Subject: : AmigaOS4
Topic: : Porting to AmigaOS4 thread
Re: Porting to AmigaOS4 thread
Author: : Raziel
Date: : 2020/11/30 10:52:08
URL:

I get a new crash, unfortunately pretty basic (again) but at least it's not in the same place, so here's hoping it comes from somewhere else (and can be fixed...maybe it's even because of unsupported functions, who knows?)

Here's the crashlog (excerpt)

```
 Crash log for task "scummvm"
Generated by GrimReaper 53.19
Crash occured in module scummvm at address 0x7B85B448
Type of crash: DSI (Data Storage Interrupt) exception
Alert number: 0x80000003

Register dump:
GPR (General Purpose Registers):
  0: BD03CA8C 3B87EA10 00000000 3C00FBB0 00000D33 3C00FBB8 00020020 3C00FBD0
  8: FFFFFFFF 00000000 00000000 0000001A FFFFFFDF 388A7AAC 00000000 5036C000
 16: 3C02EA68 00000000 388A01F0 3C029FB8 00000000 00000000 00000000 00000000
 24: 00000000 00000000 00000000 3B87EA80 3B87EA18 3C00FBB0 38897E54 3B87EA60


FPR (Floating Point Registers, NaN = Not a Number):
  0:          nan            1            1            1
  4:            1  9.06029e-305          790          675
  8:          698         0.04            1  4.5036e+15
 12:           16  4.5036e+15            0            0
 16:            0            0            0            0
 20:            0            0            0            0
 24:            0            0            0            0
 28:            0            0          0.1  4.5036e+15

FPSCR (Floating Point Status and Control Register): 0x82004000


SPRs (Special Purpose Registers):
        Machine State (msr) : 0x0200B030
            Condition (cr) : 0x3D266BD4
    Instruction Pointer (ip) : 0x7B85B448
```

Xtended Exception (xer) : 0x38A46018
Count (ctr) : 0x6FF494D0
Link (lr) : 0x7FAF4920
DSI Status (dsisr) : 0x38A45DD0
Data Address (dar) : 0x021AD048

680x0 emulated registers:
DATA: 96465700 00000000 00000000 00000000 00000000 00000000 00000000 00000000
ADDR: 6FFB8700 96461500 00000000 00000000 00000000 00000000 00000000 3B87E360
FPU0: 0 0 0 0
FPU4: 0 0 0 0

Symbol info:
Instruction pointer 0x7B85B448 belongs to module "scummvm" (PowerPC)
Symbol: _ZN6OpenGL9ContextGL10initializeENS_14ContextOGLTypeE +
0x8C in section 12 offset 0x002B9ED0

Stack trace:
[graphics/opengl/context.cpp:73] scummvm:_ZN6OpenGL9ContextGL10initializeENS_14ContextOGLTypeE
()+0x8c (section 12 @ 0x2B9ED0)
[graphics/opengl/context.cpp:59] scummvm:_ZN6OpenGL9ContextGL10initializeENS_14ContextOGLTypeE
()+0x78 (section 12 @ 0x2B9EBC)
[backends/platform/sdl/sdl.cpp:218] scummvm:_ZN11OSystem_SDL11initBackendEv()+0x90 (section 12 @
0x6EBC8)
[base/main.cpp:478] scummvm:scummvm_main()+0x778 (section 12 @ 0x715C0)
[backends/platform/sdl/amigaos/amigaos-main.cpp:79] scummvm:main()+0x158 (section 12 @ 0x6FF30)
native kernel module newlib.library.kmod+0x000020a4
native kernel module newlib.library.kmod+0x00002d0c
native kernel module newlib.library.kmod+0x00002ee8
scummvm:_start()+0x170 (section 12 @ 0x1AB8)
native kernel module dos.library.kmod+0x000255c8
native kernel module kernel+0x000420ac
native kernel module kernel+0x000420f4

PPC disassembly:
7b85b440: 93fd0004 stw r31,4(r29)
7b85b444: 3be10050 addi r31,r1,80
*7b85b448: 807a0000 lwz r3,0(r26)
7b85b44c: 812300f8 lwz r9,248(r3)
7b85b450: 7d2903a6 mtctr r9

System information:

CPU
Model: P.A. Semi PWRficient PA6T-1682M VB1
CPU speed: 1800 MHz
FSB speed: 900 MHz

Extensions: altivec

Machine
Machine name: AmigaOne X1000
Memory: 2097152 KB
Extensions: bus.pci bus.pcie

and here's the file where it complains (I know that it still says ResidualVM, they didn't change it yet)
https://github.com/scummvm/scummvm/blo ... aphics/opengl/context.cpp

I can see two things:
1) It's again the glContext i got a crash from earlier (but this crashlog was done *with* the fix already in place, so i guess it's close, but not yet there)
2) The first affected line points to something related to FBO (which i know is not supported in OpenGL, but is maybe the reason why it bombs?)

