Subject: Graphics Context and Draw Object
Posted by arixion on Wed, 18 Oct 2006 12:52:42 GMT
View Forum Message <> Reply to Message

Hey, I am trying to create a version of the Scintilla control for UPP <http://www.scintilla.org/>. I have mostly finished my concept design, but have one major problem: Scintilla requires the use of a Surface Class, which has access to the Graphics Context. How do I get the Graphics Device Contexts (for both Windows and Linux) of Components in UPP? Or alternatively, how can I get the Draw objects associated with the painting of the components?

Subject: Re: Graphics Context and Draw Object
Posted by unodgs on Wed, 18 Oct 2006 13:33:45 GMT
View Forum Message <> Reply to Message

You can get it from using Draw::GetHandle(). It returns HDC handle. I don't know about Linux.

PS: Why do you need it? HWND should be all you need I think.

Subject: Re: Graphics Context and Draw Object
Posted by arixion on Wed, 18 Oct 2006 13:50:14 GMT
View Forum Message <> Reply to Message

Scintilla comes with its own class hierarchy. The most important class that Scintilla uses is the Editor Class. To do its painting, Editor calls on the AutoSurfaace Class, which creates a surface for painting on. This Surface needs access to GCs in order to do its painting. Actually, I need to be able to create a Draw Object that is linked to the Control. Could anyone help me??

Subject: Re: Graphics Context and Draw Object
Posted by mirek on Fri, 20 Oct 2006 13:47:20 GMT
View Forum Message <> Reply to Message

Actually, in X11 there are two - X11 surface and XftDraw pointer...

Both can be accessed from Draw.

Mirek

Subject: Re: Graphics Context and Draw Object
Posted by arixion on Sat, 21 Oct 2006 04:11:05 GMT
View Forum Message <> Reply to Message

Hmm, Is there a way to create a Draw Object related to a control? That is really what I'm looking

for. Is ViewDraw sufficient for the task? What is the paintqueue for ViewDraw? If I use ViewDraw to draw something on the screen, will that be erased on the next painting in the event loop?

Subject: Re: Graphics Context and Draw Object
Posted by mirek on Sat, 21 Oct 2006 04:24:48 GMT
View Forum Message <> Reply to Message

arixion wrote on Sat, 21 October 2006 00:11Hmm, Is there a way to create a Draw Object related to a control?

Well, the standard way how to paint view content is to use Paint method. You can request refreshing the view area (or its portion) by calling Refresh. You can also force immediate repainting by Sync.

ViewDraw is helper, rarely used thing, but most likely it would do what you want. But yes, next paint would repaint the view again, OTOH perhaps you can leave it empty....

Anyway, I have a bad feeling about this This is not how U++ code is supposed to work. OTOH, pushing corner limits might help to improve the U++.

Mirek

Subject: Re: Graphics Context and Draw Object
Posted by arixion on Sun, 22 Oct 2006 01:33:24 GMT
View Forum Message <> Reply to Message

If I use a Drawing Object, is there a way to erase the Drawing?

Subject: Re: Graphics Context and Draw Object
Posted by mirek on Sun, 22 Oct 2006 02:11:01 GMT
View Forum Message <> Reply to Message

First of all, I am not sure whether this question is related to others in this thread. If it is, then the main problem with Drawing is that it does not have any X11 handles in it (because DrawingDraw just stores all operation for later replay).

Anyway, if what you ask is how to have "background" with Drawing, just clear it with DrawRect...

Subject: Re: Graphics Context and Draw Object
Posted by arixion on Mon, 23 Oct 2006 03:03:40 GMT

Sigh

My main problem is that I'm trying to use the Scintilla Library Classes alongside UPP classes.

Here's how a port (any port) of Scintilla is supposed to work:-

1) The port code sets up the Window on the display device.

2) The code links a customized Scintilla class to the window wrapper.

3) This customized class inherits from ScintillaBase in the Scintilla Core Library, which provides lexing and folding support. ScintillaBase in turn derives itself from an Editor Class, which provides find/ replace, basic text-editting and autocomplete. The Editor Class calls on the helper classes Window, Surface and Menu to help it do its work:-

- Menu generates the auto-complete context menu from another helper class ListBox.

- Window provides access to windowing functions like resizing, moving, minimizing/maximizing and closing. It is meant, I think, as a link between the port code and the Scintilla Library.

- Surface provides functions for painting onto specified areas in the screen.

4) Editor calls on an AutoSurface Class whenever it needs to do painting. The AutoSurface class basically generates a surface upon which painting methods are called on.

My main problem here is how to create an implementation class for Surface that can draw onto the component area with persistence, because it is useless to paint things like context menus or cursors or whatever else only for a fraction of a second before it is erased again by TimerAndPaint in the Ctrl class. So, does anyone have any suggestions on how to achieve this?

the confused programmer

Subject: Re: Graphics Context and Draw Object
Posted by mirek on Mon, 23 Oct 2006 14:53:16 GMT

arixion wrote on Sun, 22 October 2006 23:03*Sigh*

My main problem is that I'm trying to use the Scintilla Library Classes alongside UPP classes.

Here's how a port (any port) of Scintilla is supposed to work:-

1) The port code sets up the Window on the display device.

2) The code links a customized Scintilla class to the window wrapper.

