PS4 SOLUTIONS

PS4-1 2D transformation of a point and a parabola

(1)

Translation matrix $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}$

Scale matrix $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Rotate matrix $\begin{bmatrix} c45 & -s45 & 0 \\ s45 & c45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

By multiplying these transformation matrices from left to right in the order, we can get the result.

$$\begin{bmatrix} c45 & -s45 & 0 \\ s45 & c45 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 2 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 4 \\ 7 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{-3\sqrt{2}}{2} \\ \frac{11\sqrt{2}}{2} \\ 1 \end{bmatrix}$$

(2)

Translation matrix $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \equiv T$

Rotate matrix $\begin{bmatrix} c(-30) & -s(-30) & 0 \\ s(-30) & c(-30) & 0 \\ 0 & 0 & 1 \end{bmatrix} \equiv R(-30)$

Scale matrix $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \equiv S$

By multiplying these transformation matrices from left to right in the order, we can get the result.

$$SR(-30)T\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Take the inverse to represent x, y by using x' and y'.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \left(SR(-30)T\right)^{-1}\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = T^{-1}R(-30)^{-1}S^{-1}\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Here,

$$T^{-1} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ S^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$R(-30)^{-1} = R(30) = \begin{bmatrix} c(30) & -s(30) & 0 \\ s(30) & c(30) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The result transformation matrix is,

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{4} & -1 \\ \frac{1}{2} & \frac{\sqrt{3}}{4} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$x = \frac{\sqrt{3}}{2}x' - \frac{1}{4}y' - 1, \quad y = \frac{1}{2}x' + \frac{\sqrt{3}}{4}y'$$

By plugging these into the given equation $y = x^2$, we can get the solution.

$$\frac{1}{2}x' + \frac{\sqrt{3}}{4}y' = \left(\frac{\sqrt{3}}{2}x' - \frac{1}{4}y' - 1\right)^2$$

PS4-2 Homogeneous geometric/coordinate transformation

(1)

$$R = \begin{bmatrix} \cos\frac{\pi}{6} & -\sin\frac{\pi}{6} & 0 & 0 \\ \sin\frac{\pi}{6} & \cos\frac{\pi}{6} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad T = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = TR \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

By using these, we can express x, y, z by using x', y', z' as follows.

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = R^{-1}T^{-1} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\frac{\pi}{6} & \sin\frac{\pi}{6} & 0 & 0 \\ -\sin\frac{\pi}{6} & \cos\frac{\pi}{6} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 & -\frac{\sqrt{3}}{2} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

So,

$$x = \frac{\sqrt{3}}{2}x' + \frac{1}{2}y' - \frac{\sqrt{3}}{2}$$

$$y = -\frac{1}{2}x' + \frac{\sqrt{3}}{2}y' + \frac{1}{2}$$

$$z = z'$$

Plug these into the original plane equation.

$$x+y+z+1=\frac{\sqrt{3}-1}{2}x'+\frac{1+\sqrt{3}}{2}y'+\frac{1-\sqrt{3}}{2}+z'+1=0$$

$$\boxed{\frac{\sqrt{3}-1}{2}x'+\frac{1+\sqrt{3}}{2}y'+\frac{1-\sqrt{3}}{2}+z'+1=0}$$

(2)

Transformation matrices

$${}^{W}A=\begin{bmatrix}0&0&-1&4\\0&-1&0&2\\-1&0&0&2\\0&0&0&1\end{bmatrix},\quad {}^{W}B=\begin{bmatrix}-\frac{\sqrt{2}}{2}&-\frac{\sqrt{2}}{2}&0&3\\\frac{\sqrt{2}}{2}&-\frac{\sqrt{2}}{2}&0&6\\0&0&1&0\\0&0&0&1\end{bmatrix},\quad {}^{W}C=\begin{bmatrix}-\frac{\sqrt{2}}{2}&-\frac{\sqrt{2}}{2}&0&3\\-\frac{\sqrt{2}}{2}&\frac{\sqrt{2}}{2}&0&6\\0&0&-1&4\\0&0&0&1\end{bmatrix}$$

