1 Translating Geometric Objects

1.1 Duplicating and moving cubes

The following commands, which we have copied from the previous worksheet contain essentially everything we have done to this point to create our cube(s). In this chapter we will begin to make use of the colon ":" terminator for commands to suppress unwanted output. Note that the colon can always be replaced by a semicolon and the command re-executed if we need to look at the output.

```
front:=[v1,v2,v3,v4];
back:=[v5,v6,v7,v8];
left:=[v4,v8,v5,v1];
right:=[v3,v2,v6,v7];
bottom:=[v3,v4,v8,v7];
top:=[v1,v2,v6,v5];

v1:=[1,0,1]:  v2:=[1,1,1]:  v3:=[1,1,0]:
v4:=[1,0,0]:
v5:=[0,0,1]:  v6:=[0,1,1]:  v7:=[0,1,0]:  v8:=[0,0,0]:

box:=[top, bottom, right, left, front, back]:

bluetop:=
plots[polygonplot3d](top, style=patch,
color = blue ):
purplebottom:=plots[polygonplot3d]
(bottom, style=patch,color = purple):
yellowsides:=plots[polygonplot3d]([left,right,front,back],
style=patch,color = yellow):

plots[display]([bluetop, purplebottom,
yellowsides]);
```

It is evident that simply by changing the assigned values of v1..v8 we can create images of different cubes. However, rather than doing exactly that we will make another copy in this sheet which we will modify so that our previously defined objects will continue to have the same meaning. This illustrates a practice that is always good to follow. Copy and paste working code whenever possible (as opposed to re-typing it which almost always introduces mistakes). Since only the precise symbols we have defined have current meaning a slight change creates entirely new symbols and relationships. For instance we may elect to change the names of the vertices to w1,...,w8 and simply add a "w" to each of the previously defined words. (The reader will suspect immediately see that this could get out of hand if we keep doing the same thing - we will soon learn a more systematic approach).

```
frontw:=[w1,w2,w3,w4];backw:=[w5,w6,w7,w8];
leftw:=[w4,w8,w5,w1];rightw:=[w3,w2,w6,w7];
bottomw:=[w3,w4,w8,w7];topw:=[w1,w2,w6,w5];
```

Now we have a "template" for a second cube and need only assign values to the vertices w1...w8 as before. The question is which values to assign.

Let us assume we want to stack a duplicate of the first cube directly on top of it. To "lift" our first cube (which has height 1) straight up so that its base moves to the level of its top requires that the "z-coordinate" of each of its points be increased by exactly 1. That is if [a,b,c] is a point on the cube that that point moves to [a,b,c+1]. We don't need to explicitly move every point,

only the vertices. Moreover we note that with an obvious meaning for addition $[a,\, b,\, c + 1] = [a,\, b,\, c] + [0,\, 0,\, 1]$. Maple (Relese 4 and higher) understands this addition so, assuming that the vertex w1 is where v1 goes, w2 is where v2 goes, ..., etc. we can assign values to our vertices w1...w8 as follows:

```
shift:=[0,0,1];

w1:=v1+shift;w2:=v2+shift;w3:=v3+shift;w4:=v4
+shift;w5:=v5+shift;
w6:=v6+shift;w7:=v7+shift;w8:=v8+shift;

boxw:=[topw, bottomw, rightw, leftw, frontw,
backw];

bluetopw:=plots[polygonplot3d](topw,
style=patch,color = blue ):
purplebottomw:=plots[polygonplot3d](bottomw, style=patch,
color = purple):
yellowsidesw:=plots[polygonplot3d]([leftw,rightw,frontw,backw],
style=patch,color = yellow):
```

Let's put in an origin to show the shift. Note a list with one point in it is a polygon.

```
ORIGIN := plots[polygonplot3d]([[0,0,0]]):

plots[display]([ ORIGIN,bluetopw,
purplebottomw, yellowsidesw], axes=normal);
```

Reference to the axes verifies that our new cube is indeed a copy of the old one shifted up exactly one uint. The question now is how to display them at the same time. The answer is that we do exactly what we did before.

```
plots[display]([ plots[display]([bluetop,
purplebottom, yellowsides]),
plots[display]([bluetopw, purplebottomw, yellowsidesw], axes=normal)]
);
```

The command above obviously stacked our cubes although we do see here the need to tell Maple not to scale the image if we want the cubes to appear as such. That is easily remedied. More important is to understand how what the above command did and how to abbreviate it. The plots[display]

command takes in *lists of plot structures and outputs a plot structure. Each of our cubes is such an output and . Thus this command is of the form plots[display]([plotstructure1, plotstructure2], axes=normal). We can make the assignments:*

```
cube1:=plots[display]([bluetop,
purplebottom, yellowsides]):

cube2:=plots[display]([bluetopw,
purplebottomw, yellowsidesw]):
```

Then we can express the command which generated the "stacked" cubes in the form

```
plots[display]([cube1, cube2], axes=normal,
```

```
scaling=constrained);
```

Note that we have added the option scaling=constrained to give our pictures the proper scale.

