It can't be too hard .... the guide you've all been waiting for, written by a complete eejit, even the cat can understand it.

Eejit's Guide

simple

to Object Building for BVE TrainSim

written by graymac, with a bit of sense from guillyman

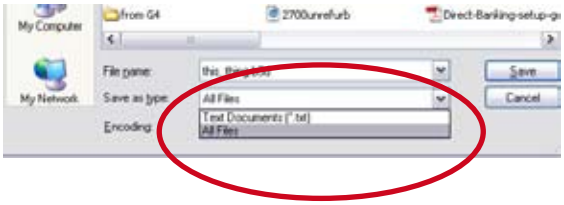
BVE new

open

Lesson 1 – Create a mesh for a single surface

Items needed to do this stuff

A basic text editor such as your windows notepad will do nicely. The object files are written in plain text but you must save them with a .b3d file extension, not as a .txt file, or they won't work.
A program to view the object with. Mackoys simple "Structure Viewer" is the best to start with and quicker to use for simple basics such as these. You can use openBVE object viewer if you prefer.
Download and install "Structure Viewer". To open the object file just drag its icon onto the open structure viewer window. You can use the mouse pointer to drag the object around in the window. PageUp and PageDown keys will change the size, and the up/down arrows will zoom in and out. To reload your object (after you alter and save it) key F5. To clear screen key Delete. Simple!



Be sure to save with a .b3d file extension, not as a .txt file, or it won't work

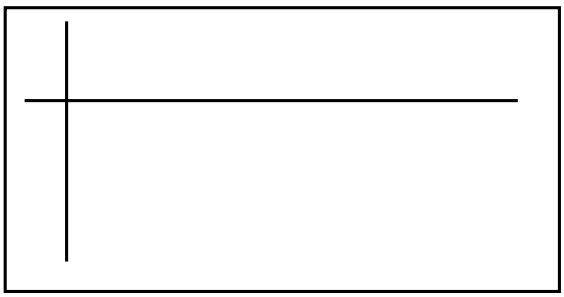
A bit of explanation

At first sight an object file makes as much sense as a frog on a bicycle. Look at any object file, these words are found in both .csv and .b3d files:

Meshbuilder
Vertex
Face
Texture
Coordinates
Color
What do they all mean?

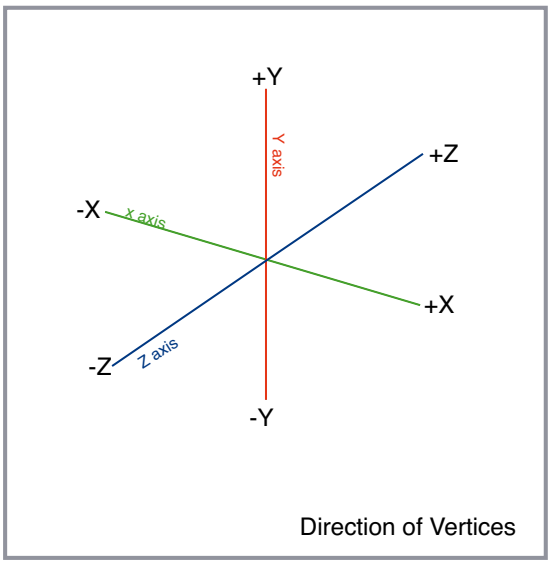
Objects are made of two parts. The invisible bit where the points in space are marked out, which is the "vertices", and the visible bit, known as the "face".
A flat square has two faces but you can only see one at a time.
A cube has twelve faces, six inside and six outside. A dice is a sort of cube where you can only see the outside faces, and a room is a sort of cube where you can only see the inside faces.
So everybody understands that bit quite easily.
The "face" may be a plain colour, or it can have a photographic image applied to it.
If there's a photographic image it is called a "texture".
Coordinates dictate how the picture (texture) behaves in relation to the vertices.

The two main kinds of object file are .csv format and .b3d. The wording and punctuation of them differ slightly. The order of words commas and spaces is called "Syntax" and early on you will get it wrong a lot and the wretched thing won't work, so you must look carefully at what you type in. And when you have been doing it for ages you will still get it wrong sometimes! So don't worry too much. The examples shown are for the .b3d file format. Later we will look at .csv files and you will see the difference and understand them quite easily.



A rectangle 5m x 3m will be coded first.

Our item always starts with the word "meshbuilder" in it.
Then we write in the location of the points in space, each point is called a "vertex".
The visible face is defined by stating the position of each vertex in terms of X,Y and Z. See the direction of vertices diagram.
Let's begin with a simple, rectangle, standing upright. It will be 3m high and 5m wide.
This is only a two-dimensional object so it has only width and height. Each corner of this square occupies a point in space dependent on its dimensions. The three dimensions in 3D are left and right; up and



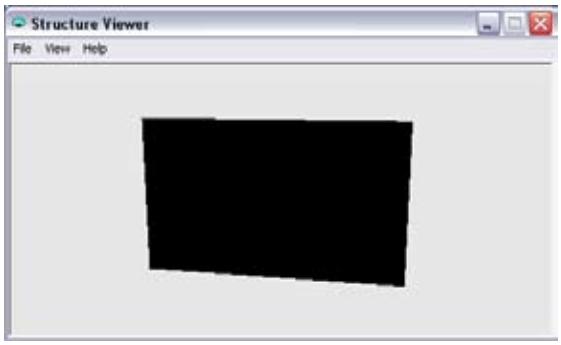
Direction of Vertices

down; and forward and backward.
We will call left and right X, up and down Y, and forward and backward Z. Vertex can be minus or plus value. If we take a single point and let it be zero (0) then from that point -X is left, +X is right.
Also -Y goes down and +Y goes up from the point. And -Z goes backwards and +Z goes forwards.
So a vertex is a definition of where a single point in space is. Vertices are defined in the order X;Y;Z.

