Subject: Optimizing DrawImage across platforms
Posted by Tom1 on Wed, 15 Apr 2020 07:52:54 GMT
View Forum Message <> Reply to Message

Hi,

There is an issue with ViewDraw when using the GTK3 backend on Linux. The following testcase works correctly on Windows (GUI) and Linux (GUI X11) but not on Linux (GUI) using GTK3. The cross should follow the cursor on the white background, but on GTK3 the entire window goes black and only the area updated with ViewDraw is valid afterwards.

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

using namespace Upp;

class Testcase5 : public TopWindow {
public:
 typedef Testcase5 CLASSNAME;

 Rect loc;

 Testcase5(){
  loc.SetNull();
 }

 void Paint(Draw &w){
  w.DrawRect(Rect(GetSize()),White());
  loc.SetNull();
 }

 void MouseMove(Point p, dword keyflags){
  ImageBuffer sb(32,32);
  BufferPainter sbp(sb);
  sbp.Clear(RGBAZero());
  sbp.Move(0,0).Line(31,31);
  sbp.Move(0,31).Line(31,0);
  sbp.Stroke(5,Gray());

  ViewDraw draw(this);
  if(!loc.IsNullInstance()) draw.DrawRect(loc,White());
  loc.Set(p.x-16,p.y-16,p.x+16,p.y+16);
  draw.DrawImage(loc.left,loc.top,Image(sb));
 }
};


GUI_APP_MAIN
{
```

```
 Testcase5().Run();
}
```

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Wed, 15 Apr 2020 12:35:50 GMT
View Forum Message <> Reply to Message

Well, this is what is possible with gtk3. Let us say that what happens to areas that you do not paint to is implementation defined.

If this is a showstopper, I could probably add ViewDraw constructor that actually specifies the area of Ctrl that you wish to repaint...

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Wed, 15 Apr 2020 13:15:38 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Well, yes.. I will have some small targets shown on a map display and updated between once and five times per second depending on their positioning rate. It would be terrible waste to update the entire window area each time any of the targets move.

So, if it is not too much work, I would certainly prefer a way to update just a small rectangle within the Ctrl. Painting the entire map possibly tens of times per second is just too much.

I assume, this ViewDraw constructor will then apply to all backends?

Thanks and best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Wed, 15 Apr 2020 20:21:16 GMT
View Forum Message <> Reply to Message

So I have got both good news and bad news...

It is done for X11, Gtk, Win32.

However, while trying to do that on MacOS, I have found that even existing ViewDraw stopped working. Unfortunately, it seems like MacOS issue, critical method is broken, as other people found too:

https://cgit.freedesktop.org/libreoffice/core/commit/?id=959e8ae7ea33ce94dd80ee8ea172b6db64593873

Given this, it actually seems impossible to implement ViewDraw for MacOS... It looks like we should actually deprecate it as well; simply use Refresh and optimized Paint....

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Thu, 16 Apr 2020 19:08:51 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thank you very much for solving this issue for me! :)

I did some further tuning of the render/paint structure using the new ViewDraw and ended up with a nice target update time of 60-150 micro seconds on both Windows and Linux GTK3 with the following code:

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

using namespace Upp;

class Testcase5 : public TopWindow {
public:
 typedef Testcase5 CLASSNAME;

 Rect loc;

 ImageBuffer map;

 Testcase5(){
  loc.SetNull();
 }

 void Paint(Draw &w){
  int64 t0=usecs();

  SetSurface(w,Rect(map.GetSize()),map,map.GetSize(),Point(0,0));
```

```
  loc.SetNull();

  int64 t1=usecs();
  Title(Format("Paint took %lld us",t1-t0));
 }

 void Layout(){
  Size sz=GetSize();
  map.Create(sz);
  BufferPainter bp(map);
  bp.RectPath(Rect(sz));
  bp.Fill(Pointf(0,0),Green(),Pointf(sz.cx-1,sz.cy-1),Yellow());
  bp.Move(0,0).Line(Pointf(sz.cx-1,sz.cy-1)).Stroke(1,Black());
  bp.Move(0,sz.cy-1).Line(Pointf(sz.cx-1,0)).Stroke(1,Black());
 }

 void MouseMove(Point p, dword keyflags){
  int64 t0=usecs();

  Rect area;
  area.SetNull();
  if(!loc.IsNullInstance()) area.Union(loc);
  loc.Set(p.x-16,p.y-16,p.x+16,p.y+16);
  area.Union(loc);
  area.Inflate(2);
  area.Intersect(Rect(GetSize()));

  ImageBuffer sb(area.GetSize());
  for(int y=0;y<area.Height();y++)
memcpy(sb[y],&map[y+area.top][area.left],sizeof(RGBA)*area.Width());

  BufferPainter sbp(sb);
  sbp.Translate(loc.left-area.left,loc.top-area.top);
  sbp.Move(0,0).Line(31,31);
  sbp.Move(0,31).Line(31,0);
  sbp.Stroke(5,Black());

  ViewDraw draw(this,area);
  SetSurface(draw,area.GetSize(),sb,sb.GetSize(),Point(0,0));

  int64 t1=usecs();
  Title(Format("Move took %lld us",t1-t0));

 }
};


GUI_APP_MAIN
```

```
{
 Testcase5().Sizeable().MaximizeBox().MinimizeBox().Run();
}
```

