

컴퓨터그래픽스

제2강 컴퓨터 그래픽스 소프트웨어

오늘의 학습목차

- 01 컴퓨터 그래픽스 영상의 표현
- 02 그래픽스 소프트웨어의 유형
- 03 OpenGL 프로그래밍

컴퓨터과학과 이병래 교수



01

컴퓨터 그래픽스 영상의 표현

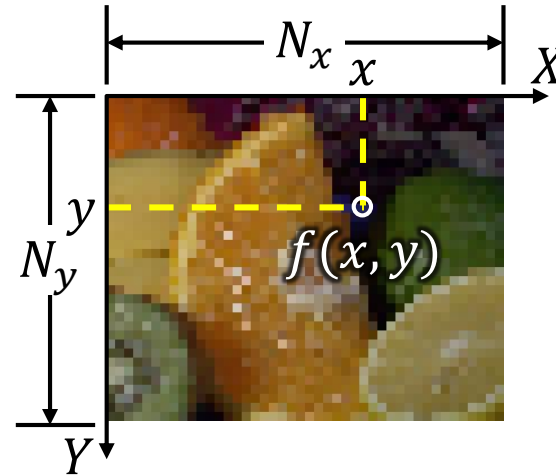
1. 래스터 그래픽스 영상
2. 벡터 그래픽스 영상



1. 래스터 그래픽스 영상

● 래스터 그래픽스 영상이란?

- 래스터 : CRT의 래스터 주사(raster scan) 방식에서 사용된 용어
- 사각형 격자 좌표 형태의 픽셀 배열로 표현됨
- 디지털 카메라, 스캐너 등을 통해 입력한 영상, MS Paint나 Photoshop, GIMP와 같은 영상 편집기를 이용하여 만든 영상
- TIFF, BMP, PNG, JPEG 등의 파일 형식으로 저장



1. 래스터 그래픽스 영상

래스터 그래픽스 영상의 특성

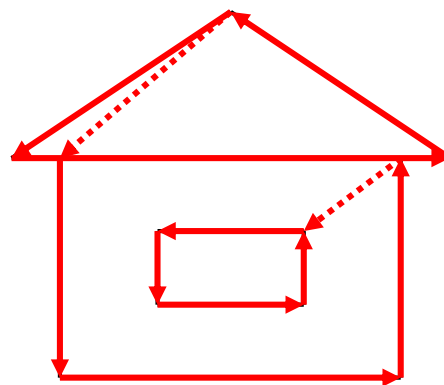
- 색 표현의 정밀도 : 각 픽셀당 색을 표현하는데 사용되는 비트 수에 의해 결정됨
- 영상의 크기 : 영상의 가로 픽셀 수 × 세로 픽셀 수로 표현
 - * 영상을 저장하기 위한 메모리 양은 영상의 크기 및 색 표현의 정밀도에 의해 결정됨
 - * 그림을 확대해도 더 세밀한 그림을 얻을 수는 없음



2. 벡터 그래픽스 영상

벡터 그래픽스 영상이란?

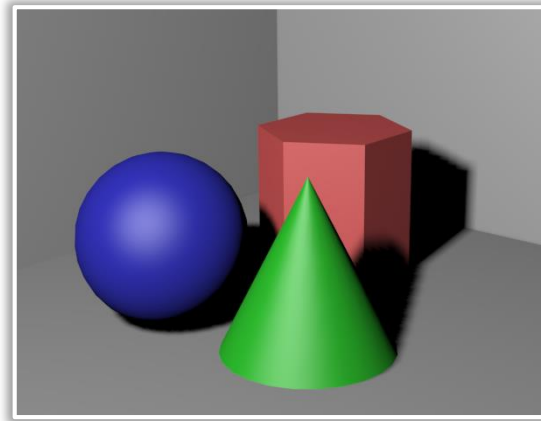
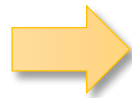
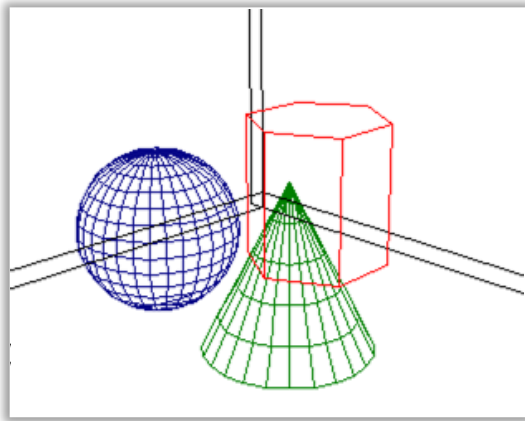
- 벡터(vector) : 크기와 방향을 동시에 나타내는 값
- 벡터 그래픽스 : 수학 방정식을 기반으로 점, 직선, 곡선, 다각형 등을 표현하는 방법
- Adobe Illustrator, CorelDRAW 등의 그래픽스 에디터로 제작
- 3dsMax, Maya 등의 3차원 그래픽스 패키지에서도 벡터 방식으로 물체를 설계함
- SVG, PDF, EPS, WMF, DXF 등의 파일 형식



