Subject: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by kohait00 on Wed, 26 May 2010 11:15:12 GMT
View Forum Message <> Reply to Message

hi there people

i am dealing with a performance issue, tackling the borders of Upp (or maybe better: my mind

got a MT environment wich diversly generates a LOT of frequent refreshes on a LOT of small Ctls.

example:

some 5+ Threads (communication from devices) generate visual data for ~25 Views (with a LOT of Ctrl hierarchy) every 200 ms for each view, each data element causes a refresh of a particular Ctrl (maybe somewhere *deep* in hierarchy).

so the problem is the perfomance is *sh..*, no fluidity

>> How to deal with a lot of frequent updates / refreshes to a lot of Ctrls from several threads (!= main GUI thread)?

i imagine the problem is my design itself, where the threads use GuiLock to do the stuff in the Gui or need to wait for MainThread to finish the work for them and so there is a lot of context switches.. or the Ctrls are pretty deep in hierarchy and refreshing them toggles refreshes on upper ones as well?

is there a possib to disable a view for repainting, have all the update ctrl stuff done and then by enabling trigger a refresh in Main GUI thread for all *dirty* Ctrl (which would be done without context switches)

help apriciated

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by dolik.rce on Wed, 26 May 2010 14:45:04 GMT
View Forum Message <> Reply to Message

Hi kohait,

Just a few ideas:

You can try to update the GUI using PostCallback rather then GuiLock. It sends callback to main thread and gets executed there. I believe it could speed things a bit.

Alternatively, it might (I did not test this) help a little bit to just change the GUI without refresh and then call refresh from main thread periodically.

Best regards,
Honza

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by mr_ped on Wed, 26 May 2010 14:55:45 GMT
View Forum Message <> Reply to Message

GuiLock works in similar way like PostCallback?!

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by Mindtraveller on Wed, 26 May 2010 15:20:32 GMT
View Forum Message <> Reply to Message

1. By the way, is GUI MT-safe? If it is so, maybe it is a good idea to have some flag (like GUI_MT) to enable/disable GUI-MT features? This will speed things too.

2. Maybe you could make some descendant class of ParentCtrl which keeps cached image and drws it upon refresh. Actual refresh of child Ctrl is to be made only on explicit request (let's call it ParentCtrlCache). I don't know exactly (didn't test it) but this could give additional speedup by reducing actual GUI updates.

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by sergeynikitin on Wed, 26 May 2010 16:14:54 GMT
View Forum Message <> Reply to Message

I propose to decide: How frequently refreshing needed?
Usually I update Ctrl only one times per second, And ignore more frequently updates. But you can decide to refresh more frequently.

It makes no sense to update the widget and spend system resources on it, more often than it is able to notice the eyes.

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by kohait00 on Wed, 26 May 2010 16:43:41 GMT
View Forum Message <> Reply to Message

thanks guys for replys.

the app is kind of an audio devices monitoring tool with meterbridges data coming in as push data (i dont have control over time when it comes in). it should display kind of fluidly, 200 ms is alright, but if you imagine 25+ devices firing thir live data, that comes in at random time in different threads, there it's the mess.

Quote:
It makes no sense to update the widget and spend system resources on it, more often than it is able to notice the eyes.

i'd need even 1/20 = 50 ms refreshtime..so 200ms is a compromise already

Quote:
is GUI MT-safe?

as far as i know GUI "is" MT safe, as long as you use GuiLock __; scope helper in the other threads to modify the gui.. internally it maps some stuff to Main Thread and waits for it anyway. but not everythin in GUI is MT, like DnD stuff etc.. dig in the CtrlCore win32 code, and you will find some ASSERT(IsMainThread).

