심화전공실습

HW #04

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 09월 24일

I. Practice

Practice 01. part1, part2

```
PS D:\4학년\2학기\심전실\과제제출\2019203010_HW_04\EXE> ./P04.exe 1
3x1 vectors
a = vec3(0.000000, 0.000000, 0.000000)
b = vec3(3.000000, 2.000000, 1.000000)
a = vec3(1.000000, 2.000000, 3.000000)
a[0] = 1
a.z = 3
PS D:\4학년\2학기\심전실\과제제출\2019203010_HW_04\EXE> ./P04.exe 2
Vector operations
a = vec3(1.000000, 2.000000, 3.000000)
b = vec3(3.000000, 2.000000, 1.000000)
a + b = vec3(4.000000, 4.000000, 4.000000)
a - b = vec3(-2.000000, 0.000000, 2.000000)
-a = vec3(-1.000000, -2.000000, -3.000000)
1.5*a = vec3(1.500000, 3.000000, 4.500000)
dot(a, b) = 10
cross(a, b) = vec3(-4.000000, 8.000000, -4.000000)
length(a) = 3.74166
```

Practice 02. part3, part4

```
PS D:\4학년\2학기\심전실\과제제출\2019203010_HW_04\EXE> ./P04.exe 3
3x3 matrices
A = mat3x3((0.000000, 0.000000, 0.000000), (0.000000, 0.000000, 0.000000), (0.000000, 0.000000, 0.000000))
A = mat3x3((1.000000, 0.000000, 0.000000), (0.000000, 1.000000, 0.000000), (0.000000, 0.000000, 1.000000))
B = mat3x3((1.000000, 0.000000, 0.000000), (2.000000, 1.000000, 0.000000), (3.000000, 0.000000, 1.000000))
B = mat3x3((1.000000, 0.000000, 0.000000), (2.000000, 1.000000, 0.000000), (3.000000, 0.000000, 1.000000))
3rd col of B = vec3(3.000000, 0.000000, 1.000000)
3rd col B = vec3(3.000000, 0.000000, 1.000000)
3rd row of B = vec3(0.000000, 0.000000, 1.000000)
1st row 3rd col of B = 3
1st row 3rd col of B = 3
PS D:\4학년\2학기\심전실\과제제출\2019203010_HW_04\EXE> ./P04.exe 4
Matrix operations
A = mat3x3((1.000000, 0.000000, 0.000000), (0.000000, 1.000000, 0.000000), (0.000000, 0.000000, 1.000000))
B = mat3x3((1.000000, 0.000000, 0.000000), (2.000000, 1.000000, 0.000000), (3.000000, 0.000000, 1.000000))
A + B = mat3x3((2.000000, 0.000000, 0.000000), (2.000000, 2.000000, 0.000000), (3.000000, 0.000000, 2.000000))
A - B = mat3x3((0.000000, 0.000000, 0.000000), (-2.000000, 0.000000, 0.000000), (-3.000000, 0.000000, 0.000000))
-A = mat3x3((-1.000000, 0.000000, 0.000000), (0.000000, -1.000000, 0.000000), (0.000000, 0.000000, -1.000000))
1.5*A = mat3x3((1.500000, 0.000000, 0.000000), (0.000000, 1.500000, 0.000000), (0.000000, 0.000000, 1.500000))
A x B = mat3x3((1.000000, 0.000000, 0.000000), (2.000000, 1.000000, 0.000000), (3.000000, 0.000000, 1.000000))
transpose(B) = mat3x3((1.000000, 2.000000, 3.000000), (0.000000, 1.000000, 0.000000), (0.000000, 0.000000, 1.000000))
inverse(B) = mat3x3((1.000000, -0.000000, 0.000000), (-2.000000, 1.000000, -0.000000), (-3.000000, -0.000000, 1.000000))

inverse(B) * B = mat3x3((1.000000, 0.000000, 0.000000), (0.000000, 1.000000, 0.000000), (0.000000, 0.000000, 1.000000))
```

Practice 03. part5

```
PS D:\4학년\2학기\심전실\과제제출\2019203010_HW_04\EXE> ./P04.exe 5
Matrix-vector multiplication and assembling
a = vec3(1.000000, 2.000000, 3.000000)
B = mat3x3((1.000000, 0.000000, 0.000000), (2.000000, 1.000000, 0.000000), (3.000000, 0.000000, 1.000000))
B x a = vec3(14.000000, 2.000000, 3.000000)
a x B = vec3(1.000000, 4.000000, 6.000000)
(a, 1.0) = vec4(1.000000, 2.000000, 3.000000, 1.000000)
(1.0, a) = vec4(1.000000, 1.000000, 2.000000, 3.000000)
C = mat4x4((1.000000, 0.000000, 0.000000, 0.000000), (2.000000, 1.000000, 0.000000, 0.000000), (3.000000, 0.000000, 1.000000, 0.000000), (0.000000, 0.000000, 0.000000, 1.000000))
```

II. Exercise

Exercise 01. part1

```
PS D:\4학년\2학기\심전실\과제제출\2019203010_HW_04\EXE> ./E04.exe 1
Vector operations
 v1 = vec3(1.000000, 2.000000, 3.000000)
 v2 = vec3(3.000000, 2.000000, 1.000000)
 v1 + v2 = vec3(4.000000, 4.000000, 4.000000)
 v1 - v2 = vec3(-2.000000, 0.000000, 2.000000)
 -v1 = vec3(-1.000000, -2.000000, -3.000000)
 v1 - 2.0*v2 = vec3(-5.000000, -2.000000, 1.000000)
 dot(v1, v2) = 10
 cross(v1, v2) = vec3(-4.000000, 8.000000, -4.000000)
```