I do not need support for this feature in Mac, but what worries me is the potential deprecation of ViewDraw... Is there a real risk that you would drop ViewDraw entirely? As you can see from running the example above, an optimized Paint (as above, drawing a readily rendered ImageBuffer with SetSurface()) of the entire view on a 4K screen takes something like 6-12 ms, the small update with ViewDraw runs at around just 1 % of that time. (That's the difference between success and failure for my task.)

How would similar code and its performance look with what you refer to as 'Refresh and optimized Paint'?

Thanks and best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Thu, 16 Apr 2020 20:06:48 GMT
View Forum Message <> Reply to Message

Hi,

Just noticed a related issue: RectTracker (using ViewDraw) is not working in GTK3, while it is OK in X11 and Win32. This is also one important feature I need in my application. (On top of that I also use an inherited LineTracker to visualize drawing of straight lines.)

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Thu, 16 Apr 2020 22:41:11 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Thu, 16 April 2020 21:08How would similar code and its performance look with what you refer to as 'Refresh and optimized Paint'?

So I guess we are in similar situation, painting maps and complex polygon. What we actually do is that we paint the map to Image, then in Paint we just put this image on screen and all "indicators" paint over that map.

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Fri, 17 Apr 2020 11:20:27 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 17 April 2020 01:41Tom1 wrote on Thu, 16 April 2020 21:08How would similar code and its performance look with what you refer to as 'Refresh and optimized Paint'?

So I guess we are in similar situation, painting maps and complex polygon. What we actually do is that we paint the map to Image, then in Paint we just put this image on screen and all "indicators" paint over that map.

Mirek

Well, so it seems. This basically sounds the same what I'm doing. My chain is vector map > Painter > ImageBuffer > SetSurface. And then the small moving targets or overlay texts/objects/indicators on top of that with a higher update rate. Just like my example above. The situation reminds me of sprites in the old times, which would fit in nicely. :)

The cost of 6-12 ms per 4K screen update (using SetSurface) for just a tiny little change is too much in my opinion. The new small-area ViewDraw based approach is far more efficient and should be here to stay even if MacOS does not support it at the moment -- or ever.

Another approach could possibly be to have 'Refresh(Rect) and Paint(Rect)/Paint(Vector<Rect>)' routine variants for partial updates via Paint. But I'm not sure if this solves the RectTracker requirements. And I would still prefer the new partial ViewDraw for the purpose...

Yet another question is, if OpenGL, Direct2D/3D, or some other backend would offer faster SetSurface capability taking the cost of 4K screen update well below millisecond level. Then again, I would like to avoid that path for now to ensure best compatibility with existing low-end hardware the clients may have.

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Fri, 17 Apr 2020 11:38:49 GMT
View Forum Message <> Reply to Message

Well, I think the difference might be that SetSurface is fine as long as you use it just once.

For this scenario, converting to Image first is better option, because you will repaint that one many

times. Converting to Image basically means that the data gets transfered to GPU and the next time you paint it it goes from GPU memory.

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Fri, 17 Apr 2020 12:36:23 GMT
View Forum Message <> Reply to Message

Hi Mirek,

```
int64 t0=usecs();
w.DrawImage(0,0,image);
int64 t1=usecs();
Title(Format("Paint took %lld us",t1-t0));
```

In Windows this takes about 21 milliseconds in the first run and about the same 6.. milliseconds in the following runs as with SetSurface steadily.

In Linux with GTK3, the DrawImage() and SetSurface() both perform equally at around 12 milliseconds.

HOWEVER, Linux with X11 re-using Image is super fast! Repeated use of Image in DrawImage() takes just 5-20 microseconds, while initial DrawImage() takes around 10 milliseconds. In comparison, SetSurface() takes here over 20 milliseconds for some reason...

I guess this last DrawImage() result with X11 was the one you were referring to. Unfortunately, this is not the case with Windows and GTK3. Infact, it would change the game entirely if similar results were available with Windows and GTK3.

