|  |
| --- |
| **OpenGL (Basic)** |

|  |  |
| --- | --- |
| 적용 분야 | S/W |
| 소속 부서 | 의료영상SW팀 |
| 작성 일자 | 2018. 06. 15. |
| 작성자 | 김재혁 |
| 보안 등급 |  |
| 기여율 | 100% |

목차

[1 개요 3](#_Toc497721509)

[2 연구(검토) 목적 4](#_Toc497721510)

[3 연구(검토) 세부 내용 4](#_Toc497721511)

[4 적용 사례 및 예시 7](#_Toc497721516)

[5 추가 및 보완(개선) 사항 8](#_Toc497721517)

[6 향후 계획 9](#_Toc497721518)

[7 참고 문헌 및 자료 10](#_Toc497721519)

# 개요

|  |  |  |  |
| --- | --- | --- | --- |
| 문서 구분 | ■ 기술 보고 □ 선행 조사 □ 경쟁사 분석 □ 기타 | | |
| 개발 구분 | ■ 개발용 □ 양산용 □ 선행 검토용 ■ 교육용 □ 기타 | | |
| 적용 분야 | □ 공통 ■ 의료기기 □ 산업기기 □ 신사업 □ 기타 | | |
| □ HW ■ SW □ FPGA/FW □ 광학 □ 기구 □ 기타 | | |
| 적용 모델 |  | | |
| 보고서 이름 | OpenGL (Basic) | | |
| 작성일 | 2018. 03. 15 | 작성 부서 | 의료영상SW팀 |
| 검토 기간 |  | | |
| 시험 기간 |  | | |
| 시험 시료 |  | | |
| 시험 장비  프로그램 | Visual studio 2015, OpenGL, GLEW, GLFW, GLM | | |
| 키워드 | OpenGL | | |
| 요약 정리 | OpenGL사용법을 알아보자. | | |

# 연구(검토) 목적

OpenGL의 기본적인 사용법을 알아보자.

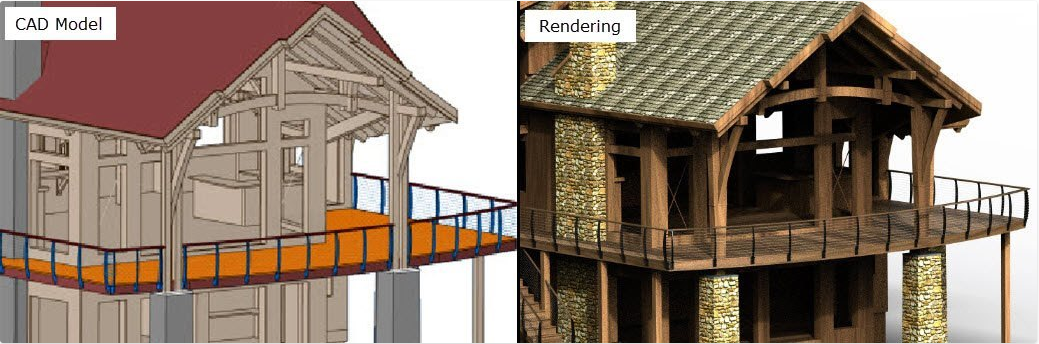
# 연구(검토) 세부 내용

## Graphics Fundamental

### 3D Modeling

* 3D Model은 점들의 집합을 다양한 Geometric entities을 연결하여 3D공간에서 Object의 형태를 만들어 내는 것이다.
* 3D Model Categories
  + Solid  
    : Object의 볼륨(Volume)을 정의하는 Model들이다. 대게 Engineering과 Medical simulation에서 사용된다.
  + Shell/boundary  
    : Object의 표면(Surface)을 나타내는 Model들이다. 대게 게임과 영화 같은 분야에서 사용되는 visual model에서 사용된다.

