THE SIMNET ROTATION MATRIX

Technical Report

AD-A241 022

PUBLICATION NUMBER IST-TR-89-4

JORGE CADIZ, RUEY OUYANG, JACK THOMPSON

NETWORKING AND COMMUNICATIONS TECHNOLOGY LABORATORY

The Institute for Simulation and Training
12424 Research Parkway, Orlando, FL 32826

October 8, 1989

University of Central Florida

91-11417
This report presents a description of the nine element (3x3) rotation matrix used in the SIMNET Vehicle Appearance Protocol Data Unit (VA PDU) to describe a vehicle's orientation (attitude) in free space. Also included is a program (written in Microsoft C) which calculates this matrix given a vehicle's pitch, roll, and yaw angles (in degrees).
INTRODUCTION
In this Technical Report we present a description of the nine element (3 x 3) rotation matrix used in the SIMNET Vehicle Appearance Protocol Data Unit (VA PDU) to describe a vehicle's orientation (attitude) in free space. Also included is a program (written in Microsoft C) which calculates this matrix given a vehicle's pitch, roll, and yaw angles (in degrees).

SIMNET COORDINATE SYSTEMS
If we are to understand the vehicle rotation, some knowledge of the SIMNET coordinate systems is necessary. There are two coordinate systems in the SIMNET system; world and vehicle (hull) coordinate systems.

- World Coordinates (Figure 1): The origin of this coordinate system is located at the Southwest corner of the terrain database. The Y-axis points North, the X-axis East, and the Z-axis Up.

![Figure 1. SIMNET World Coordinates](image)

- Hull Coordinates (Figure 2): The Y'-axis points out of the front of the vehicle, the X'-axis points through the right side of the vehicle, and the Z'-axis points through the top of the hull. The Computer Image Generator (CIG) system relies on hull coordinates as a basis for computing all display views.

Figure 2 shows the angles \( p, r, \) and \( y \), which the hull of the vehicle makes with the world coordinate system. These angles correspond to the pitch, roll, and yaw angles (respectively) that define the vehicle's orientation, and are used in the calculation of the rotation matrix described in the subsequent discussion.
The orientation of a vehicle is constantly changing as it traverses the battlefield. As a result, new pitch, roll, and yaw angles must be continually computed and transmitted across the network. The SIMNET Vehicle Appearance PDU requires a nine element rotation matrix to describe the vehicle's orientation.

The method used by SIMNET rotates the vehicle coordinate system from the "old" frame of reference, into the "updated" frame. A maximum of three angle rotations is sufficient to bring any two frames into coincidence. The three rotations are performed about the X, Y, and Z axes using the following matrices:

Rotation about the x-axis: 
\[
X = \begin{bmatrix}
1 & 0 & 0 \\ 
0 & \cos(p) & -\sin(p) \\ 
0 & \sin(p) & \cos(p)
\end{bmatrix}
\]

Rotation about the y-axis: 
\[
Y = \begin{bmatrix}
\cos(r) & 0 & \sin(r) \\ 
0 & 1 & 0 \\ 
-\sin(r) & 0 & \cos(r)
\end{bmatrix}
\]

Rotation about the z-axis: 
\[
Z = \begin{bmatrix}
\cos(y) & -\sin(y) & 0 \\ 
\sin(y) & \cos(y) & 0 \\ 
0 & 0 & 1
\end{bmatrix}
\]

Each matrix represents rotation about a certain axis; "p" is the pitch angle (rotation about the X world coordinate axis), "r" is the roll angle (rotation about the Y-axis), and "y" is the yaw angle (rotation about the Z-axis). All angles are expressed in radians for the matrix calculations. Upon performing matrix multiplication on X, Y, and Z, the resulting matrix is:

\[
R = \begin{bmatrix}
R_{11} & R_{12} & R_{13} \\ 
R_{21} & R_{22} & R_{23} \\ 
R_{31} & R_{32} & R_{33}
\end{bmatrix}
\]
The elements of $R$ are given by:

\[
\begin{align*}
R_{11} &= \cos(r) \cos(y) \\
R_{12} &= -\cos(r) \sin(y) \\
R_{13} &= \sin(r) \\
R_{21} &= \cos(p) \sin(y) + \sin(p) \sin(r) \cos(y) \\
R_{22} &= \cos(p) \cos(y) - \sin(p) \sin(r) \sin(y) \\
R_{23} &= -\sin(p) \cos(r) \\
R_{31} &= \sin(p) \sin(y) - \cos(p) \sin(r) \cos(y) \\
R_{32} &= \sin(p) \cos(y) + \cos(p) \sin(r) \sin(y) \\
R_{33} &= \cos(p) \cos(r)
\end{align*}
\]

**ROTATION PROGRAM**

We generated a Microsoft C program that calculates the rotation matrix $R$. With this program it is possible to input a desired orientation (by providing angles for pitch, roll, and yaw) and have the matrix elements calculated and placed in the matrix array. A copy of this program is provided in Appendix A.

