Subject: Extensions to Draw...Ops
Posted by Tom1 on Mon, 24 Nov 2008 13:51:06 GMT
View Forum Message <> Reply to Message

Hi,

What are the chances to get Draw...Op family of functions extended with the following?:

1) Possibility to use different line styles for all line widths. This of course requires separate parameters for line width and line style. Support for geometric pens required for implementation of this feature is available in Windows since NT. Geometric pens tend to be slower than the standard pens, but the performance suffers only in the specific case when wide lines are drawn with styling. Otherwise geometric pens are not used.

2) Uniform raster operation (ROP) capability for all Draw functions. This works for a single DrawOp with ROP mode parameter and for a sequence of DrawOps with preset ROP mode. This is very useful for implementing various complex rubber band tools required in CAD and similar drawing applications. It saves a lot of time spent in redrawing when editing a drawing object on top of complex drawings.

--

I have already written working code for this (both Win32 and X11) as a separate class just querying context from Draw class. So, before starting the work to merge the code, I wish to know what are the chances to get it included in Ultimate++.

// Tom

Subject: Re: Extensions to Draw...Ops
Posted by mirek on Tue, 25 Nov 2008 22:37:03 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 24 November 2008 08:51Hi,

What are the chances to get Draw...Op family of functions extended with the following?:

1) Possibility to use different line styles for all line widths. This of course requires separate parameters for line width and line style. Support for geometric pens required for implementation of this feature is available in Windows since NT. Geometric pens tend to be slower than the standard pens, but the performance suffers only in the specific case when wide lines are drawn with styling. Otherwise geometric pens are not used.

2) Uniform raster operation (ROP) capability for all Draw functions. This works for a single DrawOp with ROP mode parameter and for a sequence of DrawOps with preset ROP mode. This is very useful for implementing various complex rubber band tools required in CAD and similar drawing applications. It saves a lot of time spent in redrawing when editing a drawing object on top of complex drawings.

--

I have already written working code for this (both Win32 and X11) as a separate class just querying context from Draw class. So, before starting the work to merge the code, I wish to know what are the chances to get it included in Ultimate++.

// Tom

Well, the problem is cross-platform compatibility. That is why Draw is as minimal as possible.

We plan to solve the problem using sofware rendering, mostli like with AGG 2.4.

My idea is that we should aim for drawing capabilities level that makes possible rendering of SVG images (which we should support as well) and PDFs.

The planned bonus is that we might be able to have X11-less (or GDI-less) drawing capability - a very good thing for web servers.

BTW, if you would like to get involved, you are welcome. There were already some attempts in the past, so no need to start from scratch.

Mirek

Subject: Re: Extensions to Draw...Ops
Posted by Tom1 on Wed, 26 Nov 2008 13:53:40 GMT
View Forum Message <> Reply to Message

For the problem in cross-platform compatibility, which platforms are you specifically referring to?

Generally, I'm all for a U++ licensed software 2D graphics renderer -- i.e. ImageDraw without GDI or X11 dependencies. Thanks for the invitation, but at this time I'm too busy for such a large project. And additionally, it would not solve my specific problem.

The operations I'm referring to are and should be implemented in the Draw->GDI and Draw->X11 interfaces. (By the way, are there other low-level interfaces that should be supported?)

My view of the U++ 2D graphics sub-system in general is that it provides (and should also provide in future) one clean and simple Draw API with basic capabilities that are directly mapped to primitives on GDI and X11. This is vitally important for performance. (I'm seriously counting milliseconds spent for rendering the window contents after each refresh, so everything matters.)

My wish is to include the following in the basic Draw API:

1) Pen width and basic pen style set independently
2) Raster operations (foreground mix mode, see GDI::SetROP2, mostly XOR)

--

BTW: In the future, I would really like to see other higher performance output channels appearing for Draw in addition to current GDI and X11.

// Tom

Subject: Re: Extensions to Draw...Ops
Posted by mirek on Wed, 26 Nov 2008 19:59:25 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 26 November 2008 08:53
1) Pen width and basic pen style set independently
2) Raster operations (foreground mix mode, see GDI::SetROP2, mostly XOR)

At least raster operations are not the same as in X11...

However, you are not left in void. There are BeginGDI and EndGDI methods just for this - when you need something system specific and do not care about X11 in the moment.

You can quite easily start extending Draw by external functions as desired.

Meanwhile, I believe that places when you really need full ROPs and pen styles and others are in fact relatively local. I am all for host platform tweaks when needed...

Mirek

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

Subject: Re: Extensions to Draw...Ops
Posted by rylek on Fri, 28 Nov 2008 12:36:40 GMT
View Forum Message <> Reply to Message

I think binary ROP's are the same in X11 and W32. Also, line drawing capabilities in X11 always seemed to me much smarter than in W32. It's just a shame that under the old GDI, i.e. W98, the line style will have to be ignored for nonzero line widths, but perhaps we can see W98 as an

obsolete and only half-supported OS and so leave it to the hacker to write a W98-compatible wide dashed line renderer. (Or use the one I wrote some time ago though I woudn't recommend this as I'm not very satisfied with it and I hope time is ripening for its rewrite.)

Regards

Tomas

Subject: Re: Extensions to Draw...Ops
Posted by Tom1 on Fri, 28 Nov 2008 16:00:43 GMT
View Forum Message <> Reply to Message

The attached ZIP archive contains what I have in mind for adding wide styled lines for Win32 platform (read Windows NT and beyond). (I will later make the changes required for X11 if this gets accepted.) I have only very briefly tested it, so it should not be considered completed yet, but please feel free to test it and tell me if it breaks something vital. I have tried to maintain backward compatibility. Also, all new front-end functions to give access to these new line styles is still missing.

// Tom

File Attachments

1) DrawChanges.zip, downloaded 444 times