Quote:
ParentCtrl which keeps cached image and drws it upon refresh

this would increase memory consumptiion for rather huge ctrl's, but is an option. but what about all the small ctrls already present? it would be a lot of work to adjust them

Quote:
GuiLock works in similar way like PostCallback?!

no, not quite the same. PostCallback gets executen in the same TimerThread, while the GuiLock is only a mutex to aquire GUI use rights for your own different thread (!= main)

Quote:
little bit to just change the GUI without refresh and then call refresh from main thread periodically.

this was my intention, but i dnot know of any means in Ctrl to update a Ctrl (using its API, SetData() etc) without causing an internal Refresh(). setting data to controls always results in an Refresh(), unlike i.e. setting Ctrl properties like maybe font, color, min max etc..this needs a manual Refresh() as far as i know. this all is by design. Ctrls distinguish in their between "parametrising" a Ctrl (setting it up) which does *not* trigger a Refresh() automatically, and "using, providing actual data" to Ctrl, which automatically Triggers a Refresh().
Now I am "using" my Ctrls, so they refresh that frequently..

it would be great to "postpone" the Refresh() of a Ctrl for later.. but donno how to do it, or if the Ctrl API already supports it

---

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by Mindtraveller on Wed, 26 May 2010 17:57:03 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Wed, 26 May 2010 20:43i'd need even 1/20 = 50 ms refreshtime..so 200ms is a

compromise
Then, believe me, you MUST use OpenGL/DirectX for this. No alternatives.

The frame rate you require is beyond GUI abilities (this means that there is no way to guarantee that operating system will refresh window with rate you required). This is especially important for POSIX/X11 systems.

So, graphics acceleration is your only choice here.
And - if you wish to use accelerated control like DHCtrl (I don't remember exact name), please consider using ONE accelerated control per application, not many.

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by dolik.rce on Thu, 27 May 2010 07:20:21 GMT
View Forum Message <> Reply to Message

Hi Kohait,

If you develop some kind of graphics/video/game application where framerate matters, then you should use graphic acceleration as Mindtraveller suggested. But if you just want to show some rapidly changing data in real time, you should be fine with some 5 fps max, user don't react any quicker

Here is something that might or might not help you:#include <CtrlLib/CtrlLib.h>
using namespace Upp;

```
template <class T>
class Cached : public T{
 Value val;
 bool refreshflag;
public:
 void operator<<=(Callback action) {T::operator<<=(action);}
 void operator<<=(Value data)      {refreshflag=true; val=data;}
 void SetData(Value data)          {refreshflag=true; val=data;}
 Value GetData()                   {return val;}
 Value operator~()                 {return val;}
 void Apply()                      {if(refreshflag) T::SetData(val);}
 bool IsChanged()                  {return refreshflag;}
};

class guitest : public TopWindow {
public:
 typedef guitest CLASSNAME;
 Cached<EditIntSpin> s;
 guitest(){
  s.SetRect(0,0,50,24);
  Add(s);
  s<<=0;
```

```
  s.Apply();
 }
 void LeftDown(Point p,dword keyflags){
  s<<=int(~s)+1;
 }
 void RightDown(Point p,dword keyflags){
  s.Apply();
 }
};

GUI_APP_MAIN{
 guitest().Run();
}
```

It is a simple wrapper template that should work on any Ctrl overloading it's SetData and GetData methods, so they don't trigger Refresh. It will help you only if: You just change the data, not the Ctrls.
Your app has some idea about the hierarchy of Ctrl so it can call Apply on all of them when needed.To update I would use single function called using SetTimeCallback with reasonable interval, let's say 200ms.

Best regards,
Honza

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by kohait00 on Thu, 27 May 2010 21:36:52 GMT
View Forum Message <> Reply to Message

thanx guys, i'll spend some time thinking about it.. maybe there is also an option to use a "globaly accessible" data cache, where to deposit data, that frequently comes in, and then, independantly trigger refresh operations, that take the data and forward it to the controls which would refresh then in one.., thus decoupling the refreshment if visual data itself from the data that comes in.

but anoter general question:

how to best update a whole LOT of controls at once (just once, not frequently, but at once) without causing the GUI to repaint each time in different locations..but at the end maybe to repaint all? maybe thats the way to go as well..

the OpenGL option is worth thinking about..