Best regards,

Tom

*** Please note that Linux and Windows results are not comparable with each other as they are run on different machines. ***

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Fri, 17 Apr 2020 15:00:00 GMT
View Forum Message <> Reply to Message

One finding:

Increasing cache size in GTK3 SysDrawImageOp improves repeated DrawImage() performance by a factor of two to about 5-6 ms per update.

```
void SystemDraw::SysDrawImageOp(int x, int y, const Image& img, Color color)
{
...
// cache.Shrink(4 * 1024 * 768, 1000); // Cache must be after Paint because of PaintOnly!
 cache.Shrink(4 * 3840 * 2160, 1000); // Cache must be after Paint because of PaintOnly!
}
```

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Sat, 18 Apr 2020 12:05:29 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I noticed an update in trunk for GTK3 SysDrawImageOp(), but for some reason a fixed value works twice as fast in repeated use.

```
 //Size sz = Ctrl::GetPrimaryScreenArea().GetSize();
 Size sz(3840,2160); // Twice as fast compared to the above... why?
 cache.Shrink(4 * sz.cx * sz.cy, 1000); // Cache must be after Paint because of PaintOnly!
```

I just cannot figure out why. I tried even using static Size, with no improvement.

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Sat, 18 Apr 2020 13:06:41 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 18 April 2020 14:05Hi Mirek,

I noticed an update in trunk for GTK3 SysDrawImageOp(), but for some reason a fixed value works twice as fast in repeated use.

```
 //Size sz = Ctrl::GetPrimaryScreenArea().GetSize();
 Size sz(3840,2160); // Twice as fast compared to the above... why?
 cache.Shrink(4 * sz.cx * sz.cy, 1000); // Cache must be after Paint because of PaintOnly!
```

I just cannot figure out why. I tried even using static Size, with no improvement.

Best regards,

Tom

Try DDUMP(Ctrl::GetPrimaryScreenArea()); there. Perhaps it is wrong?

In related news, I have reimplemented RectTracker to actually avoid using ViewDraw (what can I do if it does not work on MacOS, right?).

It now works by "snapshoting" master widget into Image and then using normal Paint. In the end I am quite happy about it, as this completely removes the need for RectTracker support in all hosts - it was really ugly in gtk3 and macos...

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Sat, 18 Apr 2020 14:43:40 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Two things about the changes:

```
 case WM_PAINT:
  DLOG("WM_PAINT " << Name(this));
```

The above DLOG in Win32Proc.cpp prevents Release mode compilation.

Second: RectTracker no longer works at every drag direction. It used to allow it after:

tracker.MinSize(Size(-100000,-100000));

Well, maybe it works behind the scenes, but it does not show the rect on screen for north or west drag directions.

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Sat, 18 Apr 2020 14:48:55 GMT
View Forum Message <> Reply to Message

As for windows numbers, looks like Image in Win32 was overengineered, one of optimizations was that the first time it is painted, it paints with SetSurface, only for subsequent paints actually move Image to system handles.

I have now exluded all those things whith #if 0 - looks like it improved at least current RectTracker performance quite nicely...

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Sat, 18 Apr 2020 14:57:22 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 18 April 2020 16:06Tom1 wrote on Sat, 18 April 2020 14:05Hi Mirek,

I noticed an update in trunk for GTK3 SysDrawImageOp(), but for some reason a fixed value works twice as fast in repeated use.

```
//Size sz = Ctrl::GetPrimaryScreenArea().GetSize();
Size sz(3840,2160); // Twice as fast compared to the above... why?
cache.Shrink(4 * sz.cx * sz.cy, 1000); // Cache must be after Paint because of PaintOnly!
```

I just cannot figure out why. I tried even using static Size, with no improvement.

Best regards,

Tom

Try DDUMP(Ctrl::GetPrimaryScreenArea()); there. Perhaps it is wrong?

In related news, I have reimplemented RectTracker to actually avoid using ViewDraw (what can I do if it does not work on MacOS, right?).

It now works by "snapshoting" master widget into Image and then using normal Paint. In the end I am quite happy about it, as this completely removes the need for RectTracker support in all hosts - it was really ugly in gtk3 and macos...

Mirek

I tried DUMPing and got:
Ctrl::GetPrimaryScreenArea() = [0, 0] - [3840, 2080] : (3840, 2080)
This is what I get for LinuxMint using 4K display... It obviously drops the taskbar out of PrimaryScreenArea as the screen height is 2160 in reality. However, the performance is twice as good with a fixed value (Size(3840, 2080)) than when calling the function. It looks like calling Ctrl::GetPrimaryScreenArea() takes quite a long time to complete.

