1 Chapter 1 First steps – Drawing boxes

1.1 What is visual Problem solving?

Visual problem solving is concerned with problems whose statements or solutions (hopefully both) have a visual character which can be appreciated and even assessed without reference to the underlying mathematics. This is analogous to the manner in which we can appreciate music without reference to or even understanding of its

creator's tools - **the fact notwithstanding** those with such knowledge can appreciate it at a higher level. In the course of this study we will find that there is an incredible variety of problems of this nature and that there are techniques for developing families of them from basic contexts. In this introduction to the subject we will primarily restrict our attention to problems which can be analyzed with the non-calculus mathematics and basic science studied in middle and high school. However, since part of what we will be doing is generating new problems from old we must be prepared to find ourselves formulating questions in that vocabulary whose solution requries new tools. We will celebrate when we do for the quest for problems not solvable within a science is the most basic process in by which it is extended. problem solving is about marshalling the intellectual tools at our disposal to attack a problem.

The expected result of a serious attack on a problem is a *contribution* to its solution. A contribution may well fall short of a complete solution but it will always present one's *formulation* of the problem, one's approach to *analyzing* the problem, and the *results* of one's analysis. In keeping with our visual approach at least one of these **components will contain elements of a visual nature.** In this course our contributions will always take form on a Maple (release 4) worksheet.

Maple is a computer program (sometimes called a symbolic manipulation program) which can generate and manipulate algebraic, numerical, and geometric objects. It has a language of its own (the Maple language) through which it can be directed to perform virtually any mathematical or geometric operation. More over this language is *extensible* which means that, like any natural language, it can be used to define new words which the program can recognize and new symbols which it can manipulate. A Maple Worksheet is an electronic file containing text and inserted images (which have meaning only to people), Maple commands (words) which direct the program to perform certain operations, and output which is text, images, or other constructs generated by the program in response to the commands. In general the responses (also called the output) are not needed if one has the commands since they can always be recovered from the latter. Such a sequence of commands is typically a small amount of text which can readily be communicated by electronic mail. This is the form in which all work in this course - all contributions - will be made, and the primary form of correspondence among the participants in this course. Since this is the case our first order of business will be to begin to learn how to use Maple as a *mathematical communications* tool. We learn to do this by doing it. We start with the geometric vocabulary.

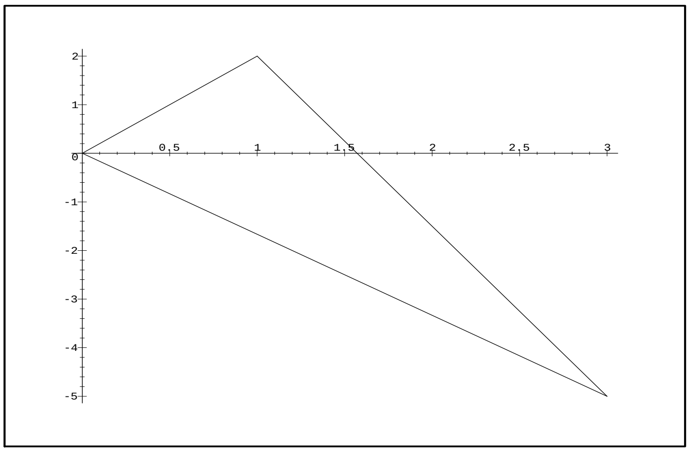
1.2 Mathematical Boxes

We assume that the reader is familiar with the fact that any two mutually perpendicular lines in the plane or three mutually perpendicular number lines in space define a *cartesian coordinate system* relative to which every point in the plane or space has a unique name, called its *coordinates*. Points in the plane or space may have other names such as "X","Lexington", or "Detroit", etc. and for a particular purpose these may be far more descriptive than a list of coordinates. The importance of coordinates is that that algebraic relationships among the coordinates of (collections of) points fully express geometric relationships among the points themselves. This means that we can use algebra to express and investigate geometric relationships (and visa versa). Moreover, since computers can "understand" and "communicate" algebraic relationships, this provides both a path to the physical/geometric/visual worlds we propose to explore and a vehicle take us at least part of the way there. The coordinates of a point are a list so our first task is learn about lists in Maple.

