The Solar System Animation

Term Project #2 Final Presentation

과목명 : 컴퓨터그래픽스

교수명 : 김병철 교수님

발표자 : 92015946 허찬

발표일 : 2024-06-17

JBU 중부대학교

목 차

중부대학교

한 학기동안 제작한 태양계 모사 애니메이션 발표

개발과정

- 행성 그리기
- 공전과 자전
- 텍스처
- 조명
- 조작 기능

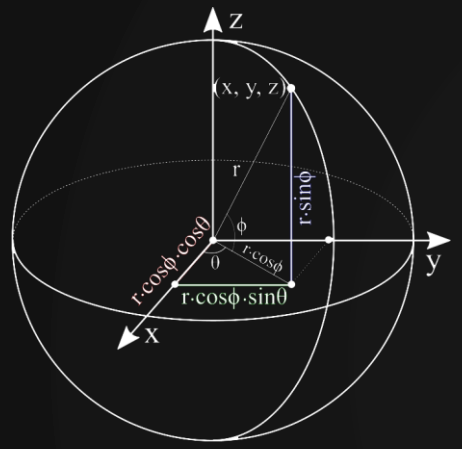
[1] 행성 그리기

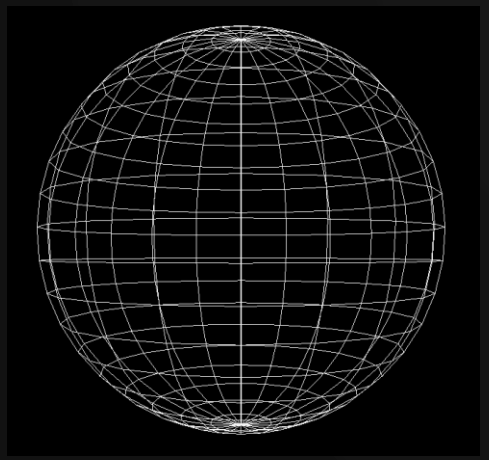
- Planet.h 정의
- 각 행성 객체로 생성. Ex) Solar(radius, position, rotate, revo)
- 객체의 drawPlanet() 함수

Planet.h

```
class Planet {
public :
    Planet(float _radius, float _position)
        : radius(_radius), position(_position) {};
    void drawPlanet();
    float radius;
    float position;
    float rot;
    float revo;
};
```

[1] 행성 그리기





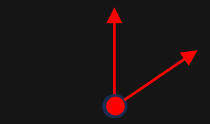
```
//원그리는 변수_
float pi = M_PI;
float u = 0.05;
float v = 0.05;
float db = u * 2 * pi;
float da = v * pi;
```

```
for (float i = 0; i < 1.0; i += u) //horizonal
    for (float j = 0; j < 1.0; j += v) //vertical
    {
        float b = i * 2 * pi;
        float a = (j - 0.5) * pi;

        //1
        glVertex3f(
            r * cos(a) * cos(b),
            r * sin(a),
            r * cos(a) * sin(b));
        //2
        glVertex3f(
            r * cos(a) * cos(b + db),
            r * sin(a),
            r * cos(a) * sin(b + db));
        //3
        glVertex3f(
            r * cos(a + da) * cos(b + db),
            r * sin(a + da),
            r * cos(a + da) * sin(b + db));
        //3
        glVertex3f(
            r * cos(a + da) * cos(b),
            r * sin(a + da),
            r * cos(a + da) * sin(b));

        glEnd();
    }
```

[2] 공전과 자전

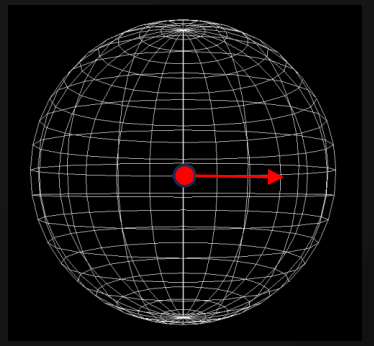
- 기존의 행렬을 스택에 저장
- Rotate -〉 translate -〉 Rotate
- 구 그리기 후 스택 최상단 변환행렬 삭제

```
glPushMatrix();

glRotatef(revo, 0, 1.f, 0);
glTranslatef(0,0,position);
glRotatef(rot, 0, 1.f, 0);

//구 그리기

glPopMatrix();
```