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by koldo on Fri, 28 May 2010 06:12:09 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Thu, 27 May 2010 09:20Hi Kohait,

If you develop some kind of graphics/video/game application where framerate matters, then you should use graphic acceleration as Mindtraveller suggested. But if you just want to show some rapidly changing data in real time, you should be fine with some 5 fps max, user don't react any quicker

Here is something that might or might not help you:#include <CtrlLib/CtrlLib.h>
using namespace Upp;

```
template <class T>
class Cached : public T{
 Value val;
 bool refreshflag;
public:
 void operator<<=(Callback action) {T::operator<<=(action);}
 void operator<<=(Value data)      {refreshflag=true; val=data;}
 void SetData(Value data)          {refreshflag=true; val=data;}
 Value GetData()                   {return val;}
 Value operator~()                 {return val;}
 void Apply()                      {if(refreshflag) T::SetData(val);}
 bool IsChanged()                  {return refreshflag;}
};

class guitest : public TopWindow {
public:
 typedef guitest CLASSNAME;
 Cached<EditIntSpin> s;
 guitest(){
  s.SetRect(0,0,50,24);
  Add(s);
  s<<=0;
  s.Apply();
 }
 void LeftDown(Point p,dword keyflags){
  s<<=int(~s)+1;
 }
 void RightDown(Point p,dword keyflags){
  s.Apply();
 }
};

GUI_APP_MAIN{
 guitest().Run();
}
```

It is a simple wrapper template that should work on any Ctrl overloading it's SetData and GetData methods, so they don't trigger Refresh. It will help you only if: You just change the data, not the Ctrls.

Your app has some idea about the hierarchy of Ctrl so it can call Apply on all of them when needed.To update I would use single function called using SetTimeCallback with reasonable interval, let's say 200ms.

Best regards,
Honza

I like this sample .

And you told you are not a programmer...

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by Sender Ghost on Fri, 28 May 2010 10:53:22 GMT
View Forum Message <> Reply to Message

Hello, Jan.

Another sample:

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

using namespace Upp;

template <class T>
class CachedCtrl : public T {
private:
 Value value;
 bool change;
public:
 CachedCtrl() : change(false) { }
 Callback operator<<=(Callback action) { return T::operator<<=(action); }
 void operator<<=(const Value& data) { change = true; value = data; }
 void SetData(const Value& data) { change = true; value = data; }
 Value GetData() const { return value; }
 Value operator~() const { return value; }
 void Apply() { if (change) T::SetData(value); }
 bool IsChanged() { return change; }
};

const int NUM_CTRLS = 10,
 REFRESH_RATE = 50; // ms

class App : public TopWindow {
private:
 bool doing;
public:
```

```
 typedef App CLASSNAME;
 App();
 ~App();

 Thread work;
 typedef CachedCtrl<EditInt> CEditInt;
 Array<CEditInt> ctrls;

 void ChangeData();
 void UpdateData();
 void LeftDown(Point p, dword keyflags);
 void RightDown(Point p, dword keyflags);
};

App::App() : doing(false)
{
 Title("CachedCtrl test application");
 CenterScreen().Sizeable().MinimizeBox().MaximizeBox();
 SetRect(Size(640, 480));

 for (int i = 0; i < NUM_CTRLS; ++i)
 {
  ctrls.Add().HSizePosZ(4, 4).TopPosZ(4 + i*(19 + 4), 19).SetData(i + 1);
  ctrls[i].Apply();
  Add(ctrls[i]);
 }
}

App::~App()
{
 Thread::ShutdownThreads();
 work.Wait();
}

void App::ChangeData()
{
 if (!doing) doing = true;
 else return;

 const PaintRect curRect = GetBackground();
 Background(PaintRect(ColorDisplay(), SColorPaper()));
 work.Run(THISBACK(UpdateData));

 while (doing)
 {
  Sleep(1);
  GuiLock __;
  if (Thread::IsShutdownThreads()) break;
```

```
  for (int i = 0; i < NUM_CTRLS; ++i)
  {
   ctrls[i].SetData(int(~ctrls[i]) % (NUM_CTRLS * 100) + 1);
  }
 }

 Background(curRect);
 doing = false;
}

void App::UpdateData()
{
 while (doing)
 {
  Sleep(REFRESH_RATE);
  GuiLock __;
  if (Thread::IsShutdownThreads()) break;

  for (int i = 0; i < NUM_CTRLS; ++i)
  {
   ctrls[i].Apply();
  }
 }
}

void App::LeftDown(Point p, dword keyflags)
{
 doing = !doing;
}

void App::RightDown(Point p, dword keyflags)
{
 work.Run(THISBACK(ChangeData));
}