There is no reason to for both cubes the same color or style, we can give the faces of cube2 any color or style we want. What if we want a cube3? The obvious ways to produce a cube3 are repeat what we did to get cube2 or even to take the tack we avoided then and simply modify our previous code to produce a cube3. Keeping in mind that once we have produced cube2 if we then redefine the vertices w1..w8 we can produce a separate cube3, retaining the *picture of our "original" cube2 but not cube2 itself. We could do the same thing to get a cube4, cube5, etc. If all we want are the pictures this will work but there are problems. For instance, suppose we wanted 100 cubes made them all yellow and then decided that they should be red. We would have to repeat all of these steps "by hand". Fortunately, Maple provides tools to automate this type of manufacturing process. What we need to do is build a new Maple word which takes the original cube and produces a set of "stacked" copies (or a picture of such a set) assembled according to our instructions. We will take up this process in the next chapter but first we want to actually build a few new cubes and assemble them in order to gain a firm grasp of exactly what process we want to automate. This will be a process we will repeat many times in this study. First let us block out what we did to get cube1.*

1.2 The steps in building a cube

1. Define the original, abstract cube "box" as a list of faces where each face was itself a list of vertices v1..v8.

2. Assign specific values to the vertices v1..v8. Doing so automatically caused Maple to interpret the abstract cube, box, as the specific cube with the vertices assigned to v1..v8.

3. Execute a command cube1:=plots[polygonplot3d](box, options) which produced a plot structure representing the current "value" of "box"

Now let us extend these steps to the process which produced cube2 - but this time re-assigning the values of the v1..v8 rather than introducing the new vertices.

4. Assign a specific value to the word "shift" (note there is nothing special about the word "shift", "yzkotqp" would work just as well but would be less intuitive and perhaps harder to remember).

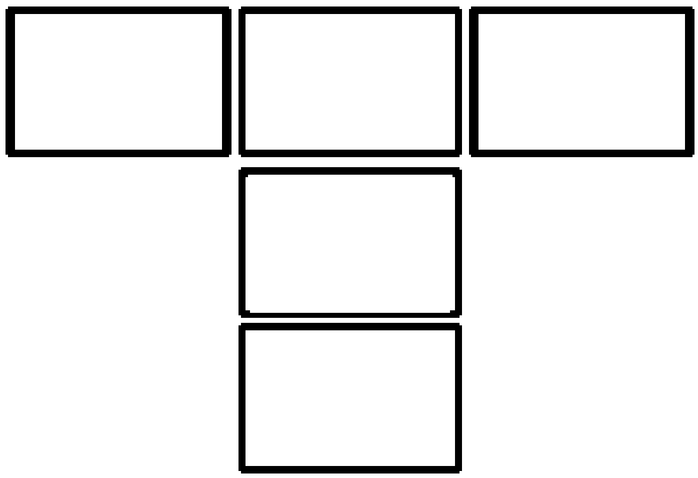
5. Assign new values to v1..v8 by the assignment commands v1:=v1+shift; v2:=v2+shift; ..., which say "the new value of v1 is the current value of v1 plus shift, etc.

6. Execute the command cube2:=plots[polygonplot3d](box, options)

7. go to step 4.

1.3 Building a TEE

Lets create an assemblage of cubes in the shape of a "T" as in the following sketch.



Although we don't really need to we will repeat the initial steps. This time we will use a different notation for the pictures of the faces which doesn't designate their

colors so that we may feel free to vary these as we wish. In addition, we introduce the "shift" immediately and simply let it be [0,0,0] to start.

```
v1:=[1,0,1]:  v2:=[1,1,1]:  v3:=[1,1,0]:
v4:=[1,0,0]:
v5:=[0,0,1]:  v6:=[0,1,1]:  v7:=[0,1,0]:  v8:=[0,0,0]:

shift:=[0,0,0]:

v1:=v1+shift:  v2:=v2+shift:  v3:= v3+shift:
v4:=v4+shift:
v5:=v5+shift:  v6:=v6+shift:  v7:=v7+shift:  v8:=v8+shift:
front:=[v1,v2,v3,v4]:back:=[v5,v6,v7,v8]:left:=[v4,v8,v5,v1]:
right:=[v3,v2,v6,v7]:bottom:=[v3,v4,v8,v7]:top:=[v1,v2,v6,v5]:

toppic:=plots[polygonplot3d](top,
style=patch,color = blue ):
bottompic:=plots[polygonplot3d](bottom, style=patch,
color = purple):
sidespic:=plots[polygonplot3d]([left,right,front,back], style=patch,
color = yellow):

cube1:=plots[display]([ toppic, bottompic,
sidespic]):
```

