Subject: : AmigaOS4
Topic: : The OpenGL ES 2.0 thread
Re: The OpenGL ES 2.0 thread
Author: : kas1e
Date: : 2019/12/25 19:07:28
URL:

@All
I am not Daniel, but as there is XMAS time and everyone is busy, and news on amigans.net is didn't appears with changelog, there is what was changed since v2.9 up to current 2.11:

- Fix: glScissor didn't gracefully handle negative XY or height. Thanks to Kasie Kasovich (kas1e) for reporting.

- Improvement: under certain circumstances, ogles2 could be triggered to resize its internal client-RAM-emu-VBOs multibuffer to very very large values, in fact so large that Nova allocated almost all VRAM for those. At least on some Nova/RadeonHD.chip/graphics.lib setups the VBO resizing also triggered a crash bug in Nova's DeleteBuffer function under such low VRAM conditions. This improvement therefore also acts as a workaround for that issue. This fixes all such issues with some of kas1e's Irrlicht engine examples. Thanks to Kasie Kasovich (kas1e) for reporting.

- Fix: there was a bug in the copy-converter for 8bit and 16bit index arrays (Nova doesn't support 8bit indices and is very slow with 16bit indices, therefore those are converted to 32bit internally, even if the ogles2-client passes them inside his own VBO, then a "hidden" 32bit clone is prepared and used) which in theory could result in the indices not being updated at all. It was extremely unlikely to happen and AFAIK it never did, but anyway: fixed.

- Workaround/Fix: if you attached a GL_BGRA_EXT texture as color render target to an FBO, then the texture's red and blue channels would appear to be swapped when you later used that texture for rendering. The reason for this is that the BGRA texture's internal pixel format is actually RGBA, only that its channels are "swizzled". At the moment Nova ignores that swizzle setting when rendering to the texture. The current workaround is to reset any eventual swizzle to the defaults if the texture is being used as FBO render target, so simply spoken an BGRA texture becomes an RGBA texture if you use it as render target. This fixes color bugs in e.g. an irrlicht demo and Tuxkart. Thanks to Kasie Kasovich (kas1e) for reporting.

- maintenance, C++ modernizations: all attribute initializations moved from constructor initializer lists into the respective class definitions. Makes the code smaller and it's harder to forget something (and oops, I actually found one uninitialized attribute during the process (uncritical though)...).

- maintenance, C++ modernizations: "auto" return-value replacement run on all inline functions, where appropriate.

- (small) hashing functions speed-up (without hash quality loss).

- Fix: a bug in the uniform shader variable handler resulted in same-named (and thus logically identical) vertex- and fragment-uniforms to eventually be assigned different virtual location numbers. If the user issued a glUniform(location_number,xyz) call then the lib would only find and set the vertex-shader's uniform and skip the writing to the fragment-shader's uniform because it would simply not find it. OpenGL doesn't distinguish between such same-named but physically different uniform variables in the shaders of a GPU program - in stark contrast to Nova, where all that info is shader and not shader-program based. Consequently Nova's ShaderGetObjectInfo, which is used to query uniform variable information, works on shaders only. Also, Nova only gives us a GLSL / GL compatible "location" info if that has been explicitly defined inside the shader, otherwise, it doesn't generate one but only returns W3DN_NOLOCATION. Because of all this ogles2.lib has to generate such location Infos by itself, in a GL compatible way. This means that in case of a same-named variable in both of a program's shaders the same uniform location has to be enforced. Which is where the flaw was. In gl4es this bug manifested in a fog not working. Thanks to kas1e for reporting and testing!