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Sat, 18 Apr 2020 15:05:59 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 18 April 2020 17:48As for windows numbers, looks like Image in Win32 was overengineered, one of optimizations was that the first time it is painted, it paints with SetSurface, only for subsequent paints actually move Image to system handles.

I have now exluded all those things whith #if 0 - looks like it improved at least current RectTracker performance quite nicely...

Mirek

Sounds good... Can you put it in trunk for me to test?

BR, Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Sat, 18 Apr 2020 15:12:55 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 18 April 2020 16:43Hi Mirek,

Two things about the changes:

```
 case WM_PAINT:
  DLOG("WM_PAINT " << Name(this));
```

The above DLOG in Win32Proc.cpp prevents Release mode compilation.

Second: RectTracker no longer works at every drag direction. It used to allow it after:

tracker.MinSize(Size(-100000,-100000));

Well, maybe it works behind the scenes, but it does not show the rect on screen for north or west drag directions.

Best regards,

Tom

Fixed.

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Sat, 18 Apr 2020 15:27:54 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 18 April 2020 18:12Tom1 wrote on Sat, 18 April 2020 16:43Hi Mirek,

Two things about the changes:

```
 case WM_PAINT:
  DLOG("WM_PAINT " << Name(this));
```

The above DLOG in Win32Proc.cpp prevents Release mode compilation.

Second: RectTracker no longer works at every drag direction. It used to allow it after:

tracker.MinSize(Size(-100000,-100000));

Well, maybe it works behind the scenes, but it does not show the rect on screen for north or west drag directions.

Best regards,

Tom

Fixed.

Thanks! RectTracker works beautifully now. And also my inherited LineTracker. :)

BR, Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Sat, 18 Apr 2020 16:21:34 GMT
View Forum Message <> Reply to Message

BTW, tested on R PI 3, seems to work fine (I mean, it is fast enough in UWord, resizing images).

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend

Posted by Tom1 on Sat, 18 Apr 2020 17:21:07 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 18 April 2020 17:48As for windows numbers, looks like Image in Win32 was overengineered, one of optimizations was that the first time it is painted, it paints with SetSurface, only for subsequent paints actually move Image to system handles.

I have now exluded all those things whith #if 0 - looks like it improved at least current RectTracker performance quite nicely...

Mirek

void ImageSysData::Paint(SystemDraw& w, int x, int y, const Rect& src, Color c)

Is this the one (#if 0) you are referring to?

This caused a slowdown of 15-20 % for me. In this case I'm running on Core i7 with integrated Intel HD Graphics 4600 or something...

BR, Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Sun, 19 Apr 2020 12:28:17 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 18 April 2020 19:21mirek wrote on Sat, 18 April 2020 17:48As for windows numbers, looks like Image in Win32 was overengineered, one of optimizations was that the first time it is painted, it paints with SetSurface, only for subsequent paints actually move Image to system handles.

I have now exluded all those things whith #if 0 - looks like it improved at least current RectTracker performance quite nicely...

Mirek

void ImageSysData::Paint(SystemDraw& w, int x, int y, const Rect& src, Color c)

Is this the one (#if 0) you are referring to?

This caused a slowdown of 15-20 % for me. In this case I'm running on Core i7 with integrated Intel HD Graphics 4600 or something...

BR, Tom

Feel free to experiment with it... :) And report results.

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Sun, 19 Apr 2020 20:37:00 GMT
View Forum Message <> Reply to Message

Hi Mirek,

A 4K maximized window Paint from readily available Image/ImageBuffer Size(3840, 2035) comparison:

- Direct SetSurface: 5800 us
- Original DrawImage (1st>2nd>3rd...etc): 10900 us > 35000 us > 5100 us ...
- #if 0 variant (1st>2nd...etc): 18100 us > 6400 us ...
- Original without SetSurface optimization (1st>2nd...etc): 37000 us > 5000 us ...

This is still on Windows 10, Intel Core i7 with integrated Intel HD 4600.

So, SetSurface is always about 5800 us, which is a good all-around solution. #if 0 variant does not really help here. Original code with or without SetSurface optimization only starts to deliver after quite a few rounds when initialization penalty of 35-37 milliseconds is payed back with small gains like 700-800 us per round compared to direct SetSurface.

Maybe computers with better GPUs deliver better with these optimizations. (I don't know this as I do not have such hardware.) Anyway, with typical business setup this is not the case as high end GPUs only come with gaming rigs.

In the end I will probably stick with SetSurface for predictable performance.

On the Linux/GTK3 front the Ctrl::GetPrimaryScreenArea().GetSize(); (used for cache management) still eats half of the time. I think it should be cached in a per-monitor way.

