

심화전공실습

HW #06



Self-scoring table

	P01	P02	P03	E01	E02	Total
Score	1	1	1	1	1	5

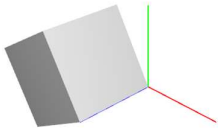
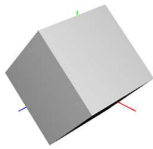
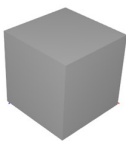
과목명	심화전공실습
학부	소프트웨어학부
학번	2019203010
이름	김민철
제출일자	2020년 10월 13일

I. Practice

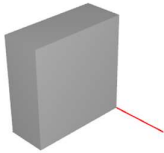
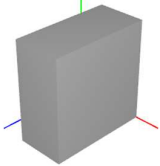
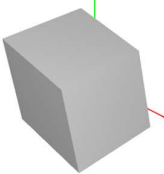
```
#define GLM_ENABLE_EXPERIMENTAL
```

주어진 코드에서 GLM_GTX_dual_quaternion 과 GLM_GTX_string_cast 라는 GLM 라이브러리 모듈에서 오류가 생겨 다음과 같은 코드를 추가했다.

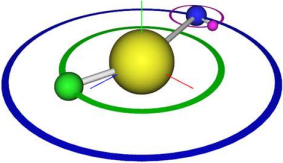
Practice 01. Three rotation examples

rotationExample()	rotationPivotExample()	RotationPivotExampleGLM()
		

Practice 02. Three scaling examples

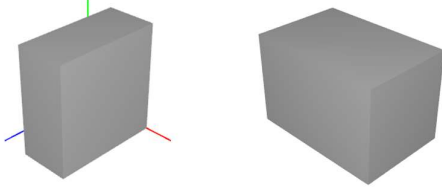
scalingExample()	scalingPivotExample()	scalingDirectionExample()
		

Practice 03. A solar system example

solarSystemExample()


II. Exercise

Exercise 01. Scaling wrt a pivot with GLM

scalingPivotExampleGLM()	
	
코드	
<pre>void scalingPivotExampleGLM() { float s = 1.0f + 0.95f * sin(frame * 0.1f / period); glm::vec3 pivot(0.5, 0.5, 0.5); //identity matrix glm::mat4 M(1.0); // M = M* translate(pivot) M = glm::translate(M, pivot); //M = M* scale(direction) M = glm::scale(M, glm::vec3(s, 1, 1)); // M = M* translate(-pivot) M = glm::translate(M, -pivot); glmMultMatrixf(value_ptr(M)); setDiffuseColor(glm::vec3(1, 1, 1)); drawCube(); }</pre>	

이 함수는 GLM 라이브러리를 사용하여 3D 공간에서 객체를 축소 또는 확대하는 기능을 구현한다. 특히, 객체를 특정한 축을 중심으로 확대 또는 축소하는 기능을 가진다.

1. 변수 선언 :

s: 스케일링 비율을 저장하는 변수이다. sin 함수를 사용하여 시간에 따라 변화하는 값을 계산한다.

pivot: 객체의 축소 또는 확대가 이루어지는 축을 나타내는 벡터이다.

2. 항등 행렬 생성 :

glm::mat4 M(1.0): 4x4 항등 행렬을 생성한다. 이 행렬은 초기 상태에서 객체에 아무런 변환을 적용하지 않는다.

3. Translation :

M = glm::translate(M, pivot): 객체를 pivot 벡터만큼 이동시킨다. 이는 축소 또는 확대가 이루어지는 중심점을 설정한다.

4. Scaling :

M = glm::scale(M, glm::vec3(s, 1, 1)): 객체를 x 축 방향으로 s 배만큼, y 축과 z 축

방향으로 1 배만큼 확대 또는 축소한다.

5. 역변환 :

`M = glm::translate(M, -pivot)`: 객체를 다시 원래 위치로 되돌린다. 이는 축소 또는 확대가 원래 위치를 기준으로 이루어지도록 한다.

6. 행렬 적용 :

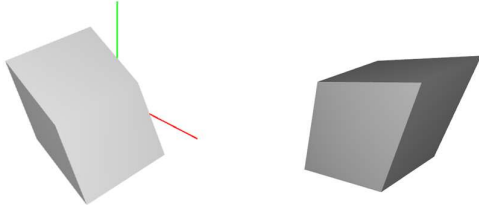
`glm::mat4x4 value_ptr(M)`: 계산된 변환 행렬 `M` 을 OpenGL 의 현재 행렬에 곱합니다. 이를 통해 객체에 변환을 적용한다.

7. 색상 설정 및 큐브 그리기 :

`setDiffuseColor(glm::vec3(1, 1, 1))`: 객체의 확산색을 설정한다.

