Subject: Should we double-buffer by default?
Posted by mirek on Tue, 18 Jul 2006 13:23:00 GMT
View Forum Message <> Reply to Message

I am in process of refactoring U++ painting engine.

This is quite complicated topic, much more complicated than it seems, if the goal is to avoid flickering e.g. when resizing windows.

There of course is "simple" solution to avoid flickering forever - full window double-buffering.

So far, I stayed away from this, taking the hard approach and using double-buffering just for areas that really need it (namely areas covered by transparent Ctrls).

This leads to very complicated algorithm and the results are questionable. Now I have tried to benchmark the new (improved) painting engine in several modes on both my A64 and venerable testing NT4.0 Celeron@433Mhz with 4MB S3 VGA and 96MB ram. Numbers are time for "repaint the window when resizing" routine:

1. raw mode - no backbuffering:
    A64 - 2.8 ms
    Celeron433 - 90 ms

2. full window backbuffering:
    A64 - 2.4 ms
    Cel433 - 115 ms
note that this mode could be probably further improved a little bit...

3. multirun mode - in this mode, transparent areas are doublebuffered and painted one by one:
    A64 - 8.8 ms
    Cel433 - 266 ms

4. single run mode - single back buffer is created for all transparent area and they are painted in single run:
    A64 - 6 ms
    Cel433 - 145 ms

Looks like trying to be smart does not really pay off...

Now reading Qt docs, since Qt4.0 they do doublebuffering as the only option. Should not we too? (OK, this whole message is for Daniel in the first place...

Mirek

Subject: Re: Should we double-buffer by default?
Posted by mirek on Tue, 18 Jul 2006 13:31:18 GMT

More thoughts:

- maybe I will simply try that - make U++ back-buffering for a couple of weeks and will see. We can always get back to more complicated model

- maybe we could do some form of automatic adaptation - if average repainting time for window breaks some threshold, use direct painting instead, even if it is flickering (I would rather have resizing smooth at the price of flickering than vice versa).

Mirek

Subject: Re: Should we double-buffer by default?
Posted by unodgs on Tue, 18 Jul 2006 13:56:14 GMT

luzr wrote on Tue, 18 July 2006 09:23
Now reading Qt docs, since Qt4.0 they do doublebuffering as the only option. Should not we too?
(OK, this whole message is for Daniel in the first place...

Thank you Mirek

Indeed, to answer if double-buffering should be used as default is not easy. Generaly idea is great and it solves many corner issues with painting.

The main problem with double-buffering is speed of drawing, which is depended mainly on graphics card and the screen resolution. Unofrtunately I always had feeling that apps that are working in full screen in double-buffering mode are less responsive (at least when win32 api (gdi) is used) than "classical" painted ones (even if cpu has 1ghz and modern gfx card like gf4 were used).

I will try your new code today evening and then I'll be able to say more. I think that we should make tests in apps working in maximized modes in resolutions 1024x768 and 1280x1024. In smaller windows there should not be any noticeable difference.

Subject: Re: Should we double-buffer by default?
Posted by mirek on Tue, 18 Jul 2006 14:36:13 GMT

unodgs wrote on Tue, 18 July 2006 09:56
(even if cpu has 1ghz and modern gfx card like gf4 were used).

Just a sidenote - while I do not quite like that kind of argument, I have to mention that gf4 is not "modern gfx". 7xxx is "modern". gf4 is 5 years old...

That said, I am quite happy with double-buffer performance on my 1.2Ghz/GF4 rig....

Mirek

Subject: Re: Should we double-buffer by default?
Posted by unodgs on Tue, 18 Jul 2006 21:43:52 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 18 July 2006 10:36
Just a sidenote - while I do not quite like that kind of argument, I have to mention that gf4 is not "modern gfx". 7xxx is "modern". gf4 is 5 years old...

Modern in sense of momory bandwitch, speed of drawing 2d primitives - not 3d stuff

From my experince I know that in win32 env gdi work faster (and the double-buffering as well) if memory<->gfx transfers are fast.
Eg p41.7 + gf2mx 32mb was faster (I cosider double-buffering mode in upp and opera that uses qt) than duron 1ghz + gf4200Ti (128MB).

Subject: Re: Should we double-buffer by default?
Posted by mirek on Tue, 18 Jul 2006 21:57:15 GMT
View Forum Message <> Reply to Message

I believe that double-buffering does not do memory<->GPU transfers, on decent card with good driver...