Now we have cube1, all we need do now is reset the shift and repeat the second group of commands, changing only the name of the result to "cube2".

```
v1:=[1,0,1]:  v2:=[1,1,1]:  v3:=[1,1,0]:
v4:=[1,0,0]:
v5:=[0,0,1]:  v6:=[0,1,1]:  v7:=[0,1,0]:  v8:=[0,0,0]:

shift:=[0,0,1]:
```

v1:=v1+shift: v2:=v2+shift: v3:= v3+shift: v4:=v4+shift: v5:=v5+shift: v6:=v6+shift: v7:=v7+shift: v8:=v8+shift: front:=[v1,v2,v3,v4]:back:=[v5,v6,v7,v8]:left:=[v4,v8,v5,v1]: right:=[v3,v2,v6,v7]:bottom:=[v3,v4,v8,v7]:top:=[v1,v2,v6,v5]:

```
toppic:=plots[polygonplot3d](top,
style=patch,color = blue ):
bottompic:=plots[polygonplot3d](bottom, style=patch,color = purple):
sidespic:=plots[polygonplot3d]([left,right,front,back], style=patch,
color = yellow):

cube2:=plots[display]([ toppic, bottompic,
sidespic]):
```

Now cube3 results from a shift of two units straight up

```
v1:=[1,0,1]:  v2:=[1,1,1]:  v3:=[1,1,0]:
v4:=[1,0,0]:
v5:=[0,0,1]:  v6:=[0,1,1]:  v7:=[0,1,0]:  v8:=[0,0,0]:

shift:=[0,0,2]:


v1:=v1+shift:  v2:=v2+shift:  v3:= v3+shift:
v4:=v4+shift:
v5:=v5+shift:  v6:=v6+shift:  v7:=v7+shift:  v8:=v8+shift:
front:=[v1,v2,v3,v4]:back:=[v5,v6,v7,v8]:left:=[v4,v8,v5,v1]:
right:=[v3,v2,v6,v7]:bottom:=[v3,v4,v8,v7]:top:=[v1,v2,v6,v5]:

toppic:=plots[polygonplot3d](top,
style=patch,color = blue ):
bottompic:=plots[polygonplot3d](bottom, style=patch,color = purple):
sidespic:=plots[polygonplot3d]([left,right,front,back], style=patch,
color = yellow):

cube3:=plots[display]([ toppic, bottompic,
sidespic]):
```

Cube4 is two up and one to the "right"

```
v1:=[1,0,1]:  v2:=[1,1,1]:  v3:=[1,1,0]:
v4:=[1,0,0]:
v5:=[0,0,1]:  v6:=[0,1,1]:  v7:=[0,1,0]:  v8:=[0,0,0]:

shift:=[-1,0,2]:


v1:=v1+shift:  v2:=v2+shift:  v3:= v3+shift:
v4:=v4+shift:
v5:=v5+shift:  v6:=v6+shift:  v7:=v7+shift:  v8:=v8+shift:
front:=[v1,v2,v3,v4]:back:=[v5,v6,v7,v8]:left:=[v4,v8,v5,v1]:
right:=[v3,v2,v6,v7]:bottom:=[v3,v4,v8,v7]:top:=[v1,v2,v6,v5]:

toppic:=plots[polygonplot3d](top,
style=patch,color = blue ):
bottompic:=plots[polygonplot3d](bottom, style=patch,color = purple):
```

```
sidespic:=plots[polygonplot3d]([left,right,front,back], style=patch,
color = yellow):

cube4:=plots[display]([ toppic, bottompic,
sidespic]):
```

And cube5 results from a shift of two up and one to the "left"

```
v1:=[1,0,1]:  v2:=[1,1,1]:  v3:=[1,1,0]:
v4:=[1,0,0]:
v5:=[0,0,1]:  v6:=[0,1,1]:  v7:=[0,1,0]:  v8:=[0,0,0]:

shift:=[1,0,2]:


v1:=v1+shift:  v2:=v2+shift:  v3:= v3+shift:
v4:=v4+shift:
v5:=v5+shift:  v6:=v6+shift:  v7:=v7+shift:  v8:=v8+shift:
front:=[v1,v2,v3,v4]:back:=[v5,v6,v7,v8]:left:=[v4,v8,v5,v1]:
right:=[v3,v2,v6,v7]:bottom:=[v3,v4,v8,v7]:top:=[v1,v2,v6,v5]:

toppic:=plots[polygonplot3d](top,
style=patch,color = blue ):
bottompic:=plots[polygonplot3d](bottom, style=patch,color = purple):
sidespic:=plots[polygonplot3d]([left,right,front,back], style=patch,
color = yellow):

cube5:=plots[display]([ toppic, bottompic,
sidespic]):
```

