Subject: How to efficiently update a large Image?
Posted by Tom1 on Mon, 27 Jul 2009 10:24:14 GMT
View Forum Message <> Reply to Message

Hi,

I need to frequently update a small amount of pixels in a relatively large image (e.g. 1920x1200) and then draw the parts of image on screen. Unfortunately the following code is too slow:

```
ImageBuffer ib(image);
// .. here's the manipulation of the ib
image=ib;
```

Help, anybody?

(I used to have my own graphics layer implemented for GDI and X11 but they just badly broke down due to the recent Draw virtualization.)

// Tom

Subject: Re: How to efficiently update a large Image?
Posted by mirek on Mon, 27 Jul 2009 11:39:25 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 27 July 2009 06:24Hi,

I need to frequently update a small amount of pixels in a relatively large image (e.g. 1920x1200) and then draw the parts of image on screen. Unfortunately the following code is too slow:

```
ImageBuffer ib(image);
// .. here's the manipulation of the ib
image=ib;
```

Help, anybody?

The question is why it is slow...

The first thing to try is

ib.SetKind(....)

(I guess it will be ib.SetKind(IMAGE_OPAQUE)).

this would prevent optimization scan.

Just for better undestanding, what is slow here? Are we speaking about milliseconds or second?

Quote:
(I used to have my own graphics layer implemented for GDI and X11 but they just badly broke down due to the recent Draw virtualization.)

// Tom

In most cases, dynamic_cast<SystemDraw *> makes original code work just fine (SystemDraw has all interface and attributes of former Draw).

Mirek

Subject: Re: How to efficiently update a large Image?
Posted by Tom1 on Mon, 27 Jul 2009 14:47:54 GMT
View Forum Message <> Reply to Message

Hi Mirek,

SetKind(IMAGE_OPAQUE) gives marginal improvement.

Now I see the visit from Image to ImageBuffer and back to Image does not take more than about 1.5 ms. However, after this visit the DrawImage does something more than usual and becomes slow. It takes about 30 ms to update a small stripe of an image using DrawImage whereas without the ImageBuffer visit, this takes less than a millisecond.

I also noticed that the application allocates and deallocates several megabytes worth of memory when running probably because of what is happening around this image or imagebuffer.

--

Is there any way to directly update the Image pixel contents without ImageBuffer and the associated DrawImage overhead?

(I need to get familiar with the dynamic_cast thing soon... It was not quite as easy as I had hoped it would be.)

Thanks,

Tom

Subject: Re: How to efficiently update a large Image?

Posted by mirek on Mon, 27 Jul 2009 15:15:15 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 27 July 2009 10:47Hi Mirek,

SetKind(IMAGE_OPAQUE) gives marginal improvement.

Now I see the visit from Image to ImageBuffer and back to Image does not take more than about 1.5 ms. However, after this visit the DrawImage does something more than usual and becomes slow. It takes about 30 ms to update a small stripe of an image using DrawImage whereas without the ImageBuffer visit, this takes less than a millisecond.

It is because the raster needs to be converted to windows object (HBMP) that then can be painted on the screen.

That said, Image is really optimized for thousands of small images, not single big one...

That said (2), there is SetSurface function (which is sort of semi-public), maybe it can be helpful somehow in this case.

Another option is to divide the big image into many smaller ones.

Quote:
I also noticed that the application allocates and deallocates several megabytes worth of memory when running probably because of what is happening around this image or imagebuffer.

Most like HBMP associated data.

Quote:
Is there any way to directly update the Image pixel contents without ImageBuffer and the associated DrawImage overhead?

No, but I believe it should not really be a problem. Image->ImageBuffer->Image is extremely fast. The problem is that it is necessarry to move those pixels to GDI afterwards.

Mirek

Subject: Re: How to efficiently update a large Image?
Posted by mirek on Mon, 27 Jul 2009 15:18:12 GMT
View Forum Message <> Reply to Message

P.S.: In Bazaar, there is RasterCtrl package and RasterCtrlTest. AFAIK, these are classes for handling extremely large images. Maybe it could help.

Mirek

Subject: Re: How to efficiently update a large Image?
Posted by Tom1 on Tue, 28 Jul 2009 09:16:04 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thanks for the analysis. The SetSurface note was especially helpful. Namely, it appears to use the same GDI function I used in my own graphics library and therefore it should give the performance I need.

I noticed the SetSurface(Draw& w, ..) in Image.h should probably be updated to SetSurface(SystemDraw& w,..) to work.

--

The SetSurface() also appears to be the far most efficient way to put ImageBuffers (e.g. the ones created with Painter) on screen.

That in mind, may I suggest adding a version of SetSurface that uses a source rectangle within the image buffer and a target rectangle for the control, and can therefore optimally update a smaller area of a Control.

