HAI719I – Programmation 3D

TP Transformations

Andrew Mansour

1 Question 1

1.1 Question 1.1

Ajoutez une variable uniforme de type mat4 au vertex shader représentant la matrice de transformation à appliquer aux sommets. Construire une matrice identité.

1.1.1 Vertex Shader

```
uniform mat4 transform;
void main(){
        // TODO : Output position of the vertex, in clip space : MVP * position
        gl_Position = transform * vec4(vertices_position_modelspace,1);
}
```

Listing 1: vertex shader code

1.1.2 Fonction Draw dans tp.cpp

```
    glm::mat4 transform = glm::mat4(1.0f);
```

Listing 2: Code ajouté a la fonction Draw

1.2 Question 1.2

Faites-en sorte que les effets de zoom et de translation du TP précédent (avec contrôle au clavier) fonctionnent à nouveaux via cette matrice de transformation.

1.2.1 Modification du Draw

```
void draw () {
    glUseProgram(programID);
    // Model matrix : an identity matrix (model will be at the origin) then change
    glm::mat4 transform = glm::mat4(1.0f);

    // in the "Model View Projection" to the shader uniforms

    transform = glm::scale(transform,glm::vec3(0.5f*zoom,0.5f*zoom,0.5f*zoom));
    GLuint transformLoc = glGetUniformLocation(programID,"transform");
    transform = glm::translate(transform,position);
    glUniformMatrix4fv(transformLoc,1,GL_FALSE,&transform[0][0]);
    // 1rst attribute buffer : vertices
    glEnableVertexAttribArray(0);
    glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
    glVertexAttribPointer(
                0,                  // attribute
                3,                  // size
                GL_FLOAT,           // type
                GL_FALSE,           // normalized?
```

```
                0,                  // stride
                (void*)0            // array buffer offset
                );

    // Index buffer
    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, elementbuffer);

    // Draw the triangles !
    glDrawElements(
                GL_TRIANGLES,      // mode
                indices.size(),    // count
                GL_UNSIGNED_SHORT,   // type
                (void*)0           // element array buffer offset
                );

    // Afficher une seconde chaise

    // Afficher une troisieme chaise!

    glDisableVertexAttribArray(0);
}
```

Listing 3: Draw Method

1.2.2 Modification du switch

```
    case '+':
        zoom += 0.1;
        break;
    case '-':
        if(zoom > 0.1){
            zoom -= 0.1;
        }
        break;
    case '8':
        position+=glm::vec3(0.,0.1f,0.);
        break;
    case '6':
        position+=glm::vec3(0.1f,0.,0.);
        break;
    case '4':
        position+=glm::vec3(-0.1f,0.,0.);
        break;
    case '2':
        position+=glm::vec3(0.,-0.1f,0.);
        break;
```

Listing 4: Suite du Switch case

1.2.3 Rendu

Figure 1: Exemple du zoom

Figure 2: Exemple du dezoom

Figure 3: Exemple de deplacement avec transform

1.3 Question 1.3

1.3.1 Code Modifié

```
// Afficher une seconde chaise
glm::mat4 transform2 = glm::mat4(1.0f);
transform2 = glm::scale(transform2,glm::vec3(0.5f*zoom,0.5f*zoom,0.5f*zoom));
transform2 = glm::translate(transform2,glm::vec3(-1.f,-2.f,0.f));
glUniformMatrix4fv(transformLoc,1,GL_FALSE,&transform2[0][0]);
glDrawElements(
        GL_TRIANGLES,      // mode
        indices.size(),    // count
        GL_UNSIGNED_SHORT,   // type
        (void*)0           // element array buffer offset
        );

// Afficher une troisieme chaise!
glm::mat4 transform3 = glm::mat4(1.0f);
transform3 = glm::scale(transform3,glm::vec3(-0.5f*zoom,0.5f*zoom,0.5f*zoom));
transform3 = glm::translate(transform3,glm::vec3(-1.f,-2.f,0.f));
glUniformMatrix4fv(transformLoc,1,GL_FALSE,&transform3[0][0]);
glDrawElements(
            GL_TRIANGLES,      // mode
```

```
            indices.size(),     // count
            GL_UNSIGNED_SHORT,   // type
            (void*)0            // element array buffer offset
            );

//Afficher la quatrieme chaise
glm::mat4 transform4 = glm::mat4(1.0f);
transform4 = glm::translate(transform4, glm::vec3(0.0f, 0.5f, 0.0f));
transform4 = glm::rotate(transform4, glm::radians(angle), glm::vec3(0.0f, 0.0f, 1.0f))
    ;
transform4 = glm::translate(transform4, glm::vec3(0.0f, -0.5f, 0.0f));
transform4 = glm::scale(transform4, glm::vec3(1.f * zoom, 1.f * zoom, 1.f * zoom));
glUniformMatrix4fv(transformLoc, 1, GL_FALSE, &transform4[0][0]);
glDrawElements(
    GL_TRIANGLES,      // mode
    indices.size(),    // count
    GL_UNSIGNED_SHORT, // type
    (void*)0           // element array buffer offset
);

glDisableVertexAttribArray(0);
```

