Emulator - Emulator Issues #267

Sonic Heroes graphical problems (EFB scaling issue)

11/01/2008 09:17 PM - superempra

Status:	Fixed	**% Done:**	0%
Priority:	Normal		
Assignee:			
Category:			
Target version:			
Operating system:	N/A	**Relates to performance:**	No
Issue type:	Bug	**Easy:**	No
Milestone:		**Relates to maintainability:**	No
Regression:	No	**Regression start:**	
Relates to usability:	No	**Fixed in:**	

Description

What steps will reproduce the problem?
1. Use OGL plugin
2. Dual Core enabled
3. Play Sonic heroes

What is the expected output? What do you see instead?
The ground is the wrong colour and whenever you move forward that graphical
glitch moves with you.

What version of the product are you using? On what operating system?
Dolphin 32 bit Rev 1029

Please provide any additional information below.
Intel Core 2 Duo 2.40GHz
ATI Radeon HD2600 XT 256MB of VRAM
1GB RAM

Related issues:

Has duplicate Emulator - Emulator Issues #3888: Sonic Heroes black areas ingame	**Duplicate**
Has duplicate Emulator - Emulator Issues #5346: sonic heroes ground glich	**Duplicate**

History

#1 - 11/03/2008 04:36 PM - Sonicadvance1

Try again with a rev above 1048 and see if this problem still persists.

#3 - 11/09/2008 07:03 PM - superempra

It still happens with R1097

#4 - 03/04/2009 10:47 PM - marcus

- Status changed from New to Fixed

Fixed now.

#5 - 01/21/2010 11:44 PM - knuckles500

Sorry for the bump, but even though it's fixed for the OpenGL plugin - it isn't for
the DX9 plugin. But other than this bug, it should be a fully playable game.

As of R4878, the bug is still present in the DX9 plugin:

http://i50.tinypic.com/2z6iae9.png

I don't know the revision number that fixed this problem with the OGL plugin, though. :(

#6 - 01/22/2010 12:31 AM - PeterAndre666

I only get that in DX9 if i check the widescreen hack, so i don't. :p

#7 - 01/22/2010 12:39 AM - knuckles500

What's your revision? Maybe I should update my copy of Dolphin, heh.

Also - 32bit or 64bit? Need some details. :\

#8 - 01/22/2010 06:24 PM - jayork42

I did see this recently, but I can't remember the plugin that was used. It has to do with EFB copy reading from junk data in RAM and mapping random pixels I believe. I also think that the reason why it stretches out everywhere might be because those pixels would otherwise be transprent if EFB copy were handling the character shadows correctly. But I'm not sure, that's just my guess.

#9 - 03/22/2010 01:58 PM - jayork42

This issue still remains in the latest revisions. I've come to find out that disabling EFB copy, enabling native, enabling EFB copy, and disabling native in that order fix the problem until I get to a certain part of the level where the issue happens all over again. Native fixes the problem.

#10 - 03/24/2010 07:40 PM - pascal.jouy

+1 to crashda..., expect that I have the same problem with OpenGL.
(Issue #2392)

#11 - 08/13/2010 01:31 AM - knuckles500

Hmm, I checked the Software Video Plugin, and the problem isn't there.

It's still there in every plugin besides the Software one.

#12 - 11/28/2010 09:11 AM - knuckles500

Just FYI, I finally figured out the fix for this issue, but with a catch.

You need to use the DirectX11 plugin.

The fix itself is very simple, and it's the only way to fix it (no other settings or video plugin besides the software one or maybe the OpenGL plugin will correct it).

What you have to do is set the EFB Scale to 1x. That's all you have to do.

#13 - 01/29/2011 06:29 PM - DimitriPilot3

- Status changed from Fixed to Accepted

I'm reopening this issue, as it still exists. I guess this might be an issue with EFB scaling which results into the main character's shadow's texture having non-transparent edges, causing that texture to be "extended" beyond its edges...

According to my experiments, I can say the following:
- It apparently occurs all the time in DX9 (meaning that no option can make the glitch disappear)
- In DX11 and OGL, it doesn't occur all the time - it mainly depends on the EFB scale option (example from DX11 which may vary in OGL):
- no problem using values lower or equal to 1 (0.375x, 0.5x, 0.75x, or 1x)
- when using values greater than 1 (2x or 3x), the problem appears (2 edges of the shadow texture are affected)
- when using Integral or Fractional scaling, the problem appears whenever the renderer window's width and/or height are greater than the native EFB resolution (640x574 - the affected edges depend on the width and the height)

Other notable things (example from DX11 which may vary in OGL):
- When the problem occurs (horizontally, "vertically" or both), some lines on the left/top borders of the screen look like they aren't drawn/refreshed correctly
(see http://i51.tinypic.com/2edm7at.jpg and http://i53.tinypic.com/2qtvez4.jpg and http://i55.tinypic.com/ny79dh.jpg)
- The problem disappears for about one frame when the renderer is resized, which effectively refreshes the buggy lines I mentioned above...

