Subject: Re: Incomplete ouput PNG files.
Posted by David Fanning on Wed, 14 Dec 2011 16:02:16 GMT
View Forum Message <> Reply to Message

alx writes:

> I recently wanted to process a large number of data files in order to
> produce the corresponding number of output image files, each in PNG
> format. I choose to use NG graphics, since the sequence: "p =
> image(some_data) & p.save('this_image.png') & p.close", appears pretty
> well suitable for this purpose.

Ah, you didn't choose wisely, my son. ;-)

You might want to give Coyote Graphics a go:

```
cgImage, image, Output='this_image.png'
```

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Incomplete ouput PNG files.
Posted by Mark Piper on Wed, 14 Dec 2011 16:22:34 GMT
View Forum Message <> Reply to Message

On 12/14/2011 8:33 AM, alx wrote:
> I recently wanted to process a large number of data files in order to
> produce the corresponding number of output image files, each in PNG
> format. I choose to use NG graphics, since the sequence: "p =
> image(some_data)& p.save('this_image.png')& p.close", appears pretty
> well suitable for this purpose. No problem in coding the corresponding
> IDL 8.1 project on my (Windows) workstation, and running it on a few
> test files. Since the data files were stored on a remote (Windows)
> server, I decided to run the project directly on the server, through a
> remote connection from my workstation to the server, by using Terminal
> Server. Still no problem: the job was running perfectly, and I could
> remotely see each graphic window being open, then the corresponding
> PNG file created, etc...

> Now, to avoid any flashing on my screen, I just MINIMIZED the TS
> window which was open on my workstation (no logout, no disconnect from
> the remote server), the remote job continuing to run unattended.
> Patatras ! From this point the PNG files were created but also harshly
> truncated (basically reduced to a header and a void image).
> I guess (but have not tested) that any other image file format would
> have produced the same unsatisfying result.
> A "true screen" seems then mandatory for having the "graphic.save"
> function to work correctly: is it a bug of NG graphics or a feature?
> Have some unix/linux people got same experience when using any X-
> server in this way ?
> alx.

This is a slightly different workflow, but could you please try setting
the BUFFER keyword in your call to IMAGE? E.g.,

p = image(data, /buffer)
p.save, 'this_image.png'
p.close

The graphic will be rendered in an offscreen buffer. I have a hunch that
this may help, since this feels like a tricky (to me, at least) X server
issue.

mp

Subject: Re: Incomplete ouput PNG files.
Posted by lecacheux.alain on Wed, 14 Dec 2011 16:43:22 GMT
View Forum Message <> Reply to Message

On 14 déc, 17:22, Mark Piper <mpi...@ittvis.com> wrote:
> On 12/14/2011 8:33 AM, alx wrote:
>
>
>
>
>
>> I recently wanted to process a large number of data files in order to
>> produce the corresponding number of output image files, each in PNG
>> format. I choose to use NG graphics, since the sequence: "p =
>> image(some_data)& p.save('this_image.png')& p.close", appears pretty
>> well suitable for this purpose. No problem in coding the corresponding
>> IDL 8.1 project on my (Windows) workstation, and running it on a few
>> test files. Since the data files were stored on a remote (Windows)
>> server, I decided to run the project directly on the server, through a
>> remote connection from my workstation to the server, by using Terminal
>> Server. Still no problem: the job was running perfectly, and I could

>> remotely see each graphic window being open, then the corresponding
>> PNG file created, etc...
>> Now, to avoid any flashing on my screen, I just MINIMIZED the TS
>> window which was open on my workstation (no logout, no disconnect from
>> the remote server), the remote job continuing to run unattended.
>> Patatras ! From this point the PNG files were created but also harshly
>> truncated (basically reduced to a header and a void image).
>> I guess (but have not tested) that any other image file format would
>> have produced the same unsatisfying result.
>> A "true screen" seems then mandatory for having the "graphic.save"
>> function to work correctly: is it a bug of NG graphics or a feature?
>> Have some unix/linux people got same experience when using any X-
>> server in this way ?
>> alx.
>
> This is a slightly different workflow, but could you please try setting
> the BUFFER keyword in your call to IMAGE? E.g.,
>
> p = image(data, /buffer)
> p.save, 'this_image.png'
> p.close
>
> The graphic will be rendered in an offscreen buffer. I have a hunch that
> this may help, since this feels like a tricky (to me, at least) X server
> issue.
>
> mp- Masquer le texte des messages précédents -
>
> - Afficher le texte des messages précédents -

Setting the BUFFER keyword actually solved the problem. I now
understand what this keyword is useful to, a long lasting
interrogation for me! Thanks Mark.
alain.

