

Introduction to 3D

To all intents and purposes, the world we live in is three dimensional. Therefore, if we want to construct a realistic computer model of it, the model should be three dimensional as well.

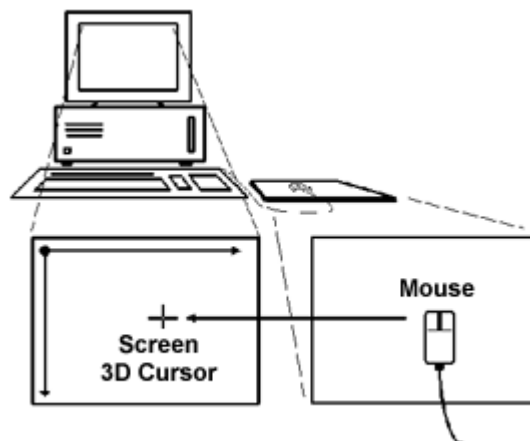
Unfortunately, the usual display device of a computer (the monitor) is 2 dimensional. However, it seems that 2 dimensions may be enough to convince the viewer that he or she is watching a realistic model of the real world. For example, the images seen on a television.

Nevertheless, we have been restricted to two-dimensional television and two-dimensional cinema for many years and seem very happy with it. This is because the images we get on the TV have many subliminal visual cues that tell us "This is 3 Dimensional". And the brain believes what the eyes tell it even though we know the screen is flat! Thus we can conclude that a 2-D screen can do a good job of presenting the 3D world, if we do it right!

We have a not insignificant hurdle in working in 3D on the computer. The screen is flat, it has width and height but we also need depth. Depth is the dimension into and out of the screen. All computer screens are made up from small rectangles of color, each rectangle is called a "pixel" (short for picture element), the more pixels on the screen the higher the resolution of the screen.

It is usual to specify the resolution of the computers display by quoting the number of pixels across the screen and the number of pixels down the screen rather than the very large number of pixels on the screen! For example a low resolution output for a display of 320 pixels across the screen by 200 pixels down the screen requires 64000 pixels in total.

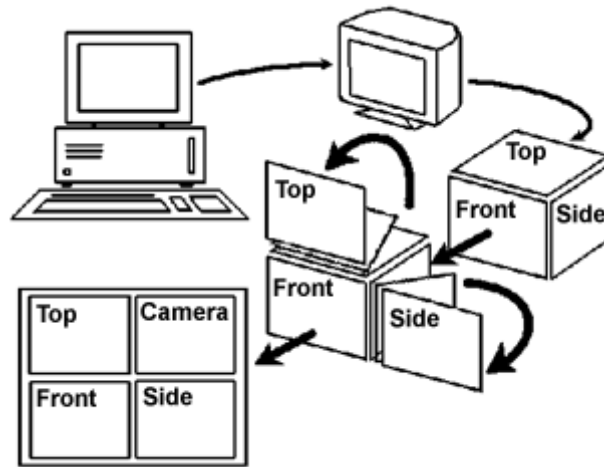
Two dimensional art, animation or drafting (CAD) packages usually use a graphics cursor (an arrow or some other icon) so that you can make a specific action at an appropriate place on the screen. They may also use a co-ordinate reference (rather like a map-reference) to specifically refer to a point on the screen. Like a map reference, you need two numbers to specify a point on the screen, an across number and a down number. Again like a map reference you need a fixed reference point. This is usually taken as the top left corner of the display and is referred to as (0,0). Moving across the screen points are referenced as (1,0) (2,0) etc. Moving down the screen points are referenced as (0,1), (0,2) etc. So any point can be defined with two numbers. For example the center of the screen might be (160,100).



In 3D we need to add a third number to the across and down coordinates (a depth coordinate "in/out") giving us (0,0,0), (0,0,1) etc. We have a problem in visualizing this third coordinate. Because the screen has no depth, we cannot see a move that goes in/out. Ideally we would like a second screen on the side of the monitor to see the depth move.

We can't expect computer manufactures to make monitors with screens on the side as well as the front, even if we (as users) were prepared to move round to the side to look at it!