2. 벡터 그래픽스 영상

벡터 그래픽스 영상의 특성

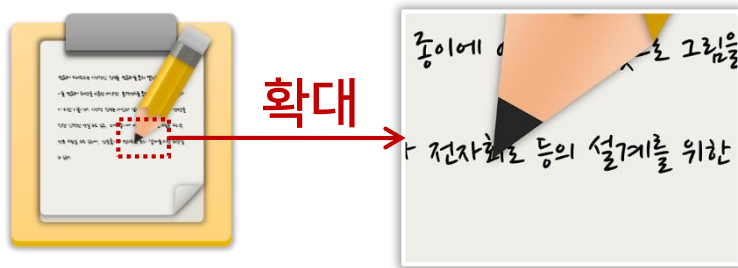
- 영상을 저장하는데 필요한 메모리 양은 영상의 크기가 아니라 영상에 들어 있는 그래픽스 기본요소의 수에 의해 결정됨
- 렌더링을 통해 래스터 영상을 만들어 디스플레이 함



2. 벡터 그래픽스 영상

벡터 그래픽스 영상의 특성

- 영상을 확대해도 화질이 떨어지지 않으며, 자연스럽게 형체를 유지하며 확대됨



- 사진과 같은 영상을 표현하기에는 적절하지 않음



02

그래픽스 소프트웨어의 유형

1. 컴퓨터 그래픽스 패키지
2. 그래픽스 API의 역할
3. 범용 그래픽스 API의 유형



1. 컴퓨터 그래픽스 패키지

특수목적 패키지

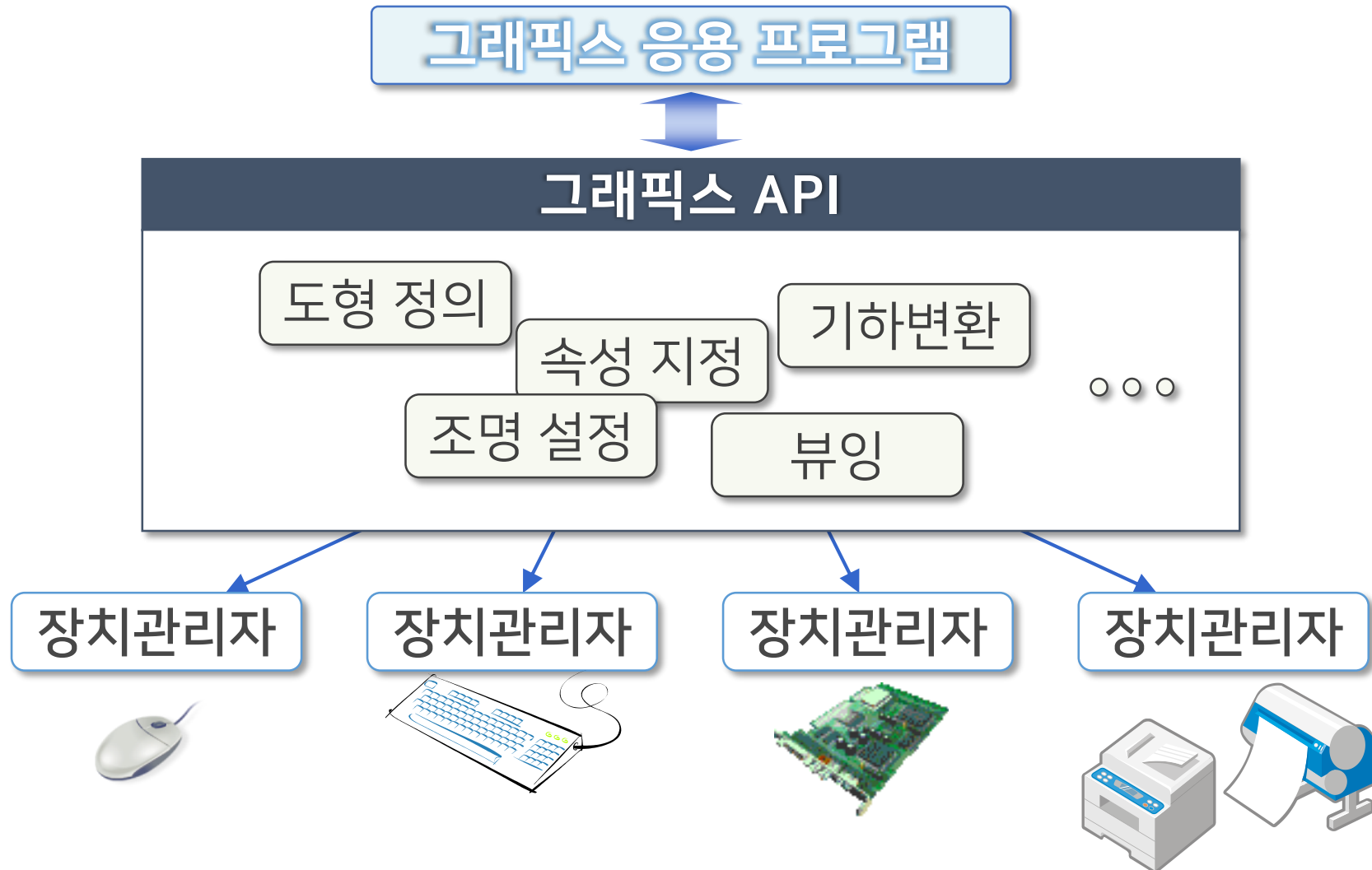
- 응용분야에 맞는 그림을 생성하기 위한 그래픽스 패키지
- CAD 패키지 : 건축, 기계, 전자회로 등의 제품 설계
- 페인트 패키지 : 래스터 그림이나 사진 등을 제작하거나 편집
- 3차원 모델링 및 애니메이션 : 3ds Max, Maya, unity 등