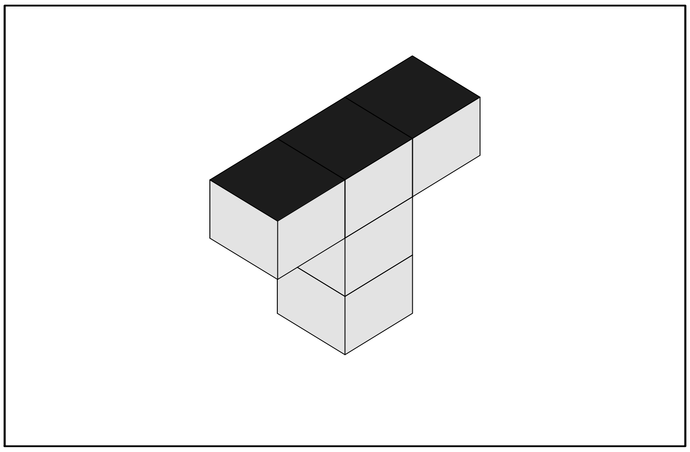
We have not taken the time to change the colors on our various cubes but we could have, simply by changing the options for each of them. Lets make our cubes into a list and display them

```
cubes:=[cube1,cube2,cube3,cube4,cube5]:
```

Finally, we can display our cubes. Since its to be a "T" we'll call it "Tee". The following command will then assign that plot structure to the symbol "Tee". It will not display it. On the other hand, Tee then becomes a Maple word whose value is that plot structure so when Maple receives "Tee" from the command line it will respond return "Tee" in the manner it represents such structures - namely by plotting them.

```
Tee:=plots[display](cubes,
scaling=constrained):

Tee;
```



1.3.1 Exercise: The "T" generated previously is an example of a figure that can be made by five identical cubes such that each of them share a face with at least one other. There is an "L" figure which can be made with 4. Find all of the figures of this type which can be made using either three or four cubes and construct a Maple word for each one which displays it. We regard two figures as the same if they are congruent in the sense of geometry - that is if one could be moved and rotated in space to become coincident with the other. Include a careful explanation of why your list is complete.

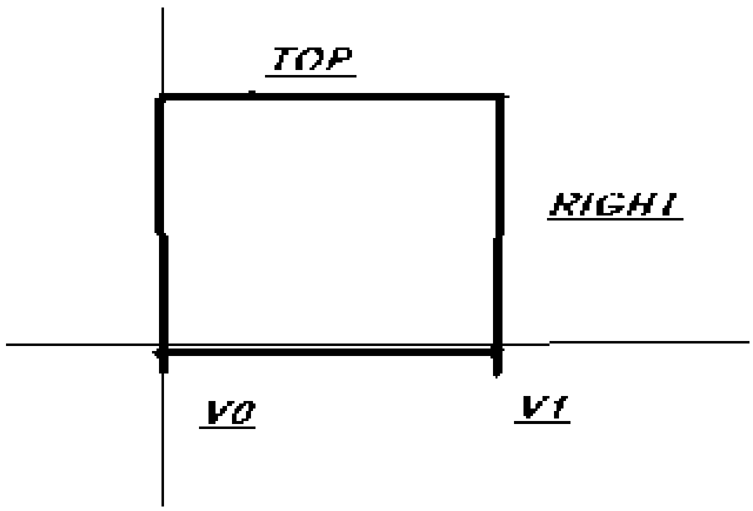
1.4 Exercises: More Pictures in the Plane

Although we have concentrated on "3d" graphics in the text one often wants to work with "2d" versions of the same problems. Not only may the problem be inherently two dimensional but computer resources required for three dimensions may severely limit what we can do on a particular machine. In this exercise we will discuss translating planar boxes.

1.4.1 *0. If you have not read through and executed the main worksheet for this exercise go to the beginning of this worksheet and execute the first few commands.*

1.4.2 *1. Copy this entire exercise set into a new worksheet and begin work in the new worksheet.*

1.4.3 *3.. Use "Paint" or a similar program to draw a figure like this oneLabel the upper right vertex "V2" and the upper left "V3". Label the other edges "left" and "bottom" as appropriate*



PASTE YOUR DIAGRAM IN THIS AREA

1.4.4 *4. Enter the commands* `top; right; left; bottom;`

1.4.5 *at the prompt. If Maple returns anything but these names then they already have assigned values - this will be the case if you have just executed the first few commands in the worksheet. Note that all worksheets open at any one time share a common set of variables. Thus (assuming exercise 1 was done) even though we are in a different worksheet, the value of "top" from the original is shared by this "new" one.*

1.4.6 *We want to assign new values to these symbols for our "plane" work. For instance we will want "bottom" to be the list "[v0,v1]" and to represent the bottom edge of the figure we have drawn. If we are careful in making these assignments then we will simply "overwrite" the previous meanings of the words with new assignments. However were we to forget to actually make the assignment for, say, "left" then subsequent execution errors will occur which can be quite difficult to track down. In addition, when we change the value of "top" in the new worksheet its "new" value will be used in subsequent calculations in other worksheets - but will not affect the current values from earlier calculations. That is, it doesn't work like a spreadsheet, previous calculations are not automatically "updated". In most cases the simplest thing to do is issue the command restart; This resets all variables to their default value which is simply their name. That is what we will do. Issue the command restart;*

