# 기초 컴퓨터 그래픽스 Assigment 1

## 1. 환경 명세

### 작업 환경

Windows 11 64bit, i7-1185G7 3GHz, Intel Iris Xe Graphics(내장그래픽), Visual Studio 2022 Debug x64

### 테스트 환경

다양한 기기에서 테스트를 수행하고자 Mackbook Pro(2021, M1 Pro)의 Parallels 18 Virtual Machine으로 Windows 11 가상환경에서 실행했지만, OpenGL context version을 4.0에서 3.3으로 낮춰야 하는 문제가 있어서 기존 방식대로 version을 4.0을 유지하고 가상화 없이 Windows 11에서 실행하는 전제로 작업했습니다.

## 2. 요구사항

과제의 명세서에서 요구하는 모든 사항을 구현했습니다.

### (a) 윈도우 화면

#### 작동 확인 방법

프로그램을 실행하면 가로와 세로가 각각 **750px**인 화면이 뜹니다.

#### 구현 방법

main 함수 실행 시 glutInitWindowSize 함수에 의해 parameter로 넘겨진 750px의 가로와 세로를 지니는 윈도우가 화면에 그려집니다.

### (b) 선분 그리기

#### 작동 확인 방법

프로그램을 실행하면 OpenGL의 좌표계를 기준으로 위치의 **파란색 꼭짓점**과 위치의 **흰색 꼭짓점**을 연결하는 **빨간색 선분**이 그려집니다.

#### 구현 방법

선분을 구성하는 파란색과 흰색 꼭짓점의 위치는 initialize\_renderer 함수에서 px, py, qx, qy의 변수에 값을 할당하여 결정됩니다. 이후 윈도우 화면을 그릴 때 register\_callbacks에서 glutDisplayFunc callback 함수로 지정한 display 함수가 실행되는데, 함수의 내용을 순차적으로 실행할 때 draw\_line 함수가 호출되어 빨간색 선분의 파란색과 흰색 꼭짓점을 앞서 초기화한 px, py, qx, qy 변수를 사용하여 그립니다.

### (c) 비대칭 다각형 그리기

#### 작동 확인 방법

프로그램 실행 시 OpenGL 좌표계를 기준으로 , , , , , 위치의 6개의 흰색 꼭짓점, 파란색의 직사각형의 무게 중심점, 그리고 녹색 변을 가지는 **비대칭 다각형**을 그립니다.

#### 구현 방법

흰색 꼭짓점의 위치와 비대칭 다각형의 검정색 무게 중심점은 initialize\_renderer 함수에서 배열 object의 원소에 값을 할당하여 결정됩니다. 이후 윈도우 화면을 그릴 때 register\_callbacks에서 glutDisplayFunc callback 함수로 지정한 display 함수가 실행되는데, 함수의 내용을 순차적으로 실행할 때 draw\_object 함수가 호출되어 다각형의 구성 요소들을 앞서 초기화한 배열 object를 사용하여 그립니다.

### (d) 선분 회전하기

**마우스 휠 (위, 아래) 스크롤**

##### 작동 확인 방법

**마우스 휠을 위쪽으로 스크롤**하면 선분이 파란색 꼭짓점을 중심으로 반시계방향으로 회전합니다. 한편 **아래쪽으로 스크롤**하면 시계방향으로 회전합니다.

##### 구현 방법

register\_callbacks에서 glutMouseFunc callback 함수에 파라미터로 넣은 mousepress 함수는 마우스 휠을 위 또는 아래로 스크롤하면 호출됩니다. 이 함수가 호출될 때 파라미터로 넘어오는 button는 위 방향 스크롤이 감지되면 3이, 아래 방향 스크롤이 감지디면 4를 지닙니다. button 값이 3이면 회전하는 정도에 영향을 미치는 rotation\_angle\_in\_degree가 양의 값의 TO\_RADIAN으로 설정되어서 rotate\_polygon 함수에 의해 선분이 반시계방향으로 회전합니다. 반대로 button 값이 4이면 rotation\_angle\_in\_degree가 음의 값의 TO\_RADIAN으로 설정되어서 선분이 시계방향으로 회전합니다.

