PROJECT REPORT

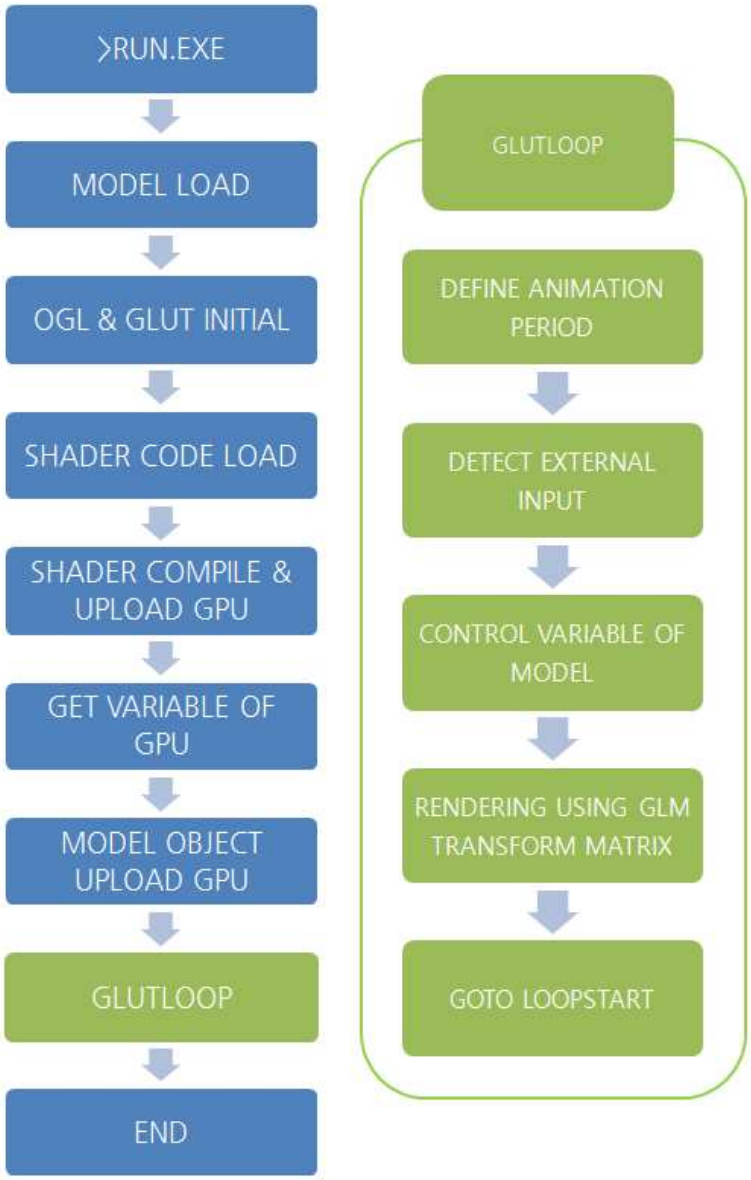
과목	COMPUTER APPLIED DESIGN	성명	Amylose
제목	RADIAL ENGINE SIMULATOR		
목표	SIMULATE RADIAL ENGINE, USING KINEMATIC KNOWLEDGE AND OPENGL		

BRIEF DESCRIPTION OF PROGRAM

I. 개요 및 특징

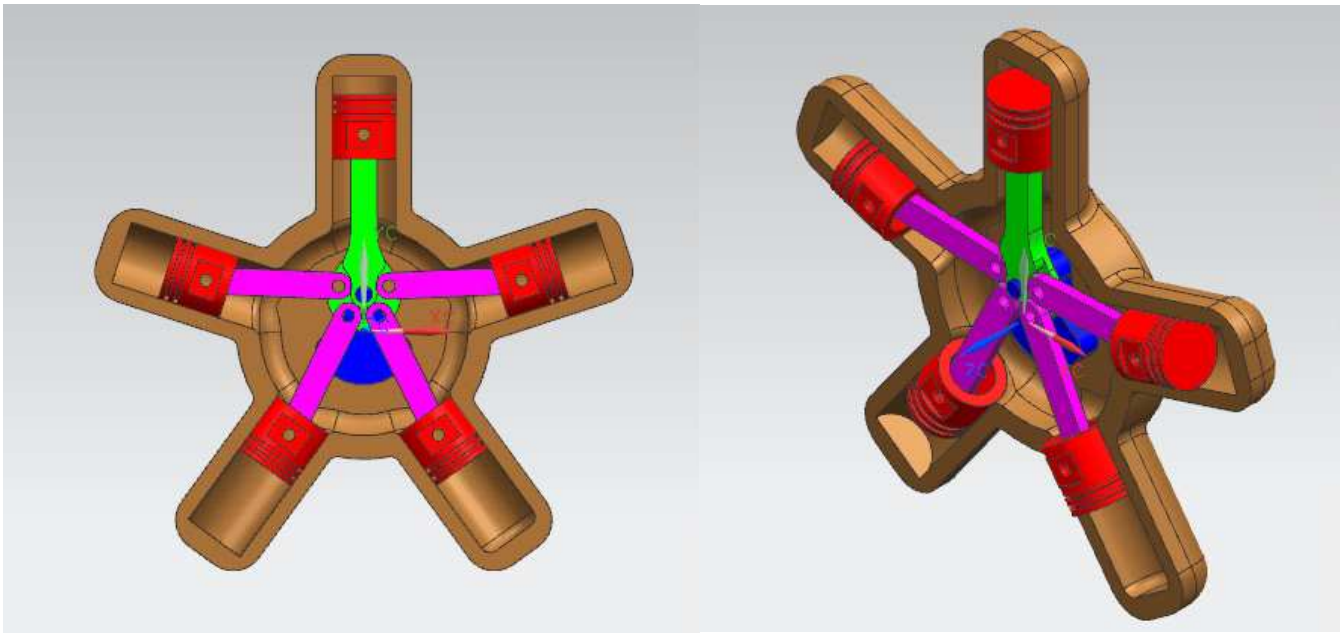
- "*.STL" 파일을 적용, openGL에서 따로 모델링 할 필요 없이 로드하여 활용하였다.
- 기본적으로 마우스 입력을 통하여 시점을 제어할 수 있다. 특정 시점 초기화 방법은 F8키(정면뷰) END키(ISO뷰)를 통하여 할 수 있다.
- 키보드 방향키를 입력하여 회전수를 제어할 수 있고, 애니메이션 시작/정지를 기능키(F1/F2)를 통해 제어 할 수 있다. 또한 F4를 통해 애니메이션을 초기화 할 수 있다.