GUI_APP_MAIN
{
 Ctrl::GlobalBackPaint();

 App app;
 app.Run();
}
```

But may be here needed manual refresh mode for all Ctrls instead of automatic (fullrefresh). No need to store extra data. Also IsChanged() equals IsModified() in last case.

For example, in .Net Framework:

```
// Stop refreshing
ctrl.BeginUpdate();
// Change data
// ...
// Start refreshing again
ctrl.EndUpdate();
```

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by kohait00 on Fri, 11 Jun 2010 12:11:57 GMT
View Forum Message <> Reply to Message

the sample from dolik is interesting, it intercepts the data change stuff, nice idea. tnhanks.

the only drawback is, that there are several controls where you dont only have GetData/SetData which triggers the refresh, like Label or Static or sth.. SetText(), SetLabel()...

it'd be great to have kind of a more general way of disabling refresh in entire gui for some time..

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by dolik.rce on Fri, 11 Jun 2010 14:06:41 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Fri, 11 June 2010 14:11the sample from dolik is interesting, it intercepts the data change stuff, nice idea. tnhanks.

the only drawback is, that there are several controls where you dont only have GetData/SetData which triggers the refresh, like Label or Static or sth.. SetText(), SetLabel()...

it'd be great to have kind of a more general way of disabling refresh in entire gui for some time..

Hi Kohait,

I admit I didn't think about that. One usually expects Static widgets to be, uhm, kind of static

Anyway, if you are changing those as well, you can use template specialization. E.g. for label:

```
#include <CtrlLib/CtrlLib.h>
using namespace Upp;

template <class T>
class Cached : public T{
 Value val;
```

```
 bool refreshflag;
public:
 void operator<<=(Callback action) {T::operator<<=(action);}
 void operator<<=(Value data)      {refreshflag=true; val=data;}
 void SetData(Value data)          {refreshflag=true; val=data;}
 Value GetData()                   {return val;}
 Value operator~()                 {return val;}
 void Apply()                      {if(refreshflag) T::SetData(val);}
 bool IsChanged()                  {return refreshflag;}
};
template <>
class Cached<Label> : public Label{
 Value val;
 bool refreshflag;
public:
 void operator<<=(Value data)      {refreshflag=true; val=data;}
 void SetData(Value data)          {refreshflag=true; val=data;}
 Value GetData()                   {return val;}
 Value operator~()                 {return val;}
 void Apply()                      {if(refreshflag) Label::SetLabel(AsString(val));}
 bool IsChanged()                  {return refreshflag;}
};

class guitest : public TopWindow {
public:
 typedef guitest CLASSNAME;
 Cached<EditIntSpin> s;  Cached<Label> l;
 guitest(){
  s.SetRect(10,10,50,24); l.SetRect(10,40,100,54);
  Add(s);   Add(l);
  s<<=0;   l<<="Nothing yet";
  s.Apply();  l.Apply();
 }
 void LeftDown(Point p,dword keyflags){
  s<<=int(~s)+1;  l<<=String("Last value: ")+IntStr(s.EditIntSpin::GetData());
 }
 void RightDown(Point p,dword keyflags){
  s.Apply();  l.Apply();
 }
};

GUI_APP_MAIN{
 guitest().Run();
}
```

This will unify the interface between usual widgets and Label. Of course you could also make a different template that would keep the SetLabel interface and just added the caching capabilities. That is up to your choice

BTW: I guess you don't need it, but I quite like the possibility to access the actual value displayed by the control, regardless on what is in cache. So I put it into the example code as a little bonus

Best regards,
Honza

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by kohait00 on Fri, 11 Jun 2010 15:19:10 GMT
View Forum Message <> Reply to Message

thanks dolik, nice idea with templates
i think i generally need to rethink my concept of propagation of data. the aspecht of refresh 5 times a sec. above from Mindtraveler is a word.. maybe i can dig in code to make some kind of switch to disable refreshing while updating controls, and then enabling it again and trigger some deep RefreshLayout..

i'll pass with you here for more info and ideas guys

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by kohait00 on Tue, 15 Jun 2010 13:00:33 GMT
View Forum Message <> Reply to Message

hey folks

the problem is still there and i cant handle it properly because the basic question is not solved yet:

?? how to Update/Refresh a LOT of different small Ctrl's in as little time as possible ???.

as far as i understand there is (for Win32 at least) the ::InvalidateRect functions called in a Refresh(), invoking a WM_PAINT message somewhere. now if shortly after this a *second* Ctrl is Updated/Refreshed, it will cause the same procedure..or is it accumulated and in some time later, the WM_PAINT would outline a rect to repaint, that contains *both* dirty Ctrl's..

i think this is a very basic question. without knowing this behaviour i cant say to wich direction to optimize the gui handling. maybe this should even go into documentation: for performance reasons..

maybe there should be a static flag which is normally on and enables refresh by default, turning it off can then give the possibility to update a bunch of ctsl at once and then reenable normal operation and call a deep refresh.. i thinnk this would be faster..

what do you think?

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by Sender Ghost on Tue, 15 Jun 2010 20:18:30 GMT
View Forum Message <> Reply to Message

Hello, Konstantin.

Another possibility is changing Ctrl class.
You can start with Ctrl::RefreshFrame method(s).

