
   eventCtrl = this;
   Ctrl *q = this;
   for(Ctrl *t = q; t; t=q->ChildFromPoint(p)) q = t;
   bool b = q->Pen(p, pen, GetMouseFlags());
   SyncCaret();
   Image m = CursorOverride();
   if(IsNull(m)) SetMouseCursor(q->CursorImage(p,GetMouseFlags()));
   else SetMouseCursor(m);
   return b;
  };
```

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 31 Mar 2021 13:03:25 GMT
View Forum Message <> Reply to Message

And one more issue shown with the testcase:

The Splitter wall cannot be moved with the pen. Again, the Splitter does not get capture until the pen has moved 2 cm, but unfortunately at that time the pen is no longer on top of the wall, but rather 2 cm away from it. If I drag back and cross the Splitter wall 2 or more centimeters away from the original starting point, it gets captured and the wall starts moving.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Thu, 01 Apr 2021 11:53:23 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Just noticed that SetCapture()/ReleaseCapture() was not working with Pen. Here's a fix for that:

```
  auto DoPen = [&](Point p) {
   GuiLock __;
   eventCtrl = this;
   Ctrl *q = this;
   if(captureCtrl){
    q=captureCtrl;
    p+=GetScreenRect().TopLeft()-captureCtrl->GetScreenRect().TopLeft();
   }
   else for(Ctrl *t = q; t; t=q->ChildFromPoint(p)) q = t;

   bool b = q->Pen(p, pen, GetMouseFlags());
   SyncCaret();
   Image m = CursorOverride();
   if(IsNull(m)) SetMouseCursor(q->CursorImage(p,GetMouseFlags()));
   else SetMouseCursor(m);
   return b;
  };
```

Best regards,

Tom

EDIT: Fixed coordinate offset in captureCtrl...

Subject: Re: Using Pen with U++
Posted by Tom1 on Thu, 01 Apr 2021 14:22:47 GMT
View Forum Message <> Reply to Message

Mirek,

OK, it seems the following (along with the previous changes) fixes Splitter behavior with pen.

Splitter.h:

```
class Splitter : public Ctrl {
 int           pointer;
public:
 virtual void   Layout();
 virtual void   Paint(Draw& draw);
 virtual void   DoMouseMove(Point p, dword keyflags);
 virtual void   MouseMove(Point p, dword keyflags);
 virtual bool   Pen(Point p, const PenInfo &pn, dword keyflags);
 virtual void   LeftDown(Point p, dword keyflags);
 virtual void   LeftUp(Point p, dword keyflags);
...
```

Splitter.cpp:

...

```
void   Splitter::MouseMove(Point p, dword) {
 if(pointer==2) DoMouseMove(p,0);
}

void   Splitter::DoMouseMove(Point p, dword) {
 if(HasCapture() && mouseindex >= 0 && mouseindex < pos.GetCount()) {
  SetPos(ClientToPos(p), mouseindex);
  Refresh();
  WhenAction();
 }
}

bool   Splitter::Pen(Point p, const PenInfo &pn, dword keyflags){
 switch(pn.action){
  case 0:
   if(pointer==1 && !pn.history) DoMouseMove(p, keyflags);
   break;
  case PEN_DOWN:
   if(IsNull(pointer)) pointer=1;
   LeftDown(p, keyflags);
   break;
  case PEN_UP:
   LeftUp(p, keyflags);
   break;
 }
 return true;
}

void   Splitter::LeftDown(Point p, dword) {
 if(IsNull(pointer)) pointer=2;
 SetCapture();
 Refresh();
 mouseindex = FindIndex(p);
}

int Splitter::FindIndex(Point client) const {
 int best = -1;
 int maxdist = chstyle->width;
 for(int i = 0; i < pos.GetCount(); i++) {
  int dist = abs((vert ? client.y : client.x) - PosToClient(pos[i]));
  if(dist <= maxdist) {
   best = i;
   maxdist = abs(dist);
  }
 }
 return best;
}
```

```
void   Splitter::LeftUp(Point p, dword keyflags) {
 pointer=Null;
 if(HasCapture())
  WhenSplitFinish();
 ReleaseCapture();
 Refresh();
}
...
Splitter::Splitter() {
 pointer = Null; // << Initialize here
 chstyle = NULL;
...
```

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Thu, 01 Apr 2021 15:29:53 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Thu, 01 April 2021 16:22Mirek,

OK, it seems the following (along with the previous changes) fixes Splitter behavior with pen.

Splitter.h:

```
class Splitter : public Ctrl {
 int           pointer;
public:
 virtual void   Layout();
 virtual void   Paint(Draw& draw);
 virtual void   DoMouseMove(Point p, dword keyflags);
 virtual void   MouseMove(Point p, dword keyflags);
 virtual bool   Pen(Point p, const PenInfo &pn, dword keyflags);
 virtual void   LeftDown(Point p, dword keyflags);
 virtual void   LeftUp(Point p, dword keyflags);
...
```

Splitter.cpp:

```
...
void   Splitter::MouseMove(Point p, dword) {
 if(pointer==2) DoMouseMove(p,0);
}

void   Splitter::DoMouseMove(Point p, dword) {
 if(HasCapture() && mouseindex >= 0 && mouseindex < pos.GetCount()) {
  SetPos(ClientToPos(p), mouseindex);
```

```
  Refresh();
  WhenAction();
 }
}

bool   Splitter::Pen(Point p, const PenInfo &pn, dword keyflags){
 switch(pn.action){
  case 0:
   if(pointer==1 && !pn.history) DoMouseMove(p, keyflags);
   break;
  case PEN_DOWN:
   if(IsNull(pointer)) pointer=1;
   LeftDown(p, keyflags);
   break;
  case PEN_UP:
   LeftUp(p, keyflags);
   break;
 }
 return true;
}

void   Splitter::LeftDown(Point p, dword) {
 if(IsNull(pointer)) pointer=2;
 SetCapture();
 Refresh();
 mouseindex = FindIndex(p);
}

int Splitter::FindIndex(Point client) const {
 int best = -1;
 int maxdist = chstyle->width;
 for(int i = 0; i < pos.GetCount(); i++) {
  int dist = abs((vert ? client.y : client.x) - PosToClient(pos[i]));
  if(dist <= maxdist) {
   best = i;
   maxdist = abs(dist);
  }
 }
 return best;
}

