단국대학교 멀티미디어공학 그래픽스 프로그래밍 중간고사 (2016년 봄학기) 2016년 4월 26일 화요일

학과 _____ 학번 _____ 이름 _____

중간고사

담당교수: 단국대학교 멀티미디어공학전공 박경신

- **답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안 함. 답에는 반드시 네모를 쳐서 확실히 표시할 것.**
- **답안지에 학과, 학번, 이름 외에 본인의 암호를 기입하면 성적공고시 학번 대신 암호를 사용할 것임.**

1. 다음 문제에 답하시오. (총 30점)

1) **아핀 공간 (Affine Space)과 벡터 공간 (Vector Space)을 간단히 서술하라.** (5점)

Affine Space – 벡터공간에 점을 추가한 공간. 임의의 벡터와 임의의 점의 표현이 가능.

Vector Space – 벡터 공간은 실수(scalar)와 벡터(vector)만 표현이 가능한 공간. 임의의 벡터 v = 기저벡터 (v1, v2, .. vn)

2) **OpenGL에서 Immediate Mode Graphics와 Retained Mode Graphics의 차이점을 간단히 설명하시오.** (5점)

Immediate Mode Graphics. – 과거 OpenGL에서 glVertex를 사용해서 렌더링했던 방식. 즉, 응용프로그램에서 매번 정점의 정보가 지정되고 렌더링되는 방식

Retained Mode Graphics – 모든 정점(vertex)와 속성(attribute) 정보를 배열 (array)로 지정해서 이 배열을 GPU로 보내어 저장하고 렌더링에 사용하는 방식.

3) **OpenGL 렌더링을 하기 위해 사용하는 Vertex Array Object (VAO), Vertex Buffer Object (VBO), Index Buffer Object (IBO)이 무엇인지 간단히 설명하라.** (5점)

VAO (Vertex Array Object) – 모든 정점자료(즉, position, color, ...)를 하나로 묶어줌. 보통 하나의 Mesh (즉, 기하객체)마다 하나의 VAO를 사용함. 즉, 하나의 VAO안에는 하나 또는 여러 개의 VBO가 존재함.

VBO (Vertex Buffer Object) – 정점자료(즉, position, color, normal, 등등)의 데이터를 저장하는 메모리 버퍼. 대용량 자료를 GPU에 보내줄 수 있음.

IBO (Index Buffer Object) – 인덱스자료(즉, 정점의 index) 데이터를 저장하는 버퍼

4) **OpenGL에서 glDrawArray 와 glDrawElements를 사용하여 그릴 때의 차이점을 4개의 정점으로 이루어진 사각형 (Quad)를 예로 들어 자세히 설명하라.** (5점)

glDrawArrays(GL_TRIANGLES, 0, 6) - Vertex 정보만을 사용해서 그리는 방식.
즉, 사각형을 그리기 위해서, 4개의 정점 정보를 3개씩 2개의 삼각형에 대한 정점리스트(즉 6개)를 사용하여 그림을 그리는 방식.

glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0) - Vertex와 Index를 동시에 사용해서 그리는 방식.
즉, 사각형을 그리기 위해서, 4개의 정점 정보를 가지고 있는 정점리스트(즉 4개)와 3개씩 2개의 삼각형에 대한 인덱스 리스트(즉 6개)를 사용하여 그림을 그리는 방식.

학과 _____ 학번 _____ 이름 _____

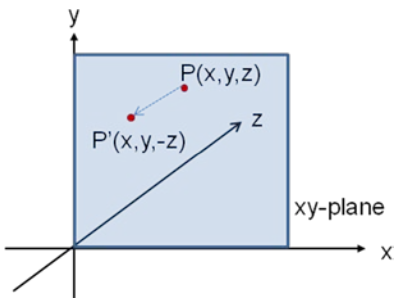
5) **Model-View-Projection Matrix가 무엇인지 설명하라. 각각의 행렬을 생성하는 대표적인 glm 함수를 적어라.** (10점)

Model Transformation Matrix - 모델의 정점 정보를 Model Coordinate System에서 World Coordinate System으로 변환
glm::translate(dx, dy, dz)
glm::rotate(angle, ax, ay, az)
glm::scale(sx, sy, sz)

View Transformation Matrix - World Coordinate System으로 변환된 모델의 정점들을 View Coordinate System (카메라 시점)으로 변환
glm::lookat(g_eye, g_at, g_up)

Projection Transformation Matrix - View Coordinate System (카메라 시점)으로 변환된 모델의 정점들을 Projection Plane에 투영시킨 점들로 변환
Glm::perspective(g_fovy, g_aspect, g_zNear, g_zFar..)