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Mon, 20 Apr 2020 08:31:45 GMT
View Forum Message <> Reply to Message

Worth testing:

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

using namespace Upp;

GUI_APP_MAIN
{
```

```
 DDUMP(GetDeviceCaps(GetDC(NULL), SHADEBLENDCAPS) & SB_PIXEL_ALPHA);
}
```

This should detect whether AlphaBlend is HW accelerated on the platform.... It is 0 on my computer with RX580...

It would also be interesting to test how SW emulation fares, simply by changing

```
 if(0 && fnAlphaBlend() && IsNull(c) && !ImageFallBack &&
```

In general, I think it might be a good idea to "reactivate" SetSurface, but maybe we can do that with some Image hinting system? I do not really like the idea of GetKind anymore, it is from old days when 1024x768 bitmap was considered huge...

gtk3: Have you tested:

```
static Size sz = Ctrl::GetPrimaryScreenArea().GetSize();
```

?

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Mon, 20 Apr 2020 09:13:59 GMT
View Forum Message <> Reply to Message

BTW, my test in win32 show:

SetSurface about 2x faster than call to AlphaBlend (which does not seem to be HW accelerated based on GetDeviceCaps).

However, AlphaBlend (alone) is 2x faster than SW emulation...

Also time to scan Image for test its kind (GetKind) takes about the same time as AlphaBlend function....

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Mon, 20 Apr 2020 10:57:11 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 20 April 2020 11:31Worth testing:

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

using namespace Upp;

GUI_APP_MAIN
{
 DDUMP(GetDeviceCaps(GetDC(NULL), SHADEBLENDCAPS) & SB_PIXEL_ALPHA);
}
```

This should detect whether AlphaBlend is HW accelerated on the platform.... It is 0 on my computer with RX580...

It would also be interesting to test how SW emulation fares, simply by changing

```
 if(0 && fnAlphaBlend() && IsNull(c) && !ImageFallBack &&
```

In general, I think it might be a good idea to "reactivate" SetSurface, but maybe we can do that with some Image hinting system? I do not really like the idea of GetKind anymore, it is from old days when 1024x768 bitmap was considered huge...

gtk3: Have you tested:

```
static Size sz = Ctrl::GetPrimaryScreenArea().GetSize();
```

?

Mirek

Hi

1. DUMP gives me: GetDeviceCaps(GetDC(NULL), SHADEBLENDCAPS) & SB_PIXEL_ALPHA = 0

2. SW emulation (with "if(0 && ...") yields 28-31 milliseconds on a maximized 4k window.

3. On the Linux/GTK3 dept. "static Size sz = Ctrl::GetPrimaryScreenArea().GetSize();" does not work because first call yields zero size... The following calls return the monitor size. The initialization of "static Size sz" should be linked with a valid non-zero size returned.

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Mon, 20 Apr 2020 11:13:37 GMT
View Forum Message <> Reply to Message

Quote:
3. On the Linux/GTK3 dept. "static Size sz = Ctrl::GetPrimaryScreenArea().GetSize();" does not work because first call yields zero size... The following calls return the monitor size. The initialization of "static Size sz" should be linked with a valid non-zero size returned.
Tom

And then it works? :)

Anyway, I am now trying to make some sense / resolution for all the facts presented.

So it looks like while AlphaBlend is NOT HW accelerated, it is still good to use it anyway, as SW emulation is way slower. Now opaque images so far seem to be best handled with SetSurface. However, testing that Image is opaque takes up several ms. OTOH, using SetSurface has a nice advantage that usually large image does not have to go into the cache...

What is the source of large opaque images in your case? Painter or ImageDraw or something else?

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Mon, 20 Apr 2020 12:11:02 GMT
View Forum Message <> Reply to Message

Hi,

This works:

```
void SystemDraw::SysDrawImageOp(int x, int y, const Image& img, Color color)
{
...
 static Size sz;
 if(!sz.cx) sz = Ctrl::GetPrimaryScreenArea().GetSize();
 cache.Shrink(4 * sz.cx * sz.cy, 1000); // Cache must be after Paint because of PaintOnly!
}
```

(Possibly not perfect on multi monitor environments if not started on largest monitor.)

Quote:What is the source of large opaque images in your case? Painter or ImageDraw or something else?

It's nearly always Painter -- or something similar (e.g. custom rendering) but the result is always in ImageBuffer.

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Mon, 20 Apr 2020 13:11:49 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 20 April 2020 14:11
It's nearly always Painter -- or something similar (e.g. custom rendering) but the result is always in ImageBuffer.

