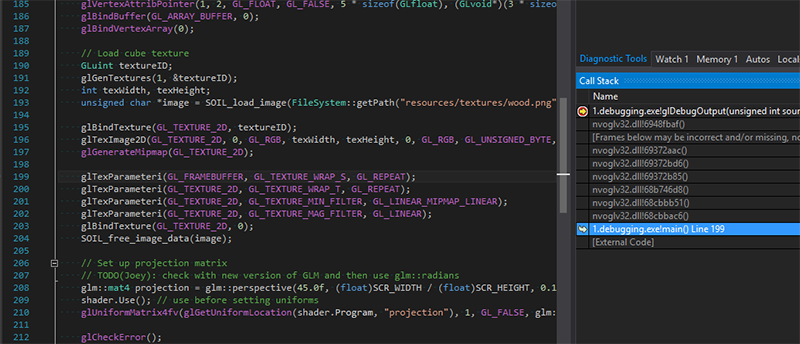
Debugging 2

Backtracking the debug error source

-이전에서 error타입, 위치, 등을 알아냈는데 이러한 오류가 발생한 이유와 라인을 알게 된다면 코드에서 중단점을 설정해 어느 위치에서 오류가 일어났는지 확인하기가 매우 좋음



Custom error output

-glDebugMessgaeInsert를 이용해서 디버그 출력 시스템에 메시지를 넣을 수 있음

|  |
| --- |
| glDebugMessageInsert(GL\_DEBUG\_SOURCE\_APPLICATION,  GL\_DEBUG\_TYPE\_ERROR, 0,  GL\_DEBUG\_SEVERITY\_MEDIUM, -1, "error message here"); |

-다른 응용 프로그램이나 OpenGL 코드에 연결하는 경우 유용하게 쓰임

-다른 개발자가 내가 커스텀한 OpenGL에서 발생하는 오류를 알아내는데 도움이 됨

-디버그 출력, 오류 출력은 오류를 빠르게 잡아내는데 큰 도움을 줌

Debugging shader output

-GLSL인 shader에서는 glGetError와 같은 함수로 엑세스하는 방법이 없음

-단순 구문 오류일 경우에는 찾기 수월하지만, semantic 의미 오류일 경우에는 찾아내기가 어려움이 있음

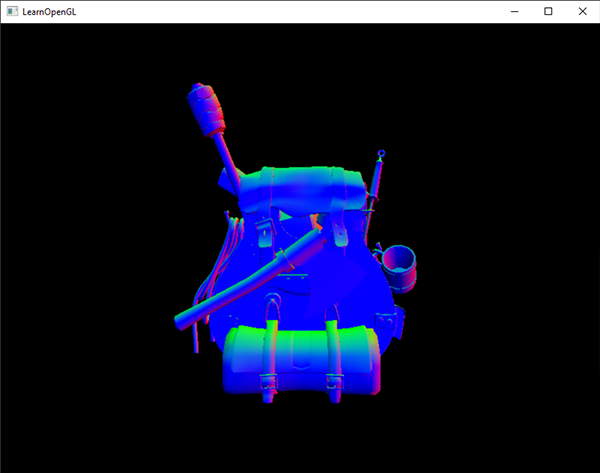
-shader의 문제점을 파악하기 위해서 사용되는 방법은 쉐이더 프로그램의 모든 관련 변수를 fragment shader의 출력 채널로 직접 전송하여 알아내는 방법이 있음

-변수를 모두 fragment shader에서 컬러 채널로 출력함으로 시각적인 결과를 검사해서 정보를 알아낼 수 있음

-ex) normal vector가 올바른지 확인하는 방법

-normal vector들을 각 fragment shader에서 확인

|  |
| --- |
| #version 460 core  out vec4 FragColor;  in vec3 Normal;  [...]    void main()  {  [...]  FragColor.rgb = Normal; //normal vector을 color에 넣어서 확인  FragColor.a = 1.0f;  } |