Transformation from Cube A to Cube B: ${}^{A}B$

$${}^{A}B={}^{A}W{}^{W}B=\left[{}^{W}A\right]^{-1}{}^{W}B=\begin{bmatrix}0&0&-1&4\\0&-1&0&2\\-1&0&0&2\\0&0&0&1\end{bmatrix}^{-1}\begin{bmatrix}-\frac{\sqrt{2}}{2}&-\frac{\sqrt{2}}{2}&0&3\\\frac{\sqrt{2}}{2}&-\frac{\sqrt{2}}{2}&0&6\\0&0&1&0\\0&0&0&1\end{bmatrix}=\begin{bmatrix}0&0&-1&2\\-\frac{\sqrt{2}}{2}&\frac{\sqrt{2}}{2}&0&-4\\\frac{\sqrt{2}}{2}&\frac{\sqrt{2}}{2}&0&1\\0&0&0&1\end{bmatrix}$$

Transformation form Cube B to Cube C: ${}^{B}C$

$${}^{B}C={}^{B}W{}^{W}C=\left[{}^{W}B\right]^{-1}{}^{W}C=\begin{bmatrix}-\frac{\sqrt{2}}{2}&-\frac{\sqrt{2}}{2}&0&3\\\frac{\sqrt{2}}{2}&-\frac{\sqrt{2}}{2}&0&6\\0&0&1&0\\0&0&0&1\end{bmatrix}^{-1}\begin{bmatrix}-\frac{\sqrt{2}}{2}&-\frac{\sqrt{2}}{2}&0&3\\-\frac{\sqrt{2}}{2}&\frac{\sqrt{2}}{2}&0&6\\0&0&-1&4\\0&0&0&1\end{bmatrix}=\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&-1&4\\0&0&0&1\end{bmatrix}$$

Transformation from Cube C to Cube A: ${}^{C}A$

$${}^{C}A={}^{C}W{}^{W}A=\left[{}^{W}C\right]^{-1}{}^{W}A=\begin{bmatrix}-\frac{\sqrt{2}}{2}&-\frac{\sqrt{2}}{2}&0&3\\-\frac{\sqrt{2}}{2}&\frac{\sqrt{2}}{2}&0&6\\0&0&-1&4\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&0&-1&4\\0&-1&0&2\\-1&0&0&2\\0&0&0&1\end{bmatrix}=\begin{bmatrix}0&\frac{\sqrt{2}}{2}&\frac{\sqrt{2}}{2}&\frac{3\sqrt{2}}{2}\\0&-\frac{\sqrt{2}}{2}&\frac{\sqrt{2}}{2}&\frac{5\sqrt{2}}{2}\\1&0&0&2\\0&0&0&1\end{bmatrix}$$

PS4-3 Converting a polygon file to a VRML file

Next problem solution includes reading .dat file and writing the data into vrml file. Please refer to the solution for PS4-4.