If anyone has any hints, workarounds i could try or other help, please shout

```
/* ResidualVM - A 3D game interpreter
 *
 * ResidualVM is the legal property of its developers, whose names
 * are too numerous to list here. Please refer to the COPYRIGHT
 * file distributed with this source distribution.
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
 *
 */

#include "graphics/opengl/context.h"

#include "common/debug.h"
#include "common/str.h"
#include "common/textconsole.h"
#include "common/tokenizer.h"

#include "graphics/opengl/system_headers.h"

#if defined(USE_OPENGL_GAME) || defined(USE_OPENGL_SHADERS) || defined(USE_GLES2)
```

```
namespace Common {
DECLARE_SINGLETON(OpenGL::ContextGL);
}

namespace OpenGL {

ContextGL::ContextGL() {
    reset();
}

void ContextGL::reset() {
    maxTextureSize = 0;

    NPOTSupported = false;
    shadersSupported = false;
    framebufferObjectSupported = false;
    packedDepthStencilSupported = false;
    unpackSubImageSupported = false;
    framebufferObjectMultisampleSupported = false;
    OESDepth24 = false;
    multisampleMaxSamples = -1;
}

void ContextGL::initialize(ContextOGLType contextType) {
    // Initialize default state.
    reset();

    type = contextType;

    // Obtain maximum texture size.
    glGetIntegerv(GL_MAX_TEXTURE_SIZE, (GLint *)&maxTextureSize);
    debug(5, "OpenGL maximum texture size: %d", maxTextureSize);

    const char *extString = (const char *)glGetString(GL_EXTENSIONS);

    bool ARBShaderObjects = false;
    bool ARBShadingLanguage100 = false;
    bool ARBVertexShader = false;
    bool ARBFragmentShader = false;
    bool EXTFramebufferMultisample = false;
    bool EXTFramebufferBlit = false;

    Common::StringTokenizer tokenizer(extString, " ");
    while (!tokenizer.empty()) {
        Common::String token = tokenizer.nextToken();

        if (token == "GL_ARB_texture_non_power_of_two" || token == "GL_OES_texture_npot") {
            NPOTSupported = true;
        } else if (token == "GL_ARB_shader_objects") {
            ARBShaderObjects = true;
```

```
    } else if (token == "GL_ARB_shading_language_100") {
        ARBShadingLanguage100 = true;
    } else if (token == "GL_ARB_vertex_shader") {
        ARBVertexShader = true;
    } else if (token == "GL_ARB_fragment_shader") {
        ARBFragmentShader = true;
    } else if (token == "GL_EXT_framebuffer_object") {
        framebufferObjectSupported = true;
    } else if (token == "GL_EXT_packed_depth_stencil" || token == "GL_OES_packed_depth_stencil") {
        packedDepthStencilSupported = true;
    } else if (token == "GL_EXT_unpack_subimage") {
        unpackSubImageSupported = true;
    } else if (token == "GL_EXT_framebuffer_multisample") {
        EXTFramebufferMultisample = true;
    } else if (token == "GL_EXT_framebuffer_blit") {
        EXTFramebufferBlit = true;
    } else if (token == "GL_OES_depth24") {
        OESDepth24 = true;
    }

  }

  int glslVersion = getGLSLVersion();
  debug(5, "OpenGL GLSL version: %d", glslVersion);

  if (type == kOGLContextGLES2) {
    // GLES2 always has (limited) NPOT support.
    NPOTSupported = true;

    // GLES2 always has shader support.
    shadersSupported = true;

    // GLES2 always has FBO support.
    framebufferObjectSupported = true;

    // ScummVM does not support multisample FBOs with GLES2 for now
    framebufferObjectMultisampleSupported = false;
    multisampleMaxSamples = -1;
  } else {
    shadersSupported = ARBShaderObjects && ARBShadingLanguage100 && ARBVertexShader &&
ARBFragmentShader && glslVersion >= 120;

    // Desktop GL always has unpack sub-image support
    unpackSubImageSupported = true;

    framebufferObjectMultisampleSupported = EXTFramebufferMultisample && EXTFramebufferBlit;

    if (framebufferObjectMultisampleSupported) {
        glGetIntegerv(GL_MAX_SAMPLES, (GLint *)&multisampleMaxSamples);
    }
  }
```

```
    // Log context type.
    switch (type) {
        case kOGLContextGL:
            debug(5, "OpenGL: GL context initialized");
            break;

        case kOGLContextGLES2:
            debug(5, "OpenGL: GLES2 context initialized");
            break;
    }

    // Log features supported by GL context.
    debug(5, "OpenGL: NPOT texture support: %d", NPOTSupported);
    debug(5, "OpenGL: Shader support: %d", shadersSupported);
    debug(5, "OpenGL: FBO support: %d", framebufferObjectSupported);
    debug(5, "OpenGL: Packed depth stencil support: %d", packedDepthStencilSupported);
    debug(5, "OpenGL: Unpack subimage support: %d", unpackSubImageSupported);
}

int ContextGL::getGLSLVersion() const {
    const char *glslVersionString = (const char *)glGetString(GL_SHADING_LANGUAGE_VERSION);
    if (!glslVersionString) {
        warning("Could not get GLSL version");
        return 0;
    }

    const char *glslVersionFormat;
    if (type == kOGLContextGL) {
        glslVersionFormat = "%d.%d";
    } else {
        glslVersionFormat = "OpenGL ES GLSL ES %d.%d";
    }

    int glslMajorVersion, glslMinorVersion;
    if (sscanf(glslVersionString, glslVersionFormat, &glslMajorVersion, &glslMinorVersion) != 2) {
        warning("Could not parse GLSL version '%s'", glslVersionString);
        return 0;
    }

    return glslMajorVersion * 100 + glslMinorVersion;
}

} // End of namespace OpenGL

#endif
```