Comp380 Report : PA3

Programming Assignment #3

Name: In-Young Cho (조인영)
ID : 20150720

● 일단 cpp파일에서 //edit ~ //end라고 묶여져 있는 부분은 모두 제가 수정한 코드입니다.

```
#include <cstdlib> // for using rand function
#include <ctime>
-----------------------------------------------------------------------------------------------
int onX = 1; int onY = 0; int onZ = 0; // K키가 눌러지면 onK는 1, 아니라면 0. K = X , Y, Z 중 하나. 일단 처음에는 x룩 방향으로 움직이는게 default
int tmpOnX = onX; int tmpOnY = onY; int tmpOnZ = onZ; // r 키가 눌러졌을 때 onK의 값을 tmpOnK에 임시저장하고 onK는 0으로. 그러면 회전도중 드래그를 해도 소는 움직이지 않음. 다시 r키가 눌러지면 저장했던 값을 onK에.
GLdouble angleX = 0; GLdouble angleY = 0; GLdouble angleZ = 0; // glRotated의 x,y,z 자리에 들어갈 글로벌 변수
int isRotate = 0; // 회전모드에서 1, 회전모드가 아닐 때 0
int modelOrView = 0; // 변환을 원하는 공간이 modelSpace면 0, viewSpace면 1
-----------------------------------------------------------------------------------------------
void drawRotFrame() { // drawFrame()과 흡사하다. 회전하는 방향(angleX, ...)에 대응하는 회전축을 흰색으로 그린다. 길이는 10이고 중심은 소의 모델링 공간 (0,0,0)에 있다.
	if (isRotate != 0) { // 즉, 소가 돌지 않고 멈춰있다면 회전축을 그릴 필요가 없다.
		GLdouble norm = sqrt(angleX*angleX + angleY*angleY + angleZ*angleZ);
		glDisable(GL_LIGHTING);
		glBegin(GL_LINES);
		glColor3d(1, 1, 1);
		if (norm != 0.0) {
			glVertex3d(-5 * angleX / norm, -5 * angleY / norm, -5 * angleZ / norm);
			glVertex3d(5 * angleX / norm, 5 * angleY / norm, 5 * angleZ / norm);
		}
		glEnd();
	}
}
-----------------------------------------------------------------------------------------------
void onMouseDrag( int x, int y ) {
    y = height - y - 1;
    printf( "in drag (%d, %d)\n", x - oldX,  y - oldY );

    // (Project 2,3,4) TODO : Implement here to perform properly when drag the mouse on each case, respectively.
	//edited
	glPushMatrix();

		if (modelOrView == 0) { //만약 modeling space에서 이동을 원한다면,
			glLoadMatrixd(cow2wld.matrix()); // 소의 변환 행렬을 로드한다
			glTranslated((x - oldX)*onX*0.05, (x - oldX)*onY*0.05, (x - oldX)*onZ*0.05);// x - oldX는 드래그 보폭, onK는 어떤 방향으로 움직여야 하는지. 0.1은 임의의 속력 상수
			//cow2wld 행렬을 로드했으므로, 위 arguments 의 값은 소 좌표계 기준이다. 이 translation은 world에 반영된다.
		}
		else { // viewing space에서 이동을 원한다면,
			glLoadMatrixd(cam2wld[cameraIndex].matrix());
```

```
                        glTranslated((x - oldX)*(onX || onY)*0.05, (y - oldY)*(onX || onY)*0.05, (x - oldX)*onZ*0.05); // 'x'또는 'y'를 눌렀다면 x-y plane 상에서 이동. 'z'를 눌렀다면 (캠 좌표상) z축 방향으로 이동.
                        //위와 마찬가지로, cam2wld[cameraIndex] 행렬을 로드했으므로, 위 arguments 의 값은 캠 좌표계 기준이다.
                        glMultMatrixd(wld2cam[cameraIndex].matrix());
                        glMultMatrixd(cow2wld.matrix());
                        //위 두 행렬의 곱은 결국 cow2cam 과 동치
                        //위 두 행렬을 current matrix에 곱해줌으로써 캠 좌표계 기준의 변환을 소 좌표계 기준으로 환산한다. translation을 world에 반영한다.
                }

                if (isRotate == 1 && modelOrView == 1) { // 돌 수 있는 상태이고, viewing space에서 이동을 원한다면,
                        glLoadMatrixd(cow2wld.matrix());
                        glRotated((x - oldX), angleX, angleY, angleZ);
                        //renderRotaion() 에서도 썼던 기본적인 modeling space상 회전. 축은 v상태에서 r키를 누를 때 결정된다.
                }

                glGetDoublev(GL_MODELVIEW_MATRIX, cow2wld.matrix());
                oldX = x; // 위치를 저장한다
                oldY = y;

        glPopMatrix();
        //end
    glutPostRedisplay();
}
-----------------------------------------------------------------------------------------------
void renderRotation(void) { // IdleAnimation으로 소를 회전시키는 함수
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glPushMatrix();
        glLoadMatrixd(cow2wld.matrix());
        glRotated(1, angleX, angleY, angleZ); // 정해진 회전축 (angleX, ...) 를 기준으로 1만큼 회전
        glGetDoublev(GL_MODELVIEW_MATRIX, cow2wld.matrix());
        glPopMatrix();