rotate\_line 함수에서는 translate\_polygon 함수를 호출하여 파란색 꼭짓점인 px와 py를 으로 translation을 하는 과정을 적용합니다. 이후 rotation을 수행하기 위해 의 transformation matrix에 해당하는 배열의 의 부분 배열의 값을 초기화합니다. 이러한 설정을 바탕으로 matrix multiplication에 의해 선분을 회전하여 다시 translate\_polygon 함수를 호출해서 파란색 꼭짓점을 원래 위치로 이동시키면, 파란색 꼭짓점은 그대로인 채 선분이 특정 방향으로 회전된 상태로 보입니다.

### (e) 선분의 파란색 꼭짓점 이동하기

**SHIFT + 파란색 꼭짓점을 왼쪽 마우스 버튼으로 누른 상태 + 원하는 방향으로 이동**

#### 작동 확인 방법

SHIFT 키와 마우스의 왼쪽 버튼으로 선분의 파란색 꼭짓점을 동시에 누른 상태에서 마우스를 원하는 방향으로 이동하면 마우스의 경로에 따라 선분의 파란색 꼭짓점이 이동합니다.

#### 구현 방법

register\_callbacks에서 glutMouseFunc callback 함수에 파라미터로 넣은 mousepress 함수는 마우스의 왼쪽 버튼을 누를 때 전역변수 leftbuttonpressed의 값을 1로 설정합니다. 이때 왼쪽 마우스로 클릭한 위치가 선분의 파란색 꼭짓점의 위치인지를 알아야 하므로 꼭짓점의 위치인 px, py로부터 네 방향으로 만큼 뻗어나간 사각형 범위안에 들어오는지를 조건문으로 판단하며, 이를 만족하면 p\_pressed\_by\_left\_button이 true 값을 지닙니다. 또한 왼쪽 마우스 버튼을 떼면 leftbuttonpressed 값이 0으로 설정됩니다.

register\_callbacks에서 glutMotionFunc callback 함수에 파라미터로 넣은 mousemove 함수는 파란색 꼭짓점의 위치인 px와 py를 각각 mouse\_cur\_x와 mouse\_cur\_y에서 빼 줌으로써 파란색 꼭짓점이 현재 위치에서 이동해야 할 방향과 거리를 구해서 각각 mouse\_dx와 mouse\_dy에 저장합니다. 만약 leftbuttonpressed 값이 1이면서 shift를 누른 상태이고, p\_pressed\_by\_left\_button도 true 상태이면 translate\_line 함수를 실행합니다.

translate\_line 함수는 파란색 꼭짓점의 위치인 px와 py를 기준으로 앞서 구한 mouse\_dx와 mouse\_dy의 값만큼 translation을 수행합니다.

### (f) 비대칭 다각형 이동하기

**ALT + 오른쪽 마우스 버튼을 누른 상태 + 원하는 방향으로 이동**

#### 작동 확인 방법

ALT 키와 마우스의 오른쪽 버튼을 동시에 누른 상태에서 마우스를 원하는 방향으로 이동하면 다각형이 마우스가 움직이는 경로의 방향을 따라 같은 속도로 이동합니다.

#### 구현 방법

register\_callbacks에서 glutMouseFunc callback 함수에 파라미터로 넣은 mousepress 함수는 마우스의 오른쪽 버튼을 누를 때 전역변수 rightbuttonpressed의 값을 1로 설정합니다. 만약 오른쪽 마우스 버튼을 떼면 rightbuttonpressed 값이 0으로 설정됩니다. register\_callbacks에서 glutMotionFunc callback 함수에 파라미터로 넣은 mousemove 함수는 마우스의 이전 위치 좌표인 mouse\_prev\_x와 mouse\_prev\_y의 값을 가지고 각각 mouse\_cur\_x와 mouse\_cur\_y에서 빼 줌으로써 마우스가 얼만큼 이동했는지를 구하여 각각 mouse\_dx와 mouse\_dy에 저장합니다. 만약 rightbuttonpressed 값이 1이면 translate\_polygon 함수에 다각형의 데이터(다각형의 무게 중심점, 다각형을 구성하는 꼭짓점의 좌표, 다각형의 꼭짓점의 개수)를 파라미터로 넘겨서 실행합니다.

translate\_polygon 함수는 다각형의 무게 중심점인 object\_center\_x와 object\_center\_y를 기준으로 앞서 구한 mouse\_cur\_x와 mouse\_cur\_y의 값만큼 translation을 수행합니다.