II. 프로그램 흐름도

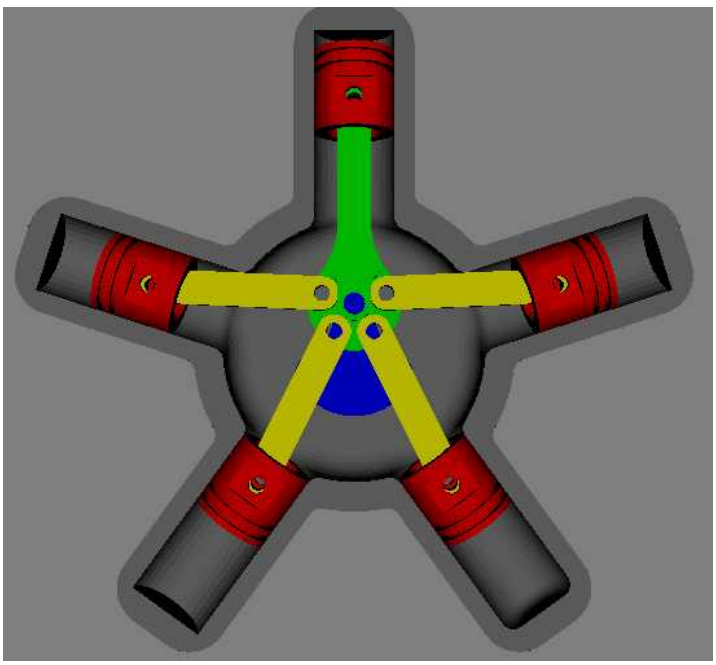


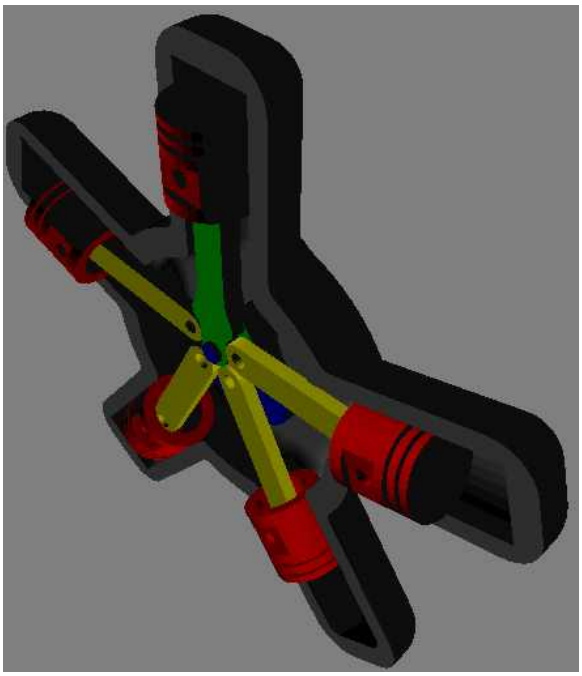
PROGRAM PREVIEW AND REAL PROGRAM

<PROGRAM PREVIEW(UG NX 7.0 MODEL VIEW)>



<REAL PROGRAM(OPENGL REAL EXECUTION)>





FORMULA APPROACH

Basic parameter calculation

$\omega = (6 * rpm * adp)/1000$ Where, ω is degree speed per ms
rpm is rate per minute
adp is animation display period

$sl \mathrel{+}= \omega$ Where, sl is the model crack shaft location

Calculation for Master Rod

$x_0 = 17.5*(\sin(r(-sl)))$
$y_0 = 17.5*(\cos(r(-sl)))$
$z_0 = \theta*(\arcsin(x_0/75))$

Where, x_0 is the model master rod x location
y_0 is the model master rod y location
z_0 is the model master rod z rotation location
r is radians
θ is the degrees

Calculation for 1st Slave Rod

$x_1 = 17.5*\sin(r(-sl))+12.5*\sin(r(-z_{1o}-z_0))$
$y_1 = 17.5*\cos(r(-sl))+12.5*\cos(r(-z_{1o}-z_0))$
$x_{1p} = \cos(r(z_{1o}))*x_1+\sin(r(z_{1o}))*y_1$
$y_{1p} = -\sin(r(z_{1o}))*x_1+\cos(r(z_{1o}))*y_1$
$z_{1a} = \theta*(\arcsin(x_{1p}/l))$
$z_1 = z_{1a}+ z_{1o}$

Where, x_1 is the model 1st slave rod x location
y_1 is the model 1st slave rod y location
x_{1p} is the model 1st slave rod x location prime
y_{1p} is the model 1st slave rod y location prime
z_1 is the model 1st slave rod z rotation location
z_{1o} is the model 1st slave rod z rotation location origin
z_{1a} is the model 1st slave rod z rotate angle
l is the length of rod

Calculation for 2nd Slave Rod

$x_2 = 17.5*\sin(r(-sl))+12.5*\sin(r(-z_{2o}-z_0))$
$y_2 = 17.5*\cos(r(-sl))+12.5*\cos(r(-z_{2o}-z_0))$
$x_{2p} = \cos(r(z_{2o}))*x_2+\sin(r(z_{2o}))*y_2$
$y_{2p} = -\sin(r(z_{2o}))*x_2+\cos(r(z_{2o}))*y_2$
$z_{2a} = \theta*(\arcsin(x_{2p}/l))$
$z_2 = z_{2a}+ z_{2o}$

Where, x_2 is the model 2nd slave rod x location
y_2 is the model 2nd slave rod y location
x_{2p} is the model 2nd slave rod x location prime
y_{2p} is the model 2nd slave rod y location prime
z_2 is the model 2nd slave rod z rotation location
z_{2o} is the model 2nd slave rod z rotation location origin
z_{2a} is the model 2nd slave rod z rotate angle
l is the length of rod