I suspect the issue in DX9 to be different (worse?) from the one in DX11/OGL... But that's only a guess :P

#14 - 01/29/2011 06:59 PM - DimitriPilot3

FWIW, this video showing the effects of randomly changing options was posted in issue 3888:
http://www.youtube.com/watch?v=w6BnEoAcbwI&hd=1

The SSAA option in the DX11 plugin appears to be greyed out using my NVIDIA GeForce 8800 GTS 512. I assume it's because that GPU supports DX10 but not DX11...

#15 - 01/29/2011 06:59 PM - DimitriPilot3

issue 3888 has been merged into this issue.

#16 - 02/12/2012 04:06 PM - jayork42

Excellent demo Dimitri!

#17 - 02/12/2012 04:28 PM - NeoBrainX

Is this still an issue?

If it is, could anyone upload a fifo player file for a frame where the problem is visible? (preferably at a place where it's VERY obvious)
To create one, follow these steps:
1. Start Dolphin
2. Select "Tools->Fifo Player" from the menu
3. Select the "Record" tab in the dialog that pops up.
4. Set the number of "frames to record" to 1.
5. Start the game (do NOT close the dialog) and get to the place where the problem occurs
6. When you've reached a place in the game where the problem is obvious, press the "Record" button in the Fifo Player dialog.

7. Wait a bit until the "Save" button isn't grayed out anymore.
8. Press the Save button and pick a location to store the fifo log.
9. Post the fifo log here; do NOT attach it to this bug report, upload it to an external host instead and just put a link here.

Please create multiple log files of the same scene:
1. With DX11, native internal resolution, EFB to texture (the problem should NOT be visible in this case)
2. With DX9 (problem should be visible)
2. With any configuration where it happens in both OGL and DX9 (problem should be visible)

#18 - 02/12/2012 06:18 PM - jayork42

This is still a problem with the latest binary, 3.0-415. This still slways happens with DX9. Same behavior with OpenGL, issue only appears if setting the EFB resolution is greater than 1x or enabling AA. I'm unable to test DX11 because I'm running Windows XP. Can someone test this with the 3.0-415 32-bit build?

Here are the 2 fifo snapshots. DX9.dff is DX9 with EFB resolution set to 1x and no AA. OGL_15.dff is OpenGL using 1.5x and no AA. I would've tested AA between plugins but these settings differ, so we should ignore AA for the moment to keep variables to a minimum.

http://www.mediafire.com/?bb3er2bk2vhhna5
http://www.mediafire.com/?23rbm2q792fpoj9

I may have a hypothesis as to what's going on here. I think we can agree by now that the edges of the mapped texture are nontransparent when this problem appears. Let's also assume the game engine is always stretching this texture to infnity in all directions. My guess for this is when upscaling EFB copies to 1.5x and above, the area of the actual image of the shadow exceeds the area of the bounding box used by the game engine to map the shadow to the terrain. I'm thinking the size of this bounding box doesn't change, yet the engine has no way of detecting that there are nontransparent pixels out of bounds of the box (because all of these settings are hacks). Because AA also upscales everything, this must be why AA breaks this too (but should AA really effect EFB copies other than the main frame buffer?) The images below explain this; the first being 1x and the second 1.5x.

These are EFB copies ripped using "Dump EFB to texture." With 1x the frame size is 128x128, with 1.5x it's 192x192, as expected. The image of the shadow is scaling with the frame.

#19 - 02/12/2012 06:30 PM - NeoBrainX

Thanks so far; Could you create another fifo log of the same scene with opengl but where the problem is not visible?

#20 - 02/12/2012 06:39 PM - jayork42

Hmm. That's not the case. According to Dimitri's screenshots the shadows are clearly NOT going out of bounds. Something else is going on here. If you move around you'll notice the streaks change colors, even when you're not moving. My next guess is that something in memory is overflowing into the space used for the current shadow frame. Maybe a texture that's being upscaled next to this buffer. The interesting thing is these pixels appear to be garbage. Perhaps a nearby texture of a different format? If you move around you will eventually notice the lower right skew is flashing in a predictable pattern. This appears to be reflecting the current memory space used for the animated coin on screen as the flashing pattern is spot on with the coin's looping animation. If someone takes the time to debug VRAM this should explain what's happening.