### (g) 비대칭 다각형 크기 변경하기

**CTRL + 오른쪽 마우스 버튼을 누른 상태 + 왼쪽 or 오른쪽 이동**

#### 작동 확인 방법

**CTRL 키**를 누른 상태에서 **왼쪽 마우스 버튼**을 클릭하여 **왼쪽으로**(즉, 음의 축 방향으로) **움직이면** 커서가 윈도우에서 움직인 양에 따라서 다각형이 자신의 무게 중심점을 중심으로 크기가 작아집니다.

**CTRL 키**를 누른 상태에서 반대로 **오른쪽으로** 움직이면 그 크기가 커집니다.

#### 구현 방법

register\_callbacks에서 glutMouseFunc callback 함수에 파라미터로 넣은 mousepress 함수는 마우스의 오른쪽 버튼을 누를 때 전역변수 rightbuttonpressed의 값을 1로 설정합니다. 만약 오른쪽 마우스 버튼을 떼면 rightbuttonpressed 값이 0으로 설정됩니다. register\_callbacks에서 glutMotionFunc callback 함수에 파라미터로 넣은 mousemove 함수는 마우스의 이전 위치 좌표인 mouse\_prev\_x와 mouse\_prev\_y의 값을 가지고 각각 mouse\_cur\_x와 mouse\_cur\_y에서 빼 줌으로써 마우스가 얼만큼 이동했는지를 구하여 각각 mouse\_dx와 mouse\_dy에 저장합니다. 만약 rightbuttonpressed 값이 1이고 mouse\_dx 값이 0보다 작으면 마우스 오른쪽 버튼을 클릭한 상태에서 마우스가 왼쪽으로 이동한 것이므로 scale\_polygon 함수의 파라미터인 scaling\_factor에 0.95를 넘기고, mouse\_dx 값이 0보다 크면 마우스를 오른쪽으로 이동한 것이므로 scaling\_factor에 1.05를 넣어서 실행합니다.

scale\_polygon 함수는 다각형의 무게 중심점인 object\_center\_x와 object\_center\_y를 기준으로 translate\_polygon 함수를 호출하여 다각형을 원점으로 translation 한 후 scaling에 대응되도록 의 transformation matrix에 해당하는 배열의 대각선 성분을 특정 값으로 설정하는데, 크기를 작아지게 만들어야 하므로 scaling factor를 0.95로 설정합니다. 이후 scaling을 수행하고 다시 다각형이 원래 자리로 돌아오게 합니다.

### (h) 추가 구현

추가 구현에서는 하나의 새로운 직사각형에 관하여 scaling, shearing, rotation, translation의 네 가지 affine transformation을 수행할 수 있는 기능을 구현했습니다.

#### 직사각형 그리기

##### 작동 확인 방법

프로그램 실행 시 OpenGL 좌표계를 기준으로 , , , 위치의 네 개의 회색 꼭짓점, 검정색의 직사각형의 무게 중심점, 그리고 녹색 선분과 남색 선분을 가지는 **직사각형**을 그립니다.

##### 구현 방법

흰색 꼭짓점의 위치와 직사각형의 검정색 무게 중심점은 initialize\_renderer 함수에서 배열 rectangle의 원소에 값을 할당하여 결정됩니다. 이후 윈도우 화면을 그릴 때 register\_callbacks에서 glutDisplayFunc callback 함수로 지정한 display 함수가 실행되는데, 함수의 내용을 순차적으로 실행할 때 draw\_rectangle 함수가 호출되어 직사각형의 구성 요소들을 앞서 초기화한 배열 rectangle을 사용하여 그립니다.