Listing 5: Fonction Draw()

1.3.2 Rendu

Figure 4: 1ère chaise

Figure 5: Chaises 1 et 2

Figure 6: Toutes les chaises avec exemple de rotation

1.4 Question 1.4

Changer le model chargé par suzanne.off

1.4.1 Main

```
std::string filename("data/suzanne.off");
```

Listing 6: Modification dans le main

1.4.2 Draw

```
glm::mat4 transform5 = glm::mat4(1.0f);
glm::vec3 repereMonde = glm::vec3(1.f, 1.f, 1.f);
glm::vec3 characterAxis = glm::vec3(0.f, 1.f, 0.f);
glm::vec3 rotationAxis = glm::normalize(glm::cross(repereMonde, characterAxis));
transform5 = glm::rotate(transform5, glm::radians(angle), rotationAxis);
transform5 = glm::scale(transform5, glm::vec3(zoom, zoom, zoom));
glUniformMatrix4fv(transformLoc, 1, GL_FALSE, &transform5[0][0]);
glDrawElements(GL_TRIANGLES, indices.size(), GL_UNSIGNED_SHORT, (void*)0);
```

Listing 7: Modification dans la fonction main

1.4.3 Switch

```
case 'r':
    angle+=1.f;
    if(angle>=360.f) angle = 0.f;
    break;
```

Listing 8: Modification dans le switch

1.4.4 Rendu

Figure 7: Affichage avec modèle Suzanne

2 Question 3

2.1 Question 3.1

Pour passer d'un espace 2D à une "vraie" vue 3D, il nous faut tout d'abord définir une matrice de projection et l'exploiter dans nos shaders.

2.1.1 Matrice de projection

```
// Projection matrix : 45 Field of View, 4:3 ratio, display range : 0.1 unit <-> 100 units
ProjectionMatrix = glm::perspective(glm::radians(initialFoV), 4.0f / 3.0f, 0.1f, 100.0f);
```

Listing 9: Modification dans la fonction draw

2.2 Question 3.2

La seconde étape consiste à contrôler la position et orientation de la caméra via une matrice de vue.

2.2.1 Matrice de vue

```
// View matrix : camera/view transformation lookat() utiliser camera_position camera_target camera_up
ViewMatrix = glm::lookAt(camera_position,camera_target,camera_up);
```

Listing 10: Modification dans la fonction draw

2.2.2 Envoie des matrices au shader

```
GLuint projectionLoc = glGetUniformLocation(programID,"projection");
glUniformMatrix4fv(projectionLoc,1,GL_FALSE,&ProjectionMatrix[0][0]);

GLuint viewLoc = glGetUniformLocation(programID,"view");
glUniformMatrix4fv(viewLoc,1,GL_FALSE,&ViewMatrix[0][0]);
```

Listing 11: Modification dans la fonction draw

2.2.3 Récupération des matrices et test de camera

```
uniform mat4 transform;
uniform mat4 projection;
uniform mat4 view;

void main(){
        gl_Position = projection * view * transform * vec4(
            vertices_position_modelspace,1);
}
```

Listing 12: Modification du Vertex_shader

2.2.4 Rendu

Figure 8: Suzanne avec camera et perspective

2.3 Question 3.3

Vous pouvez zommer et dé-zoomer en utilisant les touches Z et S. Inspirez-vous de ce code pour ajouter les déplacements latéraux.

