Subject: Abstract Draw
Posted by mirek on Sun, 03 May 2009 18:06:40 GMT
View Forum Message <> Reply to Message

I am now working on final stage of recent Draw refactoring. The aim is to create 'headless' Draw (one that does not require X11 or GDI).

The critical moment is cleanup of Draw class and making it abstract. This is the first attempt:

```
class Draw {
public:
 enum {
  DOTS = 0x001,
  SYSTEM = 0x002,
  PRINTER = 0x004,
  BACK = 0x008,
  DRAWING = 0x010,
  PALETTE = 0x020,
  MONO = 0x040,
 };

 virtual dword GetInfo() = 0;
 virtual Size  GetPagePixels() const = 0;

 virtual void StartPage();
 virtual void EndPage();

 virtual void BeginOp() = 0;
 virtual void EndOp() = 0;
 virtual void OffsetOp(Point p) = 0;
 virtual bool ClipOp(const Rect& r) = 0;
 virtual bool ClipoffOp(const Rect& r) = 0;
 virtual bool ExcludeClipOp(const Rect& r) = 0;
 virtual bool IntersectClipOp(const Rect& r) = 0;
 virtual Rect GetClipOp() const = 0;
 virtual bool IsPaintingOp(const Rect& r) const = 0;

 virtual void DrawRectOp(int x, int y, int cx, int cy, Color color) = 0;
 virtual void DrawImageOp(int x, int y, int cx, int cy, const Image& img, const Rect& src, Color
color) = 0;
 virtual void DrawDataOp(int x, int y, int cx, int cy, const String& data, const char *id) = 0;
 virtual void DrawLineOp(int x1, int y1, int x2, int y2, int width, Color color) = 0;

 virtual void DrawPolyPolylineOp(const Point *vertices, int vertex_count,
                                 const int *counts, int count_count,
                                 int width, Color color, Color doxor) = 0;
 virtual void DrawPolyPolyPolygonOp(const Point *vertices, int vertex_count,
```

```
                              const int *subpolygon_counts, int scc,
                              const int *disjunct_polygon_counts, int dpcc,
                              Color color, int width, Color outline,
                              uint64 pattern, Color doxor) = 0;
 virtual void DrawArcOp(const Rect& rc, Point start, Point end, int width, Color color) = 0;

 virtual void DrawEllipseOp(const Rect& r, Color color, int pen, Color pencolor) = 0;
 virtual void DrawTextOp(int x, int y, int angle, const wchar *text, Font font,
                         Color ink, int n, const int *dx) = 0;
 virtual void DrawDrawingOp(const Rect& target, const Drawing& w) = 0;
 virtual void DrawPaintingOp(const Rect& target, const Painting& w) = 0;

 virtual void GetNativeDpi() const;
 virtual void BeginNative();
 virtual void EndNative();

 virtual int  GetCloffLevel() const = 0;

// --------------
 Size  GetPixelsPerInch() const                { return native ? nativeDpi : Dots() ? Size(600, 600) :
Size(96, 96); }
 Size  GetPageMMs() const                      { compute... }

 bool  Dots() const                           { return GetInfo() & DOTS; }
 bool  Pixels() const                         { return !Dots(); }
 bool  IsSystem() const                         { return GetInfo() & SYSTEM; }
 bool  IsPrinter() const                       { return GetInfo() & PRINTER; }
 bool  IsDrawing() const                        { return GetInfo() & DRAWING; }
 bool  IsNative() const                        { return GetInfo() & NATIVE; }
 bool  IsBack() const                          { return GetInfo() & BACK; }

 bool  IsPaletteMode() const                     { return GetInfo() & PALETTE; }
 bool  IsMono() const                          { return GetInfo() & MONO; }

 int  GetNativeX(int x) const;
 int  GetNativeY(int x) const;
 void Native(int& x, int& y) const;
 void Native(Point& p) const;
 void Native(Size& sz) const;
 void Native(Rect& r) const;

 void  Begin()                               { BeginOp(); }
 void  End()                                 { EndOp(); }
 void  Offset(Point p)                          { OffsetOp(p); }
 void  Offset(int x, int y);
 bool  Clip(const Rect& r)                       { return ClipOp(r); }
 bool  Clip(int x, int y, int cx, int cy);
 bool  Clipoff(const Rect& r)                    { return ClipoffOp(r); }
```

```
bool  Clipoff(int x, int y, int cx, int cy);
bool  ExcludeClip(const Rect& r)                { return ExcludeClipOp(r); }
bool  ExcludeClip(int x, int y, int cx, int cy);
bool  IntersectClip(const Rect& r)              { return IntersectClipOp(r); }
bool  IntersectClip(int x, int y, int cx, int cy);
Rect  GetClip() const                           { return GetClipOp(); }
bool  IsPainting(const Rect& r) const           { return IsPaintingOp(r); }
bool  IsPainting(int x, int y, int cx, int cy) const;

Point GetOffset() const                         { return actual_offset; }

void DrawRect(int x, int y, int cx, int cy, Color color);
void DrawRect(const Rect& rect, Color color);

void DrawImage(int x, int y, int cx, int cy, const Image& img, const Rect& src);
void DrawImage(int x, int y, int cx, int cy, const Image& img);
void DrawImage(int x, int y, int cx, int cy, const Image& img, const Rect& src, Color color);
void DrawImage(int x, int y, int cx, int cy, const Image& img, Color color);

void DrawImage(const Rect& r, const Image& img, const Rect& src);
void DrawImage(const Rect& r, const Image& img);
void DrawImage(const Rect& r, const Image& img, const Rect& src, Color color);
void DrawImage(const Rect& r, const Image& img, Color color);