#### 직사각형 크기 변경하기

**CTRL + 왼쪽 마우스 버튼을 누른 상태 + 왼쪽 or 오른쪽 이동**

##### 작동 확인 방법

**CTRL 키**를 누른 상태에서 **왼쪽 마우스 버튼을 클릭**하여 **왼쪽으로**(즉, 음의 축 방향으로) **움직이면** 커서가 윈도우에서 움직인 양에 따라서 직사각형이 자신의 무게 중심점을 중심으로 크기가 작아집니다.

**CTRL 키**를 누른 상태에서 반대로 **오른쪽으로 움직이면** 그 크기가 커집니다.

##### 구현 방법

register\_callbacks에서 glutMouseFunc callback 함수에 파라미터로 넣은 mousepress 함수는 마우스의 왼쪽 버튼을 누를 때 전역변수 leftbuttonpressed의 값을 1로 설정합니다.

만약 왼쪽 마우스 버튼을 떼면 leftbuttonpressed 값이 0으로 설정됩니다. register\_callbacks에서 glutMotionFunc callback 함수에 파라미터로 넣은 mousemove 함수는 마우스의 이전 위치 좌표인 mouse\_prev\_x와 mouse\_prev\_y의 값을 가지고 각각 mouse\_cur\_x와 mouse\_cur\_y에서 빼 줌으로써 마우스가 얼만큼 이동했는지를 구하여 각각 mouse\_dx와 mouse\_dy에 저장합니다. 만일 leftbuttonpressed 값이 1이고 mouse\_dx 값이 0보다 작으면 마우스 왼쪽 버튼을 클릭한 상태에서 마우스가 왼쪽으로 이동한 것이므로 scale\_polygon 함수를 실행할 때 scaling\_factor에 0.95를 넘기고, mouse\_dx가 0보다 크면 scaling factor를 1.05로 설정하여 scale\_polygon 함수를 호출합니다.

scale\_polygon 함수는 직사각형의 무게 중심점인 rectangle\_center\_x와 rectangle\_center\_y를 기준으로 translate\_polygon 함수를 호출하여 직사각형을 원점으로 translation 한 후 scaling에 대응되도록 의 transformation matrix에 해당하는 배열의 대각선 성분을 특정 값으로 설정하는데, 크기를 작아지게 만들어야 하므로 scale factor를 0.95로 설정합니다. 이후 scaling을 수행하고 다시 직사각형이 원래 자리로 돌아오게 합니다.

#### 직사각형 회전하기

**ALT + 방향키(←, →)**

##### 작동 확인 방법

**ALT 키를 누른 상태**에서 **왼쪽 방향키(←)**를 누르면 직사각형이 반시계방향으로 회전하고, 반대로 **오른쪽 방향키(→)**를 누르면 직사각형이 시계방향으로 회전합니다.

##### 구현 방법

register\_callbacks에서 glutSpecialFunc callback 함수에 파라미터로 넣은 special 함수는 방향키 중 하나를 누르면 자동으로 호출됩니다. 이때 함수 안에서 glutGetModifiers 함수에 의해 ALT 키를 눌렀는지 확인합니다. 왼쪽 방향키와 ALT 키를 동시에 누른 조건이 만족되면 회전하는 정도에 영향을 미치는 rotation\_angle이 양의 값의 TO\_RADIAN으로 설정되어서 rotate\_polygon 함수에 의해 직사각형이 반시계방향으로 회전합니다. 반대로 오른쪽 방향키와 ALT 키를 동시에 누른 조건이 만족되면 rotation\_angle이 음의 값의 TO\_RADIAN으로 설정되어서 직사각형이 시계방향으로 회전합니다.

rotate\_polygon 함수에서는 translate\_polygon 함수를 호출하여 직사각형의 중심 좌표인 rectangle\_center\_x와 rectangle\_center\_y를 으로 translation 하는 과정을 적용합니다. 이후 rotation을 수행하기 위해 의 transformation matrix에 해당하는 배열의 의 부분 배열의 값을 초기화합니다. 이러한 설정을 바탕으로 matrix multiplication에 의해 직사각형을 회전하여 다시 translate\_polygon 함수를 호출하면 회전된 직사각형이 원래의 위치로 이동합니다.