`drawCube()`: 큐브를 그린다. 이 함수는 이전에 정의된 큐브 그리기 함수를 호출하여 변환된 객체를 화면에 출력한다.

Exercise 02. Scaling wrt a direction with GLM

scalingDirectionExampleGLM()	
	
코드	
<pre>void scalingDirectionExampleGLM() { float s = 1.0f + 0.95f * sin(frame * 0.1f / period); glm::vec3 direction(1, 1, 0); direction = normalize(direction); //unit vector //compute the rotation axis and angle between the x-axis and the given direction glm::vec3 axis = cross(glm::vec3(1, 0, 0), direction); float sinTheta = length(axis); float cosTheta = dot(glm::vec3(1, 0, 0), direction); float theta = (float)atan2(sinTheta, cosTheta) * float(180.0 / M_PI); axis /= sinTheta; //unit axis //identity matrix glm::mat4 M(1.0); // M = M* rotate(angle, axis) M = glm::rotate(M, glm::radians(theta), axis); //M = M* scale(direction) M = glm::scale(M, glm::vec3(s,1,1)); // M = M* rotate(angle, axis) M = glm::rotate(M, glm::radians(-theta), axis); glmMultMatrixf(value_ptr(M)); setDiffuseColor(glm::vec3(1, 1, 1)); drawCube(); }</pre>	

이 함수는 GLM 라이브러리를 사용하여 3D 공간에서 객체를 특정한 방향으로 확대 또는 축소하는 기능을 구현한다.

1. 변수 선언 :

s: 스케일링 비율을 저장하는 변수이다. sin 함수를 사용하여 시간에 따라 변화하는 값을 계산한다.

direction: 확대 또는 축소가 이루어질 방향을 나타내는 벡터이다.

2. 방향 벡터 정규화 :

direction = normalize(direction): direction 벡터를 단위 벡터로 정규화한다. 단위 벡터는 크기가 1 인 벡터로, 방향만을 나타낸다.

3. 회전 축 및 각도 계산 :

glm::vec3 axis = cross(glm::vec3(1, 0, 0), direction): x 축과 direction 벡터의 외적을 계산하여 회전 축을 구한다.

float sinTheta = length(axis): 회전 축의 길이를 계산하여 sin 값을 구한다.

float cosTheta = dot(glm::vec3(1, 0, 0), direction): x 축과 direction 벡터의 내적을 계산하여 코사인 값을 구한다.

float theta = (float)atan2(sinTheta, cosTheta) * float(180.0 / M_PI): sin 값과 cos 값을 이용하여 회전 각도를 계산한다.

axis /= sinTheta: 회전 축을 단위 벡터로 정규화한다.

4. 항등 행렬 생성 :

glm::mat4 M(1.0): 4x4 항등 행렬을 생성한다.

5. Rotation :

M = glm::rotate(M, glm::radians(theta), axis): 객체를 axis 축을 중심으로 theta 라디안만큼 회전시킨다. 이는 확대 또는 축소가 direction 방향으로 이루어지도록 한다.

6. Scaling :

M = glm::scale(M, glm::vec3(s, 1, 1)): 객체를 x 축 방향으로 s 배만큼, y 축과 z 축 방향으로 1 배만큼 확대 또는 축소한다.

7. 역변환 :

M = glm::rotate(M, glm::radians(-theta), axis): 객체를 원래 방향으로 되돌리기 위해 역회전을 수행한다.

8. 행렬 적용 :

glMultMatrixf(value_ptr(M)): 계산된 변환 행렬 M 을 OpenGL 의 현재 행렬에 곱한다. 이를 통해 객체에 변환을 적용한다.

9. 색상 설정 및 큐브 그리기 :

setDiffuseColor(glm::vec3(1, 1, 1)): 객체의 확산색을 설정한다.

`drawCube()`: 큐브를 그린다. 이 함수는 이전에 정의된 큐브 그리기 함수를 호출하여 변환된 객체를 화면에 출력한다.

III. 느낀점

Practice에서 진행했던 `scalingPivotExample()`과 `scalingDirectionExample()`을 GLM 라이브러리를 이용하면서 직접 구현해보면서 OpenGL 기본 함수와 GLM 라이브러리의 차이에 대해 학습해 볼 수 있었다. GLM을 이용해서 좀 더 편리하게 벡터, 행렬 계산을 시도해볼 수 있었다. 하지만 프로그래밍을 해보면서 스스로 관련 수학 지식들에 대한 학습이 필요하다고 느꼈다. 관련 수학적 개념에 대한 학습을 더 진행하여 그래픽 학습이 좀 더 수월하도록 할 것이다.