However, we have designed our 3D software so that it folds this 'imaginary' screen on the side of the monitor round, until it lies in the same plane as the monitor's screen. It is also helpful to fold an imaginary screen from the top of monitor so that we have three views arranged on the screen and all visible at the same time.



Our next difficulty arises because the mouse only works when it is on a flat surface. You cannot pick it up and move it about on the side of a computer monitor! But wait, you don't pick the mouse up and move it about the screen for a 2-D drawing either. You just assume that its pointer is on the screen in front of you.

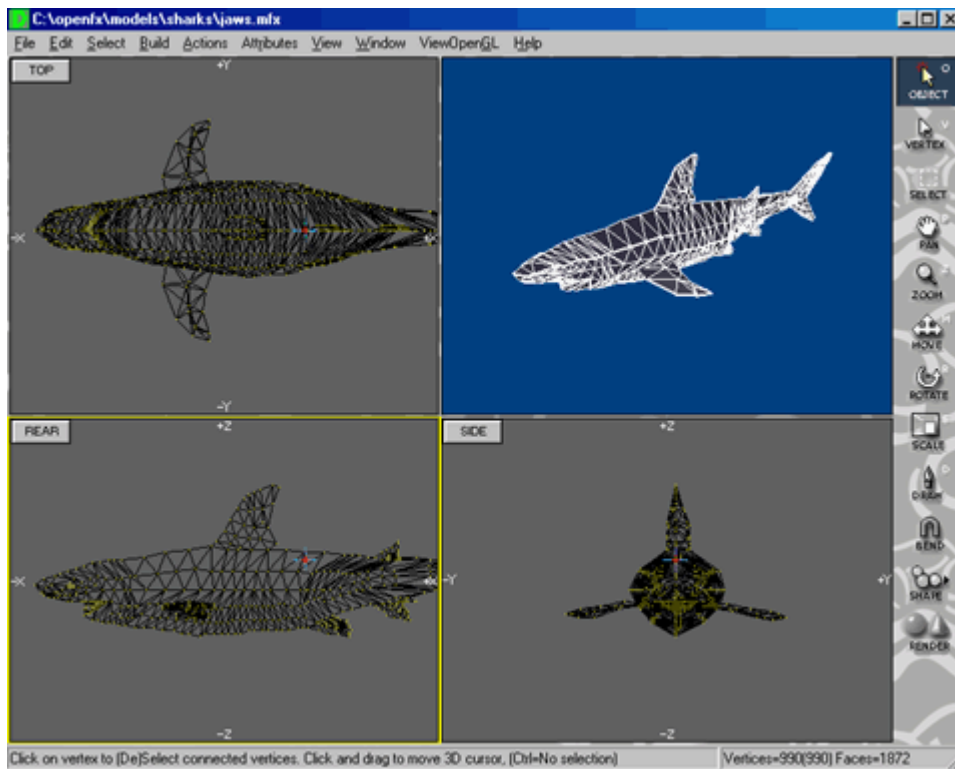
Therefore, let's assume that when the mouse pointer is in a part of the screen which we are using as one of the folded views. it moves as if it were on the side or top of the monitor.

There we are; that's got the 3D screen and mouse sorted out.

3D Views of Objects

Lets imagine we are looking through the front face of a cubic fish tank, with some goldfish swimming inside. The other sides, the top and the bottom of the tank are covered up. Our view into the tank shows the goldfish swimming up and down, turning, going behind a rock and moving behind and in front of one another.

As a fish turns we can see its left side and then its right side. If we are going to make an animation of the goldfish in the tank we need to know what the fish looks like from the left side, the right side and from any other point of view we care to choose.



If we had a 2D animation package we would need to be very good artists to make the necessary drawings as our fish turns, even if the computer helps by doing the 'in- between's'. A 3D package is very clever because it will draw the appropriate view of the goldfish, but you don't get something for nothing, you will still have to tell the computer what the goldfish looks like.

