The secret math behind modern computer graphics

by Magnus Ranlöf and Johan Winell may 1998

Preface

Computer graphic may seem to become more and more complicated, but the fact is, it is controlled by rather simple geometry and linear algebra and has always been. The major reason why computer graphics improve all the time is because of the higher performances of computers. By reading the following you will hopefully increase your understanding of computer graphics and the secrets behind it.

This article will take you on a journey from the simplest concepts of 2D-graphics, through 3D-graphic representation, to the dealing with shading and perspective problems. No pre-knowledge in computer graphics is needed, but some knowledge in fundamental mathematics and geometry may be useful.

Contents:

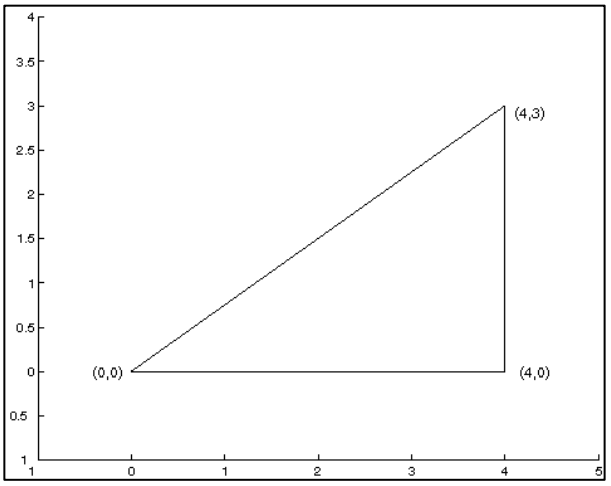
2D - graphics

2D – object representation

2D – objects are represented by an amount of points in a two dimensional co-ordinate system. The points in an object are declared in a matrix. For 2D – objects the matrix has the 2 x N dimension.

$$\underbrace{\begin{bmatrix} 0 & 4 & 4 & 0 \\ 0 & 3 & 0 & 0 \end{bmatrix}}_{N=4}$$

If lines are drawn between the points a figure will appear.



2D - transformation

The following transformation can be done to a 2D – object.

- Scaling, to make the object smaller or bigger.
- Translation, to move the object in the co-ordinate system.
- Rotation, to rotate the object in the co-ordinate system.

These transformations are applied to the object by the use of matrix operations. The object matrix is multiplied or added to a transformation matrix.

Scaling

To obtain a scaling of the object you must multiply " **A** " to a scaling matrix, " **S** ". The matrix obtained after the multiplication is a scaled object, " **A'** ".

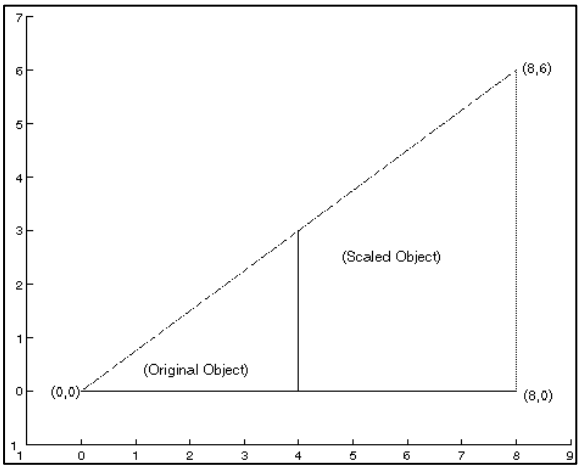
$$\mathbf{S} \cdot \mathbf{A} = \mathbf{A'}$$

$$\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \end{bmatrix} = \begin{bmatrix} A'_{11} & A'_{12} & A'_{13} & A'_{14} \\ A'_{21} & A'_{22} & A'_{23} & A'_{24} \end{bmatrix}$$

The value 2 for S_x and S_y will double the size of the object. The value 1 wouldn't change the object at all. With the scaling matrix, one can easily scale just the X – axis, or the Y-axis independent of each other.

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 4 & 4 & 0 \\ 0 & 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 8 & 8 & 0 \\ 0 & 6 & 0 & 0 \end{bmatrix}$$

If the "**A'**"-object were to be plotted it would be twice the size of the "**A**" – object.