범용 그래픽스 API

- 전문 프로그래머가 C/C++, Java 등의 프로그래밍 언어에서 사용할 수 있는 그래픽스 라이브러리
- 장면 설계 및 렌더링을 위한 최적화된 함수를 제공함



2. 그래픽스 API의 역할



3. 범용 그래픽스 API의 유형

저수준 그래픽스 API

- 그래픽 기본요소 및 그 속성 지정, 기하변환, 뷰잉 등 장면을 정의하여 컴퓨터 화면에 표시하는 일련의 과정을 지시하는 함수들을 제공
- 프로그래머는 이 함수들을 이용하여 장면을 구성하고, 이를 화면에 그리기 위한 세부적인 처리 과정을 하나하나 프로그램으로 작성
- GL, OpenGL, Direct3D 등



3. 범용 그래픽스 API의 유형

고수준 그래픽스 API

- 장면 묘사를 위주로 하는 기능을 제공
- 다양한 모형(큐브, 다각형, 재질, 카메라, 광원 등)을 제공
 - * 기존 모형을 새로운 모형으로 쉽게 변형할 수 있게 함
- 모형들로 구성된 객체들을 계층적으로 조직화하여 장면을 구성함
- 고수준 그래픽스 API의 예
 - * Open Inventor : OpenGL에 기반을 둔 객체지향 3D 그래픽스 API
 - * VRML(Virtual-Reality Modeling Language)
 - * Java 3D



03

OpenGL 프로그래밍

1. OpenGL 개요
2. OpenGL의 라이브러리
3. OpenGL의 자료형
4. OpenGL의 함수 이름 형식
5. 셰이더
6. OpenGL 프로그램 예제



1. OpenGL 개요

OpenGL이란?

- SGI의 그래픽 워크스테이션에 IRIS GL(Integrated Raster Imaging System Graphics Library)이라는 그래픽스 API를 제공
- GL을 다양한 워크스테이션에서 활용할 수 있도록 하드웨어에 독립적인 버전인 OpenGL을 개발
 - * SGI, DEC, IBM, Apple, Microsoft, Nvidia 등과 함께 OpenGL Architecture Review Board(ARB)를 구성하여 OpenGL을 유지·관리
 - * 2006년부터 Khronos Group에서 관리
- 저수준 절차적 API



1. OpenGL 개요

OpenGL의 진화

- 그래픽스 하드웨어 발전에 맞추어 새로운 버전으로 진화

OpenGL 1.0 (1992) ..→ 고정 기능 그래픽스 파이프라인



OpenGL 2.0 (2004) ..→ GLSL(OpenGL Shading Language)



OpenGL 3.0 (2008) ..→ 디프리케이션(deprecation) 모델 도입



OpenGL 4.0 (2010) ..→ 두 단계의 새로운 테셀레이션 셰이더 추가



OpenGL 4.6 (2017)