Mirek

Subject: Re: Should we double-buffer by default?
Posted by mirek on Tue, 18 Jul 2006 22:42:38 GMT
View Forum Message <> Reply to Message

Reply to ICQ:

> Mirek I don't understand. If theide is now all double-buffered why I can see eg in thisbacks window when scrolling array that first body of array ctrl is scrolled and then child controls

- because scroll is used to scroll window area. Double buffering takes effect later... (yes, I still have "detect Ctrl position change that is scrolled and avoid repainting" in my list. When that will be done, this will disappear).

- also, sometimes (e.g. HeaderCtrl) there is Sync that forces immediate repaint

Mirek

Subject: Re: Should we double-buffer by default?
Posted by unodgs on Wed, 19 Jul 2006 06:09:53 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 18 July 2006 18:42
- because scroll is used to scroll window area. Double buffering takes effect later... (yes, I still have "detect Ctrl position change that is scrolled and avoid repainting" in my list. When that will be done, this will disappear).

I came to the same conclusion thinking about it later

Subject: Re: Should we double-buffer by default?
Posted by unodgs on Wed, 19 Jul 2006 06:34:16 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 18 July 2006 17:57I believe that double-buffering does not do memory<->GPU transfers, on decent card with good driver...

I've found on usenet:

Quote:
Bitmaps created by CreateCompatibleBitmap() are managed by the video driver. They may or may not be in video memory.

John - Microsoft Developer Support

So if the backbuffer is created in system memory (maybe from unknown reasons on my duron it was) gfx<->memory transfer speed is important.

Subject: Re: Should we double-buffer by default?
Posted by mirek on Wed, 19 Jul 2006 07:01:59 GMT

BTW, any idea how to detect a system that is better to flicker than backbuffer?

I was thinking about measuring time of backbuffer paint routine and when it reaches some threshold (50ms), switch to direct flickering mode.

I am however afraid that random time fluctuation would switch decent system as well...

Mirek

Subject: Re: Should we double-buffer by default?
Posted by unodgs on Wed, 19 Jul 2006 08:09:57 GMT

luzr wrote on Wed, 19 July 2006 03:01BTW, any idea how to detect a system that is better to flicker than backbuffer?

I was thinking about measuring time of backbuffer paint routine and when it reaches some threshold (50ms), switch to direct flickering mode.

I am however afraid that random time fluctuation would switch decent system as well...

Unfortunately I have no idea how to detect it (if it is detectable at all) Time measuring seems to be a better idea, but instead of doing it in every Paint() and dynamicaly switch the render mode better would be to do some typical tests at the beginning of the application.

Ok, now lets talk a little about problems with double-buffering.
I tested it mainly in my GridTest application. This is simply app with one tab control and two sheets. In first is grid control, in second array control.
What is intersting in debug mode painting is noticeable slower with double buffering (before it was faster). I checked this on 2 computers (app was maximized):

1. Sempron 64 2.1 ghz, gf6600 1280x1024
2. P4 1.7, gf2mx, 1024x768

In release mode on machine 1 speed was very good, on machine 2 refreshing speed was worst (and it was also worst than in "old" upp rendering mode).

I tested mainly column resizeing. On machine 2 in my grid there is significant lag between mouse position and resized column position. In array ctrl is even worst because header scrolls normaly (no lag) but the body is painted 5-8 pixels behind (I guess this is not synchronized and after fixing it the result will be the same as in my grid).
Anyway in both cases we lost the most important thing: responsivity and feeling of "lightweight" ui.

I also noticed that in my grid ScrollView do nothing (it worked before). The grid consist of 4

frames. Top header, left header , scrollbars and the main frame - grid body.
If scroll was detected the OnScroll() routine from main frame was called. And it looks like this:

```
void GridCtrl::OnScroll()
{
 if(!doscroll)
  return;

 Point newpos(sbx, sby);
 Size delta = oldpos - newpos;

 scrollxdir = delta.cx == 0 ? scrollxdir : delta.cx < 0 ? 1 : -1;
 scrollydir = delta.cy == 0 ? scrollydir : delta.cy < 0 ? 1 : -1;

 if(th.drawColLine || lh.drawRowLine)
 {
  oldpos = newpos;
  return;
 }

 //if(!IsFullRefresh())
 {
  ScrollView(delta);
  if(delta.cx != 0)
  {
   Size sz = th.GetSize();
   th.ScrollView(Rect(fixedWidth, 0, sz.cx, sz.cy), delta.cx, 0);
   scrollLeftRight = true;
  }
  if(delta.cy != 0)
  {
   lh.ScrollView(0, delta.cy);
  }
 }
 oldpos = newpos;
 UpdateCtrlsPos(0, 0, 0, 0, 1);
}
```

I commented if(!IsFullRefresh) to be sure the header will be scrolled. th is top header lh is left header. Unfortunately
th.ScrollView and lh.ScrollView stopped working. Even if I add
lh/th.Sync() nothing changes. Bug? or there are some important changes I should be aware of?