Translation

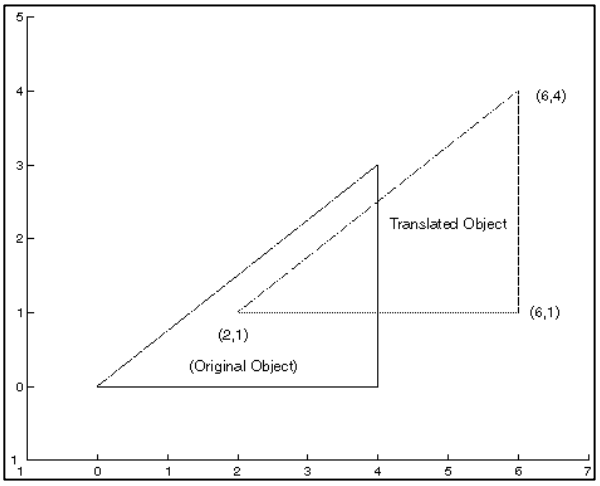
Translation of an object is obtained by adding a translation matrix, " **T** ", to the original matrix, "**A**". The resulting matrix will be translated object called " **A'** ".

$$\mathbf{T} + \mathbf{A} = \mathbf{A'}$$

$$\begin{bmatrix} T_{x1} & T_{x2} & T_{x3} & T_{x4} \\ T_{y1} & T_{y2} & T_{y3} & T_{y4} \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \end{bmatrix} = \begin{bmatrix} A'_{11} & A'_{12} & A'_{13} & A'_{14} \\ A'_{21} & A'_{22} & A'_{23} & A'_{24} \end{bmatrix}$$

The following " T "-matrix has the values "T_x" = 2 and " T_y "=1. The result will be the " A " – matrix object moved 2 units in the X – direction, and 1 unit in the Y – direction.

$$\begin{bmatrix} 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 4 & 4 & 0 \\ 0 & 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 6 & 6 & 2 \\ 1 & 4 & 1 & 1 \end{bmatrix}$$



Rotation

To rotate a 2D-object you must multiply a rotation matrix, " **R** ", to your original matrix, " **A** ". The rotation matrix is defined as follows:

$$R = \begin{bmatrix} \cos q & -\sin q \\ \sin q & \cos q \end{bmatrix}$$

θ is the rotation angle, defined counter clockwise. To rotate the object "A", perform the following operation.

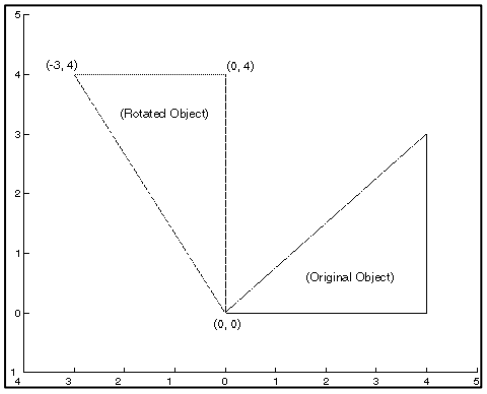
$$\mathbf{R} \cdot \mathbf{A} = \mathbf{A'}$$

$$\begin{bmatrix} \cos q & -\sin q \\ \sin q & \cos q \end{bmatrix} \cdot \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \end{bmatrix} = \begin{bmatrix} A'_{11} & A'_{12} & A'_{13} & A'_{14} \\ A'_{21} & A'_{22} & A'_{23} & A'_{24} \end{bmatrix}$$

If the rotation angle is 90 degrees, the result will be as follows:

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 4 & 4 & 0 \\ 0 & 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -3 & 0 & 0 \\ 0 & 4 & 4 & 0 \end{bmatrix}$$

Note that the object is rotated around the origin (0,0).