Let's do this, start a document with indows Notepad.....
Write this code exactly as you see it here.

```
[meshbuilder]
vertex 0,3,0
vertex 5,3,0
vertex 5,0,0
vertex 0,0,0
face 0,1,2,3
color 0,0,0
```

save it as faceA.b3d.....
Open it with Structure Viewer and there is a vertical black surface.



The 5x3 metre rectangle, viewed with Structure Viewer.

Here's an explanation of the code:

[meshbuilder]	;starts the section for this structure ;the vertices are listed next, numbering from 0 at top downwards ***position:*** (see vertex direction diagram)	
vertex 0,3,0	; vertex 0, X=0, Y=3, Z=0	3m above zero
vertex 5,3,0	; vertex 1, X=5, Y=3, Z=0	3m above and 5m to right of zero
vertex 5,0,0	; vertex 2, X=5, Y=0, Z=0	5m to right of zero
vertex 0,0,0	; vertex 3, X=0, Y=0, Z=0	at zero
face 0,1,2,3	;the numbers of the vertices and the order in which they make up the face.	

This simple surface only has four vertices, The face order could also be 1,2,3,0 or 3,0,1,2, the only important thing is that they are ordered in a clockwise direction. Try substituting face 0,2,3. See how a triangle appears now, because the face is drawn between top left and both bottom vertices. Try face 1,2,3. Same shape, but flipped. So you have a mesh with four vertices, but the faces will appear dependent on which vertices you order in the face line.

color 0,0,0	;Specifies a single color applied to the face. Our example is 0,0,0 = black. The numbers are the color values for Red, Green and blue components of the color and range from 0 minimum to 255 maximum. 255,0,0 is pure red, 255,255,255 is pure white. You will be able to pick colors from any decent picture editing program and see the RGB values displayed. Try different values and see for yourself.

Turn the square around in your structure viewer. As you turn it so that you are looking at the back it seems to disappear.
This is because we specified the visible face in a clockwise direction

as face 0,1,2,3
When we look at the back the same vertices are now running anti-clockwise. Turn it round till the face disappears. Now edit the face to face 3,2,1,0 and save it. Go back to the structure viewer and press the F5 key to reload the object and hey presto! There it is.
Now, suppose we wanted to be able to see BOTH sides of the item as we go past it. That means we need to have both front and back faces.
Two possible ways to do that.
The first is to use “face2” instruction:

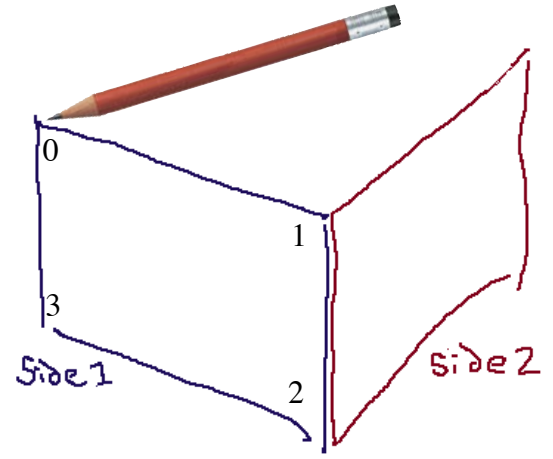
```
[meshbuilder]
vertex 0,3,0
vertex 5,3,0
vertex 5,0,0
vertex 0,0,0
face2 0,1,2,3
color 0,0,0          ;face2 will apply the color to both sides of the surface
```

Or we can code a second additional face:

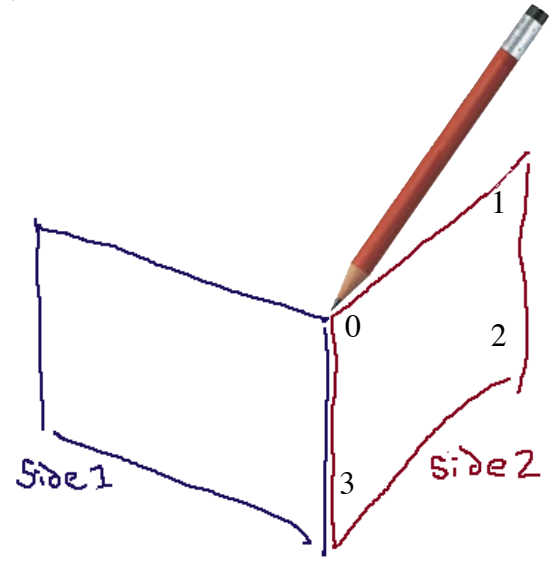
```
[meshbuilder]
vertex 0,3,0
vertex 5,3,0
vertex 5,0,0
vertex 0,0,0
face  0,1,2,3        ;the first face we defined
face  3,2,1,0        ;a second face is added
color 0,0,0
```

Try it and see the result in structure viewer.

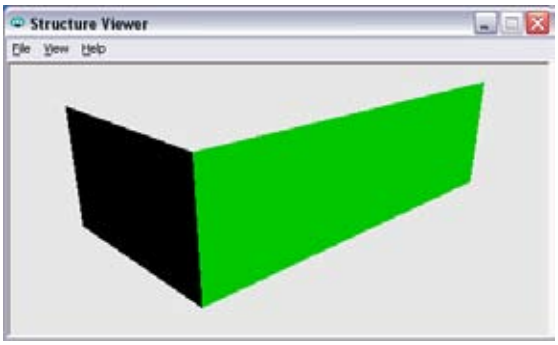
You should now understand how to construct a simple mesh and apply a face to either side or both sides.
When you are confident that you understand this part then move on to part two, where we will add another side at right angles to the one we just made.



Vertices in side 1



Vertices in side 2



Side two is done

Adding a bit more on

Let's add another side at right angles to the one we just made.
Tip: It is often useful to have a pencil and a scrap of paper handy, if you want to make a quick sketch of the shape you are aiming to make it helps visualise it and you can mark dimension and vertex information to help keep track.