PS4-4 Rotation and scaling of an object

MATLAB Code:

```
%%
% ps3-1
clear all
close all
clc
str = 'triceratops.dat';
%str = 'shape.dat';
%str = 'cube.dat';
fid = fopen(str, 'r'); % for triceratops.data


Vertex=[]; % initialize vertex container matrix
Order=[]; % initialize vertex order container matrix
while (1)
   Line = fgetl(fid); % read a line from the file
   if (Line == -1) break;  end % if the line is end, terminate this loop
   Line = strread(Line,'%s'); % read the first character
   if( strcmp(Line{1}, 'v')) % if the character is v, read the remaining information as a number and
store it to the vertex matrix
       Vertex = [Vertex ; str2num(Line{2}), str2num(Line{3}), str2num(Line{4})];
   elseif( strcmp(Line{1}, 'f')) % if the character is f,
       Numbers = { Line{2:end} }; % change the string cell matrix to string array.
       Data = []; % initialize the Data array
       for i=1:size(Numbers,2) % iterate i to the number of remaining ordering numbers, read remaining
information as a number
          Data=[Data,str2num(Numbers{i})];
       end
       %Data = Data+1; % the index of facet starts from 0. So, modify the indeces by adding one
       Order = [Order ; {Data}]; % make the vertex order
   end
end
fclose(fid); % reading file ends. close the file.
```

```
fid = fopen('triceratops.wrl', 'w');
fprintf(fid, '#VRML V2.0 utf8\n');
fprintf(fid, 'Background{ skyColor 1 1 1 }\n');
fprintf(fid, 'DEF Axes Shape{\n');
fprintf(fid, '              geometry Extrusion\n');
fprintf(fid, '              {\n');
fprintf(fid, '                 spine[0 0 0, 10 0 0, 10 0 0, 11 0 0]\n');
fprintf(fid, '                 crossSection[0.5 0.5, 0.5 -0.5, -0.5 -0.5, -0.5 0.5, 0.5 0.5]\n');
fprintf(fid, '                 scale[0.1 0.1, 0.1 0.1, 0.3 0.3, 0 0]\n');
fprintf(fid, '              }\n');
fprintf(fid, '              appearance Appearance{material Material{diffuseColor 0 0 0}}\n');
fprintf(fid, '            }\n');
fprintf(fid, 'Transform{\n');
fprintf(fid, '          rotation  0 0 1 1.570796327\n');
fprintf(fid, '          children[ USE Axes]\n');
fprintf(fid, '        }\n');
fprintf(fid, 'Transform{\n');
fprintf(fid, '          rotation  0 1 0 -1.570796327\n');
fprintf(fid, '          children[ USE Axes]\n');
fprintf(fid, '        }\n');
fprintf(fid, 'Transform{\n');
fprintf(fid, '          translation  5 10 0\n');
fprintf(fid, '          children[Shape{geometry Text{\n');
fprintf(fid, '                                string["Sooho Park"]\n');
fprintf(fid, '                                fontStyle DEF Fonts \n');
fprintf(fid, '                                FontStyle{\n');
fprintf(fid, '                                          family "SANS"\n');
