Final Project

제출일	2025.06.17	전 공	컴퓨터소프트웨어공학과
과 목	컴퓨터그래픽스	학 번	20214045
담당교수	홍민 교수님	이름	류재민

목차

1. 문제분석 3

1.1 1번 문제 3

1.2 2번 문제 3

1.3 3번 문제 3

1.4 4번 문제 4

1.5 5번 문제 5

2. 소스코드 6

2.1 20214045_류재민_FinalProject.cpp 6

2.2 model.h 16

2.3 model.cpp 16

2.4 camera.h 18

2.5 camera.cpp 18

2.6 lodepng.h / lodepng.cpp 19

3. 소스코드 분석 20

3.1 20214045_류재민_FinalProject.cpp 20

3.2 model.cpp 29

3.3 camera.cpp 30

4. 실행화면 32

5. 느낀점 48

1. 문제분석

1.1 1번 문제

> 2025 컴퓨터그래픽스 Final Project 문제
>
> 1. 본인이 디자인한 obj 파일을 읽어서 사각형 바닥과 함께 화면에 렌더링 하는 프로그램을 작성 하시오. (5점)

마지막 수업 시간에 교수님께서 주신 **OBJ Viewer**의 소스코드를 참고했습니다. 다만 **model.h**와 **model.cpp**로 분리해서 메인 소스코드의 가독성을 향상시켰습니다. 소스코드에서 .obj 파일의 이름만 고쳐서 실행하면 크기와 위치가 다르기 때문에 전체가 보이지 않았습니다. 그래서 크기와 위치를 변경해야 했고 직교 투영을 사용하기 위해서 **glFrustum()** 대신 **glOrtho()**를 사용해주었습니다. 공룡의 전체가 보이도록 렌더링이 완료된 후 발 밑에 사각형 바닥을 그려주었습니다. 크기는 위에서 볼 때 사각형 안에 공룡이 다 담길 수 있는 크기로 그려주었습니다. 마지막으로 .obj 파일 로드 시 동적 메모리 해제를 **free()**를 통해 관리했습니다.

1.2 2번 문제

> 2. 위, 아래, 옆, 무작위 4방향에서 바라보는 4개의 Viewport를 보여주는 프로그램을 작성하시오. (10점)

수업 시간에 실습했던 **04_2 Viewport 예제**의 소스코드를 참고했습니다. 창을 4개의 동일한 크기의 뷰포트로 나누었습니다. 각 뷰 포트는 **gluLookAt()** 함수 사용하여 고유한 카메라의 시점을 가집니다. 무작위의 뷰 포트는 3DS MAX에서 Perspective 시점을 참고했습니다. 해당 시점이 공룡이 가장 입체적으로 보여 자신 있는 시점이었습니다. **glPushMatrix()** 함수와 **glPopMatrix()** 함수를 사용하여 각 뷰 포트의 변환 상태를 독립적으로 관리했습니다. 그리고 각 뷰 포트의 구분을 하기 위해서 중앙에 십자선을 그려주었습니다.

1.3 3번 문제

> 3. OpenGL에서 제공하는 gluLookAt 카메라 제어 함수를 이용하여 4개의 뷰포트에서 따로 움직일 수 있는 카메라 기능을 작성하시오. (10점)
> ex) 1번 누르면 '위' 뷰포트의 카메라 선택, 방향키를 이용해서 카메라 이동(상하좌우).
> 3번 누르면 '앞' 뷰포트의 카메라 선택, 방향키를 이용해서 카메라 이동(상하좌우).

처음에 문제가 살짝 이해가 되지 않았습니다. 상하좌우가 카메라가 상하좌우로 이동을 의미하는 건지, 모델을 중심으로 회전하는 듯한 상하좌우 이동을 의미하는 건지 헷갈렸습니다. 하지만 교수님께서 과제에 대해 설명해주실 때 **07_4 RGB Color Cube**처럼 물체가 아닌 카메라가 돌고 있는 걸 떠올리고 거

기서 마우스가 아닌 방향키로 하라고 하셔서 저는 후자로 이해했고 모델을 중심으로 회전하듯이 카메라 제어를 했습니다. 따라서 **07_4 RGB Color Cube**의 **eyePosition()** 함수를 참고했고 움직임 처리도 참고했습니다. 숫자키를 통해 뷰 포트를 선택하고 해당 뷰 포트에서만 방향키를 통해 카메라가 이동되도록 구현했습니다. **gluLookAt()** 함수를 사용했고 회전은 **theta**와 **phi**를 조정하여 구현됩니다. 이를 통해 카메라의 위치와 상향 벡터를 계산합니다. 카메라의 위치는 구면 좌표계(theta, phi, r)를 사용하여 계산되고 **theta**와 **phi**를 조정하여 카메라의 위치를 업데이트합니다. 앞선 **model.h**와 **model.cpp**와 마찬가지로 **camera.h**와 **camera.cpp**로 분리했습니다.

1.4 4번 문제

4. [GLUT: 팝업메뉴] 마우스 오른쪽 버튼을 누를 시, 아래 메뉴를 출력하고 해당하는 기능을 구현하시오. (15점)
메뉴 1: 사각형 바닥의 색 변경 (랜덤)
메뉴 2: obj 파일 물체의 색 변경 (랜덤)
메뉴 3: 조명 On/Off(토글)
메뉴 4: 조명 색상 변경 (랜덤)
메뉴 5: 텍스쳐 On/Off(토글)

- 팝업 메뉴

마우스 오른쪽 버튼 시 팝업 메뉴가 나타나고 문제의 5가지 메뉴로 구성했습니다. 각 메뉴는 **glutCreateMenu()** 함수와 **glutAddMenuEntry()** 함수를 사용해서 생성했고 선택 시에 **menuCallback()** 함수를 호출해서 기능이 실행됩니다. 메뉴는 **glutAttachMenu(GLUT_RIGHT_BUTTON)** 함수를 통해서 마우스 오른쪽 버튼을 누를 때 메뉴가 출력되도록 했습니다.

- 메뉴 1: 사각형 바닥의 색 변경

바닥 색은 처음에 배열에 초기값이 저장되어 있고 메뉴를 누르면 RGB 값을 랜덤으로 변경해서 적용했습니다.

- 메뉴 2: obj 파일 물체의 색 변경

바닥 색과 마찬가지로 처음에 배열에 초기값이 저장되고 RGB 값을 랜덤으로 변경해서 적용했습니다.

- 메뉴3: 조명 On/Off

조명은 변수를 통해 상태를 관리하고 **glEnable(GL_LIGHT0)** 함수와 **glDisalbe(GL_LIGHT0)** 함수로 On/Off를 적용했습니다.

- 메뉴4: 조명 색상 변경

조명 색상은 diffuse와 specular 배열에 RGB 값을 랜덤으로 설정하고 **glLightfv()** 함수로 적용했습니다. 제가 만든 공룡 .obj를 저장하면서 같이 생성된 **.mtl** 파일에서 **Kd**, **Ks**만 명시되어 있어서 ambient는

따로 할당하지 않았습니다. 조명 위치는 배열에 저장했습니다.

- 메뉴5: 텍스처 On/Off

텍스처도 조명과 마찬가지로 변수를 통해 상태를 관리합니다. 텍스처는 수업 시간에 실습한 **lodepng.h**와 **lodepng.cpp**를 사용했습니다. 조명을 통해 어두워질 것을 고려하여 밝은 느낌의 텍스처를 찾아서 적용했습니다. 또한 공룡의 등에 있는 뿔, 눈, 눈 흰자, 이빨들에 서로 다른 텍스처를 입혀주고 싶어서 본래 하나였던 공룡 .obj를 3DS MAX에서 Detach로 분리해주었고 각각의 .obj로 저장했습니다. 각각의 .obj 파일을 불러와서 서로 다른 텍스처를 입혀주었습니다.

1.5 5번 문제

> 5. [보너스] 본인이 원하는 기능을 이 프로그램에 추가로 구현하시오. (5점)

개인적으로 보너스 문제가 가장 어려웠습니다. 좋은 아이디어가 잘 떠오르지 않았고 어떤 것을 해야 높은 점수를 받을 수 있을지 감이 오지 않았습니다. 하지만 높은 점수를 받으려는 것 보다는 4번까지 구현을 마친 후 좀 더 있었으면 좋겠다는 기능들을 추가해보자는 마음가짐으로 진행했습니다. 기본적으로 3DS MAX의 기능들을 추가해보려고 했고 그렇게 추가한 새로운 기능들 5가지입니다**. g/G키 입력 시 그리드(격자) 그리기, 조명 그리기(시각화), 조명 이동하기, 애니메이션 추가하기, 카메라 이동 단축키** 들로 구성되어 있습니다.

- 그리드(격자) 그리기

3DS MAX에서 g/G 키를 입력하면 그리드가 나타나는데 이를 똑같이 구현해봤습니다.

- 조명 그리기 및 이동하기

조명에 따라서 모델이 어떻게 렌더링 되는지 직접 확인하고 싶어서 화면에 조명을 그리고 x/X, y/Y, z/Z 키를 입력하여 이동할 수 있는 기능을 추가했습니다. 실시간으로 조명 위치를 업데이트 하여 모델과 바닥에 적용되는 조명 효과를 확인할 수 있습니다.

- 메뉴 6: 애니메이션 추가하기

완전 멋진 애니메이션을 구현하진 못했지만 회전과 진폭을 사용하여 구현했습니다. 처음에는 회전으로만 구현했는데 너무 밋밋해서, 팔과 다리를 3DS MAX에서 Detach로 분리해주고 .obj로 저장해서 적용했습니다. 진폭이 아래로 가면 팔/다리와 몸통이 분리되어서 빈 공간이 보이는데 이를 방지하기 위해서 분리돼서 빈 공간처럼 보이는 부분의 팔/다리를 따로 .obj로 저장해서 여기에는 진폭을 주지 않았습니다. 그렇게 했더니 자연스러운 애니메이션을 만들 수 있었습니다.

- 카메라 이동 단축키

3DS MAX에서 단축키를 누르면 바로 모델의 앞이나 뒤로 카메라가 이동하는 기능이 있는데 이를 똑같이 구현했습니다. 이는 화살표의 짧은 이동을 한 번에 할 수 있고 선택된 뷰 포트에서만 작동합니다.

2. 소스코드

2.1 20214045_류재민_FinalProject.cpp

```cpp
// 20214045_류재민_FinalProject
#pragma warning(disable:4996)
#include <time.h>
#include <iostream>
#include "model.h"
#include "camera.h"
#include "lodepng.h"

/*****************
    1번 문제
*****************/

// 기본 함수
void init();
void display();
void reshape(int w, int h);

// 모델
Model dino;
Model eye;
Model eyeground;
Model teeth;
Model horn;
Model rleg;
Model lleg;
Model legCover;
Model rarm;
Model larm;
Model armCover;
void free();
void drawModel();

// 색상
GLfloat floorColor[3] = { 0.7, 0.7, 0.7 }; // 바닥 기본 색상
GLfloat modelColor[3] = { 1.0, 1.0, 1.0 }; // 모델 기본 색상

// 사각형 바닥 그리기
void drawRect();

/*****************
    2번 문제
*****************/

// 뷰포트
int viewportPosition[16] = {
    0, 0, 400, 400,     // 0: 앞
    400, 0, 400, 400,   // 1: 무작위
    0, 400, 400, 400,   // 2: 위
    400, 400, 400, 400  // 3: 옆
};

// 중앙 십자선 그리기 (뷰포트 구분용)
void drawLine();

/*****************
    3번 문제
*****************/

// 뷰포트 선택 상태 (-1은 선택되지 않음)
int viewport = -1;