```
void Ctrl::RefreshFrame(const Rect& r) {
 GuiLock __;
 if (manualRefresh || !IsOpen() || !IsVisible() || r.IsEmpty()) return;
```

Add switches in private area of Ctrl:

```
private:
 bool manualRefresh;
```

in public area:

```
public:
 bool IsUpdate() const { return manualRefresh; }
 void BeginUpdate() { manualRefresh = true; }
 void EndUpdate() { manualRefresh = false; RefreshFrame(); }
```

Default value in Ctrl constructor:

```
Ctrl::Ctrl() {
 GuiLock __;
 manualRefresh = false;
```

And use in some threading function:

```
 Array<Ctrl *> ctrls;
 // Adding needed ctrls to update
 // ...
 // Begin updating of ctrls
 for (int i = 0, n = ctrls.GetCount(); i < n; ++i)
  ctrls[i]->BeginUpdate();
 // Changing data
 // ...
 // End updating of ctrls
 for (int i = 0, n = ctrls.GetCount(); i < n; ++i)
  ctrls[i]->EndUpdate();
```

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by mirek on Wed, 16 Jun 2010 05:26:21 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 15 June 2010 09:00hey folks

the problem is still there and i cant handle it properly because the basic question is not solved yet:

?? how to Update/Refresh a LOT of different small Ctrl's in as little time as possible ???.

as far as i understand there is (for Win32 at least) the ::InvalidateRect functions called in a Refresh(), invoking a WM_PAINT message somewhere.

InvalidateRect does not invoke WM_PAINT directly.

In either case, in Win32 or X11 (where it has to be implemented by u++), Paint is always as lazy as possible - it only happens when input queue is empty (unless you request immediate repaint using Sync).

Quote:what do you think?

I think we have spent years optimizing Refresh/Paint. You can only make it worse:)

Simply do not worry. U++ will do the best possible.

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by mirek on Wed, 16 Jun 2010 05:27:20 GMT
View Forum Message <> Reply to Message

Sender Ghost wrote on Tue, 15 June 2010 16:18Hello, Konstantin.

Another possibility is changing Ctrl class.
You can start with Ctrl::RefreshFrame method(s).

General word of advice: Do not be so eager to change U++ sources. Some time later you can find yourself standing on diverging platform

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by kohait00 on Wed, 16 Jun 2010 07:56:49 GMT
View Forum Message <> Reply to Message

hy guys

Quote:
 if (manualRefresh || !IsOpen() || !IsVisible() || r.IsEmpty()) return;

this is exactly what i had been thinking of.

Quote:
InvalidateRect does not invoke WM_PAINT directly.

In either case, in Win32 or X11 (where it has to be implemented by u++), Paint is always as lazy as possible - it only happens when input queue is empty (unless you request immediate repaint using Sync).

Quote:

what do you think?

I think we have spent years optimizing Refresh/Paint. You can only make it worse:)

Simply do not worry. U++ will do the best possible.

now this is a word. this exactly answers my questions..and creates others. how to keep the queue busy so ith wont trigger a refresh each and everytime.

imagine: 50 devices push their live monitoring data (acutally not the problem, 48 bytes only) to the gui, where the data income is linked to Ctrls (SetData, SetText, cusom Ctrls as well) for each device separately. the devices fire unsynchronously, more or less every 200 ms.
==> consequence: Refreshes over and over (even with 4 devices its a problem already), because queue is kept empty..any idea about that?

Quote:
Another possibility is changing Ctrl class.

this is what *I* want the *least*. i know of upp beeing prepared for any cases and ooptimized to the edge. i'd rather need to rethink my model.

cheers

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by dolik.rce on Wed, 16 Jun 2010 09:18:37 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Wed, 16 June 2010 09:56how to keep the queue busy so ith wont trigger a refresh each and everytime.

imagine: 50 devices push their live monitoring data (acutally not the problem, 48 bytes only) to the gui, where the data income is linked to Ctrls (SetData, SetText, cusom Ctrls as well) for each

device separately. the devices fire unsynchronously, more or less every 200 ms.
==> consequence: Refreshes over and over (even with 4 devices its a problem already), because queue is kept empty..any idea about that?

Hi Kohait,

Thanks for better description of the problem. I still think that this is not a problem of how to refresh Ctrls... It is about the app design. You can solve your problems by altering it just a little bit.