void   Splitter::LeftUp(Point p, dword keyflags) {
 pointer=Null;
 if(HasCapture())
  WhenSplitFinish();
 ReleaseCapture();
 Refresh();
}
```

...
Splitter::Splitter() {
 pointer = Null; // << Initialize here
 chstyle = NULL;
...

Best regards,

Tom

In other words, have we failed again? :) (The very moment you start fixing code in CtrlLib, it is failure...)

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Thu, 01 Apr 2021 16:01:53 GMT
View Forum Message <> Reply to Message

Well, I do not know... When using default (non-pen) processing, pen must be dragged 2 cm before WM_LBUTTONDOWN is first sent. :?

I do not know if this is the reason why the Splitter wall does not get captured with SetCapture().

So, I'm not sure at all, if this is our/my failure or if it is Wacom or Microsoft issue... After all, a mouse drag effectively begins only few pixels away from the starting point while for wacom it takes 2 cm.

Best regards,

Tom

EDIT: More specifically: When I put the pen down on standard splitter and start dragging, the splitter first gets nothing. As pen proceeds on top of the adjacent Canvas, the Canvas receives Pen() move actions with pressure on. When pen has moved 2 cm, I would expect the splitter to finally react, but in fact it requires pen passing over the splitter wall at a range greater than 2 cm from the starting point in order for the capture to realize.

EDIT2: The reason for not getting the WM_LBUTTONDOWN is obviously the lack of default processing when all the WM_POINTER* are in fact processed in Canvas and end with return 0L; instead. However, if we let it do default processing, we will start getting some WM_MOUSEMOVEs too from pen even when WM_POINTERUPDATEs have actually been processed already.

Maybe this must be fixed on the application side then: If Pen() returns false for such moves when pen is not actively being used in the Canvas i.e. a pendown condition therein, then Splitter can work almost normally despite the 2 cm start-up threshold. Obviously, the MouseMove()s must be

processed accordingly.

Subject: Re: Using Pen with U++
Posted by mirek on Thu, 01 Apr 2021 20:46:42 GMT
View Forum Message <> Reply to Message

If all works with "pre-Pen-U++" (?), maybe we should try to get back to "pen is mouse" mode (no Pen method). Return 0L in pen handling.

The only problem then is how to implement "IsPen" method (we definitely need that one). I can do that with XP-PEN (quoted that a while back), but we need a realiable method that works for Wacom too.

Mirek

Subject: Re: Using Pen with U++
Posted by mirek on Thu, 01 Apr 2021 20:49:28 GMT
View Forum Message <> Reply to Message

Are you 100% sure this does not work with Wacom:

https://stackoverflow.com/questions/29857587/detect-if-wm-mo usemove-is-caused-by-touch-pen

?

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 02 Apr 2021 06:26:07 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 01 April 2021 23:49Are you 100% sure this does not work with Wacom:

https://stackoverflow.com/questions/29857587/detect-if-wm-mo
usemove-is-caused-by-touch-pen

?

Last time I tried it did not work. It was zero all the time. Since then, I have reinstalled the Wacom drivers.

Now I get: "GetMessageExtraInfo() = 4283520772" for pen and "GetMessageExtraInfo() = 0" for mouse. These are present for e.g. WM_MOUSEMOVE and WM_LBUTTONDOWN.

So, I may have made some stupid mistake last time or the driver re-installation may have had an effect.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 02 Apr 2021 08:05:47 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 02 April 2021 08:26mirek wrote on Thu, 01 April 2021 23:49Are you 100% sure this does not work with Wacom:

https://stackoverflow.com/questions/29857587/detect-if-wm-mousemove-is-caused-by-touch-pen

?

Last time I tried it did not work. It was zero all the time. Since then, I have reinstalled the Wacom drivers.

Now I get: "GetMessageExtraInfo() = 4283520772" for pen and "GetMessageExtraInfo() = 0" for mouse. These are present for e.g. WM_MOUSEMOVE and WM_LBUTTONDOWN.

So, I may have made some stupid mistake last time or the driver re-installation may have had an effect.

Best regards,

Tom

OK, backup your changes, I am going to try yet another approach... :)

Mirek

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 02 Apr 2021 09:43:57 GMT
View Forum Message <> Reply to Message

Please try trunk reference/Pen now...

Notes: This is just a first rough attempt, if this works, it needs a cleanup and handling of history.

Subject: Re: Using Pen with U++

Posted by Tom1 on Fri, 02 Apr 2021 10:30:13 GMT
View Forum Message <> Reply to Message

Hi,

I'm currently out of town, but will be back home in the evening. I'll test it then ASAP.

Thanks and best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 02 Apr 2021 19:27:08 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 02 April 2021 12:43Please try trunk reference/Pen now...

Notes: This is just a first rough attempt, if this works, it needs a cleanup and handling of history.

Hi Mirek,

I just updated to latest SVN. All drags start 2 cm late. (Let's call this issue 'DragLag' for short from now on.) This is no wonder as WM_POINTERUPDATE does not send MouseMove() events to the app through any channel. Giving WM_POINTER* messages to the DefWindowProc*() to handle, causes DragLag before WM_LBUTTONDOWN and respective WM_MOUSEMOVE messages are generated. I have spent last two weeks on the issue and have not found any clean and easy way out with Wacom.

I understand that you don't have this DragLag -nightmare with XP-PEN and everything you do works with it. And that my attempts here are complex and ugly.

However, in order for Wacom to work with immediate response, we have to process everything from WM_POINTER* messages to application events (such as MouseMove(), LeftDown(), etc. or Pen() with proper PenInfo to decode the operations in the app.) If we also decide to break into DefWindowProc*() processing at the end of processing WM_POINTER* messages, we will need to block its resulting output to the Pen -enabled controls.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 03 Apr 2021 06:44:12 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 02 April 2021 21:27mirek wrote on Fri, 02 April 2021 12:43Please try trunk

reference/Pen now...

Notes: This is just a first rough attempt, if this works, it needs a cleanup and handling of history.

Hi Mirek,

I just updated to latest SVN. All drags start 2 cm late. (Let's call this issue 'DragLag' for short from now on.) This is no wonder as WM_POINTERUPDATE does not send MouseMove() events to the app through any channel. Giving WM_POINTER* messages to the DefWindowProc*() to handle, causes DragLag before WM_LBUTTONDOWN and respective WM_MOUSEMOVE messages are generated. I have spent last two weeks on the issue and have not found any clean and easy way out with Wacom.

What about other issues? I mean, does it work as mouse elsewhere?

Quote:
I understand that you don't have this DragLag -nightmare with XP-PEN and everything you do works with it. And that my attempts here are complex and ugly.

I think there is some small draglag.

Mirek

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 03 Apr 2021 06:52:04 GMT
View Forum Message <> Reply to Message

https://docs.microsoft.com/cs-cz/windows/win32/tablet/timeli
ne-of-mouse-messages-and-system-events

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 03 Apr 2021 06:57:44 GMT
View Forum Message <> Reply to Message

In general: I strongly believe that whatever model we choose in the end, it must not require changes in CtrlLib.

WRT to draglag, after reading MSDN docs, I have the impression that this is exactly what M$ wants to happen.

Anyway, a draglag related question - it is not quite clear to me whether the draglag which you experience is

a) drag starts late, after moving pen a bit, not immediately
b) drag has offset

(With XPPEN, with current model, drag just starts a little late, but position is correct)

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 03 Apr 2021 08:37:16 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 03 April 2021 09:57In general: I strongly believe that whatever model we choose in the end, it must not require changes in CtrlLib.

WRT to draglag, after reading MSDN docs, I have the impression that this is exactly what M$ wants to happen.

Anyway, a draglag related question - it is not quite clear to me whether the draglag which you experience is

a) drag starts late, after moving pen a bit, not immediately
b) drag has offset

(With XPPEN, with current model, drag just starts a little late, but position is correct)

Mirek

What I decided to call draglag is a), it starts 2 cm late on screen, but does not introduce any offset.

Now that I read your MSDN finding, I understand draglag is the decision lag for Windows Ink to figure out 'pen gestures'.

Quote:What about other issues? I mean, does it work as mouse elsewhere?
In principle almost everything works. Draglag is the only serious problem.

Other related peculiarities (Windows pen gestures?):
- LeftHold translates to RightDown+RightUp
- While tap with barrel pressed tranlates to RightDown+RightUp, a drag with barrel becomes a left drag (=LeftDown+LeftDrag+MouseMoves+LeftUp).

Also, my Sketcher no longer works, but that should be fixable here as RectTracker works.

So, in my opinion, we should get rid of DragLag and hopefully Windows pen gestures too, and receive just the same event behavior as is available with a mouse. This way CtrlLib (and apps) would not require any changes to support pen, but with GetPenInfo() we could benefit from the pen. I would suggest that a pressed barrel would change pen down to RightDown and so on.

That would be the most natural experience for a user. If we want to emulate Right click with pen hold, we would map LeftHold to RightUp in our app.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 03 Apr 2021 09:14:59 GMT
View Forum Message <> Reply to Message

What about

https://docs.microsoft.com/cs-cz/windows/win32/tablet/wm-tablet-querysystemgesturestatus-message

?

Especially the "press and hold" comment might be useful.

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 03 Apr 2021 09:37:03 GMT
View Forum Message <> Reply to Message

Adding:

