**기초 컴퓨터 그래픽스**

**HW5 README**

20181625 남현준

1. **프로그램 작성 및 실행환경**

본 HW 3의 작성 및 구동은 Windows 10 64bit, AMD Ryzen 5 5600X, NVIDIA GeForce RTX 3070, Visual Studio Release x64에서 진행되었습니다.

1. **HW5 구현사항**
   1. **Gouraud/phong shading을 적용한 물체**
      * + - 구현여부 : 예
          - 확인방법 : 프로그램이 실행된 다음, 특정 위치에서 텍스처 매핑이 미적용된 호랑이가 움직이고 있는 것을 확인할 수 있을 것이다. 해당 호랑이는 시작될 때, phong shading의 효과가 적용되고 있으며 ‘1’와 ‘2’의 입력을 통해 색이 다르게 적용되는 것을 확인할 수 있을 것이다.
          - 구현방법 :

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

Gouraud shader의 구현을 위해 prepare\_shader\_program()에서 Gouraud shader의 info를 추가해주었다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

이때, Gouraud shader와 phong shader는 glGetUniformLocation을 통해 다른 변수로 할당되어야 한다. 그렇지 않을 경우, shader에서 충돌이 일어나 둘 중 하나의 shader 효과만 구현되기 때문이다.

셰이더의 설정이 끝났다면 입력에 따라 다른 셰이더의 효과가 호랑이 오브젝트에 적용될 수 있도록 구현을 해야했다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

다음은 draw\_tiger\_20181625()에서 화면상에 그려지게 될 호랑이에 Phong shader효과를 적용시켜 그리게 하는 코드이다. Gouraud shader또한 비슷한 방식으로 구현이 되어있으며 ShaderProgram\_TXPS 대신 ShaderProgram\_Gouraud를 사용하고 phong shader와는 다른 변수들을 사용하지만, 기본적으로 동일한 로직을 따라 코드가 진행된다.

이 때, shader flag 변수를 통해 현재 상태가 ‘1’을 누른 상태인지, ‘2’를 누른 상태인지 알 수 있게 된다.

* 1. **임의의 3D 물체에 텍스처 적용**
     + - * 구현여부 : 예
         * 확인방법 : 프로그램이 실행되며 그려지는 화면 상에 움직이는 늑대와 옵티머스 오브젝트에 기본적으로 제공된 텍스처가 아닌 직접 만든 그림파일을 사용한 텍스처가 입혀지게 될 것이다.
         * 구현방법 : 해당 오브젝트들에 적용될, 텍스처로 사용하고 싶은 그림을 포토샵의 다양한 브러시들과 필터효과 등을 이용해 입히고 싶은 느낌의 효과를 제작했다. 해당 파일을 My\_glTexImage2D\_from\_file\_20181625()을 이용해 .png 확장자나 .jpg 확장자로 된 파일을 읽어 물체에 적용시킨다.
  2. 세상 좌표계를 기준으로 하는 광원
     + - * 구현여부 : 예
         * 확인방법 : 프로그램이 실행되면 ‘3’키를 통해 해당 광원을 껐다 켰다할 수 있다. 이를 통해 확인할 수 있을 것이다.
         * 구현방법 : initialize\_lights()함수에서 해당 광원의 초기설정을 진행했다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

0번 광원의 경우, HW5와 함께 기본적으로 제공되는 광원이므로 light[1]에 세상 좌표계를 기준으로 하는 광원의 정보를 저장했다.

* 1. 눈 좌표계를 기준으로 하는 광원
     + - * 구현여부 : 예
         * 확인방법 : 프로그램이 시행되면 ‘4’키를 통해 해당 광원을 껐다 켰다할 수 있다. 기본적으로 프로그램이 시작됨과 동시에 같이 그려지기 때문에 쉽게 확인할 수 있을 것이다.
         * 구현방법 : initialize\_lights()함수에서 해당 광원의 초기설정을 진행했다.

텍스트, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명

해당 광원은 눈좌표계를 기준으로 할 것이기 때문에 초기 위치의 설정을 카메라의 position에 맞춰 설정했다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

키보드와 마우스를 통해 프로그램에서 사용되는 카메라의 정보를 갱신하게 된다. 이에 맞춰, 광원 또한 정보를 갱신해줘야 할 필요가 있었다. 따라서, 마우스를 통해 시점의 방향을 변경될 경우, updatevector를 이용해 시점의 방향이 바뀔 때마다 정보를 갱신했고, 키보드 방향키 상하좌우의 입력이 들어올 때마다 카메라의 position을 갱신해 광원 또한 카메라와 같이 이동할 수 있도록 설정했다 .