Here's our first mesh:

```
[meshbuilder]
vertex 0,3,0
vertex 5,3,0
vertex 5,0,0
vertex 0,0,0
face 0,1,2,3
color 0,0,0
```

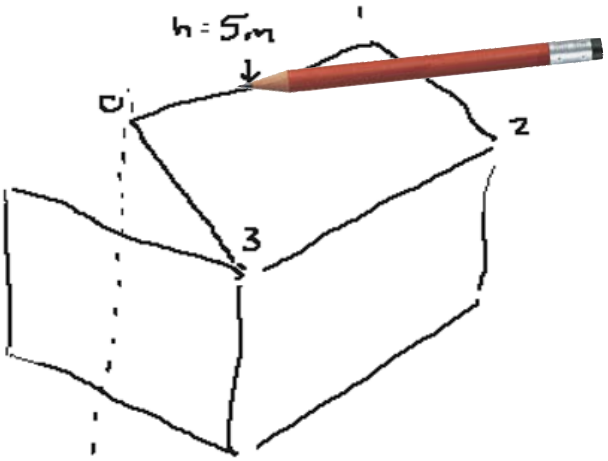
We will add a mesh to make a side to it of 10m length, same height, at the right and at 90 degrees.
Width: The X value of the right hand side is 5, so all X values are going to be 5
Height: The Y values are the same, between 0 and 3
Length: The Z values are now from 0 to 10
Write this code exactly as it appears here under the first mesh

```
[meshbuilder]
vertex 5,3,0
vertex 5,3,10
vertex 5,0,10
vertex 5,0,0
face 0,1,2,3
color 0,255,0
```

Note that the color has changed. It is now green.
Now save the file and view it, you should see it like the picture on the left.

(You will probably need to mouse drag the item in the Structure Viewer window to turn it round a bit to see the second side).

Lesson 2 – Adding another surface

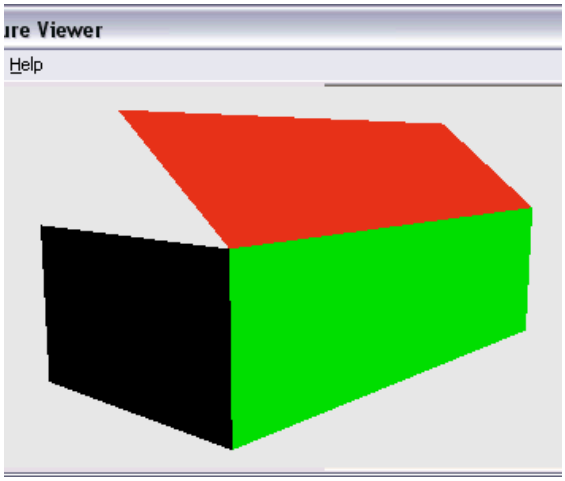


All it needs is a roof and it would be the right shape to look like a building.
A rough sketch, and let's make the tip of the roof 5m from the ground.
It runs the length of the item and the apex is over the centre.
As the item is 5m wide the centre will be half, thus 2.5m
(as you can see, the maths gets tricky!)

```
[meshbuilder]
vertex 2.5,5,0
vertex 2.5,5,10
vertex 5,3,10
vertex 5,3,0
face 0,1,2,3
color 185,40,20
```

Save and view .....
This time the color is a mix of R;G;B to make a rooftile sort of color

It will want a gable end added to look like a house. We can do this in two ways.
If we want, we make a triangle on top of the end wall. To do this, add these lines to the file:




```
[meshbuilder]
vertex 0,3,0
vertex 2.5,5,0
vertex 5,3,0
face 0,1,2
color 0,0,0
```

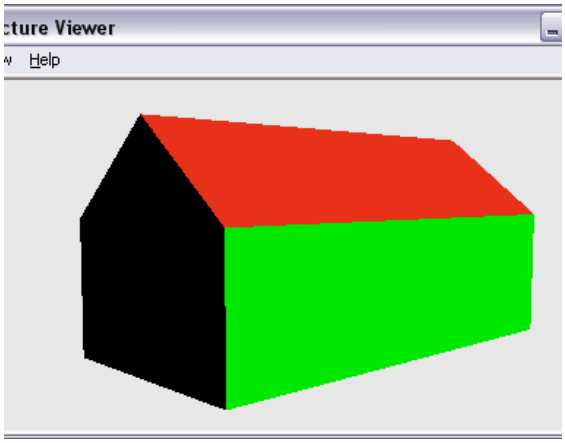
Save and view .....
And the hole is filled in.

The other way is to make a single mesh with five vertices, thus:
(take out the triangle you just made first, or start a new file)

Back to our first mesh:

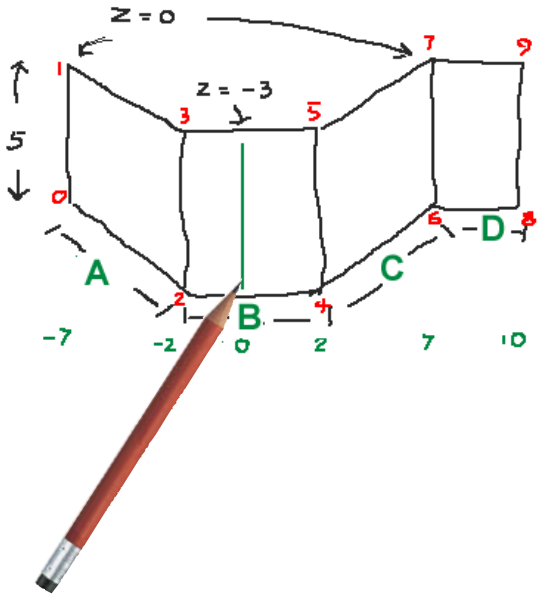
```
[meshbuilder]
vertex 0,3,0
vertex 2.5,5,0  ;Add this extra vertex to the file
vertex 5,3,0
vertex 5,0,0
vertex 0,0,0
face 0,1,2,3,4 ;There are now five points to the face
color 0,0,0
```

Add the code in red,
Save and view .....
And there you have it.