```
case WM_TABLET_QUERYSYSTEMGESTURESTATUS:
  return TABLET_DISABLE_PRESSANDHOLD;
```

Removes the pen down/hold/up mapping to RightDown/RightUp and instead gives LeftDown/LeftUp. No LeftHold is generated during the prolonged pen-down though.

Unfortunately, this does not help at all with draglag.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 03 Apr 2021 13:52:56 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 03 April 2021 11:37Adding:

```
case WM_TABLET_QUERYSYSTEMGESTURESTATUS:
  return TABLET_DISABLE_PRESSANDHOLD;
```

Removes the pen down/hold/up mapping to RightDown/RightUp and instead gives LeftDown/LeftUp. No LeftHold is generated during the prolonged pen-down though.

Unfortunately, this does not help at all with draglag.

Best regards,

Tom

Another thing we can try is to retain Pen method, but discard bool return value, always process with DefWindowProc. Create new RectTracker just for Pen...

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 03 Apr 2021 14:02:42 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 03 April 2021 16:52Tom1 wrote on Sat, 03 April 2021 11:37Adding:

```
case WM_TABLET_QUERYSYSTEMGESTURESTATUS:
  return TABLET_DISABLE_PRESSANDHOLD;
```

Removes the pen down/hold/up mapping to RightDown/RightUp and instead gives LeftDown/LeftUp. No LeftHold is generated during the prolonged pen-down though.

Unfortunately, this does not help at all with draglag.

Best regards,

Tom

Another thing we can try is to retain Pen method, but discard bool return value, always process with DefWindowProc. Create new RectTracker just for Pen...

OK, let's see how it turns out. GetMessageExtraInfo() may be useful to classify events to application so that we can filter out already processed pen events and not do it again as if they are coming from the mouse.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 03 Apr 2021 14:05:33 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 03 April 2021 16:02mirek wrote on Sat, 03 April 2021 16:52Tom1 wrote on Sat, 03 April 2021 11:37Adding:
```
 case WM_TABLET_QUERYSYSTEMGESTURESTATUS:
   return TABLET_DISABLE_PRESSANDHOLD;
```

Removes the pen down/hold/up mapping to RightDown/RightUp and instead gives LeftDown/LeftUp. No LeftHold is generated during the prolonged pen-down though.

Unfortunately, this does not help at all with draglag.

Best regards,

Tom

Another thing we can try is to retain Pen method, but discard bool return value, always process with DefWindowProc. Create new RectTracker just for Pen...

OK, let's see how it turns out. GetMessageExtraInfo() may be useful to classify events to application so that we can filter out already processed pen events and not do it again as if they are coming from the mouse.

Exactly. Will you try that approach or should I do it? (I would prefer doing something else... :)

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 03 Apr 2021 14:17:15 GMT
View Forum Message <> Reply to Message

OK, I'll try, but I'm afraid it may fall back to you eventually...

As for the RectTracker, I would prefer it is a single RectTracker capable of reacting to both pen and mouse, so that I do not need to create separate Sketchers for each.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 03 Apr 2021 15:28:15 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 03 April 2021 16:17OK, I'll try, but I'm afraid it may fall back to you eventually...

As for the RectTracker, I would prefer it is a single RectTracker capable of reacting to both pen and mouse, so that I do not need to create separate Sketchers for each.

Best regards,

Tom

I can live with that.

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 03 Apr 2021 20:57:37 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Please find attached the changed CtrlCore files. Everything works now.

The widget specific pen support works on Pen() interface now. Returning true from Pen() avoids duplicates on mouse interface, (such as MouseMove(), LeftDown(), etc...) for pen events.

RectTracker works now with both mouse and pen.

There is also duplicate coordinate skipping optimization on pen history events.

Please test and merge.

Best regards,

Tom

File Attachments
1) CtrlCore.7z, downloaded 158 times

Subject: Re: Using Pen with U++
Posted by mirek on Sun, 04 Apr 2021 16:55:19 GMT
View Forum Message <> Reply to Message

Looks good (and applied), except "supports_pen". The problem I see is that we are not quite sure IMO that the current mouse message is translated pen message that has set it to true. Plus the

only thing it seems to do is to prevent the final dispatch of event to Ctrl - you can do that as well in the client code IMO.

Plus, it seems to fix the pointer image to Arrow (no CursorImage), which can be a problem too...

IMO I would try to get rid of it. If not possible, lets us be upfront and just make it Ctrl flag (not Pen return value).

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Sun, 04 Apr 2021 18:35:21 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thanks for accepting the changes.

It was a stupid mistake of me to use Image::Arrow() instead of CursorImage(). It should be in CtrlMouse.cpp(145) like:
```
 if(supports_pen && is_pen_event) return CursorImage(p, keyflags); // Avoid duplicated pen events
```

Although, after returning from every other event except 'case CURSORIMAGE:' in 'Image Ctrl::MouseEvent()' causes returning Image::Arrow()...

Without 'supports_pen' it would become a tedious battle in client code to avoid various pen originated duplicate mouse events, so I rather block them just before. Anyway, making 'bool supports_pen;' a configurable flag in Ctrl:: in CtrlCore.h is fine with me:
```
 Ctrl&   EnablePenSupport(bool b = true)                 { supports_pen = b; return *this; }
```

Then, to avoid Pen() return value assignment in Win32Proc.cpp(147), we should remove 'supports_pen =' and have just:
```
    q->Pen(p, pen, GetMouseFlags());
```

This works here. (I will of course add 'EnablePenSupport();' to constructors of Pen() enabled widgets.)

This includes RectTracker::RectTracker() in LocalLoop.cpp:

```
RectTracker::RectTracker(Ctrl& master)
{
 EnablePenSupport();
```

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sun, 04 Apr 2021 18:42:03 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sun, 04 April 2021 20:35Hi Mirek,

Thanks for accepting the changes.

It was a stupid mistake of me to use Image::Arrow() instead of CursorImage(). It should be in CtrlMouse.cpp(145) like:
 if(supports_pen && is_pen_event) return CursorImage(p, keyflags); // Avoid duplicated pen events

Without 'supports_pen' it would become a tedious battle in client code to avoid various pen originated duplicate mouse events, so I rather block them just before. Anyway, making 'bool supports_pen;' a configurable flag in Ctrl:: in CtrlCore.h is fine with me:

I understand that, but would that be really that hard? Esp. now that we have K_PEN?

Quote:
 Ctrl&   EnablePenSupport(bool b = true)   { supports_pen = b; return *this; }

Then, to avoid Pen() return value assignment in Win32Proc.cpp(147), we should remove 'supports_pen =' and have just:
    q->Pen(p, pen, GetMouseFlags());

This works here. (I will of course add 'EnablePenSupport();' to constructors of Pen() enabled widgets.)

This includes RectTracker::RectTracker() in LocalLoop.cpp:

RectTracker::RectTracker(Ctrl& master)
{
 EnablePenSupport();

Best regards,

Tom

Actually, maybe you could rather not call EnablePenSupport in RectTracker contructor and leave

that to client code?

But then, should not that prevent call of Pen virtual method too?

BTW, given what it does, should not that rather be called "DiasbleMouse"? :)

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Sun, 04 Apr 2021 20:02:00 GMT
View Forum Message <> Reply to Message

I agree that 'EnablePenSupport' is a bad name for it. 'DisableMouse' is not any better. In fact it should be something like 'DisableEmulatedMouseEventsForPen'.

But, as practically always, you are right: I can do everything related to 'supports_pen' flag at client side. I just add this to my Pen enabled widget:

```
 virtual Image MouseEvent(int event, Point p, int zdelta, dword keyflags) override {
  if(keyflags&K_PEN) return CursorImage(p, keyflags);
  return Ctrl::MouseEvent(event, p, zdelta, keyflags);
 }
```

In this case, I would drop everything in CtrlCore having something to do with 'supports_pen'.

So the last question is: Do you really want to get rid of 'supports_pen' flag? Both ways are fine with me.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sun, 04 Apr 2021 21:15:33 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sun, 04 April 2021 22:02I agree that 'EnablePenSupport' is a bad name for it. 'DisableMouse' is not any better. In fact it should be something like 'DisableEmulatedMouseEventsForPen'.

But, as practically always, you are right: I can do everything related to 'supports_pen' flag at client side. I just add this to my Pen enabled widget:

```
 virtual Image MouseEvent(int event, Point p, int zdelta, dword keyflags) override {
  if(keyflags&K_PEN) return CursorImage(p, keyflags);
  return Ctrl::MouseEvent(event, p, zdelta, keyflags);
 }
```

In this case, I would drop everything in CtrlCore having something to do with 'supports_pen'.

So the last question is: Do you really want to get rid of 'supports_pen' flag? Both ways are fine with me.

Best regards,

Tom

Occam's razor... :)

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 05 Apr 2021 10:33:20 GMT
View Forum Message <> Reply to Message

OK, this one is 'shaved'. :)

- 'supports_pen' flag is gone
- Pen interface is now 'virtual void Pen()'

Best regards,

Tom

EDIT: Nevermind... you did it already!

Thanks!

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 05 Apr 2021 10:59:40 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Next I will improve pen support in my app and if everything goes smoothly, then we need to look at the Linux side. (I see that K_PEN is missing and barrel is not detected, but there's no hurry with it right now.)

Thanks and best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 07 Apr 2021 09:04:44 GMT
View Forum Message <> Reply to Message

Mirek,

Otherwise it's going just fine, but we need to remove the following from WindowProc():
```
 case WM_TABLET_QUERYSYSTEMGESTURESTATUS:
   return TABLET_DISABLE_PRESSANDHOLD; // For clean press and hold behavior
```

The problem is that barrel to right mouse button mapping does not work correctly in non-pen enabled applications. (E.g. a barrel-tap on ArrayCtrl does not show associated context menus, but instead behaves as left click.)

EDIT: Another improvement to barrel -> right mouse button mapping is here in the same file Win32Proc.cpp line 57:

```
bool GetMouseRight()  { return Ctrl::GetPenInfo().barrel || !!(GetKeyStateSafe(VK_RBUTTON) & 0x8000); }
```

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 07 Apr 2021 10:00:21 GMT
View Forum Message <> Reply to Message

Mirek,

One more thing: My 3D view inherited from GLCtrl does not receive Pen() events at all.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Thu, 08 Apr 2021 15:36:35 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Here's a fix for missing barrel button detection in Linux (GTK). In GtkEvent.cpp Ctrl::AddEvent():
```
...
 e.event = NULL;
```