* 1. 물체의 모델링 좌표를 기준으로 하는 광원
     + - * 구현여부 : 예
         * 확인방법 : 프로그램에서 그려지는 호랑이를 따라다니는 광원을 확인할 수 있을 것이다. ‘5’의 입력을 통해 해당 광원 on/off시킬 수 있다.
         * 구현방법 : initialize\_lights()함수에서 해당 광원의 초기설정을 진행했다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Light[3]에 해당 구현에서 사용될 광원의 정보를 저장했다. 해당 광원 물체의 모델링 좌표를 기준으로 배치가 진행된다.

호랑이를 따라다니도록 의도하기 때문에 호랑이를 그릴 때 사용한 행렬의 정보를 이용하도록 했다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

스크린샷, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명

호랑이를 그릴 때 사용된 행렬을 MClight에 저장하고, MClight를 이용해서 light[3]의 position, direction을 갱신해주도록 했다.

* 1. 육면체와 같은 물체에 투명한 효과를 생성
     + - * 구현여부 : 예
         * 확인방법 : 프로그램 상에 그려진 큐브는 ‘6’키를 누르면 블렌딩 모드에 진입하고 ‘.’, ‘,’키를 통해 큐브의 불투명도를 조절할 수 있을 것이다.
         * 구현방법 :

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

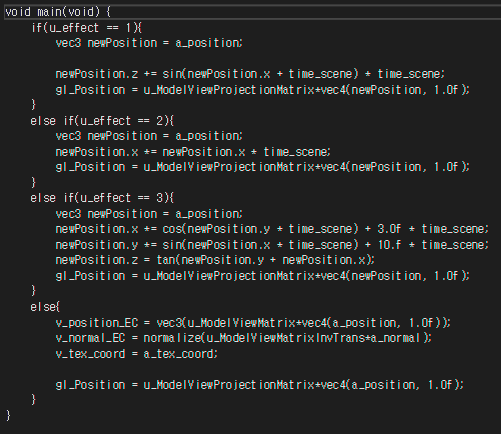
위와 같은 코드를 통해 blending 옵션을 on/off 상태일 때 큐브를 그리게 된다. glCullFace()를 통해 은면을 제거한다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

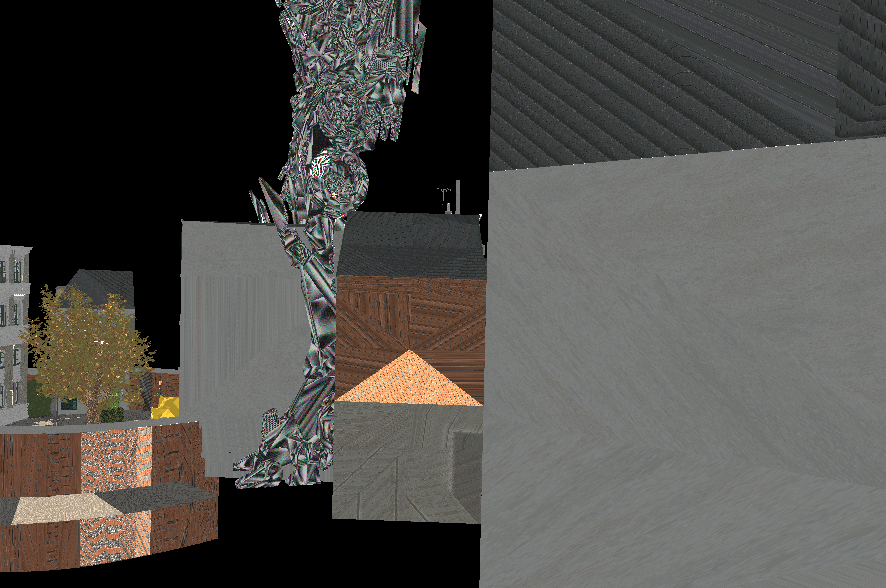
해당 그림을 통해 blending 옵션과 어떤 방식으로 큐브의 alpha 값을 조절할 수 있는지 확인할 수 있다.

* 1. 추가 구현
     + - * 구현여부 : 예
         * 확인방법 : ‘7’, ‘8’, ‘9’에 해당되는 키를 누르면 해당 키에 구현된 효과가 화면 상에 출력된다. 해당 구현들은 퐁 쉐이딩을 적용한다고 가정된 상태에서 구현이 진행되었으므로 ‘1’와 ‘2’ 중 어떤 것이 눌러진 상태인지에 따라 화면 상에 출력되는 결과물이 달라질 수 있다.
         * 구현방법 : HW5을 구현하기 위해 주어진 phong\_tx.vert 파일에서 어떤 입력이 들어오는지에 따라 vertex의 설정 조건을 다르게 해주었다.