↻Rotate

→Translate

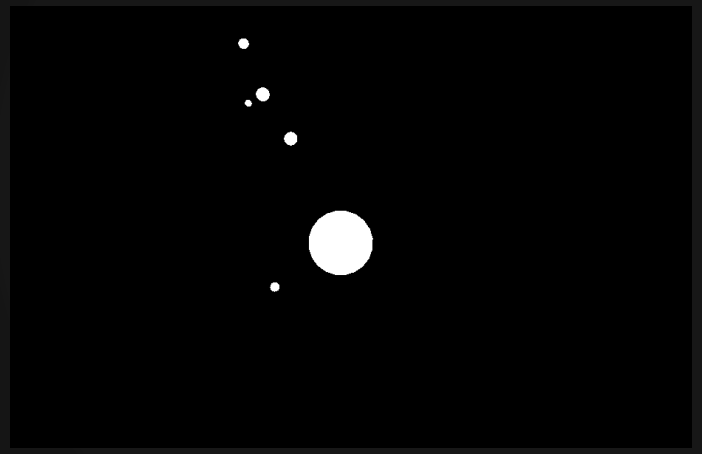


DrawSphere()

[2] 공전과 자전

- OnTimer 이벤트
- 행성들의 값 증가
- WorldSpeed 에 따라 속도가 달라짐

```
SetTimer(1, 1000 / 60, NULL);
```



```
void CMFCWithOpenGLView::OnTimer(UINT_PTR nIDEvent)
{
        //공전
        Mercury.revo += 1.f / 88 * WorldSpeed;
        if (Mercury.revo > 360.f) { Mercury.revo = 0; }

        Venus.revo += 1.f / 225 * WorldSpeed;
        if (Venus.revo > 360.f) { Venus.revo = 0; }

        Earth.revo += 1.f / 365 * WorldSpeed;
        if (Earth.revo > 360.f) { Earth.revo = 0; }

        Moon.revo += 1.f / 27.32 * WorldSpeed;
        if (Moon.revo > 360.f) { Moon.revo = 0; }

        Mars.revo += 1.f / 687 * WorldSpeed;
        if (Mars.revo > 360.f) { Mars.revo = 0; }

        //자전
        Solar.rot += 1.f / 25 * WorldSpeed;
        if (Solar.rot > 360.f) { Solar.rot = 0; }

        Mercury.rot += 1.f / 58 * WorldSpeed;
        if (Mercury.rot > 360.f) { Mercury.rot = 0; }

        Venus.rot -= 1.f / 243 * WorldSpeed;
        if (Venus.rot < -360.f) { Venus.rot = 0; }

        Earth.rot += 1.f * WorldSpeed;
        if (Earth.rot > 360.f) { Earth.rot = 0; }

        Moon.rot += 1.f / 27.32 * WorldSpeed;
        if (Moon.rot > 360.f) { Moon.rot = 0; }

        Mars.rot += 1.3f * WorldSpeed;
        if (Mars.rot > 360.f) { Mars.rot = 0; }

        DrawGLScene();
}
```