```
 GdkDevice *d = gdk_event_get_source_device(event);
 if(d && gdk_device_get_source(d) == GDK_SOURCE_PEN) {
  e.pen = true;
  e.pen_barrel = (bool)(MouseState & GDK_BUTTON3_MASK); // Add barrel button detection
...
```

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 09 Apr 2021 07:18:51 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 07 April 2021 11:04

bool GetMouseRight() { return Ctrl::GetPenInfo().barrel || !!(GetKeyStateSafe(VK_RBUTTON) & 0x8000); }

Should not there at least be some sort of Ctrl::IsPen() test mixed in?

(Other patches applied).

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 09 Apr 2021 08:00:16 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 09 April 2021 10:18Tom1 wrote on Wed, 07 April 2021 11:04

bool GetMouseRight() { return Ctrl::GetPenInfo().barrel || !!(GetKeyStateSafe(VK_RBUTTON) & 0x8000); }

Should not there at least be some sort of Ctrl::IsPen() test mixed in?

(Other patches applied).

Mirek

Well, I think not. As GetMouseRight() can be called from anywhere at any time, there is no forced association to specific pen or mouse event. So, we just get the info if right mouse button or barrel is pressed or not. I'm merely trying to make both mouse and pen behave the same way in apps and this helps towards this target. Obviously, we do not want mixed blinking response from GetMouseRight() depending on which pointing device has more recently updated its button status.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 09 Apr 2021 08:24:13 GMT
View Forum Message <> Reply to Message

OK then.

Not sure about what is wrong with GL. Could be related to DHCtrl...

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 09 Apr 2021 15:25:35 GMT
View Forum Message <> Reply to Message

Mirek,

Thanks for applying the changes. :)

When you have a moment to spare, could you please take a look at the pen history in Gtk? I guess that would complete pen support. (In addition to fixing Pen() for GLCtrl...)

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 09 Apr 2021 18:15:19 GMT
View Forum Message <> Reply to Message

Hi Mirek,

It seems routing Pen() to ctrl in GLPane fixes the Pen() in GLCtrl:

```
#ifdef PLATFORM_WIN32
 struct GLPane : DHCtrl {
```

```
  friend class GLCtrl;

  GLCtrl *ctrl;

  void DoGLPaint();

 public:
  GLPane() { NoWantFocus(); }

  virtual void    State(int reason);
  virtual LRESULT WindowProc(UINT message, WPARAM wParam, LPARAM lParam);
  virtual Image   MouseEvent(int event, Point p, int zdelta, dword keyflags);
  virtual void Pen(Point p, const PenInfo& pen, dword keyflags) { if(ctrl) ctrl->Pen(p, pen, keyflags);
} // ADD THIS FOR ROUTING PEN
  void Init();
  void Destroy();

  void ActivateContext();

  void ExecuteGL(HDC hdc, Event<> paint, bool swap_buffers);
  void ExecuteGL(Event<> paint, bool swap_buffers);
 };
#endif
```

However, I have a funny feeling this is not the full story. E.g. how does mouseTarget fit into this picture?

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sun, 11 Apr 2021 22:27:50 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 09 April 2021 20:15Hi Mirek,

It seems routing Pen() to ctrl in GLPane fixes the Pen() in GLCtrl:

```
#ifdef PLATFORM_WIN32
 struct GLPane : DHCtrl {
  friend class GLCtrl;

  GLCtrl *ctrl;

  void DoGLPaint();
```

```
 public:
  GLPane() { NoWantFocus(); }

  virtual void    State(int reason);
  virtual LRESULT WindowProc(UINT message, WPARAM wParam, LPARAM lParam);
  virtual Image   MouseEvent(int event, Point p, int zdelta, dword keyflags);
  virtual void Pen(Point p, const PenInfo& pen, dword keyflags) { if(ctrl) ctrl->Pen(p, pen, keyflags);
} // ADD THIS FOR ROUTING PEN
  void Init();
  void Destroy();

  void ActivateContext();

  void ExecuteGL(HDC hdc, Event<> paint, bool swap_buffers);
  void ExecuteGL(Event<> paint, bool swap_buffers);
 };
#endif
```

However, I have a funny feeling this is not the full story. E.g. how does mouseTarget fit into this picture?

Best regards,

Tom

Applied, hopefully including the mouseTarget issue.

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 12 Apr 2021 07:10:16 GMT
View Forum Message <> Reply to Message

Thanks! :)

BR,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sun, 25 Apr 2021 21:20:34 GMT
View Forum Message <> Reply to Message

It unfortunately looks like I am not getting any history with GTK. I think this should log them:

CtrlCore/GtkEvent.cpp:283

```
#if GTK_CHECK_VERSION(3, 22, 0)
 static guint32 prev_time;
 if(event) {
  GdkDevice *d = gdk_event_get_source_device(event);
  if(d && gdk_device_get_source(d) == GDK_SOURCE_PEN) {
   e.pen = true;
   e.pen_barrel = MouseState & GDK_BUTTON3_MASK;
   double *axes = NULL;
   if(event->type == GDK_MOTION_NOTIFY) {
    GdkEventMotion *mevent = (GdkEventMotion *)event;
    axes = mevent->axes;
    GdkTimeCoord **events;
    int n_events;
    if(gdk_device_get_history(d, mevent->window, prev_time, mevent->time, &events, &n_events))
{
     DLOG("=====");
     DDUMP(e.mousepos);
       for(int i = 0; i < n_events; i++) {
        double x = 0, y = 0, pressure = 0.5;
        gdk_device_get_axis (d, events[i]->axes, GDK_AXIS_X, &x);
        gdk_device_get_axis (d, events[i]->axes, GDK_AXIS_Y, &y);
        gdk_device_get_axis (d, events[i]->axes, GDK_AXIS_PRESSURE, &pressure);
        DDUMP(x);
        DDUMP(y);
        DDUMP(pressure);
     }
     gdk_device_free_history (events, n_events);
    }
    prev_time = gdk_event_get_time(event);
   }
   if(findarg(event->type, GDK_BUTTON_PRESS, GDK_2BUTTON_PRESS,
GDK_3BUTTON_PRESS, GDK_BUTTON_RELEASE) >= 0)
    axes = ((GdkEventButton *)event)->axes;
   if(axes) {
    double h;
    if(axes && gdk_device_get_axis(d, axes, GDK_AXIS_PRESSURE, &h))
     e.pen_pressure = h;
    if(axes && gdk_device_get_axis(d, axes, GDK_AXIS_ROTATION, &h))
     e.pen_rotation = h;
    if(axes && gdk_device_get_axis(d, axes, GDK_AXIS_XTILT, &h))
     e.pen_tilt.x = h;
    if(axes && gdk_device_get_axis(d, axes, GDK_AXIS_YTILT, &h))
     e.pen_tilt.y = h;
   }
```