Subject: Re: Incomplete ouput PNG files.
Posted by Mark Piper on Wed, 14 Dec 2011 17:26:28 GMT
View Forum Message <> Reply to Message

On 12/14/2011 9:43 AM, alx wrote:
> Setting the BUFFER keyword actually solved the problem. I now
> understand what this keyword is useful to, a long lasting
> interrogation for me! Thanks Mark.
> alain.

Cool -- you're welcome! Another idea: since there's overhead in
creating/destroying NG windows, it may help to use the SetData method

for batching a series of plots:

```
data = list(of_images)
p = image(data[0], /buffer)
p.save, 'image0.png'

for i=1, data.count()-1 do begin
  p.setdata, data[i]
  p.save, 'image' + strtrim(i,2) + '.png'
endfor

p.close
```

mp

Subject: Re: Incomplete ouput PNG files.
Posted by David Fanning on Wed, 14 Dec 2011 22:21:40 GMT
View Forum Message <> Reply to Message

Mark Piper writes:

> This is a slightly different workflow, but could you please try setting
> the BUFFER keyword in your call to IMAGE? E.g.,
>
> p = image(data, /buffer)
> p.save, 'this_image.png'
> p.close
>
> The graphic will be rendered in an offscreen buffer. I have a hunch that
> this may help, since this feels like a tricky (to me, at least) X server
> issue.

I was curious to see how Coyote Graphics output would
stack up against the output from these function graphics
routines. But I wanted to be able to compare apples
to apples, so I spent some time today modifying the
Coyote Graphic routines so that I could control
the output file parameters, and in particular, the
resolution of the output.

This is now done with cgWindow_SetDefs, just like
it is for cgWindow. In my first comparisons, I noticed
that the function graphics output was a bit darker
than the Coyote Graphics output, so I defined a new
keyword for PS_START, called DEFAULT_THICKNESS so that
I can set the default line and character thickness for
the PostScript output. I set the default to 3 to better

match the function graphics output.

Anyway, you will need an updated Coyote Library to run
the program described, if you want to play around with this:

http://www.idlcoyote.com/programs/zip_files/coyoteprograms.z ip

This is tagged release 1.5.1, if you are using the Subversion
repository.

So, here is the program. I'm doing a simple plot command and
saving the data as JPEG, PNG, and encapsulated PostScript files.
(Coyote Graphics routines actually produce landscape PostScript
files, which function graphics commands do not, so I am using
encapsulated PostScript for my comparisons. Both will produce
encapsulated output in Portrait mode.) I've saved the files
at 600 dpi, 300 dpi and 75 dpi.

I was careful to make sure I was using the same size window
in both cases, 640 in X and 512 in Y.

In general, I can't really tell much difference in the output.
The title is set too close to the plot, but that has always
been the case in direct graphics. That is about the only
difference that really jumps out at me.

A couple of odd things. The PostScript files are all the
same size at every resolution. They are 11KB for Coyote
Graphics output and 9 KB for function graphics output.
Here is a table of values in KM. The size values are
a comparison of the output. You can see that Coyote
Graphics routines are consistently larger in dimensions,
but smaller in total size. I don't know how to account for
this. In any case, the visual output is comparable so
I assume this is just a different way of setting the
resolution. The XSIZE and YSIZE dimensions are for the
JPEG file in every case, but the comparable PNG file
has the same dimensions.

	EPS	JPEG	PNG	XSIZE	YSIZE
cg75	11	39	63	717	573
fg75	9	39	33	667	534
cg300	11	227	46	2867	2292
fg300	9	254	165	2669	2135

cg600 11   568   131   5733   4583
fg600 9   736   379   5339   4271

I guess the bottom line is that I am EXTREMELY happy
with the performance of Coyote Graphics in this
comparison. Not only are my routines faster, but the output
I care about is essentially identical to the output
from function graphics routines. As an added bonus,
my output files are significantly smaller at high
resolution. I don't know why this would be the case.