### Rendering

* 컴퓨터 프로그램을 이용하여, 2D or 3D 모델로부터 그림이나 애니메이션 등의 이미지를 생성해 내는 과정을 의미한다.
* 아래 이미지의 좌측이 Modeling의 결과이고, 우측이 Rendering의 결과 영상이다.  
    
  <출처 : https://www.keuka-studios.com/our-design-process/>

### Visualization

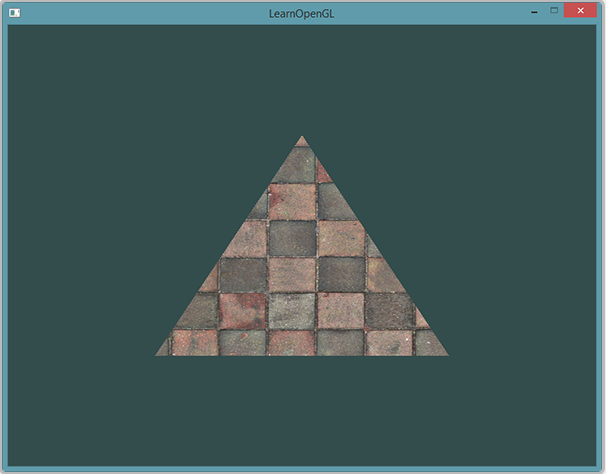
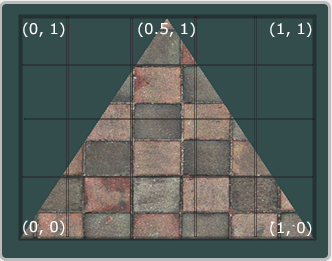
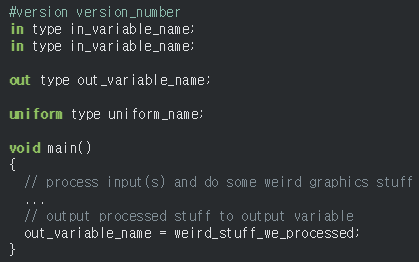
* 사용자의 데이터를 Graphic primitive으로 변환하는 과정  
  ※ Graphic primitive: Rendering을 통해 이미지로 변환할 수 있는 그래픽 데이터   
   ex) 점, 선, 면 등
* 즉, 일반 데이터를 이미지로 표현하기 위한 그래픽 데이터로 바꾸는 과정

## OpenGL

### OpenGL 개요

* OpenGL(Open Graphics Library)은 Silicon Graphics에서 만든 Graphic H/W와의 Interface 역할을 하는 Graphics API이다. (현재는 Khronos Group에 의해 관리되고 있다.)
* MS사의 Direct3D와 함께 컴퓨터 그래픽 세계를 양분하고 있다.
  + Direct3D: Windows 환경에서의 Graphics API이며, 모든 Programming language들은 COM이다. OpenGL에 비해 Graphic card의 최신기술에 대한 Update가 빠르게 이루어 진다.
  + OpenGL: Windows환경에서만 제공하는 Direct3D와 달리 다양한 Platform에서 동작하는 Cross-platform Graphics API이다.
* 많은 Graphics API들이 OpenGL을 기반으로 만들어 졌다. Ex) Vtk
* OpenGL ES(OpenGL for Embedded Systems)는 OpenGL의 Subset으로 경량화되어 Mobile용으로 많이 사용된다.  
  <참조: <https://m.blog.naver.com/statistical/50116738539>>
* OpenGL로 만들어진 프로그램들은 Network를 통해 Server-Client 구조로 동작 할 수 있다.  
  (Client는 OpenGL 프로그램이 실행되는 PC, Server는 그리기를 수행하는 PC)

### Graphics Term (in OpenGL) <참조: http://schorsch.efi.fh-nuernberg.de/roettger/index.php/MedicalVisualization/OpenGLTerminology>