2. 다음은 3차원 그래픽에서 점 P(x, y, z)가 표준 좌표 평면인 xy-plane, xz-plane, yz-plane 평면에 반사(reflection)된 점 P'을 각각 그림으로 나타내고 그 값을 계산하라. 그리고 3차원 아핀 변환 행렬 (4x4 matrix)인 반사행렬을 유도하라. (15점)



P가 xy-plane에 반사된 점 P' = (x, y, -z)
P가 xz-plane에 반사된 점 P' = (x, -y, z)
P가 yz-plane에 반사된 점 P' = (-x, y, z)
P가 (0,0,0)에 반사된 점 P' = (-x, -y, -z)

$$R_{xy-plane} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_{xz-plane} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_{yz-plane} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

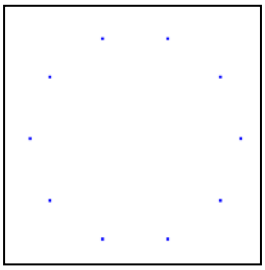
$$R_{(0,0,0)} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

학과 _____ 학번 _____ 이름 _____

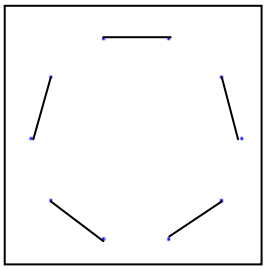
3. 다음 OpenGL 코드는 Circle 클래스의 일부이다. 다음 물음에 답하라.
만약 glDrawArrays(GL_POINTS,...) 대신 GL_LINES, GL_LINE_LOOP, GL_LINE_STRIP, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_POLYGON 을 사용했을 때 실행 결과를 그려라. (15점)

```
void Circle::init() // radius = 0.5 & slices = 10
{
        glm::vec3 vertex;

        float theta = (float) (2*M_PI/slices);
        for (int i=0; i<=slices; i++)
        {
                vertex[0] = p[0] + radius * cosf(theta * i);

                vertex[1] = p[1] + radius * sinf(theta * i);
                vertex[2] = p[2];
                vbo.addData(&vertex, sizeof(glm::vec3));
                vbo.addData(&color, sizeof(glm::vec3));
        }
        numVertices = slices;

         // 중간 생략...
        isLoaded = true;
}

void Circle::draw()
{
        if (!isLoaded) return;
        glBindVertexArray(vao);
        glDrawArrays(GL_POINTS, 0, numVertices);
}
```