```

```
62. // 키 입력
63. void specialKeys(int key, int x, int y);
64. void keyboard(unsigned char key, int x, int y);
65.
66. /*****************
67.     4번 문제
68. *****************/
69.
70. // 메뉴
71. int light = 1; // 조명 상태 (1: 켜짐, 0: 꺼짐)
72. int texture = 1; // 텍스처 상태 (0: 사용 안 함, 1: 사용)
73. void menuCallback(int value);
74.
75. // 조명 설정
76. GLfloat lightPosition[4] = { 20.0, 20.0, 30.0, 1.0 };
77. GLfloat diffuse[4] = { 0.5, 0.5, 0.5, 1.0 };
78. GLfloat specular[4] = { 1.0, 1.0, 1.0, 1.0 };
79. float lightSpeed = 1.0f; // 조명 이동 속도
80.
81. // 텍스처
82. GLuint textureID[12];
83. void initTexture(GLuint* texture, const char* path);
84.
85. /*****************
86.     5번 문제
87. *****************/
88.
89. // 애니메이션
90. float baseAmplitude = 1.5f; // 기본 진폭
91. float animationSpeed = 20.0f; // 애니메이션 속도
92. float animationAngle = 0.0f; // 애니메이션 각도
93. int animationDirection = 1; // 애니메이션 방향 (1: 증가, -1: 감소)
94. int animation = 0; // 애니메이션 상태 (1: 켜짐, 0: 꺼짐)
95. void animate(int value);
96.
97. // 그리드
98. int grid = 0; // 그리드 상태 (1: 켜짐, 0: 꺼짐)
99. void drawGrid();
100.
101. int main(int argc, char** argv) {
102.     glutInit(&argc, argv);
103.     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
104.     glutInitWindowSize(800, 800);
105.     glutInitWindowPosition(100, 100);
106.     glutCreateWindow("OBJ VIEWER");
107.
108.     init();
109.
110.     glutDisplayFunc(display);
111.     glutReshapeFunc(reshape);
112.     glutSpecialFunc(specialKeys);
113.     glutKeyboardFunc(keyboard);
114.     glutTimerFunc(0, animate, 0);
115.     glutMainLoop();
116.     free();
117.     return 0;
118. }
119.
120. void init() {
121.     // 모델 불러오기
122.     dino = ObjLoad("dino.obj");
123.     eye = ObjLoad("eye.obj");
124.     eyeground = ObjLoad("eyeground.obj");
125.     teeth = ObjLoad("teeth.obj");
126.     horn = ObjLoad("horn.obj");
127.     rleg = ObjLoad("rleg.obj");
128.     lleg = ObjLoad("lleg.obj");
129.     legCover = ObjLoad("legCover.obj");
```

```
130.     rarm = ObjLoad("rarm.obj");
131.     larm = ObjLoad("larm.obj");
132.     armCover = ObjLoad("armCover.obj");
133.
134.     glClearColor(0.2, 0.2, 0.2, 0.0);
135.     glEnable(GL_DEPTH_TEST);
136.
137.     glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
138.     glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
139.     glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
140.
141.     // 감쇠율
142.     glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 0.001);
143.     glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.0001);
144.
145.     glEnable(GL_LIGHTING);
146.     glEnable(GL_LIGHT0);
147.
148.     // 랜덤 시드 초기화
149.     srand(time(NULL));
150.
151.     // 팝업 메뉴 생성
152.     glutCreateMenu(menuCallback);
153.     glutAddMenuEntry("바닥 색상 변경", 0);
154.     glutAddMenuEntry("모델 색상 변경", 1);
155.     glutAddMenuEntry("조명 On/Off", 2);
156.     glutAddMenuEntry("조명 색상 변경", 3);
157.     glutAddMenuEntry("텍스처 On/Off", 4);
158.     glutAddMenuEntry("애니메이션 On/Off", 5);
159.     glutAttachMenu(GLUT_RIGHT_BUTTON);
160.
161.     // 텍스처 이미지 불러오기
162.     initTexture(&textureID[0], "ground.png"); // 바닥 텍스처
163.     initTexture(&textureID[1], "dino.png");   // 공룡 텍스처
164.     initTexture(&textureID[2], "eye.png");    // 눈 텍스처
165.     initTexture(&textureID[3], "eyeground.png"); // 눈 바닥 텍스처
166.     initTexture(&textureID[4], "teeth.png"); // 이빨 텍스처
167.     initTexture(&textureID[5], "horn.png"); // 뿔 텍스처
168.     initTexture(&textureID[6], "dino.png"); // 오른다리 텍스처
169.     initTexture(&textureID[7], "dino.png"); // 왼다리 텍스처
170.     initTexture(&textureID[8], "dino.png"); // 다리 커버 텍스처
171.     initTexture(&textureID[9], "dino.png"); // 오른팔 텍스처
172.     initTexture(&textureID[10], "dino.png"); // 왼팔 텍스처
173.     initTexture(&textureID[11], "dino.png"); // 팔 커버 텍스처
174. }
175.
176. void display() {
177.     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
178.     glShadeModel(GL_SMOOTH);
179.     glEnable(GL_NORMALIZE);
180.
181.     for (int i = 0; i < 4; i++) {
182.         glPushMatrix();
183.
184.         /*
185.         뷰 포트
186.                 2 3
187.                 0 1
188.         */
189.
190.         glViewport(viewportPosition[i * 4], viewportPosition[i * 4 + 1], viewportPosition[i * 4 + 2],
viewportPosition[i * 4 + 3]);
191.
192.         glMatrixMode(GL_PROJECTION);
193.         glLoadIdentity();
194.         glOrtho(-50, 50, -50, 50, -50.0, 500.0);
195.         glMatrixMode(GL_MODELVIEW);
196.         glLoadIdentity();
```

```
197.
198.         gluLookAt(cameras[i].eyeX, cameras[i].eyeY, cameras[i].eyeZ,
199.             0.0, 0.0, 0.0,
200.             cameras[i].upX, cameras[i].upY, cameras[i].upZ);
201.
202.         // 조명 설정 (카메라 뷰 적용 후 조명 위치 설정)
203.         glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
204.
205.         // 조명 그리기
206.         if (light) {
207.             glDisable(GL_LIGHTING);
208.             glPushMatrix();
209.             glTranslatef(lightPosition[0], lightPosition[1], lightPosition[2]);
210.             glColor3f(0.0, 0.0, 0.0);
211.             glutWireCube(2.0);
212.             glPopMatrix();
213.             glEnable(GL_LIGHTING);
214.         }
215.
216.         // 사각형 바닥 그리기
217.         glPushMatrix();
218.         drawRect();
219.         glPopMatrix();
220.
221.         // 그리드 그리기
222.         if (grid) {
223.             glPushMatrix();
224.             drawGrid();
225.             glPopMatrix();
226.         }
227.
228.         // 모델 그리기
229.         drawModel();
230.
231.         glPopMatrix();
232.     }
233.
234.     drawLine();
235.
236.     glutSwapBuffers();
237. }
238.
239. void reshape(int w, int h) {
240.     glViewport(0, 0, w, h);
241.     glMatrixMode(GL_PROJECTION);
242.     glLoadIdentity();
243.     glOrtho(-50, 50, -50, 50, 1.0, 500.0);
244.     glMatrixMode(GL_MODELVIEW);
245.     glLoadIdentity();
246. }
247.
248. void free()
249. {
250.     for (int i = 0; i < dino.vNum; i++) free(dino.vPoint[i]);
251.     free(dino.vPoint);
252.     for (int i = 0; i < dino.fNum; i++) free(dino.fPoint[i]);
253.     free(dino.fPoint);
254.     for (int i = 0; i < eye.vNum; i++)free(eye.vPoint[i]);
255.     free(eye.vPoint);
256.     for (int i = 0; i < eye.fNum; i++)free(eye.fPoint[i]);
257.     free(eye.fPoint);
258.     for (int i = 0; i < eyeground.vNum; i++)free(eyeground.vPoint[i]);
259.     free(eyeground.vPoint);
260.     for (int i = 0; i < eyeground.fNum; i++)free(eyeground.fPoint[i]);
261.     free(eyeground.fPoint);
262.     for (int i = 0; i < teeth.vNum; i++)free(teeth.vPoint[i]);
263.     free(teeth.vPoint);
264.     for (int i = 0; i < teeth.fNum; i++)free(teeth.fPoint[i]);
265.     free(teeth.fPoint);
266.     for (int i = 0; i < horn.vNum; i++)free(horn.vPoint[i]);
```

```
267.     free(horn.vPoint);
268.     for (int i = 0; i < horn.fNum; i++)free(horn.fPoint[i]);
269.     free(horn.fPoint);
270.     for (int i = 0; i < rleg.vNum; i++)free(rleg.vPoint[i]);
271.     free(rleg.vPoint);
272.     for (int i = 0; i < rleg.fNum; i++)free(rleg.fPoint[i]);
273.     free(rleg.fPoint);
274.     for (int i = 0; i < lleg.vNum; i++)free(lleg.vPoint[i]);
275.     free(lleg.vPoint);
276.     for (int i = 0; i < lleg.fNum; i++)free(lleg.fPoint[i]);
277.     free(lleg.fPoint);
278.     for (int i = 0; i < legCover.vNum; i++)free(legCover.vPoint[i]);
279.     free(legCover.vPoint);
280.     for (int i = 0; i < legCover.fNum; i++)free(legCover.fPoint[i]);
281.     free(legCover.fPoint);
282.     for (int i = 0; i < rarm.vNum; i++)free(rarm.vPoint[i]);
283.     free(rarm.vPoint);
284.     for (int i = 0; i < rarm.fNum; i++)free(rarm.fPoint[i]);
285.     free(rarm.fPoint);
286.     for (int i = 0; i < larm.vNum; i++)free(larm.vPoint[i]);
287.     free(larm.vPoint);
288.     for (int i = 0; i < larm.fNum; i++)free(larm.fPoint[i]);
289.     free(larm.fPoint);
290.     for (int i = 0; i < armCover.vNum; i++)free(armCover.vPoint[i]);
291.     free(armCover.vPoint);
292.     for (int i = 0; i < armCover.fNum; i++)free(armCover.fPoint[i]);
293.     free(armCover.fPoint);
294. }
295.
296. void drawModel()
297. {
298.     glTranslatef(0.0f, -20.0f, -10.0f);
299.     glScalef(0.7f, 0.7f, 0.7f);
300.     if (animation) glRotatef(animationAngle, 0.0f, 1.0f, 0.5f); // 회전 애니메이션
301.
302.     // 몸통 (dino)
303.     glPushMatrix();
304.     if (animation) glTranslatef(0.0f, -baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f), 0.0f);
305.     rendering(dino, 1);
306.     glPopMatrix();
307.
308.     // 눈 (eye)
309.     glPushMatrix();
310.     if (animation) glTranslatef(0.0f, -baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f), 0.0f);
311.     rendering(eye, 2);
312.     glPopMatrix();
313.
314.     // 눈 흰자 (eyeground)
315.     glPushMatrix();
316.     if (animation) glTranslatef(0.0f, -baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f), 0.0f);
317.     rendering(eyeground, 3);
318.     glPopMatrix();
319.
320.     // 이빨 (teeth)
321.     glPushMatrix();
322.     if (animation) glTranslatef(0.0f, -baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f), 0.0f);
323.     rendering(teeth, 4);
324.     glPopMatrix();
325.
326.     // 뿔 (horn)
327.     glPushMatrix();
328.     if (animation) glTranslatef(0.0f, -baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f), 0.0f);
329.     rendering(horn, 5);
330.     glPopMatrix();
```

```
331.
332.     // 커버는 진폭으로 인해 비워지는 공간을 메우기 위해 사용하므로 진폭 애니메이션 X
333.     rendering(legCover, 8);
334.     rendering(armCover, 11);
335.
336.     // 오른쪽 다리(rleg)
337.     glPushMatrix();
338.     if (animation) glTranslatef(0.0f, -(baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f)), 0.0f);
339.     rendering(rleg, 6);
340.     glPopMatrix();
341.
342.     // 왼쪽 다리(lleg)
343.     glPushMatrix();
344.     if (animation) glTranslatef(0.0f, -(baseAmplitude * (sinf((animationAngle + 180.0f) * 3.14159f /
animationSpeed) * 0.5f + 0.5f)), 0.0f);
345.     rendering(lleg, 7);
346.     glPopMatrix();
347.
348.     // 오른쪽 팔(rarm)
349.     glPushMatrix();
350.     if (animation) glTranslatef(0.0f, -(baseAmplitude * (sinf((animationAngle + 180.0f) * 3.14159f /
animationSpeed) * 0.5f + 0.5f)), 0.0f);
351.     rendering(rarm, 9);
352.     glPopMatrix();
353.
354.     // 왼쪽 팔(larm)
355.     glPushMatrix();
356.     if (animation) glTranslatef(0.0f, -(baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f)), 0.0f);
357.     rendering(larm, 10);
358.     glPopMatrix();
359. }
360.
361. void drawRect() {
362.     glEnable(GL_LIGHTING);
363.
364.     if (!light) {
365.         glDisable(GL_LIGHTING);
366.         glColor3f(floorColor[0], floorColor[1], floorColor[2]);
367.     }
368.     else {
369.         glEnable(GL_LIGHTING);
370.         glMaterialfv(GL_FRONT, GL_DIFFUSE, floorColor);
371.     }
372.
373.     if (texture) {
374.         glEnable(GL_TEXTURE_2D);
375.         glBindTexture(GL_TEXTURE_2D, textureID[0]);
376.         // 텍스처와 조명 조합 모드 설정
377.         glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
378.     }
379.     else {
380.         glDisable(GL_TEXTURE_2D);
381.     }
382.
383.     glBegin(GL_QUADS);
384.     // 법선 벡터 설정 (바닥은 위를 향하므로 (0, 1, 0))
385.     glNormal3f(0.0f, 1.0f, 0.0f);
386.     if (texture) glTexCoord2f(0.0f, 0.0f);
387.     glVertex3f(-25.0f, -21.0f, -45.0f);
388.     if (texture) glTexCoord2f(1.0f, 0.0f);
389.     glVertex3f(25.0f, -21.0f, -45.0f);
390.     if (texture) glTexCoord2f(1.0f, 1.0f);
391.     glVertex3f(25.0f, -21.0f, 40.0f);
392.     if (texture) glTexCoord2f(0.0f, 1.0f);
393.     glVertex3f(-25.0f, -21.0f, 40.0f);
394.     glEnd();
395.
```

```
396.     if (texture) {
397.         glDisable(GL_TEXTURE_2D);
398.     }
399. }
400.
401. void drawGrid() {
402.     glDisable(GL_LIGHTING);
403.     glDisable(GL_TEXTURE_2D);
404.     glLineWidth(1.0f);
405.
406.     // 일반 격자선 (회색)
407.     glColor3f(0.3f, 0.3f, 0.3f);
408.     glBegin(GL_LINES);
409.     for (float x = -50.0f; x <= 50.0f; x += 5.0f) {
410.         if (x != 0.0f) { // 중앙선은 제외
411.                          glVertex3f(x, -20.9f, -50.0f); // 바닥 바로 위에 위치
412.             glVertex3f(x, -20.9f, 50.0f);
413.         }
414.     }
415.     for (float z = -50.0f; z <= 50.0f; z += 5.0f) {
416.         if (z != 0.0f) { // 중앙선은 제외
417.             glVertex3f(-50.0f, -20.9f, z);
418.             glVertex3f(50.0f, -20.9f, z);
419.         }
420.     }
421.     glEnd();
422.
423.     // 중앙 십자선 (검은색)
424.     glColor3f(0.0f, 0.0f, 0.0f);
425.     glBegin(GL_LINES);
426.     glVertex3f(0.0f, -20.9f, -50.0f);
427.     glVertex3f(0.0f, -20.9f, 50.0f);
428.     glVertex3f(-50.0f, -20.9f, 0.0f);
429.     glVertex3f(50.0f, -20.9f, 0.0f);
430.     glEnd();
431.
432.     glEnable(GL_LIGHTING);
433. }
434.
435. void drawLine() {
436.     glDisable(GL_LIGHTING);
437.     glDisable(GL_TEXTURE_2D);
438.     glViewport(0, 0, 800, 800);
439.
440.     glPushMatrix();
441.     gluLookAt(0.0, 50.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0);
442.     glColor3f(0.0, 0.0, 0.0);
443.     glLineWidth(5.0);
444.
445.     glBegin(GL_LINES);
446.     glVertex3f(-50.0, 0.0, 0.0);
447.     glVertex3f(50.0, 0.0, 0.0);
448.     glVertex3f(0.0, 0.0, -50.0);
449.     glVertex3f(0.0, 0.0, 50.0);
450.     glEnd();
451.
452.     glLineWidth(1.0);
453.     glPopMatrix();
454.
455.     glEnable(GL_LIGHTING);
456. }
457.
458. void specialKeys(int key, int x, int y) {
459.     if (viewport == -1) return;
460.
461.     Camera* c = &cameras[viewport];
462.
463.     if (key == GLUT_KEY_UP) c->theta -= 5.0;
464.     if (key == GLUT_KEY_DOWN) c->theta += 5.0;
465.     if (key == GLUT_KEY_LEFT) c->phi -= 5.0;
```

```
466.     if (key == GLUT_KEY_RIGHT) c->phi += 5.0;
467. 
468.     if (c->theta > 360.0) c->theta = fmod((double)c->theta, 360.0);
469.     if (c->phi > 360.0) c->phi = fmod((double)c->phi, 360.0);
470. 
471.     eyePosition(viewport);
472. }
473. 
474. void keyboard(unsigned char key, int x, int y) {
475.     switch (key) {
476.     case '1':
477.         viewport = 2;
478.         printf("뷰포트 선택: 위\n");
479.         break;
480.     case '2':
481.         viewport = 3;
482.         printf("뷰포트 선택: 옆\n");
483.         break;
484.     case '3':
485.         viewport = 0;
486.         printf("뷰포트 선택: 앞\n");
487.         break;
488.     case '4':
489.         viewport = 1;
490.         printf("뷰포트 선택: 무작위\n");
491.         break;
492.     case 'x':
493.         lightPosition[0] += lightSpeed;
494.         glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
495.         break;
496.     case 'X':
497.         lightPosition[0] -= lightSpeed;
498.         glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
499.         break;
500.     case 'y':
501.         lightPosition[1] += lightSpeed;
502.         glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
503.         break;
504.     case 'Y':
505.         lightPosition[1] -= lightSpeed;
506.         glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
507.         break;
508.     case 'z':
509.         lightPosition[2] -= lightSpeed;
510.         glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
511.         break;
512.     case 'Z':
513.         lightPosition[2] += lightSpeed;
514.         glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
515.         break;
516.     case 'g':
517.     case 'G':
518.         grid = !grid; // 그리드 토글
519.         printf("그리드 %s\n", grid ? "On" : "Off");
520.         break;
521.     case 'f':
522.     case 'F':
523.         if (viewport != -1) {
524.             initCamera(0); // 앞쪽 뷰 설정
525.             cameras[viewport] = cameras[0]; // 현재 뷰포트에 복사
526.             eyePosition(viewport);
527.             printf("카메라: 앞쪽 뷰 (뷰포트 %d)\n", viewport);
528.         }
529.         break;
530.     case 'b':
531.     case 'B':
532.         if (viewport != -1) {
533.             initCamera(0); // 앞쪽 뷰 설정
534.             cameras[viewport] = cameras[0];
```

```
535.            cameras[viewport].phi += 180.0f; // 뒤쪽으로 180도 회전
536.            if (cameras[viewport].phi > 360.0) cameras[viewport].phi = fmod((double)cameras[viewport].phi,
360.0);
537.            eyePosition(viewport);
538.            printf("카메라: 뒤쪽 뷰 (뷰포트 %d)\n", viewport);
539.        }
540.        break;
541.    case 'l':
542.    case 'L':
543.        if (viewport != -1) {
544.            initCamera(3); // 옆쪽 뷰 설정
545.            cameras[viewport] = cameras[3];
546.            eyePosition(viewport);
547.            printf("카메라: 옆쪽 뷰 (뷰포트 %d)\n", viewport);
548.        }
549.        break;
550.    case 'u':
551.    case 'U':
552.        if (viewport != -1) {
553.            initCamera(2); // 위쪽 뷰 설정
554.            cameras[viewport] = cameras[2];
555.            eyePosition(viewport);
556.            printf("카메라: 위쪽 뷰 (뷰포트 %d)\n", viewport);
557.        }
558.        break;
559.    case 'p':
560.    case 'P':
561.        if (viewport != -1) {
562.            initCamera(1); // 무작위 뷰 설정
563.            cameras[viewport] = cameras[1];
564.            eyePosition(viewport);
565.            printf("카메라: 무작위 뷰 (뷰포트 %d)\n", viewport);
566.        }
567.        break;
568.    }
569.    glutPostRedisplay();
570. }
571.
572. void menuCallback(int value) {
573.    switch (value) {
574.    case 0: // 바닥 색상 변경
575.        floorColor[0] = (float)rand() / RAND_MAX;
576.        floorColor[1] = (float)rand() / RAND_MAX;
577.        floorColor[2] = (float)rand() / RAND_MAX;
578.        printf("바닥 색상 변경: RGB(%.2f, %.2f, %.2f)\n", floorColor[0], floorColor[1], floorColor[2]);
579.        break;
580.    case 1: // 모델 색상 변경
581.        modelColor[0] = (float)rand() / RAND_MAX;
582.        modelColor[1] = (float)rand() / RAND_MAX;
583.        modelColor[2] = (float)rand() / RAND_MAX;
584.        printf("모델 색상 변경: RGB(%.2f, %.2f, %.2f)\n", modelColor[0], modelColor[1], modelColor[2]);
585.        break;
586.    case 2: // 조명 토글
587.        light = !light;
588.        if (light) {
589.            glEnable(GL_LIGHT0);
590.            printf("조명 On\n");
591.        }
592.        else {
593.            glDisable(GL_LIGHT0);
594.            printf("조명 Off\n");
595.        }
596.        break;
597.    case 3: // 조명 색상 변경
598.        diffuse[0] = (float)rand() / RAND_MAX;
599.        diffuse[1] = (float)rand() / RAND_MAX;
600.        diffuse[2] = (float)rand() / RAND_MAX;
601.        specular[0] = (float)rand() / RAND_MAX;
```

```
602.         specular[1] = (float)rand() / RAND_MAX;
603.         specular[2] = (float)rand() / RAND_MAX;
604.         glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
605.         glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
606.         printf("조명 색상 변경\n");
607.         break;
608.     case 4: // 텍스처 토글
609.         texture = !texture;
610.         if (texture) {
611.             printf("텍스처 On\n");
612.         }
613.         else {
614.             printf("텍스처 Off\n");
615.         }
616.         break;
617.     case 5: // 애니메이션 토글
618.         animation = !animation;
619.         if (animation) {
620.             printf("애니메이션 On\n");
621.             glutTimerFunc(0, animate, 0);
622.         }
623.         else {
624.             printf("애니메이션 Off\n");
625.         }
626.         break;
627.     }
628.     glutPostRedisplay();
629. }
630.
631. void initTexture(GLuint* texture, const char* path)
632. {
633.     std::vector<unsigned char> image;
634.     unsigned width, height;
635.     unsigned error = lodepng::decode(image, width, height, path);
636.     if (!error)
637.         std::cout << "error " << error << ": " << lodepng_error_text(error) << std::endl;
638.
639.     glGenTextures(1, texture);
640.     glBindTexture(GL_TEXTURE_2D, *texture);
641.     glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0, GL_RGBA, GL_UNSIGNED_BYTE, &image[0]);
642.     glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
643.     glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
644.
645.     glEnable(GL_TEXTURE_2D);
646. }
647.
648. void animate(int value) {
649.     if (animation) {
650.         // -5도 ~ 5도 사이
651.         if (animationAngle > 5.0f) animationDirection = -1;
652.         if (animationAngle < -5.0f) animationDirection = 1;
653.
654.         animationAngle += animationDirection * 0.4f;
655.
656.         glutPostRedisplay();
657.         glutTimerFunc(16, animate, 0);
658.     }
659. }
```

