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