[3] 텍스처

- #include <cstdio>
- fread(), fopen()
- Gluint textureID

```
//Texture
GLuint solarImg = 0;
GLuint mercuryImg = 0;
GLuint venusImg = 0;
GLuint earthImg = 0;
GLuint moonImg = 0;
GLuint marsImg = 0;
GLuint spaceImg = 0;
```

```
//텍스쳐 로딩
spaceImg = LoadBMP("space.bmp");
solarImg = LoadBMP("solar.bmp");
mercuryImg = LoadBMP("mercury.bmp");
venusImg = LoadBMP("venus.bmp");
earthImg = LoadBMP("earth.bmp");
moonImg = LoadBMP("moon.bmp");
marsImg = LoadBMP("mars.bmp");
```

```
GLuint LoadBMP(const char* imagepath) {

    GLuint textureID;
    unsigned char header[54];
    unsigned int width, height;
    unsigned int imageSize;   // = 너비*높이*3
    // 실제 RGB 데이터
    unsigned char* data;

    FILE* file = fopen(imagepath, "rb"); // 전달된 파일 경로 사용
    if (!file) {
        printf("파일을 열 수 없습니다: %s\n", imagepath);
        return 0;
    }

    // 파일 헤더로부터 너비와 높이 추출
    fread(header, 1, 54, file);
    width = *(int*)&(header[0x12]);
    height = *(int*)&(header[0x16]);
    imageSize = width * height * 3;

    data = new unsigned char[imageSize];

    // 파일에서 데이터를 버퍼로 읽기
    fread(data, imageSize,1, file);

    // 파일 닫기
    fclose(file);

    glGenTextures(1, &textureID);
    glBindTexture(GL_TEXTURE_2D, textureID);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_BGR_EXT, GL_UNSIGNED_BYTE, data);
    glBindTexture(GL_TEXTURE_2D, 0);

    return textureID;
}
```

[3] 텍스처

- DrawGLScene()에서 텍스처 바인딩
- drawPlanet() 텍스처 추가
- 텍스처 좌표의 u, v = i, j

```
void CMFCWithOpenGLView::DrawGLScene(void) {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    glEnable(GL_TEXTURE_2D);

    glBindTexture(GL_TEXTURE_2D, spaceImg);
    DrawSpace();

    glBindTexture(GL_TEXTURE_2D, solarImg);
    Solar.drawPlanet();
    glPopMatrix();
```

```
for (float i = 0; i < 1.0; i += u) //horizonal
for (float j = 0; j < 1.0; j += v) //vertical
{

    float b = i * 2 * pi;      //0     to  2pi
    float a = (j - 0.5) * pi;  //-pi/2 to pi/2

    /*first*/
    glTexCoord2f(i, j);
    glVertex3f(
        r * cos(a) * cos(b),
        r * sin(a),
        r * cos(a) * sin(b));
    //Second
    glTexCoord2f(i + u, j);
    glVertex3f(
        r * cos(a) * cos(b + db),
        r * sin(a),
        r * cos(a) * sin(b + db));

    //third
    glTexCoord2f(i + u, j + v);
    glVertex3f(
        r * cos(a + da) * cos(b + db),
        r * sin(a + da),
        r * cos(a + da) * sin(b + db));

    //fourth
    glTexCoord2f(i, j + v);
    glVertex3f(
        r * cos(a + da) * cos(b),
        r * sin(a + da),
        r * cos(a + da) * sin(b));

    glEnd();

}
```

[3] 텍스처

[4] 조명

- Phong reflection model.
- DrawGLScene()에서 Lighting() 호출
- 조명이 태양에 위치

```
//Light
GLfloat AmbientLightValue[] = { 0.6f, 0.6f, 0.6f, 1 };
GLfloat DiffuseLightValue[] = { 1, 1, 1, 1 };
GLfloat SpecularLightValue[] = { 1, 1, 1, 1 };
GLfloat PositionLightValue[] = { 0,0,0,1 };

void Lighting() {

    glEnable(GL_LIGHTING); //조명을 사용
    glEnable(GL_LIGHT0);

    glLightfv(GL_LIGHT0, GL_AMBIENT, AmbientLightValue);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, DiffuseLightValue);
    glLightfv(GL_LIGHT0, GL_SPECULAR, SpecularLightValue);
    glLightfv(GL_LIGHT0, GL_POSITION, PositionLightValue);
}
```