2.2 model.h

```
// 20214045_류재민_FinalProject
#pragma once
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <GL/glut.h>

typedef struct {
    double** vPoint;
    int** fPoint;
    int vNum;
    int fNum;
} Model;

#ifdef __cplusplus
extern "C" {
#endif

    Model ObjLoad(const char* name);
      void calcNormal(double* v0, double* v1, double* v2, double* normal);
    void rendering(Model model, int sort);

    extern Model dino;
    extern GLfloat modelColor[3];
    extern int light;
    extern int texture;
    extern GLuint textureID[12];

#ifdef __cplusplus
}
#endif
```

2.3 model.cpp

```
// 20214045_류재민_FinalProject
#pragma warning(disable:4996)
#include "model.h"

Model ObjLoad(const char* name)
{
    Model model = { 0 };
    char line[256];
    FILE* fp = fopen(name, "r");
    if (!fp)
    {
        printf("파일 열기 실패: %s\n", name);
        exit(1);
    }

    while (fgets(line, sizeof(line), fp))
    {
        if (strncmp(line, "v ", 2) == 0) model.vNum++;
        else if (strncmp(line, "f ", 2) == 0) model.fNum++;
    }
    rewind(fp);

    model.vPoint = (double**)malloc(sizeof(double*) * model.vNum);
    for (int i = 0; i < model.vNum; i++)
        model.vPoint[i] = (double*)malloc(sizeof(double) * 3);

    model.fPoint = (int**)malloc(sizeof(int*) * model.fNum);
    for (int i = 0; i < model.fNum; i++)
        model.fPoint[i] = (int*)malloc(sizeof(int) * 3);

```

```
31.     int vCount = 0, fCount = 0;
32.     while (fgets(line, sizeof(line), fp))
33.     {
34.         if (strncmp(line, "v ", 2) == 0)
35.         {
36.             sscanf(line, "v %lf %lf %lf", &model.vPoint[vCount][0], &model.vPoint[vCount][1],
&model.vPoint[vCount][2]);
37.             vCount++;
38.         }
39.         else if (strncmp(line, "f ", 2) == 0)
40.         {
41.             char v1[32], v2[32], v3[32];
42.             sscanf(line, "f %s %s %s", v1, v2, v3);
43.             sscanf(v1, "%d", &model.fPoint[fCount][0]);
44.             sscanf(v2, "%d", &model.fPoint[fCount][1]);
45.             sscanf(v3, "%d", &model.fPoint[fCount][2]);
46.             model.fPoint[fCount][0]--;
47.             model.fPoint[fCount][1]--;
48.             model.fPoint[fCount][2]--;
49.             fCount++;
50.         }
51.     }
52.     fclose(fp);
53.     printf("MODEL 로드 성공: %d 정점, %d 면\n", model.vNum, model.fNum);
54.     return model;
55. }
56.
57. void calcNormal(double* v0, double* v1, double* v2, double* normal) {
58.     double u[3] = { v1[0] - v0[0], v1[1] - v0[1], v1[2] - v0[2] };
59.     double v[3] = { v2[0] - v0[0], v2[1] - v0[1], v2[2] - v0[2] };
60.     normal[0] = u[1] * v[2] - u[2] * v[1];
61.     normal[1] = u[2] * v[0] - u[0] * v[2];
62.     normal[2] = u[0] * v[1] - u[1] * v[0];
63.     double len = sqrt(normal[0] * normal[0] + normal[1] * normal[1] + normal[2] * normal[2]);
64.     if (len > 0) {
65.         normal[0] /= len; normal[1] /= len; normal[2] /= len;
66.     }
67. }
68.
69. void rendering(Model model, int sort)
70. {
71.      // 모델 색상 설정
72.     if (!light) {
73.         glDisable(GL_LIGHTING);
74.         glColor3f(modelColor[0], modelColor[1], modelColor[2]);
75.     }
76.     else {
77.         glEnable(GL_LIGHTING);
78.         glMaterialfv(GL_FRONT, GL_DIFFUSE, modelColor);
79.     }
80.
81.     if (texture) {
82.         glEnable(GL_TEXTURE_2D);
83.         if(sort == 1) glBindTexture(GL_TEXTURE_2D, textureID[1]);
84.         if(sort == 2) glBindTexture(GL_TEXTURE_2D, textureID[2]);
85.         if(sort == 3) glBindTexture(GL_TEXTURE_2D, textureID[3]);
86.         if(sort == 4) glBindTexture(GL_TEXTURE_2D, textureID[4]);
87.         if(sort == 5) glBindTexture(GL_TEXTURE_2D, textureID[5]);
88.                 if (sort == 6) glBindTexture(GL_TEXTURE_2D, textureID[6]);
89.                 if (sort == 7) glBindTexture(GL_TEXTURE_2D, textureID[7]);
90.                 if (sort == 8) glBindTexture(GL_TEXTURE_2D, textureID[8]);
91.                 if (sort == 9) glBindTexture(GL_TEXTURE_2D, textureID[9]);
92.                 if (sort == 10) glBindTexture(GL_TEXTURE_2D, textureID[10]);
93.                 if (sort == 11) glBindTexture(GL_TEXTURE_2D, textureID[11]);
94.     }
95.     else {
96.         glDisable(GL_TEXTURE_2D);
97.     }
98.
99.     for (int i = 0; i < model.fNum; i++) {
```

```
100.         double* v0 = model.vPoint[model.fPoint[i][0]];
101.         double* v1 = model.vPoint[model.fPoint[i][1]];
102.         double* v2 = model.vPoint[model.fPoint[i][2]];
103.         double normal[3];
104.         calcNormal(v0, v1, v2, normal);
105.
106.         glBegin(GL_TRIANGLES);
107.         glNormal3dv(normal);
108.
109.         for (int j = 0; j < 3; j++) {
110.             if (texture) {
111.                 // 공룡의 위치에 따라 텍스처 좌표 계산
112.                 float u = (model.vPoint[model.fPoint[i][j]][0] + 20.0f) / 40.0f;
113.                 float v = (model.vPoint[model.fPoint[i][j]][2] + 20.0f) / 40.0f;
114.                 glTexCoord2f(u, v);
115.                 glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
116.             }
117.
118.             glVertex3f(model.vPoint[model.fPoint[i][j]][0], model.vPoint[model.fPoint[i][j]][1],
model.vPoint[model.fPoint[i][j]][2]);
119.         }
120.
121.         glEnd();
122.     }
123.
124.     if (texture) {
125.         glDisable(GL_TEXTURE_2D);
126.     }
127. }
```