I am rather asking because of GetKind. At this point, it appears that it would be nice to have information about whether Image is opaque or not (and use SetSurface or AlphaBlend based on this info). With BufferPainter, I can get this info based on Clear. With generic ImageBuffer, test will be needed, unless client code uses SetKind....

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Mon, 20 Apr 2020 13:25:42 GMT
View Forum Message <> Reply to Message

For custom rendering I can use SetKind(IMAGE_OPAQUE); if that's what it takes to make it fast. :)

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Mon, 20 Apr 2020 13:34:25 GMT
View Forum Message <> Reply to Message

How about putting it readily in constructor:
ImageBuffer(Size sz, int kind=IMAGE_UNKNOWN)
and default to something that always works, while not optimal for all uses.

And also in:
ImageBuffer::Create(Size sz, int kind=IMAGE_UNKNOWN)...
Mostly I know exactly what is going to be found in the buffer anyway...

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Mon, 20 Apr 2020 15:55:38 GMT
View Forum Message <> Reply to Message

I have comitted intended changes for Win32. Some of Image kinds are deprecated and SetKind(IMAGE_OPAQUE) has synonyme Opaque.

Please check whether this is acceptable...

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Mon, 20 Apr 2020 18:37:43 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Win32: Current solution looks good! Probably as good as it can get with GDI.

Linux/GTK3: While SetSurface scores 12 ms times, DrawImage now yields about 5-6 ms results after initial run at 13-14 ms (due to caching).

(There is one RTIMESTOP("cairo_paint"); left in the code.)

I think this has now been taken pretty much as far as it goes with these backends, unless GTK3 still has some optimization available for opaque images.

Thanks and best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Mon, 20 Apr 2020 18:59:23 GMT
View Forum Message <> Reply to Message

Hi,

One Win32 question: Is it really necessary to cache the image if it is actually going to be drawn as a colored rect or via SetSurface?

Best regards,

Tom

[EDIT] Did some testing and this is faster than using SetSurface after caching:

```
void SystemDraw::SysDrawImageOp(int x, int y, const Image& img, const Rect& src, Color color)
{
 GuiLock __;
 if(img.GetLength() == 0)
  return;
 LLOG("SysDrawImageOp " << img.GetSerialId() << ' ' << img.GetSize());

// Insert this optimization here:
 int kind = img.GetKindNoScan();
 if(kind == IMAGE_OPAQUE && !IsNull(color)) {
  Size sz=img.GetSize();
  DrawRect(x, y, sz.cx, sz.cy, color);
  return;
 }
 if(kind == IMAGE_OPAQUE && (GetDeviceCaps(GetHandle(), RASTERCAPS) &
RC_DIBTODEV)) {
  LTIMING("Image Opaque direct set");
  Size sz=img.GetSize();
  SetSurface(*this, x, y, sz.cx, sz.cy, ~img);
  return;
 }
// End of insertion

 ImageSysDataMaker m;
 ...
```

Or is there some other reason to cache the image before display?

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Tue, 21 Apr 2020 14:20:18 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 20 April 2020 20:37

I think this has now been taken pretty much as far as it goes with these backends, unless GTK3

still has some optimization available for opaque images.

Actually, it has cairo_surface_create_similar_image.

I have just finished experimenting with this feature. It is very confusing: It works really well, bringing DrawImage to microseconds time (probably using XRender path), however it has really nasty feature that it kills the drawing performance after DrawImage.

My guess is that it flips cairo into "GPU" mode and then when you want to draw e.g. rectangles, it starts copying memory between GPU and CPU for each element. Or something like that....

