

2019320090 강지원 과제3 레포트

서론

과제3의 세부 사항에 대한 레포트입니다.

본론

- OBJ
 - 총 6개의 obj가 사용됩니다

```
vector<string> obj_list = {  
    "../objs/steve.obj",  
    "../objs/spider.obj",  
    "../objs/blackspider.obj",  
    "../objs/grass.obj",  
    "../objs/stones.obj",  
    "../objs/cube.obj",  
    "../objs/skybox.obj",  
};
```

- 3 인물 모델 - 2 배경 요소 - 1 Glossy box (Environment Mapping) - 1 Skybox 순으로 로드합니다

- Texture
 - `vector<string> image_list` : Skybox를 제외한 5개 텍스처 이미지 경로를 담고 있습니다.

```
vector<string> image_list = {  
    "../textures/steve.bmp",  
    "../textures/spider.bmp",  
    "../textures/blackspider.bmp",  
    "../textures/grass.bmp",  
    "../textures/stones.bmp",  
};
```

- `vector<string> skybox_list` : Skybox 텍스처 경로를 담고 있습니다.

```
vector<string> skybox_list = {  
    "../textures/skybox/right.bmp",  
    "../textures/skybox/left.bmp",  
    "../textures/skybox/bottom.bmp",  
    "../textures/skybox/top.bmp",  
    "../textures/skybox/front.bmp",  
    "../textures/skybox/back.bmp"};
```

- 주요 User 함수 설명
 - `void loadObj()` : 모든 obj 파일들을 로드합니다
 - `void loadSkyboxTexture()` : Skybox texture를 로드합니다.
 - `void idle()` : 시간을 측정하고, 모델 Animation(Rotation) 행렬을 계산합니다.

```
if (dt > 0.01)  
{  
    for (int i = 0; i < 3; i++)  
        World_list[i] = RotateY(1) * World_list[i];  
    old_t = t;  
}
```

- 렌더링 과정
 - `init()`
 - `loadObj()` 를 통해 모든 obj 파일들의 데이터를 로딩합니다.

- `vertices`, `uvs`, `normals` 벡터에 모든 메쉬 정보를 저장합니다.
- `indices_list` 는 `obj_list` 순서대로 오브젝트별 인덱스 정보를 저장합니다.
- 읽어들이는 정보를 이용해 VBO 및 IBO를 생성합니다.
 - `ibo_list` 에는 오브젝트별 IBO를 저장합니다.
- 총 3개의 셰이더 프로그램을 사용합니다
 - `p` : 일반 모델 렌더링
 - `p2` : Skybox 렌더링
 - `p3` : Glossy box 렌더링
- 모든 텍스처 파일을 로딩합니다.
 - for문으로 `images_list` 의 모든 텍스처를 로드하고, `color_tex_list` 에 텍스처 버퍼 정보를 저장합니다.
 - `loadSkyboxTexture()` 를 통해 Skybox(Cubemap) 텍스처를 로드합니다.
- `renderScene(void)`
 - Rotation, Translation, Scale을 고려하여 World, View 및 Projection 행렬을 계산합니다.
 - 가장 먼저, Skybox를 렌더링합니다.
 - Rotation 효과만을 적용한 `cur_modelView_skybox` 을 사용합니다.
 - Depth Buffer를 사용하지 않고 렌더링합니다.
 - 이후 일반 모델 → Glossy box 순으로 렌더링을 진행합니다.

Shader Variable

Input of vertex shader

- `vPosition` : vertex position
- `vNormal` : vertex normal
- `vTexCoord` : texture coordinate

Output of vertex shader (Input of fragment shader)

- `fTexCoord` : texture coordinate
- `fTexCoord_cube` : cube mapping을 위한 texture coordinate (Skybox)
- `worldNormal` : World 좌표계 기준 Vertex Normal
- `worldPos` : World 좌표계 기준 Vertex Position
- `cameraPos` : World 좌표계 기준 Eye coordinate

Output of fragment shader

- `fColor` : fragment color

결론

- 실행 화면은 다음과 같습니다.