Here is the code I used, if you want to try this for
yourself:

  http://www.idlcoyote.com/misc/compare_resolution.pro

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Incomplete ouput PNG files.
Posted by lecacheux.alain on Thu, 15 Dec 2011 10:16:46 GMT
View Forum Message <> Reply to Message

On 14 déc, 18:26, Mark Piper <mpi...@ittvis.com> wrote:
> On 12/14/2011 9:43 AM, alx wrote:
>
>> Setting the BUFFER keyword actually solved the problem. I now
>> understand what this keyword is useful to, a long lasting
>> interrogation for me! Thanks Mark.
>> alain.
>
> Cool -- you're welcome! Another idea: since there's overhead in
> creating/destroying NG windows, it may help to use the SetData method
> for batching a series of plots:
>
>   data = list(of_images)
>   p = image(data[0], /buffer)
>   p.save, 'image0.png'
>
>   for i=1, data.count()-1 do begin
>   p.setdata, data[i]

> p.save, 'image' + strtrim(i,2) + '.png'
> endfor
>
> p.close
>
> mp

Using setdata/getdata would be efficient. But most of the overhead, in my case, is due to the computing/writing on disk of the PNG file (i.e. "p.save").
alx.

Subject: Re: Incomplete ouput PNG files.
Posted by lecacheux.alain on Thu, 15 Dec 2011 10:22:31 GMT
View Forum Message <> Reply to Message

On 14 déc, 23:21, David Fanning <n...@dfanning.com> wrote:
> Mark Piper writes:
>> This is a slightly different workflow, but could you please try setting
>> the BUFFER keyword in your call to IMAGE? E.g.,
>
>> p = image(data, /buffer)
>> p.save, 'this_image.png'
>> p.close
>
>> The graphic will be rendered in an offscreen buffer. I have a hunch that
>> this may help, since this feels like a tricky (to me, at least) X server
>> issue.
>
> I was curious to see how Coyote Graphics output would
> stack up against the output from these function graphics
> routines. But I wanted to be able to compare apples
> to apples, so I spent some time today modifying the
> Coyote Graphic routines so that I could control
> the output file parameters, and in particular, the
> resolution of the output.
>
> This is now done with cgWindow_SetDefs, just like
> it is for cgWindow. In my first comparisons, I noticed
> that the function graphics output was a bit darker
> than the Coyote Graphics output, so I defined a new
> keyword for PS_START, called DEFAULT_THICKNESS so that
> I can set the default line and character thickness for
> the PostScript output. I set the default to 3 to better
> match the function graphics output.
>
> Anyway, you will need an updated Coyote Library to run

> the program described, if you want to play around with this:
>
> http://www.idlcoyote.com/programs/zip_files/coyoteprograms.z ip
>
> This is tagged release 1.5.1, if you are using the Subversion
> repository.
>
> So, here is the program. I'm doing a simple plot command and
> saving the data as JPEG, PNG, and encapsulated PostScript files.
> (Coyote Graphics routines actually produce landscape PostScript
> files, which function graphics commands do not, so I am using
> encapsulated PostScript for my comparisons. Both will produce
> encapsulated output in Portrait mode.) I've saved the files
> at 600 dpi, 300 dpi and 75 dpi.
>
> I was careful to make sure I was using the same size window
> in both cases, 640 in X and 512 in Y.
>
> In general, I can't really tell much difference in the output.
> The title is set too close to the plot, but that has always
> been the case in direct graphics. That is about the only
> difference that really jumps out at me.
>
> A couple of odd things. The PostScript files are all the
> same size at every resolution. They are 11KB for Coyote
> Graphics output and 9 KB for function graphics output.
> Here is a table of values in KM. The size values are
> a comparison of the output. You can see that Coyote
> Graphics routines are consistently larger in dimensions,
> but smaller in total size. I don't know how to account for
> this. In any case, the visual output is comparable so
> I assume this is just a different way of setting the
> resolution. The XSIZE and YSIZE dimensions are for the
> JPEG file in every case, but the comparable PNG file
> has the same dimensions.
>
> EPS JPEG PNG XSIZE YSIZE
> cg75 11 39 63 717 573
> fg75 9 39 33 667 534
>
> cg300 11 227 46 2867 2292
> fg300 9 254 165 2669 2135
>
> cg600 11 568 131 5733 4583
> fg600 9 736 379 5339 4271
>
> I guess the bottom line is that I am EXTREMELY happy
> with the performance of Coyote Graphics in this

