Homework #3: Phong reflection assimp

컴퓨터 그래픽스
국민대학교 소프트웨어학부

Abstract

본 과제에서는 Open Asset Import Library (Assimp)를 사용하여 간단한 모델 파일을 불러와서 phong reflection model을 적용하고 간단한 키보드 조작을 통해 object의 회전, light의 위치 변화, 카메라 mode 조작 등을 OpenGL을 이용해 랜더링한다.

1 코드 설명

1.1 vertex의 normal 정보 확인

```
bool load_asset(const std::string& filename)
{
  scene = aiImportFile(filename.c_str(), aiProcessPreset_TargetRealtime_MaxQuality);
  // ...
}
```

- assimp library의 "postprocess.h" file을 통해 normal을 자동으로 계산해준다.
- "aiProcessPreset_TargetRealtime_MaxQuality"의 value인 "aiProcessPreset_TargetRealtime_Quality"의 value인 "aiProcess_GenSmoothNormals"을 통해 normal을 자동으로 계산해준다.
- 이외의 추가 내용은 "postprocess.h"문서를 참조한다.

```
void print_mesh_info(const aiMesh* mesh)
{
  std::cout << "print mesh info" << std::endl;

  std::cout << "num vertices " << mesh->mNumVertices << std::endl;
  for (int i = 0; i < mesh->mNumVertices; ++i)
  {
    aiVector3D vertex = mesh->mVertices[i];
    std::cout<<"vertex("<<vertex.x<<","<<vertex.y<<","<<vertex.z<<")"<<std::endl;

    if (mesh->mNormals != NULL)
    {
      aiVector3D normal = mesh->mNormals[i];
      std::cout<<"normal(" <<normal.x<<","<<normal.y<<","<<normal.z<<")"<<std::endl;
    }
  }
  // ...
}
```

- normal에 대한 값들은 3차원 vector normal에 저장된다.
- normal값들은 이전 과제의 vertex, color와 같은 방식으로 저장이 된다.

1.2 set_transform() 함수

```
void set_transform()
{
  kmuvcl::math::vec3f eye     = camera.position();
  kmuvcl::math::vec3f up      = camera.up_direction();
  kmuvcl::math::vec3f center  = eye + camera.front_direction();

  mat_view = kmuvcl::math::lookAt(eye[0], eye[1], eye[2],
                                  center[0], center[1], center[2],
                                  up[0], up[1], up[2]);

  float n = camera.near();
  float f = camera.far();

  if (camera.mode() == Camera::kOrtho)
  {
    float l = camera.left();
    float r = camera.right();
    float b = camera.bottom();
    float t = camera.top();

    mat_proj = kmuvcl::math::ortho(l, r, b, t, n, f);
  }
  else if (camera.mode() == Camera::kPerspective)
  {
    mat_proj = kmuvcl::math::perspective(camera.fovy(), g_aspect, n, f);
  }

  // set object transformation
  mat_model = kmuvcl::math::rotate(g_angle*0.7f, 0.0f, 0.0f, 1.0f);
  mat_model = kmuvcl::math::rotate(g_angle*1.0f, 0.0f, 1.0f, 0.0f)*mat_model;
  mat_model = kmuvcl::math::rotate(g_angle*0.5f, 1.0f, 0.0f, 0.0f)*mat_model;
  mat_model = kmuvcl::math::translate(0.0f, 0.0f, -4.0f)*mat_model;
}
```

- camera의 'eye, up, center'의 좌표값을 설정한다.
- 'eye, up, center' vector들을 활용하여 view matrix를 설정한다.
- camera의 near, far를 설정한다.
- camera의 mode가 'Orthogonal', 'Perspective'에 따라 projection matrix를 설정한다.
- rotate와 translate를 통해 model matrix를 설정한다.

2 함수 작성

2.1 init_buffer_object() 함수 작성

```
void init_buffer_objects()
{
  // TODO: Fill this function to initiazlize buffer objects

}
```

- 이전 'HW02 helloassimp' 과제에서 수행한 init_buffer_object() 함수를 수정한다.