        glutPostRedisplay();
}

void selectRotationAxis(void) { // 경우에 따라 랜덤하게 축을 선택하거나, 화면의 x 축에 해당하는 축을 선택한다. 이는 modelOrView의 값에 의존한다.

        if (cameraIndex >= (int)wld2cam.size()) // 정해진 카메라 수 보다 큰 값이 저장되는 것을 방지한다.
                cameraIndex = 0;

        if (modelOrView == 0) { //modeling space. PA2 에서는 onKeyPress()에 있던걸 가져와 합쳤다.
                srand((unsigned int)time(NULL));
                angleX = (GLdouble)rand(); angleY = (GLdouble)rand(); angleZ = (GLdouble)rand();
        }
        else if (modelOrView == 1) { //viewing space
                FrameXform tmp;
```

```
                glPushMatrix();
                        tmp = cow2wld.inverse(); //wld2cow
                        glLoadMatrixd(tmp.matrix());
                        glMultMatrixd(cam2wld[cameraIndex].matrix()); // wld2cow * cam2wld = cam2cow
                        glGetDoublev(GL_MODELVIEW_MATRIX, tmp.matrix()); // tmp에 위 행렬 저장
                glPopMatrix();
                //이제 tmp 행렬은 cam 좌표계 위의 점 (혹은 벡터)을 cow 좌표계 위로 보내는 아핀변환이다.

                angleX = tmp.matrix()[0];
                angleY = tmp.matrix()[1];
                angleZ = tmp.matrix()[2];
                // that is, tmp * {1,0,0}^t
                // -> 즉 cam space 의 x축 방향 단위벡터를 변환한다.
        }
}
-----------------------------------------------------------------------------------------------
void onKeyPress( unsigned char key, int x, int y ) {
    // If 'c' or space bar are pressed, alter the camera.
    // If a number is pressed, alter the camera corresponding the number.
    if ( ( key == ' ' ) || ( key == 'c' ) ) {
        printf( "Toggle camera %d\n", cameraIndex );
        cameraIndex += 1;
                //edit
                if ( isRotate == 1 && modelOrView == 1 ) selectRotationAxis(); // viewing space에서 회전하는 중,
카메라를 변경할 때 회전축도 같이 바꿔준다.
                //end
        }
        else if ((key >= '0') && (key <= '9')) {
                cameraIndex = key - '0';
                //edit
                if (isRotate == 1 && modelOrView == 1) selectRotationAxis(); // viewing space에서 회전하는 중,
카메라를 변경할 때 회전축도 같이 바꿔준다.
                //end
        }

        if (cameraIndex >= (int)wld2cam.size())
                cameraIndex = 0;

    // (Project 2,3,4) TODO : Implement here to handle keyboard input.
        //edited
        if (key == 'v') {
                if (isRotate == 1 && modelOrView == 0) { // modeling space에서 회전 중 이면,
                        glutIdleFunc(NULL); // stop rotating
                        modelOrView = 1;
                        selectRotationAxis();
                        //일단 회전을 멈추고 viewing space에서 축을 설정하고 기다린다.
                }
                modelOrView = 1;
        }
```

```
	if (key == 'm') {
		if (isRotate == 1 && modelOrView == 1) { // viewing space에서 회전 중 이면,
			modelOrView = 0;
			selectRotationAxis();
			glutIdleFunc(renderRotation);
			//modeling space에서 축을 설정한 뒤 계속 회전한다.
		}
		modelOrView = 0;
	}

	if (key == 'x' || key == 'y' || key == 'z'){
		if (isRotate == 1) { //회전 중이면 회전을 멈춰라.
			if (modelOrView == 0)		glutIdleFunc(NULL); // stop rotating
			isRotate = 0;
		}
		onX = (key == 'x');	onY = (key == 'y'); onZ = (key == 'z'); // 각 키가 눌러지면, 해당하는 onK 를 1로, 아니면 0으로
	}

	if (key == 'r') {
		if (isRotate == 1) { //회전 중이면 회전을 멈춰라. 이전 translation 방향을 유지하라
			if (modelOrView == 0)		glutIdleFunc(NULL); // stop rotating
			isRotate = 0;
			onX = tmpOnX; onY = tmpOnY; onZ = tmpOnZ; // re-assignment
		}
		else {
			tmpOnX = onX; tmpOnY = onY; tmpOnZ = onZ; // tmpOnK <-swap-> onK
			onX = 0; onY = 0; onZ = 0; // 0으로 만들면 평행이동이 작동하지 않는다.
			isRotate = 1;
			selectRotationAxis();
			if (modelOrView == 0)		glutIdleFunc(renderRotation); //modeling 공간에서 회전을 원하면 IdleFunc실행
		}
	}
	//end

    glutPostRedisplay();
}
```