2.4 camera.h

```
 1. // 20214045_류재민_FinalProject
 2. #pragma once
 3. #include <math.h>
 4. #include <GL/glut.h>
 5.
 6. typedef struct {
 7.     GLfloat eyeX, eyeY, eyeZ;
 8.     GLfloat upX, upY, upZ;
 9.     GLfloat theta, phi, r;
10. } Camera;
11.
12. #ifdef __cplusplus
13. extern "C" {
14. #endif
15.
16.     void eyePosition(int index);
17.     void initCamera(int index);
18.
19.     extern Camera cameras[4];
20.     extern int viewport;
21.
22. #ifdef __cplusplus
23. }
24. #endif
```

2.5 camera.cpp

```
 1. // 20214045_류재민_FinalProject
 2. #pragma warning(disable:4996)
 3. #include "camera.h"
 4.
 5. // gluLookAt 함수에 필요한 카메라 구조체
 6. Camera cameras[4] = {
 7.     {0.0, 0.0, 50.0, 0.0, 1.0, 0.0, 90.0, 0.0, 50.0},    // 앞
 8.     {35.0, 25.0, 35.0, 0.0, 1.0, 0.0, 65.0, 45.0, 50.0}, // 무작위
```

```
    {0.0, 50.0, 0.0, 0.0, 0.0, -1.0, 0.0, 0.0, 50.0},    // 위
    {50.0, 0.0, 0.0, 0.0, 1.0, 0.0, 90.0, 90.0, 50.0}    // 옆
};

void initCamera(int index) {
    switch (index) {
    case 0: // 앞
        cameras[index] = { 0.0, 0.0, 50.0, 0.0, 1.0, 0.0, 90.0, 0.0, 50.0 };
        break;
    case 1: // 무작위
        cameras[index] = { 35.0, 25.0, 35.0, 0.0, 1.0, 0.0, 65.0, 45.0, 50.0 };
        break;
    case 2: // 위
        cameras[index] = { 0.0, 50.0, 0.0, 0.0, 0.0, -1.0, 0.0, 0.0, 50.0 };
        break;
    case 3: // 옆
        cameras[index] = { 50.0, 0.0, 0.0, 0.0, 1.0, 0.0, 90.0, 90.0, 50.0 };
        break;
    }
    eyePosition(index); // 초기 위치로 카메라 설정
}

void eyePosition(int index) {
    Camera* cam = &cameras[index];
    GLfloat dt = 1.0;
    GLfloat eyeXtemp = cam->r * sin((cam->theta + 0.0174532 - dt) * 0.0174532) * sin(cam->phi * 0.0174532);
    GLfloat eyeYtemp = cam->r * cos((cam->theta + 0.0174532 - dt) * 0.0174532);
    GLfloat eyeZtemp = cam->r * sin((cam->theta + 0.0174532 - dt) * 0.0174532) * cos(cam->phi * 0.0174532);

    cam->eyeX = cam->r * sin(cam->theta * 0.0174532) * sin(cam->phi * 0.0174532);
    cam->eyeY = cam->r * cos(cam->theta * 0.0174532);
    cam->eyeZ = cam->r * sin(cam->theta * 0.0174532) * cos(cam->phi * 0.0174532);

    cam->upX = eyeXtemp - cam->eyeX;
    cam->upY = eyeYtemp - cam->eyeY;
    cam->upZ = eyeZtemp - cam->eyeZ;

    glutPostRedisplay();
}
```

2.6 lodepng.h / lodepng.cpp

lodepng.h와 **lodepng.cpp**는 교수님께서 주신 소스코드를 그대로 사용했고 소스코드가 너무 긴 관계로 따로 레포트에 소스코드를 넣지 않겠습니다.

3. 소스코드 분석

3.1 20214045_류재민_FinalProject.cpp

- 주요 변수 및 초기화 (init() 함수)

```
 1. void init() {
 2.     // 모델 불러오기
 3.     dino = ObjLoad("dino.obj");
 4.     eye = ObjLoad("eye.obj");
 5.     eyeground = ObjLoad("eyeground.obj");
 6.     teeth = ObjLoad("teeth.obj");
 7.     horn = ObjLoad("horn.obj");
 8.     rleg = ObjLoad("rleg.obj");
 9.     lleg = ObjLoad("lleg.obj");
10.     legCover = ObjLoad("legCover.obj");
11.     rarm = ObjLoad("rarm.obj");
12.     larm = ObjLoad("larm.obj");
13.     armCover = ObjLoad("armCover.obj");
14.
15.     glClearColor(0.2, 0.2, 0.2, 0.0);
16.     glEnable(GL_DEPTH_TEST);
17.
18.     glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
19.     glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
20.     glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
21.
22.     // 감쇠율
23.     glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 0.001);
24.     glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.0001);
25.
26.     glEnable(GL_LIGHTING);
27.     glEnable(GL_LIGHT0);
28.
29.     // 랜덤 시드 초기화
30.     srand(time(NULL));
31.
32.     // 팝업 메뉴 생성
33.     glutCreateMenu(menuCallback);
34.     glutAddMenuEntry("바닥 색상 변경", 0);
35.     glutAddMenuEntry("모델 색상 변경", 1);
36.     glutAddMenuEntry("조명 On/Off", 2);
37.     glutAddMenuEntry("조명 색상 변경", 3);
38.     glutAddMenuEntry("텍스처 On/Off", 4);
39.     glutAddMenuEntry("애니메이션 On/Off", 5);
40.     glutAttachMenu(GLUT_RIGHT_BUTTON);
41.
42.     // 텍스처 이미지 불러오기
43.     initTexture(&textureID[0], "ground.png"); // 바닥 텍스처
44.     initTexture(&textureID[1], "dino.png");   // 공룡 텍스처
45.     initTexture(&textureID[2], "eye.png");    // 눈 텍스처
46.     initTexture(&textureID[3], "eyeground.png"); // 눈 바닥 텍스처
47.     initTexture(&textureID[4], "teeth.png"); // 이빨 텍스처
48.     initTexture(&textureID[5], "horn.png"); // 뿔 텍스처
49.     initTexture(&textureID[6], "dino.png"); // 오른다리 텍스처
50.     initTexture(&textureID[7], "dino.png"); // 왼다리 텍스처
51.     initTexture(&textureID[8], "dino.png"); // 다리 커버 텍스처
52.     initTexture(&textureID[9], "dino.png"); // 오른팔 텍스처
53.     initTexture(&textureID[10], "dino.png"); // 왼팔 텍스처
54.     initTexture(&textureID[11], "dino.png"); // 팔 커버 텍스처
55. }
```

Model 객체인 **dino, eye, eyeground, teeth, horn, rleg, lleg, legCover, rarm, larm, armCover**는 각각

공룡의 몸통, 눈, 눈 흰자, 이빨, 뿔, 다리 팔 등의 모델입니다. **floorColor**와 **modelColor**는 바닥과 모델의 기본 색상입니다. 기본 색상을 밝게 해서 본래의 텍스처가 변함이 거의 없도록 했습니다. **viewportPosition**은 4개의 뷰 포트를 정의하는 배열입니다. 800*800의 윈도우 창을 400*400의 4개로 나눕니다. **lightPosition**, **diffuse**, **specular**는 조명 위치와 확산광, 반사광 설정을 위한 배열입니다. 앞서 설명했지만 .mtl 파일에 ambient에 대한 정보는 없어서 따로 추가하지 않았습니다. **lightSpeed**는 조명 이동 속도입니다. **textureID**는 바닥이랑 각 모델의 텍스처를 저장하는 배열입니다. 그 외에 애니메이션을 위한 변수들도 존재합니다.

Init() 함수는 처음에 **ObjLoad()** 함수를 사용하여 .obj 들을 가져옵니다. 그 다음 기본 OpenGL 설정을 해주는데 **glEnable(GL_DEPTH_TEST)**로 깊이 테스트를 활성화해서 3D 렌더링을 합니다. 메뉴를 생성해주고 마지막으로 기본 조명을 설정하고 **initTexture()** 함수로 텍스처를 초기화해줍니다.

- display() 함수

```
1. void display() {
2.     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
3.     glShadeModel(GL_SMOOTH);
4.     glEnable(GL_NORMALIZE);
5.
6.     for (int i = 0; i < 4; i++) {
7.         glPushMatrix();
8.
9.         /*
10.        뷰 포트
11.                2 3
12.                0 1
13.        */
14.
15.        glViewport(viewportPosition[i * 4], viewportPosition[i * 4 + 1], viewportPosition[i * 4 + 2],
viewportPosition[i * 4 + 3]);
16.
17.        glMatrixMode(GL_PROJECTION);
18.        glLoadIdentity();
19.        glOrtho(-50, 50, -50, 50, -50.0, 500.0);
20.        glMatrixMode(GL_MODELVIEW);
21.        glLoadIdentity();
22.
23.        gluLookAt(cameras[i].eyeX, cameras[i].eyeY, cameras[i].eyeZ,
24.            0.0, 0.0, 0.0,
25.            cameras[i].upX, cameras[i].upY, cameras[i].upZ);
26.
27.        // 조명 설정 (카메라 뷰 적용 후 조명 위치 설정)
28.        glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
29.
30.        // 조명 그리기
31.        if (light) {
32.            glDisable(GL_LIGHTING);
33.            glPushMatrix();
34.            glTranslatef(lightPosition[0], lightPosition[1], lightPosition[2]);
35.            glColor3f(0.0, 0.0, 0.0);
36.            glutWireCube(2.0);
37.            glPopMatrix();
38.            glEnable(GL_LIGHTING);
39.        }
40.
41.        // 사각형 바닥 그리기
42.        glPushMatrix();
43.        drawRect();
44.        glPopMatrix();
45.
```

```
46.         // 그리드 그리기
47.         if (grid) {
48.             glPushMatrix();
49.             drawGrid();
50.             glPopMatrix();
51.         }
52.
53.         // 모델 그리기
54.         drawModel();
55.
56.         glPopMatrix();
57.     }
58.
59.     drawLine();
60.
61.     glutSwapBuffers();
62. }
```

기본적으로 화면에 4개의 뷰 포트를 렌더링하고 각 뷰 포트에서 공룡, 바닥, 조명, 그리드를 그립니다. **viewportPosition** 배열을 사용하여 4개의 뷰 포트를 설정하고 각 뷰 포트마다 **glViewport()** 함수로 위치와 크기를 지정하고 **glOrtho()** 함수로 직교 투영을 설정합니다. 3번 문제를 위해 **gluLookAt()** 함수로 각 뷰 포트의 카메라 위치를 설정합니다.

light가 1이면 조명을 활성화하고 조명을 **glutWireCube()** 함수로 그려줍니다. **drawRect()** 함수로 바닥을 그려줍니다. **grid**가 1이면 **drawGrid()** 함수로 바닥 바로 위에 그리드를 그려줍니다. **drawModel()** 함수로 공룡을 렌더링 해줍니다. **drawLine()** 함수로 뷰 포트를 구분하기 위한 중앙 십자선을 그려줍니다.

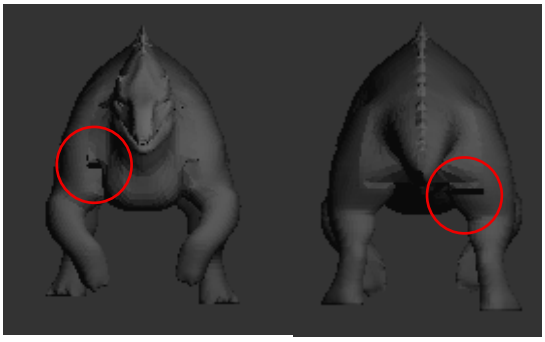
glPushMatrix() 함수와 **glPopMatrix()** 함수를 사용해 각 뷰 포트의 변환 상태를 관리합니다. **glutSwapBuffers()** 함수로 렌더링을 구현합니다.

- drawModel() 함수

```
 1. void drawModel()
 2. {
 3.     glTranslatef(0.0f, -20.0f, -10.0f);
 4.     glScalef(0.7f, 0.7f, 0.7f);
 5.     if (animation) glRotatef(animationAngle, 0.0f, 1.0f, 0.5f); // 회전 애니메이션
 6.
 7.     // 몸통 (dino)
 8.     glPushMatrix();
 9.     if (animation) glTranslatef(0.0f, -baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f), 0.0f);
10.     rendering(dino, 1);
11.     glPopMatrix();
12.
13.     // 눈 (eye)
14.     glPushMatrix();
15.     if (animation) glTranslatef(0.0f, -baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f), 0.0f);
16.     rendering(eye, 2);
17.     glPopMatrix();
18.
19.     // 눈 흰자 (eyeground)
20.     glPushMatrix();
21.     if (animation) glTranslatef(0.0f, -baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f), 0.0f);
22.     rendering(eyeground, 3);
23.     glPopMatrix();
24.
```

```
25.     // 이빨 (teeth)
26.     glPushMatrix();
27.     if (animation) glTranslatef(0.0f, -baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f), 0.0f);
28.     rendering(teeth, 4);
29.     glPopMatrix();
30.
31.     // 뿔 (horn)
32.     glPushMatrix();
33.     if (animation) glTranslatef(0.0f, -baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f), 0.0f);
34.     rendering(horn, 5);
35.     glPopMatrix();
36.
37.     // 커버는 진폭으로 인해 비워지는 공간을 메우기 위해 사용하므로 진폭 애니메이션 X
38.     rendering(legCover, 8);
39.     rendering(armCover, 11);
40.
41.     // 오른쪽 다리(rleg)
42.     glPushMatrix();
43.     if (animation) glTranslatef(0.0f, -(baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f)), 0.0f); // 아래로만 이동
44.     rendering(rleg, 6);
45.     glPopMatrix();
46.
47.     // 왼쪽 다리(lleg) - 반대 위상
48.     glPushMatrix();
49.     if (animation) glTranslatef(0.0f, -(baseAmplitude * (sinf((animationAngle + 180.0f) * 3.14159f /
animationSpeed) * 0.5f + 0.5f)), 0.0f); // 반대 위상
50.     rendering(lleg, 7);
51.     glPopMatrix();
52.
53.     // 오른쪽 팔(rarm)
54.     glPushMatrix();
55.     if (animation) glTranslatef(0.0f, -(baseAmplitude * (sinf((animationAngle + 180.0f) * 3.14159f /
animationSpeed) * 0.5f + 0.5f)), 0.0f); // 반대 위상
56.     rendering(rarm, 9);
57.     glPopMatrix();
58.
59.     // 왼쪽 팔(larm) - 반대 위상
60.     glPushMatrix();
61.     if (animation) glTranslatef(0.0f, -(baseAmplitude * (sinf(animationAngle * 3.14159f / animationSpeed) *
0.5f + 0.5f)), 0.0f); // 아래로만 이동
62.     rendering(larm, 10);
63.     glPopMatrix();
64. }
```

공룡의 각 부분을 렌더링하고 **animation**이 1이면 애니메이션 효과를 적용해줍니다. 공룡을 화면 가운데로 위치하도록 위치와 크기를 조정해줍니다. **rendering()** 함수에 이름과 함께 정수를 매개변수로 보내서 정수로 부위를 구분하고 해당 부위에 맞는 텍스처를 적용합니다. **legCover**와 **armCover**는 애니메이션이 작동할 때 빈 공강을 채우기 위해서 애니메이션을 적용하지 않습니다. 애니메이션은 **baseAmplitude**와 **animationSpeed**를 사용해서 사인 기반의 상하 이동 애니메이션을 구현했습니다.