> comparison. Not only are my routines faster, but the output
> I care about is essentially identical to the output
> from function graphics routines. As an added bonus,
> my output files are significantly smaller at high
> resolution. I don't know why this would be the case.
>
> Here is the code I used, if you want to try this for
> yourself:
>
> http://www.idlcoyote.com/misc/compare_resolution.pro
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming:http://www.idlcoyote.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

I could note that the "p.save"d PNG file size is depending on the window size when using an open NG graphics window. I guess that the saved graphic file will depend on the off-screen buffer size when BUFFER keyword is used. But what is this size? I could not find the answer in 8.1 documentation. Maybe larger that Coyote's one (IDLgrBuffer has a maximum size of 82192x8192) ?
alx.

Subject: Re: Incomplete ouput PNG files.
Posted by lecacheux.alain on Thu, 15 Dec 2011 10:32:47 GMT
View Forum Message <> Reply to Message

On 15 déc, 11:22, alx <lecacheux.al...@wanadoo.fr> wrote:
> On 14 déc, 23:21, David Fanning <n...@dfanning.com> wrote:
>
>
>
>
>
>> Mark Piper writes:
>>> This is a slightly different workflow, but could you please try setting
>>> the BUFFER keyword in your call to IMAGE? E.g.,
>
>>> p = image(data, /buffer)
>>> p.save, 'this_image.png'
>>> p.close
>

>>> The graphic will be rendered in an offscreen buffer. I have a hunch that
>>> this may help, since this feels like a tricky (to me, at least) X server
>>> issue.
>
>> I was curious to see how Coyote Graphics output would
>> stack up against the output from these function graphics
>> routines. But I wanted to be able to compare apples
>> to apples, so I spent some time today modifying the
>> Coyote Graphic routines so that I could control
>> the output file parameters, and in particular, the
>> resolution of the output.
>
>> This is now done with cgWindow_SetDefs, just like
>> it is for cgWindow. In my first comparisons, I noticed
>> that the function graphics output was a bit darker
>> than the Coyote Graphics output, so I defined a new
>> keyword for PS_START, called DEFAULT_THICKNESS so that
>> I can set the default line and character thickness for
>> the PostScript output. I set the default to 3 to better
>> match the function graphics output.
>
>> Anyway, you will need an updated Coyote Library to run
>> the program described, if you want to play around with this:
>
>> http://www.idlcoyote.com/programs/zip_files/coyoteprograms.z ip
>
>> This is tagged release 1.5.1, if you are using the Subversion
>> repository.
>
>> So, here is the program. I'm doing a simple plot command and
>> saving the data as JPEG, PNG, and encapsulated PostScript files.
>> (Coyote Graphics routines actually produce landscape PostScript
>> files, which function graphics commands do not, so I am using
>> encapsulated PostScript for my comparisons. Both will produce
>> encapsulated output in Portrait mode.) I've saved the files
>> at 600 dpi, 300 dpi and 75 dpi.
>
>> I was careful to make sure I was using the same size window
>> in both cases, 640 in X and 512 in Y.
>
>> In general, I can't really tell much difference in the output.
>> The title is set too close to the plot, but that has always
>> been the case in direct graphics. That is about the only
>> difference that really jumps out at me.
>
>> A couple of odd things. The PostScript files are all the
>> same size at every resolution. They are 11KB for Coyote
>> Graphics output and 9 KB for function graphics output.

>> Here is a table of values in KM. The size values are
>> a comparison of the output. You can see that Coyote
>> Graphics routines are consistently larger in dimensions,
>> but smaller in total size. I don't know how to account for
>> this. In any case, the visual output is comparable so
>> I assume this is just a different way of setting the
>> resolution. The XSIZE and YSIZE dimensions are for the
>> JPEG file in every case, but the comparable PNG file
>> has the same dimensions.
>
>> EPS JPEG PNG XSIZE YSIZE
>> cg75 11 39 63 717 573
>> fg75 9 39 33 667 534
>
>> cg300 11 227 46 2867 2292
>> fg300 9 254 165 2669 2135
>
>> cg600 11 568 131 5733 4583
>> fg600 9 736 379 5339 4271
>
>> I guess the bottom line is that I am EXTREMELY happy
>> with the performance of Coyote Graphics in this
>> comparison. Not only are my routines faster, but the output
>> I care about is essentially identical to the output
>> from function graphics routines. As an added bonus,
>> my output files are significantly smaller at high
>> resolution. I don't know why this would be the case.
>
>> Here is the code I used, if you want to try this for
>> yourself:
>
>> http://www.idlcoyote.com/misc/compare_resolution.pro
>
>> Cheers,
>
>> David
>> --
>> David Fanning, Ph.D.
>> Fanning Software Consulting, Inc.
>> Coyote's Guide to IDL Programming:http://www.idlcoyote.com/
>> Sepore ma de ni thui. ("Perhaps thou speakest truth.")
>
> I could note that the "p.save"d PNG file size is depending on the
> window size when using an open NG graphics window. I guess that the
> saved graphic file will depend on the off-screen buffer size when
> BUFFER keyword is used. But what is this size? I could not find the
> answer in 8.1 documentation. Maybe larger that Coyote's one
> (IDLgrBuffer has a maximum size of 82192x8192) ?