Something like:

SetSurface(SystemDraw& w, Rect src, Rect target, Size sz, RGBA *pixelbuffer)

This would further boost the Painter when used on screen as the ImageBuffer would need to be updated only when the content changes and the Paint routine would update the screen using the current buffer contents and only update the areas required by Paint.

Thanks for your help Mirek,

Tom

Subject: Re: How to efficiently update a large Image?
Posted by mirek on Tue, 28 Jul 2009 14:29:26 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 28 July 2009 05:16
SetSurface(SystemDraw& w, Rect src, Rect target, Size sz, RGBA *pixelbuffer)

If you can provide X11 and Win32 versions, I would be happy to add it.

Anyway, now thinking, maybe it should be Draw& after all, because that is what we have in Paint, with runtime check that it is in fact SystemDraw...

Mirek

Subject: Re: How to efficiently update a large Image?
Posted by mirek on Tue, 28 Jul 2009 14:32:12 GMT
View Forum Message <> Reply to Message

Even better idea:

SetSurface that is optimized as current one if given Draw is SystemDraw (and the target is screen), but uses Image for all other cases (e.g. printer).

Mirek

Subject: Re: How to efficiently update a large Image?
Posted by Tom1 on Wed, 29 Jul 2009 07:18:06 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Here's the raw GDI code for SetSurface with added flexibility:

```
void SetSurface(HDC dc, Rect &dest, Point &srcoff, Size &bufsz, const RGBA *pixels){
 GuiLock __;
 BitmapInfo32__ bi(bufsz.cx, bufsz.cy);
 ::SetDIBitsToDevice(dc,dest.left,dest.top,dest.Width(),dest.Height(),srcoff.x,-srcoff.y-dest.Height()
+bufsz.cy,0,bufsz.cy,pixels, bi, DIB_RGB_COLORS);
}

void SetSurface(SystemDraw &w, Rect &dest, Point &srcoff, Size &bufsz, const RGBA *pixels){
 SetSurface(w.GetHandle(),dest,srcoff,bufsz,pixels);
}
```

I'm not quite confident I can reproduce this in X11 in a truly powerful way, so I hope someone with solid X11 knowledge will pick it up.

// Tom

Subject: Re: How to efficiently update a large Image?
Posted by mirek on Sun, 02 Aug 2009 13:11:06 GMT
View Forum Message <> Reply to Message

OK, I have added X11 variant and via-Image variant (for non-screen targets), maybe changing parameters a little.

According to my benchmarks, using optimized SetSurface is about 2.5x faster than using Image.

Thanks for the idea.

Mirek

Subject: Re: How to efficiently update a large Image?
Posted by Tom1 on Wed, 26 Aug 2009 05:57:05 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Sorry for the delay in my response -- been out of the office for a while.

Anyway, thanks for adding this functionality. In my application it really gives me far more that 2.5x speed boost.

Unfortunately, there is still some problem with X11. Generally, the new SetSurface works OK with partial updates and offsets, but when dragging another window over a window with image contents updated with SetSurface, the view gets severely scrambled.

```
virtual void Paint(Draw &draw){
  // The image to draw is in ImageBuffer ib;
  Rect paintrect=draw.GetPaintRect();
  SetSurface(draw,paintrect,ib,ib.GetSize(),Point(paintrect.left,paintrect.top));
}
```

With GDI it's working correctly.

// Tom

Subject: Re: How to efficiently update a large Image?
Posted by mirek on Thu, 27 Aug 2009 22:24:45 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 26 August 2009 01:57Hi Mirek,

Sorry for the delay in my response -- been out of the office for a while.

Anyway, thanks for adding this functionality. In my application it really gives me far more that 2.5x speed boost.

Unfortunately, there is still some problem with X11. Generally, the new SetSurface works OK with partial updates and offsets, but when dragging another window over a window with image contents updated with SetSurface, the view gets severely scrambled.

I am sorry, but I have trouble reproducing the problem.

My current testcase:

```
#include <CtrlLib/CtrlLib.h>

using namespace Upp;

struct MyApp : TopWindow {
 RGBA pixels[65536];

 virtual void Paint(Draw& w) {
  Rect r = w.GetPaintRect();
  w.DrawRect(r, LtGray());
  Point p = r.TopLeft();
  if(p.x < 256 && p.y < 256)
   SetSurface(w, r, pixels, Size(256, 256), p);
 }

 MyApp() {
  RGBA *t = pixels;
  for(int x = 0; x < 256; x++)
   for(int y = 0; y < 256; y++) {
    int d = (x - 128) * (x - 128) + (y - 128) * (y - 128);
    RGBA c = Black();
    if(d > 120 * 120)
     c = White();
    if(d < 120 * 120)
     c.r = 255 - min(255 * d / (120 * 120), 255);
    *t++ = c;
   }
 }
};

GUI_APP_MAIN
{
```

```
 MyApp().Run();
}
```

Could you create some testcase that shows the problem?