fprintf(fid, '                                          justify "MIDDLE"\n');
fprintf(fid, '                                          style "BOLD"\n');
fprintf(fid, '                                          size 1\n');
fprintf(fid, '                                       }\n');
fprintf(fid, '                                  }\n');
fprintf(fid, '                        appearance DEF colors Appearance{material Material{diffuseColor 0 0
0}}\n');
fprintf(fid, '                       }\n');
fprintf(fid, '                  ]\n');
```

```
fprintf(fid, '       }\n');
%%
fprintf(fid, 'Shape{ \n');
fprintf(fid, '\t geometry IndexedFaceSet{ \n');
fprintf(fid, '\t\t coord Coordinate{ point[\n');

for i=1:size(Vertex,1)
    fprintf(fid, '\t\t\t %f %f %f,\n', Vertex(i,1), Vertex(i,2), Vertex(i,3));
end

fprintf(fid, '\t\t] }\n');
fprintf(fid, 'coordIndex[\n');
for i = 1:size(Order,1) % check how many the point is. iterate this loop
    Polygon = Order(i); % read a cell of a polygon
    VertexOrder = Polygon{1}(1:size(Polygon{1},2)); % read the vertex order
    fprintf(fid, '\t\t\t');
    for j=1:size(VertexOrder,2)
        fprintf(fid, '%d, ', VertexOrder(j));
    end
    fprintf(fid, '-1, ');
    fprintf(fid,'\n');
end
fprintf(fid,'\t\t]\n');
fprintf(fid,'\t}\n');
fprintf(fid,'appearance Appearance{ material Material{ diffuseColor 0 0 1}}\n');
fprintf(fid,'}\n');

%%
fprintf(fid, 'Shape{ \n');
fprintf(fid, '\t geometry IndexedFaceSet{ \n');
fprintf(fid, '\t\t coord Coordinate{ point[\n');
theta = pi/4;
c = cos(theta);
s = sin(theta);
t = 1-c;
axis = [1,1,1];
```

```
x = axis(1)/norm(axis);
y = axis(2)/norm(axis);
z = axis(3)/norm(axis);

RotMatrix = [ t*x*x+c,t*x*y-z*s,t*x*z+y*s,0;
              t*x*y+z*s,t*y*y+c,t*y*z-x*s,0;
              t*x*z-y*s,t*y*z+x*s,t*z*z+c,0;
              0,0,0,1
          ];
ScaleMatrix = [2,0,0,0;
               0,1,0,0;
               0,0,1,0;
               0,0,0,1
               ];
for i=1:size(Vertex,1)
   Vertex2 =  ScaleMatrix * RotMatrix * [Vertex(i,:),1]';
   fprintf(fid, '\t\t\t %f %f %f,\n', Vertex2(1), Vertex2(2), Vertex2(3));
end

fprintf(fid, '\t\t] }\n');
fprintf(fid, 'coordIndex[\n');
for i = 1:size(Order,1) % check how many the point is. iterate this loop
   Polygon = Order(i); % read a cell of a polygon
   VertexOrder = Polygon{1}(1:size(Polygon{1},2)); % read the vertex order
   fprintf(fid, '\t\t\t');
   for j=1:size(VertexOrder,2)
      fprintf(fid, '%d, ', VertexOrder(j));
   end
   fprintf(fid, '-1, ');
   fprintf(fid,'\n');
end
fprintf(fid,'\t\t]\n');
fprintf(fid,'\t}\n');
fprintf(fid,'appearance Appearance{ material Material{ diffuseColor 1 0 1}}\n');
fprintf(fid,'}\n');
fclose(fid);
```