> alx.- Masquer le texte des messages précédents -
>
> - Afficher le texte des messages précédents -

sorry, please read 8192x8192 in my previous message.

Subject: Re: Incomplete ouput PNG files.
Posted by David Fanning on Thu, 15 Dec 2011 13:12:30 GMT
View Forum Message <> Reply to Message

alx writes:

> I could note that the "p.save"d PNG file size is depending on the
> window size when using an open NG graphics window. I guess that the
> saved graphic file will depend on the off-screen buffer size when
> BUFFER keyword is used. But what is this size? I could not find the
> answer in 8.1 documentation. Maybe larger that Coyote's one
> (IDLgrBuffer has a maximum size of 82192x8192) ?

I don' think an open window makes any difference
in Function Graphics. In my tests, I got the same
huge raster file output whether I created a normal
sized window, or used the buffer. The only way to
control this (confirmed by a re-read of one of
Mark's articles on his blog) is to use the
RESOLUTION keyword.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Incomplete ouput PNG files.
Posted by lecacheux.alain on Thu, 15 Dec 2011 14:07:52 GMT
View Forum Message <> Reply to Message

On 15 déc, 14:12, David Fanning <n...@dfanning.com> wrote:
> alx writes:
>> I could note that the "p.save"d PNG file size is depending on the

>> window size when using an open NG graphics window. I guess that the
>> saved graphic file will depend on the off-screen buffer size when
>> BUFFER keyword is used. But what is this size? I could not find the
>> answer in 8.1 documentation. Maybe larger that Coyote's one
>> (IDLgrBuffer has a maximum size of 82192x8192) ?
>
> I don' think an open window makes any difference
> in Function Graphics. In my tests, I got the same
> huge raster file output whether I created a normal
> sized window, or used the buffer. The only way to
> control this (confirmed by a re-read of one of
> Mark's articles on his blog) is to use the
> RESOLUTION keyword.
>
> Cheers,
>
> David
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming:http://www.idlcoyote.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

I agree. RESOLUTION keyword is a solution, but the (true or implicit)
window size also plays some role. For an original image of size
2224x1124 and PNG output(from p=image(DIMENSIONS=...) &
p.save,BUFFER=...,RESOLUTION=...), I find:
  BUFFER=1, RESOLUTION=96, DIMENSIONS=[800,600] -> FILE SIZE = 99 kB
  BUFFER=1, RESOLUTION=300, DIMENSIONS=[800,600] -> FILE_SIZE = 290
kB
  BUFFER=1, RESOLUTION=600, DIMENSIONS=[800,600] -> FILE SIZE = 488
kB
  BUFFER=1, RESOLUTION=600, DIMENSIONS=[640,512] -> FILE SIZE = 423
kB
  BUFFER=1, RESOLUTION=600, DIMENSIONS=[1024,768] -> FILE_SIZE = 602
kB
  BUFFER=0, RESOLUTION=600, DIMENSIONS=[640,512] -> FILE SIZE = 423
kB
  BUFFER=0, RESOLUTION=600, DIMENSIONS=[800,600] -> FILE SIZE = 408
kB
  BUFFER=0, RESOLUTION=600, DIMENSIONS=[1024,768] -> FILE_SIZE = 602
kB
Note the (strange ?) difference between BUFFER=0 and 1 for
DIMENSIONS=[800,600]
alx.