Calculation for 3rd Slave Rod

$x_3 = 17.5*\sin(r(-sl))+12.5*\sin(r(-z_{3o}-z_0))$
$y_3 = 17.5*\cos(r(-sl))+12.5*\cos(r(-z_{3o}-z_0))$
$x_{3p} = \cos(r(z_{2o}))*x_3+\sin(r(z_{3o}))*y_3$
$y_{3p} = -\sin(r(z_{3o}))*x_3+\cos(r(z_{3o}))*y_3$
$z_{3a} = \theta*(\arcsin(x_{3p}/l))$
$z_3 = z_{3a}+ z_{3o}$

Where, x_3 is the model 3rd slave rod x location
y_3 is the model 3rd slave rod y location
x_{3p} is the model 3rd slave rod x location prime
y_{3p} is the model 3rd slave rod y location prime
z_3 is the model 3rd slave rod z rotation location
z_{3o} is the model 3rd slave rod z rotation location origin
z_{3a} is the model 3rd slave rod z rotate angle
l is the length of rod

Calculation for 4th Slave Rod

$x_4 = 17.5*\sin(r(-sl))+12.5*\sin(r(-z_{4o}-z_0))$
$y_4 = 17.5*\cos(r(-sl))+12.5*\cos(r(-z_{4o}-z_0))$
$x_{4p} = \cos(r(z_{4o}))*x_4+\sin(r(z_{4o}))*y_4$
$y_{4p} = -\sin(r(z_{4o}))*x_4+\cos(r(z_{4o}))*y_4$
$z_{4a} = \theta*(\arcsin(x_{4p}/l))$
$z_4 = z_{4a}+ z_{4o}$

Where, x_4 is the model 4th slave rod x location
y_4 is the model 4th slave rod y location
x_{4p} is the model 4th slave rod x location prime
y_{4p} is the model 4th slave rod y location prime
z_4 is the model 4th slave rod z rotation location
z_{4o} is the model 4th slave rod z rotation location origin
z_{4a} is the model 4th slave rod z rotate angle
l is the length of rod

$p_1 = -(17.5- y_0)-(75-(75*\cos(r(z_0))))$
$p_2 = -(92.5-(y_{1p}+(\cos(r(z_{1a})))))$
$p_3 = -(92.5-(y_{2p}+(\cos(r(z_{2a})))))$
$p_4 = -(92.5-(y_{3p}+(\cos(r(z_{3a})))))$
$p_5 = -(92.5-(y_{4p}+(\cos(r(z_{4a})))))$

Where, p_1 is the position1 location
p_2 is the position2 location
p_3 is the position3 location
p_4 is the position4 location
p_5 is the position5 location

VIEW AND PROJECTION MATRIX AREA

MATRIX_PROJECTION

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-1}{f} & \frac{L}{f} \end{bmatrix} \begin{bmatrix} x_v \\ y_v \\ z_v \\ 1 \end{bmatrix}$$

MATRIX_VIEW_TEMP

$$view\ vector = \frac{viewsite - viewpoint}{\|viewsite - viewpoint\|} = \frac{[0 \quad 0 \quad 0] - [0 \quad 0 \quad 350]}{\|[0 \quad 0 \quad 0] - [0 \quad 0 \quad 350]\|}$$

$$T_v^w = \begin{bmatrix} R_v^w & p_v^w \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} = \begin{bmatrix} x_v^w & y_v^w & z_v^w & p_v^w \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Rot(x)} = \begin{bmatrix} cos\theta & sin\theta & 0 & 0 \\ -sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Rot(y)} = \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Rot(z)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & sin\theta & 0 \\ 0 & -sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{MATRIX\_VIEW\_ROTATION} = \begin{bmatrix} cos\theta & sin\theta & 0 & 0 \\ -sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & sin\theta & 0 \\ 0 & -sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{MATRIX\_TRANSLATION} = \begin{bmatrix} 1 & 0 & 0 & \text{Rot(x)} \\ 0 & 1 & 0 & \text{Rot(y)} \\ 0 & 0 & 1 & \text{Rot(z)} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{MATRIX\_VIEW} = \begin{bmatrix} x_v^w & y_v^w & z_v^w & p_v^w \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} cos\theta & sin\theta & 0 & 0 \\ -sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & sin\theta & 0 \\ 0 & -sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & 0 & \text{Rot(x)} \\ 0 & 1 & 0 & \text{Rot(y)} \\ 0 & 0 & 1 & \text{Rot(z)} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

LIGHT TRANSFROMANTION AREA

$$\text{light} = \begin{bmatrix} cos\theta & sin\theta & 0 & 0 \\ -sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & sin\theta & 0 \\ 0 & -sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & 0 & \text{Rot(x)} \\ 0 & 1 & 0 & \text{Rot(y)} \\ 0 & 0 & 1 & \text{Rot(z)} \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \\ 350 \\ 1 \end{bmatrix}$$

BASE.STL MODEL TRANSFORM MATRIX AREA

$$\text{MATRIX\_MODEL\_BASE} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{MATRIX\_MVP\_BASE} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-1}{f} & \frac{L}{f} \end{bmatrix} \begin{bmatrix} x_v \\ y_v \\ z_v \\ 1 \end{bmatrix} \bullet \begin{bmatrix} x_v^w & y_v^w & z_v^w & p_v^w \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

CRANK_SHAFT.STL MODEL TRANSFORM MATRIX AREA

MATRIX_rotZ_CRANK_SHAFT = glm::rotate(glm::mat4x4(1.0f), radians(MODEL_CRANK_SHAFT_LOCATION), glm::vec3(0.0f, 0.0f, 1.0f));

$$\text{MATRIX\_MODEL\_CRANK\_SHAFT} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} * \text{MATRIX\_rotZ\_CRANK\_SHAFT}$$

$$\text{MATRIX\_RotALL\_CRANK\_SHAFT} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

$$\text{MATRIX\_MVP\_CRANK\_SHAFT} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&\frac{-1}{f}&\frac{L}{f}\end{bmatrix}\begin{bmatrix}x_v\\y_v\\z_v\\1\end{bmatrix} * \begin{bmatrix}x_v^w & y_v^w & z_v^w & p_v^w\\0&0&0&1\end{bmatrix} * \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} * \text{MATRIX\_rotZ\_CRANK\_SHAFT}$$