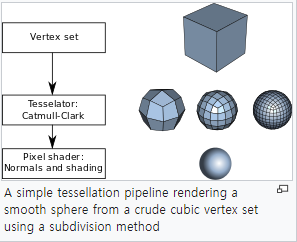
* Vertex
  + 정점의 정보를 가지고 있는 Data 구조체로 기본적으로 위치 정보를 갖는다.
  + Vertex는 일련의 Attribute들로 Position 외에도 다양한 Attribute를 갖는다.  
    ex) Color (Red, Green, Blue, Alpha), texture coordinate, normal vector, tangent vector 등
  + Vertex의 수가 늘어 날수록 해상도는 증가한다.
* Vertex stream
  + 정렬된 Vertex들의 List
* Primitive
  + Graphic으로 만들 수 있는 기본 요소들 (Vertex들로로 이루어져 있다.)로
    - 초창기에는 3D Object를 표시할 때, 삼각형과 사각형 중 어느 것이 더 좋은가에 대한 논쟁이 있었지만 이제 대부분의 3D Object는 삼각형으로 표시한다.  
      (OpenGL은 다양한 Type을 제공 하는 반면, OpenGL ES의 경우, 삼각형만을 제공한다)
  + OpenGL에서 Primitive는 두 가지로 사용된다.
    - Vertex stream이 무엇을 나타내는지 결정하는 OpenGL에서 사용하는 해석 체계
    - Primitive assembly 단계의 부분으로써, Vertex stream의 해석 결과
  + 따라서, 위 두 primitive 해석 중 하나를 사용해, vertex stream을 처리하면 Primitive의 순서가 지정되고, 각각의 Primitive는 base primitive라고도 불린다.
* Texture
  + OpenGL에서 사용하는 Object로 1개 이상의 Image를 포함한다.  
    (이 Image들은 모두 같은 Format을 갖는다.)
  + Model의 각 Vertex에 Texture의 Vertex를 일치시키는 작업을 Texture mapping이라 한다.
  + Texture는 두가지 방법으로 사용된다.
    - Shader의 입력 Data로 사용
    - 화면에 그려지는 Target으로 사용
  + Example) 아래 영상은 삼각형에 벽 Texture image가 맵핑 된 것이다.  
    
  + Sampling: Texture coordinate system을 사용해서 Texture color를 얻는 것.  
    < 2D Texture image의 Texture coordinate system >  
    
* Shader program
  + Vertex는 그 자체로 화면에 보여질 수 없으므로 프로그램의 처리를 통해 화면에 보여질 수 있는 정보로 변환되어야 하는데, 이런 역할을 하는 프로그램을 Shader라 한다.
  + 화면에 실제 보여지는 Pixel 정보로 까지 변환되는 과정은 Graphics pipeline에서 설명하기로 한다
  + Shader program들은 GLSL(OpenGL Shading Language)로 작성되고, 이는 C언어를 기반으로 변형된 언어이다.
    - Shader는 일반적으로 다음과 같은 구조를 갖는다.  
      
    - Vertex Attributes: 밖(in C++)에서 사용되는 정보들을 Vertex Shader에서 사용하기 위한 입력 값
      * 최소 16개로 GL\_MAX\_VERTEX\_ATTRIBS를 Query 함으로써 최대 Attributes를 얻어낼 수 있다.
* Frame buffer
  + Pixel들이 모니터로 표시되기 전, 저장되어 있는 메모리 공간
  + Resolution(해상도): Frame buffer에 저장 할 수 있는 Pixel 개수
  + Depth / Precision: 각 Pixel 마다 사용되는 Bit 수
  + Full Color System: Pixel 마다 24Bit 이상인 Graphic system. 다른 말로, True Color System / RGB Color System 이라고도 불림
  + HDR(High Dynamic Range) System: RGB 요소 하나가 12bit 보다 많은 수로 구성되어 있는 System. 더 선명하고 화려한 색을 표현할 수 있다.