```
 }
}
#endif
```

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 26 Apr 2021 13:55:15 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Unfortunately, it is the same here. No history: gdk_device_get_history() returns false and that's it. :(

How to try if this helps:

gdk_window_set_event_compression(win, false);

I mean where to put it and which win to use?

https://developer.gnome.org/gdk3/stable/gdk3-Windows.html#gdk-window-set-event-compression

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Mon, 26 Apr 2021 14:06:48 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 26 April 2021 15:55Hi Mirek,

Unfortunately, it is the same here. No history: gdk_device_get_history() returns false and that's it. :(

How to try if this helps:

gdk_window_set_event_compression(win, false);

I already tried as the first option, makes everything stop working (not sure why, tried gdk_device_get_history after that).

Good position to insert it is somewhere around CtrlCore/GtkCreate.cpp:59.

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 26 Apr 2021 14:33:10 GMT
View Forum Message <> Reply to Message

Hi,

The gdk_window_set_event_compression() does not help here either. Drawing is not working when using that either.

I tested Wacom pen with GIMP and it works as fast as I can possibly draw. (And it is not smoothing my strokes either, because a stupid super fast zig-zag looks just like that.) So there must be something we are missing here...

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 26 Apr 2021 15:13:44 GMT
View Forum Message <> Reply to Message

Mirek,

Does GIMP source file gimp-master/app/display/gimpdisplayshell-tool-events.c (possibly around line 860) give you any hint what they could be doing differently with gdk_device_get_history()?

Also on line 553 they seem to activate gdk_window_set_event_compression(w, FALSE); when the button goes down and then inactivate again when button is released.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 28 Apr 2021 10:00:03 GMT

Hi Mirek,

More bad news: It seems that on Ubuntu 2021.4 U++ Pen does not work. Or rather, it works in mouse emulation without any pen specific features.

In Ubuntu settings there is a page for Wacom and there the tablet and pen is configurable and it works (including pressure) in the test-canvas provided by the settings page.

The pen also works in Krita with pressure and all. (I do not know if Krita works on top of Wayland directly or via Xwayland.)

In GIMP, the pen also works correctly. Therein it is configured through Edit > Input Devices. It seems to come in via "xwayland-tablet stylus" device on the list.

Best regards,

Tom

EDIT: In Ubuntu 2021.4 pen is reported as source device called GDK_SOURCE_TOUCHSCREEN instead of GDK_SOURCE_PEN. Pressure can be detected here.

Subject: Re: Using Pen with U++
Posted by mirek on Wed, 28 Apr 2021 13:42:54 GMT

Tom1 wrote on Wed, 28 April 2021 12:00Hi Mirek,

More bad news: It seems that on Ubuntu 2021.4 U++ Pen does not work. Or rather, it works in mouse emulation without any pen specific features.

In Ubuntu settings there is a page for Wacom and there the tablet and pen is configurable and it works (including pressure) in the test-canvas provided by the settings page.

The pen also works in Krita with pressure and all. (I do not know if Krita works on top of Wayland directly or via Xwayland.)

In GIMP, the pen also works correctly. Therein it is configured through Edit > Input Devices. It seems to come in via "xwayland-tablet stylus" device on the list.

Best regards,

Tom

EDIT: In Ubuntu 2021.4 pen is reported as source device called

GDK_SOURCE_TOUCHSCREEN instead of GDK_SOURCE_PEN. Pressure can be detected here.

I suggest ignoring this for 2021.1 release... Hopefully we shall figure that out for the next one.

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 28 Apr 2021 14:39:18 GMT
View Forum Message <> Reply to Message

Mirek,

OK, I can live without history in GTK for now.

Do you think it would still be better to include the GDK_SOURCE_TOUCHSCREEN option to support pen on Ubuntu, although without history processing? Or maybe just announce pen support for Windows at this time?

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Fri, 30 Apr 2021 05:32:08 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 28 April 2021 16:39Mirek,

OK, I can live without history in GTK for now.

Do you think it would still be better to include the GDK_SOURCE_TOUCHSCREEN option to support pen on Ubuntu, although without history processing? Or maybe just announce pen support for Windows at this time?

Best regards,

Tom

No reason not to add GDK_SOURCE_TOUCHSCREEN... Can you send the tested code?

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 30 Apr 2021 09:08:30 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Before applying the below code, please consider the following: Although this makes it work on Ubuntu 2021.4 in addition to Linux Mint 20.1 with Wacom, it seems like accepting a bug in Ubuntu and adapting to it. (Pretty much like what we have to do in Windows normally...)

I also tried to address the missing tilt information here by reading gdk_device_get_axes() and behaving accordingly, but that does not seem to work either. Tilt remains constantly on both axis at 0.00787... Is it the same with your XPPEN tablet?

```
#if GTK_CHECK_VERSION(3, 22, 0)
 GdkDevice *d = gdk_event_get_source_device(event);
 int s=d?gdk_device_get_source(d):0;
 if(d && ((s==GDK_SOURCE_PEN) || (s==GDK_SOURCE_TOUCHSCREEN))) {
  e.pen = true;
  e.pen_barrel = MouseState & GDK_BUTTON3_MASK;
  double *axes = NULL;
  if(event->type == GDK_MOTION_NOTIFY)
   axes = ((GdkEventMotion *)event)->axes;
  if(findarg(event->type, GDK_BUTTON_PRESS, GDK_2BUTTON_PRESS,
GDK_3BUTTON_PRESS, GDK_BUTTON_RELEASE) >= 0)
   axes = ((GdkEventButton *)event)->axes;
  if(axes) {
   GdkAxisFlags flags = gdk_device_get_axes (d);
   if(flags & GDK_AXIS_FLAG_PRESSURE) gdk_device_get_axis(d, axes,
GDK_AXIS_PRESSURE, &e.pen_pressure);
   if(flags & GDK_AXIS_FLAG_ROTATION) gdk_device_get_axis(d, axes,
GDK_AXIS_ROTATION, &e.pen_rotation);
   if(flags & GDK_AXIS_FLAG_XTILT) gdk_device_get_axis(d, axes, GDK_AXIS_XTILT,
&e.pen_tilt.x);
   if(flags & GDK_AXIS_FLAG_YTILT) gdk_device_get_axis(d, axes, GDK_AXIS_YTILT,
&e.pen_tilt.y);
  }
 }
#endif
```

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 30 Apr 2021 15:08:56 GMT

Mirek,

This might only apply after release 2021.1, but I found a solution for history -- or rather the backlog of mouse events:

In GtkCreate.cpp on line 44 we should have:
```
 gtk_widget_set_events(top->window, GDK_ALL_EVENTS_MASK &
~GDK_POINTER_MOTION_HINT_MASK);
```
This drops the deprecated event mask for reducing the number of GDK_MOTION_NOTIFY events.

Additionally in GtkEvent.cpp we need:
```
...
static Point s_mousepos;