MASTER_ROD.STL MODEL TRANSFORM MATRIX AREA

MATRIX_rotZ_MASTER_ROD = glm::rotate(glm::mat4x4(1.0f), radians(MODEL_MASTER_ROD_Z_ROTATE_LOCATION), glm::vec3(0.0f, 0.0f, 1.0f));
MATRIX_TRANSLATION_MASTER_ROD = glm::translate(glm::mat4x4(1.0f), glm::vec3(MODEL_MASTER_ROD_X_LOCATION, MODEL_MASTER_ROD_Y_LOCATION, 0.0f));

$$\text{MATRIX\_MODEL\_MASTER\_ROD} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} * \text{MATRIX\_TRANSLATION\_MASTER\_ROD} * \text{MATRIX\_rotZ\_MASTER\_ROD};$$

$$\text{MATRIX\_RotALL\_MASTER\_ROD} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

$$\text{MATRIX\_MVP\_MASTER\_ROD} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&\frac{-1}{f}&\frac{L}{f}\end{bmatrix}\begin{bmatrix}x_v\\y_v\\z_v\\1\end{bmatrix} * \begin{bmatrix}x_v^w & y_v^w & z_v^w & p_v^w\\0&0&0&1\end{bmatrix} * \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} * \text{MATRIX\_TRANSLATION\_MASTER\_ROD} * \text{MATRIX\_rotZ\_MASTER\_ROD};$$

PISTON1.STL MODEL TRANSFORM MATRIX AREA

$$\text{MATRIX\_MODEL\_PISTON1\_ORIGIN} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

MATRIX_TRANSLATION_PISTON1 = glm::translate(glm::mat4x4(1.0f), glm::vec3(0.0f, MODEL_PISTON1_LOCATION, 0.0f));

$$\text{MATRIX\_MODEL\_PISTON1} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} * \text{MATRIX\_TRANSLATION\_PISTON1};$$

$$\text{MATRIX\_RotALL\_PISTON1} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

$$\text{MATRIX\_MVP\_PISTON1} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&\frac{-1}{f}&\frac{L}{f}\end{bmatrix}\begin{bmatrix}x_v\\y_v\\z_v\\1\end{bmatrix} * \begin{bmatrix}x_v^w & y_v^w & z_v^w & p_v^w\\0&0&0&1\end{bmatrix} * \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} * \text{MATRIX\_TRANSLATION\_PISTON1};$$

SLAVE_ROD1(SLAVE_ROD1.STL) MODEL TRANSFORM MATRIX AREA

```
MATRIX_rotZ_SLAVE_ROD_1 = glm::rotate(glm::mat4x4(1.0f), radians(MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION), glm::vec3(0.0f, 0.0f, 1.0f));
MATRIX_TRANSLATION_SLAVE_ROD_1 = glm::translate(glm::mat4x4(1.0f), glm::vec3(MODEL_SLAVE_ROD_1_X_LOCATION,  MODEL_SLAVE_ROD_1_Y_LOCATION, 0.0f));
```

$$MATRIX\_MODEL\_SLAVE\_ROD\_1 = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} * MATRIX\_TRANSLATION\_SLAVE\_ROD\_1 * MATRIX\_rotZ\_SLAVE\_ROD\_1$$

$$MATRIX\_RotALL\_SLAVE\_ROD\_1 = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

$$MATRIX\_MVP\_SLAVE\_ROD\_1 = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&\frac{-1}{f}&\frac{L}{f}\end{bmatrix}\begin{bmatrix}x_v\\y_v\\z_v\\1\end{bmatrix} * \begin{bmatrix}x_v^w&y_v^w&z_v^w&p_v^w\\0&0&0&1\end{bmatrix} * \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} * MATRIX\_TRANSLATION\_SLAVE\_ROD\_1 * MATRIX\_rotZ\_SLAVE\_ROD\_1$$

TECHNIQUES USED

〈STL FILE LOADING METHOD (ASCII TYPE)〉

An ASCII STL file begins with the line

solid (name) // name is option

The file continues with any number of triangles, each represented as follows:

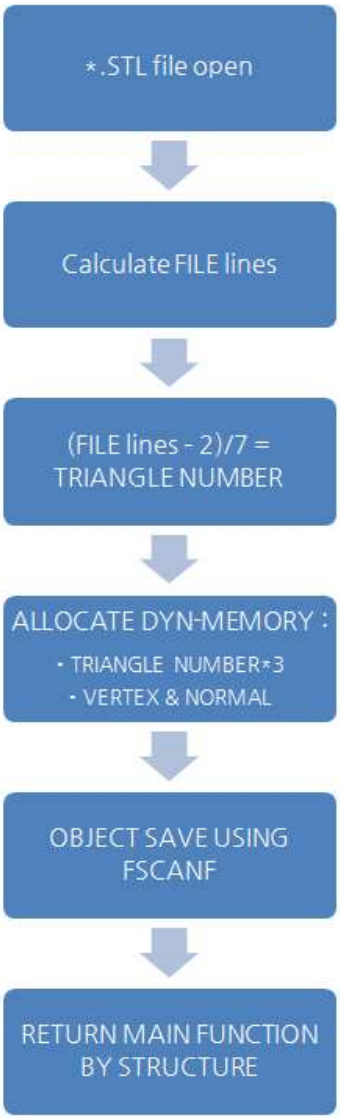
```
facet normal ni nj nk
   outer loop
      vertex v1x v1y v1z
      vertex v2x v2y v2z
      vertex v3x v3y v3z
   endloop
endfacet
```

The file concludes with:

endsolid (name) // name is option

STL파일의 이러한 구성을 이용하여 VERTEX, NORMAL, INDEX를 구하였다.

다음은 STL파일에서 OPJECT들을 저장하는 프로세스이다.