1. OpenGL 개요

OpenGL의 진화

- 고정 기능 파이프라인에서 프로그램 가능 파이프라인으로 진화

유연성 및 혁신

고정기능(fixed-function)
파이프라인




프로그램 가능(programmable)
파이프라인

- 응용프로그램이 설정할 수 있는 고정된 개수의 파라미터에 의해 제어되는 동작을 하는 처리 단계로 구성되는 그래픽 파이프라인의 버전
- 꼭짓점, 프래그먼트(fragments)의 처리 및 이들과 관련된 데이터(예를 들면 텍스처 좌표)가 셰이더(shader) 프로그램에 의해 제어되는 동작 모드
- GLSL : 셰이더 프로그래밍을 위한 C-스타일의 문장으로 표현되는 언어



2. OpenGL의 라이브러리

OpenGL core profile

- 셰이더(shader) 프로그램의 준비
 - 셰이더가 사용할 데이터의 준비
 - * 2, 3차원 기하학적 구조, 텍스처 이미지 등
 - 여러 가지 OpenGL 상태 설정, 기하학적 구조의 그리기 지시 명령 등
-  OpenGL compatibility profile : 구 버전과의 호환성을 위해 제공되는 기능을 포함



2. OpenGL의 라이브러리

OpenGL의 확장 기능

- OpenGL은 하드웨어 발전에 맞추어 진화하는 API임
 - * 하드웨어 공급업체에서 확장 기능을 제공
 - ..➡ 다수의 공급자의 동의 하에 구현 - EXT
 - ..➡ ARB의 승인 - ARB
 - ..➡ 새로운 버전의 OpenGL에 수용
- OpenGL의 확장 기능을 관리하는 라이브러리 활용
 - * GLEW(OpenGL Extension Wrangler)
 - * GLEE(OpenGL Easy Extension)



2. OpenGL의 라이브러리

윈도 입출력을 위한 라이브러리

- OpenGL은 장치 독립적인 라이브러리

→ 특정 윈도 시스템을 위한 기능은 해당 윈도 시스템을 위한 라이브러리가 필요함

- * Microsoft Windows : WGL 인터페이스 루틴 사용
- * X Window : GLX 루틴 사용
- * 애플 시스템 : AGL 루틴 사용

 윈도 시스템에 의존적임



2. OpenGL의 라이브러리

윈도우 시스템 독립적 윈도우 인터페이스 라이브러리

- GLUT(OpenGL Utility Toolkit) 라이브러리
 - * 함수에는 'glut'라는 접두사를 사용
 - * 현재는 더 이상 유지보수가 제공되지 않음
- freeglut 라이브러리
 - * GLUT를 대신할 수 있는 라이브러리
 - * freeglut32.lib 라이브러리 파일 및 freeglut.h 헤더파일을 사용



3. OpenGL의 자료형

OpenGL 자료형	C/C++ 자료형	접미사
GLbyte	8비트 정수	b
GLshort	16비트 정수	s
GLint, GLsizei	32비트 정수	i
GLfloat, GLclampf	32비트 실수	f
GLdouble, GLclampd	64비트 실수	d
GLubyte, GLboolean	8비트 부호 없는 정수	ub
GLushort	16비트 부호 없는 정수	us
GLuint, GLenum, GLbitfield	32비트 부호 없는 정수	ui



4. OpenGL의 함수 이름 형식

```
return_type  <lib_prefix> FunctionName <arg_count> <arg_type> {v}  
              (<arguments>);
```

- <lib_prefix> : 함수 접두사
- <arg_count> : 함수가 갖는 인수의 수
- <arg_type> : 인수의 데이터형
- {v} : 인수가 벡터인 경우 첨가