- drawRect() 함수

```
1. void drawRect() {
2.     glEnable(GL_LIGHTING);
3.
4.     if (!light) {
5.         glDisable(GL_LIGHTING);
6.         glColor3f(floorColor[0], floorColor[1], floorColor[2]);
7.     }
8.     else {
9.         glEnable(GL_LIGHTING);
10.        glMaterialfv(GL_FRONT, GL_DIFFUSE, floorColor);
11.    }
12.
13.    if (texture) {
14.        glEnable(GL_TEXTURE_2D);
15.        glBindTexture(GL_TEXTURE_2D, textureID[0]);
16.        // 텍스처와 조명 조합 모드 설정
17.        glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
18.    }
19.    else {
20.        glDisable(GL_TEXTURE_2D);
21.    }
22.
23.    glBegin(GL_QUADS);
24.    // 법선 벡터 설정 (바닥은 위를 향하므로 (0, 1, 0))
25.    glNormal3f(0.0f, 1.0f, 0.0f);
26.    if (texture) glTexCoord2f(0.0f, 0.0f);
27.    glVertex3f(-25.0f, -21.0f, -45.0f);
28.    if (texture) glTexCoord2f(1.0f, 0.0f);
29.    glVertex3f(25.0f, -21.0f, -45.0f);
30.    if (texture) glTexCoord2f(1.0f, 1.0f);
31.    glVertex3f(25.0f, -21.0f, 40.0f);
32.    if (texture) glTexCoord2f(0.0f, 1.0f);
33.    glVertex3f(-25.0f, -21.0f, 40.0f);
34.    glEnd();
35.
36.    if (texture) {
37.        glDisable(GL_TEXTURE_2D);
38.    }
39. }
```

조명이 꺼져 있으면 **glColor3f()** 함수로 **floorColor**의 색을 입히고 켜져 있으면 **glMaterialfv()** 함수로 재질을 설정합니다. 텍스처가 켜져 있으면 **glBindTexture()** 함수로 "ground.png"를 입히고 **GL_MODULATE** 모드로 조명과 텍스처를 조합해줍니다. **glBegin(GL_QUADS)** 함수를 사용하여 사각형을 그립니다.

- drawGrid() 함수

```
1. void drawGrid() {
2.     glDisable(GL_LIGHTING);
3.     glDisable(GL_TEXTURE_2D);
4.     glLineWidth(1.0f);
5.
6.     // 일반 격자선 (회색)
7.     glColor3f(0.3f, 0.3f, 0.3f);
8.     glBegin(GL_LINES);
9.     for (float x = -50.0f; x <= 50.0f; x += 5.0f) {
10.        if (x != 0.0f) { // 중앙선은 제외
11.                            glVertex3f(x, -20.9f, -50.0f); // 바닥 바로 위에 위치
12.            glVertex3f(x, -20.9f, 50.0f);
13.        }
14.    }
15.    for (float z = -50.0f; z <= 50.0f; z += 5.0f) {
16.        if (z != 0.0f) { // 중앙선은 제외
```

```
            glVertex3f(-50.0f, -20.9f, z);
            glVertex3f(50.0f, -20.9f, z);
        }
    }
    glEnd();

    // 중앙 십자선 (검은색)
    glColor3f(0.0f, 0.0f, 0.0f);
    glBegin(GL_LINES);
    glVertex3f(0.0f, -20.9f, -50.0f);
    glVertex3f(0.0f, -20.9f, 50.0f);
    glVertex3f(-50.0f, -20.9f, 0.0f);
    glVertex3f(50.0f, -20.9f, 0.0f);
    glEnd();

    glEnable(GL_LIGHTING);
}
```

바닥 바로 위에 그리드를 그려줍니다. 조명과 텍스처를 비활성화해서 선만 그려줍니다. 중앙 그리드는 검은색으로 그려줍니다.

- drawLine() 함수

```
void drawLine() {
    glDisable(GL_LIGHTING);
    glViewport(0, 0, 800, 800);

    glPushMatrix();
    gluLookAt(0.0, 50.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0);
    glColor3f(0.0, 0.0, 0.0);
    glLineWidth(5.0);

    glBegin(GL_LINES);
    glVertex3f(-50.0, 0.0, 0.0);
    glVertex3f(50.0, 0.0, 0.0);
    glVertex3f(0.0, 0.0, -50.0);
    glVertex3f(0.0, 0.0, 50.0);
    glEnd();

    glLineWidth(1.0);
    glPopMatrix();

    glEnable(GL_LIGHTING);
}
```

4개의 뷰 포트를 구분해주는 중앙 십자선을 그려줍니다. 마찬가지로 조명과 텍스처를 비활성화해서 그려줍니다.

- specialKeys() 함수

```
void specialKeys(int key, int x, int y) {
    if (viewport == -1) return;

    Camera* c = &cameras[viewport];

    if (key == GLUT_KEY_UP) c->theta -= 5.0;
    if (key == GLUT_KEY_DOWN) c->theta += 5.0;
    if (key == GLUT_KEY_LEFT) c->phi -= 5.0;
    if (key == GLUT_KEY_RIGHT) c->phi += 5.0;

    if (c->theta > 360.0) c->theta = fmod((double)c->theta, 360.0);
    if (c->phi > 360.0) c->phi = fmod((double)c->phi, 360.0);

    eyePosition(viewport);
}
```

1, **2**, **3**, **4**를 통해 선택된 뷰 포트가 없으면 바로 리턴합니다. 선택된 뷰 포트가 있으면 방향키를 사용하여 카메라의 **theta**와 **phi**를 조정해서 상하좌우 이동을 처리해줍니다. **eyePosition()** 함수로 조정된 카메라의 위치를 갱신해줍니다.

- keyboard() 함수

```
1. void keyboard(unsigned char key, int x, int y) {
2.     switch (key) {
3.     case '1':
4.         viewport = 2;
5.         printf("뷰포트 선택: 위\n");
6.         break;
7.     case '2':
8.         viewport = 3;
9.         printf("뷰포트 선택: 옆\n");
10.        break;
11.    case '3':
12.        viewport = 0;
13.        printf("뷰포트 선택: 앞\n");
14.        break;
15.    case '4':
16.        viewport = 1;
17.        printf("뷰포트 선택: 무작위\n");
18.        break;
19.    case 'x':
20.        lightPosition[0] += lightSpeed;
21.        glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
22.        break;
23.    case 'X':
24.        lightPosition[0] -= lightSpeed;
25.        glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
26.        break;
27.    case 'y':
28.        lightPosition[1] += lightSpeed;
29.        glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
30.        break;
31.    case 'Y':
32.        lightPosition[1] -= lightSpeed;
33.        glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
34.        break;
35.    case 'z':
36.        lightPosition[2] -= lightSpeed;
37.        glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
38.        break;
39.    case 'Z':
40.        lightPosition[2] += lightSpeed;
41.        glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
42.        break;
43.    case 'g':
44.    case 'G':
45.        grid = !grid; // 그리드 토글
46.        printf("그리드 %s\n", grid ? "On" : "Off");
47.        break;
48.    case 'f':
49.    case 'F':
50.        if (viewport != -1) {
51.            initCamera(0); // 앞쪽 뷰 설정
52.            cameras[viewport] = cameras[0]; // 현재 뷰포트에 복사
53.            eyePosition(viewport);
54.            printf("카메라: 앞쪽 뷰 (뷰포트 %d)\n", viewport);
```

```
        }
        break;
    case 'b':
    case 'B':
        if (viewport != -1) {
            initCamera(0); // 앞쪽 뷰 설정
            cameras[viewport] = cameras[0];
            cameras[viewport].phi += 180.0f; // 뒤쪽으로 180 도 회전
            if (cameras[viewport].phi > 360.0) cameras[viewport].phi = fmod((double)cameras[viewport].phi,
360.0);
            eyePosition(viewport);
            printf("카메라: 뒤쪽 뷰 (뷰포트 %d)\n", viewport);
        }
        break;
    case 'l':
    case 'L':
        if (viewport != -1) {
            initCamera(3); // 옆쪽 뷰 설정
            cameras[viewport] = cameras[3];
            eyePosition(viewport);
            printf("카메라: 옆쪽 뷰 (뷰포트 %d)\n", viewport);
        }
        break;
    case 'u':
    case 'U':
        if (viewport != -1) {
            initCamera(2); // 위쪽 뷰 설정
            cameras[viewport] = cameras[2];
            eyePosition(viewport);
            printf("카메라: 위쪽 뷰 (뷰포트 %d)\n", viewport);
        }
        break;
    case 'p':
    case 'P':
        if (viewport != -1) {
            initCamera(1); // 무작위 뷰 설정
            cameras[viewport] = cameras[1];
            eyePosition(viewport);
            printf("카메라: 무작위 뷰 (뷰포트 %d)\n", viewport);
        }
        break;
    }
    glutPostRedisplay();
}
```

1, **2**, **3**, **4** 키를 입력하면 뷰 포트를 선택합니다. **x/X**, **y/Y**, **z/Z** 키를 입력하면 조명 위치를 **lightSpeed** 만큼 이동해줍니다. **g/G** 키를 입력하면 그리드를 활성화해줍니다. **f/F**, **b/B**, **l/L**, **u/U**, **p/P** 키를 입력하면 선택된 뷰 포트에서 카메라를 **앞**, **뒤**, **옆**, **위**, **무작위** 뷰로 바로 전환해줍니다.

- menuCallback() 함수

```
void menuCallback(int value) {
    switch (value) {
    case 0: // 바닥 색상 변경
        floorColor[0] = (float)rand() / RAND_MAX;
        floorColor[1] = (float)rand() / RAND_MAX;
        floorColor[2] = (float)rand() / RAND_MAX;
        printf("바닥 색상 변경: RGB(%.2f, %.2f, %.2f)\n", floorColor[0], floorColor[1], floorColor[2]);
        break;
    case 1: // 모델 색상 변경
        modelColor[0] = (float)rand() / RAND_MAX;
        modelColor[1] = (float)rand() / RAND_MAX;
        modelColor[2] = (float)rand() / RAND_MAX;
        printf("모델 색상 변경: RGB(%.2f, %.2f, %.2f)\n", modelColor[0], modelColor[1], modelColor[2]);
        break;
    case 2: // 조명 토글
```

```
        light = !light;
        if (light) {
            glEnable(GL_LIGHT0);
            printf("조명 On\n");
        }
        else {
            glDisable(GL_LIGHT0);
            printf("조명 Off\n");
        }
        break;
    case 3: // 조명 색상 변경
        diffuse[0] = (float)rand() / RAND_MAX;
        diffuse[1] = (float)rand() / RAND_MAX;
        diffuse[2] = (float)rand() / RAND_MAX;
        specular[0] = (float)rand() / RAND_MAX;
        specular[1] = (float)rand() / RAND_MAX;
        specular[2] = (float)rand() / RAND_MAX;
        glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
        glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
        printf("조명 색상 변경\n");
        break;
    case 4: // 텍스처 토글
        texture = !texture;
        if (texture) {
            printf("텍스처 On\n");
        }
        else {
            printf("텍스처 Off\n");
        }
        break;
    case 5: // 애니메이션 토글
        animation = !animation;
        if (animation) {
            printf("애니메이션 On\n");
            glutTimerFunc(0, animate, 0);
        }
        else {
            printf("애니메이션 Off\n");
        }
        break;
    }
    glutPostRedisplay();
}
```

마우스 오른쪽 버튼을 누르면 호출되는 메뉴입니다. **floorColor**를 랜덤 값으로 변경해서 바닥 색상을 변경합니다. **modelColor**를 랜덤 값으로 변경해서 모델 색상을 변경합니다. **light**를 토글해서 조명을 On/Off 합니다. **diffuse**와 **specular**를 랜덤 값으로 변경해서 조명 색상을 변경합니다. **texture**를 토글해서 텍스처를 On/Off 합니다. **animation**을 토글해서 애니메이션을 On/Off 합니다.

- initTexture() 함수

수업 때 실습한 코드를 그대로 가져왔습니다.

- animate() 함수

```
void animate(int value) {
    if (animation) {
        // -5도 ~ 5도 사이
        if (animationAngle > 5.0f) animationDirection = -1;
        if (animationAngle < -5.0f) animationDirection = 1;

        animationAngle += animationDirection * 0.4f;

        glutPostRedisplay();
```

```
10.         glutTimerFunc(16, animate, 0);
11.     }
12. }
```

공룡의 회전과 진폭 애니메이션을 갱신합니다. **animation**이 켜져 있으면 **animationAngle**을 0.4도씩 증가/감소시켜서 -5도와 5도 사이를 반복합니다.