<KEY & MOUSE INPUT>

GLUT함수 안에는 키보드와 마우스를 입력받는 함수들이 있으며 이들을 사용하여 키보드와 마우스 입력을 받았다.

```
glutSpecialFunc(SpecialKeyboard); //키보드의 기능키를 입력받는 함수
void SpecialKeyboard(int key, int x, int y) { //기능키를 입력받았을 때 일어날 이벤트를 등록하는 함수
    switch(key) {
        case GLUT_KEY_UP :
                        RATE_PER_MINUTE += 10.0f;
                        printf("        ANIMATION >>> RATE PER MINUTE : %5.3f₩n",RATE_PER_MINUTE);
        break;
        case GLUT_KEY_DOWN :
                        RATE_PER_MINUTE -= 10.0f;
                        printf("        ANIMATION >>> RATE PER MINUTE : %5.3f₩n",RATE_PER_MINUTE);
        break;
        //case GLUT_KEY_LEFT : KEY_ROTATE_MODEL_III_Z += 2.5f; break;
        //case GLUT_KEY_RIGHT : KEY_ROTATE_MODEL_III_Z -= 2.5f; break;
        case GLUT_KEY_F1 : ANIMATION_SWITCH = 1; printf("    ACTION >>> ANIMATION [ON]₩n"); break;
        case GLUT_KEY_F2 : ANIMATION_SWITCH = 0; printf("    ACTION >>> ANIMATION [OFF]₩n"); break;
        case GLUT_KEY_F3 :
            printf("        ANIMATION >>> VIEW MODEL STATUS₩n");
            printf("       ##############################################₩n");
            printf("       #               MODEL STATUS VIEWER              #₩n");
            printf("       ##############################################₩n");
            printf("         MODEL ENGINE RPM :              [%4.3f RPM]    ₩n",RATE_PER_MINUTE);
```

```
            printf("        CRANK SHAFT ROTATED ANGLE :  [%4.3f]    ₩n",MODEL_CRANK_SHAFT_LOCATION);
            printf("        PISTON 1 LOCATION :          [%4.3f]    ₩n",MODEL_PISTON1_LOCATION);
            printf("                                [TOP POINT ORIGIN]     ₩n");
            printf("        MASTER ROD LOCATION :                          ₩n");
            printf("                                                [X    Axis    :    %4.3f]          [Y    Axis    :    %4.3f]
₩n",MODEL_MASTER_ROD_X_LOCATION,MODEL_MASTER_ROD_Y_LOCATION);
            printf("        MASTER ROD ROTATED ANGLE :   [%4.3f DEG]    ₩n",MODEL_MASTER_ROD_Z_ROTATE_LOCATION);
            printf("      ##############################################₩n");
        break;

        case GLUT_KEY_F4 : if ( ANIMATION_SWITCH == 0 ) ANIMATION_MODEL_RESET = 1; break;
        case GLUT_KEY_F8 :
            printf("    ACTION >>> VIEW POSITION RESET...!!!₩n");
            VIEW_MOVEMENT_Z = 0;
            VIEW_MOVEMENT_X = 0; VIEW_MOVEMENT_Y = 0;
            VIEW_ROTATION_X = 0; VIEW_ROTATION_Y = 0; break;
        case GLUT_KEY_END :
            printf("    ACTION >>> VIEW POSITION [ISOMATRIC]₩n");
            VIEW_ROTATION_X = 45.0f; VIEW_ROTATION_Y = -45.0f;
            break;
    }
}
glutKeyboardFunc(DoKeyboard); // 일반키보드를 입력받는 함수
void DoKeyboard(unsigned char key, int x, int y)   { //일반 키보드를 입력받았을 때 일어날 이벤트를 등록하는 함수.
          switch (key) {
                    case 033 : glutLeaveMainLoop(); break; // ESC KEY FUNCTION IS PROGRAM EXIT.
          }
}
glutEntryFunc(MouseActivatonConfirm); //마우스가 opengl실행 창 안에 있는지 아닌지를 확인하는 함수
glutMouseFunc(MouseButtonActivationConfirm); //마우스버튼 입력함수
glutMotionFunc(MouseMotionConfirm); //마우스 움직임 검출 함수
void MouseActivatonConfirm(int state) { //마우스가 opengl실행창 안에 있을때와 아닐 때 일어날 이벤트를 등록하는 함수
    if( state == GLUT_LEFT) { //마우스가 창 밖에 있는 경우 마우스 기능 비활성화
        MOUSE_ACTIVATION = 0;
    }
    else { //마우스가 창 안에 있는 경우 마우스 기능 활성화.
        MOUSE_ACTIVATION = 1;
    }
}
void MouseButtonActivationConfirm(int button, int state, int x, int y) { //마우스 버튼을 누를 때 일어날 이벤트를 등록하는 함수
    if ( MOUSE_ACTIVATION == 1 ) { //마우스가 창 안에 있을때
        if ( state == GLUT_DOWN ) { //눌렀을때
            if ( button == GLUT_LEFT_BUTTON ) { //마우스 왼쪽버튼
                LEFT_MOUSE_ACTIVATE = 1;
            }
            else if( button == GLUT_MIDDLE_BUTTON ) { //마우스 오른쪽버튼
                MIDDLE_MOUSE_ACTIVATE = 1;
            }
        }
        else {
            if ( button == GLUT_WHEEL_UP ) VIEW_MOVEMENT_Z += 5.0f; //마우스 휠을 올릴때
            else if ( button == GLUT_WHEEL_DOWN ) VIEW_MOVEMENT_Z -= 5.0f; //마우스 휠을 내릴때
            else { //그 밖에 경우 기능정지
                LEFT_MOUSE_ACTIVATE = 0;
                MIDDLE_MOUSE_ACTIVATE = 0;
            }
        }
    }
}
```

```
void MouseMotionConfirm(int x, int y) { //마우스의 움직임을 검출하는 함수
    if ( MOUSE_ACTIVATION == 1 ) { //마우스가 창 안에 있을때
        if (MOUSE_POSTION_DETECTION == 0) { //위치 검출 안되있을때
            MOUSE_POS_X_OLD = x;
            MOUSE_POS_Y_OLD = y;
            MOUSE_POSTION_DETECTION = 1;
        }
        else if (MOUSE_POSTION_DETECTION == 1) { //마우스 위치검출이 확인된경우
            MOUSE_POS_X = x;
            MOUSE_POS_Y = y;
            if ( LEFT_MOUSE_ACTIVATE == 1) { //왼쪽 마우스를 켰을때
                if ( (MOUSE_POS_X - MOUSE_POS_X_OLD) > 0) {
                    VIEW_MOVEMENT_X += 2.5f;
                }
                else if ((MOUSE_POS_X - MOUSE_POS_X_OLD) < 0) {
                    VIEW_MOVEMENT_X -= 2.5f;
                }
                if ( (MOUSE_POS_Y - MOUSE_POS_Y_OLD) < 0) {
                    VIEW_MOVEMENT_Y += 2.5f;
                }
                else if ((MOUSE_POS_Y - MOUSE_POS_Y_OLD) > 0) {
                    VIEW_MOVEMENT_Y -= 2.5f;
                }
            }
            if ( MIDDLE_MOUSE_ACTIVATE == 1) { //휠버튼을 눌렀을때
                if ( (MOUSE_POS_X - MOUSE_POS_X_OLD) > 0) {
                    VIEW_ROTATION_Y += 2.5f;
                }
                else if ((MOUSE_POS_X - MOUSE_POS_X_OLD) < 0) {
                    VIEW_ROTATION_Y -= 2.5f;
                }
                if ( (MOUSE_POS_Y - MOUSE_POS_Y_OLD) > 0) {
                    VIEW_ROTATION_X += 2.5f;
                }
                else if ((MOUSE_POS_Y - MOUSE_POS_Y_OLD) < 0) {
                    VIEW_ROTATION_X -= 2.5f;
                }
            }
            MOUSE_POSTION_DETECTION = 0;
        }
    }
}
```