[4] 조명

- drawPlanet() 노말 추가
- 조명이 태양 안에 있어 태양만 노말 벡터를 뒤집음

```
int minus=1;
if(this->radius == 1) minus = -1;

for (float i = 0; i < 1.0; i += u) //horizonal
    for (float j = 0; j < 1.0; j += v) //vertical
    {
        float b = i * 2 * pi;      //0     to  2pi
        float a = (j - 0.5) * pi;  //-pi/2 to pi/2

        //normal
        glNormal3f(
            minus * cos(a) * cos(b),
            minus * sin(a),
            minus * cos(a) * sin(b));
        /*first*/
        glTexCoord2f(i, j);
        glVertex3f(
            r * cos(a) * cos(b),
            r * sin(a),
            r * cos(a) * sin(b));

        //normal
        glNormal3f(
            minus * cos(a) * cos(b + db),
            minus * sin(a),
            minus * cos(a) * sin(b + db));
        //Second
        glTexCoord2f(i + u, j);
        glVertex3f(
            r * cos(a) * cos(b + db),
            r * sin(a),
            r * cos(a) * sin(b + db));

        //normal
        glNormal3f(
            minus * cos(a + da) * cos(b + db),
            minus * sin(a + da),
            minus * cos(a + da) * sin(b + db));
        //third
        glTexCoord2f(i + u, j + v);
        glVertex3f(
            r * cos(a + da) * cos(b + db),
            r * sin(a + da),
            r * cos(a + da) * sin(b + db));

        //normal
        glNormal3f(
            minus * cos(a + da) * cos(b),
            minus * sin(a + da),
            minus * cos(a + da) * sin(b));
        //fourth
        glTexCoord2f(i, j + v);
        glVertex3f(
            r * cos(a + da) * cos(b),
            r * sin(a + da),
            r * cos(a + da) * sin(b));

        glEnd();

    }
}
```

[4] 조명

[5] 조작기능 : 카메라

– 카메라 회전

```
float camX = 0, camY = 0
```

```
CPoint prevPoint;
void CMFCWithOpenGLView::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.

    if (isClick) {
        float xChange = point.x - prevPoint.x;
        float yChange = point.y - prevPoint.y;

        camY += 0.01f * yChange;
        camX += 0.01f * xChange;
        prevPoint = point;
    }

    CView::OnMouseMove(nFlags, point);
}
```

```
void CMFCWithOpenGLView::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.
    isClick = true;
    prevPoint = point;
    CView::OnLButtonDown(nFlags, point);

}

void CMFCWithOpenGLView::OnLButtonUp(UINT nFlags, CPoint point)
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.
    isClick = false;
    CView::OnLButtonUp(nFlags, point);

}
```

[5] 조작기능 : 카메라

- 카메라 줌

```
zoom = 10;
```

```
BOOL CMFCWithOpenGLView::OnMouseWheel(UINT nFlags, short zDelta, CPoint pt)
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.
    if (zDelta > 0) {
        zoom -= 1.f;
    }
    else { zoom += 1.f; }

    return CView::OnMouseWheel(nFlags, zDelta, pt);
}
```

[5] 조작기능 : 카메라

- 카메라 변경 값 적용 , DrawGLScene()에서 호출

```
CameraRot(camX, camY, zoom);
```

```
void CameraRot(double xAngle, double yAngle, double zoom) {
    double xx = zoom * cos(yAngle) * cos(xAngle);
    double yy = zoom * sin(yAngle);
    double zz = zoom * cos(yAngle) * sin(xAngle);
    int yoption = 1;

    if ((cos(yAngle) > 0 && sin(xAngle)) < 0 ||
        (cos(yAngle) < 0 && sin(xAngle)) > 0) {
        yoption = -1;
    }

    gluLookAt(xx, yy, zz, 0, 0, 0, 0, yoption, 0);
}
```

[5] 조작기능 : 모드 변경

- F1~F4로 Viewmode 및 Light 변경

```
enum ViewMode {Point,Lines, polygon};
bool isLight= true;
```

```
switch (viewmode) {
case Point: glBegin(GL_POINTS);
    break;
case Lines: glBegin(GL_LINE_STRIP);
    break;
case polygon: glBegin(GL_POLYGON);
    break;
}
```

```
void CMFCWithOpenGLView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.
    CWnd::OnKeyDown(nChar, nRepCnt, nFlags);

    // 상하좌우 키를 구분해 메시지를 출력합니다.
    switch (nChar) {

    case VK_F1:
        viewmode = Point;
        break;
    case VK_F2:
        viewmode = Lines;
        break;
    case VK_F3:
        viewmode = polygon;
        break;
    case VK_F4:
        isLight = !isLight;
        break;
```