C++ Code:

```
// ps3.cpp by Arbtip Dheeravongkit

#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <string.h>
#include "ps3.h"

int main()
{
    //Open and read data from file
    ifstream dataFile("triceratops.dat");
    if ( !dataFile ) {
        cerr << "File could not be opened\n";
        exit (1);
    }

    //Open output files
    ofstream outFile;
    outFile.open ("triceratops.wrl", ios::out);
    if (!outFile) {
        cerr << "Cannot open this file for output.\n";
        exit (-1);
    }

    ofstream outFile2;
    outFile2.open ("triceratops2.wrl", ios::out);
    if (!outFile2) {
        cerr << "Cannot open this file for output.\n";
        exit (-1);
    }

    //Draw axes on vrml output files
    vrml_axes (outFile, MY_NAME);
    vrml_axes (outFile2, MY_NAME);


    //Declare and initialize variables
    double dataVertex[2900][4];     //Matrix contains coordinates of vertices
    int dataFace [2900][15];   //Matrix contains coordIndex of faces

    for (int x=0; x<2900; x++)
        for(int y=0; y<14; y++)
            dataFace[x][y] = 0;

    char buffer[100];
    char *tokenPtr;
    double test;
    int vIndex;
    int j=0;
    int b=0;
    double U[3];                //Unit vector to rotate about
    double R[4][4];                 //Rotation matrix
    double S[4][4];                 //Scale matrix
    double RS[4][4];            //Rotation then Scale matrix

    //Assign values to matrices and vector
    U[0] = U[1] = U[2] =1.0/(sqrt(3.0));

    R[0][0] = cos(angle) + (U[0]*U[0]*(1-cos(angle)));
    R[0][1] = (-U[2]*sin(angle)) + (U[0]*U[1]*(1-cos(angle)));
    R[0][2] = (U[1]*sin(angle)) + (U[2]*U[0]*(1-cos(angle)));
    R[0][3] = 0.0;
    R[1][0] = (U[2]*sin(angle)) + (U[0]*U[1]*(1-cos(angle)));
    R[1][1] = cos(angle) + (U[1]*U[1]*(1-cos(angle)));
    R[1][2]  = (-U[0]*sin(angle)) + (U[1]*U[2]*(1-cos(angle)));
    R[1][3]  = 0.0;
    R[2][0]  = (-U[1]*sin(angle)) + (U[2]*U[0]*(1-cos(angle)));
    R[2][1] = (U[0]*sin(angle)) + (U[1]*U[2]*(1-cos(angle)));
    R[2][2] = cos(angle) + (U[2]*U[2]*(1-cos(angle)));
    R[2][3] = 0.0;
```

```
R[3][0] = R[3][1] = R[3][2] = 0.0;
R[3][3] = 1.0;

for(int s=0;s<4;s++) {
    for(int t=0;t<4;t++)
        S[s][t] = 0.0;
}
S[0][0] = 2.0;
S[3][3] = S[1][1] = S[2][2] = 1.0;

//Rotate-then-Scale matrix
for(int l=0; l< 4; l++) {
    for(int m=0; m<4; m++) {
        RS[l][m]=0;
        for (int n=0; n<4; n++)
            RS[l][m] += R[l][n]*S[n][m];
    }
}


//Read values from input file
while (!dataFile.eof())
{
    dataFile.getline(buffer,100);                          //Reading by line
    tokenPtr = strtok(buffer," ");

        int i=0;
        if (tokenPtr[0] != 'f') {                          //Read coordinates of vertices
            while (tokenPtr != NULL) {                     //and store values in a matrix dataVertex
                if (tokenPtr[0]!='v') {
                    test = atof(tokenPtr);
                    dataVertex[j][i] = test;
                    i++;
                }
                tokenPtr = strtok(NULL," ");
            }
            dataVertex[j][3] = 1.0;
            j++;
        }

        else {                                                 //Read face data
            int a=0;                                       //and store values in a matrix dataFace
            while (tokenPtr!=NULL) {
                if (tokenPtr[0]!='f') {
                    vIndex = atoi(tokenPtr);
                    dataFace[b][a+1] = vIndex;
                    a++;
                }
                tokenPtr = strtok(NULL," ");
            }
            dataFace[b][0] = a;
            b++;
        }
        //Read the next line from input file
}

int numVertex = j;                                         //Total number of vertices
int numFace = b;                                       //Total number of faces

//Transform coordinates
double transf[2900][4];
for(int f=0; f< numVertex; f++) {
    for(int g=0; g<4; g++) {
        transf[f][g]=0;
        for (int h=0; h<4; h++)
            transf[f][g] += dataVertex[f][h]*RS[h][g];
    }
}

//Draw Faces on output vrml files
vrml_drawFace (outFile, dataFace, dataVertex, numFace, numVertex, 1, 0);    //Original on first output
vrml_drawFace (outFile2, dataFace, dataVertex, numFace, numVertex, 1, 0);   //Original on second output
vrml_drawFace (outFile2, dataFace, transf, numFace, numVertex, 0, 1);       //Transformed on second out