There is a very subtle difference between drawing a goldfish and defining what a goldfish will look like! You can think of the difference between an architects plan and an artists drawing of a house. This analogy with the plan of the house is quite a good one. The plan tells you everything about the house but it doesn't show you what it will look like. A plan enables the builders to put the house together, you can then walk round and through the house to see what it looks like from wherever you want. However every time an artist wants to draw the house from a different view point he or she will have to start from scratch. The same thing applies to goldfish or any other object. A plan contains much more information than a drawing but it is not what we like to look at, a plan doesn't sell many houses, an artists impression may do!

The essence of 3D animation is that it takes a plan for objects and draws them as an artist would. The plan that we use is not quite the same as an architect's plan and thus we prefer to call it a model. In the example of our goldfish, the model provides information on what a goldfish looks like and it is stored in the computers memory.

You are now probably asking yourselves; What do we mean by a model that we can store in the computers memory? It can't be anything physical such as clay, plastic or balsa wood!

It has got to be numbers. What numbers? How many? What do the numbers mean? How do we use them to get the computer to draw a lifelike goldfish?

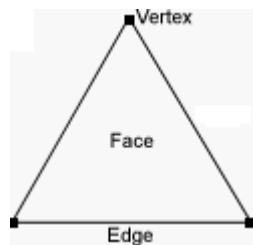
We will now attempt to answer these questions, but first, here is another question. What do you see when you look at a goldfish?

You see it's skin (it's surface). Thus the minimum that the model must do is to represent the surface of the fish. There is no point in modelling internal bits, they won't be visible. This surface is likely to be quite irregular (fins, eyes etc.). So how do we represent the surface in terms of computer data? Click on the link below to find out.

3D Models of Objects

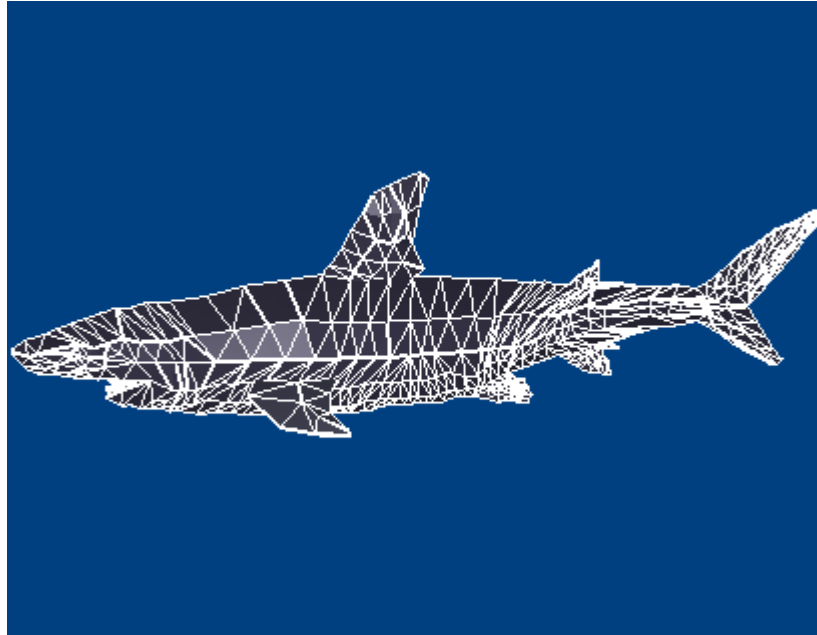
Ultimately we must know where every point on the surface is, what colour it is and, what it is made of and how much light is falling on it. We might think of recording a 3D co-ordinate for every point on the surface. But how many points are there on the surface of a goldfish, or put another way, what is the size of a point?

A point has no size, therefore we would need an infinite number of points to specify the surface of our goldfish (well one per molecule anyway). No computer can store an infinite number of points so we must give our point some size, some area. The simplest shape that has area is a triangle. A triangle has three edges, two of which meet at a point (a vertex).

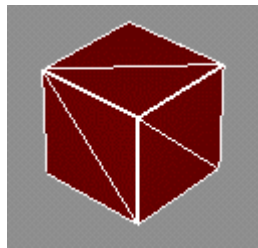


Therefore there are three vertices per triangle. If we specify a 3D co-ordinate for each vertex we have completely specified the position and orientation of a triangle. We can even find the co-ordinates of points within a triangle if we need to do so.