Je ne suis pas sur avoir bien compris l'énoncé, donc le deplacement de la camera donne des résultats similaires a la rotation de du maillage.

2.3.1 Ajout du déplacement lattéral

```
case 's':
    camera_position -= cameraSpeed * camera_target;
    break;
case 'z':
    camera_position += cameraSpeed * camera_target;
    break;
case 'q':
    camera_position -= glm::normalize(glm::cross(camera_target - camera_position,
        camera_up)) * cameraSpeed;
    break;
case 'd':
    camera_position += glm::normalize(glm::cross(camera_target - camera_position,
        camera_up)) * cameraSpeed;
    break;
```

Listing 13: Modification du switch

2.3.2 Rendu

Figure 9: Maillage tourné avec la touche 'q'

3 Question 4

3.1 Question 4.1

Créer un système solaire minimaliste

Sur cette question, j'ai été bloqué longtemps car je n'avais pas pensé à l'ordre des instructions. Le rotate était mis après le translate ce qui faisait que la terre tournait autour d'elle même.

3.1.1 Création du Soleil et de la Terre

```
// Soleil
glm::mat4 transformSun = glm::mat4(1.0f);
transformSun = glm::translate(transformSun, glm::vec3(0.0f, 0.0f, 0.0f));
transformSun = glm::scale(transformSun, glm::vec3(1.f, 1.f, 1.f));
glUniformMatrix4fv(transformLoc, 1, GL_FALSE, &transformSun[0][0]);
glDrawElements(GL_TRIANGLES, indices.size(), GL_UNSIGNED_SHORT, (void*)0);

// Terre
glm::mat4 transformEarth = glm::mat4(1.0f);
transformEarth = glm::rotate(transformEarth, glm::radians(angle), glm::vec3(0.0f, 1.0f
    , 0.0f));
transformEarth = glm::translate(transformEarth, glm::vec3(2.0f, 0.0f, 0.0f));
transformEarth = glm::scale(transformEarth, glm::vec3(0.3f, 0.3f, 0.3f));
glUniformMatrix4fv(transformLoc, 1, GL_FALSE, &transformEarth[0][0]);
glDrawElements(GL_TRIANGLES, indices.size(), GL_UNSIGNED_SHORT, (void*)0);
```