```

```
R[3][0] = R[3][1] = R[3][2] = 0.0;
R[3][3] = 1.0;

for(int s=0;s<4;s++) {
    for(int t=0;t<4;t++)
        S[s][t] = 0.0;
}
S[0][0] = 2.0;
```

```
//vrml.cpp by Arbtip Dheeravongkit

#include "ps3.h"

void vrml_axes (ofstream &outFile, char* my_name) {
// Draw in the output vrml file, background, heading, and axis
//  outFile: pointer to the output vrml file
//  my_name: my name to be shown in the heading

    // Create Header
    outFile << "#VRML V2.0 utf8\n";
    outFile << "#-----------------------------------------------\n";
    outFile << "# 24-786: Geometric Modeling\n";
    outFile << "# "<<MY_NAME<<"\n";
    outFile << "# Transforming Triceratops\n";
    outFile << "#-----------------------------------------------\n\n";

    // Set up background
    outFile << "# Specify the background color\n";
    outFile << "#----------------------------------\n";
    outFile << "Background {\n\tskyColor 1 1 1\n}\n\n"; // White background

    // Put text heading on the VRML screen
    outFile << "# Draw texts\n";
    outFile << "#----------------------------------\n";
    outFile << "Transform {\n";
    outFile << "\ttranslation 0 13 0\n";
    outFile << "\tchildren Shape {\n";
    outFile << "\t\tappearance Appearance {\n";
    outFile << "\t\t\tmaterial Material {\n";
    outFile << "\t\t\t\tdiffuseColor 0 0 0\n";
    outFile << "\t\t\t}\n\t\t}\n";
    outFile << "\t\tgeometry Text {\n";
    outFile << "\t\t\tstring [\"24-786: Geometric Modeling\",\n";
    outFile << "\t\t\t\"Transforming Triceratops\",\n";
    outFile << "\t\t\t\""<<MY_NAME<<"\"]\n";
    outFile << "\t\t\tfontStyle FontStyle {\n";
    outFile << "\t\t\t\tfamily \"SANS\"\n";
    outFile << "\t\t\t\tjustify \"MIDDLE\"\n";
    outFile << "\t\t\t\tstyle \"BOLD\"\n";
    outFile << "\t\t\t\tsize 1\n\t\t\t}\n\t\t}\n\t}\n}\n\n";

    // Add the x-axis
    outFile << "# Define the x axis\n";
    outFile << "#----------------------------------\n";
    outFile << "Transform {\n";
    outFile << "\ttranslation 0 0 0\n";
    outFile << "\tchildren DEF AXIS Shape {\n";
    outFile << "\t\tappearance Appearance {\n";
    outFile << "\t\t\tmaterial Material {\n";
    outFile << "\t\t\t\tdiffuseColor 0.0 0.0 0.0\n\t\t\t}\n\t\t}\n";
    outFile << "\t\tgeometry Extrusion {\n";
    outFile << "\t\t\tcrossSection [2 0, 0 -2, -2 0, 0 2, 2 0]\n";
    outFile << "\t\t\tscale [0.035 0.035, 0.035 0.035, 0.15 0.15, 0 0]\n";
    outFile << "\t\t\tspine [-10 0 0, 8 0 0, 8 0 0, 10.5 0 0]\n\t\t}\n\t}\n}\n\n";

    // Add the y-axis
    outFile << "# Draw the y axis\n";
    outFile << "#----------------------------------\n";
    outFile << "Transform {\n";
    outFile << "\trotation 0 0 1 1.5707\n";
    outFile << "\tchildren USE AXIS\n}\n\n";

    // Add the z-axis
    outFile << "# Draw the z axis\n";
    outFile << "#----------------------------------\n";
    outFile << "Transform {\n";
    outFile << "\trotation 0 1 0 -1.5707\n";
    outFile << "\tchildren USE AXIS\n}\n\n";
}

void vrml_drawFace (ofstream &outFile2, int M[][15], double N[][4], int numFace, int numVertex, int R, int (
//Draw a faces on output vrml file
```

```
//   outFile: pointer to the output vrml file
//   M: Matrix contains index of vertices of the faces
//   N: Matrix contains coordinates of vertices
//   numFace: Total number of faces
//   numVertex: Total number of vertices
//   R: The first number of RGB
//   G: The second number of RGB


    outFile2 <<   "\tShape{\n";
    outFile2 <<   "\t\tgeometry IndexedFaceSet{\n";
    outFile2 <<   "\t\t\tcoord Coordinate{\n";
    outFile2 <<   "\t\t\t\tpoint [";
    for ( int i=0; i<numVertex ; i++ ) {
        for ( int  j=0; j<3; j++)
            outFile2 <<  N[i][j] << " ";
        if (i!= numVertex-1)
            outFile2 << ", ";
    }

    outFile2 << "]\n";


    outFile2 <<   "\t\t\t}\n";
    outFile2 <<   "\t\t\tcoordIndex[\n";

    for ( int k=0; k< numFace; k++ ) {
        int temp = M[k][0];
        for (int l=0; l<temp; l++)
            outFile2 << M[k][l+1] << ",";
        outFile2 << "-1\n";
    }

    outFile2 <<   "]\n";
    outFile2 <<   "\t\t}\n";
    outFile2 <<   "\t\tappearance Appearance {\n";
    outFile2 <<   "\t\t\tmaterial Material {\n";
    outFile2 <<   "\t\t\t\tdiffuseColor ";
    outFile2 << R << " " << G << " 0\n";
    outFile2 <<   "\t\t\t}\n";
    outFile2 <<   "\t\t}\n";
    outFile2 <<   "\t}\n";
}
```