[5] 조작기능 : 모드 변경

[5] 조작기능 : 속도 조절

- 기본값 : pauseSpeed = 0, WorldSpeed=15
- 키코드 80(P) 입력 시 WorldSpeed ⇔ pauseSpeed
- 키보드 위, 아래 = WorldSpeed +3, -3
- 0보다 클 때만 작동

```
case 80:
    if (pauseSpeed == 0) {
        pauseSpeed = WorldSpeed;
        WorldSpeed = 0;
    }
    else {
        WorldSpeed = pauseSpeed;
        pauseSpeed=0;
    }
    break;

case VK_UP:
    if (pauseSpeed != 0) return;

    WorldSpeed += 3;
    break;
case VK_DOWN:
    if (pauseSpeed != 0) return;

    if (WorldSpeed > 0)
    WorldSpeed -= 3;
    break;
}
```

[5] 조작기능 : 속도 조절

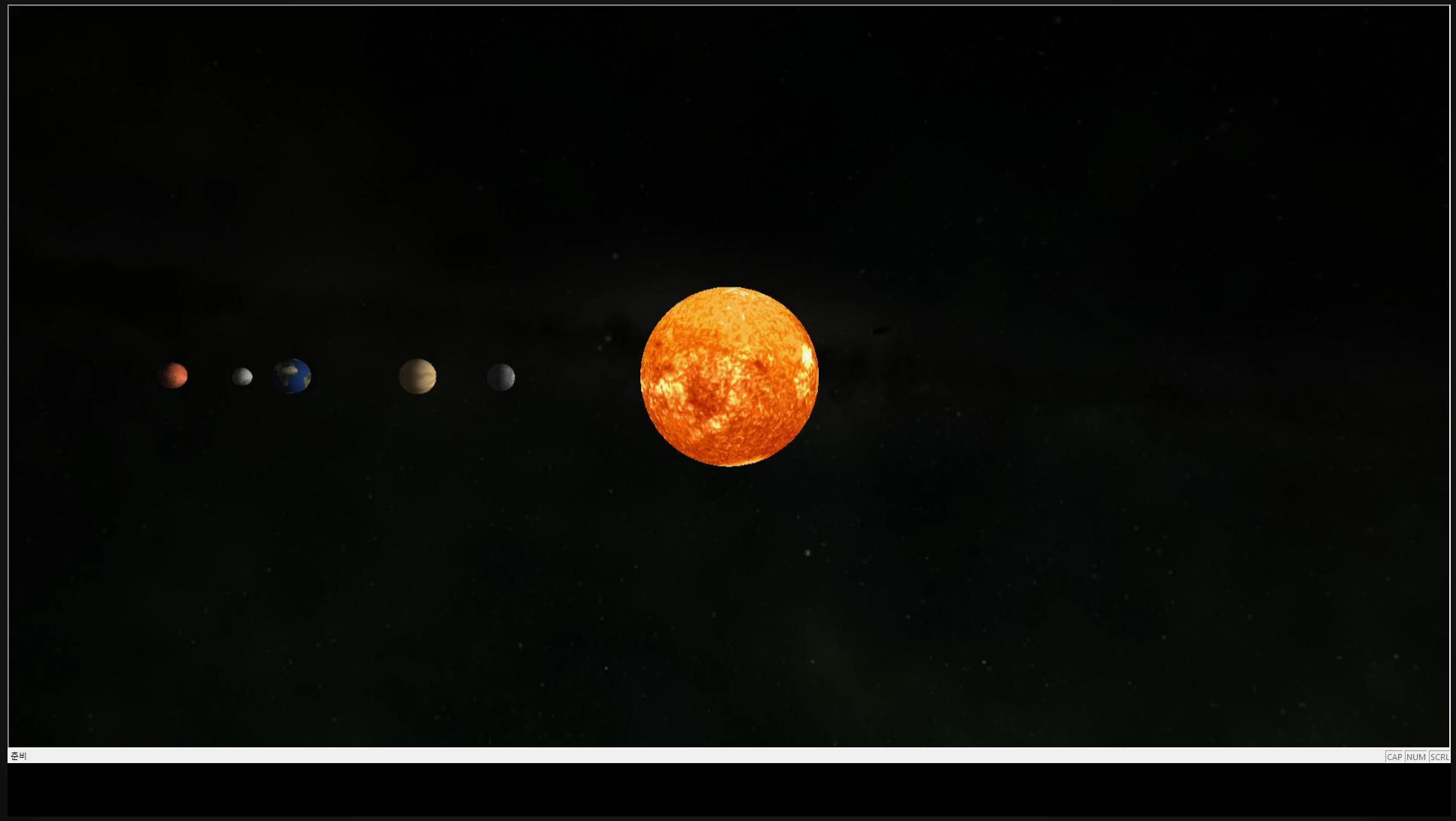
[*] DrawGLScene()

```
void CMFCWithOpenGLView::DrawGLScene(void) {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    CameraRot(camX, camY, zoom);

    Lighting();
    if (!isLight) { glDisable(GL_LIGHTING); }

    glEnable(GL_TEXTURE_2D);

    glBindTexture(GL_TEXTURE_2D, spaceImg);
    DrawSpace();

    glBindTexture(GL_TEXTURE_2D, solarImg);
    Solar.drawPlanet();
    glPopMatrix();

    glBindTexture(GL_TEXTURE_2D, mercuryImg);
    Mercury.drawPlanet();
    glPopMatrix();

    glBindTexture(GL_TEXTURE_2D, venusImg);
    Venus.drawPlanet();
    glPopMatrix();

    glBindTexture(GL_TEXTURE_2D, earthImg);
    Earth.drawPlanet();
    glRotatef(-Earth.rot, 0, 1.f, 0);

    glBindTexture(GL_TEXTURE_2D, moonImg);
    Moon.drawPlanet();
    glPopMatrix();
    glPopMatrix();

    glBindTexture(GL_TEXTURE_2D, marsImg);
    Mars.drawPlanet();
    glPopMatrix();

    SwapBuffers(m_hDC);
}
```