For future reference, before I revert them for now, here are changes I have tried:

```
struct ImageSysData {
 Image           img;
 cairo_surface_t *surface = NULL;

 void Init(const Image& m, cairo_surface_t *other);
 ~ImageSysData();
};

cairo_surface_t *CreateCairoSurface(const Image& img, cairo_surface_t *other)
{
 Size isz = img.GetSize();
 cairo_format_t fmt = CAIRO_FORMAT_ARGB32;
 cairo_surface_t *surface = other && 0 ? cairo_surface_create_similar_image(other, fmt, isz.cx,
isz.cy)
                                       : cairo_image_surface_create(fmt, isz.cx, isz.cy);
 cairo_surface_flush(surface);
 byte *a = (byte *)cairo_image_surface_get_data(surface);
 int stride = cairo_format_stride_for_width(fmt, isz.cx);
 for(int yy = 0; yy < isz.cy; yy++) {
  Copy((RGBA *)a, img[yy], isz.cx);
  a += stride;
 }
 cairo_surface_mark_dirty(surface);
 return surface;
}

cairo_surface_t *CreateCairoSurface(const Image& img)
{
 return CreateCairoSurface(img, NULL);
}

void ImageSysData::Init(const Image& m, cairo_surface_t *other)
```

```
{
 img = m;
 surface = CreateCairoSurface(m, other);
 SysImageRealized(img);
}

ImageSysData::~ImageSysData()
{
 SysImageReleased(img);
 cairo_surface_destroy(surface);
}

struct ImageSysDataMaker : LRUCache<ImageSysData, int64>::Maker {
 Image img;
 cairo_surface_t *other;

 virtual int64  Key() const                    { return img.GetSerialId(); }
 virtual int    Make(ImageSysData& object) const { object.Init(img, other); return img.GetLength(); }
};

void SystemDraw::SysDrawImageOp(int x, int y, const Image& img, Color color)
{
 GuiLock __;
 FlushText();
 if(img.GetLength() == 0)
  return;
 LLOG("SysDrawImageOp " << img.GetSerialId() << ' ' << x << ", " << y << ", "<< img.GetSize());
 ImageSysDataMaker m;
 static LRUCache<ImageSysData, int64> cache;
 static int Rsz;
 Rsz += img.GetLength();
 if(Rsz > 200 * 200) { // we do not want to do this for each small image painted...
  Rsz = 0;
  cache.Remove([](const ImageSysData& object) {
   return object.img.GetRefCount() == 1;
  });
 }
 LLOG("SysImage cache pixels " << cache.GetSize() << ", count " << cache.GetCount());
 m.img = img;
 m.other = cairo_get_target(cr);
 ImageSysData& sd = cache.Get(m);
 if(!IsNull(color)) {
  SetColor(color);
  cairo_mask_surface(cr, sd.surface, x, y);
 }
 else {
  RTIMESTOP("cairo_paint");
  if(img.GetKindNoScan() == IMAGE_OPAQUE)
```

```
   cairo_set_operator(cr, CAIRO_OPERATOR_SOURCE);
  cairo_set_source_surface(cr, sd.surface, x, y);
  cairo_paint(cr);
  cairo_set_operator(cr, CAIRO_OPERATOR_OVER);
 }
 static Size ssz;
 if(ssz.cx == 0)
  ssz = Ctrl::GetVirtualScreenArea().GetSize();
 cache.Shrink(4 * ssz.cx * ssz.cy, 1000); // Cache must be after Paint because of PaintOnly!
}
```

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Tue, 21 Apr 2020 16:03:46 GMT
View Forum Message <> Reply to Message

Hi,

Nice! I mean the latest GTK3 DrawImage optimization... Tried it and it feels very appealing and would perform great with these Painter generated scenes. I'm just wondering how well it would work with a typical GUI application workload, if the GUI was first rendered with Painter and then painted to window with this new efficient DrawImage solution. I also wonder if rectangles could possibly be drawn more efficiently using DrawImage with opaque color, since this uses an image mask and should therefore not need to copy it back to CPU accessible memory. I could not benchmark this properly here, but it might be something to look at.

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Tue, 21 Apr 2020 16:28:09 GMT
View Forum Message <> Reply to Message

Ha, I kept digging and in the end found that them problem is caused by InvertColor in DrawDragFrame (probably it is not possible to implement difference operator with Xrandr, so cairo needs to copy memory chunks from/to GPU).

As using that is not mandatory, e.g. alternating white and black dots work as well, I have removed InvertColor and the whole thing started to work as expected. Gtk3 for repeated image draw are now in microseconds range.

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Tue, 21 Apr 2020 17:25:04 GMT
View Forum Message <> Reply to Message

Mirek,

Absolutely brilliant! It works just great! :)

I'm working on a 'SpriteCtrl' and after this improvement it is actually faster with full Paint than with a partial ViewDraw update!!!

I only wish Windows had something similar. As GDI in Windows 10 probably only lives virtually in a layer of software feeding the GPU, I'm getting a strong feeling that we should gradually start looking at Direct2D as it is the MS recommended 2D graphics solution for new applications.

Thanks and best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Didier on Tue, 21 Apr 2020 20:19:17 GMT
View Forum Message <> Reply to Message

Hi,

All this is very interresting.

I just compiled RectTracker projet with svn.14195 (3 weeks old) : I wanted to see the performance difference with new solution
... and it didn't work at all !

After updating to svn.14346, it works fine again :)

I also compiled my current project in which I use my GraphCtrl widget ... and the drag/scroll is completely broken : While draging, I get a completely white widget.
When scrolling is finished, image commes back again.

My usage of Localloop seems to be broken :?