2.2 key_callback() 함수 작성

```
void key_callback(GLFWwindow * window, int key, int scancode, int action, int mods)
{
  // TODO: Fill this function to keyboard setting

}
```

- 'Q' 키를 누르면 camera의 mode가 바뀌어야 한다.
  예) 'Orthogonal mode' ↔ 'perspective mode'
- 'L' 키를 누르면 light가 -x축 방향으로 이동이 되어야한다.
- 'R' 키를 누르면 light가 +x축 방향으로 이동이 되어야한다.
- 'D' 키를 누르면 light가 -y축 방향으로 이동이 되어야한다.
- 'U' 키를 누르면 light가 +y축 방향으로 이동이 되어야한다.
- 'B' 키를 누르면 light가 -z축 방향으로 이동이 되어야한다.
- 'F' 키를 누르면 light가 +z축 방향으로 이동이 되어야한다.
- 'P' 키를 누르면 model이 회전이 되도록한다.

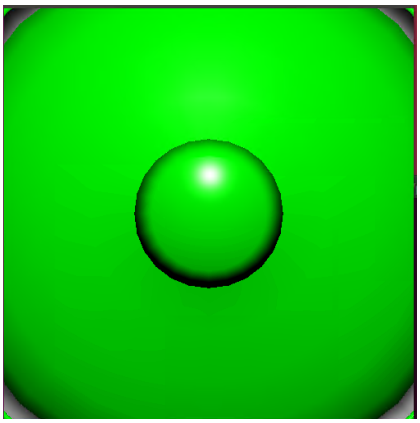
2.3 render_object() 함수 작성

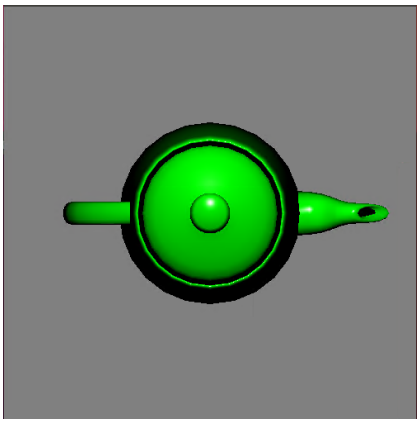
```
void render_object()
{
  // 특정 쉐이더 프로그램 사용
  glUseProgram(program);

  // TODO: Fill this function to render the scene


  // 쉐이더 프로그램 사용해제
  glUseProgram(0);
}
```

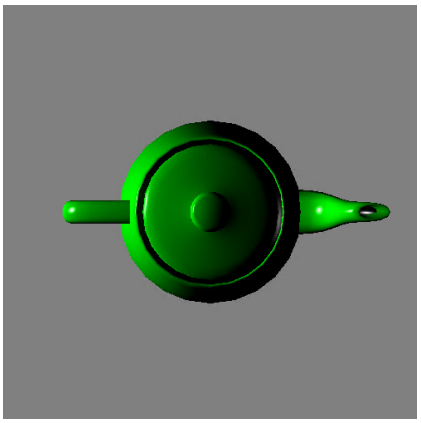
- 이전 'HW02 helloassimp' 과제에서 수행한 render_object() 함수를 수정한다.

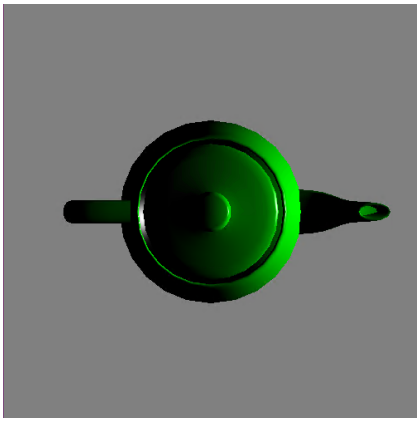
3 실행 방법 및 결과

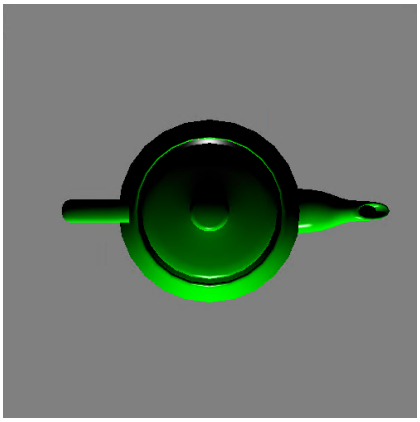
실행 방법으로 makefile을 활용하여 컴파일을 진행한 후, 실행 결과로 assimp library API를 활용하여 model을 불러 온다. 이 때, model은 phong reflection이 적용되어야 하며, 위의 조건에 맞춰서 카메라와 light에 대한 변화가 있어야한다. 아래의 사진은 "./phongassimp models/teapot.ply" 실행 결과의 예이다. **반드시 "./phongassimp [filename]" 형식으로 실행시킬 것!!!**