If you have followed this carefully so far and understood it properly you should now be able to make simple 3D object shapes to any dimensions you care to choose.
You have made progress!
Next we will do some meshes with more than four vertices and one face. Once we understand the meshbuilding we will be ready to see how we apply the textures (pictures) to the mesh.

One mesh, four faces



This time we shall order the vertices for the shape in the diagram, which is for a wall. Although there are four faces involved there is only one meshbuilder command needed to make this.

See the dimensions in the sketch of the wall. 0x is in the centre of the 4m wide second face (B) from left, therefore the left side of that face is x= -2 and the right side is x= 2. Likewise the rearmost part of the object is on the axis z= 0, so the front isa t z= -3. Minus 7 and 7 are the X offsets for faces A and C,not the actual lengths which are, of course a little longer.
The vertices are listed in the order numbered in red.

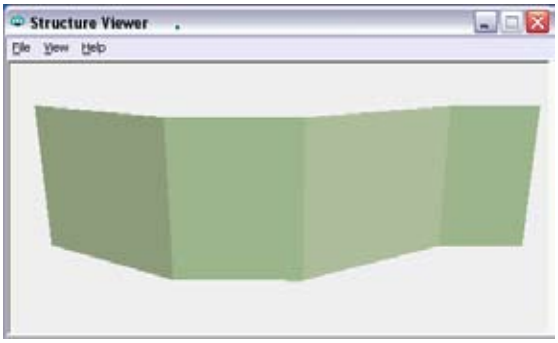
```
[meshbuilder]
vertex -7,0,0
vertex -7,5,0
vertex -2,0,-3
vertex -2,5,-3
vertex 2,0,-3
vertex 2,5,-3
vertex 7,0,0
vertex 7,5,0
vertex 10,0,0
vertex 10,5,0
```

With reference to the numbered vertices we should see that the far left face is as follows:

```
face 0,1,3,2          ;A
```

And the remaining three faces go like this:

```
face 2,3,5,4          ;B
face 4,5,7,6          ;C
face 6,7,9,8          ;D
```

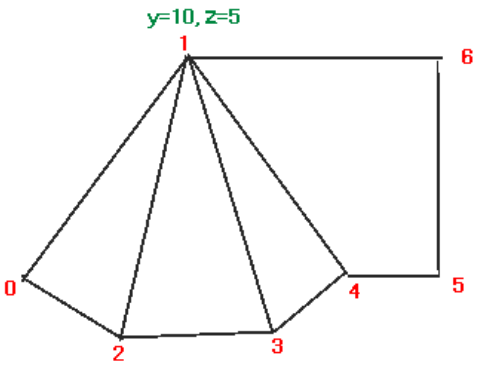


Give it a colour,

```
color 220,240,200
```

Save and view .....
And the wall should look like this.....
If it does, you're doing just fine.

For a bit more practice let's stick a roof on (I live in a wet country, we LIKE roofs!)
Again, a quick sketch.....

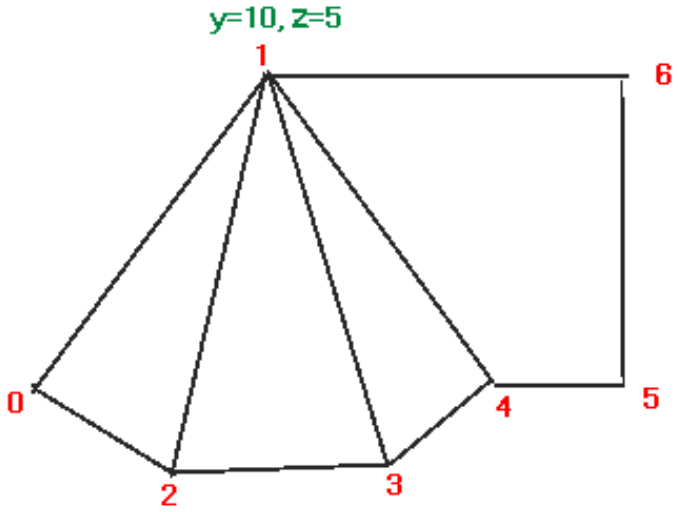


Vertices 0,2,3,4 and 5 are the same value as vertices 1,3,5,7 and 9 of the wall.
The apex of the roof is x= 0, y= 10 and z= 5

List the vertices and their xyz positions according to the information given, then describe the faces.
With the triangular faces in the roof there will, of course, only be three points in those faces.
You should arrive at this result with your coding:

Lesson 3 – Multiple faces in a mesh

(a grey is used for a slate colour, 120,120,120)

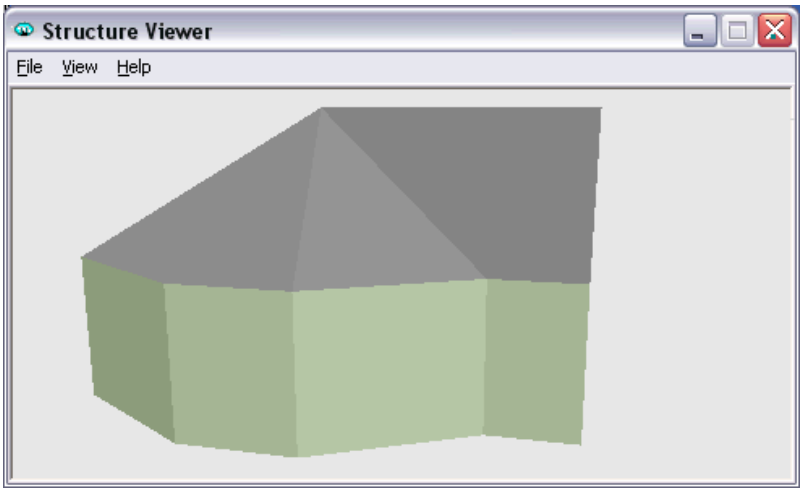



```
[meshbuilder]
vertex -7,5,0
vertex 0,10,5
vertex -2,5,-3
vertex 2,5,-3
vertex 7,5,0
vertex 10,5,0
vertex 10,10,5
face 0,1,2
face 2,1,3
face 3,1,4
face 1,6,5,4
color 120,120,120
```