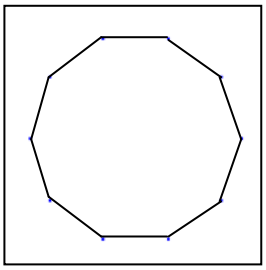
GL_POINTS



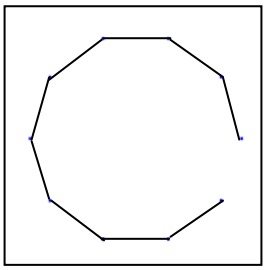
GL_LINES



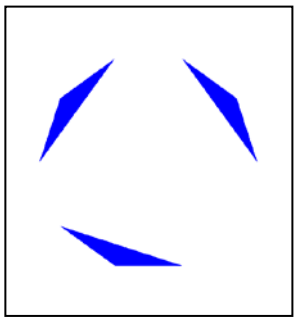
GL_LINE_LOOP



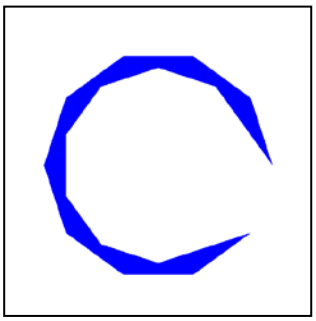
GL_LINE_STRIP



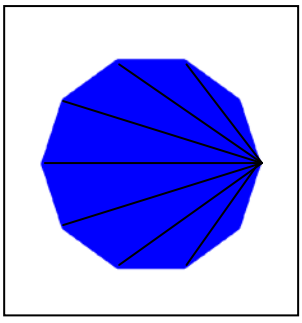
GL_TRIANGLES



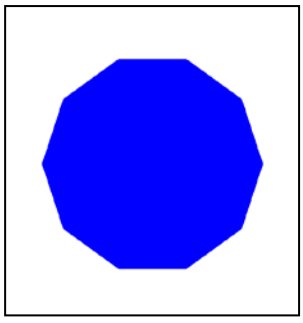
GL_TRIANGLE_STRIP



GL_TRIANGLE_FAN

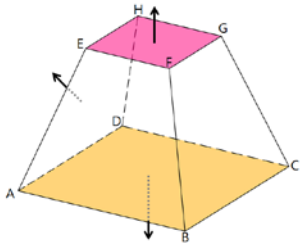


GL_POLYGON



학과 _____ 학번 _____ 이름 _____

4. 다음 OpenGL 코드는 TruncatedPyramid 클래스의 일부이다. 법선벡터 (Normal Vector)의 방향에 맞게 CCW Winding order를 지켜서 빈칸을 채워라. (15점)




```
void TruncatedPyramid::init()
{
        // left face
        vbo.addData(__&D[0]__, sizeof(glm::vec3));
        vbo.addData(__&A[0]__, sizeof(glm::vec3));
        vbo.addData(__&E[0]__, sizeof(glm::vec3));
        vbo.addData(__&D[0]__, sizeof(glm::vec3));
        vbo.addData(__&E[0]__, sizeof(glm::vec3));
        vbo.addData(__&H[0]__, sizeof(glm::vec3));

        // top face
        vbo.addData(__&E[0]__, sizeof(glm::vec3));
        vbo.addData(__&F[0]__, sizeof(glm::vec3));
        vbo.addData(__&G[0]__, sizeof(glm::vec3));
        vbo.addData(__&E[0]__, sizeof(glm::vec3));
        vbo.addData(__&G[0]__, sizeof(glm::vec3));
        vbo.addData(__&H[0]__, sizeof(glm::vec3));

        // bottom face
        vbo.addData(__&C[0]__, sizeof(glm::vec3));
        vbo.addData(__&B[0]__, sizeof(glm::vec3));
        vbo.addData(__&A[0]__, sizeof(glm::vec3));
        vbo.addData(__&C[0]__, sizeof(glm::vec3));
        vbo.addData(__&A[0]__, sizeof(glm::vec3));
        vbo.addData(__&D[0]__, sizeof(glm::vec3));

// 중간생략..
}
```

5. 다음 OpenGL 코드를 보고, 질문에 답을 하시오. (10점)

```
/////// GeometryPositionColor.cpp//////////////
GeometryPositionColor::GeometryPositionColor()
{
// 중간생략..
        start = glm::vec3(-5, 0, 0);
        end = glm::vec3(5, 0, 0);
        duration = 5000;
        active = false;
}
// update는 active가 true하면 duration에 따라 start부터 end까지 중심점 p를 이동
bool GeometryPositionColor::update(float dt)
{
        if (active)     {
                p = start + (end - start) * dt/duration; // 1. 선의 공식 이용하여 코드 작성
                if (dt >= duration) active = false;
                init();
        }
        return true;
}
```

```
// setPosition은 p_로 중심점 p를 지정
bool GeometryPositionColor::setPosition(glm::vec3 p_)
{
	p = p_;
	init();
}

/////// GeometryPrimitiveMouse.cpp//////////////
void init( void )
{
// 중간 생략...
	tri = new Triangle();
	glClearColor( 1.0, 1.0, 1.0, 1.0 ); // white background
}

void display( void )
{
	glClear( GL_COLOR_BUFFER_BIT ); // 2. 매프레임 색 버퍼를 clear color로 지움
	tri->draw();
	glutSwapBuffers(); // 3. 더블 버퍼 사용시 매프레임마다 마지막으로 버퍼 스왑핑
}

void update()
{
	static int currentTime, deltaTime, prevTime = 0;
	static int elapsedTime = 0;
	currentTime = glutGet(GLUT_ELAPSED_TIME);
	deltaTime = currentTime - prevTime;
	prevTime = currentTime;
	elapsedTime = currentTime - startTime;
	tri->update(elapsedTime);
	glutPostRedisplay(); // 4. 정보를 update 하고 displayFunc callback 함수를 호출
}

void motion(int mx, int my)
{
	int w = glutGet(GLUT_WINDOW_WIDTH);
	int h = glutGet(GLUT_WINDOW_HEIGHT);

	// 0~600(x+ right) x 0~600(y+ down) => -5~5(x+ right) x -5~5(y+ up)
	float x = (float) 10 * (mx - w*0.5) / w;
	float y = (float) 10 * (h*0.5 - my) / h; //5. Glut 마우스 좌표계를 OpenGL 좌표계로 변환

	if (g_mousemove) tri->setPosition(glm::vec3(x, y, 0));
	glutPostRedisplay();
}
```