<ANIMATION>

opengl에서 애니메이션을 실시 할 경우, 따로 애니메이션 시간을 정해주지 않으면 프로세스의 속도에 따라서 루프의 속도가 달라진다. -> 원하는시간으로 제어할 수 없음.
따라서 특정시간에 특정 이벤트를 실시하는 glut함수를 사용하였다.

```
glutTimerFunc(ANIMATION_DISPLAY_PERIOD, AnimationTimer, 1); //ANIMATION_DISPLAY_PERIOD시간마다 animationtimer을 실행
void AnimationTimer(int value) {
    if ( ANIMATION_SWITCH == 1) {
        RPM_TO_DEGREE_PER_MILISECONDS = (6*RATE_PER_MINUTE*ANIMATION_DISPLAY_PERIOD)/1000;
        MODEL_CRANK_SHAFT_LOCATION += RPM_TO_DEGREE_PER_MILISECONDS;
        if ( MODEL_CRANK_SHAFT_LOCATION > 359) MODEL_CRANK_SHAFT_LOCATION = 0;

        MODEL_MASTER_ROD_X_LOCATION = 17.5*(sin(radians(-MODEL_CRANK_SHAFT_LOCATION)));
        MODEL_MASTER_ROD_Y_LOCATION = 17.5*(cos(radians(-MODEL_CRANK_SHAFT_LOCATION)));
        MODEL_MASTER_ROD_Z_ROTATE_LOCATION = degrees(asin(MODEL_MASTER_ROD_X_LOCATION/75));
```

```
	MODEL_SLAVE_ROD_1_X_LOCATION	=	17.5*(sin(radians(-MODEL_CRANK_SHAFT_LOCATION)))	+
12.5*(sin(radians(-MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
	MODEL_SLAVE_ROD_1_Y_LOCATION	=	17.5*(cos(radians(-MODEL_CRANK_SHAFT_LOCATION)))	+
12.5*(cos(radians(-MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
	MODEL_SLAVE_ROD_1_X_LOCATION_PRIME	=
(cos(radians(MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_1_X_LOCATION)	+
(sin(radians(MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_1_Y_LOCATION);
	MODEL_SLAVE_ROD_1_Y_LOCATION_PRIME	=
(-sin(radians(MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_1_X_LOCATION)	+
(cos(radians(MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_1_Y_LOCATION);
	MODEL_SLAVE_ROD_1_Z_ROTATE_ANGLE	=
degrees(asin(MODEL_SLAVE_ROD_1_X_LOCATION_PRIME/MODEL_SLAVE_ROD_LENGTH));
	MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION	=	MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN	+
MODEL_SLAVE_ROD_1_Z_ROTATE_ANGLE;

	MODEL_SLAVE_ROD_2_X_LOCATION	=	17.5*(sin(radians(-MODEL_CRANK_SHAFT_LOCATION)))	+
12.5*(sin(radians(-MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
	MODEL_SLAVE_ROD_2_Y_LOCATION	=	17.5*(cos(radians(-MODEL_CRANK_SHAFT_LOCATION)))	+
12.5*(cos(radians(-MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
	MODEL_SLAVE_ROD_2_X_LOCATION_PRIME	=
(cos(radians(MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_2_X_LOCATION)	+
(sin(radians(MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_2_Y_LOCATION);
	MODEL_SLAVE_ROD_2_Y_LOCATION_PRIME	=
(-sin(radians(MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_2_X_LOCATION)	+
(cos(radians(MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_2_Y_LOCATION);
	MODEL_SLAVE_ROD_2_Z_ROTATE_ANGLE	=
degrees(asin(MODEL_SLAVE_ROD_2_X_LOCATION_PRIME/MODEL_SLAVE_ROD_LENGTH));
	MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION	=	MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN	+
MODEL_SLAVE_ROD_2_Z_ROTATE_ANGLE;

	MODEL_SLAVE_ROD_3_X_LOCATION	=	17.5*(sin(radians(-MODEL_CRANK_SHAFT_LOCATION)))	+
12.5*(sin(radians(-MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
	MODEL_SLAVE_ROD_3_Y_LOCATION	=	17.5*(cos(radians(-MODEL_CRANK_SHAFT_LOCATION)))	+
12.5*(cos(radians(-MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
	MODEL_SLAVE_ROD_3_X_LOCATION_PRIME	=
(cos(radians(MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_3_X_LOCATION)	+
(sin(radians(MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_3_Y_LOCATION);
	MODEL_SLAVE_ROD_3_Y_LOCATION_PRIME	=
(-sin(radians(MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_3_X_LOCATION)	+
(cos(radians(MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_3_Y_LOCATION);
	MODEL_SLAVE_ROD_3_Z_ROTATE_ANGLE	=
degrees(asin(MODEL_SLAVE_ROD_3_X_LOCATION_PRIME/MODEL_SLAVE_ROD_LENGTH));
	MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION	=	MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN	+
MODEL_SLAVE_ROD_3_Z_ROTATE_ANGLE;

	MODEL_SLAVE_ROD_4_X_LOCATION	=	17.5*(sin(radians(-MODEL_CRANK_SHAFT_LOCATION)))	+
12.5*(sin(radians(-MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
	MODEL_SLAVE_ROD_4_Y_LOCATION	=	17.5*(cos(radians(-MODEL_CRANK_SHAFT_LOCATION)))	+
12.5*(cos(radians(-MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
	MODEL_SLAVE_ROD_4_X_LOCATION_PRIME	=
(cos(radians(MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_4_X_LOCATION)	+
(sin(radians(MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_4_Y_LOCATION);
	MODEL_SLAVE_ROD_4_Y_LOCATION_PRIME	=
(-sin(radians(MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_4_X_LOCATION)	+
(cos(radians(MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_4_Y_LOCATION);
	MODEL_SLAVE_ROD_4_Z_ROTATE_ANGLE	=
degrees(asin(MODEL_SLAVE_ROD_4_X_LOCATION_PRIME/MODEL_SLAVE_ROD_LENGTH));
	MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION	=	MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN	+
MODEL_SLAVE_ROD_4_Z_ROTATE_ANGLE;
```

```
        MODEL_PISTON1_LOCATION                  =                  -(17.5-MODEL_MASTER_ROD_Y_LOCATION)                  -
(75-(75*cos(radians(MODEL_MASTER_ROD_Z_ROTATE_LOCATION))));
        MODEL_PISTON2_LOCATION              =              -(92.5f-(MODEL_SLAVE_ROD_1_Y_LOCATION_PRIME                  +
(cos(radians(MODEL_SLAVE_ROD_1_Z_ROTATE_ANGLE)) * MODEL_SLAVE_ROD_LENGTH)));
        MODEL_PISTON3_LOCATION              =              -(92.5f-(MODEL_SLAVE_ROD_2_Y_LOCATION_PRIME                  +
(cos(radians(MODEL_SLAVE_ROD_2_Z_ROTATE_ANGLE)) * MODEL_SLAVE_ROD_LENGTH)));
        MODEL_PISTON4_LOCATION              =              -(92.5f-(MODEL_SLAVE_ROD_3_Y_LOCATION_PRIME                  +
(cos(radians(MODEL_SLAVE_ROD_3_Z_ROTATE_ANGLE)) * MODEL_SLAVE_ROD_LENGTH)));
        MODEL_PISTON5_LOCATION              =              -(92.5f-(MODEL_SLAVE_ROD_4_Y_LOCATION_PRIME                  +
(cos(radians(MODEL_SLAVE_ROD_4_Z_ROTATE_ANGLE)) * MODEL_SLAVE_ROD_LENGTH)));

    }
    else if ( ANIMATION_SWITCH == 0) {
        if ( ANIMATION_MODEL_RESET == 1 ) {
            printf("        ANIMATION >>> ALL ANIMATION MODEL LOCATION RESET!!!₩n");
            MODEL_CRANK_SHAFT_LOCATION = 0;
            MODEL_MASTER_ROD_X_LOCATION = 17.5*(sin(radians(-MODEL_CRANK_SHAFT_LOCATION)));
            MODEL_MASTER_ROD_Y_LOCATION = 17.5*(cos(radians(-MODEL_CRANK_SHAFT_LOCATION)));
