컴퓨터 그래픽스

프로젝트 보고서

20163087 김도은 : 물체 배치, 각 물체 선택 및 이동 등
20163095 김수은 : 물체 배치, 각 물체 선택 및 이동 등
20163159 이희지 : 키보드로 카메라 이동, 충돌 처리 등
20163180 홍자현 : 키보드로 카메라 이동, 충돌 처리 등

목차

1. 장면 파일

2. 박물관 네비게이션

3. 물체 변환

4. 물체 표현

5. 단일 선택

6. 충돌 감지

7. FPS게임 요소

8. 창의적 요소

*물체의 텍스처 매핑

-텍스처 마다 인덱스를 두어 물체를 그리기 전에 어떤 텍스처를 쓸 지 인덱스를 통해 선택하고 텍스처를 입혀준다.

*가상 박물관의 장면(scene) 데이터의 표현

1. 장면 파일

ex) (obj 파일명) (y축 회전각:rotation) (확대축소값:scale) (삼차원 이동벡터:translate) (텍스쳐 파일명)

- 텍스트 파일을 만들어주고 처음 초기의 object의 값들을 기록해 준다. 그리고 물체가 움직일 때마다 값이 달라지므로 텍스트 파일에 추가로 적어준다. 움직인 정도와 값의 변화를 쉽게 파악할 수 있다. 아래 예시로 모든 객체의 초기값들이 기록되고 TV 객체를 움직였을 때의 변화를 추가로 기록해준다.

```
desk_with_normals   0: angle    (0,1,0)    (-1.5,1.5,1.5)    (-5,0,0)      desk,jpg
tv_with_normals     0: angle    (0,1,0)    (2,2,2)           (0,0,-5)      tv.jpg
sofa_with_normals   180: angle  (0,1,0)    (1.5,1.5,1.5)     (0,0,5)       sofa.jpg
fan_with_normals    30: angle   (0,1,0)    (1.5,1.5,1.5)     (0,5,0)       fan.jpg
tv_with_normals     0: angle    (0,1,0)    (2,2,2)           (-0.1,0,-5)   tv.jpg
tv_with_normals     0: angle    (0,1,0)    (2,2,2)           (-0.2,0,-5)   tv.jpg
tv_with_normals     0: angle    (0,1,0)    (2,2,2)           (-0.2,0,-5.1) tv.jpg
tv_with_normals     0: angle    (0,1,0)    (2,2,2)           (-0.2,0,-5.2) tv.jpg
```

*가상 박물관 네비게이션

2. 박물관 네비게이션: 키보드를 통해 가상 박물관을 네비게이션 할 수 있도록 한다.

- 키보드의 방향키를 누르면 카메라가 좌, 우, 앞, 뒤로 이동해서 볼 수있다.
- 키보드의 'q', 'e'를 누르면 카메라의 yaw효과를 주도록 설계했다.

3. 물체의 변환: 가상 박물관에 전시 될 각각의 물체는 OBJ 파일로 구성되며, 각 물체의 위치를 world 좌표계로 변환되기 위해서는 각 물체별로 회전/확대축소/변환에 관련된 행렬이 정의되어 있어야 한다.

- 물체의 변환은 행렬과 벡터의 형식을 빌어 표현할 수 있다. 따라서, 보고서에는 이 부분에 대해 물체의 변환을 수식을 통해 기술하고, 해당 수식이 OpenGL 상에 어떻게 대응되어 코딩되었는지 표현하도록 한다.

- 물체의 변환은 각각 mat_Translate, mat_Roate, mat_Scale 행렬을 만든다. 각 행렬을 물체마다 적절한 크기로 곱하여 (T*R*S) 물체를 변화시킨다.

- **Code**
  mat_model = translateMatrix * scaleMatrix * rotateMatrix;
  mat_PVM = mat_Proj * mat_View_inv * mat_model;

4. 물체의 표현: 각 물체는 재질(material)과 빛(light)의 상호 작용을 통한 색상과 텍스쳐 이미지 정보를 통해 렌더링 된다.

- 물체의 렌더링이 어떠한 방식으로 이루어지는지에 관해 이론적으로 설명하고, OpenGL 코드에서 어떻게 이를 표현하였는지 기술한다.

- 각 물체는 쉐이더 기반의 Modern OpenGL (OpenGL 2.x 이상)을 사용하여 Phong reflection model을 통해 렌더링한다.

