

# Semester Project Linear Algebra, Spring 2004

Instructor: Matthias Heymann

The task is to design a three-dimensional object, to represent it in  $(x, y, z, 1)$ -coordinates, and to apply several linear transformations to it such as turning, mirroring, shearing or shifting the object.

The resulting object is then projected onto the (2-dimensional) paper, and we get a nice picture in the quality of modern computer games.

## 1 Designing Your Object

Take a piece of scratch paper and invent any three dimensional object. The only restriction is of course that all edges have to be straight lines. Try to place the vertices of the object in such a way that all their coordinates are integers (that will make your calculations easier later on).

The center of your object should be (close to) the origin. In that way all turn-, stretch-, shear- and mirroring operations will keep the center at the origin, and it is easier for you to keep track of where your object is. But if you want to place it somewhere else (for demonstration purposes, e.g. to turn around a point other than the origin), that's fine, too.

As a convention, we will use the “right-hand rule” that if the  $x$ -axis points to the right and the  $y$ -axis points up, then the  $z$ -axis points towards you (and not “into the table”).

The object should have about 20 to 30 vertices - the more the better. And don't forget that not every edge has to be parallel to the axes—some diagonal lines will make everything look a lot cooler in the end.

Now write down the coordinates of every vector in the form  $(x, y, z, 1)$  and fill them into the columns of a  $4 \times n$  matrix, where  $n$  is your number of vertices. Call this data matrix  $D$ .

*Examples for objects:* Letters, geometric objects (as complex as possible), houses, spaceships etc. . . Be creative!

## 2 Computing Your Transformation Matrix

Now it is time to make up some series of linear transformations that you apply to your object—not too many, though, just about three or four. The idea is that in the end one can look at the picture and easily see how it must have originated. For example, if you shear it, then mirror it at the

$x$ -axis, then turn it a bit around the  $z$ -axis, then turn it a bit around the  $y$ -axis and in the end shift it to the back a bit, the viewer can look at the picture, compare it with the original one, and still say what transformations you must have used.

To achieve that, keep the following in mind: For turns, use small angles of about 15 to 40 degrees, especially if you apply several turns, otherwise it can get confusing for the viewer. Also don't deform the object too badly (e.g. shear, turn, shear), or otherwise it will just look ugly. Keep it simple, but not too simple—coolness is achieved by complexity.

The last transformation should be a shift “away from you”, that is in negative  $z$ -direction. Remember that the viewer has his eyes on the positive  $z$ -axis, he is looking along the  $z$ -axis towards the origin, and the picture is what he sees through a window in the  $(x, y)$ -plane. Therefore, after the transformations the whole object should have negative  $z$ -values.

Now write down the matrices for all your transformations and multiply them in the right order (e.g., first  $A_1$ , then  $A_2$ , then  $A_3$  corresponds to the matrix product  $A_3A_2A_1!$ ). You can find some matrix “templates” at the end of the document.

Finally write down the projection matrix  $P$  whose template you will also find at the end of the document. In this template,  $d > 0$  determines the position  $(0, 0, d)$  of the viewer's eye position. Keep in mind that the closer the viewer is to the object (that is, the smaller  $d$  is), the stronger is the three-dimensional effect—parallel lines do not appear parallel anymore, but they meet at the horizon (“parallel lines cross at infinity”). If  $d$  is too small, though, it might happen that the viewer does not see most of the lines anymore because they are hidden behind the object. It is maybe a good idea to define your  $d$  to be about 1-2 times the height of your object.

Now multiply your total transformation matrix *from the left* by  $P$  (so you get  $P \cdot A_3A_2A_1$ , for example). Call this matrix  $M$ . It is  $3 \times n$  since  $P$  has only three rows.

### 3 Calculation of the Screen Coordinates

Now calculate the product  $MD$  to get the homogeneous coordinates of the vertices of your transformed object. That means the columns of the  $3 \times n$ -matrix  $MD$  are the coordinates of the transformed vertices, given in homo-

geneous vector form  $(X, Y, H)$ .

That means that the points that you have to plot on your paper are given by  $(\frac{X}{H}, \frac{Y}{H})$ . In other words, you have to take the columns of your matrix  $MD$  and divide their first two entries by their last one.

Write down these 2-dimensional vectors  $(\frac{X}{H}, \frac{Y}{H})$  into the columns of a new  $2 \times n$  matrix  $N$ .

## 4 Plot

Now it is time to draw the final picture. It is smart to use paper with preprinted coordinate grids, as fine as possible. Look at your matrix  $N$  and determine the largest and the smallest  $x$ - and  $y$ -values. Use a sharp pencil to draw your coordinate axes accordingly. Choose the annotations in such a way that the picture will fill out the whole paper. Don't ruin all the work that you did so far by drawing a miniature picture!

Now use a thin black pen to draw all the transformed vertices that you find in the columns of  $N$ . Look at your original picture and find out which vertices are connected by an edge. Carefully connect the corresponding points in your drawing with your pencil.

Finally, look at your picture and determine which lines are (maybe partially) hidden behind the object and which ones are visible. Erase those parts of the hidden lines that should be invisible. Redraw the visible lines with your black pen. If you want, you can keep the hidden lines, either drawn with the pencil, or as dashed lines with your black pen.

If you still have enough energy, you can also draw a perfectly three-dimensional picture of your original object. For this simply repeat the above procedure without your transformations (that is, calculate  $PD$ , find the 2D coordinates again by dividing the first two entries by the last, then draw a new picture). This picture can help the viewer to guess which transformations you used.