6. 다음 문제에 답하시오. (15점)

1) 다음 GLM 변환 함수가 만들어 내는 3차원 아핀 변환 행렬 (4x4 matrix) M을 구하라.

```
glm::mat4 M1 = glm::translate(glm::mat4(1.0f), glm::vec3(-4, -4, 0));
glm::mat4 M2 = glm::scale(glm::mat4(1.0f), glm::vec3(0.5, 1, 1));
glm::mat4 M3 = glm::rotate(glm::mat4(1.0f), (float)(M_PI/2), glm::vec3(0, 0, 1));
glm::mat4 M4 = glm::translate(glm::mat4(1.0f), glm::vec3(4, 4, 0));
glm::mat4 M = M4 * M3 * M2 * M1;
```

학과 _____ 학번 _____ 이름 _____

$$M = \begin{pmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -4 \\ 0 & 1 & 0 & -4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & -1 & 0 & 8 \\ 0.5 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

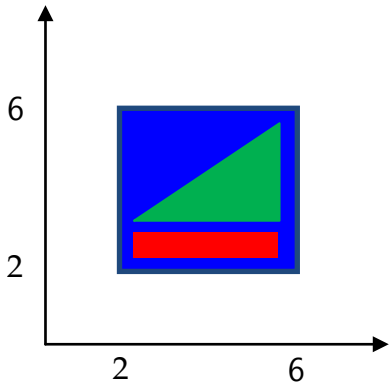
2) 정점 P(2, 2, 0)과 Q(6, 6, 0)을 위의 아핀 변환 행렬 M과 곱해서 나온 P′과 Q′을 구하라.

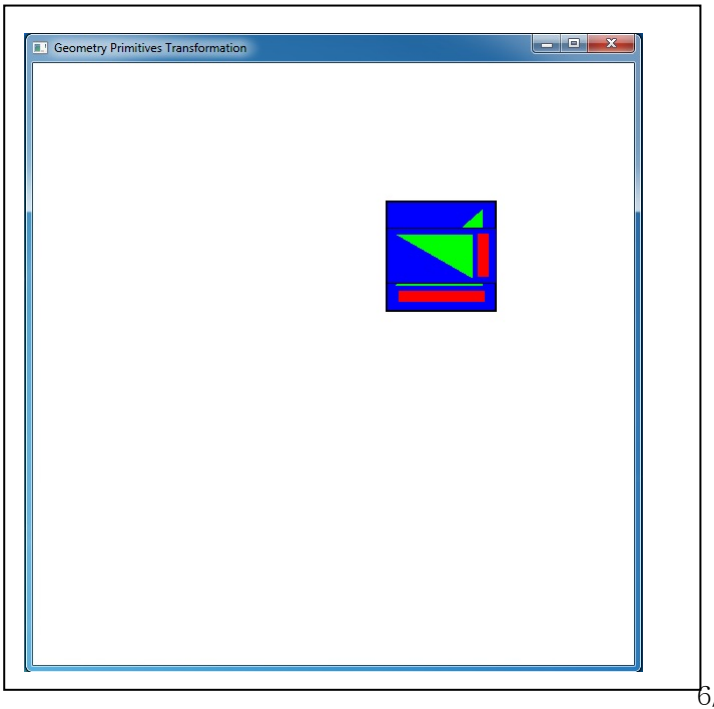
```
glm::vec4 p1 = M * glm::vec4(2, 2, 0, 1);
glm::vec4 q1 = M * glm::vec4(6, 6, 0, 1);
```