            MODEL_MASTER_ROD_Z_ROTATE_LOCATION = degrees(asin(MODEL_MASTER_ROD_X_LOCATION/75));

            MODEL_SLAVE_ROD_1_X_LOCATION          =          17.5*(sin(radians(-MODEL_CRANK_SHAFT_LOCATION)))          +
12.5*(sin(radians(-MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
            MODEL_SLAVE_ROD_1_Y_LOCATION          =          17.5*(cos(radians(-MODEL_CRANK_SHAFT_LOCATION)))          +
12.5*(cos(radians(-MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
            MODEL_SLAVE_ROD_1_X_LOCATION_PRIME                                                                  =
(cos(radians(MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_1_X_LOCATION)                  +
(sin(radians(MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_1_Y_LOCATION);
            MODEL_SLAVE_ROD_1_Y_LOCATION_PRIME                                                                  =
(-sin(radians(MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_1_X_LOCATION)                  +
(cos(radians(MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_1_Y_LOCATION);
            MODEL_SLAVE_ROD_1_Z_ROTATE_ANGLE                                                                    =
degrees(asin(MODEL_SLAVE_ROD_1_X_LOCATION_PRIME/MODEL_SLAVE_ROD_LENGTH));
            MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION       =       MODEL_SLAVE_ROD_1_Z_ROTATE_LOCATION_ORIGIN       +
MODEL_SLAVE_ROD_1_Z_ROTATE_ANGLE;

            MODEL_SLAVE_ROD_2_X_LOCATION          =          17.5*(sin(radians(-MODEL_CRANK_SHAFT_LOCATION)))          +
12.5*(sin(radians(-MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
            MODEL_SLAVE_ROD_2_Y_LOCATION          =          17.5*(cos(radians(-MODEL_CRANK_SHAFT_LOCATION)))          +
12.5*(cos(radians(-MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
            MODEL_SLAVE_ROD_2_X_LOCATION_PRIME                                                                  =
(cos(radians(MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_2_X_LOCATION)                  +
(sin(radians(MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_2_Y_LOCATION);
            MODEL_SLAVE_ROD_2_Y_LOCATION_PRIME                                                                  =
(-sin(radians(MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_2_X_LOCATION)                  +
(cos(radians(MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_2_Y_LOCATION);
            MODEL_SLAVE_ROD_2_Z_ROTATE_ANGLE                                                                    =
degrees(asin(MODEL_SLAVE_ROD_2_X_LOCATION_PRIME/MODEL_SLAVE_ROD_LENGTH));
            MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION       =       MODEL_SLAVE_ROD_2_Z_ROTATE_LOCATION_ORIGIN       +
MODEL_SLAVE_ROD_2_Z_ROTATE_ANGLE;

            MODEL_SLAVE_ROD_3_X_LOCATION          =          17.5*(sin(radians(-MODEL_CRANK_SHAFT_LOCATION)))          +
12.5*(sin(radians(-MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
            MODEL_SLAVE_ROD_3_Y_LOCATION          =          17.5*(cos(radians(-MODEL_CRANK_SHAFT_LOCATION)))          +
12.5*(cos(radians(-MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
            MODEL_SLAVE_ROD_3_X_LOCATION_PRIME                                                                  =
(cos(radians(MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_3_X_LOCATION)                  +
(sin(radians(MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_3_Y_LOCATION);
            MODEL_SLAVE_ROD_3_Y_LOCATION_PRIME                                                                  =
```

```
(-sin(radians(MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_3_X_LOCATION) +
(cos(radians(MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_3_Y_LOCATION);
        MODEL_SLAVE_ROD_3_Z_ROTATE_ANGLE =
degrees(asin(MODEL_SLAVE_ROD_3_X_LOCATION_PRIME/MODEL_SLAVE_ROD_LENGTH));
        MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION = MODEL_SLAVE_ROD_3_Z_ROTATE_LOCATION_ORIGIN +
MODEL_SLAVE_ROD_3_Z_ROTATE_ANGLE;

        MODEL_SLAVE_ROD_4_X_LOCATION = 17.5*(sin(radians(-MODEL_CRANK_SHAFT_LOCATION))) +
12.5*(sin(radians(-MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
        MODEL_SLAVE_ROD_4_Y_LOCATION = 17.5*(cos(radians(-MODEL_CRANK_SHAFT_LOCATION))) +
12.5*(cos(radians(-MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN - MODEL_MASTER_ROD_Z_ROTATE_LOCATION)));
        MODEL_SLAVE_ROD_4_X_LOCATION_PRIME =
(cos(radians(MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_4_X_LOCATION) +
(sin(radians(MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_4_Y_LOCATION);
        MODEL_SLAVE_ROD_4_Y_LOCATION_PRIME =
(-sin(radians(MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_4_X_LOCATION) +
(cos(radians(MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN))*MODEL_SLAVE_ROD_4_Y_LOCATION);
        MODEL_SLAVE_ROD_4_Z_ROTATE_ANGLE =
degrees(asin(MODEL_SLAVE_ROD_4_X_LOCATION_PRIME/MODEL_SLAVE_ROD_LENGTH));
        MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION = MODEL_SLAVE_ROD_4_Z_ROTATE_LOCATION_ORIGIN +
MODEL_SLAVE_ROD_4_Z_ROTATE_ANGLE;

        MODEL_PISTON1_LOCATION = -(17.5-MODEL_MASTER_ROD_Y_LOCATION) -
(75-(75*cos(radians(MODEL_MASTER_ROD_Z_ROTATE_LOCATION))));
        MODEL_PISTON2_LOCATION = -(92.5f-(MODEL_SLAVE_ROD_1_Y_LOCATION_PRIME +
(cos(radians(MODEL_SLAVE_ROD_1_Z_ROTATE_ANGLE)) * MODEL_SLAVE_ROD_LENGTH)));
        MODEL_PISTON3_LOCATION = -(92.5f-(MODEL_SLAVE_ROD_2_Y_LOCATION_PRIME +
(cos(radians(MODEL_SLAVE_ROD_2_Z_ROTATE_ANGLE)) * MODEL_SLAVE_ROD_LENGTH)));
        MODEL_PISTON4_LOCATION = -(92.5f-(MODEL_SLAVE_ROD_3_Y_LOCATION_PRIME +
(cos(radians(MODEL_SLAVE_ROD_3_Z_ROTATE_ANGLE)) * MODEL_SLAVE_ROD_LENGTH)));
        MODEL_PISTON5_LOCATION = -(92.5f-(MODEL_SLAVE_ROD_4_Y_LOCATION_PRIME +
(cos(radians(MODEL_SLAVE_ROD_4_Z_ROTATE_ANGLE)) * MODEL_SLAVE_ROD_LENGTH)));

        ANIMATION_MODEL_RESET = 0;
    }
  }
    glutPostRedisplay();
    glutTimerFunc(ANIMATION_DISPLAY_PERIOD, AnimationTimer, 1); //애니메이션 함수 안에 스스로를 호출하는 함수를 집어넣어야
                                                                //연속적으로 실행이 된다.
}
```