3.2 model.cpp

.obj 파일을 읽어서 저장하는 소스코드로 수업 때 사용한 **OBJ Viewer**를 참고했습니다. 수정되거나 추가된 함수 부분만 설명하겠습니다.

- calcNormal() 함수

```
 1. void calcNormal(double* v0, double* v1, double* v2, double* normal) {
 2.     double u[3] = { v1[0] - v0[0], v1[1] - v0[1], v1[2] - v0[2] };
 3.     double v[3] = { v2[0] - v0[0], v2[1] - v0[1], v2[2] - v0[2] };
 4.     normal[0] = u[1] * v[2] - u[2] * v[1];
 5.     normal[1] = u[2] * v[0] - u[0] * v[2];
 6.     normal[2] = u[0] * v[1] - u[1] * v[0];
 7.     double len = sqrt(normal[0] * normal[0] + normal[1] * normal[1] + normal[2] * normal[2]);
 8.     if (len > 0) {
 9.         normal[0] /= len; normal[1] /= len; normal[2] /= len;
10.     }
11. }
```

삼각형 면의 법선 벡터를 계산하고 조명 효과를 위한 표면 방향을 결정하는 함수입니다. 세 정점의 좌표를 받아서 법선 벡터를 계산합니다. 두 벡터 u와 v를 **u=v1-v0, v=v2-v0**로 계산합니다. **normal=u*v**로 외적을 통해 법선 벡터를 계산합니다. **len=sqrt(normal[0]^2+normal[1]^2+normal[2]^2)**, **normal/=len**으로 법선 벡터를 정규화하고 단위 벡터로 변환합니다. 벡터 길이가 0이면 정규화하지 않습니다. 법선 벡터는 조명 계산에 필요하고 glNormal3dv() 함수에 사용되며 조명 효과를 구현합니다.

- rendering() 함수

```
 1. void rendering(Model model, int sort)
 2. {
 3.       // 모델 색상 설정
 4.     if (!light) {
 5.         glDisable(GL_LIGHTING);
 6.         glColor3f(modelColor[0], modelColor[1], modelColor[2]);
 7.     }
 8.     else {
 9.         glEnable(GL_LIGHTING);
10.         glMaterialfv(GL_FRONT, GL_DIFFUSE, modelColor);
11.     }
12.
13.     if (texture) {
14.         glEnable(GL_TEXTURE_2D);
15.         if(sort == 1) glBindTexture(GL_TEXTURE_2D, textureID[1]);
16.         if(sort == 2) glBindTexture(GL_TEXTURE_2D, textureID[2]);
17.         if(sort == 3) glBindTexture(GL_TEXTURE_2D, textureID[3]);
18.         if(sort == 4) glBindTexture(GL_TEXTURE_2D, textureID[4]);
19.         if(sort == 5) glBindTexture(GL_TEXTURE_2D, textureID[5]);
20.                 if (sort == 6) glBindTexture(GL_TEXTURE_2D, textureID[6]);
21.                 if (sort == 7) glBindTexture(GL_TEXTURE_2D, textureID[7]);
22.                 if (sort == 8) glBindTexture(GL_TEXTURE_2D, textureID[8]);
23.                 if (sort == 9) glBindTexture(GL_TEXTURE_2D, textureID[9]);
24.                 if (sort == 10) glBindTexture(GL_TEXTURE_2D, textureID[10]);
```

```
                if (sort == 11) glBindTexture(GL_TEXTURE_2D, textureID[11]);
    }
    else {
        glDisable(GL_TEXTURE_2D);
    }

    for (int i = 0; i < model.fNum; i++) {
        double* v0 = model.vPoint[model.fPoint[i][0]];
        double* v1 = model.vPoint[model.fPoint[i][1]];
        double* v2 = model.vPoint[model.fPoint[i][2]];
        double normal[3];
        calcNormal(v0, v1, v2, normal);

        glBegin(GL_TRIANGLES);
        glNormal3dv(normal);

        for (int j = 0; j < 3; j++) {
            if (texture) {
                // 공룡의 위치에 따라 텍스처 좌표 계산
                float u = (model.vPoint[model.fPoint[i][j]][0] + 20.0f) / 40.0f;
                float v = (model.vPoint[model.fPoint[i][j]][2] + 20.0f) / 40.0f;
                glTexCoord2f(u, v);
                glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
            }

            glVertex3f(model.vPoint[model.fPoint[i][j]][0], model.vPoint[model.fPoint[i][j]][1],
model.vPoint[model.fPoint[i][j]][2]);
        }

        glEnd();
    }

    if (texture) {
        glDisable(GL_TEXTURE_2D);
    }
}
```

매개변수로 받은 모델을 렌더링하고 조명과 텍스처를 적용해줍니다. 조명이 꺼져 있으면 **glDisable(GL_FIGHTING)** 함수로 조명을 비활성화 하고 **glColor3f()** 함수로 **modelColor**의 색을 적용해줍니다. 켜져 있으면 **glEnable(GL_LIGHTING)** 함수로 조명을 활성화하고 **glMaterialfv()** 함수로 **modelColor**를 재질로 설정해줍니다. 텍스처가 켜져 있으면 **glEnable(GL_TEXTURE_2D)** 함수로 텍스처를 활성화하고 sort의 값에 따라서 적절한 **textureID**를 입혀줍니다. 꺼져 있으면 **glDisable(GL_TEXTURE_2D)** 함수로 텍스처를 비활성화 해줍니다. 세 정점으로 **calcNormal()** 함수를 사용해서 법선 벡터를 계산하고 **glBegin(GL_TRIANGELS)** 함수로 그려줍니다. **glNormal3dv()** 함수로 법선 벡터를 설정하고 텍스처가 켜져 있으면 정점의 x, z 좌표로 텍스처 좌표 u, v를 계산해서 적용해줍니다. **GL_MODULATE** 모드로 조명과 텍스처를 조합해줍니다. **glVertex3f()** 함수로 좌표를 지정해서 삼각형을 그립니다. 텍스처가 켜져 있으면 렌더링 후에 텍스처를 비활성화 해줍니다.

3.3 camera.cpp

- 주요 구조체 및 변수

```
typedef struct {
    GLfloat eyeX, eyeY, eyeZ;
    GLfloat upX, upY, upZ;
    GLfloat theta, phi, r;
} Camera;

// gluLookAt 함수에 필요한 카메라 구조체
Camera cameras[4] = {
```

```
    {0.0, 0.0, 50.0, 0.0, 1.0, 0.0, 90.0, 0.0, 50.0},    // 앞
    {35.0, 25.0, 35.0, 0.0, 1.0, 0.0, 65.0, 45.0, 50.0}, // 무작위
    {0.0, 50.0, 0.0, 0.0, 0.0, -1.0, 0.0, 0.0, 50.0},    // 위
    {50.0, 0.0, 0.0, 0.0, 1.0, 0.0, 90.0, 90.0, 50.0}    // 옆
};
```

Camera 구조체는 카메라의 위치와 방향을 저장합니다. eyeX/eyeY/eyeZ는 카메라의 위치 좌표, upX/upY/upZ는 카메라의 상향 벡터, theta/phi는 수직/수평 회전 각도, r은 카메라와 원점 사이의 거리입니다. camera[4]는 4개의 뷰 포트에 대한 카메라 설정을 저장한 배열입니다. 0은 앞쪽, 1은 무작위, 2는 위쪽, 3은 옆쪽 뷰의 설정을 저장하고 있습니다.

- initCamera() 함수

```
void initCamera(int index) {
    switch (index) {
    case 0: // 앞
        cameras[index] = { 0.0, 0.0, 50.0, 0.0, 1.0, 0.0, 90.0, 0.0, 50.0 };
        break;
    case 1: // 무작위
        cameras[index] = { 35.0, 25.0, 35.0, 0.0, 1.0, 0.0, 65.0, 45.0, 50.0 };
        break;
    case 2: // 위
        cameras[index] = { 0.0, 50.0, 0.0, 0.0, 0.0, -1.0, 0.0, 0.0, 50.0 };
        break;
    case 3: // 옆
        cameras[index] = { 50.0, 0.0, 0.0, 0.0, 1.0, 0.0, 90.0, 90.0, 50.0 };
        break;
    }
    eyePosition(index); // 초기 위치로 카메라 설정
}
```

뷰 포트 인덱스를 받아서 해당 뷰 포트의 카메라를 초기화해줍니다. 추가 기능인 카메라 이동 단축키 기능에서 사용하는 함수입니다.

- eyePosition() 함수

```
void eyePosition(int index) {
    Camera* cam = &cameras[index];
    GLfloat dt = 1.0;
    GLfloat eyeXtemp = cam->r * sin((cam->theta + 0.0174532 - dt) * 0.0174532) * sin(cam->phi * 0.0174532);
    GLfloat eyeYtemp = cam->r * cos((cam->theta + 0.0174532 - dt) * 0.0174532);
    GLfloat eyeZtemp = cam->r * sin((cam->theta + 0.0174532 - dt) * 0.0174532) * cos(cam->phi * 0.0174532);

    cam->eyeX = cam->r * sin(cam->theta * 0.0174532) * sin(cam->phi * 0.0174532);
    cam->eyeY = cam->r * cos(cam->theta * 0.0174532);
    cam->eyeZ = cam->r * sin(cam->theta * 0.0174532) * cos(cam->phi * 0.0174532);

    cam->upX = eyeXtemp - cam->eyeX;
    cam->upY = eyeYtemp - cam->eyeY;
    cam->upZ = eyeZtemp - cam->eyeZ;

    glutPostRedisplay();
}
```

07_4 RGB Color Cube의 **eyePosition()** 함수를 참고했고 구면 좌표계를 사용하여 카메라의 위치와 상향 벡터를 계산하는 함수입니다.

4. 실행화면

- 최초 실행화면


```
E:₩대학₩3학년₩1학기₩컴퓨E
MODEL 로드 성공: 6471 정점, 12596 면
MODEL 로드 성공: 82 정점, 144 면
MODEL 로드 성공: 62 정점, 80 면
MODEL 로드 성공: 798 정점, 756 면
MODEL 로드 성공: 689 정점, 984 면
MODEL 로드 성공: 1551 정점, 3050 면
MODEL 로드 성공: 1551 정점, 3050 면
MODEL 로드 성공: 776 정점, 1380 면
MODEL 로드 성공: 1541 정점, 3024 면
MODEL 로드 성공: 1541 정점, 3024 면
MODEL 로드 성공: 870 정점, 1564 면
error 0: no error, everything went ok
error 0: no error, everything went ok
error 0: no error, everything went ok
error 0: no error, everything went ok
error 0: no error, everything went ok
error 0: no error, everything went ok
error 0: no error, everything went ok
error 0: no error, everything went ok
error 0: no error, everything went ok
error 0: no error, everything went ok
error 0: no error, everything went ok
error 0: no error, everything went ok
```

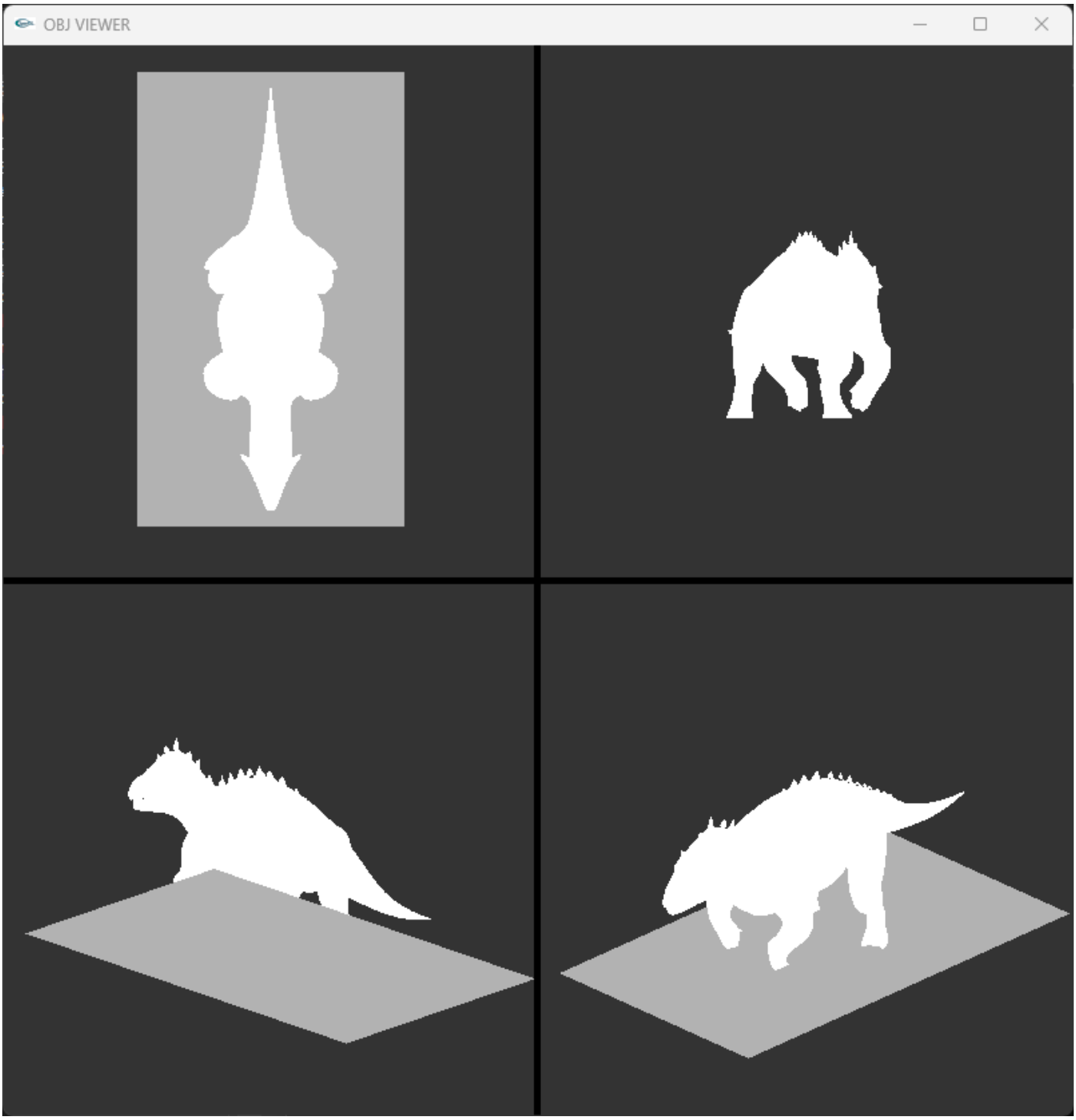
- 3 키 입력으로 뷰 포트(앞)를 선택해서 카메라 이동


```
error 0: no error, everything went ok
error 0: no error, everything went ok
error 0: no error, everything went ok
뷰포트 선택: 앞
```