$$P' = \begin{pmatrix} 0 & -1 & 0 & 8 \\ 0.5 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 6 \\ 3 \\ 0 \\ 1 \end{pmatrix} \qquad Q' = \begin{pmatrix} 0 & -1 & 0 & 8 \\ 0.5 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 6 \\ 6 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 5 \\ 0 \\ 1 \end{pmatrix}$$

(2, 2, 0) => (6, 3, 0)
(6, 6, 0) => (2, 5, 0)

3) 다음 아래 기본 도형에, 위의 아핀 변환 행렬 **M**을 적용하여 나타난 도형의 모습을 아래 네모 칸 안에 그려서 나타내라. **좌측하단과 우측상단의 점 위치 값을 정확히 명시해 줄 것.**

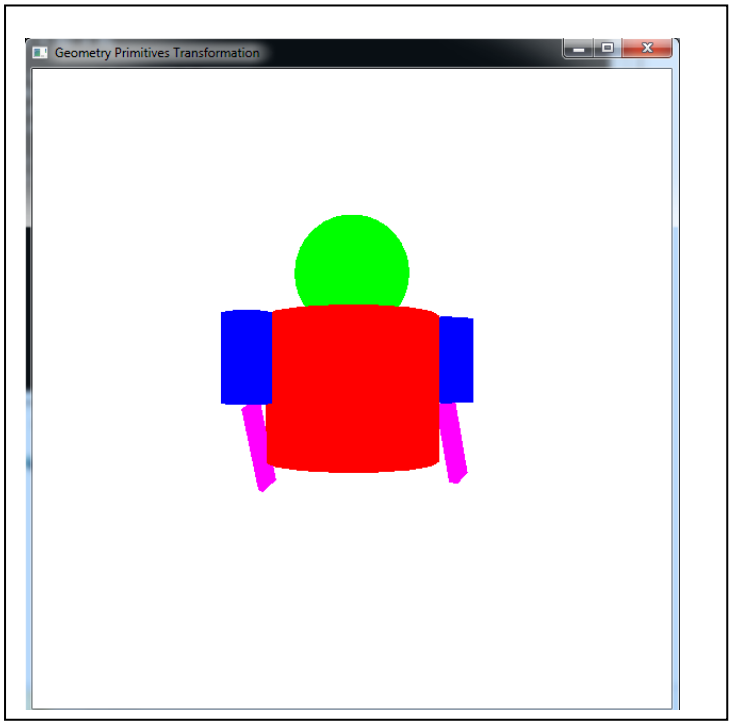




7. HGEO 클래스는 계층적 변환 (hierarchical transformation)을 표현한 구조를 보여주고 있다. 화면에 나타나는 모습을 그리고, 동작이 활성화 (active)되었을 때의 계층적 변환 구조를 설명하라. 계층적 변환 구조에서 중심의 위치를 표시해준다. (extra 10점)

```
HGeo::HGeo()
{
	theta = 0.0f; // 0~360
	phi = 0.0f; // 0~360
	psi = 0.0f; // 0~30
	direction = 1.0f; // +/-1
	active = false;
	init();
}
void HGeo::init()
{
	geo[0] = new Cylinder(glm::vec3(0, 0, 0), 3, 5, 16);
	geo[0]->setColor(glm::vec3(1, 0, 0));
	geo[1] = new Sphere(glm::vec3(0, 0, 0), 2, 16, 16);
	geo[1]->setColor(glm::vec3(0, 1, 0));
	geo[2] = new Cylinder(glm::vec3(0, 0, 0), 1, 3, 8);
	geo[2]->setColor(glm::vec3(0, 0, 1));
	geo[3] = new Parallelepiped(glm::vec3(-0.5f, 0.0f, -0.5f), glm::vec3(1, 0, 0), glm::vec3(0, 0, 1),
glm::vec3(0, -3, 0));
	geo[3]->setColor(glm::vec3(1, 0, 1));

	transform[0] = glm::rotate(glm::mat4(1.0f), theta, glm::vec3(0.0f, 1.0f, 0.0f));
	transform[1] = glm::translate(glm::mat4(1.0f), glm::vec3(0, 4, 0));
	transform[2] = glm::translate(glm::mat4(1.0f), glm::vec3(3.5, 1, 0)) * glm::rotate(glm::mat4(1.0f), phi,
glm::vec3(0.0f, 1.0f, 0.0f));
	transform[3] = glm::translate(glm::mat4(1.0f), glm::vec3(-3.5, 1, 0)) * glm::rotate(glm::mat4(1.0f), phi,
glm::vec3(0.0f, 1.0f, 0.0f));
	transform[4] = glm::translate(glm::mat4(1.0f), glm::vec3(0, -1.5, 0)) * glm::scale(glm::mat4(1.0f),
glm::vec3(0.5f, 1.0f, 0.5f)) * glm::rotate(glm::mat4(1.0f), psi, glm::vec3(0.0f, 0.0f, 1.0f));
}