Mirek

```
virtual void Paint(Draw &draw){
  // The image to draw is in ImageBuffer ib;
  Rect paintrect=draw.GetPaintRect();
  SetSurface(draw,paintrect,ib,ib.GetSize(),Point(paintrect.left,paintrect.top));
}
```

With GDI it's working correctly.

// Tom
[/quote]

Subject: Re: How to efficiently update a large Image?
Posted by Tom1 on Tue, 01 Sep 2009 06:24:53 GMT
View Forum Message <> Reply to Message

Mirek,

Here's the testcase. Only after running your test code I realized the problem was related to having a control inside the main window. So, I guess this has something to do with the control offset within the parent control.

```
#include <CtrlLib/CtrlLib.h>
#include <Painter/Painter.h>

using namespace Upp;

class PainterCtrl : public Ctrl {
 ImageBuffer ib;

public:
 virtual void Paint(Painter &pntr,Size &sz)=0;

 virtual void Layout(){
  Size sz=GetSize();
```

```
  ib.Clear();
  ib.Create(sz);
  BufferPainter pntr(ib);
  Paint(pntr,sz);
 }

 virtual void Paint(Draw &draw) {
  Rect paintrect=draw.GetPaintRect();

  Point p = paintrect.TopLeft();
  if(p.x < ib.GetSize().cx && p.y < ib.GetSize().cy)
   SetSurface(draw, paintrect, ib, ib.GetSize(), p);
 }

};

struct ExampleCtrl : PainterCtrl {

 virtual void Paint(Painter &pntr,Size &sz){
  pntr.Move(0,0).Line(sz.cx,0).Line(sz.cx,sz.cy).Line(0,sz.cy).Close().Fill(White()).Stroke(5,Black())
;
  pntr.Move(0,0).Line(sz.cx,sz.cy).Stroke(5,Black());
 }
};

class ExampleTopWindow: public TopWindow{
public:
 ExampleCtrl ec;

 virtual void Layout(){
  ec.SetRect(20,0,GetSize().cx-20,GetSize().cy);
 }

 ExampleTopWindow(){
  Add(ec);
 }
};

GUI_APP_MAIN
{
   ExampleTopWindow win;
   win.Sizeable();
   win.Open();
   win.Run();
}
```

(This example also sort of demonstrates how to boost application performance when drawing

complex items with e.g. Painter and then not having to regenerate them each time the window gets an OS initiated paint request.)

// Tom

Subject: Re: How to efficiently update a large Image?
Posted by mirek on Wed, 02 Sep 2009 09:37:01 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 01 September 2009 02:24Mirek,

Here's the testcase. Only after running your test code I realized the problem was related to having a control inside the main window. So, I guess this has something to do with the control offset within the parent control.

```
#include <CtrlLib/CtrlLib.h>
#include <Painter/Painter.h>

using namespace Upp;

class PainterCtrl : public Ctrl {
 ImageBuffer ib;

public:
 virtual void Paint(Painter &pntr,Size &sz)=0;

 virtual void Layout(){
  Size sz=GetSize();
  ib.Clear();
  ib.Create(sz);
  BufferPainter pntr(ib);
  Paint(pntr,sz);
 }

 virtual void Paint(Draw &draw) {
  Rect paintrect=draw.GetPaintRect();

  Point p = paintrect.TopLeft();
  if(p.x < ib.GetSize().cx && p.y < ib.GetSize().cy)
   SetSurface(draw, paintrect, ib, ib.GetSize(), p);
 }

};

struct ExampleCtrl : PainterCtrl {
```

```
 virtual void Paint(Painter &pntr,Size &sz){
  pntr.Move(0,0).Line(sz.cx,0).Line(sz.cx,sz.cy).Line(0,sz.cy).Close().Fill(White()).Stroke(5,Black())
;
  pntr.Move(0,0).Line(sz.cx,sz.cy).Stroke(5,Black());
 }
};

class ExampleTopWindow: public TopWindow{
public:
 ExampleCtrl ec;

 virtual void Layout(){
  ec.SetRect(20,0,GetSize().cx-20,GetSize().cy);
 }

 ExampleTopWindow(){
  Add(ec);
 }
};

GUI_APP_MAIN
{
   ExampleTopWindow win;
   win.Sizeable();
   win.Open();
   win.Run();
}
```

(This example also sort of demonstrates how to boost application performance when drawing complex items with e.g. Painter and then not having to regenerate them each time the window gets an OS initiated paint request.)

// Tom

Thanks for testcase. You are right, the cause was that offset of child widget was not taken into account.

Hopefully fixed now.

Mirek

Subject: Re: How to efficiently update a large Image?
Posted by Tom1 on Thu, 03 Sep 2009 09:45:44 GMT
View Forum Message <> Reply to Message

Thanks Mirek, it works beautifully now!

// Tom