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...