In Maple, lists are sequences of objects enclosed in brackets [ ] . The order of the objects in a list is important. Thus the lists [1,2,3] and [2,1,3] are different and the lists [a,b,c] and [d,e,f] are equal (as lists) only if a=d, b=e, and c=f. Note also that, to Maple, only the first of [1,2,3], {1,2,3}, and (1,2,3) is a list. Our initial Maple command to plot points in space is "plots[polygonplot3d]" which is a command which accepts lists of points and plots the (spatial) polygon with those points as its vertices. Thus if we want to create a visual representation of the spatial triangle with vertices [1,2,3], [4,1,5] and [7,8,3] we enter the following at the command prompt "$>$" . plots[polygonplot3d]([[1,2,3],[4,1,5],[7,8,3]]);

The analogous command which displays a polygon in the plane is "plots[polygonplot]."

```
plots[polygonplot]([ [ 0,0], [1,2],
[3,-5]]);
```

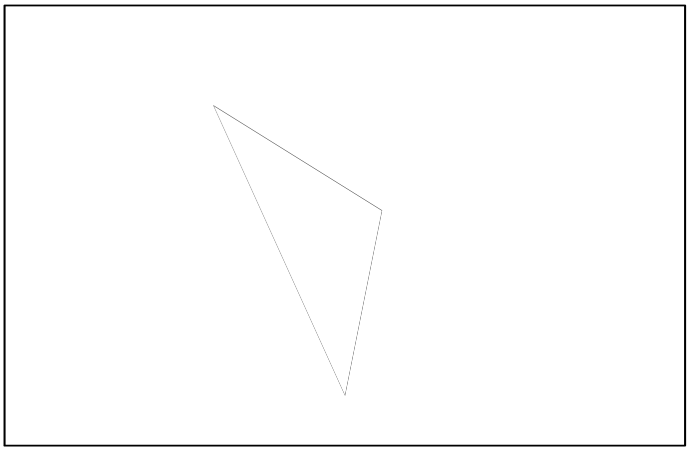


The "plots" part of these command tells Maple to look in the "package" of commands called "plots" for the word polygonplot3d . We can first tell Maple to "load" all of the words in the plots package into the current session by the command with(plots); after which it will recognize only the polygonplot3d part.

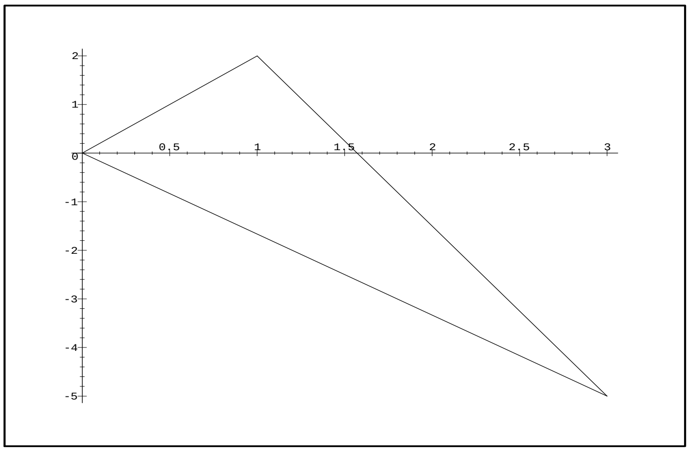
```
with(plots);
```

[*animate*, *animate3d*, *changecoords*, *complexplot*, *complexplot3d*, *conformal*, *contourplot*, *contourplot3d*, *coordplot*, *coordplot3d*, *cylinderplot*, *densityplot*, *display*, *display3d*, *fieldplot*, *fieldplot3d*, *gradplot*, *gradplot3d*, *implicitplot*, *implicitplot3d*, *inequal*, *listcontplot*, *listcontplot3d*, *listdensityplot*, *listplot*, *listplot3d*, *loglogplot*, *logplot*, *matrixplot*, *odeplot*, *pareto*, *pointplot*, *pointplot3d*, *polarplot*, *polygonplot*, *polygonplot3d*, *polyhedraplot*, *replot*, *rootlocus*, *semilogplot*, *setoptions*, *setoptions3d*, *spacecurve*, *sparsematrixplot*, *sphereplot*, *surfdata*, *textplot*, *textplot3d*, *tubeplot*]

```
polygonplot3d([[1,2,3],[4,1,5], [7,8,3]]);
```



```
polygonplot([ [ 0,0], [1,2], [3,-5]]);
```

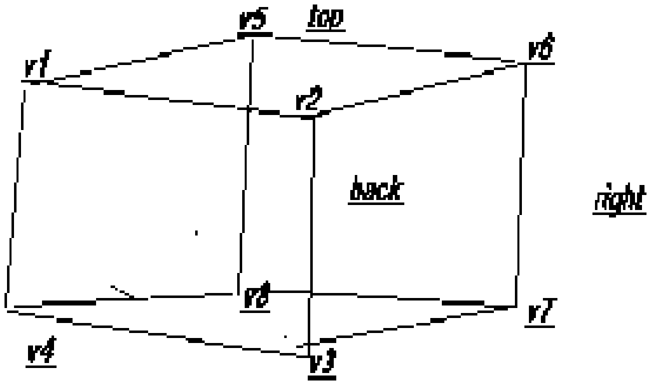


Note the large list of interesting "visual" words which were returned uopn execution of the with(plots); We will visit a number of them in this study. If you are interested in what a word means you can use the Maple "help" system in a number of ways. For the present the command help(polygonplot3d); illustrates the most obvious use of the help system.

```
help(polygonplot3d);
```

