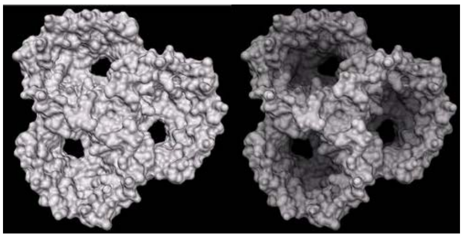
3D-Viewing

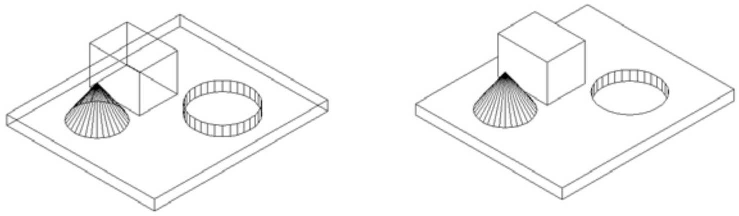
Concepts of 3D-Viewing

For representing 3D-objects on a 2D-screen in a nice or recognizable way, many techniques are combined. First, the projection has to be defined, which will be described in the next paragraph. After the projection has been set, any of the following qualities can be generated:

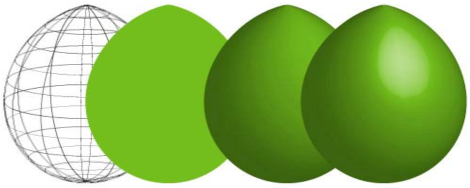
Wire-Frame-Representation = Only the edges of polygons are drawn. Hidden edges may or may not be drawn.

Depth-Cueing = Edges or parts which are nearer to the viewer are displayed with more intensity (brighter, broadened, more saturated), edges which are farther away from the viewer are displayed with less intensity (darker, thinner, grayed out).



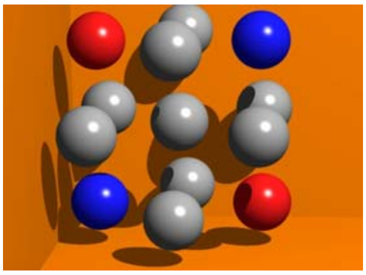


Correct Visibility = Surface-elements (edges, polygons), which are occluded by other surface-elements, are not drawn so that only visible areas are shown.



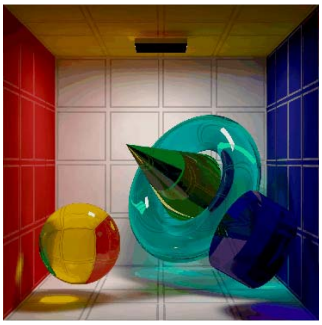
Shading = Depending on the angle of view or the angle of incident light, surfaces are colored brighter or darker.

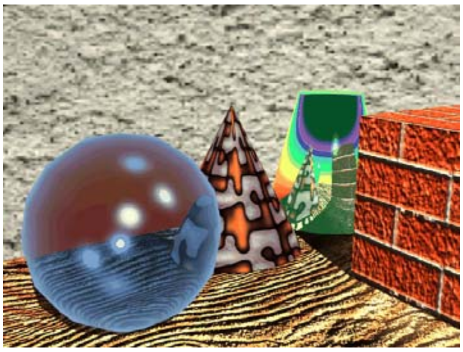
Illumination Models = Physical simulation of lighting conditions and propagation and their influence on the appearance of surfaces.



Shadows = Areas which have no line of sight to the light-source are displayed darker.

Reflections, Transparency = Reflecting objects show mirror-images, and through transparent objects the background can be seen.





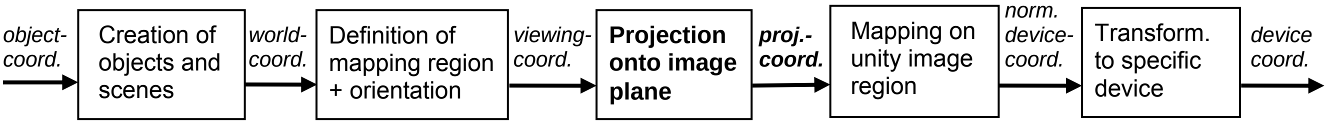
Textures = Patterns or samples are „painted“ on surfaces to give the objects a more complex look (looks much more realistic then).

Surface Details = Small geometric structures on surfaces (like orange peel, bark, cobblestones, tire profiles) are simulated using tricks.

Stereo Images = A separate image is created and presented (with various techniques) for each eye to generate a 3D-impression.