Exercise 02. part2

```
PS D:\4학년\2학기\심전실\과제제출\2019203010_HW_04\EXE> ./E04.exe 2
Matrix operations
 v1 = vec3(1.000000, 2.000000, 3.000000)
 v2 = vec3(3.000000, 2.000000, 1.000000)
 A1 = mat3x3((1.000000, 2.000000, 1.000000), (2.000000, 3.000000, 1.000000), (3.000000, 2.000000, 2.000000))
 A2 = mat3x3((2.000000, 2.000000, 1.000000), (1.000000, 2.000000, 1.000000), (2.000000, 1.000000, 1.000000))
 A1 + A2 = mat3x3((3.000000, 4.000000, 2.000000), (3.000000, 5.000000, 2.000000), (5.000000, 3.000000, 3.000000))
 A1 - A2 = mat3x3((-1.000000, 0.000000, 0.000000), (1.000000, 1.000000, 0.000000), (1.000000, 1.000000, 1.000000))
 -A1 = mat3x3((-1.000000, -2.000000, -1.000000), (-2.000000, -3.000000, -1.000000), (-3.000000, -2.000000, -2.000000))
 A1 - 2.0*A2 = mat3x3((-3.000000, -2.000000, -1.000000), (0.000000, -1.000000, -1.000000), (-1.000000, 0.000000, 0.000000))
 A1 x A2 = mat3x3((9.000000, 12.000000, 6.000000), (8.000000, 10.000000, 5.000000), (7.000000, 9.000000, 5.000000))
 A2 x A1 = mat3x3((6.000000, 7.000000, 4.000000), (9.000000, 11.000000, 6.000000), (12.000000, 12.000000, 7.000000))
 A1 x v1 = vec3(14.000000, 14.000000, 9.000000)
 A2 x v2 = vec3(10.000000, 11.000000, 6.000000)
```

Exercise 코드

```
void part1() {
    cout << "Vector operations" << endl;

    // vectors
    glm::vec3       v1(1, 2, 3);
    glm::vec3       v2(3, 2, 1);
    cout << " v1 = " << to_string(v1) << endl;
    cout << " v2 = " << to_string(v2) << endl;

    //addition, subtraction
    cout << " v1 + v2 = " << to_string(v1 + v2) << endl;
    cout << " v1 - v2 = " << to_string(v1 - v2) << endl;
    // negation
    cout << " -v1 = " << to_string(-v1) << endl;
    // scalar multiplicatio and subtraction
    cout << " v1 - 2.0*v2 = " << to_string(v1 - 2.0f * v2) << endl;
    // dot product
    cout << " dot(v1, v2) = " << dot(v1, v2) << endl;
    // cross product
    cout << " cross(v1, v2) = " << to_string(cross(v1, v2)) << endl;
}

void part2() {
    cout << "Matrix operations" << endl;

    // vectors
    glm::vec3       v1(1, 2, 3);
    glm::vec3       v2(3, 2, 1);
    cout << " v1 = " << to_string(v1) << endl;
    cout << " v2 = " << to_string(v2) << endl;
    // matrices
    glm::mat3   A1(1.0, 2.0, 1.0, 2.0, 3.0, 1.0, 3.0, 2.0, 2.0);
    glm::mat3   A2(2.0, 2.0, 1.0, 1.0, 2.0, 1.0, 2.0, 1.0, 1.0);
    cout << " A1 = " << to_string(A1) << endl;
    cout << " A2 = " << to_string(A2) << endl;

    // additon, subtraction
    cout << " A1 + A2 = " << to_string(A1 + A2) << endl;
    cout << " A1 - A2 = " << to_string(A1 - A2) << endl;
    // negation
    cout << " -A1 = " << to_string(-A1) << endl;
    // scalar multiplication, subtraction
    cout << " A1 - 2.0*A2 = " << to_string(A1 - 2.0f * A2) << endl;
    // matrix multiplication
    cout << " A1 x A2 = " << to_string(A1 * A2) << endl;
    cout << " A2 x A1 = " << to_string(A2 * A1) << endl;
    //matrix-vector multiplication
    cout << " A1 x v1 = " << to_string(A1 * v1) << endl;
    cout << " A2 x v2 = " << to_string(A2 * v2) << endl;

}
```

part1 - 벡터 연산

glm 을 이용하여 벡터 v1 과 v2 를 설정한 후 두 벡터를 출력해 초기화가 제대로 되었는지 확인해본다. 그 후 주어진 벡터 더하기, 벡터 빼기, 벡터 스칼라곱, 벡터 내적, 벡터 외적과 같은 여러가지 벡터 연산들을 수행하고 to_string()을 사용하여 그 값을 출력한다.

part2- 행렬 연산

연산에서 활용될 행렬 A1, A2 와 벡터 v1, v2 를 초기값으로 설정한 후 제대로 초기화 되었는지 두 행렬과 두 벡터들을 출력해 확인한다. 이후, 앞선 part1 과 동일하게 행렬 더하기, 행렬 빼기, 행렬 스칼라곱, 행렬곱, 벡터와 행렬의 연산과 같은 여러 행렬의 연산들을 수행한다. 이 값들 또한 to_string()을 통해 출력하고 그 값이 제대로 계산되었는 지를 검토한다.