1.4.7 *Now re-enter the commands* `top; right;` *left; bottom;*

1.4.8 *5. Assign the list "[v0,v1]" to the name "bottom" with the command* `bottom:=[v0,v1];` *Similarly assign the appropriate lists to "right", "left", and "bottom". Assign the name "box" to the list [bottom, right, top, left]*

1.4.9 *6. The computer is unable graphically interpret quantities that are not completely described, A person can relate the list [v0,v1] to a general line segment but the computer needs to know precisely what v0 and v1 are in order to interpret it as a line segment. Issue the command* `plots[polygonplot]([v0,v1]);` *then issue the command plots[polygonplot]([[0,0],[1,1]]);*

1.4.10 *7. Assign values [0,0] to v0, [1,0[to v1, [1,1] to v2, and [0,1] to v3 and have Maple draw the edges of the square with the command plots[polygonplot] Draw the top and left edges at the same time with plots[polygonplot]([top, left]); Draw the entire box with plots[polygonplot](box);*

NOTE: If/when you subsequently re-define the values of v0,v1,v2,v3 if you do not also "update" the interpretation of "box" (for instance by re-issueing the defining command that defined "box") then the command `plots[polygonplot](box);` will draw the *previously* defined box.

1.4.11 *8.Recall that there are many options available to enhance pictures drawn with Maple. Some of them are "axes", "scaling", "color", "thickness" and "style". These and the others are described in the "help" system. Some comon possibilities are:*

1.4.12

axes = none, axes = framed, axes = boxed,
scaling = constrained , scaling = unconstrained,
color = red, color = tan, color = blue,
style = patch, style = POINT, style = LINE,
thickness = 2, thickness = 4

1.4.13

1.4.14 *Multiple options can be invoked in the same plot. For instance ,*

1.4.15 *plots[polygonplot](box,color=green, style=patch,axes=none, thickness=4); will draw the box with no axes, and will "color it in" in green. Specifying a color will automatically set the style to "patch" so if one wants just the edges of a polygon colored one invokes the LINE style.*

*

1.4.16 *10. If P= [a,b] is a point in the plane and we want the point Q which is the result of shifting P by 1 to the right and up 4 then Q=[a+1,b+4]. Similarly if we want Z which is the result of shifting P by 3 to the left and down 7 then Z=[a -3,b-7]. We can write Q= P+[1,4] and Z = P+[-3,-7]. Beginning with release 4, Maple understands this "list arithmetic" which allows us to instruct Maple to "shift" objects in the plane.*

1.4.17 *If v0,v1,v2,v3 are the vertices of a box which is shifted so that each point of the box move "a" to the right and "b" to the left then if we "shift" be [a,b], the shifted box is [v0+shift,v1+shift, v2+shift,v3+shift]. If "shift" is [0,0] then this is simply the original box.*

1.4.18 *We want to define a new "abstract" box called "sbox" (for "shifted box) which contains the (currently undefined) quantity "shift". However if we do the "obvious" something which might not be expected will happen if we have been doing the previous exercises in the current session. Execute the command sbox:=[v0+shift,v1+shift, v2+shift,v3+shift];*

If you got the error message "adding lists of different length" this was probably because v0...v3 had meaning as lists of length 2 from the previous work while "shift" is just a name with no such meaning. One way out of this is to use symbols other than v0,v1..,v3. Edit the previous command to sbox:=[w0+shift,w1+shift, w2+shift,w3+shift]; If w0..etc had not been previously assigned some meaning and "shift" hasn't acquired one then this will work fine. However it begs the question of what to do if we subsequently need "w0" for another purpose. What one needs to do is "reset" v0,v1, etc (and possibly "shift") to their default values. In principle we could use the command restart but this would reset all variables in all worksheets in this session.

The commands `v0:='v0'; v1:='v1'; v2:='v2';` v3:='v3'; shift:='shift'; assign to each of these varaiables its own name, which is its default value. Once these are executed the definition of "sbox" will work fine.

1.4.19 *11. With "sbox" abstractly defined one can proceed as before to draw general quadralaterals in the plane by specifying (assigning) values for v0...v3 and "shift".*

1.4.20 *Assign v0,v1,v2,v3 the values [0,0], [1,0], [1,1], and [0,1], and "shift" the value [2,3]. The use the command*

1.4.21 *plots[polygonplot](sbox, color = green, style = patch, axes = normal);*

1.4.22 *to draw the shifted box.*

1.4.23 *12. One can assign a name to a plot and subsequently display that plot by invoking its name. For instance joe:=plots[polygonplot](box,color=blue, style=LINE, thickness=4, axes=normal); will display Maple's internal description of the figure, In this case the internal description is very short and interesting to view. In general it may run many, many pages and one does not want to see it. This is a case where the colon delimiter is very useful. Run joe:=plots[polygonplot](box,color=blue, style=LINE, thickness=4, axes=normal): then joe;*

1.4.24 *13. With the ability to make and "store" different pictures we can collect different pictures and display them together, creating more complex images. For instance the following is simply a collection of images of squares made one at a time. Each is a shift of the basic "unit box", (v0=[0,0], v1=[1,0],v2=[1,1],v3=[0,1]), made by a sequence of commands of the form*