#21 - 02/12/2012 06:44 PM - jayork42

Ok, OpenGL with 1x internal res:

http://www.mediafire.com/?r143ha4neyx8f9n

#22 - 02/12/2012 06:56 PM - jayork42

Wow, the fifo logs don't show this at all. All 3 snapshots are normal. :\

#23 - 02/12/2012 06:58 PM - NeoBrainX

Funnily enough, all three fifo logs show the reported issue for me.. :P

#24 - 02/12/2012 07:12 PM - jayork42

Well, better you than me. :P

#25 - 04/07/2012 08:26 AM - skidau

issue 5346 has been merged into this issue.

#26 - 04/08/2012 03:37 AM - copperstone123

I tried using Direct311 and the black grapchis went awy but the game went slow I am using dolphin 3.0win64.
\

#27 - 05/04/2012 08:20 AM - pascal.jouy

Issue is fixed, but only in D3D9/11 for now.
Indeed, the game seems slower than before (since the 2.0 builds), even if SPEED shows 100% or around. Don't know why.

#28 - 05/05/2012 08:56 PM - www.williamfeely.info

Just tried Dolphin 3.0-600 build and it is NOT fixed for me in Direct3D9 UNLESS the Internal Resolution is set to 1x.

#29 - 07/12/2012 04:52 AM - Anonymous

The issue doesn't seem to be related to the specific IR value. For me, the game looks correct in two circumstances:
2.5x IR with EFB scaled copy disabled
auto (window size) IR with EFB scaled copy disabled, and i randomly change window size

perhaps it is related to some discrepancy between "window size" and the actual rendering size not being linearly related.

#30 - 07/12/2012 04:58 AM - Anonymous

The above comment was with ogl. with dx9 the issue still exists, and can be triggered by just opening the graphics options dialog...
fun... :p

#31 - 04/10/2013 01:55 AM - Billiard26

- Issue type set to Bug

#32 - 08/10/2013 10:49 AM - NeoBrainX

Oddly enough, the issue shows up in the native dff player (commit 43f2c9d92be293c712686213311ca187f9c40aa9), too.

Maybe some texture cache issue with preloading or something?

#33 - 08/10/2013 11:00 AM - NeoBrainX

Can you create a new 2-frame fifo log using a revision later than revision 198d60c56964fc8a9c679cdf9ab9805c684146ef ?

#34 - 12/09/2013 12:11 PM - JMC4789

I found out the one surefire way to make the problem go away at higher resolutions IS... to keep dragging the window size as you play. The game's effects still draw properly, but the game runs at 30 fps/60 VPS and it's kinda hard to see. But still, it's something? I think?

#35 - 01/05/2014 12:58 PM - NeoBrainX

Poking again for a 2-frame fifo log using a revision later than revision ed67d1ae2f96 .

#36 - 01/05/2014 01:06 PM - JMC4789

Ask and ye shall receive.

https://dl.dropboxusercontent.com/u/484730/SonicHeroesNew.7z

#37 - 04/20/2014 07:59 AM - magumagu9

Comment 18 is on the right track. The game is trying to be a bit clever and skip some register updates, so there are four textures active when it draws the whole road: the road texture, and textures for the shadows of each of the three characters. The problem is, we aren't computing these textures accurately: http://imgur.com/00QkZvO . Note in particular the top line of the texture. That color tints the road, leading to the discoloration.

For comparison: it currently looks like http://imgur.com/jiRSCsv , and it should look like http://imgur.com/Q63RDhd .

I still need to look into how exactly the texture gets corrupted.

#38 - 04/20/2014 08:32 AM - magumagu9

Oh, and one other thing that might be important. The shadows textures are dynamically computed: the game draws a model of Sonic on top of a black background, then composites the texture on top of itself to blur the shadow.

#39 - 04/20/2014 08:41 AM - JMC4789

Excellent work deducing it; but is there any explanation why certain IRs + Window Sizes manages to avoid the error?

#40 - 04/20/2014 09:07 AM - magumagu9

I think comment #13 actually provides the answer to that: the problem with the shadows only happens when the top line of the screen has garbage pixels. The game draws the shadow texture in the top left corner of the screen. So the problem with shadows is actually the same issue as the garbage pixels at the top of the screen (which basically boils down to us using floating-point math to compute the top-left corner of the EFB, and rounding the result inconsistently).