If we imagine a goldfish's surface covered with triangles and each triangle defined by the co-ordinates of its vertices we have a set of numbers that model the shape of a goldfish. The triangular pieces (which we will call faces) don't have to be the same size or shape (they must of course be triangular).



For example, suppose we want to model a cube, 12 triangles will give a perfect cube no matter what the size of the cube is. Because several of the vertices of the triangular faces occur at the same place we don't need to store 3 vertices per face, 8 vertices will do the job for the whole cube. In the same way, if two triangles have a common edge we need only store a single edge, that can be used for both triangles.



A cube of any size can be described exactly using 8 3D co-ordinates, one for each vertex. The 12 triangular faces are recorded by storing the identity of the vertex at each corner of the triangle, 36 numbers. A cube made up from triangular pieces has 18 edges and since each edge is joined to 2 vertices, two numbers are needed to specify an edge. Therefore 36 numbers are needed to define the edges in a cube made up from triangular faces. As mentioned earlier, 3 numbers are needed to specify a 3-D co-ordinate, thus the shape of a cube can be fully specified by $24+36+36=96$ numbers.

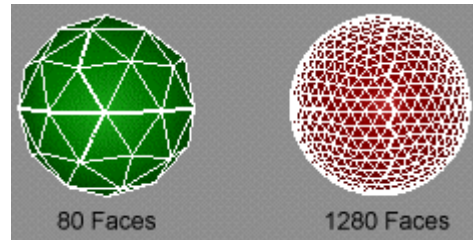
The computer loves manipulating numbers and from a numerical model it can reproduce a view of a cubic goldfish tank, goldfish or anything else. Building a numerical model for an object means that you won't have to do any additional modelling work to change you view point or the lighting conditions.

There is one drawback to using triangular faces to describe the surface of an object and that is that objects are rarely made up of triangular pieces!. The surface of a sphere is not made up of triangular pieces, it is perfectly curved. Some objects can be exactly modelled by triangular pieces, the pyramids at Gisa for example. Any object that has part of its surfaces curved cannot be represented exactly by triangular pieces, but don't despair, if we choose the right number of triangles and make then the right size and shape we will get surfaces that look as smooth and curved as the aerodynamic shape of a high performance sports car.

The pyramid example cited above would need 6 triangles to get the shape exactly right. If there had been any curved surfaces on the pyramid we would have modelled this by increasing the number of

triangles, and consequently decreasing the size of each triangle so that they cover the same surface area.

To imagine how increasing the number of triangles leads to better and better models for the real thing consider a Sphere. We can get pretty good looking spheres if we use 80 triangles covering the surface, and spheres that will be impossible to tell apart from the real thing if we use 1280 triangles to cover the surface.



One of the first skills that you will need to acquire is in deciding how many vertices and faces are needed to model an object. The more vertices and faces you use the better looking the model, but you will run out of memory on the computer and it will take longer and longer to draw (or render) the images if you just go on adding vertices, edges and faces. It is usually possible to reach a good compromise long before the machine lets you down.

It is good practice to build models, especially complex ones in pieces. Store each piece in a separate file and when all the components have been built they can be brought together to make a single model. (Much the same thing happens in any modern factory.) For example a push-bike can be made by building the frame, the front and rear wheels, the saddle, the brakes, etc. as separate bits.

Once we have build our model using the vertex, edge and face data structure its a very simple procedure to change how the model will look like, just by moving the vertices. You could for example turn our goldfish into a shark by making it larger and moving the vertices in the fins so that they look more like those of a shark.

If you keep the number of vertices the same but change their positions you will be able to "morph" the model so that it changes from a goldfish into a shark before you very eyes. You might have more difficulty in changing the goldfish into an aeroplane, but it could still be done.

Painting for Realism

You might think that with the specification of the triangular model for an objects surface we have achieved our goal. However, as we have defined it, this model only describes the geometry (the shape of the object) like any good model we need to paint the surface to make it look like the real thing.

It turns out that this is not too difficult. You just have to give a few extra numbers for each face to tell the computer the colour, or the reflectivity or whatever you think is important.