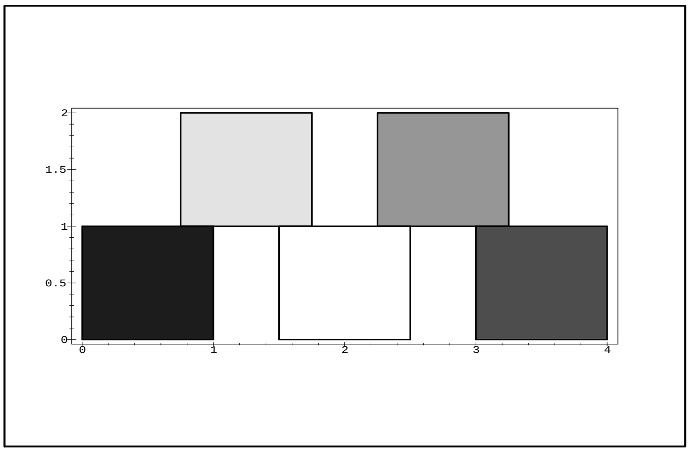
1.4.25 *shift:= [1,3]; sbox:=[v0+shift,v1+shift,v2+shift,v3+shift]; bx1:=plots[polygonplot](sbox,color=green, style=patch,axes=normal*

1.4.26

1.4.27 *The command to display the members of a list a collection of plots images at the same time is plots[display]. Thus for instance if, using the above command sequence one makes images 4 images bx1, bx2, bx3, bx4t. The command to display them all at once would be plots[display([bx1,bx2,bx2,bx3,bx4]); Of course one could make a list bx:=[bx1,bx2,bx3,bx4]; and simply do plots[display](bx);*

1.4.28 *Note that one can use different plot options in each image. One can also include options in plots[display]: plots[display](bx,scaling=constrained);*

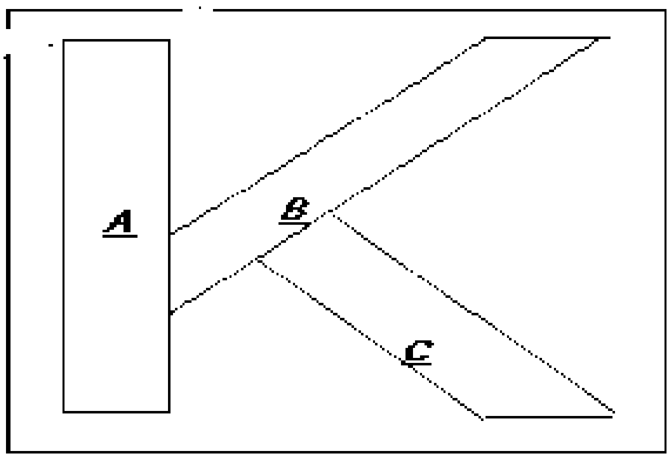
1.4.29 *Have Maple draw this picture.*



.

1.5 Exercises: Making 3d Alphabet Blocks

Introduction: One often finds colored cubes as children's toys and these often have letters painted on them. Lets investigate the possibility of making a white cube with a blue "K" on one face. The white cube is no problem, we've done that before. We need to make the "K". We know how to make polygons so we want a "K" thats made of them.



This one is made of the polygons A,B, and C. All we need to do is draw something like this on a piece of graph paper, read off the coordinates of the vertices, make up each of the polygons and put them together.

```
A:=[[1/10,0,1/10],[1/10,0,9/10],[3/10,0,9/10]
,[3/10,0,1/10]];
```

$$A := [[\frac{1}{10}, 0, \frac{1}{10}], [\frac{1}{10}, 0, \frac{9}{10}], [\frac{3}{10}, 0, \frac{9}{10}], [\frac{3}{10}, 0, \frac{1}{10}]]$$

```
B:=[[3/10,0,4/10],[3/10,0,6/10],[6/10,0,9/10]
,[9/10,0,9/10]];
```

$$B := [[\frac{3}{10}, 0, \frac{2}{5}], [\frac{3}{10}, 0, \frac{3}{5}], [\frac{3}{5}, 0, \frac{9}{10}], [\frac{9}{10}, 0, \frac{9}{10}]]$$

```
C:=[[4/10,0,5/10],[6/10,0,6/10],[9/10,0,1/10]
,[7/10,0,1/10]];
```

$$C := [[\frac{2}{5}, 0, \frac{1}{2}], [\frac{3}{5}, 0, \frac{3}{5}], [\frac{9}{10}, 0, \frac{1}{10}], [\frac{7}{10}, 0, \frac{1}{10}]]$$

```
K:=plots[polygonplot3d]([A,B,C],color=blue,st
yle=patch):
```