- Ambient Reflection + Diffuse Reflection + Specular Reflection + shinness

- 수식으로 표현
  $I = \frac{1}{a+bd+cd^2}(I_d + I_s) + I_a = \frac{1}{a+bd+cd^2}(k_d max((l \cdot n), 0)L_d + k_s max((r \cdot v)^{\alpha}, 0)L_s + k_a L_a$

- 텍스처 마다 인덱스를 두어 물체를 그리기 전에 어떤 텍스처를 쓸 지 인덱스를 통해 선택하고 텍스처를 입혀준다.

- 각각의 물체에 texture를 입히기 위해서 각각의 jpg 파일들을 mapping시켰다.

5. 단일 선택 : 삼차원 상에 여러 물체가 있을 때, 이들 중 사용자가 원하는 물체를 선택하여 정보를 변형 하는 것은 간단한 일이 아니다. 각 팀은 보고서에 가상 박물관에 전시되는 여러 모델 중 특정 모델을 어떻게 효과적으로 선택할 것인지 설계하고, 선택된 물체의 위치를 사용자의 입력을 통해 어떻게 변형하였는지 설명한다.

- **F1 키**를 누르면 **TV**가 선택이 되고, 키보드의 **a,s,d,w를 사용**하여 **상, 하, 좌, 우** 로 **0.1f**만큼 이동시킬 수 있다. 또한 키보드의 **z, c,를 사용**하여 선택된 물체를 **좌우로 10도 만큼 회전** 할 수 있는 기능을 구현하였다.

- 이와 같은 방법으로 **F2는 DESK**, **F3는 SOFA**, **F4는 FAN**을 변환시킬 수 있다.
- **F6**은 **사람**이다.

- FAN의 rotate기능은 별도로 구현하지 않았다.

- **F6** 을 누르면 선택한 물체가 **선택 취소**되고, 그 후 a, s, d, w 를 눌러도 다시 해당 물체의 선택 키를 누르기 전에는 변환되지 않는다.

6.충돌 감지 :

- 가상 박물관 네비게이션 시, 장면에 놓인 물체를 뚫고 지나가지 않도록 충돌 감지 기능을 구현하도록 하자.

- obj가 load될 때, 원시 물체들 각각의 **물체의 중심, bounding sphere의 반지름**을 구한다.

- 이때 bounding sphere에서, obj파일의 점들을 모아 평균으로 물체의 중심을 구하고, 중심에서 최대좌표 점(max)까지의 거리를 반지름으로 둔다.

- 각 물체가 변환시 translate된 만큼 중심을 이동하고, scale된 만큼 반지름을 바꾼다.

- 키보드와 방향키로 **카메라를 이동할때마다, collision함수**가 호출돼 충돌을 감지한다.

- collision함수는, 카메라의 위치를 업데이트해 **물체의 중심에서 카메라까지의 거리(d)**를 계산하고, 이를 **물체의 반지름(r)**과 비교해 **r>d인 경우에 튕겨**나오도록 구현했다.

7. FPS 게임적 요소 : 화면에 보이는 물체에 대해 물체가 없어지는 기능을 구현하였다.

- 입력키를 받아 물체를 선택한다. 물체는 main.cpp안에 display()함수에서 그려진다. 이때 물체마다 조건을 두어 조건이 true일 경우만 물체를 그릴 수 있게 하였다.

- 물체를 선택하고 키보드 **'O'** 을 누르면 조건이 false로 바뀌어 **물체가 그려지지 않는다.** 따라서 물체가 없어지고 다시 키보드 **'P'** 을 누르면 조건이 true로 바뀌어 **물체가 그려진다.**

8. 창의적인 요소 : 각자가 생각할 수 있는 창의적인 요소를 나름대로 구현하였다.

환경 구성

- 전시 환경을 연출하였다.

충돌감지

-충돌이 일어나지 않았을 경우에만 이동한다.

```
 if(collision(d) == false)
{
        g_camera.move_forward(0.1f);
}
glutPostRedisplay();
```

-물체가 앞으로 이동했을 때 충돌을 확인하고 튕겨져 나오는 느낌을 표현하였다.

```
g_camera.move_forward(0.1f);
if(collision(d) == true)
{
        g_camera.move_backward(0.2f);
}
glutPostRedisplay();
```