Listing 14: Modifications de la fonction Draw

3.1.2 Rendu système solaire minimaliste

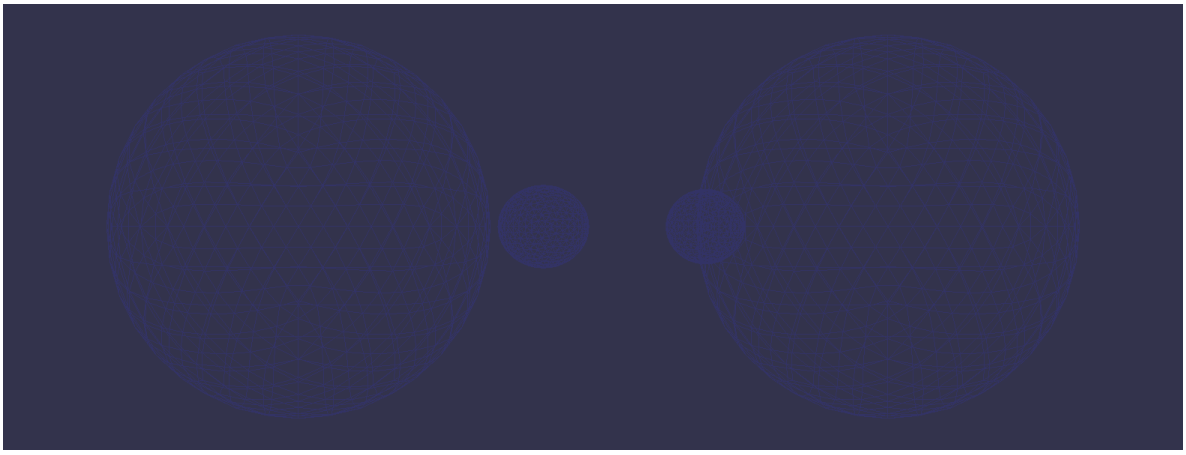

Figure 10: Systeme solaire 1

Figure 11: Systeme solaire 1 avec terre tournée

3.1.3 Idle

La méthode Idle() est modifiée afin de faire une rotation continue sans appuie de touche.

```
void idle () {
    glutPostRedisplay ();
    float time = glutGet(GLUT_ELAPSED_TIME) / 1000.f;
    deltaTime = time - lastFrame;
    lastFrame = time;
    angle+=0.5f;
}
```

Listing 15: Methode Idle

3.2 Question 4.2

Faites en sorte que la terre tourne sur elle-même autour d'un second axe

3.2.1 Modification de la Terre

```
// Terre
glm::mat4 transformEarth = glm::mat4(1.0f);
//tourner autour du soleil
transformEarth = glm::rotate(transformEarth, glm::radians(angle), glm::vec3(0.0f, 1.0f, 0.0f));
transformEarth = glm::translate(transformEarth, glm::vec3(2.0f, 0.0f, 0.0f));
//autour de son axe a 23 degrees
transformEarth = glm::rotate(transformEarth, glm::radians(23.0f), glm::vec3(0.0f, 0.0f, 1.0f));
transformEarth = glm::rotate(transformEarth, glm::radians(angle), glm::vec3(0.0f, 1.0f, 0.0f));

transformEarth = glm::scale(transformEarth, glm::vec3(0.3f, 0.3f, 0.3f));
glUniformMatrix4fv(transformLoc, 1, GL_FALSE, &transformEarth[0][0]);
glDrawElements(GL_TRIANGLES, indices.size(), GL_UNSIGNED_SHORT, (void*)0);
```

Listing 16: Modifications a la methode Draw()

3.2.2 Redu de rotation sur axe de la Terre

Cette rotation ne se voit pas très clairement en quelques images.

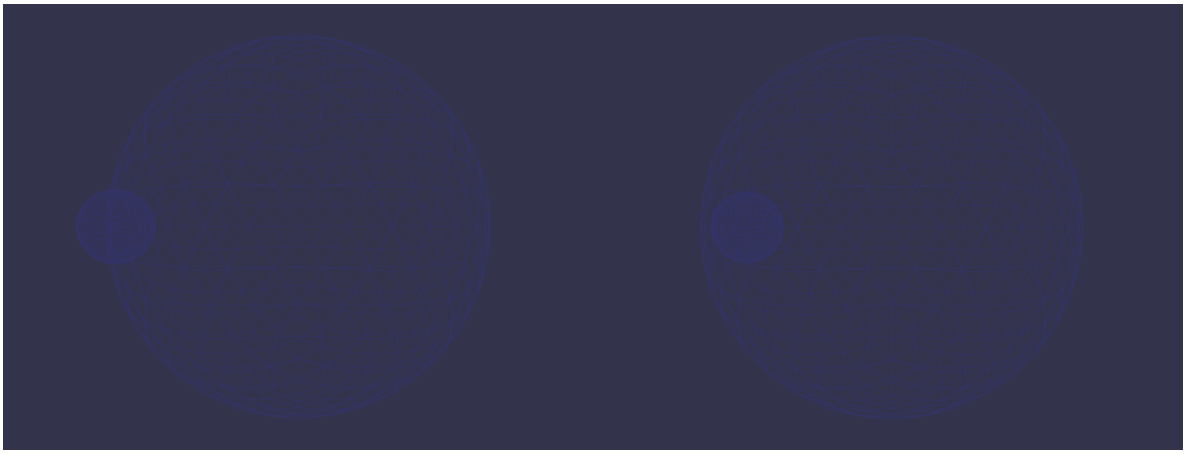

Figure 12: Systeme solaire 1 avec terre tournee sur elle meme

Figure 13: Systeme solaire 1 avec terre tournee sur elle meme 2

3.3 Question 4.3

Ajoutez une lune tournant autour de la terre

Je n'ai pas correctement adapté la distance Terre-Lune.