예

```
void glClear(GLbitfield buf);
```



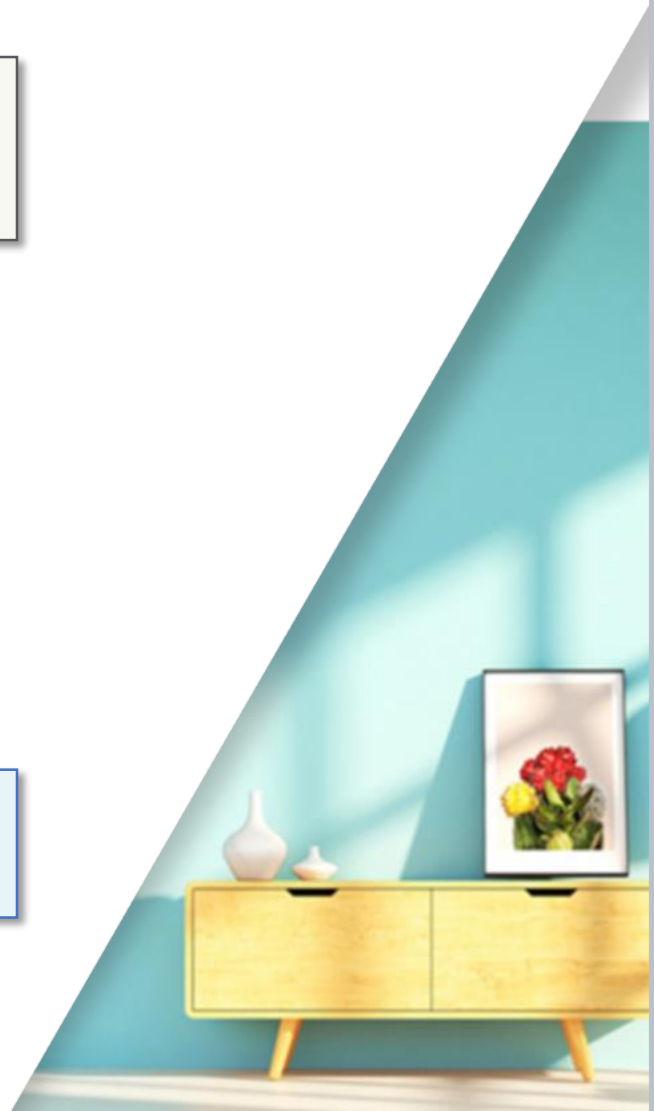
4. OpenGL의 함수 이름 형식

```
return_type <lib_prefix> FunctionName<arg_count> <arg_type> {v}  
(<arguments>);
```

- <lib_prefix> : 함수 접두사
- <arg_count> : 함수가 갖는 인수의 수
- <arg_type> : 인수의 데이터형
- {v} : 인수가 벡터인 경우 첨가

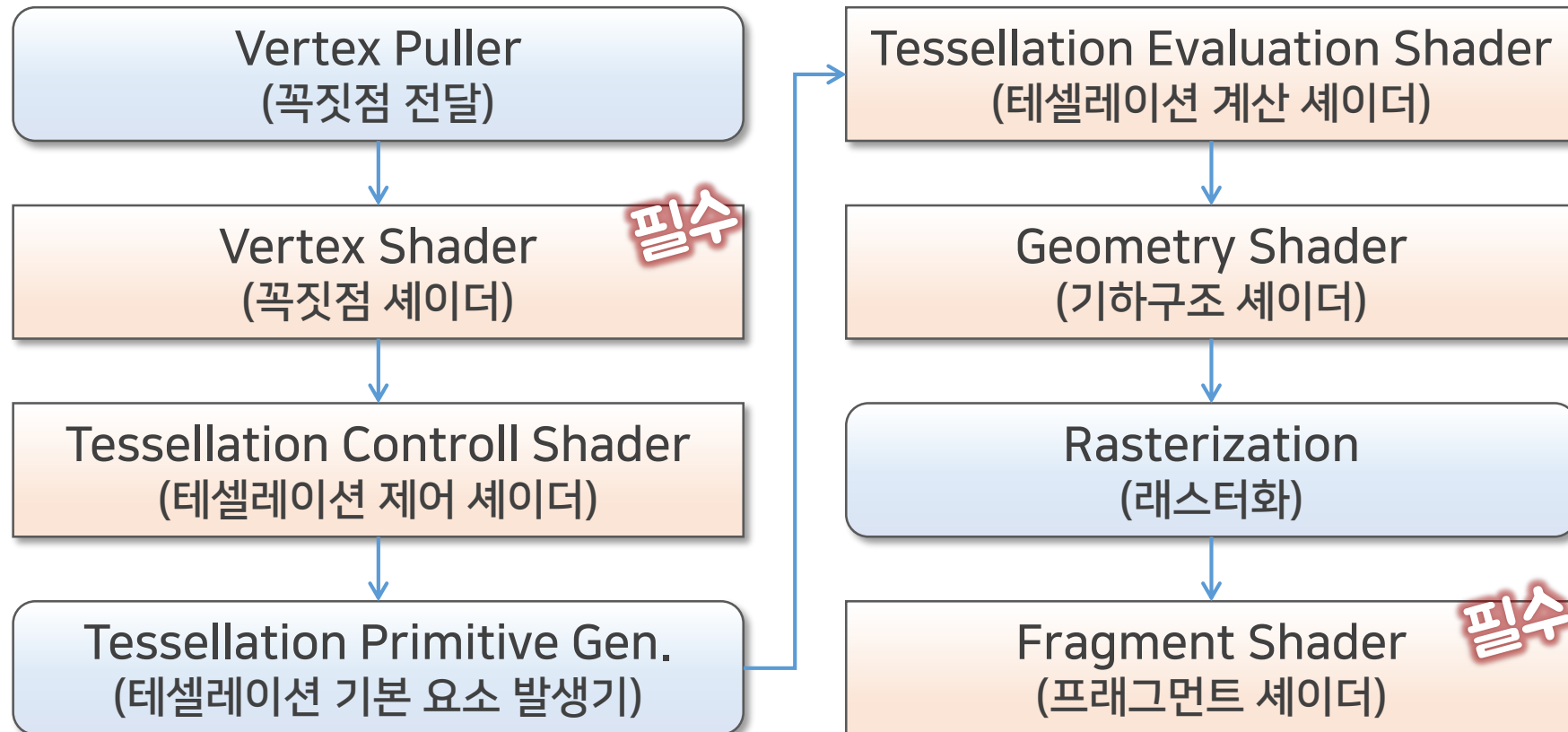
예

```
void glUniform2f(GLint location, GLfloat v0, GLfloat v1);
```