Although loading the plots package relieves us of a few keystrokes in writing down our work we will typically forego this for the far greater advantage of portability and readability of our instructions. We will continuously find that we want to go back to old worksheets and reuse pieces of code via "cut and paste" operations. Rather than having to carefully re-read such material to see if perhaps it depends on packages having been loaded earlier in the worksheet.

The ability of Maple to draw polygons in space provides us with the means to have Maple create images of any spatial object which consists of polygons. Lets begin with one of the most familiar objects, a cube. A cube has eight *vertices, and six faces, each defined by a sequence of the vertices. Since Maple can work with symbols we can represent the vertices symbolically, say with symbols v1, v2, .. and the faces (top, bottom, left, right, front, and back) as sequences of the vertices. First we sketch our cube, labeling its vertices and faces with these symbols and then "read off" the lists we need to communicate with Maple. It is very helpful to use a hand-drawn sketch or diagram. (You can draw these with for example "paint" and cut and paste the sketches into the worksheet.)*



Notice that all of this so far is hand drawings and text. We have not communicated anything to the Maple program which "understands" a limited vocabulary and has no ability at all to understand pictures. We enter Maple commands at the command prompt ">". We want to describe to Maple each of the faces of our box using terms that have meaning (the same meaning) to both us and the computer.

1.2.1 *Definition: A* **polygon** *is an ordered list of points lying in a plane in space.*

1.2.2 *Definition: An* **n-gon** *is a list of n different points. Two lists correspond to the same polygon if a circular permutation of one or yields the other or a reverse listing of the other.*

The connection between our usual geometric view of a polygon and this form is that the list which describes it may be thought of as the order in which the verticies are passed as one "walks" along the boundary of the polygon along a path which passes each vertex exactly once. The fact that there are numerous paths which so this corresponds to the fact that there are numerous lists which represent a particular polygon. Thus, for instance, the front face of our sketch is the list [v1, v2, v3, v4]. It is also [v2,v3,v4,v1] and [v3, v2,v1,v4].

1.2.3 *Problem: Exactly how many different lists of vertices corespond to any one n-gon?*

It is posible and convenient for the same object to have more than one name in Maple. Thus for instance, with reference to the above diagram we may want to refer to the ploygon whose vertices are v1,v2,v3,v4 as the "front face" or simply to give it the name "front". Similarly there is a "back", "top", etc, the latter being names for other polygons which are easy to refer to and remember. The way we tell Maple that we want to refer to the list [v1,v2,v3,v4] as "face" is via assigned equality.

```
front:=[v1,v2,v3,v4];
```

$$front := [[2, 0, 3], [2, 1, 3], [2, 1, 2], [2, 0, 2]]$$

```
front;
```

$$[[2, 0, 3], [2, 1, 3], [2, 1, 2], [2, 0, 2]]$$

What we have done is create an alternate name or label for this list. We might communicate this in English with a sentence like "Let "front" designate the list [v1,v2,v3,v4]."

The same object can have any number of names.

```
Front_face_of_the_cube:=front;
```

$$Front\_face\_of\_the\_cube := [[2, 0, 3], [2, 1, 3], [2, 1, 2], [2, 0, 2]]$$

Sometimes longer names such as the above are very useful in the development of more complicated sets of instructions.

We will repeatedly use this idea of assignment for labels: alternate convenient names for certain object which will have the same meaning to us and the computer program. We will, at the same time need to understand the difference between an equation (which uses the symbol "=") and an assigned equality which uses the symbol ":=". Consider the following:

```
joe:=1;
```

$$joe := 1$$

```
sam=10;
```

$$sam = 10$$

The first states that joe has the value 1 - that Joe is simply another name for the number "one". The second is an equation rather than an equality. It does not provide any alternative interpretation for the value of sam.

```
joe;
```

$$1$$

```
sam;
```

$$sam$$

For the moment we simply need to be aware of the fact that an equation is simply two expressions separated by a equal sign. By itself it does not affect the value of either. We can write an equation involving any two experssions.

```
1=2;
```

$$1 = 2$$

Maple will, however, balk at attempts to assign values. For instance the following commands would attempt to redefine the value of 1.

```
1:=joe;
```

```
Syntax error, `:=` unexpected
```

```
1:=2;
```

```
Syntax error, `:=` unexpected
```