### OpenGL State machine

* OpenGL은 유한개의 State들을 가진 Finite State Machine(FSM)이다.
  + State machine이란?  
    : State를 변경하는 명령을 받기 전에는 현재의 State를 계속 유지하는 machine을 말한다.
* 예를 들어, glColor3f(0,0,1)을 호출한 이후, 또다른 값으로 glColor3f(\*) 함수를 호출하지 않는 한 모든 Object들은 파란색으로 그려지게 된다.

### OpenGL Coordinate system

### Graphics Pipeline : OpenGL에서 Object를 Rendering 할 때의 연속적인 STEP

* Vertex Specification
  + Vertex의 specification을 정의 하는 과정
* Vertex Processing  
  : Programmable operation을 통해 Vertex 들을 Customize 하게 된다.
  + Vertex Shader
    - 각각의 Vertex를 Programming(GLSL)을 통해 처리하여, Vertex array를 3D 공간에 적절히 배치한다.
  + Primitive assembly
    - 이전 단계에서 처리된 Vertex들과 각각의 연결 정보를 근거로 하여, 구조화 하고,
  + Tessellation
    - Tessellation의 언어적 해석은 ‘모자이크 세공’ 이다.
    - 이전 단계에서 처리된 Vertex Data들의 Patch를 기반으로 primitive을 세분화 한다.  
      
    - Tessellation 과정은 3개의 Stage로 구분된다. 두 Stage는 Programmable Stage이고, 두 Stage의 사이에 있는 Stage는 고정된 기능을 갖는다.
      * Tessellation stage에서 사용되는 patch란 GL\_PATCHES란 상수 값에 의해 표시되는 Primitive type이다. Patch당 Vertice의 수는 Application level에서 정의할 수 있다. (glPatchParameteri() 함수 사용)
      * Tessellation Control Shader (TCS, Optional, Programmable)
        + Patch data를 조정하여, Primitive에 적용할 Tessellation의 양을 결정한다.
        + Tessellate된 인접 Primitive들의 연결을 확정한다.
      * Tessellation Primitive Generator
        + Patch data를 Input으로 받아, TCS 혹은 Default로 제공되는 값에 의해 세분화한다.(Subdivide)
      * Tessellation Evaluation Shader(TES, Optional, Programmable)
        + Tessellate된 Patch를 가지고 Vertex값을 Computing 한다.  
          ex) Interpolation

## OpenGL Coordinate Systems

### 3D Cartesian Coordinate System

* OpenGL
  + Right-handed

## OpenGL 설치

### 개요

* Pseudo-coloring에 기본으로 제공할 Palette들을 알아보자.
* DICOM 에서 제공하는 Well-known palette 들을 알아보자.

# 향후 계획

## Overlay 표시 기능 구현 : Localizer 를 Overlay로 미리 표시 하고 있는 영상 있음. (영상 Data만 출력 시, Overlay data 표시되지 않음) Example)

## Multi frame의 Concatenation Tag에 의한 분류 기능 개발

## 영상 전체의 위치를 표시하는 선이 아닌, 영상의 특정 위치를 선택 했을 때, 다른 Series에서 해당 영상을 표시하는 영상을 자동으로 표시하는 기능 개발

# 참고 문헌 및 자료

## OpenGL Overview

### http://duriansoftware.com/joe/An-intro-to-modern-OpenGL.-Table-of-Contents.html

### <http://www.songho.ca/opengl/index.html>

## Rendering Pipeline

### <https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview>

### <https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview>

### https://dai.fmph.uniba.sk/upload/2/24/Cg2\_lesson00.pdf

## [Palette](http://dicomiseasy.blogspot.kr/2013/06/getting-oriented-using-image-plane.html)

### Noura A. Semary, A PROPOSED HSV-BASED PSEUDOCOLORING SCHEME FOR ENHANCING MEDICAL IMAGES <https://airccj.org/CSCP/vol8/csit88321.pdf>

Vieworks Co., Ltd

41-3, Burim-ro, 170beon-gil,

Dongan-gu, Anyang-si, Gyeonggi-do

431-060 Republic of Korea

<http://www.vieworks.com>