What you need is not to feed the incoming data directly into GUI. If I understand correctly, you know very well what is coming from "outside", so it should not be a problem to create some structure representing those 48b of information. Doing this of the screen is very fast. Now all these structures, one for each device, can reside in Vector. If you use one thread per device there should not be no problem with MT.

Then you can just create a function that will iterate through this array and update the Ctrls based on its content. This function can be called using PostCallback. To make it really MT safe you can use some kind of locking (to prevent writing into the structures or to pause receiving new data). If you refresh "only" 10-20 times per second (which is more than enough for human eye, I usually aim for ~5fps), it should not bring any performance problems.

Well, at least that is what I would do

Best regards,
Honza

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by cbpporter on Wed, 16 Jun 2010 09:23:32 GMT
View Forum Message <> Reply to Message

Maybe your solution is a combination of the ideas in this thread. Can't you update only data in your threads and update your entire GUI at a fixed interval? Using a MVC model. You update your Model only. If you need to update a label and are worried about the label Refresh, update a string and in you "update view" method set that label. But you must make sure that your total number of updates in a given period of time is less than it would be if the threads would update at their own pace. Otherwise you wont get any performance benefit and your code will become very complex.

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by Sender Ghost on Wed, 16 Jun 2010 10:05:04 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 16 June 2010 07:27
General word of advice: Do not be so eager to change U++ sources. Some time later you can find yourself standing on diverging platform :)

Hello, Mirek.

Well, I can understand your points: merging of view and controller together (from Model-View-Controller perspective) to simplify porting problems.
My points for Konstantin is just for possible path to start thinking about (what functions calls RefreshFrame, which called by it, etc.). After understanding this path can be optimized. For example, some Ctrl functions can be virtual and overriden by inherited class functions. But for now, this solved partially and not in all cases.

The main problem here, I think, is bottleneck and not only with Refresh, but also with functions called by SetData (which showed previous Jan example with cached data). Yes, in current state this bottleneck can be as minimal as possible.

So, I tried to solve the problem first (currently in simplified form), but not to change public sources without strong reason.

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by kohait00 on Wed, 16 Jun 2010 13:43:58 GMT
View Forum Message <> Reply to Message

thanks guys for the ideas. i think there is no other option, i will do it like that.

another more general questions arises:

my gui view for each device comes from an xml file, where i use ParentCtrl to invisibly group other Ctrls to be able to specify their common offsets, positions, extents, etc generally easy arange them. since its an audio device application there is quite a lot ctrls invoved and i use quite a lot ParentCtrl to organize them.

??? does the Ctrl tree depth matter ??? (i imagine yes).

imagine a deph of easily 20-25 Ctrls, so if the downmost (actually opaque ctrl) is updated/refresht, >>> does the refresh have to recompute *all* the tree path (i.e. from topmost, parent, forwarding the dirty rect to the children or something. i imagine this to happen when WM_PAINT gets executed, one needs to determine which ctrl is actually to be drawn. as i remember there is code for determining exactly this, which ctrl lies in the dirty rect)?

this could be another bottle neck. couldnt one speed up the search top-down by "marking" the "repaint" path bottom-up with a flag?

better explain:
a Ctrl's Refresh() marks itself as dirty target and marks the parents in path up to top as the path to go for the dirty Ctrl, when a WM_PAINT is received, one only needs to check which ctrl in the tree are marked...

but sure the current way is not bad anyway..some short explanations of interna would really help..

cheers

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by Sender Ghost on Wed, 16 Jun 2010 15:17:33 GMT
View Forum Message <> Reply to Message

Please, try to uncomment LLOG(x)/LTIMING(x) defines on the top of uppsrc/CtrlCore/CtrlDraw.cpp file to see some behaviour in the application logs compiled in debug mode.
Also, you can use Ctrl::ShowRepaint static function to see this visually with prefered refresh rate in milliseconds (e.g. with (Transparent/FullBack)Paint states for Ctrls globally activated by Ctrl::GlobalBackPaint static funciton):