#### 직사각형 기울이기

**CTRL + 방향키(←, →, ↑, ↓)**

##### 작동 확인 방법

**CTRL 키를 누른 상태**에서 **왼쪽 방향키(←)**를 누르면 직사각형이 축을 기준으로 왼쪽으로 기울어지고, 반대로 **오른쪽 방향키(→)**를 누르면 직사각형이 축을 기준으로 오른쪽으로 기울어집니다.

또한 **CTRL 키를 누른 상태**에서 **위쪽 방향키(↑)**를 누르면 직사각형이 축을 기준으로 위쪽으로 기울어지고, 반대로 **아래쪽 방향키(↓)**를 누르면 직사각형이 축을 기준으로 아래쪽으로 기울어집니다.

##### 구현 방법

register\_callbacks에서 glutSpecialFunc callback 함수에 파라미터로 넣은 special 함수는 방향키 중 하나를 누르면 자동으로 호출됩니다. 이때 함수 안에서 glutGetModifiers 함수에 의해 CTRL 키를 눌렀는지 확인합니다. 왼쪽 방향키와 CTRL 키를 동시에 누른 조건이 만족되면, 기울어지는 축이 축인지 여부인 x\_axis이 true로 되고 기울어지는 정도에 영향을 주는 shearing\_fcator가 음의 값으로 설정되어서 shear\_polygon 함수에 의해 직사각형이 축 기준으로 왼쪽으로 기울어집니다. 반대로 오른쪽 방향키와 CTRL 키를 동시에 누른 조건이 만족되면 x\_axis이 true로 되고 shearing\_factor가 양의 값으로 설정되어서 직사각형이 축 기준으로 오른쪽으로 기울어집니다.

shear\_polygon 함수에서는 translate\_polygon 함수를 호출하여 직사각형의 중심 좌표인 rectangle\_center\_x와 rectangle\_center\_y를 으로 translation 하는 과정을 적용합니다. 이후 shearing을 수행하기 위해 의 transformation matrix에 해당하는 배열의 왼쪽 위의 부분 배열의 대각선 값을 shearing\_factor로 초기화합니다. 이러한 설정을 바탕으로 matrix multiplication에 의해 직사각형을 기울여서 다시 translate\_polygon 함수를 호출하면 기울어진 직사각형이 원래의 위치로 이동합니다.

#### 직사각형 이동하기

**ALT + 왼쪽 마우스 버튼을 누른 상태 + 원하는 방향으로 이동**

##### 작동 확인 방법

ALT 키와 마우스의 왼쪽 버튼을 동시에 누른 상태에서 마우스를 원하는 방향으로 이동하면 직사각형이 마우스가 움직이는 경로의 방향을 따라 같은 속도로 이동합니다.

##### 구현 방법

register\_callbacks에서 glutMouseFunc callback 함수에 파라미터로 넣은 mousepress 함수는 마우스의 왼쪽 버튼을 누를 때 전역변수 leftbuttonpressed의 값을 1로 설정합니다. 만약 왼쪽 마우스 버튼을 떼면 leftbuttonpressed 값이 0으로 설정됩니다. register\_callbacks에서 glutMotionFunc callback 함수에 파라미터로 넣은 mousemove 함수는 마우스의 이전 위치 좌표인 mouse\_prev\_x와 mouse\_prev\_y의 값을 가지고 각각 mouse\_cur\_x와 mouse\_cur\_y에서 빼 줌으로써 마우스가 얼만큼 이동했는지를 구하여 각각 mouse\_dx와 mouse\_dy에 저장합니다. 만약 leftbuttonpressed 값이 1이면 translate\_polygon 함수에 직사각형 데이터(직사각형의 무게 중심점, 직사각형을 구성하는 꼭짓점의 좌표, 직사각형의 꼭짓점의 개수)를 파라미터로 넘겨서 실행합니다.

translate\_polygon 함수는 직사각형의 무게 중심점인 rectangle\_center\_x와 rectangle\_center\_y를 기준으로 앞서 구한 mouse\_dx와 mouse\_dy의 값만큼 translation을 수행합니다.