void DrawImage(int x, int y, const Image& img, const Rect& src);
void DrawImage(int x, int y, const Image& img);
void DrawImage(int x, int y, const Image& img, const Rect& src, Color color);
void DrawImage(int x, int y, const Image& img, Color color);

void DrawData(int x, int y, int cx, int cy, const String& data, const char *type);
void DrawData(const Rect& r, const String& data, const char *type);

void DrawLine(int x1, int y1, int x2, int y2, int width = 0, Color color = DefaultInk);
void DrawLine(Point p1, Point p2, int width = 0, Color color = DefaultInk);

void DrawEllipse(const Rect& r, Color color = DefaultInk,
                 int pen = Null, Color pencolor = DefaultInk);
void DrawEllipse(int x, int y, int cx, int cy, Color color = DefaultInk,
                 int pen = Null, Color pencolor = DefaultInk);

void DrawArc(const Rect& rc, Point start, Point end, int width = 0, Color color = DefaultInk);

void DrawPolyPolyline(const Point *vertices, int vertex_count,
                      const int *counts, int count_count,
                      int width = 0, Color color = DefaultInk, Color doxor = Null);
void DrawPolyPolyline(const Vector<Point>& vertices, const Vector<int>& counts,
                      int width = 0, Color color = DefaultInk, Color doxor = Null);
void DrawPolyline(const Point *vertices, int count,
```

```
            int width = 0, Color color = DefaultInk, Color doxor = Null);
 void DrawPolyline(const Vector<Point>& vertices,
            int width = 0, Color color = DefaultInk, Color doxor = Null);

 void   DrawPolyPolyPolygon(const Point *vertices, int vertex_count,
                    const int *subpolygon_counts, int subpolygon_count_count,
                    const int *disjunct_polygon_counts, int disjunct_polygon_count_count,
                    Color color = DefaultInk, int width = 0, Color outline = Null,
                    uint64 pattern = 0, Color doxor = Null);
 void   DrawPolyPolyPolygon(const Vector<Point>& vertices,
                      const Vector<int>& subpolygon_counts,
                      const Vector<int>& disjunct_polygon_counts,
                      Color color = DefaultInk, int width = 0, Color outline = Null, uint64 pattern = 0,
Color doxor = Null);
 void   DrawPolyPolygon(const Point *vertices, int vertex_count,
                   const int *subpolygon_counts, int subpolygon_count_count,
                   Color color = DefaultInk, int width = 0, Color outline = Null, uint64 pattern = 0, Color
doxor = Null);
 void   DrawPolyPolygon(const Vector<Point>& vertices, const Vector<int>& subpolygon_counts,
                   Color color = DefaultInk, int width = 0, Color outline = Null, uint64 pattern = 0, Color
doxor = Null);
 void   DrawPolygons(const Point *vertices, int vertex_count,
                  const int *polygon_counts, int polygon_count_count,
                  Color color = DefaultInk, int width = 0, Color outline = Null, uint64 pattern = 0, Color
doxor = Null);
 void   DrawPolygons(const Vector<Point>& vertices, const Vector<int>& polygon_counts,
                  Color color = DefaultInk, int width = 0, Color outline = Null, uint64 pattern = 0, Color
doxor = Null);
 void   DrawPolygon(const Point *vertices, int vertex_count,
                 Color color = DefaultInk, int width = 0, Color outline = Null, uint64 pattern = 0, Color
doxor = Null);
 void   DrawPolygon(const Vector<Point>& vertices,
                 Color color = DefaultInk, int width = 0, Color outline = Null, uint64 pattern = 0, Color
doxor = Null);

 void DrawDrawing(const Rect& r, const Drawing& iw) { DrawDrawingOp(r, iw); }
 void DrawDrawing(int x, int y, int cx, int cy, const Drawing& iw);

 void DrawPainting(const Rect& r, const Painting& iw) { DrawPaintingOp(r, iw); }
 void DrawPainting(int x, int y, int cx, int cy, const Painting& iw);

 void DrawText(int x, int y, int angle, const wchar *text, Font font = StdFont(),
          Color ink = DefaultInk, int n = -1, const int *dx = NULL);
 void DrawText(int x, int y, const wchar *text, Font font = StdFont(),
          Color ink = DefaultInk, int n = -1, const int *dx = NULL);

 void DrawText(int x, int y, const WString& text, Font font = StdFont(),
          Color ink = DefaultInk, const int *dx = NULL);
```

```
 void DrawText(int x, int y, int angle, const WString& text, Font font = StdFont(),
          Color ink = DefaultInk, const int *dx = NULL);

 void DrawText(int x, int y, int angle, const char *text, byte charset,
            Font font = StdFont(), Color ink = DefaultInk, int n = -1, const int *dx = NULL);
 void DrawText(int x, int y, const char *text, byte charset, Font font = StdFont(),
          Color ink = DefaultInk, int n = -1, const int *dx = NULL);

 void DrawText(int x, int y, int angle, const char *text,
            Font font = StdFont(), Color ink = DefaultInk, int n = -1, const int *dx = NULL);
 void DrawText(int x, int y, const char *text, Font font = StdFont(),
          Color ink = DefaultInk, int n = -1, const int *dx = NULL);

 void DrawText(int x, int y, const String& text, Font font = StdFont(),
          Color ink = DefaultInk, const int *dx = NULL);
 void DrawText(int x, int y, int angle, const String& text, Font font = StdFont(),
          Color ink = DefaultInk, const int *dx = NULL);
};
```