5. 셰이더

OpenGL 파이프라인



5. 셰이더

GLSL(OpenGL Shading Language)

- OpenGL ARB가 셰이더 프로그램 작성용 언어로 만든 C 언어 형태의 프로그래밍 언어
- OpenGL 그래픽스 파이프라인 중 프로그램 가능 단계를 수행하는 셰이더 프로그램을 작성함
 - * 꼭짓점 셰이더, 테셀레이션 제어 셰이더, 테셀레이션 계산 셰이더, 기하구조 셰이더, 프래그먼트 셰이더



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

전체적인 프로그램 구성

선언부 (1~16행)

셰이더 프로그램 (18~34행)

AddShader 함수 (36~58행)

SetUpShaders 함수 (60~91행)

RenderCB 함수 (93~104행)

InitVBOs 함수 (106~116행)

main 함수 (118~142행)

- ..➡ 헤더파일 삽입, 자료형, 전역변수 선언 등
- ..➡ 꼭짓점 셰이더, 프래그먼트 셰이더
- ..➡ 개별 셰이더 추가
- ..➡ 전체 셰이더 준비
- ..➡ 화면 렌더링 콜백 함수
- ..➡ 꼭짓점을 저장하는 버퍼 생성
- ..➡ OpenGL 동작 환경 준비(윈도 생성, 콜백함수 설정 등) 및 실행



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

선언부

```
1  #include <iostream>
2  using namespace std;
3
4  #define FREEGLUT_STATIC
5  #define GLEW_STATIC
6  #include <gl/glew.h>
7  #include <gl/freeglut.h>
8
9  struct Vec3f { // 3차원 좌표를 표현하기 위한 구조체
10     float x, y, z;
11     Vec3f() { }
12     .....
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

선언부

```
5  .....
6  #include <gl/glew.h>
7  #include <gl/freeglut.h>
8
9  struct Vec3f { // 3차원 좌표를 표현하기 위한 구조체
10     float x, y, z;
11     Vec3f() { }
12     Vec3f(float _x, float _y, float _z) : x(_x), y(_y), z(_z) { }
13 };
14
15 enum {TRIANGLE, N_VBOs};
16 GLuint VBO[N_VBOs]; // 꼭짓점 버퍼 객체
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

꼭짓점을 저장하는 버퍼 생성(InitVBOs 함수)

```
106 static void InitVBOs()
107 {
108     Vec3f Vertices[3];           // 삼각형의 꼭짓점 좌표
109     Vertices[0] = Vec3f(-5.0f, -5.0f, 0.0f);
110     Vertices[1] = Vec3f(5.0f, -5.0f, 0.0f);
111     Vertices[2] = Vec3f(0.0f, 5.0f, 0.0f);
112     // 꼭짓점 버퍼를 생성하여 삼각형의 꼭짓점 좌표 전달
113     glGenBuffers(N_VBOs, VBO);
114     glBindBuffer(GL_ARRAY_BUFFER, VBO[TRIANGLE]);
115     glBufferData(GL_ARRAY_BUFFER, sizeof(Vertices),
116                  Vertices, GL_STATIC_DRAW);
116 }
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

화면 렌더링 콜백 함수(RenderCB 함수)

```
93 static void RenderCB()
94 {
95     glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
96     glClear(GL_COLOR_BUFFER_BIT);    // 백색으로 화면 지움
97
98     glEnableVertexAttribArray(0);
99     glBindBuffer(GL_ARRAY_BUFFER, VBO[TRIANGLE]);
100    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
101    glDrawArrays(GL_TRIANGLES, 0, 3);
102    glDisableVertexAttribArray(0);
103    glFinish();
104 }
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