3) This customized class inherits from ScintillaBase in the Scintilla Core Library, which provides lexing and folding support. ScintillaBase in turn derives itself from an Editor Class, which provides find/ replace, basic text-editting and autocomplete. The Editor Class calls on the helper classes Window, Surface and Menu to help it do its work:-

- Menu generates the auto-complete context menu from another helper class ListBox.

- Window provides access to windowing functions like resizing, moving, minimizing/maximizing and closing. It is meant, I think, as a link between the port code and the Scintilla Library.

- Surface provides functions for painting onto specified areas in the screen.

4) Editor calls on an AutoSurface Class whenever it needs to do painting. The AutoSurface class basically generates a surface upon which painting methods are called on.

My main problem here is how to create an implementation class for Surface that can draw onto the component area with persistence, because it is useless to paint things like context menus or cursors or whatever else only for a fraction of a second before it is erased again by TimerAndPaint in the Ctrl class. So, does anyone have any suggestions on how to achieve this?

the confused programmer

Well, even scintilla must support repaint on host platform demand (WM_PAINT, expose envent), which is exactly what "TimerAndPaint" does...

In other words, I am quite convinced that as addition to (4) there must some interface which host platform can call to redraw portion of view area. Use that in Paint and you are OK.

Mirek

Subject: Re: Graphics Context and Draw Object
Posted by arixion on Mon, 23 Oct 2006 15:05:59 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 23 October 2006 22:53arixion wrote on Sun, 22 October 2006 23:03*Sigh*

My main problem is that I'm trying to use the Scintilla Library Classes alongside UPP classes.

Here's how a port (any port) of Scintilla is supposed to work:-

1) The port code sets up the Window on the display device.

2) The code links a customized Scintilla class to the window wrapper.

3) This customized class inherits from ScintillaBase in the Scintilla Core Library, which provides lexing and folding support. ScintillaBase in turn derives itself from an Editor Class, which provides

find/ replace, basic text-editting and autocomplete. The Editor Class calls on the helper classes Window, Surface and Menu to help it do its work:-

- Menu generates the auto-complete context menu from another helper class ListBox.

- Window provides access to windowing functions like resizing, moving, minimizing/maximizing and closing. It is meant, I think, as a link between the port code and the Scintilla Library.

- Surface provides functions for painting onto specified areas in the screen.

4) Editor calls on an AutoSurface Class whenever it needs to do painting. The AutoSurface class basically generates a surface upon which painting methods are called on.

My main problem here is how to create an implementation class for Surface that can draw onto the component area with persistence, because it is useless to paint things like context menus or cursors or whatever else only for a fraction of a second before it is erased again by TimerAndPaint in the Ctrl class. So, does anyone have any suggestions on how to achieve this?

the confused programmer

Well, even scintilla must support repaint on host platform demand (WM_PAINT, expose envent), which is exactly what "TimerAndPaint" does...

In other words, I am quite convinced that as addition to (4) there must some interface which host platform can call to redraw portion of view area. Use that in Paint and you are OK.

Mirek

I shall look at the Scintilla Library code again. But if Scintilla has such a method, which it does seem to have, how would u synchronize it with TimerAndPaint? Or are you suggesting to use that method to send a *direct* msg to the display system? Scintilla does have a sendWndMsg function...

Subject: Re: Graphics Context and Draw Object
Posted by mirek on Mon, 23 Oct 2006 15:25:55 GMT
View Forum Message <> Reply to Message

Quote:
I shall look at the Scintilla Library code again. But if Scintilla has such a method, which it does seem to have, how would u synchronize it with TimerAndPaint? Or are you suggesting to use that method to send a *direct* msg to the display system? Scintilla does have a sendWndMsg function...

Paint is called when either system or U++ widget requires so (second thing does not happen unless you call Refresh somewhere in the widget code). So in fact, you can think about Paint as "system requested repaint". Such thing can happen at any moment in both X11 and Win32, so

scintilla MUST be able to handle such request.

So what you are supposed to do is to call scintilla's form of view area repaint in Paint method - then nothing can go wrong (even using the ViewDraw).

Mirek

Subject: Re: Graphics Context and Draw Object
Posted by arixion on Tue, 24 Oct 2006 13:23:14 GMT
View Forum Message <> Reply to Message

Which Refresh Function links to Paint? I don't see any.

Subject: Re: Graphics Context and Draw Object
Posted by unodgs on Tue, 24 Oct 2006 13:31:37 GMT
View Forum Message <> Reply to Message

arixion wrote on Tue, 24 October 2006 09:23Which Refresh Function links to Paint? I don't see any.

Ctrl::Refresh(..)

Subject: Re: Graphics Context and Draw Object
Posted by mirek on Tue, 24 Oct 2006 14:09:35 GMT
View Forum Message <> Reply to Message

arixion wrote on Tue, 24 October 2006 09:23Which Refresh Function links to Paint? I don't see any.

Well, what actually happens is this: When system requires repainting of area, this request is stored for further processing. When widget requires repainting of area, this request is stored for further processing. Now when input event queue is empty and all timers are processed comes the time to actually paint something. All painting requests are combined in any way system or U++ thinks is most effective. This is done to improve performance - if the Refresh is called several times during input event processing, only one repaint is actually done.