-색상으로 빠르게 확인할 수 있음

-만약 검은색인 경우에는 normal vector가 shader에 전달이 되지 않았음을 의미함

-색상이 이상한 경우 normal vector에 이상이 있음을 의미함

OpenGL GLSL reference compiler

-NVIDIA, ATI/AMD의 차이로 인해 한 컴퓨터의 shader가 다른 컴퓨터에서 작동하지 않을 수 있음

-Shader code가 모든 종류의 그래픽 카드에서 실행되는지 확인하고 싶으면 OpenGL GLSL reference compiler을 사용해서 직접 확인 할 수 있음

<https://www.khronos.org/opengles/sdk/tools/Reference-Compiler/>

<https://github.com/KhronosGroup/glslang>

-바이너리 GLSL lang validator가 주어지면 첫 번째 인수로 shader code를 넣으면 확인을 쉽게 할 수 있음

-확장자 목록

-.vert : vertex shader

-.frag : fragment shader

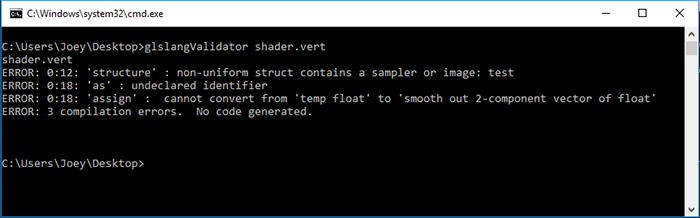
-.geom : geometry shader

-.tesc : tessellation control shader

-.tese : tessellation evaluation shader

-.comp : compute shader

|  |
| --- |
| glsllangvalidator shaderFile.vert // shaderFile의 이름인 vertex shader 전달 |

-AMD, NVIDIA, Intel GLS Compiler 간에 미묘한 차이를 알려주지는 못함

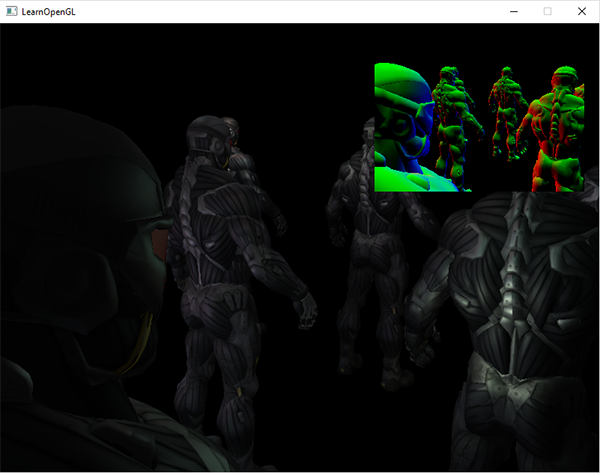
-shader을 바로 확인할 수 있다는점에는 도움이 됨

Framebuffer output

-다른 유용한 방법은 미리 정의된 영역의 화면의 일부를 framebuffer을 출력하는 것임

-단, 프레임버퍼가 texture attachments 이어야 가능함

|  |
| --- |
| // vertex shader  #version 460 core  layout (location = 0) in vec2 position;  layout (location = 1) in vec2 texCoords;  out vec2 TexCoords;  void main()  {  gl\_Position = vec4(position, 0.0f, 1.0f);  TexCoords = texCoords;  }  //-----------------------------------------------------------------------  // fragment shader  #version 460 core  out vec4 FragColor;  in vec2 TexCoords;    uniform sampler2D fboAttachment;    void main()  {  FragColor = texture(fboAttachment, TexCoords);  } |
| void DisplayFramebufferTexture(unsigned int textureID)  {  if (!notInitialized)  {  // initialize shader and vao w/ NDC vertex coordinates at top-right of the screen  [...]  }    glActiveTexture(GL\_TEXTURE0);  glUseProgram(shaderDisplayFBOOutput);  glBindTexture(GL\_TEXTURE\_2D, textureID);  glBindVertexArray(vaoDebugTexturedRect);  glDrawArrays(GL\_TRIANGLES, 0, 6);  glBindVertexArray(0);  glUseProgram(0);  }    int main()  {  [...]  while (!glfwWindowShouldClose(window))  {  [...]  DisplayFramebufferTexture(fboAttachment0);    glfwSwapBuffers(window);  }  } |