프로젝트 타임 테이블

STAGE	완성도	15주차
FREEGLUT 기초 사용법 이해	99%	
STL FILE LOADER SUB ROUTINE CONSTRUCT	99%	
프로젝트용 RADIAL ENGINE 모델링	100%	
RADIAL ENGINE 모델링 LOAD & PLACEMENT	100%	
애니메이션 적용 및 기타 DECORATION	95%	
발표	99%	

DEVELOPMENT ENVEROMENT

- **Windows Based**
  - OS : Windows 10 Pro X64

- H/W : AMD A8-6410 APU, 8GB RAM, AMD Radeon R5 & AMD Radeon HD8500 Dual GPU
- Compiler : MinGW-w64 - GCC - 6.2.0 X64
- IDE S/W : Code::Blocks 16.10, Gedit Editor
- openGL Lib : freeglut latest ver, GLEW, GLM also.

참고문헌

(1) Radial engine-Wikipedia, https://en.wikipedia.org/wiki/Radial_engine

(2) <Design a four-cylinder Internal Combution Engine>, Editor: Radoslav Plamenov Georgiev, 2011

(3) <Design a Radial Engine> Editor: M Tsankov Vasilev, 2011

(4) Radial Engine-Wikipedia, namu.wiki/w/성형엔진

(5) CAD/CAM강의자료 - 명지대학교 기계공학과

(6) openGL 강의 - http://3d.sangji.ac.kr/ppt/CG/