Add to the wall we made earlier.

```
[meshbuilder]  ;wall code
vertex -7,0,0
vertex -7,5,0
vertex -2,0,-3
vertex -2,5,-3
vertex 2,0,-3
vertex 2,5,-3
vertex 7,0,0
vertex 7,5,0
vertex 10,0,0
vertex 10,5,0
face 0,1,3,2
face 2,3,5,4
face 4,5,7,6
face 6,7,9,8
color 220,240,200

[meshbuilder]  ;roof code
vertex -7,5,0
vertex 0,10,5
vertex -2,5,-3
vertex 2,5,-3
vertex 7,5,0
vertex 10,5,0
vertex 10,10,5
face 0,1,2
face 2,1,3
face 3,1,4
face 1,6,5,4
color 120,120,120
```



Save and view .....
And the wall and roof should look like this.....

Well done. You have made good progress and should now be able to understand how to construct shapes and apply colour to the faces.
In the next part we shall see how we can apply the textures (pictures) to the mesh.
This is where many fall by the wayside, let's try and make it painless!

Lesson 4 – Applying photographic textures

Items needed to do this stuff

Exactly the same as part one plus additional files which you should download from the same place this guide came from. You should save the files you make in the course of this tutorial in the same folder as the free provided ones to be sure they get loaded when called for.

[texture]

This is the start of the code section to order and locate the picture, or texture as it is usually called.

Applying photo textures to face

In part one we learned how the object framework, or 'mesh', was formed and we described the object faces according to the vertices involved. We coloured the faces using RGB values.
In this part we will begin to understand how a photographic 'texture' can be applied to a face. Before we start on an abject file let's look at the specification for a suitable image to use.

File format: All bve versions can use windows bitmap format files (.bmp), openBVE can also use .gif and .png files.
Image files are usually converted from RGB colour to Index colour to keep file size small and prevent low frame rate performance in the program. OpenBVE can handle RGB files more efficiently than bve4 if it is felt necessary to use them.
Prepared images are scaled to 'the rule of 2'. That means that the dimensions of the sides in pixels follow the pattern "2, 4, 8, 16, 32, 64....." and so forth. This means that sometimes the picture looks 'out of proportion' when it is finally sized to use but, don't worry, bve rescales it when it's loaded into the program. If you use a picture editor to inspect examples of textures in object folders you will often see picture sizes such as '256x256' or 128x64'. Occasionally one will show up that isn't a 1:1 or 2:1 ratio - if there is a 32 x 2048 that's fine as each side is still in accord with the >2 rule.,

Original (above) looks a bit "squashed" when changed to rule >2 proportion, on the right (in this case 256x128 pixels) but no need to worry, the bve program will rescale it to fit the face.

To apply the texture (picture) to the chosen face we need to do two things:

Load the picture

The picture files in these examples are all assumed to be in the same folder as the object files we are coding. In the .b3d file format, start these lines under the 'face' commands like so,

```
[texture]
load filename.bmp
```

The file extension (.bmp, .png etc) is always included.

coordinates [i], x, y

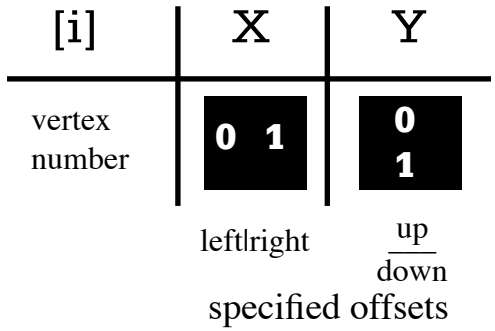
coordinates order the relation of texture parts to the vertex, see text on page 3 for explanation.

Set the texture coordinates

Think about any picture, it has four corners, top, bottom, left and right. All we are doing when we set the texture coordinates is that we pin the correct corner to the correct vertex.

Use this simple example of the coding shown below, it is a simple rectangle with four vertices which form one face, and it is to this face that we will apply a texture:
(make an object file from this given coding and view it in Structure Viewer)
Tip: It sometimes helps to think about what we are trying to do and sketch it roughly with a pencil and paper to get it 'in the head'. Don't worry if at first it's a bit slow, as you gain experience you will visualise objects in terms of vertices more instinctively.

A simple rectangle with four vertices which form one face, just like the one we made at the start. Instead of a plain colour, we shall apply a texture to it



What the three coordinate numbers define

```
[meshbuilder]
vertex -2,3,0        ;vertex 0 - top left corner
vertex 2,3,0         ;vertex 1 - top right corner
vertex 2,0,0         ;vertex 2 - bottom right corner
vertex -2,0,0        ;vertex 3 - bottom left corner
face 0,1,2,3
```

The text after the semicolon (;) shows which picture corner should fit which vertex.

The coordinates command line goes like this:

```
coordinates [i], x, y
```

where i is the index of the vertex (in this case 0,1,2 or 3)
where x is the offset of the image, 0 for left or 1 for right
where y is the offset of the image, 0 for top or 1 for bottom

Look at this line of code:

```
coordinates 0, 0, 0    ;vertex number 0, x offset =0/left,
                        y offset =0/top,
coordinates 1, 1, 0    ;vertex number 1, x offset =1/right,
                        y offset =0/top,
coordinates 2, 1, 1    ;vertex number 2, x offset =1/right,
                        y offset =1/bottom,
coordinates 3, 0, 1    ;vertex number 3, x offset =0/left,
                        y offset =1/bottom,
```

So each picture corner is related to its rightful vertex in the object file.

Any picture will have four corners, top T, bottom B, left L and right R.