```
GUI_APP_MAIN
{
 Ctrl::GlobalBackPaint(true);
 Ctrl::ShowRepaint(50); // 50 ms = 20 fps
```

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by kohait00 on Wed, 16 Jun 2010 15:26:05 GMT
View Forum Message <> Reply to Message

i'll try it..

do you have some hin concerning the repaint modell in upp? how is it done, what is the idea behind it?

thx

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by Sender Ghost on Wed, 16 Jun 2010 16:47:13 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Wed, 16 June 2010 17:26
do you have some hint concerning the repaint model in upp? how is it done, what is the idea behind it?

Not exactly, except what I said before about MVC and cached data.
We can see following words from documentation for Ctrl::RefreshFrame:
Quote:
Marks requested rectangle of frame area for repainting. Actual repainting is deferred for

performance reasons.

When application window moves or some application window(s) moves along it repaints appeared areas.
Also Mirek said about input queue.

May be, following caller graph diagram created by Doxygen and Graphviz programs for Ctrl::RefreshFrame (from CtrlCore perspective only) can be useful:

Actual painting (e.g. for Windows operating system) happens in virtual Ctrl::WindowProc function:

```
LRESULT Ctrl::WindowProc(UINT message, WPARAM wParam, LPARAM lParam)
{
//...
 HWND hwnd = GetHWND();
 switch(message) {
//...
 case WM_PAINT:
  ASSERT(hwnd);
  if(IsVisible() && hwnd) {
   PAINTSTRUCT ps;
   SyncScroll();
   HDC dc = BeginPaint(hwnd, &ps);
   fullrefresh = false;
   SystemDraw draw(dc);
//...
   UpdateArea(draw, Rect(ps.rcPaint));
//...
   EndPaint(hwnd, &ps);
  }
  return 0L;
```

File Attachments

1) RefreshFrame_CallGraph.png, downloaded 643 times
2) RefreshFrame_CallGraph_(small).png, downloaded 908 times

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by Sender Ghost on Mon, 21 Jun 2010 14:52:32 GMT
View Forum Message <> Reply to Message

As conclusion, I created following example to demonstrate this problem and solution.

On the left side you can see normal behaviour of Upp application with about 100% CPU load

periodically.
On the right side you can see reduced CPU load with using CachedCtrl class template and specified update rate.

The left and right sides of application can be activated with right mouse click and deactivated with left mouse click.

With Array<CachedCtrl<Ctrl> *> this example can be changed to support more types of Ctrls.

The source code of example attached to this message.

File Attachments
1) CachedDataScreen.png, downloaded 912 times
2) CachedData.zip, downloaded 315 times

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by kohait00 on Tue, 22 Jun 2010 09:24:58 GMT
View Forum Message <> Reply to Message

thanks for the example.

i indeed could see performance wins applying the overall approach. so together with the example, we could consider this problem to be solved to a satisfactory degree.

i just try to sum up the things we learned here:

PROBLEM:
frequent, independant refreshes to a lot of Ctrl's (deep in tree or not) caused by different threads than the Main thread produce a high CPU load, becaause the Ctrl's are indeed refreshed one by one (or at least almost). additionally, the threads passing the refresh work over to the MainThread (using ICall interface etc.) are beeing slept for that time. -> main thread invokes Refresh just too often, because the Updates (that trigger the Refresh) are independant.

SOLUTION:
separate the data Update (SetData, SetText, SetLabel, etc) of the Ctrl's from the natural / automatic triggering of Refresh() (inside Ctrl's, unaccessible), by:

* using a cached class like the above (if you only use Get/SetData). it overloads the Get/SetData() and prevents the triggering of Refresh each time..

* (more complex) providing a caching database independant of Get/SetData() (or other Interface functions).

both solutions conclude with a periodic Refresh() call for all (meanwhile) affected/changed Ctrl's. Refresh time does not need to be faster than 200 ms.

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by kohait00 on Tue, 22 Jun 2010 09:45:03 GMT
View Forum Message <> Reply to Message

for all those who are having app CRASHES when exiting the application, while a timer is running (triggering the refreshes on Ctrl's that dont exist anymore because of deletion). here an example solution with an Atomic variable, a bool flag and wait for an atomic state. any hints / advices / corrections welcome. so far it works for me.

sorry its just copy and paste from my application, with only some changes/deletes, but i hope the idea will be clear enough.