Subject: Re: Incomplete ouput PNG files.
Posted by David Fanning on Thu, 15 Dec 2011 14:23:18 GMT
View Forum Message <> Reply to Message

alx writes:

> I agree. RESOLUTION keyword is a solution, but the (true or implicit)
> window size also plays some role. For an original image of size
> 2224x1124 and PNG output(from p=image(DIMENSIONS=...) &
> p.save,BUFFER=...,RESOLUTION=...), I find:
> BUFFER=1, RESOLUTION=96, DIMENSIONS=[800,600] -> FILE SIZE = 99 kB
> BUFFER=1, RESOLUTION=300, DIMENSIONS=[800,600] -> FILE_SIZE = 290
> kB
> BUFFER=1, RESOLUTION=600, DIMENSIONS=[800,600] -> FILE SIZE = 488
> kB
> BUFFER=1, RESOLUTION=600, DIMENSIONS=[640,512] -> FILE SIZE = 423
> kB
> BUFFER=1, RESOLUTION=600, DIMENSIONS=[1024,768] -> FILE_SIZE = 602
> kB
> BUFFER=0, RESOLUTION=600, DIMENSIONS=[640,512] -> FILE SIZE = 423
> kB
> BUFFER=0, RESOLUTION=600, DIMENSIONS=[800,600] -> FILE SIZE = 408
> kB
> BUFFER=0, RESOLUTION=600, DIMENSIONS=[1024,768] -> FILE_SIZE = 602
> kB
> Note the (strange ?) difference between BUFFER=0 and 1 for
> DIMENSIONS=[800,600]

Yes, I'm confused about just what "resolution" means in this context. Even more confusing is why Coyote Graphics files seem to be smaller in overall file size, while consistently larger in image dimensions. I'm guessing "resolution" must be one of those things that is in the eye of the beholder. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Incomplete ouput PNG files.
Posted by penteado on Thu, 15 Dec 2011 17:19:23 GMT
View Forum Message <> Reply to Message

On Dec 15, 12:23 pm, David Fanning <n...@dfanning.com> wrote:
> alx writes:
>> I agree. RESOLUTION keyword is a solution, but the (true or implicit)
>> window size also plays some role. For an original image of size
>> 2224x1124 and PNG output(from p=image(DIMENSIONS=...) &
>> p.save,BUFFER=...,RESOLUTION=...), I find:
>> BUFFER=1, RESOLUTION=96, DIMENSIONS=[800,600] -> FILE SIZE = 99 kB
>> BUFFER=1, RESOLUTION=300, DIMENSIONS=[800,600] -> FILE_SIZE = 290
>> kB
>> BUFFER=1, RESOLUTION=600, DIMENSIONS=[800,600] -> FILE SIZE = 488
>> kB
>> BUFFER=1, RESOLUTION=600, DIMENSIONS=[640,512] -> FILE SIZE = 423
>> kB
>> BUFFER=1, RESOLUTION=600, DIMENSIONS=[1024,768] -> FILE_SIZE = 602
>> kB
>> BUFFER=0, RESOLUTION=600, DIMENSIONS=[640,512] -> FILE SIZE = 423
>> kB
>> BUFFER=0, RESOLUTION=600, DIMENSIONS=[800,600] -> FILE SIZE = 408
>> kB
>> BUFFER=0, RESOLUTION=600, DIMENSIONS=[1024,768] -> FILE_SIZE = 602
>> kB
>> Note the (strange ?) difference between BUFFER=0 and 1 for
>> DIMENSIONS=[800,600]
>
> Yes, I'm confused about just what "resolution" means in
> this context. Even more confusing is why Coyote Graphics
> files seem to be smaller in overall file size, while
> consistently larger in image dimensions. I'm guessing
> "resolution" must be one of those things that is in the
> eye of the beholder. :-)

Maybe such a comparison should be done in terms of pixel size, not
file size, since PNG has compression, thus making the file size not
proportional to the pixel size. Also, different filtering and
interlacing options may change the compression ratio.

Subject: Re: Incomplete ouput PNG files.
Posted by David Fanning on Thu, 15 Dec 2011 18:18:56 GMT
View Forum Message <> Reply to Message

Paulo Penteado writes:

> Maybe such a comparison should be done in terms of pixel size, not

> file size, since PNG has compression, thus making the file size not
> proportional to the pixel size. Also, different filtering and
> interlacing options may change the compression ratio.

Yes, I guess it's all very complicated. :-(

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")