#41 - 04/20/2014 07:57 PM - NeoBrainX

Note: The fifo log provided in comment #39 is unsuitable for debugging.

I uploaded a good fifo log at https://dl.dolphin-emu.org/nbx/dffs/sonic_heroes_issue_267.dff . Indeed, the software renderer does not suffer from the issue pointed out here.

#42 - 04/21/2014 02:06 AM - magumagu9

There was a bunch of discussion on IRC about this today. Quick summary:

1. This game clears the screen by drawing a black rectangle where the top left corner is something like (0.50007, 0.50009) in screen coordinates.
2. Due to the way the console does rasterization, this coordinate covers the pixel at (0,0).
3. There's a bug in the OpenGL backend which means we don't rasterize the black rectangle correctly at 1xIR. This doesn't affect the D3D backend for reasons which we haven't figured out yet. neobrain is looking into it.
4. Anything other than 1xIR not working is a consequence of the fact that higher IR levels aren't accurate emulation: if we pretend the resolution is twice as high, the top left corner of the rectangle is at (1, 1), and that simply doesn't cover the corner of the screen. It might be possible to enhance 2xIR mode to deal with cases like this more gracefully, or we could use something game-specific like a cheat code, but it's a separate issue.

#43 - 04/22/2014 01:30 PM - NeoBrainX

More updates:
- we now know for sure that each pixel is divided into a 12x12 grid with one grid point per subsample.
- hence it seems logical to have (fixed-point) screen coordinates as multiples of 1/12th.
- in the default configuration, multisampling is disabled, hence all subsamples are concentrated in the center of the subsampling grid.
- As tested in hwtests (https://github.com/dolphin-emu/hwtests/blob/872cd517cc3f8edb20161506109344652bbd02b5/gxtest/source/main.cpp#L498), the screen coordinate of the center of the subsampling grid is *roughly* 7/12th (i.e. 0.58333333). However, the tests currently indicate that it's somewhat less than that, but I'm not sure if that's really true or just because the test suffers from rounding errors.
- the point is: the screen coordinate of (what we can consider to be) the pixel center is *not* at 0.5 as is the case normally*, but slightly shifted further to the bottom-right.
- hence, the current hw renderers put the black rectangle mentioned in point 1 of comment #49 at (1,1) instead of (0,0). I'm not completely sure why D3D works fine with 1xIR, and it really shouldn't.

As far as I can tell, a perfectly fine approximation of this behavior would be to shift all of our screen space coordinates by 0.08333333 (as a hardcoded value). Given that the reason for this offset is well-understood but the actual fix (namely to emulate limited screen space coordinate precision) is needlessly complicated and probably won't yield any better emulation, I'd say we should indeed go with that hardcoded solution. Note that I don't see any good way to fix this for non-native rendering resolutions.

Any other opinions for this?

- As far as I could tell, both D3D and OpenGL guarantee the pixel center to be located at (0.5,0.5).

#44 - 04/22/2014 01:38 PM - NeoBrainX

P.S.: With a ridiculous amount of effort and care, it would be possible to fix this game even for higher internal resolutions. What you basically need to do at the end of the vertex shader stage is to preliminary convert the output coordinates to the theoretical screen space of 1xIR and then transform it back to clipspace such that the output is the same. Note that I'm intentionally being vague here because I'm not even sure myself how it would look in detail, nevertheless it will likely be quite the mess.

#45 - 04/25/2014 08:00 AM - JMC4789

- Status changed from Accepted to Fixed

Fixed by 4.0-1474 - https://github.com/dolphin-emu/dolphin/commit/25f5598e31d5ce5dca8d1b03888a9205b616280e

#47 - 03/15/2015 11:27 AM - varalakshmi0806

my I get to know the code to show movements of dolphin and the dolphins playing with a ball

#48 - 12/29/2015 01:37 PM - seapancake

- File G9SP8P-2.png added

- File G9SP8P-4.png added

- File G9SP8P.dff added

This issue is still present both in 4.0-8459 as well as 4.0-1474 which was supposed to have fixed this issue. Present on both ends whenever AA is enabled regardless of IR. 3-frame FIFI log attached as well.

Initial issue describes this is a general issue however later on there is a series of comments relating to AA so figure best to re-open this, if not happy to log a new bug.

#49 - 12/29/2015 02:07 PM - seapancake

Nevermind, didn't read this is a known issue at when using AA.

Files

G9SP8P-2.png	2.27 MB	12/29/2015	seapancake
G9SP8P-4.png	2.28 MB	12/29/2015	seapancake
G9SP8P.dff	4.63 MB	12/29/2015	seapancake