Subject: Re: pixmap problem
Posted by Karl Schultz on Thu, 28 Aug 2003 20:35:16 GMT
View Forum Message <> Reply to Message

"Steve Ready" <ready@parc.com> wrote in message
news:bilina$4ee$1@news.parc.xerox.com...
> I have replicated this with a Radion VE 32mb, Invidia GForce 64mb, and a
> Matrox 32mb under Win2k and XP. The maximum image size returned by the
> correct behavior for all these boards seems to scale loosely with the
color
> resolution (16 versus 32 bit) coupled with the amount of card memory. It
> would be nice to have a better understanding of this and have IDL
correctly
> report the limitiations.
>
> I have reported this to RSI and here was their response:
>
> **
> We have no reason to believe that you are running into any issue other
than
> the limits imposed by the Windows O.S. in its interface with your graphics
> card. One way to see this would be to try to make a bigger PIXMAP with the
> Windows Visual C++/MFC API. In both cases, you would want to make sure
that
> both IDL and the MFC comparative program would have a "virgin" O.S.
> environment; that is, there would not be other applications running about
in
> the background making demands on the video card memory. Consider that
video
> card memory could get fragmented just like process memory.
>
> IDL's WINDOWS, /PIXMAP call is making a basic call on the Win32 API pixmap
> library. I do not know the underlying source code for Microsoft's library,
> but it is very possible that that memory automatically makes a 3x or 4x
> pixmap-coordinates demand on the video card. Perhaps it does this only
when
> your display is set to true color; you could test and see if setting your
> display to 256-colors provided you the bigger pixmap you want.
>
> We would expect the larger memory video card to allow for a much larger
> maximum PIXMAP size, but, to be certain, you may need to consult with the
> video card vendor about what THEY think the maximum pixmap is that Windows
> can make on their card.
>
> ***
>
>
>

> "Karl Schultz" <kschultz_no_spam@rsinc.com> wrote in message
> news:vksds6nnjrv1a@corp.supernews.com...
>>
>> This is a known problem, # 22066 should you call IDL tech support about
> it.
>>
>> There is no clear resolution to the problem, but it does appear to be
> pretty
>> sensitive to the graphics card and/or driver software. On many
machines,
>> the attempt to create a "too large" pixmap will fail and any pixmap that
> is
>> created successfully returns the correct data on a TVRD; this is the
> correct
>> behavior. However, several users have reported the same symptoms as you
> are
>> describing. If you can report what graphics hardware and driver
software
>> (including version information) you are using, perhaps we can identifiy
a
>> pattern. (And I suppose the canned response of "check your drivers for
an
>> update" is a good thing to offer here as well.)
>>
>> Another strange behavior on the failing hardware is that the ORDER
keyword
>> can affect the pixmap size threshhold where the data is not read back
>> correctly via TVRD. IDL itself does not perform the "flip" as
controlled
> by
>> ORDER. IDL instead modifies parameters to a WIN32 GDI call, thus making
> the
>> GDI or graphics driver perform the flip. A change in behavior like this
> is
>> somewhat suggestive of a driver problem.
>>
>> Karl
>>
>>
>
>

OK, I was finally able to reproduce it here - I was not able to before with
the machines I had available to me.

Windows is reporting an out-of-memory condition when IDL asks it to copy the
image from the pixmap (video memory) to an IDL array (system memory). One
would think that this copy would usually succeed since the memory has

already been allocated in both places. The pixmap has been successfully
created and drawn to in video memory and the target array in system memory
already exists.

The problem is that there is an intermediate copy step. I think that
Windows must allocate some special GDI system pool memory that is locked or
'pinned' so it can't be swapped, presumably for a DMA, AGP, or other special
memory transfer from the video memory to the pinned system memory. If GDI
resources are scarce or fragmented, this allocation may fail, causing the
copy to fail. I experimented quite a bit and noted some unusual patterns of
success and failure, which I attribute to varying degrees of fragmentation.

I don't think that IDL can predict a failure here, but we can throw an error
on failure rather than returning an array of zeros. Or, we can do the copy
in smaller blocks, which would be a whole lot better.

I took a quick look for some sort of control in Windows XP that would
increase the GDI resource memory pool, but didn't find one. I'll post
something again if I find it later. But if you should find it, you might
try increasing it as a short term workaround. I remember seeing this sort
of thing in older versions of Windows.

Karl