Take a look at the picture on the left. The corners are marked T,L; T,R; B,R and B,L for top, bottom right and left. (the pic is in the downloads, called kids.bmp)
Let's take the mesh code at the top and apply the pic to it.
(the lines of text to the right of the semicolons can stay, they aren't part of the working code as the program ignores a "comment" after a semicolon. You can make use of this later if you need to make notes or identifiers in your object file, e.g. ;left side,)

```
[meshbuilder]
vertex -2,3,0    ;vertex 0 - top left corner
vertex 2,3,0     ;vertex 1 - top right corner
vertex 2,0,0     ;vertex 2 - bottom right corner
vertex -2,0,0    ;vertex 3 - bottom left corner
face 0,1,2,3
[texture]
load kids.bmp
coordinates 0, 0, 0    ;vertex number 0, x offset =0/left,
                        y offset =0/top,
coordinates 1, 1, 0    ;vertex number 1, x offset =1/right,
                        y offset =0/top,
coordinates 2, 1, 1    ;vertex number 2, x offset =1/right,
                        y offset =1/bottom,
coordinates 3, 0, 1    ;vertex number 3, x offset =0/left,
                        y offset =1/bottom,
```

Save the file, call it kids.b3d and take a look at it in your Structure Viewer.
(you might want to key 'PageUp' to enlarge it)
If it looks like this, then all is well.

Lesson 5 – Playing with coordinates

Next, we will play with the texture coordinates and see how we can easily flip or turn the texture if we want to.

Things to do with texture coordinates

```
[meshbuilder]
vertex -2,3,0
vertex 2,3,0
vertex 2,0,0
vertex -2,0,0
face 0,1,2,3
[texture]
load kids.bmp
coordinates 0, 0, 0
coordinates 1, 1, 0
coordinates 2, 1, 1
coordinates 3, 0, 1
```

In the last example (above) we applied a texture (picture) which showed a boy on the left and a girl on the right. We can flip the picture horizontally by swapping the x offsets.
Remember that if the x offset is 0 it sets the left side of the original picture and that if the x offset is 1 it sets the right side of the original picture.

Change .....

```
coordinates 0, 0, 0
coordinates 1, 1, 0
coordinates 2, 1, 1
coordinates 3, 0, 1
```

To .....

```
coordinates 0, 1, 0
coordinates 1, 0, 0
coordinates 2, 0, 1
coordinates 3, 1, 1
```

Save and view .....
The boy is now on the right, we have reversed the picture.

It works the same way too with the Y offsets, change it again to ....

```
coordinates 0, 1, 1
coordinates 1, 0, 1
coordinates 2, 0, 0
coordinates 3, 1, 0
```

Save and view ....

This time the boy is still on the right but, quite obviously, the picture is now upside down.

Return to the original setting of the coordinates

```
coordinates 0, 0, 0
coordinates 1, 1, 0
coordinates 2, 1, 1
coordinates 3, 0, 1
```

The offset numbers are WHOLE numbers here, but try changing the x offsets to this, then save and view:

```
coordinates 0, 0, 0
coordinates 1, 2, 0
coordinates 2, 2, 1
coordinates 3, 0, 1
```

By changing the X offset value (top and bottom) to 2 we find the texture has repeated TWICE across the x axis.

Changing again to this:

```
coordinates 0, 0, 0
coordinates 1, 0.5, 0
coordinates 2, 0.5, 1
coordinates 3, 0, 1
```

And only HALF the texture (the left half) is displayed.

Another adjustment, like so, and only the right half is displayed.

```
coordinates 0, 0.5, 0
coordinates 1, 1, 0
coordinates 2, 1, 1
coordinates 3, 0.5, 1
```

Hopefully, by now a pattern will be emerging and you will see now for yourself how and why it is working. Once these basics are well learned and experienced you will be able to progress.

Later we will need to learn how to distribute a single texture over more than one face. The texture coordinates are a mighty power when you fully understand them. Getting it right at the early stages will allow you to progress, too many try to go forward without this understanding and, as often as not, fail. Practice with what you have learned and see how many objects you can make with what you now know to strengthen the understanding.

In the next, we will make a useful building for our own BVE route using our new found skills, and we shall be introduced to another object command, transparency.

Lesson 5 – Applying textures using the co-ordinates

Items needed to do this stuff

Windows Notepad, structure viewer pencil, paper and an image editing package plus additional files which you should download from the same place this guide came from. You should save the files you make in the course of this tutorial in the same folder as the free provided ones to be sure they get loaded when called for.

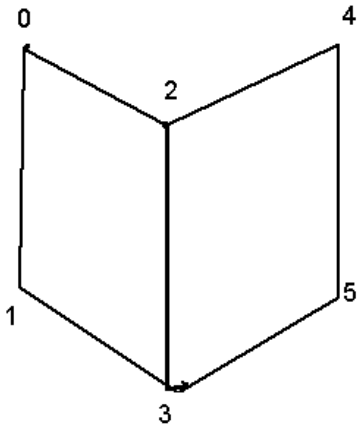
One texture for two sides of a small waiting room.

In this part

In part one we learned how the object framework, or 'mesh', was formed and we described the object faces according to the vertices involved. We coloured the faces using RGB values.
In part two we saw how a photographic 'texture' can be applied to a face. In this part we shall see how we can "wrap" an image around two or more adjacent faces using the co-ordinates for accurate placing of the texture.

Two faces, one texture

In part 2 we found how a texture is fixed to a face and that by altering the co-ordinates we could make part of the texture appear, or even make multiples of the texture appear.
It is useful to be able to spread a texture over more than one face. This is possible by ascertaining where in the texture the adjoining edge is located and defining this point to coincide with the correct vertex.
For an example see the picture of the small waiting room. The pebbledash side and the front are all one. We can wrap this texture around a two sided mesh.
Make the mesh first. In the sketch (right) are two faces, the vertices are numbered and we will define them in that order. Let the the side (x) be 3m, the height (y) be 2.5m and the front face (z) be 6m. Two faces are ordered clockwise. The code will be thus:



A mesh with two faces, vertices defined in the order as numbered.

```
[MeshBuilder]
Vertex -3, 2.5, 0
Vertex -3, 0, 0
Vertex 0, 2.5, 0
Vertex 0, 0, 0
Vertex 0, 2.5, 6
Vertex 0, 0, 6
```

Two faces are ordered clockwise

```
Face 0,2,3,1
Face 2,4,5,3
```

Load the texture provided, waitroom.bmp.

```
[Texture]
Load waitroom.bmp
```

This is to be shared out between the two faces.
At the moment we do not know the proportions so, by guesswork, let us attach the left third of the texture to the face 0,2,3,1 and the rest to the face 2,4,5,3

Coordinates 0, 0, 0
Coordinates 1, 0, 1
Coordinates 2, 0.3, 0
Coordinates 3, 0.3, 1

The side has continued around the front, so the coordinate value of 0.3 is incorrect.