셰이더 프로그램 - 꼭짓점 셰이더

```
18 static const char* pVS =           // 꼭짓점 셰이더 소스
19 "#version 330                       \n"
20 "layout (location = 0) in vec3 Position; \n"
21 "                                       \n"
22 "void main()                         \n"
23 "{                                   \n"
24 "    gl_Position = vec4(Position*0.1, 1.0); \n"
25 "};"
26
27 static const char* pFS =           // 프래그먼트 셰이더 소스
28 "#version 330                       \n"
29 "....."
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

셰이더 프로그램 - 꼭짓점 셰이더

```
23 .....
24 "   gl_Position = vec4(Position*0.1, 1.0);           \n"
25 "};"
26
27 static const char* pFS =           // 프래그먼트 셰이더 소스
28 "#version 330                               \n"
29 "out vec4 FragColor;                       \n"
30 "                                           \n"
31 "void main()                               \n"
32 "{"                                           \n"
33 "   FragColor = vec4(1.0, 0.0, 0.0, 1.0);      \n"
34 "};"
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

전체 셰이더 준비(SetupShaders 함수)

```
60 static void SetupShaders()
61 { // 셰이더 프로그램 객체 생성
62     GLuint shaderProg = glCreateProgram();
63     if (!shaderProg) {
64         cerr << "오류 - Shader 프로그램 생성" << endl;
65         exit(1);
66     }
67
68     // 꼭짓점 셰이더 및 프래그먼트 셰이더 적재
69     AddShader(shaderProg, pVS, GL_VERTEX_SHADER);
70     AddShader(shaderProg, pFS, GL_FRAGMENT_SHADER);
71     .....
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

전체 셰이더 준비(SetupShaders 함수)

```
60 static void SetupShaders()
... { .....
72     GLint success = 0;
73     GLchar errLog[256];
74
75     glLinkProgram(shaderProg); // 셰이더 프로그램 링크
76     glGetProgramiv(shaderProg, GL_LINK_STATUS, &success);
77     if (!success) {
78         glGetProgramInfoLog(shaderProg, sizeof(errLog), NULL, errLog);
79         cerr << "오류 - Shader 프로그램 링크: " << errLog << endl;
80         exit(1);
81     }
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

전체 셰이더 준비(SetupShaders 함수)

```
60 static void SetupShaders()
... { .....
83 glValidateProgram(shaderProg); // 프로그램 객체가 유효한지 검사
84 glGetProgramiv(shaderProg, GL_VALIDATE_STATUS, &success);
85 if (!success) {
86     glGetProgramInfoLog(shaderProg, sizeof(errLog), NULL, errLog);
87     cerr << "Invalid shader program: " << errLog << endl;
88     exit(1);
89 }
90 glUseProgram(shaderProg); // 현재 셰이더 프로그램 객체로 지정
91 }
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

개별 셰이더 추가(AddShader 함수)

```
36 static void AddShader(GLuint shaderProg,  
    const char* pShaderSrc, GLint ShaderType)  
37 { // 셰이더 생성  
38     GLuint shader = glCreateShader(ShaderType);  
39     if (!shader) {  
40         cerr << "오류 - Shader 생성(" << ShaderType << ")" << endl;  
41         exit(0);  
42     }  
43     // 셰이더 컴파일  
44     const GLchar* src[1] = { pShaderSrc };  
45     const GLint len[1] = { strlen(pShaderSrc) };  
46     .....
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

개별 셰이더 추가(AddShader 함수)

```
36 static void AddShader(GLuint shaderProg,  
    const char* pShaderSrc, GLint ShaderType)  
... { .....  
43     // 셰이더 컴파일  
44     const GLchar* src[1] = { pShaderSrc };  
45     const GLint len[1] = { strlen(pShaderSrc) };  
46     glShaderSource(shader, 1, src, len);  
47     glCompileShader(shader);  
48     GLint success;  
49     glGetShaderiv(shader, GL_COMPILE_STATUS, &success);  
50     if (!success) {           // 컴파일 오류 발생  
51         .....  
}
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

개별 셰이더 추가(AddShader 함수)