3D-Viewing-Pipeline

The viewing-pipeline in 3 dimensions is almost the same as the 2D-viewing-pipeline. Only after the definition of the viewing direction and orientation (i.e., of the camera) an additional projection step is done, which is the reduction of 3D-data onto a projection plane:

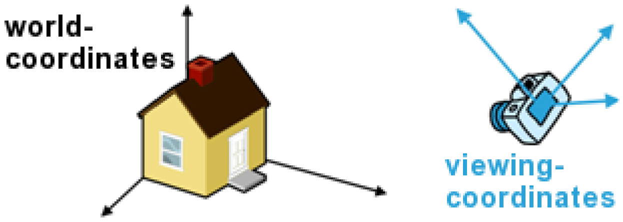

object-coord.
Creation of objects and scenes
world-coord.
Definition of mapping region + orientation
viewing-coord.
Projection onto image plane
proj.-coord.
Mapping on unity image region
norm. device-coord.
Transform. to specific device
device coord.

This projection step can be arbitrarily complex, depending on which 3D-viewing concepts should be used.

Viewing-Coordinates

Similar to photography there are certain degrees of freedom when specifying the camera:

1. Camera position in space
2. Viewing direction from this position
3. Orientation of the camera (view-up vector)
4. Size of the display window (corresponds to the focal length of a photo-camera)



With these parameters the *camera-coordinate system* is defined (viewing coordinates). Usually the xy-plane of this viewing-coordinate system is orthogonal to the main viewing direction and the viewing direction is *in the direction of the negative z-axis*.

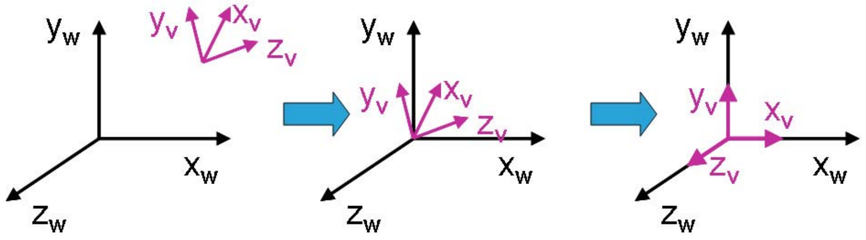
Based on the camera position the usual way to define the viewing-coordinate system is:

1. Choose a camera position (also called eye-point, or view-point).
2. Choose a viewing direction = Choose the z^- direction of the viewing-coordinates.
3. Choose a direction „upwards". From this, the x-axis and y-axis can be calculated: the image-plane is orthogonal to the viewing direction. The parallel projection of the „view-up vector" onto this image plane defines the y-axis of the viewing coordinates.
4. Calculate the x-axis as vector-product of the z- and y-axis.
5. The distance of the image-plane from the eye-point defines the viewing angle, which is the size of the scene to be displayed.

In animations the camera-definition is often automatically calculated according to certain conditions, e.g. when the camera moves around an object or in flight-simulations, such that desired effects can be achieved in an uncomplicated way.

To convert world-coordinates to viewing-coordinates a series of simple transformations is needed: mainly a translation of the coordinate origins onto each other and afterwards 3 rotations, such that the coordinate-axes also coincide (two rotations for the first axis, one for the second axis, and the third axis is already correct then). Of course, all these transformations can be merged by multiplication into one matrix, which looks about like this:

$$M_{WC,VC} = R_z \cdot R_y \cdot R_x \cdot T$$



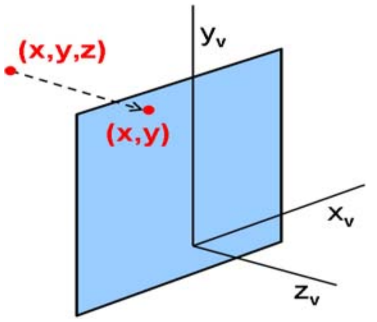
Projections

In geometry several types of projections are known. However, in computer graphics mainly only the parallel and the perspective projection are of interest.

Parallel Projection

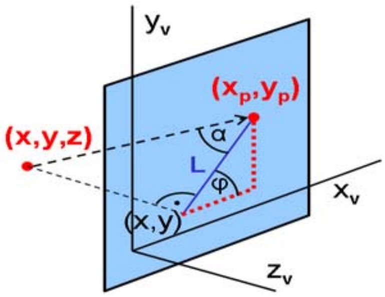
A parallel projection can either be done orthogonally to an image-plane, or obliquely (e.g. when casting shadows).

The orthogonal parallel projection is very simple. Assuming that the direction of projection is the z-axis, simply omit the z-value of a point, i.e. set its z-value to zero. The corresponding matrix is therefore also very simple, because the z-coordinate is just replaced by zero.



$$\mathbf{M}_{\text{parallel}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

An oblique parallel projection, which is defined by two angles α and φ, can be done as follows: for any point the distance L is the distance between two points, the one orthogonally projected onto the z-plane, which results in (x, y, (0)), and the result of the oblique projection, which results in (x_p, y_p, (0)). This distance L is then one cathetus of the right-angled triangle with vertices (x, y, z), (x, y, 0) and (x_p, y_p, 0), and at the same time L is the hypotenuse of the also right-angled triangle in the image-plane with the cathetus (x_p–x) and the cathetus (y_p–y). From this L = z/tan α, and x_p = x + L·cos φ and y_p = y + L·sin φ. If L is substituted into the other two equations this produces the given matrix.