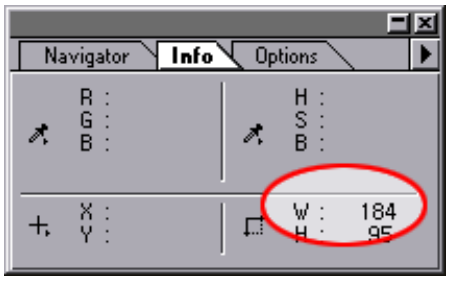
The x (horizontal) coordinates attach the left edge of the texture to the left side of the face and 0.3 (or three tenths) of the texture across the face to the right edge.
To continue, the left edge of the adjacent face starts at the same place on the texture and continues across to the right edge, so .....

```
Coordinates 4, 1, 0
Coordinates 5, 1, 1
```

Save the file, call it waitingroom.b3d and take a look at it in your Structure Viewer.

Well, OK, the texture **is** spread across the two faces, but the join between the two surfaces is in the wrong place, it isn't on the corner where it ought to be. Remember the figure of 0.3 **was** arrived at by guesswork! What we need to do is to find how the proper value can be found accurately. Look again at the texture. The size of the texture waitroom.bmp is 512x256 pixels. If we could count the number of pixels from the left of the texture to the corner join point we would be able to say what value we need with certainty.

You really need an image editing program with an information panel to tell you the position of the cursor. With the texture opened move the mouse until the pointer coincides with the corner join. Read off the number. Another way is to marquee from the left edge to the corner join and again read the number. In this case you should see that the corner is 184 pixels from the left edge.
Given this fact, and the width of the texture being 512 pixels, divide the 184 by 512 and you will arrive at a value of 0.3593. Substitute this value for the guesstimated 0.3 in the code.



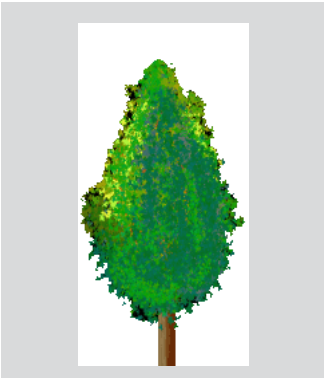
Determining the pixel count to the corner point using the info panel of an image editor.

```
[MeshBuilder]
Vertex -3, 2.5, 0
Vertex -3, 0, 0
Vertex 0, 2.5, 0
Vertex 0, 0, 0
Vertex 0, 2.5, 6
Vertex 0, 0, 6
Face 0,2,3,1
Face 2,4,5,3
[Texture]
Load waitroom.bmp
Coordinates 0,  0,    0
Coordinates 1, 0, 1
Coordinates 2,  0.3593, 0
Coordinates 3,  0.3593,   1
Coordinates 4, 1, 0
Coordinates 5, 1, 1
```

Save the file and take a look at it in your Structure Viewer. The corner join should now coincide perfectly with the join at the two faces.

This method of calculating the value of the coordinates works for both the x coords (horizontal) and the y coords (vertical) In the case of the y coords the pixel count is from the top downwards. A texture may be split across any number of adjacent faces. This is essential when constructing tunnels, rolling stock and other such objects.

Transparency



There are many instances where an object won't always have a straight edge, or be exactly rectangular. Think of a hedgerow or a tree, for example. Although the texture of the item is attached to a rectangular mesh the shape can be very irregular.
Take a look at the bush. If you were to construct a suitable size mesh with a face and attach the bush texture to the face you would have a bush object. The trouble is, you would have a bush object on a white rectangle, as white is the background colour.

Or look at the bridge arch. In this case the background colour is blue, but all the same if you put it on a suitable size mesh with a face and attached the bridge arch texture to the face you would have a bridge arch object with a solid blue looking through the hole and over the parapet. We don't want to see the white around the bush, or the blue through the arch. What can be done?

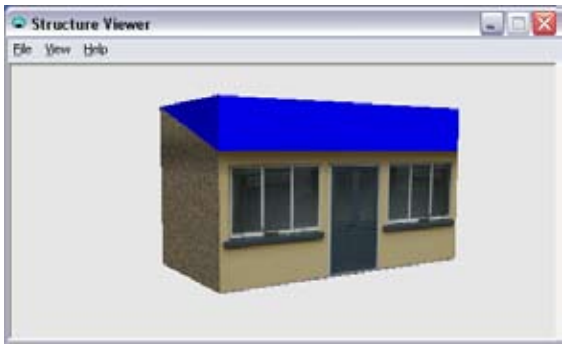
The answer is surprisingly easy - transparency! As long as we know the RGB colour value of the colour we would like to be invisible, then it can be told to vanish.

As an example of how transparency can work to help us take a look at this modified version of the waiting room texture, open the image file *waitroomB.bmp*

This version has a side that slopes down from the back to the front. The empty area of the texture happens to be filled with a pure blue background (more about which later).

Make a mesh, type or copy this:

```
[MeshBuilder]
Vertex -3, 3.45, 0
Vertex -3, 0, 0
Vertex 0, 3.45, 0
Vertex 0, 0, 0
Vertex 0, 3.45, 6
Vertex 0, 0, 6
Face 0,2,3,1
Face 2,4,5,3
[Texture]
Load waitroomb.bmp
Coordinates 0,  0,    0
Coordinates 1, 0, 1
Coordinates 2,  0.3593, 0
Coordinates 3,  0.3593,   1
Coordinates 4,  1,   0
Coordinates 5,  1,   1
```



Save the file as waitingroomB.b3d and take a look at it in your Structure Viewer.