Homogenous co-ordinates

To be able to do these transformations in a sequence only involving multiplication, "*homogenous co-ordinates*" is the easiest way to accomplish it. In the two-dimensional case, an additional row filled with a "1" is introduced to the object description. To simplify it let us consider a single point. (See example below)

$$\begin{bmatrix}1\\2\end{bmatrix}\Rightarrow\begin{bmatrix}1\\2\\1\end{bmatrix}$$

It is now possible to perform translation with a matrix multiplication, it earlier involved addition to each point. To be able to perform the multiplication the transformation matrix has to be modified. (See example below)

$$T=\begin{bmatrix}T_x\\T_y\end{bmatrix}\Rightarrow\begin{bmatrix}1&0&T_x\\0&1&T_y\\0&0&1\end{bmatrix}$$

A translation can now be done like this:

$$\begin{bmatrix}1&0&T_x\\0&1&T_y\\0&0&1\end{bmatrix}\cdot\begin{bmatrix}X\\Y\\1\end{bmatrix}=\begin{bmatrix}X+T_x\\Y+T_y\\1\end{bmatrix}$$

A rotation for example will look like this:

$$\begin{bmatrix}\cos q&-\sin q&0\\\sin q&\cos q&0\\0&0&1\end{bmatrix}\cdot\begin{bmatrix}X\\Y\\1\end{bmatrix}=\begin{bmatrix}X\cdot\cos q-Y\cdot\sin q\\X\cdot\sin q+Y\cdot\cos q\\1\end{bmatrix}$$

With the use of homogenous co-ordinates, it is now possible to perform composite transformations by using only one expression. (E g) a translation followed by a rotation:

$$\begin{bmatrix}1&0&T_x\\0&1&T_y\\0&0&1\end{bmatrix}\cdot\begin{bmatrix}\cos q&-\sin q&0\\\sin q&\cos q&0\\0&0&1\end{bmatrix}=$$

$$\begin{bmatrix}\cos q&-\sin q&T_x\\\sin q&\cos q&T_y\\0&0&1\end{bmatrix}$$

The expression is executed backwards, e g the rotation matrix is performed first and the translation matrix is performed after.

To rotate an object around an arbitrary point , first translate the point to the centre of rotation (the origin), rotate the object and translate back the point. The composite expression, C, is defined as

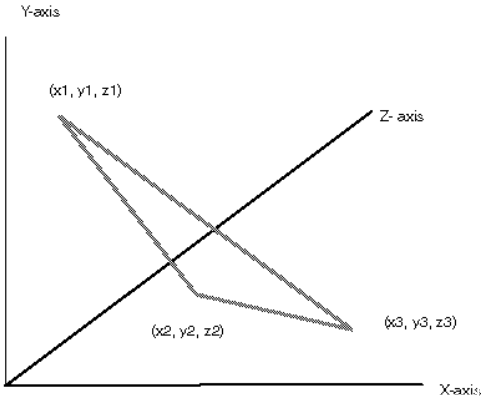
$C = T_{back} \times R \times T$ **,**where T is the translation, R the rotation and T_{back} the translation back.

3D-graphics

3D-object representation

All objects including 2D- and 3D- objects can be described by many polygon surfaces. The triangle is a very useful shape by which boxes and spheres easily can be described. (Imagine a mirror ball, which contains many thousands of quadratic mirrors.)

When dealing with 3D-objects a third co-ordinate, Z , must be considered. The 3D-co-ordinate system will look as follows:



When describing a triangle area in a 3D-object, three points must be specified, each with three co-ordinates, an X-, a Y- and a Z-co-ordinate.

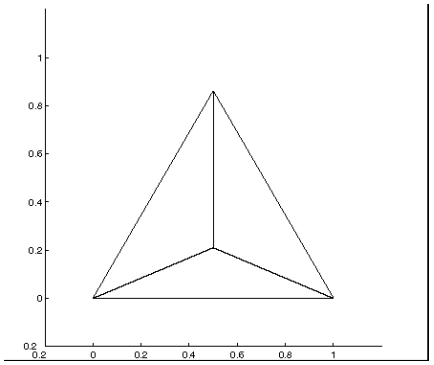
The object description for one triangle can be described like this using homogenous co-ordinates. Each of them is described with three corners (*vertexes*).

$(1,4,7)(2,5,8)(3,6,9)$ or as a matrix : $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 1 & 1 & 1 \end{bmatrix}$

A pyramid can be described by four triangles. Since the triangles in a pyramid have connecting edges, some of the corners will be the same. Therefore it is efficient to put all the corners in a single matrix:

$$\begin{bmatrix} 1 & 2 & 3 & & 2 & 2 & 3 \\ 4 & 5 & 6 & & 3 & 4 & 6 \\ 7 & 8 & 9 & . \; . & 2 & 3 & 4 \\ 1 & 1 & 1 & & 1 & 1 & 1 \end{bmatrix}$$

Specifying three specific corners of the amount of corners can now make the definition of a triangle. To draw a *wireframe* of an object, its different triangles must be drawn. To make this it is just to specify the three corners of the triangle, and draw lines between them.