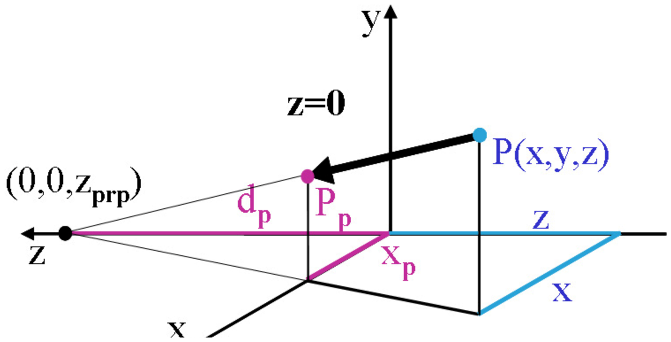


$$\mathbf{M}_{\text{parallel}} = \begin{bmatrix} 1 & 0 & \dfrac{\cos\varphi}{\tan\alpha} & 0 \\ 0 & 1 & \dfrac{\sin\varphi}{\tan\alpha} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective Projection

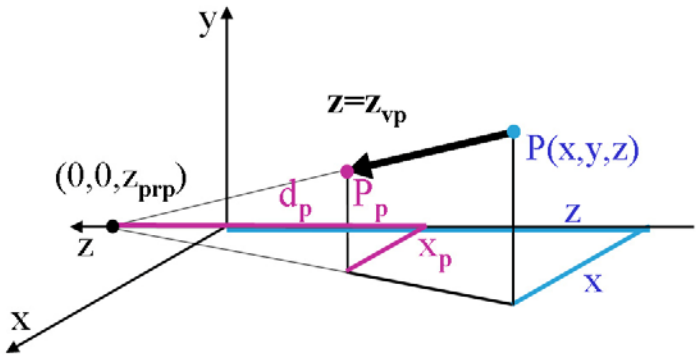
With a perspective projection some laws of affine transformations are not valid anymore (e.g. parallel lines may not be parallel anymore after the perspective projection has been applied). Therefore, since it’s not an affine transformation anymore, it can’t be described by a 3x3 matrix anymore. Luckily, again homogeneous coordinates can help in this case. However, this is the sole case where the homogeneous component h is not equal to 1 in the result. Therefore, in a subsequent step a division by this value is needed.

Let us first examine the case where the projected image lies on the plane z=0. The image of a point P(x, y, z) is the point, where the line through the point itself and the center of projection (0, 0, z_{prp}) cuts the plane z=0. When viewed from above, similar right-angled triangles can be found with the catheti d_p and x_p on one hand, and d_p–z (because z is on the negative side of the z-axis) and x on the other hand.



This yields $x_p : x = d_p : (d_p–z)$ or $x_p = ·d_p/(d_p–z)$
and in analogy $y_p : y = d_p : (d_p–z)$ or $y_p = y·d_p/(d_p–z)$
and of course $z_p = 0$.

If we project onto any other plane $z = z_{vp} \neq 0$ not much has to be changed: only d_p is replaced by the z-value of the center of projection z_{prp}:



$x_p : x = d_p : (z_{prp} - z)$ or $x_p = x \cdot d_p / (z_{prp} - z)$
$y_p : y = d_p : (z_{prp} - z)$ or $y_p = y \cdot d_p / (z_{prp} - z)$
$z_p = z_{vp}$.

This can be written as a homogeneous matrix also:

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -z_{vp}/d_p & z_{vp}(z_{prp}/d_p) \\ 0 & 0 & -1/d_p & z_{prp}/d_p \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

In general, the value of h of the resulting vector will not be equal to 1 (h is calculated as $(z_{prp} - z)/d_p$). This is the reason why the resulting vector has to be divided by h (in other words, it is scaled with 1/h) to get the correct resulting point:

$x_p = x_h/h$, $y_p = y_h/h$ and z_p results in z_{vp}.

By representing projections in matrix form, it is possible to formulate the whole transformation from model-coordinates to device-coordinates by multiplying the single matrices to one matrix. Only if a perspective projection is involved, it must not be forgotten to divide the result by the homogeneous component h.

Finally, it should be mentioned that the number of *principal vanishing points* is dependent on how the axes of the coordinate system are positioned against the image plane. If 2 coordinate axes are parallel to the image plane it is called *1-point perspective projection*, if only one is parallel to the image plane we call it *2-point perspective projection*, and if none of the three axes is parallel to the image plane it is called *3-point perspective projection* (because then there are 3 principal vanishing points).