Anything missing there?

Mirek

Subject: Re: Abstract Draw
Posted by mirek on Tue, 12 May 2009 10:16:18 GMT
View Forum Message <> Reply to Message

Well, I am in a phase when my apps are working just fine with new Draw interface and infrastructure in Win32.

I encourage experienced developers to check how well it works for them - all you need is to get 'newdraw' nest from svn and insert it into assembly chain (to replace old Draw etc..).

I also guess I owe you explanation of what is going on:

This is "step 2" of new headless Draw and will be finished when X11 part is fixed. Then next step is to refactor font metrics handling in X11 (to avoid dependency on Xft) and then to finally move X11/Win32 stuff from Draw to CtrlCore. The final desirable result is Draw that does not depend on X11, which is potentially important for web applications.

Mirek

Subject: Re: Abstract Draw
Posted by kodos on Tue, 12 May 2009 18:40:07 GMT

View Forum Message <> Reply to Message

I just did a quick test and everything seems alright.
The only changes I did have to do on my Cairo renderer were adding a GetInfo method and removing the code that handled the actual_offset .

Subject: Re: Abstract Draw
Posted by mirek on Tue, 12 May 2009 18:57:58 GMT
View Forum Message <> Reply to Message

kodos wrote on Tue, 12 May 2009 14:40I just did a quick test and everything seems alright.
The only changes I did have to do on my Cairo renderer were adding a GetInfo method and removing the code that handled the actual_offset .

Good, thanks.

Mirek

Subject: Re: Abstract Draw
Posted by Mindtraveller on Thu, 21 May 2009 22:40:48 GMT
View Forum Message <> Reply to Message

As I remember, using font metrics detection on *nix platforms makes some delaying on GUI program startup (on *nix) - especially if there are many installed fonts. I wonder if switching to "internal" font metrics mechanism will eliminate this startup delay...

Subject: Re: Abstract Draw
Posted by mirek on Fri, 22 May 2009 08:37:56 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Thu, 21 May 2009 18:40As I remember, using font metrics detection on *nix platforms makes some delaying on GUI program startup (on *nix) - especially if there are many installed fonts. I wonder if switching to "internal" font metrics mechanism will eliminate this startup delay...

Well, interestingly, font metrics in X11 is exactly the next thing to do

I hope to make some improvements there, but what I have seen so far, I am little bit afraid that available freetype interfaces are not very favorable here. But I need to check it more..

Mirek

Subject: Re: Abstract Draw
Posted by masu on Tue, 26 May 2009 21:15:27 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I also did a quick test (used newdraw for TheIDE) on OpenBSD and it works fine until now.

I will check on FreeBSD, too. Though I expect nothing different there.

Matthias

Subject: Re: Abstract Draw
Posted by masu on Wed, 27 May 2009 09:35:46 GMT
View Forum Message <> Reply to Message

On WinXP to get it to compile with MingW, I had to change MetaFile.cpp:201 to:

```
  Image img(iw);
  ib = img;
```

Matthias

Subject: Re: Abstract Draw
Posted by mirek on Fri, 29 May 2009 08:56:40 GMT
View Forum Message <> Reply to Message

Thanks.

Mirek