Goiing back to svn.14195 for now

I am under linu, latest Fedora, with an old GPU (Nvidia GT9600)

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Wed, 22 Apr 2020 12:08:14 GMT
View Forum Message <> Reply to Message

Hi,

Here's a completing feature for RectTracker for tracking line drawing (TrackLine) and free-hand polyline (TrackRoute):

```
class LineTracker: public RectTracker{
public:
 int mode;
 Vector<Point> route;

 Vector<Point> TrackLine(const Point& p){
  mode=1;
  route.Clear();
  route.Add(p);
  Track(Rect(p,p),0,0);
  return clone(route);
 }

 Vector<Point> TrackRoute(const Point& p){
  mode=2;
  route.Clear();
  route.Add(p);
  Track(Rect(p,p),0,0);
  return clone(route);
 }

 void MouseMove(Point p, dword){
  if(mode==2 || route.GetCount()<2){
   route.Add(p);
  }
  else route[1]=p;
  Refresh();
  sync(Rect(route[0],route[route.GetCount()-1]));
 }

 void Paint(Draw& w){
  w.DrawImage(0, 0, master_image);
  w.Clip(clip & GetMaster().GetSize());
  int style=width;
  if(pattern!=DRAWDRAGRECT_SOLID){
   Color color2 = IsDark(color) ? White() : Black();
   w.DrawPolyline(route,width,color2);
   switch(pattern){
    case DRAWDRAGRECT_NORMAL:
     style=PEN_DOT;
```

```
     break;
    case DRAWDRAGRECT_DASHED:
     style=PEN_DASHDOT;
     break;
   }
  }
  w.DrawPolyline(route,style,color);
  w.End();
 }

 LineTracker(Ctrl &master): RectTracker(master){
  MinSize(Size(-100000,-100000));
  mode=1;
 }
};
```

Feel free to include it in U++.

To test the functionality, you can try:

```
 void LeftDown(Point p, dword flags){
  LineTracker tracker(*this);
  tracker.Solid();
  tracker.Width(3);
  tracker.SetColor(Blue());

  Vector<Point> res;

  if(flags&K_CTRL) res = tracker.TrackRoute(p);
  else res = tracker.TrackLine(p);
  // ...
 }
```

It proved easy enough to inherit from RectTracker, so I made it a separate class after all.

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Wed, 22 Apr 2020 15:57:26 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 21 April 2020 19:25
I only wish Windows had something similar. As GDI in Windows 10 probably only lives virtually in a layer of software feeding the GPU, I'm getting a strong feeling that we should gradually start looking at Direct2D as it is the MS recommended 2D graphics solution for new applications.

Yes.. Already started "looking". But need to realease first.

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Fri, 24 Apr 2020 15:19:58 GMT
View Forum Message <> Reply to Message

LineTracker is nice, will probably merge it for the next release.

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Fri, 24 Apr 2020 20:43:04 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 24 April 2020 18:19LineTracker is nice, will probably merge it for the next release.
Hi,

Good to hear that, thanks. Meanwhile, let's give it some time to mature. I will re-post it when it has been stable for some weeks while I'm developing it's surroundings. Please remind me if I forget about it...

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Sat, 25 Apr 2020 13:58:24 GMT
View Forum Message <> Reply to Message

BTW, I have just checked situation on MacOS. Current draw path is drawing 4K image in 2.5ms, so I think it is good SW implementation...

EDIT: It is more complicated. It looks like it might be HW accelerated after all...

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Sun, 26 Apr 2020 09:49:01 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Sounds interesting -- despite I do not have a Mac. Just for reference, is this 2.5 ms the same for opaque and alpha images? And which Mac model did you test with?

Best regards,

Tom

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Tue, 28 Apr 2020 07:30:03 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sun, 26 April 2020 11:49Hi Mirek,

Sounds interesting -- despite I do not have a Mac. Just for reference, is this 2.5 ms the same for opaque and alpha images? And which Mac model did you test with?

Best regards,

Tom

Actually, it is is in microseconds range, just like X11, but it is all complicated, some results are now great, but sometimes it gets very slow, needs more investigation.

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by mirek on Tue, 12 May 2020 13:35:55 GMT
View Forum Message <> Reply to Message

I have made some changes to RectTracker:

- Fixed the behaviour when there are some widgets in the view area
- Added TrackLine for free line endpoint tracking
- Optimized Refreshing. This probably might break some code using RectTracker as base, it might need to override RefreshRect method.

Mirek

Subject: Re: MILESTONE: gtk3 replaces gtk2 as default linux backend
Posted by Tom1 on Wed, 13 May 2020 07:44:38 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thanks! It works just fine. :)

Best regards,

Tom