Computer Graphics Report

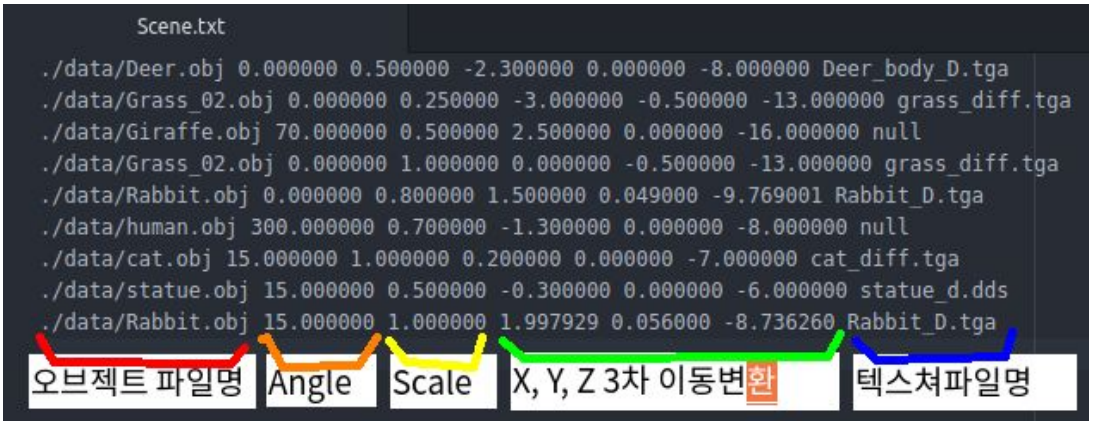
컴퓨터공학부 20143042 김용태
컴퓨터공학부 20143075 안은영
컴퓨터공학부 20163156 정지연
컴퓨터공학부 20163170 최은주

1. 동물 박물관의 장면(Scene) 데이터 (김용태, 안은영)

- Scene의 기능은 저희팀은 구조체와 txt 파일을 사용하여 구현하였습니다. 구조체 안에는 오브젝트의 파일명, 회전각, scale, (x,y,z)이동벡터, 텍스쳐 파일명이 포함되어 있습니다.

  이 기능은 화면의 오른쪽을 클릭하면 Save / Load가 나오고 누르면 바로 기능이 작동하도록 되어 있습니다. Save를 클릭하면 현재 물체들의 위치,회전값,크기값 등 의 정보가 Scene.txt 파일에 저장이 됩니다. 그리고 Load를 하면 가장 최근에 저장한 값이 불러오도록 구현했습니다. Save 함수는 호출이 되면 각 물체별로 저장되어 있는 Scene 구조체를 통해 텍스트 파일에 그 정보들을 입력하고 프로그램이 실행되는 생기는 임시변수에도 그 정보들을 입력합니다. . Load 함수는 처음 호출 할 때에 텍스트 파일에서 정보를 가져오고 그 이후로는 프로그램 내에 임시변수들로 정보들을 가지고 Load를 합니다.

  구현 시 처음에는 텍스트 파일을 이용하여 정보를 삽입하고 불러오는 형식으로만 구현하고 임시 변수를 사용하지 않았는데, 이럴경우 프로그램이 돌아가고 있는 중에는 텍스트 파일로의 정보 저장이 즉각 이루어지지 않아, 임시변수를 이용하였습니다.



  저희 Scene.txt파일은 이런 형태로 이루어져있습니다.

2. 네비게이션 (최은주, 김용태, 정지연)

- 박물관을 이리저리 다니며 관람이 가능하도록 네비게이션 기능을 구현했습니다.
  기본 이동 동작인 박물관 이동은 ←→↑↓ 키를 통해 이동할 수 있습니다.
  W A S D 키를 가지고는 원하는 방향을 볼 수 있습니다. W는 위방향, A는 왼쪽방향,

S는 아래방향, D는 오른쪽 방향입니다.

추가적으로 기능을 더 구현해 보았습니다. FPS 게임적 요소를 느끼게 하기 위해서 점프 기능을 추가했습니다. ‘z' 또는 'Z'를 누르면 1인칭 시점인 카메라가 점프를 하게 됩니다.
그리고 저희는 동물 박물관이 테마인데, 동물 박물관에 살아 있는 토끼는 돌아 다닐 수 있다고 생각하여, 토끼가 랜덤하게 점프를 하면서 이동하는 기능을 구현 하였습니다. Object들의 front_dir_과 up_dir_을 이용하여, 점프를 구현하였습니다.