void HGeo::draw(Program* p, glm::mat4& projection, glm::mat4& view, glm::mat4& model)
{
	p->useProgram();
	p->setUniform("gProjection", projection);
	p->setUniform("gView", view);

	glm::mat4 mMatrix = model * transform[0];
	p->setUniform("gModel", mMatrix);
	geo[0]->draw();

	mMatrix = model * transform[0] * transform[1];
	p->setUniform("gModel", mMatrix);
	geo[1]->draw();

	mMatrix = model * transform[0] * transform[2];
	p->setUniform("gModel", mMatrix);
	geo[2]->draw();

	mMatrix = model * transform[0] * transform[3];
	p->setUniform("gModel", mMatrix);
	geo[2]->draw();
```

```
		mMatrix = model * transform[0] * transform[2] * transform[4];
		p->setUniform("gModel", mMatrix);
		geo[3]->draw();

		mMatrix = model * transform[0] * transform[3] * transform[4];
		p->setUniform("gModel", mMatrix);
		geo[3]->draw();
}
bool HGeo::update(float deltaTime)
{
		if (active)
		{
			theta += 0.001f * deltaTime;
			phi += 0.001f * deltaTime;
			psi += 0.001f * deltaTime * direction;
			if (psi * direction > 1.0)
				direction = -direction;
		}

		transform[0] = glm::rotate(glm::mat4(1.0f), theta, glm::vec3(0.0f, 1.0f, 0.0f));
		transform[1] = glm::translate(glm::mat4(1.0f), glm::vec3(0, 4, 0));
		transform[2] = glm::translate(glm::mat4(1.0f), glm::vec3(3.5, 1, 0)) * glm::rotate(glm::mat4(1.0f), phi,
glm::vec3(0.0f, 1.0f, 0.0f));
		transform[3] = glm::translate(glm::mat4(1.0f), glm::vec3(-3.5, 1, 0)) * glm::rotate(glm::mat4(1.0f), phi,
glm::vec3(0.0f, 1.0f, 0.0f));
		transform[4] = glm::translate(glm::mat4(1.0f), glm::vec3(0, -1.5, 0)) * glm::scale(glm::mat4(1.0f),
glm::vec3(0.5f, 1.0f, 0.5f)) * glm::rotate(glm::mat4(1.0f), psi, glm::vec3(0.0f, 0.0f, 1.0f));

		return true;
}
```



모델 전체는 빨간색 body에 따라 y-축을 중심으로 360도 회전

초록색 head는 body 중심으로부터 이동 배치

파란색 left & right upper arm은 body 중심으로부터 이동배치, 추가적으로 y-축을 중심으로 360도 회전

보라색 left & right lower arm은 각각의 upper arm 중심으로부터 이동배치, x,z로 반으로 크기조정, z-축을 중심으로 약간 왔다갔다 회전

- 끝 -