Now we need the cube. As before we retrieve our basic box construction

```
v1:=[1,0,1]:  v2:=[1,1,1]:  v3:=[1,1,0]:
v4:=[1,0,0]:
v5:=[0,0,1]:  v6:=[0,1,1]:  v7:=[0,1,0]:  v8:=[0,0,0]:

box:=[top, bottom, right, left, front, back]:

front:=[v1,v2,v3,v4];back:=[v5,v6,v7,v8];left
:=[v4,v8,v5,v1];
right:=[v3,v2,v6,v7];bottom:=[v3,v4,v8,v7];top:=[v1,v2,v6,v5];
```

$$front := [[1, 0, 1], [1, 1, 1], [1, 1, 0], [1, 0, 0]]$$

$$back := [[0, 0, 1], [0, 1, 1], [0, 1, 0], [0, 0, 0]]$$

$$left := [[1, 0, 0], [0, 0, 0], [0, 0, 1], [1, 0, 1]]$$

$$right := [[1, 1, 0], [1, 1, 1], [0, 1, 1], [0, 1, 0]]$$

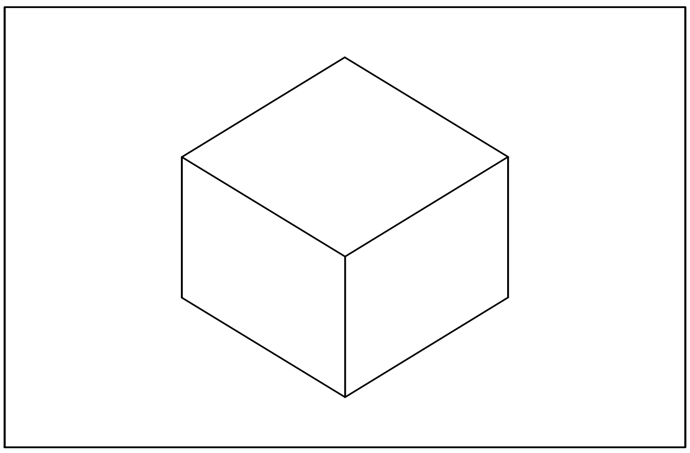
$$bottom := [[1, 1, 0], [1, 0, 0], [0, 0, 0], [0, 1, 0]]$$

$$top := [[1, 0, 1], [1, 1, 1], [0, 1, 1], [0, 0, 1]]$$

```
whitecube:=plots[polygonplot3d](box,
color=white, style=patch,thickness=3,scaling=constrained);
```

$$whitecube := \mathbf{PLOT3D}(\mathbf{POLYGONS}([[1., 0, 1.], [1., 1., 1.], [0, 1., 1.], [0, 0, 1.]], [[1., 1., 0], [1., 0, 0], [0, 0, 0], [0, 1., 0]], [[1., 1., 0], [1., 1., 1.], [0, 1., 1.], [0, 1., 0]], [[1., 0, 0], [0, 0, 0], [0, 0, 1.], [1., 0, 1.]], [[1., 0, 1.], [1., 1., 1.], [1., 1., 0], [1., 0, 0]], [[0, 0, 1.], [0, 1., 1.], [0, 1., 0], [0, 0, 0]]), \mathbf{STYLE}(\mathit{PATCH}), \mathbf{SCALING}(\mathit{CONSTRAINED}), \mathbf{THICKNESS}(3), \mathbf{COLOUR}(\mathit{RGB}, 1, 1, 1))$$

```
Kcube:=plots[display]([K,whitecube]):
Kcube;
```



There are two problems here. The first is a very sloppy "K" which results from inaccurate reading of coordinates from a hastily drawn graph. This can be partially addressed by more careful graphing. The second problem is that some views show a two-toned "K". This is what happens when two patch-style pictures are in contact. The patch pictures are made of colored triangles occupying the same positions and the computer doesn't know which onw we really want to see. The solution is to move them slightly apart. In this case its we might have the "K" as being slightly raised above the surface of the cube. That just means altering the definition of K

```
A:=[[1/10,-.01,1/10],[1/10,-.01,9/10],[3/10,-
.01,9/10],[3/10,-.01,1/10]];
```

$$A := [[\frac{1}{10}, -.01, \frac{1}{10}], [\frac{1}{10}, -.01, \frac{9}{10}], [\frac{3}{10}, -.01, \frac{9}{10}], [\frac{3}{10}, -.01, \frac{1}{10}]]$$

```
B:=[[3/10,-.01,4/10],[3/10,-.01,6/10],[6/10,-
.01,9/10],[9/10,-.01,9/10]];
```