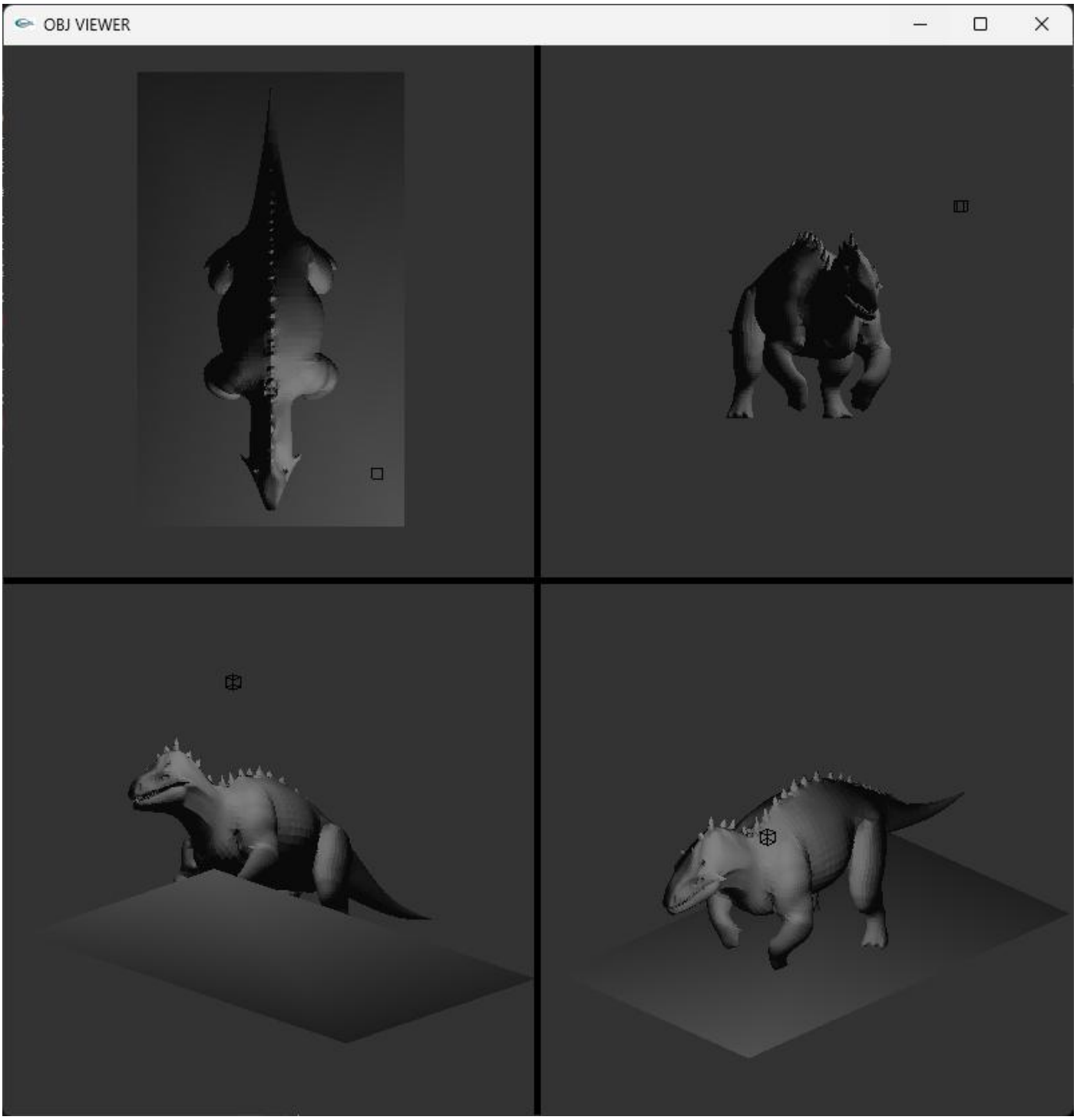
- 2 키 입력으로 뷰 포트(옆)를 선택해서 카메라 이동


```
error 0: no error, everything went ok
error 0: no error, everything went ok
뷰포트 선택: 앞
뷰포트 선택: 옆
```

- 조명 Off, 텍스처 Off



- 조명 On, 텍스처 Off



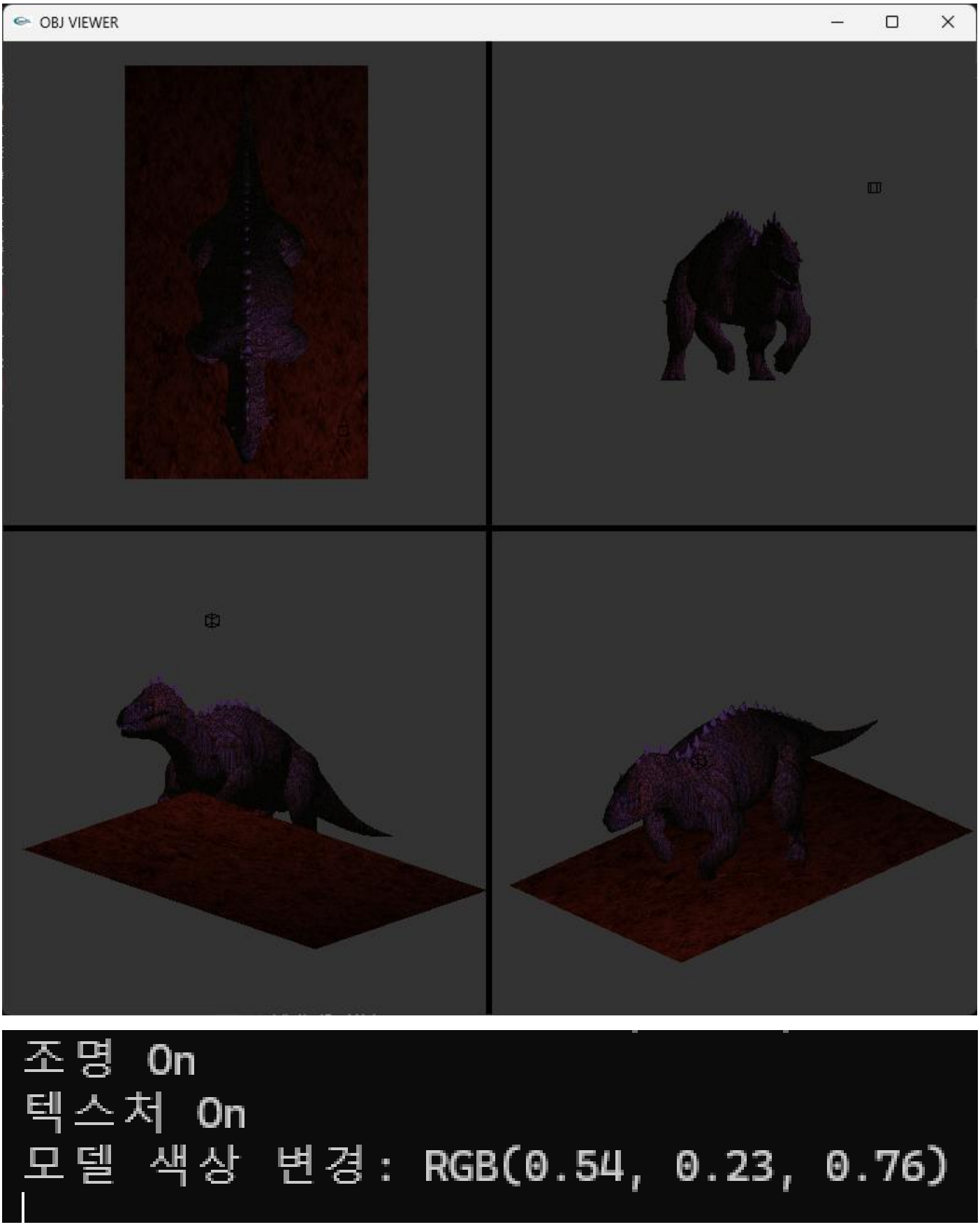
- 조명 Off, 텍스처 On

- 바닥 색상 변경

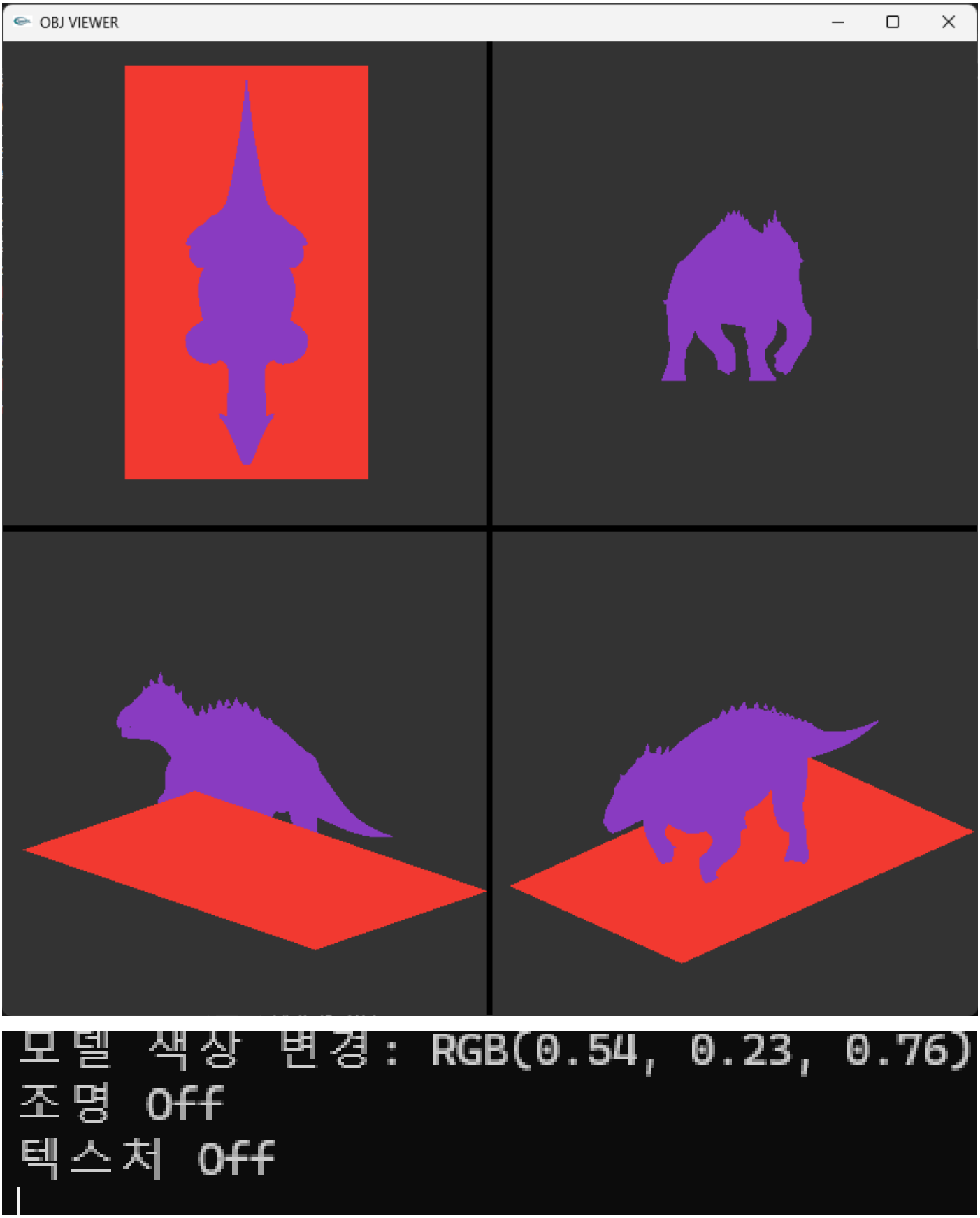

```
조명 On
텍스처 On
바닥 색상 변경: RGB(0.95, 0.22, 0.19)
```