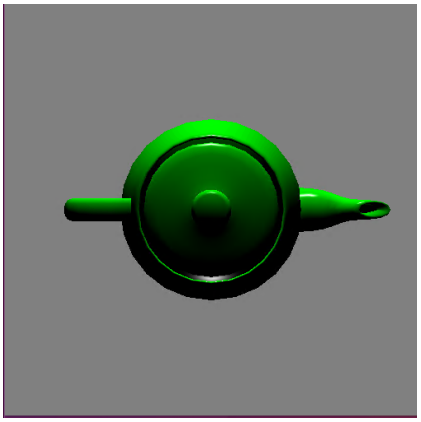

처음 실행 화면

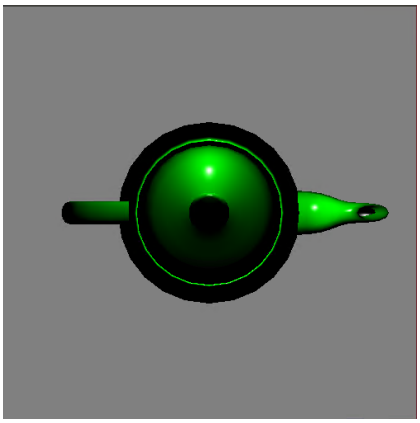

Q키를 통한 camera mode 변경

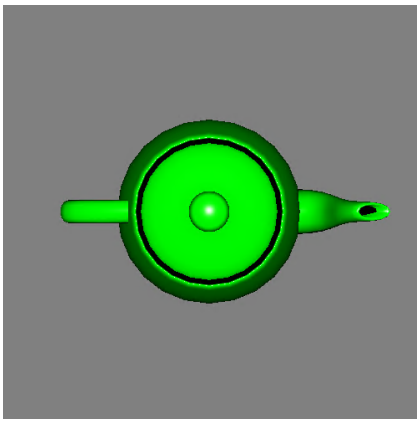

L키를 통한 light position 변경

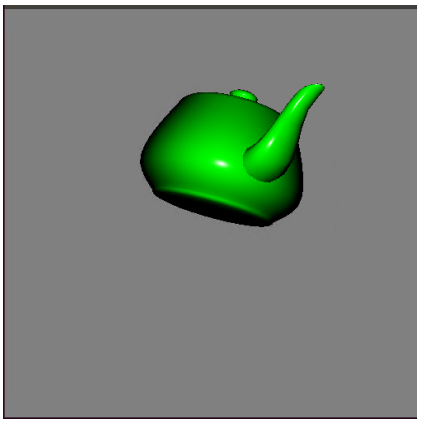

R키를 통한 light position 변경


D키를 통한 light position 변경


U키를 통한 light position 변경


B키를 통한 light position 변경


F키를 통한 light position 변경


P키를 통한 teapot 회전

4 과제 제출 방법(매우 중요!!!)

- 본 과제는 개인과제이며, 각자 자신의 코드를 완성하도록 한다.
- 공지된 마감 시간까지 과제 코드를 가상대학에 업로드하도록 한다.
- 과제 코드는 **Ubuntu 16.04 LTS 환경에서 make 명령으로 컴파일 가능**하도록 작성한다.
- 과제 코드는 다음의 파일들은 하나의 압축파일로 묶어 **tar.gz** 파일 형식이나 표준 **zip** 파일 형식으로만 제출하도록 한다. 이 때, 압축파일의 이름은 반드시 'OOOOOOOO_HW03.tar.gz(OOOOOOOO은 자신의 학번)'과 같이 자신의 학번이 드러나도록 제출한다.

  1) 소스코드 및 리소스 파일들
  2) Makefile

- 과제에 관한 질문은 오피스 아워를 활용하도록 한다. 교육조교(teaching assisant, TA)에게 메일로 약속시간을 정한 후, zoom 등을 활용한 방법을 통해 물어보는 것도 매우 권장하는 방법이다.