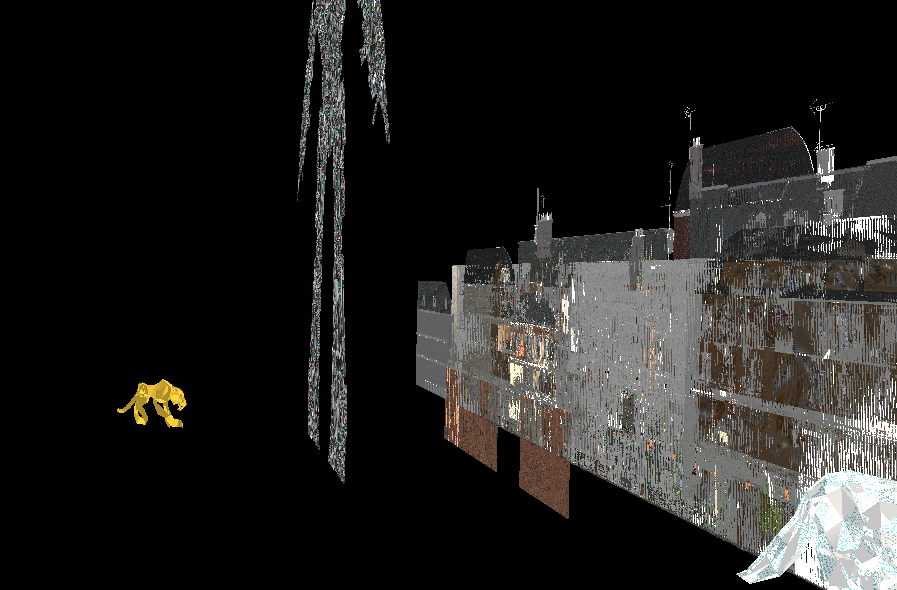
위에서부터 순서대로 ‘7’, ‘8’, ‘9’가 입력으로 주어졌을 때의 vertex shader 코드이다.

‘7’의 경우, 화면 상에 출력되는 결과물에 빛이 다각형 모양으로 쪼개져서 물체에 적용되게 된다.



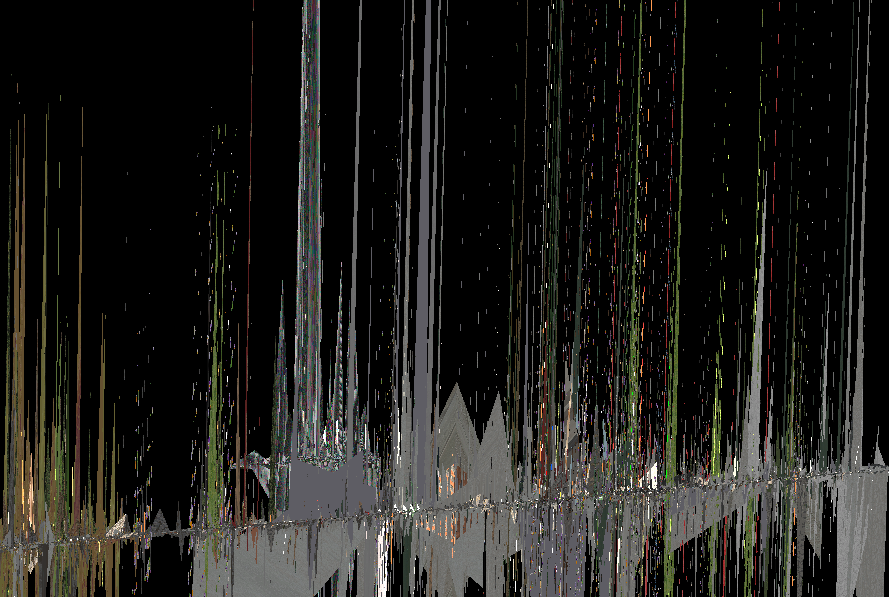
‘7’ 실행 결과 예시

‘8’의 경우, 프로그램에서 배치된 오브젝트들이 3D에서 2D형태로 납작해졌을 때의 결과물이 출력된다.



‘8’ 실행결과 예시

‘9’의 경우, 프로그램에서 사용된 오브젝트들이 가시모양으로 변한 상태의 결과물이 출력되게 된다.



‘9’ 실행결과 예시