- 바닥 색상 변경, 조명 Off, 텍스처 Off

- 모델 색상 변경

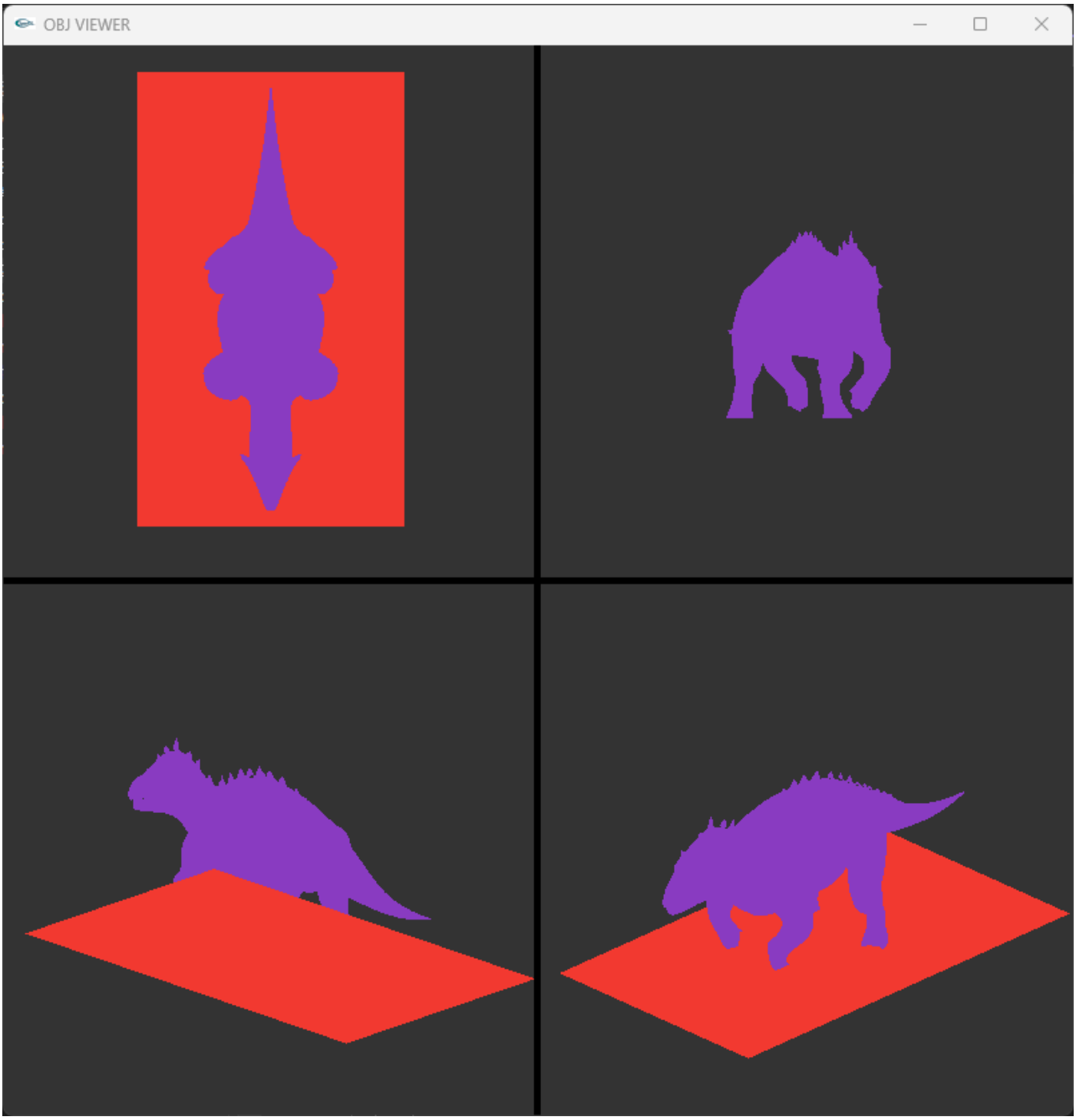


- 모델 색상 변경, 조명 Off, 텍스처 Off



- 조명 색상 변경

- 조명 색상 변경, 조명 Off, 텍스처 Off

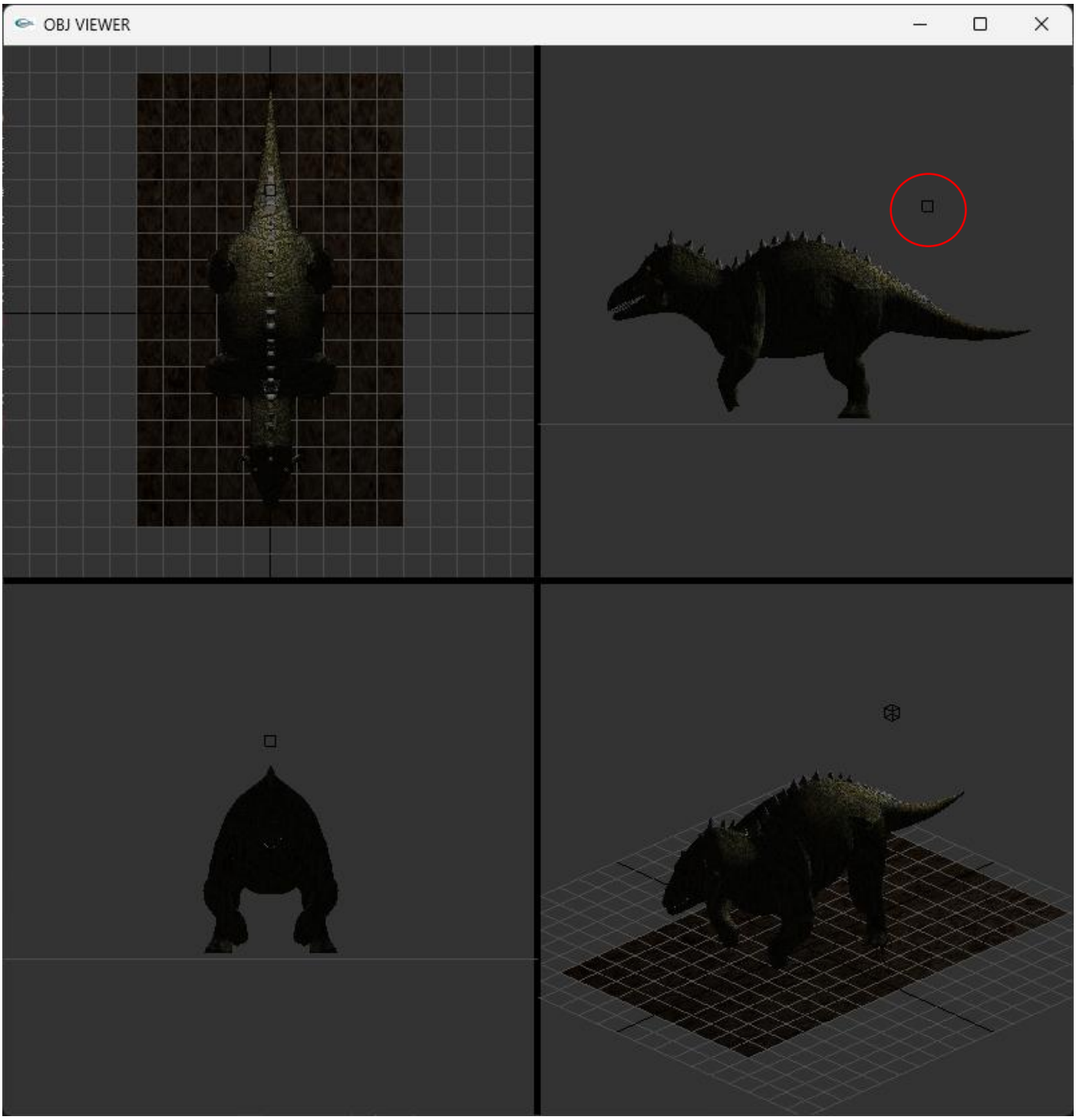


조명의 색상만 변경되었기 때문에 바닥과 모델 색상은 그대로

- 애니메이션 On

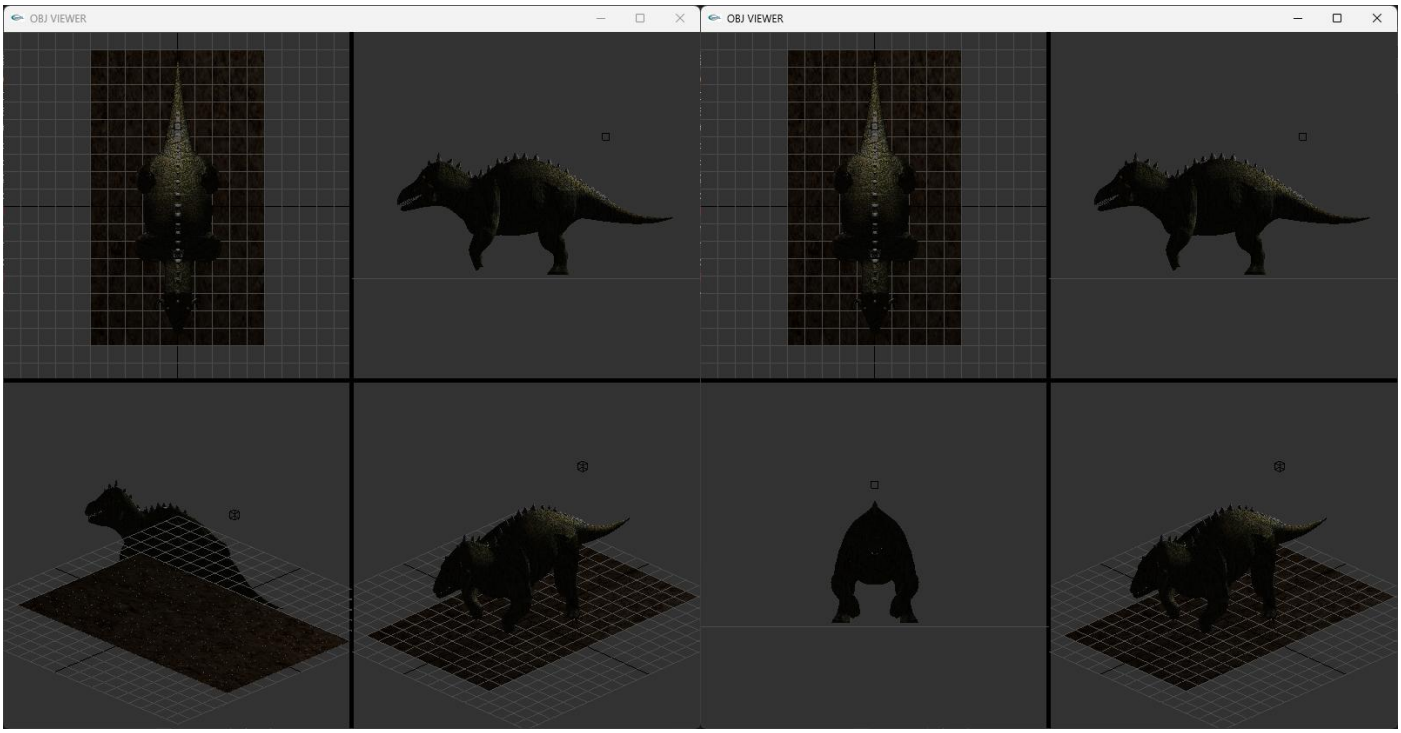
- 그리드 On

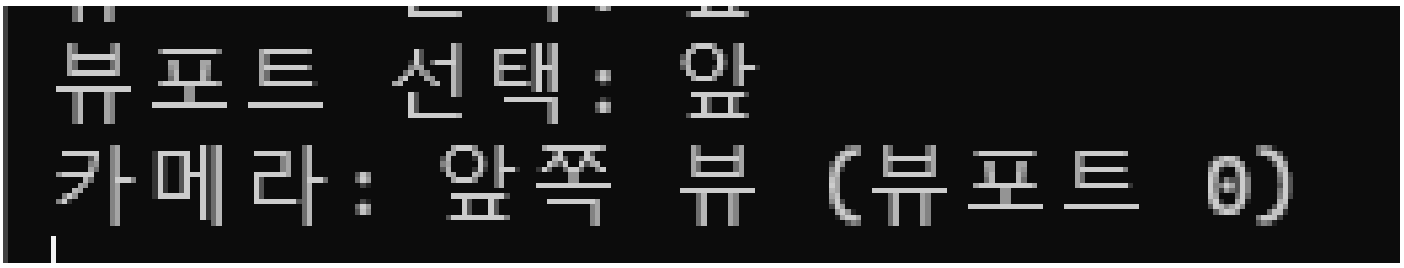
- 조명 이동



조명이 앞쪽에서 뒤쪽으로 가니까 앞쪽은 어두워져서 잘 안보이는 모습

- 3 키 입력으로 뷰 포트(앞)를 선택해서 카메라 이동한 후 F로 앞모습 전환





5. 느낀점

저번 중간과제는 정답이 없고 새로 알고리즘을 생각해서 기능을 구현해야 하는 것들이 있어서 상당히 어려웠는데 이번 기말과제는 이전에 실습한 코드들을 바탕으로 조합을 해 나가면 충분히 다 구현할 수 있는 문제들이어서 좀 더 수월하게 진행할 수 있었습니다. 그래도 조합을 해 나가는 과정에서 모델이 원하는 곳에 렌더링이 안되거나 조명이 이상하거나 모델의 색이 안 바뀌는 등의 시행착오들이 있었는데 오히려 이를 해결해 나가는 과정이 즐거웠고 순차적 코딩을 좀 더 이해할 수 있었습니다. 그저 수업시간에 열심히 집중을 항상 했었고 문제를 보자마자 바로 실습 때 했던 코드들이 생각이 나서 바로 참고할 수 있었습니다. 그래도 중간과제처럼 새로 생각해서 구현하고 싶은 목마름이 있었는데 그걸 보너스 문제가 해결해 주었습니다. 3DS MAX의 기능들을 직접 구현해보니 정말 이런 프로그램들이 편리하구나라는 생각이 들었습니다. 물론 OpenGL도 제공된 라이브러리를 사용하는 것이지만 참 대단하다는 생각이 들었습니다. 학기 초에 오랜 공을 들여서 힘들게 했던 공룡이 이렇게 기말과제를 통해 다시 만나고 사용되는 모습을 보니 괜히 반갑고 뿌듯함을 다시 한번 느낄 수 있었습니다. 3DS MAX에서 CAT 오브젝트를 통해서 구현한 애니메이션을 OpenGL로 구현해보고 싶었는데 상당히 어려웠고 방학동안에 이를 직접 구현해보고 싶습니다. 저는 아직 정확히 진로를 정하지 못했고 게임을 좋아해서 단순히 게임 프로그래밍이나 해볼까? 라는 호기심이 있었고 쉽게 생각했었습니다. 그런데 이번 컴퓨터 그래픽스 강의를 통해 이런 쉽게 생각했던 생각을 다시 고쳐먹을 수 있었습니다. 심지어 생각보다 물리와 같은 수학이 엄청 많이 사용되는 걸 알았고 중간에 벡터가 나왔을 때는 많이 당황했습니다. 뭐든 쉽게 생각해선 안 된다는 걸 느꼈고 진로에 대해 다시 한번 생각해보는 계기가 되었습니다. 여러모로 도움이 많이 되었던 과제였고 강의였습니다.