Subject: Re: Should we double-buffer by default?
Posted by mirek on Wed, 19 Jul 2006 09:01:37 GMT

View Forum Message <> Reply to Message

OK, I have tested on my SisGX/Sempron1.8 notebook and while the difference is small, I must say there is difference...

Means back to development.... Now I am thinking about some sort of more simple approach to the old problem.... In fact, the real trouble of all this is "sibling Ctrl intersection". That makes all the trouble, if I want really correct algorithm. Anyway, at the same time it is not very often corner case.

So my next idea is to detect this problem and perform non-buffered draw just for ctrls that do not have this problem. Also, maybe we could detect and handle unbuffered just Ctrls that are "big" (say bigger that 200x200 pixels).

I will test these new ideas tommorow or on Friday.

As for GridCtrl scrolling problem, I think I know where to look and I think it is in CtrlDraw.cpp , but sample code would help, if possible..... (upload to ftp, please).

Mirek

Subject: Re: Should we double-buffer by default?
Posted by unodgs on Wed, 19 Jul 2006 09:18:53 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 19 July 2006 05:01OK, I have tested on my SisGX/Sempron1.8 notebook and while the difference is small, I must say there is difference...

Good I'm not alone
Quote:
Means back to development.... Now I am thinking about some sort of more simple approach to the old problem.... In fact, the real trouble of all this is "sibling Ctrl intersection". That makes all the trouble, if I want really correct algorithm. Anyway, at the same time it is not very often corner case.
So my next idea is to detect this problem and perform non-buffered draw just for ctrls that do not have this problem. Also, maybe we could detect and handle unbuffered just Ctrls that are "big" (say bigger that 200x200 pixels).

Could you describe the problem of sibling ctrl intersections more? (if you have time, I would like to think about solution too)
As for unbuffered draw this is some kind of idea. I would also add to this ability to set by hand if control is or isn't double buffered.
Quote:
As for GridCtrl scrolling problem, I think I know where to look and I think it is in CtrlDraw.cpp , but sample code would help, if possible..... (upload to ftp, please).

I will upload it today evenig - no problem.

Subject: Re: Should we double-buffer by default?
Posted by mirek on Wed, 19 Jul 2006 09:47:18 GMT
View Forum Message <> Reply to Message

Attempt to describe sibling Ctrl intersection problem:

You have opaque Ctrl (can be painted unbuffered) which is intersected by next sibling transparent Ctrl (other part of this sibling Ctrl is above either parent Ctrl or worse, can be above ANOTHER opaque sibling).

Now the correct appearance should be the same as if you paint everything from the first child to the last, doing the same for childs of childs etc... Means next sibling Ctrl should be painted over our opaque Ctrl. Note that you want to backpaint this transparent Ctrl.

This was in U++ (in old and monday/thuesday versions) solved by scanning for all transparent (or non-opaque) areas and back-painting them whith all "bellow" Ctrls (including opaque). The trouble of this approach is that some areas get painted more times.

Mirek

Subject: Re: Should we double-buffer by default?
Posted by mr_ped on Wed, 19 Jul 2006 13:04:31 GMT
View Forum Message <> Reply to Message

There's another "span buffer" technique also as third option, but it's way too different from single/double buffer.

I was using it with my 2D parallax scrolling engine for one never released 2D shooter game, and it allowed me to do things like 10+ parallax layers with ~30 FPS on 133MHz in 640x480, but you gain performance only if lot of areas are redrawed, what IMHO doesn't seem like the case for ordinary GUI controls.
(except transparent controls, where the span buffer will help partially to detect everything hidden behind solid spans (if some controls do overlap), but in the end you will need to draw the solid background and than run as many passes again as many transparent spans are above the solid one, so there's no performance gain)

Another slight advantage is, that with span buffer you can refresh the content of window from top to down line by line, while double-buffering only single line being processed, so you need less memory on graphics card (1 frame buffer + 1 line), but again U++ is so far used on ordinary PC = plenty of memory. (maybe on PocketPC it can be more interesting option...)

But the main disadvantage is you must render everything with spans, so that would mean SW render of anything. Also antialiasing/cleartype would require lot of thinking and improving of the original simple span buffer. Manageable, but too complex.

One final note... I always use only "rectangle" during window resize, the "resize with content

visible" always seems too slow for me.