```
36 static void AddShader(.....)
... { .....
50     if (!success) {          // 컴파일 오류 발생
51         GLchar infoLog[256];
52         glGetShaderInfoLog(shader, 256, NULL, infoLog);
53         cerr << "오류 - Shader 컴파일(" << ShaderType << "): "
               << infoLog << endl;
54         exit(1);
55     }
56     // 셰이더 프로그램에 컴파일된 셰이더를 추가
57     glAttachShader(shaderProg, shader);
58 }
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

main 함수

```
118 int main(int argc, char** argv)
119 {
120     glutInit(&argc, argv);
121     glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
122     glutInitWindowPosition(50, 100);
123     glutInitWindowSize(640, 480);
124     glutCreateWindow("OpenGL Sample");
125
126     GLenum s = glewInit();
127     if (s != GLEW_OK) {
128         cerr << "오류 - " << glewGetErrorString(s) << endl;
129         .....
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

main 함수

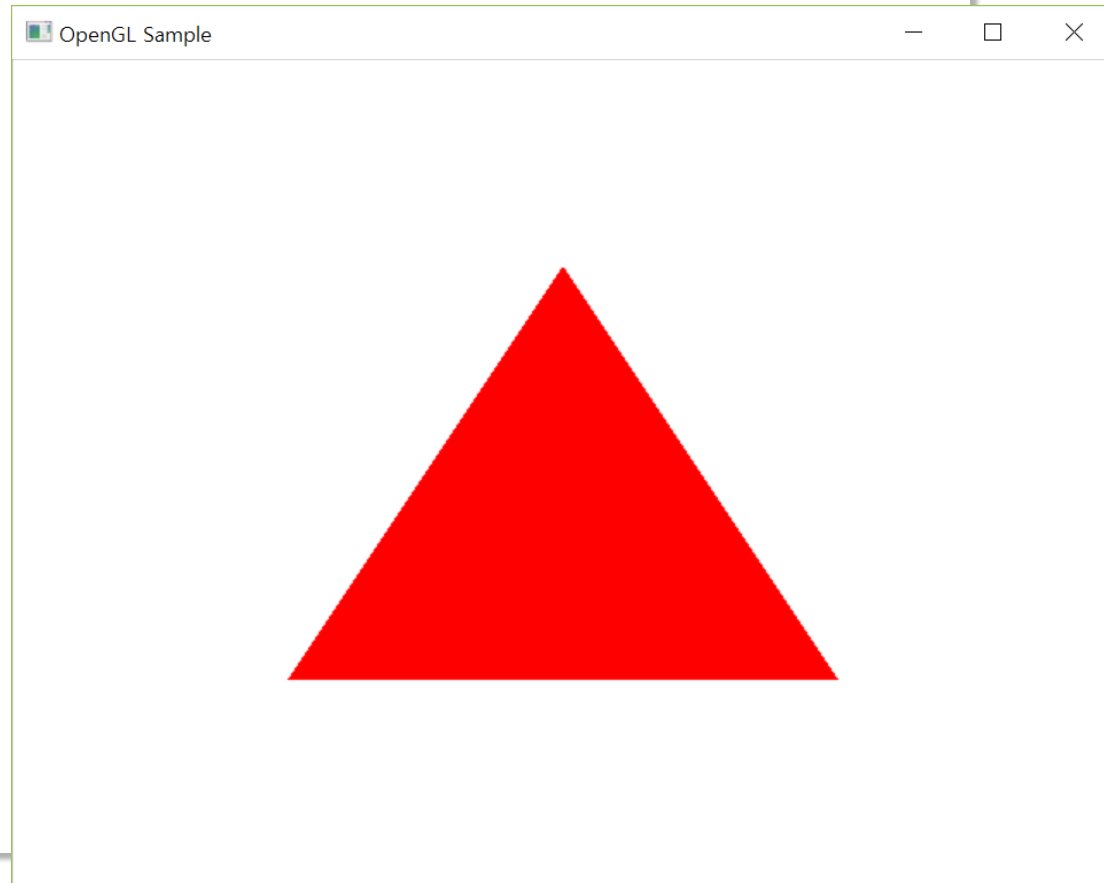
```
118 int main(int argc, char** argv)
... { .....
126     GLenum s = glewInit();
127     if (s != GLEW_OK) {
128         cerr << "오류 - " << glewGetErrorString(s) << endl;
129         return 1;
130     }
131
132     cout << "GL version: " << glGetString(GL_VERSION) << endl;
133     cout << "GLSL version: "
        << glGetString(GL_SHADING_LANGUAGE_VERSION) << endl;
134     .....
```



6. OpenGL 프로그램 예제 : OpenGLSample.cpp

main 함수

```
118 int main(int argc, char** argv)
119 {
    ...
135     SetUpShaders();
136     InitVBOs();
137
138     glutDisplayFunc(RenderCB);
139     glutMainLoop();
140
141     return 0;
142 }
```



컴퓨터그래픽스
다음시간

제3강 컴퓨터 그래픽스 기본요소(1)