물체의 충돌이 감지 될 경우 뚫고 지나가지 않게 하기 위해, Object.cpp 파일에서 각 Object들의 정보를 받을 때, vertex의 값중 x,y,z값이 가장 큰 값과 가장 작은값을 찾아내어, 각 물체에 대한 Boundary Box를 만들었습니다. 이후 박스들이 겹치면 충돌이 감지되도록 구현하려 했으나, 이는 전부 구현하지 못하고 Boundary 박스까지만 구현했습니다.

```
void Object::make_boundary()
{
  kmuvcl::math::vec4f temp0(minX, minY, minZ); V[0] = temp0;
  kmuvcl::math::vec4f temp1(minX, minY, maxZ); V[1] = temp1;
  kmuvcl::math::vec4f temp2(maxX, minY, maxZ); V[2] = temp2;
  kmuvcl::math::vec4f temp3(maxX, minY, minZ); V[3] = temp3;
  kmuvcl::math::vec4f temp4(minX, maxY, minZ); V[4] = temp4;
  kmuvcl::math::vec4f temp5(minX, maxY, maxZ); V[5] = temp5;
  kmuvcl::math::vec4f temp6(maxX, maxY, maxZ); V[6] = temp6;
  kmuvcl::math::vec4f temp7(maxX, maxY, minZ); V[7] = temp7;
}
```

(편의성 고려 - 키보드 사용)
또한 이용자가 원하면 화면에 있는 물체들을 원하는 장소에 놓을 수 있게 구현 하였습니다. 어떤 물체를 옮기고 싶다면, 해당 물체의 특정 번호를 입력하고 움직일 수 있습니다. 저희 동물 박물관에서는 'F1’ 키는 고양이, 'F2’와 ‘F3'은 토끼, 'F4’는 기린, ‘F5’는 관람객, "F6’은 사슴은, 'F7'은 조각상입니다. 해당 키를 누른 뒤 ‘I’ ‘J’ ‘K’ ‘L’ 를 이용하여 오브젝트를 원하는 위치로 옮길 수 있습니다. I는 뒤쪽방향, J는 왼쪽방향, K는 앞쪽방향, L은 오른쪽방향 으로 해당 키를 누르면 물체를 이동시킬 수 있습니다.
오브젝트의 회전과 확대축소 기능도 있습니다. ‘O’키를 누르면 물체가 왼쪽으로 회전하고, 'P'키를 누르면 물체가 오른쪽으로 회전합니다. ‘Y’ 키를 누르면 물체가 작아지고, 'U’ 키를 누르면 물체가 커집니다.
이로써 박물관을 투어하는 기능과 원하는대로 꾸미는 것이 가능합니다.

3. 물체의 변환 (정지연, 김용태)

- 물체를 변환하기 위해서는 3차원공간의 이동벡터, y축 기준 회전각, 크기값 이 세가지가 필요합니다. 그래서 저희팀은 각 물체별로 이 정보들을 따로 저장하고 관리하고 있습니다. 위에서 Scene을 구현하려면 각 물체는 이동벡터, 회전각, 크기값을 가지고 있어야 합니다. 그래서 물체의 변환을 할 때마다 Scene에 들어있는 이 정보들을 이용해서, 매 변환시 translate 행렬과 rotate행렬, scale행렬을 정의해줍니다.

저희 조각상을 예를 들어 설명하겠습니다. 다음장의 그림을 보시면

```
mat_scale = kmuvcl::math::scale(statue.scale, statue.scale, statue.scale);
mat_rotate = kmuvcl::math::rotate(statue.angle, 0.0f, 1.0f, 0.0f);
mat_trans = kmuvcl::math::translate(statue.translate(0), statue.translate(1), statue.translate(2));
mat_Model = mat_trans * mat_scale * mat_rotate;
mat_PVM = mat_Proj*mat_View_inv*mat_Model;
```

statue.scale, statue.angle, statue.translate는 Scene 구조체에 있는 변수들입니다. 이를 이용하여 trasnlate 행렬과 rotate 행렬, scale 행렬을 정의한 후 Model 행렬을 만들어 World좌표계로 만들어줍니다. 1인칭 시점으로 카메라 입장에서 바라보는 mat_PVM로 구현 이후 mat_PVM은 uniform 변수로 묶여, vertex shader에서 position을 정해주는 부분에 사용이 됩니다.

4. 물체의 표현 (정지연, 안은영, 최은주)

- 각 obj 파일과 mtl 파일을 읽어와서 각 물체의 재질(material) 설정과 텍스처 매핑을 합니다. 텍스처 매핑은 Object.cpp에 load_simple_mtl에서 map_Kd(diffuse reflection coefficient로 활용)도 읽어 오게 추가 해주어 load_texture() 에서 텍스트 파일을 로드합니다. (SOIL.h 이용)

```
void Object::load_texture(const std::string& filename)
{
  std::ifstream file(filename.c_str());

  textureid = SOIL_load_OGL_texture
  (
    filename.c_str(),
    SOIL_LOAD_AUTO,
    SOIL_CREATE_NEW_ID,
    SOIL_FLAG_INVERT_Y
  );

  if (textureid == 0)
  {
    std::cerr << "Fail to load an image file with SOIL_load_OGL_texture() function." << std::endl;
    return;
  }

  glBindTexture(GL_TEXTURE_2D, textureid);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
}
```

물체의 렌더링은 SOIL_load_OGL_texture로 GPU에 공간만들고 id를 생성하고 glBindTexture()로 id를 묶어줍니다. glTextParameteri()로 wrapping 모드, 필터링 방법을 정해줍니다.

glBindTexture()로 해당 아이디를 지정받으면, draw() 함수를 통해 렌더링합니다. 이후 fragment shader에서 phong reflection model 로 색상을 계산해줍니다. (텍스쳐 매핑 활용)

<텍스쳐 매핑 전>   <텍스쳐 매핑 후>