Subject: Re: Should we double-buffer by default?
Posted by unodgs on Wed, 19 Jul 2006 14:25:41 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 19 July 2006 05:47Attempt to describe sibling Ctrl intersection problem:

You have opaque Ctrl (can be painted unbuffered) which is intersected by next sibling transparent Ctrl.

Lets be more precise. Saying sibling you mean child control or control placed somewhere next to this opaque control (but not involved into parent->child relation).
If you saying intersected you mean that a control is partialy coverd by another control or that another control lies inside a control placed below it or both cases?

Quote:
(other part of this sibling Ctrl is above either parent Ctrl or worse, can be above ANOTHER opaque sibling)

Althought I think I understand it would be nice if you put some screenshot to ilustrate the problem.

Sorry for going so deep into details but I want to be sure that we are speaking the same language and I understand problem correctly.

Subject: Re: Should we double-buffer by default?
Posted by mirek on Wed, 19 Jul 2006 14:43:21 GMT
View Forum Message <> Reply to Message

Sibling -> has the same parent
Intersects -> GetRect for both intersects

See the screenshot.

File Attachments
1) SiblingTrouble.png, downloaded 1590 times

Subject: Re: Should we double-buffer by default?

Posted by fudadmin on Wed, 19 Jul 2006 15:59:15 GMT
View Forum Message <> Reply to Message

I would like to have an alternative "page fipping" mode, as well.
(needs detection if a computer has enough video memory for double size screen). What do you think?

Subject: Re: Should we double-buffer by default?
Posted by mirek on Wed, 19 Jul 2006 16:56:31 GMT
View Forum Message <> Reply to Message

There is no page flipping mode in Win32 GDI or X11.

Mirek

Subject: Re: Should we double-buffer by default?
Posted by fudadmin on Wed, 19 Jul 2006 17:23:56 GMT
View Forum Message <> Reply to Message

...CreateDIBSection technology to do fast animation in Windows. This approach gave the programmer direct access to the bitmap in system memory so that one can use optimized routines for drawing to the bitmap...
Or is that not correct?

Subject: Re: Should we double-buffer by default?
Posted by fudadmin on Wed, 19 Jul 2006 17:38:27 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 19 July 2006 17:56There is no page flipping mode in Win32 GDI or X11.

Mirek

Then we are talking about backbuffer?

Subject: Re: Should we double-buffer by default?
Posted by mirek on Wed, 19 Jul 2006 19:14:29 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Wed, 19 July 2006 13:23...CreateDIBSection technology to do fast animation in Windows. This approach gave the programmer direct access to the bitmap in system memory so that one can use optimized routines for drawing to the bitmap...

Or is that not correct?

Where is page fliping mentioned?

CreateDIBSection is great tool for direct manipulation of RGB values (and is used in Image code), but this if off-topic here.

Mirek

Subject: Re: Should we double-buffer by default?
Posted by fudadmin on Fri, 21 Jul 2006 02:52:47 GMT
View Forum Message <> Reply to Message

What if to write own display driver (miniport etc.) (at least for Windows), use mapping file for video memory? Customizable resolutions?...
WDDM is coming with Vista, as I as understand...
Motion blurr would be good and vsync ...
How do some people do that?:
Quote:
OpenGL API hardware and software support (Microsoft OpenGL, SGI OpenGL, and Mesa
Support for Mesa with full source code
SciTech Game Framework (with source) for creating commercial-quality games
Sprite Library for hardware and software sprite management
Support for hardware triple buffering
Support for stereo LC shutter glasses (requires hardware stereo support)
Improved performance for many low level rasterization functions
Highly optimized 32 bit assembler rasterization for maximum speed
Full hardware and software double/multi-buffering support
Hardware scrolling/panning surfaces
Rendering direct to video memory, off-screen video memory and to system memory buffers
Full linear surface virtualization under DOS and Windows
Real-time 8 bit dithering

http://www.scitechsoft.com/products/dev/mgl_home.html

Subject: Re: Should we double-buffer by default?
Posted by mirek on Fri, 21 Jul 2006 08:44:47 GMT
View Forum Message <> Reply to Message

Come on. We have abandoned idea of software rendering half year ago - (let me remind the final conclusion - the trouble is not to render things in the memory buffer, trouble is that some machines are too slow when moving content of this memory buffer to the screen - and there is nothing that can be done about it).

Mirek

Subject: Re: Should we double-buffer by default?
Posted by mirek on Fri, 21 Jul 2006 10:10:41 GMT
View Forum Message <> Reply to Message

OK, so opaque painting is back. It is however much more lightweight code, using "bad case" detection.

Mirek