Point Ctrl::GetMouseInfo(GdkWindow *win, GdkModifierType& mod)
{
#if GTK_CHECK_VERSION(3, 20, 0)
 GdkDisplay *display = gdk_window_get_display (win);
 GdkDevice *pointer = gdk_seat_get_pointer (gdk_display_get_default_seat (display));
 double x, y;
 gdk_window_get_device_position_double (win, pointer, &x, &y, &mod);
 return s_mousepos; //return Point((int)SCL(x), (int)SCL(y));
#else
 gint x, y;
 gdk_window_get_pointer(win, &x, &y, &mod);
 return Point(SCL(x), SCL(y));
#endif
}

void Ctrl::AddEvent(gpointer user_data, int type, const Value& value, GdkEvent *event)
{
 if(Events.GetCount() > 50000)
  return;
 GEvent& e = Events.AddTail();
 e.windowid = (uint32)(uintptr_t)user_data;
 e.type = type;
 e.value = value;
 GdkModifierType mod;
 e.mousepos = GetMouseInfo(gdk_get_default_root_window(), mod);
 if(event->type == GDK_MOTION_NOTIFY){
  GdkEventMotion *mevent = (GdkEventMotion *)event;
  e.mousepos = s_mousepos = Point(SCL(mevent->x_root), SCL(mevent->y_root));
 }
 e.state = (mod & ~(GDK_BUTTON1_MASK|GDK_BUTTON2_MASK|GDK_BUTTON3_MASK)) |
MouseState;
 e.count = 1;
```

```
 e.event = NULL;
#if GTK_CHECK_VERSION(3, 22, 0)
 GdkDevice *d = gdk_event_get_source_device(event);
 int s=d?gdk_device_get_source(d):0;
 if(d && ((s==GDK_SOURCE_PEN) || (s==GDK_SOURCE_TOUCHSCREEN))) {
  e.pen = true;
  e.pen_barrel = MouseState & GDK_BUTTON3_MASK;
  double *axes = NULL;
  switch(event->type){
   case GDK_BUTTON_PRESS:
    gdk_window_set_event_compression(((GdkEventButton *)event)->window, false);
   case GDK_2BUTTON_PRESS:
   case GDK_3BUTTON_PRESS:
    axes = ((GdkEventButton *)event)->axes;
    break;
   case GDK_BUTTON_RELEASE:
    gdk_window_set_event_compression(((GdkEventButton *)event)->window, true);
    axes = ((GdkEventButton *)event)->axes;
    break;
   case GDK_MOTION_NOTIFY:{
    GdkEventMotion *mevent = (GdkEventMotion *)event;
    e.mousepos = s_mousepos = Point(SCL(mevent->x_root), SCL(mevent->y_root));
    axes = ((GdkEventMotion *)event)->axes;
    break;
   }
  }
  if(axes) {
   if(!gdk_device_get_axis(d, axes, GDK_AXIS_PRESSURE, &e.pen_pressure))
e.pen_pressure=Null;
   if(!gdk_device_get_axis(d, axes, GDK_AXIS_ROTATION, &e.pen_rotation))
e.pen_rotation=Null;
   if(!gdk_device_get_axis(d, axes, GDK_AXIS_XTILT, &e.pen_tilt.x)) e.pen_tilt.x=Null;
   if(!gdk_device_get_axis(d, axes, GDK_AXIS_YTILT, &e.pen_tilt.y)) e.pen_tilt.y=Null;
  }
 }
#endif
...
And finally also in GtkEvent.cpp:
...

bool Ctrl::ProcessEvent0(bool *quit, bool fetch)
{
 ASSERT(IsMainThread());
 bool r = false;
 if(IsWaitingEvent0(fetch)) {
  while(Events.GetCount() > 1) { // GEvent compression (coalesce autorepeat, mouse
moves/wheel, configure)
   GEvent& a = Events[0];
```

```
   GEvent& b = Events[1];
   if(b.type == a.type && a.windowid == b.windowid && a.state == b.state) {
    if(a.type == GDK_KEY_PRESS && a.value == b.value)
     b.count += a.count;
       else
       if(a.type == GDK_SCROLL)
           b.value = (int)b.value + (int)a.value;
    else
//    if(findarg(a.type, GDK_MOTION_NOTIFY, GDK_CONFIGURE) < 0)
    if(findarg(a.type, GDK_CONFIGURE) < 0) // *** HERE: We cannot drop
GDK_MOTION_NOTIFY events
     break;
    Events.DropHead();
```

...
This last one dropped queued GDK_MOTION_NOTIFY events and ruined all attempts to get them through.

However, performance questions may need further considerations... It works here in virtual machine quite well on both Ubuntu 2021.4 and Linux Mint 20.1, but hardware varies of course.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 01 May 2021 08:16:11 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 30 April 2021 11:08Hi Mirek,

Before applying the below code, please consider the following: Although this makes it work on Ubuntu 2021.4 in addition to Linux Mint 20.1 with Wacom, it seems like accepting a bug in Ubuntu and adapting to it. (Pretty much like what we have to do in Windows normally...)

I also tried to address the missing tilt information here by reading gdk_device_get_axes() and behaving accordingly, but that does not seem to work either. Tilt remains constantly on both axis at 0.00787... Is it the same with your XPPEN tablet?

```
#if GTK_CHECK_VERSION(3, 22, 0)
 GdkDevice *d = gdk_event_get_source_device(event);
 int s=d?gdk_device_get_source(d):0;
 if(d && ((s==GDK_SOURCE_PEN) || (s==GDK_SOURCE_TOUCHSCREEN))) {
  e.pen = true;
  e.pen_barrel = MouseState & GDK_BUTTON3_MASK;
```

```
  double *axes = NULL;
  if(event->type == GDK_MOTION_NOTIFY)
   axes = ((GdkEventMotion *)event)->axes;
  if(findarg(event->type, GDK_BUTTON_PRESS, GDK_2BUTTON_PRESS,
GDK_3BUTTON_PRESS, GDK_BUTTON_RELEASE) >= 0)
   axes = ((GdkEventButton *)event)->axes;
  if(axes) {
   GdkAxisFlags flags = gdk_device_get_axes (d);
   if(flags & GDK_AXIS_FLAG_PRESSURE) gdk_device_get_axis(d, axes,
GDK_AXIS_PRESSURE, &e.pen_pressure);
   if(flags & GDK_AXIS_FLAG_ROTATION) gdk_device_get_axis(d, axes,
GDK_AXIS_ROTATION, &e.pen_rotation);
   if(flags & GDK_AXIS_FLAG_XTILT) gdk_device_get_axis(d, axes, GDK_AXIS_XTILT,
&e.pen_tilt.x);
   if(flags & GDK_AXIS_FLAG_YTILT) gdk_device_get_axis(d, axes, GDK_AXIS_YTILT,
&e.pen_tilt.y);
  }
 }
#endif
```

Best regards,

Tom

OK, applied with cosmetics. Hopefully the last change for 2021.1....

Mirek

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 01 May 2021 08:19:05 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 30 April 2021 17:08Mirek,

This might only apply after release 2021.1, but I found a solution for history -- or rather the backlog of mouse events:

```
//    if(findarg(a.type, GDK_MOTION_NOTIFY, GDK_CONFIGURE) < 0)
    if(findarg(a.type, GDK_CONFIGURE) < 0) // *** HERE: We cannot drop
GDK_MOTION_NOTIFY events
     break;
    Events.DropHead();
```

...

This last one dropped queued GDK_MOTION_NOTIFY events and ruined all attempts to get them through.

I am not really comfortable with not compressing motion events for MouseMove - there is a reason for that... I think we should limit this to just Pen method somehow... Either way, not in 2021.1

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 01 May 2021 14:56:46 GMT
View Forum Message <> Reply to Message

Hi,

Thanks, Mirek. I will test the applied changes on Monday at the office then.

For the future, i.e. post 2021.1: As for the pen history processing, should we consider some mechanism that would provide full motion history as a Vector<> of pen data. Then callbacks would be compressed, but the target Ctrl would decide if it wants to process the full history Vector or just the latest observation. This would scale well in my opinion, but still easily provide full pen trace.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Sat, 01 May 2021 15:57:11 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 01 May 2021 16:56Hi,

Thanks, Mirek. I will test the applied changes on Monday at the office then.

For the future, i.e. post 2021.1: As for the pen history processing, should we consider some mechanism that would provide full motion history as a Vector<> of pen data. Then callbacks would be compressed, but the target Ctrl would decide if it wants to process the full history Vector or just the latest observation. This would scale well in my opinion, but still easily provide full pen trace.

Best regards,

Tom

I think all is OK as long as those "overflow" events are delivered only to Pen. We already have history flag, do not we?

I am just afraid that doing that for all MouseMoves could overhelm some widgets that do not account for this....

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Sat, 01 May 2021 20:56:49 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 01 May 2021 18:57
I think all is OK as long as those "overflow" events are delivered only to Pen. We already have history flag, do not we?

I am just afraid that doing that for all MouseMoves could overhelm some widgets that do not account for this....

Mirek

Actually, I'm not so sure about overwhelming anything. Even when we do:
gtk_widget_set_events(top->window, GDK_ALL_EVENTS_MASK &
~GDK_POINTER_MOTION_HINT_MASK);
to disable deprecated motion compression, there will be modern default motion compression enabled, which is only disabled for full data by:
gdk_window_set_event_compression(((GdkEventButton *)event)->window, false);
And this I only used for pen and touch when some button is down in the above code. Not for mouse at all, so it should really be compressed in GDK. However, what is missing is that the history flag should be set in some smart way to improve efficiency of widgets that do not need this much detail from pen. In my opinion when the pen is in mouse emulation and not using Pen() interface, it should be compressed for best response from widgets not prepared for such stream of moves. One critical example would be a Splitter, which can cause a terrible load from refreshing complex child widgets.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Mon, 03 May 2021 07:18:09 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 01 May 2021 11:16

OK, applied with cosmetics. Hopefully the last change for 2021.1....

Mirek

Hi Mirek,

Confirmed: It works on Ubuntu 2021.4 and Linux Mint 20.1 with Wacom.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Fri, 14 May 2021 07:04:49 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Now that the release is happily completed, how shall we continue with pen history processing in GTK? (I think we should get over with it before the details fade away.)

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Wed, 02 Jun 2021 08:53:13 GMT
View Forum Message <> Reply to Message

After checking the code, looks reasonable enough, applied.

That said, during testing I had performance issue, sometimes turning the whole process into slide-show, however it seems like these need to be fixed on more fundamental level of GTK backend....

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 02 Jun 2021 09:28:47 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 02 June 2021 11:53After checking the code, looks reasonable enough, applied.

That said, during testing I had performance issue, sometimes turning the whole process into slide-show, however it seems like these need to be fixed on more fundamental level of GTK backend....

Hi Mirek,

Thanks! This seems to work fine here. Also with splitter.

However, I do have one more wish: Is it possible to get the mouse cursor to follow the pen? As you may have noticed, the pen does not currently have any cursor at all in Linux. Preferably, it should work the same way as in Windows.

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Wed, 02 Jun 2021 09:38:01 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 02 June 2021 11:28
However, I do have one more wish: Is it possible to get the mouse cursor to follow the pen? As you may have noticed, the pen does not currently have any cursor at all in Linux. Preferably, it should work the same way as in Windows.

I am not sure what you mean here. For me, it seems to work the same way as in Windows.

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 02 Jun 2021 09:40:42 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 02 June 2021 12:38
I am not sure what you mean here. For me, it seems to work the same way as in Windows.

Unfortunately not in my Wacom case... Here I can move the real mouse around the screen and simultaneously draw with pen on a different location. The cursor follows the real mouse and the pen does not have any cursor at all.

BR,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Wed, 02 Jun 2021 09:45:42 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 02 June 2021 11:40mirek wrote on Wed, 02 June 2021 12:38
I am not sure what you mean here. For me, it seems to work the same way as in Windows.

Unfortunately not in my Wacom case... Here I can move the real mouse around the screen and simultaneously draw with pen on a different location. The cursor follows the real mouse and the pen does not have any cursor at all.

BR,

Tom

Maybe some system setting?

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 02 Jun 2021 10:00:48 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 02 June 2021 12:45
Maybe some system setting?
Probably. I just realized that cannot see a cursor for pen anywhere on the desktop. They are just two completely separated pointing devices and only mouse does have the cursor. I need to do some googling here...

Thanks and best regards,

Tom

Subject: Re: Using Pen with U++
Posted by Tom1 on Wed, 02 Jun 2021 10:50:46 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Yes, it was a kind of setting: Works on VMware but not on VirtualBox. (I did not bother to read through the entire web to find out if anybody got it working on VirtualBox.)

Finally, I tested on real hardware and it works there fine. Case closed. :)

Thanks and best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Thu, 03 Jun 2021 08:05:44 GMT
View Forum Message <> Reply to Message

I am afraid that something got broken with this patch. Yesterday I have only tested with Pen example, but now testing with UWord I get immediate crash when trying to type in something. Actually, I am getting the crash with Pen too as soon as I hit any key.. The crash seems to happen inside gtk.

Can you reproduce this?

Subject: Re: Using Pen with U++
Posted by mirek on Thu, 03 Jun 2021 08:16:39 GMT
View Forum Message <> Reply to Message

Found it... Not that big problem (event in AddEvent can be NULL, so your initial if needs event && ...).

Mirek

Subject: Re: Using Pen with U++
Posted by Tom1 on Thu, 03 Jun 2021 08:33:30 GMT
View Forum Message <> Reply to Message

Hi,

It was the same here... :)

Best regards,

Tom

Subject: Re: Using Pen with U++
Posted by mirek on Thu, 03 Jun 2021 09:26:14 GMT
View Forum Message <> Reply to Message

Now that I have fixed/tuned the rendering with gtk (separate annoucement), I am basically getting the same (or maybe even better) performance/snappines of Pen as in Win32. So I guess we can consider this whole issue finally resolved (until somebody suggest to implement it for macos... :)

Mirek