3.3.1 Création de la Lune

```
// Lune
glm::mat4 transformMoon = glm::mat4(1.0f);

//tourner autour de la terre
transformMoon = glm::rotate(transformMoon, glm::radians(angle), glm::vec3(0.0f, 1.0f, 0.0f));
```

```
transformMoon = glm::translate(transformMoon, glm::vec3(2.0f, 0.0f, 0.0f));
transformMoon = glm::rotate(transformMoon, glm::radians(angle * 3.f), glm::vec3(0.0f,
    1.0f, 0.0f));
transformMoon = glm::translate(transformMoon, glm::vec3(0.5f, 0.0f, 0.0f));
//tourner sur elle meme (valeur prise du schema)
transformMoon = glm::rotate(transformMoon, glm::radians(angle * 3.0f), glm::vec3(0.0f,
     1.0f, 0.0f));
transformMoon = glm::scale(transformMoon, glm::vec3(0.1f, 0.1f, 0.1f));
glUniformMatrix4fv(transformLoc, 1, GL_FALSE, &transformMoon[0][0]);
glDrawElements(GL_TRIANGLES, indices.size(), GL_UNSIGNED_SHORT, (void*)0);
```

Listing 17: Modifications a la methode Draw()

3.3.2 Rendu du Systeme Solaire

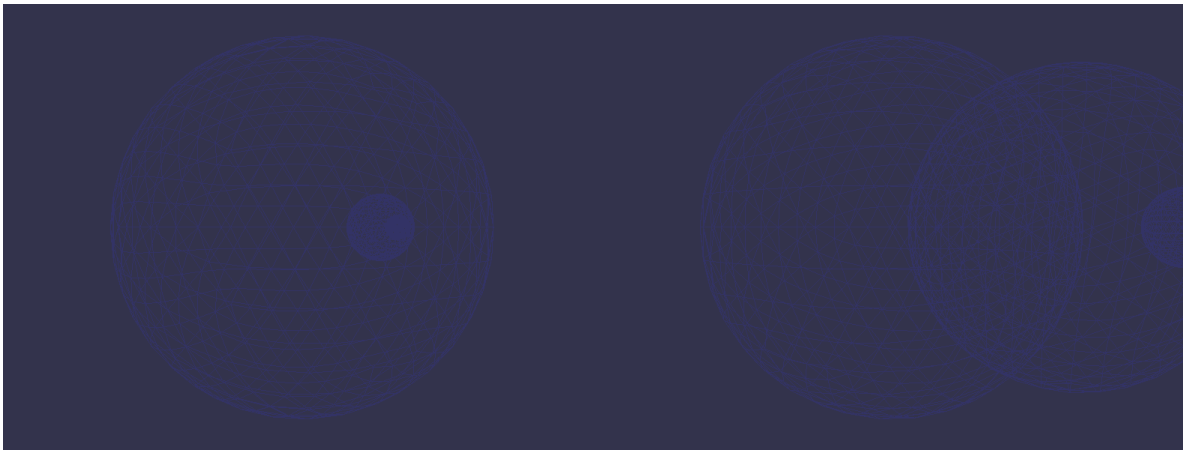

Figure 14: Capture 1 du systeme solaire

Figure 15: Capture 2 du systeme solaire

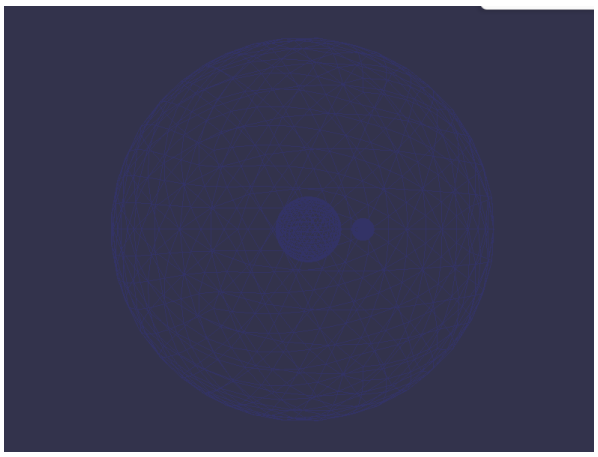

Figure 16: Capture 3 du systeme solaire