## 5 A Remark On Using Computers

Since we are dealing with applied mathematics here, of course *the use of computers is highly encouraged and recommended!!!* If you don't feel like you have enough computer skills, you should still try to do the project, just with fewer vertices to reduce your workload a bit.

You can use your favorite programming language—anything is fine. Some languages provide tools and commands for matrix and vector computations

in the standard package. Examples are MatLab, Mathematica, or (less prominent) statistical languages like R or SPLUS. If your favorite programming language does not come with any comfortable commands for matrix and vector computations, it should still be easy to program them by yourself as a subroutine—that is a matter of less than five lines.

If you want to avoid that, you can download the language R for free. Go to [www.r-project.org](http://www.r-project.org). Click on “Download-CRAN”, then scroll down on the right side of the screen and click on any link belonging to a site in the United States that works properly. A box will appear saying “Precompiled Binary Distributions”. In this box click on the link with the name of your operating system, then click on “base”. Finally, click on “rw1081.exe” to download the file. It is about 20 MB in size, so you better find a computer at NYU with a fast connection and a CDR drive to download it. Install it by double-clicking the downloaded file on your computer and choosing all the standard presets in the installation procedure.

To use R, start it by double-clicking the R icon on your desktop, and you are ready to type commands into the window that appears. You can assign a value to a scalar variable by typing `x=5` and define a vector with `x = c(1,2,3)`, for example. If you want to print the content of the variable `x` on the screen, just edit `x` and press enter. An  $m \times n$  matrix is initialized by `M = matrix(0,m,n)`. Then you can enter the entries of  $M$  for example by `M[,6]=c(1,2,3)`. This will fill in the vector  $(1, 2, 3)$  into the sixth column of  $M$  (make sure that the dimensions are right). Finally, you can multiply two matrices or a matrix with a vector with the commands `A1 %*% A2` and `M %*% x`, respectively. Don’t make the mistake to write `M*x`—that does something different. When you close the program, it will ask you if it should save a workspace image. If you say yes, then the next time you launch the program you can start right where you stopped, and you don’t have to type in your matrices again.

Be careful that R expects angles in radiant, not in degrees. So in order to compute  $\sin(x)$  when  $x$  is given in degrees, you have to write `sin(x*pi/180)`.

Also, there is a neat way to perform the very last computation step in which you divide the first two lines of your  $3 \times n$  matrix by the last line.

Simply write

```
L = M%*%D
N = L[1:2,]
N[1,] = N[1,]/L[3,]
N[2,] = N[2,]/L[3,]
```

The first line defines  $L$  to be your matrix of homogeneous coordinates,

the second line reads out the first two lines of  $L$  and stores them in a  $2 \times n$  matrix  $N$ , the third and fourth lines divide the first and second row of  $N$  respectively by the third row of  $L$ .

When you do this trick, please check this division operation by hand on your first two or three columns to make sure you did everything right!

## 6 Matrix Templates

The following is a collection of templates for some of the matrix templates that you can use for your transformations:

$$\begin{aligned}
 \text{Shift}_{(x,y,z)} &= \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}, & \text{Turn}_{x,\alpha} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \\
 \text{Turn}_{y,\alpha} &= \begin{pmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, & \text{Turn}_{z,\alpha} &= \begin{pmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \\
 \text{Str}_{x\text{-axis},k} &= \begin{pmatrix} k & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, & \text{Str}_{(x,y)\text{-pl},k} &= \begin{pmatrix} k & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \\
 \text{Str}_{\text{all dir.},k} &= \begin{pmatrix} k & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, & \text{Shear}_{x,z,k} &= \begin{pmatrix} 1 & 0 & k & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \\
 \text{Mirr}_{x\text{-ax.}} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, & \text{Mirr}_{(x,y)\text{-pl.}} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \\
 \text{Mirr}_{\text{orig.}} &= \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, & P_d &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{1}{d} & 1 \end{pmatrix}
 \end{aligned}$$

The matrices have the following functions:

$\text{Shift}_{(x,y,z)}$  shifts a given vector by  $(x, y, z)$ .

$\text{Turn}_{x,\alpha}$  turns around the  $x$ -axis by an angle  $\alpha$  ( $c = \cos \alpha$ ,  $s = \sin \alpha$ ).

The direction of the turn can be determined as follows: Make a fist with your right hand and point your thumb up. Turn your hand such that your thumb points in the direction of the axis you are turning around. Then your other four fingers will show you the direction in which you turn. (This is a standard convention!)

$\text{Str}_{x\text{-axis},k}$  stretches by a factor of  $k$  parallel to the  $x$ -axis.

$\text{Str}_{(x,y)\text{-pl.},k}$  stretches by a factor of  $k$  both in  $x$ - and  $y$ -direction.

$\text{Str}_{\text{all dir.},k}$  stretches by a factor of  $k$  in all directions.

$\text{Shear}_{x,z,k}$  shears the  $z$ -component in the direction of the  $x$ -axis with shear parameter  $k$ .

$\text{Mirr}_{x\text{-ax.}}$ ,  $\text{Mirr}_{(x,y)\text{-pl.}}$  and  $\text{Mirr}_{\text{orig.}}$  mirror with respect to the  $x$ -axis, the  $(x, y)$ -plane and the origin.

$P_d$  is the projection matrix, as explained in the text.

Similar matrices can be easily constructed by changing the position(s) of the factor(s)  $k$  or of the minus signs.