Subject: Re: Extensions to Draw...Ops
Posted by Tom1 on Thu, 27 Nov 2008 15:34:51 GMT
View Forum Message <> Reply to Message

Unfortunately, I can't inject line style changes inside e.g. Draw::DrawPolyPolyLineOp() and others by using this BeginGDI/EndGDI mechanism, since the line style is defined within the function just before drawing the line. I would really have to make changes inside the Draw implementation for this one to work.

This is how I do the stuff currently in connection with U++:

```
void My_DrawExtensions::SetLineProperties(int width,int style,const RGBA_T &color){
 if((linewidth==width)&&(linestyle==style)&&(linecolor==color)) return;
#ifdef PLATFORM_WIN32
 int ps=0;
 switch(style){
  case GS_LINESTYLE_NULL:
   ps=PS_NULL;
   break;
  default:
  case GS_LINESTYLE_SOLID:
   ps=PS_SOLID;
   break;
  case GS_LINESTYLE_DOT:
   ps=PS_DOT;
   break;
  case GS_LINESTYLE_DASH:
   ps=PS_DASH;
   break;
  case GS_LINESTYLE_DASHDOT:
   ps=PS_DASHDOT;
   break;
  case GS_LINESTYLE_DASHDOTDOT:
   ps=PS_DASHDOTDOT;
   break;
 }

 HPEN pen=0;
 if((ps==PS_SOLID)||(ps==PS_NULL)){

  if((width>=2)&&(ps==PS_SOLID)){
   LOGBRUSH lb;

   lb.lbStyle=BS_SOLID;
   lb.lbColor=RGB(color.r,color.g,color.b);
   lb.lbHatch=0;

   pen=ExtCreatePen(PS_GEOMETRIC|PS_ENDCAP_FLAT|PS_JOIN_BEVEL|PS_SOLID,width,
```

```
&lb,0,0);
  }
  else pen=CreatePen(ps,width<0?0:width, RGB(color.r,color.g,color.b));
 }
 else{
  LOGBRUSH lb;

  lb.lbStyle=BS_SOLID;
  lb.lbColor=RGB(color.r,color.g,color.b);
  lb.lbHatch=0;

  INT32 w=width<=1?0:width;
  int ending=PS_ENDCAP_FLAT|PS_JOIN_BEVEL;

  switch(style){
   case GS_LINESTYLE_DOT:
    pen=ExtCreatePen(PS_GEOMETRIC|PS_USERSTYLE|ending,w,&lb,2,L_DOT);
    break;
   case GS_LINESTYLE_DASH:
    pen=ExtCreatePen(PS_GEOMETRIC|PS_USERSTYLE|ending,w,&lb,2,L_DASH);
    break;
   case GS_LINESTYLE_DASHDOT:
    pen=ExtCreatePen(PS_GEOMETRIC|PS_USERSTYLE|ending,w,&lb,4,L_DASHDOT);
    break;
   case GS_LINESTYLE_DASHDOTDOT:
    pen=ExtCreatePen(PS_GEOMETRIC|PS_USERSTYLE|ending,w,&lb,6,L_DASHDOTDOT);
    break;
   default:
    pen=ExtCreatePen(PS_GEOMETRIC|PS_SOLID|ending,w,&lb,0,0);
    break;
  }
 }
 if(pen){
  HPEN h = (HPEN) SelectObject(hdc, pen);
  DeleteObject(h);
  linestyle=style;
  linewidth=width;
  linecolor=color;
 }
#else // X11 way
 if((style!=linestyle)||(width!=linewidth)){
  static const char dot[] = { 3, 3 };
  static const char dash[] = { 18, 6 };
  static const char dashdot[] = { 9, 6, 3, 6 };
  static const char dashdotdot[] = { 9, 3, 3, 3, 3, 3 };
  static struct {
   const char *dash;
   int   len;
```

```
  } ds[] = {
   dot, __countof(dot),
   dash, __countof(dash),
   dashdot, __countof(dashdot),
   dashdotdot, __countof(dashdotdot)
  };
  switch(style){
   case GS_LINESTYLE_SOLID:
    XSetLineAttributes(Xdisplay, gc, width<0?0:width,LineSolid,CapNotLast,JoinBevel);
    break;
   case GS_LINESTYLE_DOT:
    XSetDashes(Xdisplay, gc, 0, ds[0].dash, ds[0].len);
    XSetLineAttributes(Xdisplay, gc, width<0?0:width,LineOnOffDash,CapNotLast,JoinBevel);
    break;
   case GS_LINESTYLE_DASH:
    XSetDashes(Xdisplay, gc, 0, ds[1].dash, ds[1].len);
    XSetLineAttributes(Xdisplay, gc, width<0?0:width,LineOnOffDash,CapNotLast,JoinBevel);
    break;
   case GS_LINESTYLE_DASHDOT:
    XSetDashes(Xdisplay, gc, 0, ds[2].dash, ds[2].len);
    XSetLineAttributes(Xdisplay, gc, width<0?0:width,LineOnOffDash,CapNotLast,JoinBevel);
    break;
   case GS_LINESTYLE_DASHDOTDOT:
    XSetDashes(Xdisplay, gc, 0, ds[3].dash, ds[3].len);
    XSetLineAttributes(Xdisplay, gc, width<0?0:width,LineOnOffDash,CapNotLast,JoinBevel);
    break;
  }
  linestyle=style;
  linewidth=width;
 }
 linecolor=color;
#endif
}
```

I wish I could get at least this functionality embedded in Draw:: along with the extended Draw...Ops supporting the style parameter.

--

Otherwise it is nice to have the BeginGDI and EndGDI available for Windows. However, I sort of care about X11 too, so are there equivalents for BeginGDI and EndGDI for X11? (I can't seem to find them.)

Is it safe to call Draw... functions too between BeginGDI and EndGDI calls, or does this break everything? If it breaks, this effectively requires all drawing functions to be rewritten instead of adding just the ones needed for added functionality.

--

I noticed luckily that you have XOR operations covered for at least Polyline and Polygon primitives, so this should be enough for now on the ROP front.

// Tom