-우측상단에 작은 화면을 만들어 Normal vector가 렌더러에서 잘 표시되었는지 확인하는데 사용하는 모습

External debugging software

-외부 도구를 사용해서 디버깅 작업에 도움을 줄 수 있음

-이러한 도구는 OpenGL 기능사용 프로파일링, 병목 현상 찾기, 버퍼 메모리 검사, 텍스처 및 프레임 버퍼 첨부 파일 표시 등 여러 방법으로 도움을 줌

-여러 프로그램들이 있는데 그 중, RenderDoc, CodeXL, NVIDIA Nsight에 대해 설명

Render Doc

<https://github.com/baldurk/renderdoc> // 완전한 오픈소스

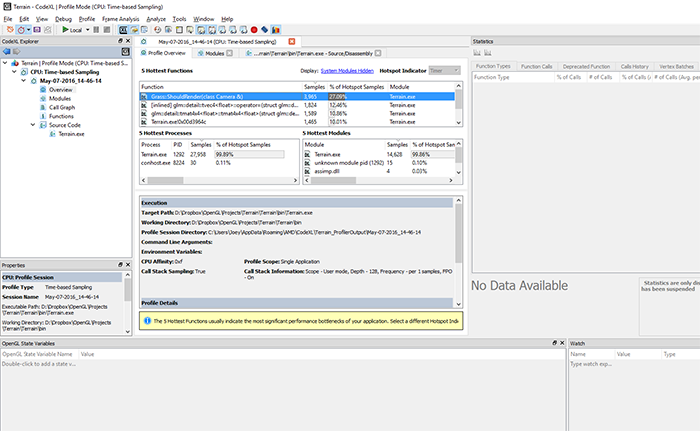
-사용할 때 작업 폴더와 실행 파일을 지정하면 프로그램이 정상적으로 실행되어짐

-특정 프레임을 검사할 때 마다 Render Doc이 실행 파일의 하나 이상의 프레임을 캡처하도록 하는데, 캡처된 프레임 내에서 파이프라인 상태, OpenGL 명령어, 버퍼 저장소, 사용 중인 텍스처를 볼 수 있음

Code XL

<https://gpuopen.com/archived/legacy-codexl/>

<https://github.com/GPUOpen-Archive/CodeXL>

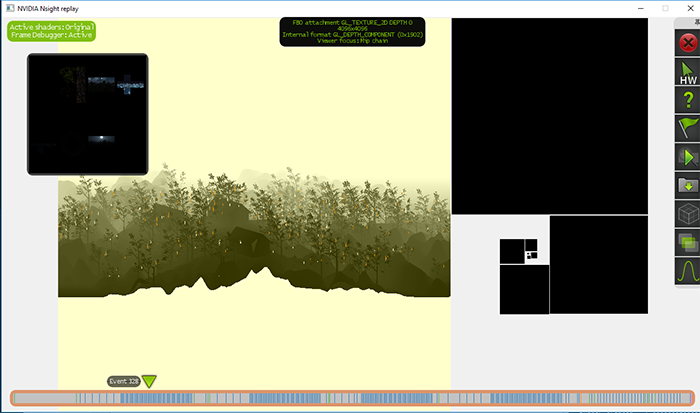
-Visual Studio의 plugin으로 출시된 GPU 디버깅 도구

-NVIDIA나 Intel 그래픽 카드에서도 작동하지만, OpenCL Debugging은 지원하지 않음

NVIDIA Nsight

<https://developer.nvidia.com/nsight-visual-studio-edition> // visual studio 플러그인

<https://developer.nvidia.com/nsight-graphics> // 독립형 프로그램

-Nsight는 GPU사용과 프레임별 GPU상태에 관한 많은 런타임 통계를 제공함

-단, NVIDIA 그래픽에서만 작동함