```
class ASPDevice
{
...
	~ASPDevice() { DemoMode(false); } //will wait for timer to exit clean
 Atomic demo; //0=disabled, 1=running demo mode, 2=a timer cb is executing
	bool isdemo;
	IsDemoMode() const { return isdemo; }
...
}

void ASPDevice::DemoMode(bool b)
{
 bool _b = isdemo;
 isdemo = b;
 if(!_b && isdemo) //start only one time, maybe here is a problem still, because not atomic
 {
  ASSERT(AtomicInc(demo)==1);
  SetTimeCallback(0, THISBACK(GenDemoData), 2);
 }
 else if(_b && !isdemo)
 {
  KillTimeCallback(2); //very important, timer might be not executin currently
  AtomicDec(demo);
  while(AtomicRead(demo) > 0) Sleep(1);
  ASSERT(AtomicRead(demo)==0);
 }
}

void ASPDevice::GenDemoData()
{
 ASSERT(AtomicInc(demo)==2);

...
//do your refresh stuff here, dont forget an AtomicDec(demo) on error exits
...
```

```
 int a = AtomicDec(demo);
 if(a<=0) return; //if has been desabled, dont restart
 ASSERT(a>=0);
 ASSERT(IsDemoMode());
 SetTimeCallback(200, THISBACK(GenDemoData), 2); //inkl a jitter
}
```

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by tojocky on Tue, 22 Jun 2010 11:05:22 GMT
View Forum Message <> Reply to Message

Sender Ghost wrote on Mon, 21 June 2010 17:52As conclusion, I created following example to demonstrate this problem and solution.

On the left side you can see normal behaviour of Upp application with about 100% CPU load periodically.
On the right side you can see reduced CPU load with using CachedCtrl class template and specified update rate.

The left and right sides of application can be activated with right mouse click and deactivated with left mouse click.

With Array<CachedCtrl<Ctrl> *> this example can be changed to support more types of Ctrls.

The source code of example attached to this message.

Nice solution!

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by Sender Ghost on Tue, 22 Jun 2010 13:36:46 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 22 June 2010 11:24
thanks for the example.

tojocky wrote on Tue, 22 June 2010 13:05
Nice solution!
Thanks. Main credits to Jan.
kohait00 wrote on Tue, 22 June 2010 11:24
additionally, the threads passing the refresh work over to the MainThread (using ICall interface etc.) are beeing slept for that time. -> main thread invokes Refresh just too often, because the

Updates (that trigger the Refresh) are independant.

Seems, there exists some race condition to access Handle of Device Context (HDC), at least on Windows operating system.
kohait00 wrote on Tue, 22 June 2010 11:24
Refresh time does not need to be faster than 200 ms.

In particular example, refresh (update) rate can be set to 0 ms (or by using PostCallback).

Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by kohait00 on Tue, 22 Jun 2010 15:36:00 GMT
View Forum Message <> Reply to Message

Quote:
(or by using PostCallback).

this is actually what i am using. my timer function just postcallback's the refresher function, just to be sure that the main thread executes it in plain workflow, without sideeffects.

thus i coudl drop cpu usage about 80 %. thats not bad , just like in your example.. impressive

---