Transformation of a 3D-object

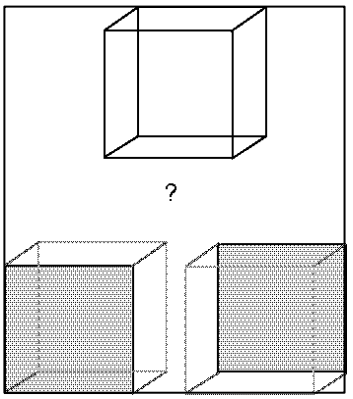
Transformations of 3D-objects are very similar to transformations of 2D-objects, the only thing that distinguishes is that a third dimension is added. The following matrixes describes a, A) scaling, B) translation, and C) rotation.

$$A)\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} B)\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} C)\begin{bmatrix} \cos q & \sin q & 0 & 0 \\ -\sin q & \cos q & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^1$$

1) The matrix represents a Z-axis rotation. Note that the Z-co-ordinates are left unchanged.

Depth problems

When looking at a wireframe it is impossible to decide which side is at the front and which is at the back.



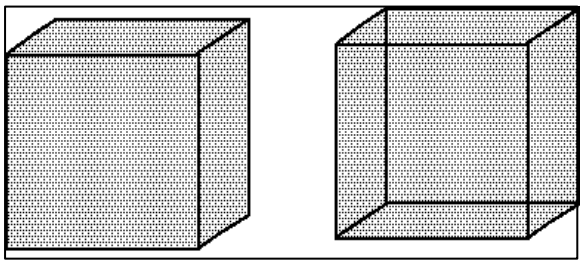
One way to create an impression of depth in the wireframe, without the use of perspective, is using a method called *depth cueing*. The lines in the front are drawn with higher intensity than the lines in the back. The use of this method makes the rear lines look shaded, compared to the front lines.

When drawing solid objects a similar problem occurs: Some surfaces of the object are not to be shown to the viewer, because of their placement in the back of the object. To present the object correctly you must either draw the surfaces in a certain order, or make sure that the surfaces in the rear never are drawn. The first option can be very difficult to handle, because the drawing order is described in the object representation.

Back face removal

The easy way to get around this problem is to set a “visibility flag” for each surface. The surfaces in the front are given a flag “show” and the surfaces in the back are given a “do not show” – flag. When the solid is to be drawn, only the surfaces with the “show – flag” are drawn. The actual result is that surfaces, which represent the rear, are not shown for the viewer.

The flags are set by calculating the scalar product between each triangles normal vector and a vector which represents the direction of the viewer. If the result is negative the normal points away from the viewer, the “do not show” – flag is set, and the surface is not shown.



Which side is the front?

Lightning and shading

To create a realistic picture, it is important to use shades, and other lightning effects in a proper way. First, set the co-ordinates of the light source, and construct a vector between the light source and the mid point of the object. Then calculate the normal of the surface. The scalar product between the “light source vector” and the normal of the illuminated surface is the light intensity in that particular surface. “1” represents the highest intensity (The normal and the “light source vector” has the same direction. “0” represents the lowest intensity (The normal has the opposite direction). The intensity value can be higher than one and less than zero. Values exceeding one, is set to one. A negative value is taken care of by the “back face removal” procedure.

Perspective

Depth in a picture has been discussed before, but then we were dealing with *depth cueing*. The other way to create a sense of depth is by the use of perspective. When drawing a perspective object, the object is scaled depending on its distance to the viewer.

One way to implement the scaling is to use the last co-ordinate in an object, described in homogenous co-ordinates, and let that co-ordinate represent the scaling. All other co-ordinates are divided by the scaling value.

Authors words:

This wasn’t so hard after all, was it? If you had any problems with the linear algebra, “Krypa – gå”, by the famous Swedish mathematician Peter Hackman, will give you all the knowledge you need.

Hopefully this brief text increased your appetite for more knowledge in computer graphics. We strongly recommend you to read another 650 pages about it in *“Computer graphics – second edition”* by Donald Hearn and M.Pauline Baker. It is a little bit tougher to read, but it will widen your knowledge in the topic. ð