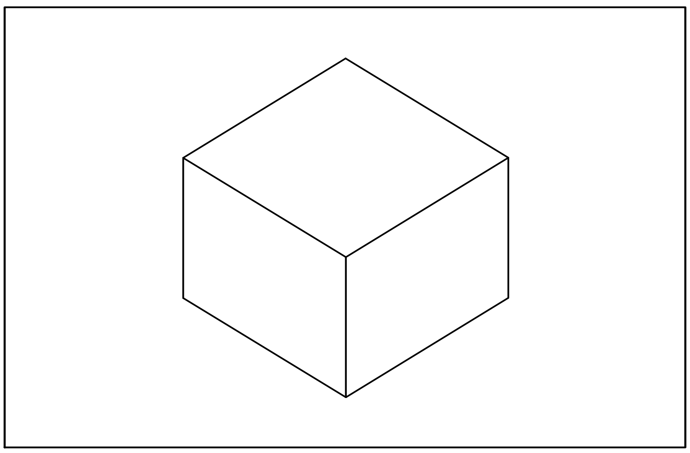
$$B := [[\frac{3}{10}, -.01, \frac{2}{5}], [\frac{3}{10}, -.01, \frac{3}{5}], [\frac{3}{5}, -.01, \frac{9}{10}], [\frac{9}{10}, -.01, \frac{9}{10}]]$$

```
C:=[[4/10,-.01,5/10],[6/10,-.01,6/10],[9/10,-
.01,1/10],[7/10,-.01,1/10]];
```

$$C := [[\frac{2}{5}, -.01, \frac{1}{2}], [\frac{3}{5}, -.01, \frac{3}{5}], [\frac{9}{10}, -.01, \frac{1}{10}], [\frac{7}{10}, -.01, \frac{1}{10}]]$$

```
K:=plots[polygonplot3d]([A,B,C],color=blue,st
yle=patch):
```

```
Kcube:=plots[display]([K,whitecube]):Kcube;
```

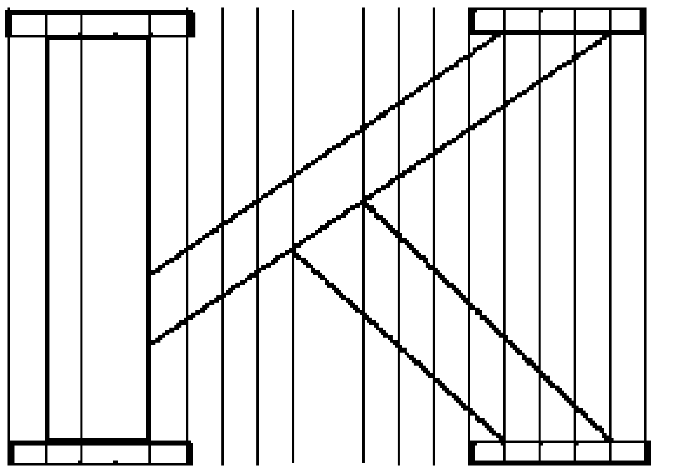


Its a simple matter to add a red "K" to the other side of this block.

```
A1:=[[1/10,1.01,1/10],[1/10,1.01,9/10],[3/10,1.01,9/10],[3/10,1.01,1/1
0]];
B1:=[[3/10,1.01,4/10],[3/10,1.01,6/10],[6/10,1.01,9/10],[9/10,1.01,9/1
0]];
C1:=[[4/10,1.01,5/10],[6/10,1.01,6/10],[9/10,1.01,1/10],[7/10,1.01,1/1
0]];
 K1:=plots[polygonplot3d]([A1,B1,C1],color=red,style=patch):
 plots[display]([K,whitecube,K1]);{
}
```

1.5.1 EXERCISE: Improve the "Kcube" by creating one with a better "K" by more careful graphing.

One way to do this is to draw a "K" on graph paper and take the coordinates from there. The command plots[coordplot](cartesian, view=[-10..10,-10..10], grid=[20,20]); will create a "graph paper" grid. Copy and paste it into "Paint" or a similar program and draw a better "K" from which coordinates of component polygons can be read.



Assume the lower left corner is the origin [0,0]. Then the "K" is made of seven polygons. For instance the three pieces comprising the left "post" are

```
P1:=[[0,0],[5,0], [5,1],[0,1]];
P2:=[[1,1],[4,1],[4,18],[1,18]];
P3:=[0,18], [5,18],[5,19],[0,19]];
```

Determine the other four polygons, make plots of each of these with commands of the form k2:=plots[polygonplot](p2, style=LINE, color=blue):

(note the seimcolon terminator). Then creats an image of the compete "K with newK:=plots[display]([k1,k2,k3,k4,k5,k6.k6]):

This has created a two dimensional K. Now follow the steps in creating the original "K" to produce a more carefully drawn "K-block".

1.5.2 *EXERCISE: Change the "Kcube" so that there is a "K" on each face.*

1.5.3 *EXERCISE: Make a yellow block with one the letters A,B,C,D,E,F on each face.*

1.5.4 *EXERCISE: Give each of the letters in the previous problem a different color.*

1.5.5 *EXERCISE: Make the block in the previous problem appear to be made of a transparent material with the colored letters painted on its faces. HINT: Wireframe*

1.5.6 *EXERCISE: Make a row of alphabet blocks that display your initials.*

1.5.7 *EXERCISE: Create a "solid" "K"*

1.5.8 *Exercise: The following are pictures of arrangements of children's blocks. Create a Maple image that "model's" each picture.*