Homework #4: assimp viewer

컴퓨터 그래픽스
국민대학교 소프트웨어학부

Abstract

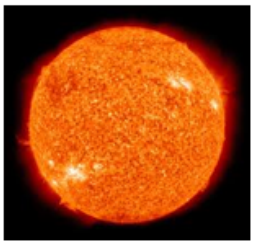
본 과제에서는 Open Asset Import Library (Assimp)를 활용하여 Phong shading과 texture mapping이 적용된 모델을 OpenGL을 이용하여 랜더링한다.

1 과제 목표

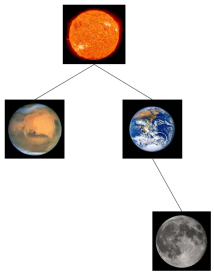
- scene을 랜더링할 때, node들을 고려해서 구현해야하는 것을 이해한다.
- Phong shading이 적용된 model을 구현한다.
- texture mapping이 적용된 model을 구현한다.
- 최종적으로 Phong shading과 texture mapping이 적용된 model을 구현한다.

2 NODE의 중요성

- 이전 과제에서 object를 랜더링하는 함수인 'render_object()'함수를 변형하였다.
- 기존의 code는 node가 1개일 때 object를 그리는 함수였다면, 이번 과제의 방법은 여러개의 node에서도 적용될 수 있도록 모든 node를 반영할 수 있도록 함수를 수정하였다. (코드 설명 참고)



node가 1개인 경우 (태양)



node가 여러개인 경우 (태양계)

- 태양, 지구, 화성, 달 등은 각각 node로 표현된다.
- 태양계를 그리기 위해서는 node의 갯수를 반영해서 그려야하며, node를 고려하지 않고 그리게 되면 오른쪽 이미지처럼 태양계를 그릴 수 없고 왼쪽의 이미지처럼 하나의 object만 그려지게 된다.
- 때문에 다양한 object들을 한 scene에 띄우기 위해서는 scene에 있는 node들을 모두 반영해서 코드를 구현해야한다.

3 코드 설명

3.1 render object

3.1.1 draw_scene() 함수

```
void draw_scene()
{
  const aiNode* node = scene->mRootNode;

  draw_node_recursive(node, mat_model);
}
```

- 각 scene은 1개 이상의 node들로 이루어져있다.
- draw_node_recursive() 함수를 통해 scene의 node들을 불러오면서 그린다.

3.1.2 draw_node_recursive() 함수

```
void draw_node_recursive(const aiNode* node, const aiMatrix4x4& mat_parent)
{
  aiMatrix4x4 mat_curr = mat_parent*node->mTransformation;

  for (int i = 0; i < node->mNumMeshes; ++i)
  {
    draw_mesh(scene->mMeshes[node->mMeshes[i]], mat_curr);
  }

  for (int i = 0; i < node->mNumChildren; ++i)
  {
    draw_node_recursive(node->mChildren[i], mat_curr);
  }
}
```

- node에 있는 mesh 갯수만큼 반복하면서 draw_mesh() 함수를 통해 node에 있는 mesh들을 불러오면서 그린다.
- node가 2개 이상이 있다면 draw_node_recursive()함수를 통해 모든 node를 불러오면서 그린다.

3.1.3 draw_mesh() 함수

```
void draw_mesh(const aiMesh* mesh, const aiMatrix4x4& mat_model)
{
  glUseProgram(program);

  aiMatrix4x4 mat_PVM = mat_proj*mat_view*mat_model;
  glUniformMatrix4fv(loc_u_PVM, 1, GL_FALSE, (float*)&mat_PVM.Transpose());

  aiMatrix4x4 m = mat_model;
  glUniformMatrix4fv(loc_u_M, 1, GL_FALSE, (float*)&m.Transpose());

  glUniform3fv(loc_u_view_position_wc, 1, (float*)&camera.mPosition);

  // ...
  }

  glUseProgram(0);
}
```

- 각 node당 갖고 있는 mesh들을 불러오면서 그리는 함수이며, 기존 코드의 'render_object()'함수에 해당한다.
- assimp github material.h의 matrix 구조는 OpenGL에서 읽어드리는 행과 열의 방향이 다르다. 그렇게 때문에 'Trans-Pose()'를 적용하여 OpenGL이 읽어드리는 방향과 맞게 적용한다.
- view position은 assimp의 camera.h로 부터 position 정보를 받아오고 'init()'함수에서 position의 값을 초기화시킨다.

3.2 material info 출력

3.2.1 print_material_info() 함수

```
void print_material_info(const aiMaterial* material)
{
  std::cout << "print material info" << std::endl;

  for (int i = 0; i < scene->mNumMaterials; i++)
  {
    aiMaterial* currentMaterial = scene->mMaterials[i];
    aiString materialName;

    currentMaterial->Get(AI_MATKEY_NAME, materialName);

    unsigned int diffuseTextureCount=currentMaterial->GetTextureCount(aiTextureType_DIFFUSE);
        for (unsigned int j = 0; j < diffuseTextureCount; j++)
    {
      aiString textureFilePath;
      if (AI_SUCCESS == scene->mMaterials[i]->GetTexture(aiTextureType_DIFFUSE,
          j, &textureFilePath))
        std::cout << "Diffuse Texture file: " << textureFilePath.data << std::endl;
      else
        std::cout << "no exist diffuseTexture." << std::endl;
    }
  }
}
```

- material과 관련된 texture의 정보를 확인할 수 있는 함수이다.
- assimp github material.h의 'GetTextureCount()' 함수를 호출하면 texture의 갯수를 확인할 수 있다.
- assimp github material.h의 'GetTexture()' 함수를 호출하면 texture에 대한 정보를 확인할 수 있다.