**LOCATION OF ROTATION DATA IN VEHICLE APPEARANCE PDU**

The SIMNET Vehicle Appearance PDU contains the rotation matrix that is broadcast to other SIMNET vehicles on the network. The exact location of the rotation information is in octets 15 through 50 of the VA PDU. The elements of the rotation matrix ($R_{11}, R_{12}, ..., R_{33}$) are placed in order, starting with octets 15 through 18 being $R_{11}$. Each element is four octets long, therefore the matrix adds up to a total of 36 octets (bytes).

<table>
<thead>
<tr>
<th>Element</th>
<th>Octet #</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{11}$</td>
<td>15-18</td>
</tr>
<tr>
<td>$R_{12}$</td>
<td>19-22</td>
</tr>
<tr>
<td>$R_{13}$</td>
<td>23-26</td>
</tr>
<tr>
<td>$R_{21}$</td>
<td>27-30</td>
</tr>
<tr>
<td>$R_{22}$</td>
<td>31-34</td>
</tr>
<tr>
<td>$R_{23}$</td>
<td>35-38</td>
</tr>
<tr>
<td>$R_{31}$</td>
<td>39-42</td>
</tr>
<tr>
<td>$R_{32}$</td>
<td>43-46</td>
</tr>
<tr>
<td>$R_{33}$</td>
<td>47-50</td>
</tr>
</tbody>
</table>

The information given in this report is in concurrence with SIMNET Software Version 5.25. This software is the current version which resides in the SIMNET M1 modules located in the IST1 laboratories.
APPENDIX A

SIMNET ROTATION MATRIX SOFTWARE PROGRAM
/* This program computes the rotation matrix (3x3) for the SIMNET PDU's given the pitch, roll and yaw of the vehicle in degrees. Final results are stored in matrix B. */

main()
{
#include <stdio.h>
#include <math.h>

int i,j,k=0;
float R,P,Y;
float RC,RS,PC,PS,YC,YS;
float A [3][3];
float B [3][3];
float z [3][3];
float x [3][3];
float y [3][3];

/* Input the rotation angles */
printf("\n Enter Roll Pitch Yaw in degrees\n");
scanf("%f %f %f",&R, &P, &Y);
printf("Roll=%5.2f Pitch=%5.2f Yaw=%5.2f\n",R,P,Y);
R=(R*3.14)/180; /* Convert angles from degrees to radians */
P=(P*3.14)/180;
Y=(Y*3.14)/180;

RC=cos(R); /* Calculate sines and cosines */
RS=sin(R);
PC=cos(P);
PS=sin(P);
YC=cos(Y);
YS=sin(Y);

printf("sin R=%7.6f cos R=%7.6f\n",RS,RC);
printf("sin Y=%7.6f cos Y=%7.6f\n",YS,YC);
printf("sin P=%7.6f cos P=%7.6f\n",PS,PC);

z[0] [0]=YC; /* Generate x, y, and z rotation matrices */
z[0] [1]=-YS;
z[0] [2]=0;
z[1] [0]=YS;
z[1] [1]=YC;
z[1] [2]=0;
z[2] [0]=0;
z[2] [1]=0;
z[2] [2]=1;

x[0] [0]=1;
x[0] [1]=0;
x[0] [2]=0;
```c
x[1] [0]=0;
x[1] [1]=PC;
x[1] [2]=-PS;
x[2] [0]=0;
x[2] [1]=PS;
x[2] [2]=PC;

y[0] [0]=RC;
y[0] [1]=0;
y[0] [2]=RS;
y[1] [0]=0;
y[1] [1]=1;
y[1] [2]=0;
y[2] [0]=RS;
y[2] [1]=0;

/* multiply x matrix by y matrix and put results in matrix A */
for (i=0;i<=2;i++){
    for (j=0;j<=2;j++){
        A[i][j]=0;
        for (k=0;k<=2;k++)
            A[i][j] += x[i][k] * y[j][k];
    }
}

/* Display matrix A results */
for (i=0;i<=2;i++){
    for (j=0;j<=2;j++)
        printf("A(%d,%d) = \%5.4f
",i,j,A[i][j]);
    printf("\n");
}

/* Multiply matrix A by z matrix and store in B matrix */
for (i=0;i<=2;i++){
    for (j=0;j<=2;j++){
        B[i][j]=0;
        for (k=0;k<=2;k++)
            B[i][j] += A[i][k] * z[k][j];
    }
}
printf("\n\n");

/* Display final B matrix results */
for (i=0;i<=2;i++){
    for (j=0;j<=2;j++)
        printf("B(%d,%d) = \%5.4f
",i,j,B[i][j]);
    printf("\n");
}
```

A-2