The front and sides are fitted perfectly over the two faces but there is a blue area appearing over the front face and the sloping gable where the texture background colour is clearly visible. What is needed is for this background colour to be invisible. This is how. . . . .

Now, in this case the background colour has an RGB value of (R) 0, (G) 0, (B) 255.
Add the following line at the end of the code: Transparent 0,0,255

```
[MeshBuilder]
Vertex -3, 3.45, 0
Vertex -3, 0, 0
Vertex 0, 3.45, 0
Vertex 0, 0, 0
Vertex 0, 3.45, 6
Vertex 0, 0, 6
Face 0,2,3,1
Face 2,4,5,3
[Texture]
Load waitroomb.bmp
Coordinates 0, 0, 0
Coordinates 1, 0, 1
Coordinates 2,  0.3593, 0
Coordinates 3,  0.3593,   1
Coordinates 4, 1, 0
Coordinates 5, 1, 1
transparent 0,0,255
```

Save and view, the blue has gone from sight as it is now transparent and therefore invisible.

Add a simple roof for now:

```
[MeshBuilder]
Vertex -3, 3.45, 0
Vertex -3, 3.45, 6
Vertex 0, 2.5, 6
Vertex 0, 2.5, 0
Face 0,1,2,3
color 100,100,100
```

You can see how the transparent colour can be used for openings, irregular shapes, gable ends and countless other objects. In the case of the bridge arch and waiting room examples the transparent colour chosen was 0,0,255, a bright pure blue.

Any RGB value may be used for the transparent colour. The choice of transparent colour should be one which does not occur in the texture however, as this would result in “holes” appearing wherever that colour occurred. Also, be careful if the background is not clearly separated from the rest of the texture, or unwanted “fringeing” might be visible.

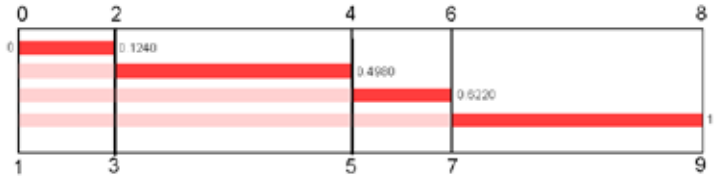
In practice blue (0,0,255), white (255,255,255) black (0,0,0) are commonly used but there are no hard and fast rules (except for x file format which always uses black, though that need not concern us here). Your image editing program should be able to select and replace backgrounds of any RGB value. With this facility you will be able to prepare your own textures with transparency, which is the key to constructing any scenic object you could possibly need.

One texture, four faces

In this final lesson for part 3 we will build a portakabin. The texture provided portacabin.bmp has all four sides in the one texture. It will be necessary to make a mesh with four faces with the vertices forming a cube. See the sketch showing the four faces, short long, short, long, from left to right. The cabin shall have a width (x) of 2.5m a height (y) of 2,5m and a length (z) of 5m. For this example it will be built so the centre of the object is located at x=0, z=0 and the base at rail top height =0. (This is also the way rolling stock exteriors are made for OpenBVE).
Define the vertices in the order numbered in the sketch and order the faces clockwise:

```
[MeshBuilder]
Vertex -1.25, 2.4, -2.5
Vertex -1.25, 0, -2.5
Vertex 1.25, 2.4, -2.5
Vertex 1.25, 0, -2.5
Vertex 1.25, 2.4, 2.5
Vertex 1.25, 0, 2.5
Vertex -1.25, 2.4, 2.5
Vertex -1.25, 0, 2.5
Vertex -1.25, 2.4, -2.5
Vertex -1.25, 0, -2.5

Face 0,2,3,1
Face 2,4,5,3
Face 4,6,7,5
Face 6,8,9,7
```



Before we load the texture let's be precise about the coordinate values needed to pin the corners exactly to the vertices. Open the texture portacabin.bmp in the image editor and using the same method as used in Lesson 5 locate the first corner from the left and note it down. You will find it is 127 pixels, Starting again from the left measure the distance to the second corner. It is 510 pixels. Again from the left measure to the third corner, which I make to be 637 pixels.

Now our coordinate values will be, from left, the pixel count divided by the total width of the texture which here is 1024 pixels:
0÷1024=0
127÷1024=0.1240
510÷1024=0.4980
637÷1024=0.6220
1024÷1024=1

I hope you have enjoyed following the three eejits guides and by now you are starting to feel confident that you can tackle some object building on your own. The more you work at it the better you will become, it really does get easier as you progress.

Due to the many different image editing programs available I haven't started to explain some of the skills and tricks which are part of the object builder's repertoire. It is no understatement to say that good image editing skills are as vital for the object maker as a knowledge of the meshbuilding and use of the coordinates.

Never give up, as your skills grow so will your satisfaction level. There is nothing so satisfying as seeing your work coming together, looking good and being able to say, "I made that!"

If you don't believe me, ask the cat!

Add the code to load the texture and enter the coordinates in the face instructions so that the correct part of the texture is applied to each face.

```
[Texture]
Load portacabin.bmp

Coordinates 0,0,0
Coordinates 1,0,1
Coordinates 2,0.1240,0
Coordinates 3,0.1240,1
Coordinates 4,0.4980,0
Coordinates 5,0.4980,1
Coordinates 6,0.6220,0
Coordinates 7,0.6220,1
Coordinates 8,1,0
Coordinates 9,1,1
```

Save the file as kabin.b3d and take a look at it in your Structure Viewer. Just add a simple roof and there it is, a useful addition for your route.

```
[MeshBuilder]
Vertex 1.25, 2.4, -2.5
Vertex -1.25, 2.4, -2.5
Vertex -1.25, 2.4, 2.5
Vertex 1.25, 2.4, 2.5
Face 0,1,2,3
color 100,100,100
```