[*] drawPlanet()

```
void Planet::drawPlanet()

{
    glPushMatrix();

    glRotatef(revo, 0, 1.f, 0);
    glTranslatef(0,0,position);
    glRotatef(rot, 0, 1.f, 0);

    float r = radius;

    int minus=1;
    if(this->radius == 1) minus = -1;

    for (float i = 0; i < 1.0; i += u) //horizonal
        for (float j = 0; j < 1.0; j += v) //vertical
        {

            float b = i * 2 * pi;      //0     to  2pi
            float a = (j - 0.5) * pi;  //-pi/2 to pi/2

            switch (viewmode) {
            case Point: glBegin(GL_POINTS);
                break;
            case Lines: glBegin(GL_LINE_STRIP);
                break;
            case polygon: glBegin(GL_POLYGON);
                break;
            }
            //normal
            glNormal3f(
                minus * cos(a) * cos(b),
                minus * sin(a),
                minus * cos(a) * sin(b));
            /*first*/
            glTexCoord2f(i, j);
            glVertex3f(
                r * cos(a) * cos(b),
                r * sin(a),
                r * cos(a) * sin(b));
            //normal
            glNormal3f(
                minus * cos(a) * cos(b + db),
                minus * sin(a),
                minus * cos(a) * sin(b + db));
            //Second
            glTexCoord2f(i + u, j);
            glVertex3f(
                r * cos(a) * cos(b + db),
                r * sin(a),
                r * cos(a) * sin(b + db));

            //normal
            glNormal3f(
                minus * cos(a + da) * cos(b + db),
                minus * sin(a + da),
                minus * cos(a + da) * sin(b + db));
            //third
            glTexCoord2f(i + u, j + v);
            glVertex3f(
                r * cos(a + da) * cos(b + db),
                r * sin(a + da),
                r * cos(a + da) * sin(b + db));

            //normal
            glNormal3f(
                minus * cos(a + da) * cos(b),
                minus * sin(a + da),
                minus * cos(a + da) * sin(b));
            //fourth
            glTexCoord2f(i, j + v);
            glVertex3f(
                r * cos(a + da) * cos(b),
                r * sin(a + da),
                r * cos(a + da) * sin(b));

            glEnd();

        }
}
```

· 시 연



결 론

태양계 모사 애니메이션 제작 완료.
학습한 내용을 토대로 OpenGL 제작 실습 예정.

감사합니다.

JBU 중부대학교