3.3 camera setting

3.3.1 set_transform() 함수

```
void set_transform()
{
  camera.GetCameraMatrix(mat_view);

  if (mode() == kortho)
    kmuvcl::ortho(-1, 1, -1, 1, mat_proj);
  else if (mode() == kperspective)
    kmuvcl::perspective(mat_proj);

  aiMatrix4x4::RotationY(g_angle*pi/180.0, mat_model);
}
```

- assimp 공식 github의 camera.h의 'GetCameraMatrix()'함수를 이용하여 view matrix를 적용한다.
- 현재 카메라 모드에 따라 orthogonal mode와 perspective mode에 맞춰서 projection matrix를 적용한다.
- assimp 공식 github의 matrix4x4.h의 'RotationY()'를 이용하여 model matrix에 회전 정보를 추가할 수 있다.

4 함수 작성

4.1 init_buffer_object() 함수

```
void init_buffer_objects()
{
  // TODO: Fill this function to initialize texture coordinate buffer objects
  //...

}
```

- 이전 과제의 'init_buffer_objects()' 함수에 texture coordinate 정보를 추가한다.

4.2 init_texture_object() 함수

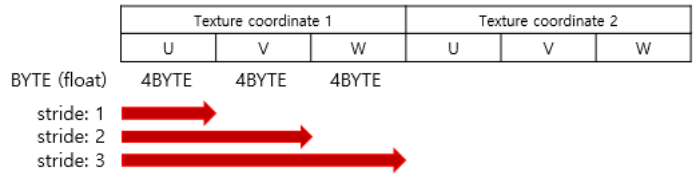
```
void init_texture_objects()
{
  // TODO: Fill this function to initialize texture objects
  //...

}
```

- kmuvcl youtube를 참고하여 'init_texture_objects()'함수를 구현한다.

4.3 draw_mesh() 함수

```
void draw_mesh(const aiMesh* mesh, const aiMatrix4x4& mat_model)
{
  // TODO: Fill this function to render the scene
  //...
}
```

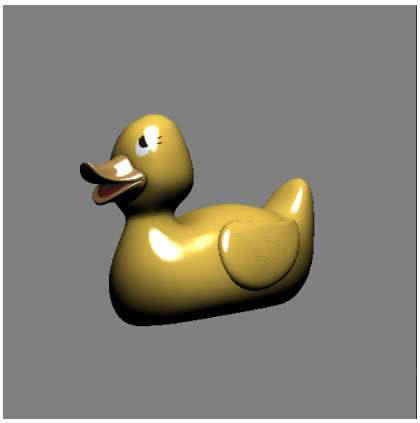
- 주어진 코드 'draw_mesh()' 함수에 object에 Phong shading과 texture mapping을 적용하기 위한 코드를 추가한다.
- 주의 사항
  - assimp의 texture coordinate는 (u,v,w)로 불러와지는데 해당 과제에서는 (u,v) 좌표만을 사용한다. 그렇기 때문에 w 좌표값을 불러오지 않도록 구현해야한다.
  - 이에 대한 방법으로는 'glVertexAttribPointer()'를 호출 할 때, stride를 적용해야 한다. 이때, texture coordinate의 자료형을 고려하여 stride 값을 주어야한다.



- 여러개의 object에 각각의 texture를 적용시키기 위해서는 mesh.h의 'unsigned int mMaterialIndex'를 참고한다.

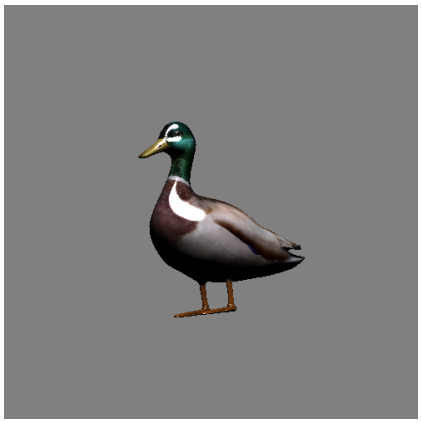
5 실행 방법 및 결과

실행 방법으로 makefile을 활용하여 컴파일을 진행한 후, 실행 결과로 assimp library API를 활용하여 model을 불러 온다. **반드시 "./viewer [filename]" 형식으로 실행시킬 것!!!**



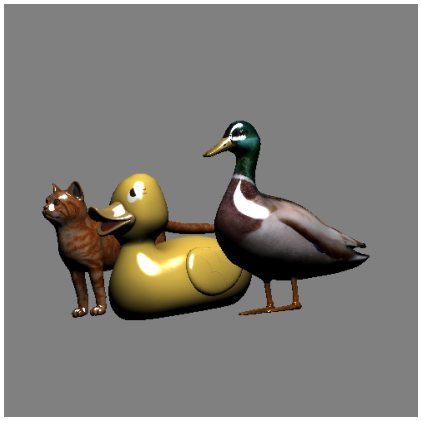
duck.obj

cat.obj



bird.obj

spider.obj



animals.obj

hulk.obj

6 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_HW04.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.

7 실시간 데모 평가

- 가상대학 공지사항의 링크를 참고하여 원하는 시간에 자신의 학번과 이름을 작성한다.
- 07월 01일(수) 07월 02일(목) 기간 동안, 개인별로 zoom을 통해 데모평가를 받도록 한다.
- 실시간 데모 평가 시, 개인별로 제출했던 HW#00, #01, #02, #03에 대한 평가 또한 받도록 한다.