We will have numberous opportunities to take advantage of the fact that Maple can solve equations

```
solve(2*x+1=0,{x});
```

$$\{x = \frac{-1}{2}\}$$

If you understand why it hasn't a clue as to what to do with the following then you probably understand the difference between "=" and ":="

```
solve(x+1:=x^2,{x});
```

```
Syntax error, `:=` unexpected
```

A command to Maple is always entered at the command
prompt (">") and terminated with a semicolon (";") or a colon ":".
If a command is terminated with a semicolon then Maple displays the result of its interpretation of the command; if the command is terminated with a colon the Maple output is not displayed.

```
1+2+3+4+5;
```

$$15$$

```
6+7+8+9+10:
```

Multiple commands can be entered at the same prompt:

```
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];
```

$$front := [[2, 0, 3], [2, 1, 3], [2, 1, 2], [2, 0, 2]]$$

$$back := [[1, 0, 3], [1, 1, 3], [1, 1, 2], [1, 0, 2]]$$

$$left := [[2, 0, 2], [1, 0, 2], [1, 0, 3], [2, 0, 3]]$$

$$right := [[2, 1, 2], [2, 1, 3], [1, 1, 3], [1, 1, 2]]$$

$$bottom := [[2, 1, 2], [2, 0, 2], [1, 0, 2], [1, 1, 2]]$$

$$top := [[2, 0, 3], [2, 1, 3], [1, 1, 3], [1, 0, 3]]$$

Lets get back to our box. We (people) have a concept of an abstract box and we have a sketch which helps us visualize it. We and the computer "know" that there certain symbols, v1...v8 which we (people) call the vertices but this designation has no meaning to the computer. We and the computer know about certain lists of the vertices and have agreed on names for them. We have agreed that certain other lists have the same meaning (when thought of as polygons) but the computer knows only about the specific lists we chose. We can "ask" Maple how it is interpreting a symbol by entering that symbol as a command.

```
top;
```

$$[[2, 0, 3], [2, 1, 3], [1, 1, 3], [1, 0, 3]]$$

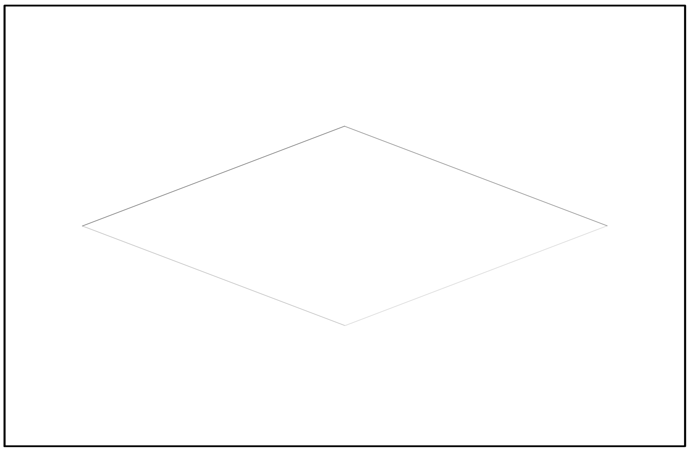
```
TOP;
```

$$TOP$$

Lower case "top" refers to the list of symbols it returned. "Top" is simply a string to which no other meaning has been assigned.

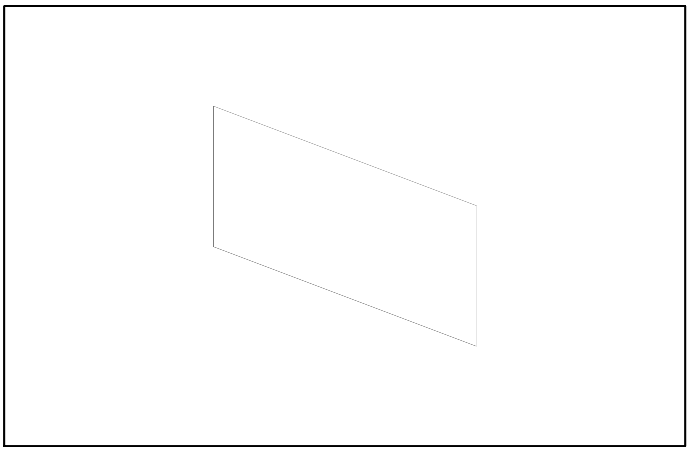
Although we can sketch and work with an abstract figure Maple can draw only draw specific figures which we specify specific completely. The polygonplot3d command works with specific, not general lists of vertices. For instance,

```
plots[polygonplot3d]([[0,0,0],[0,1,0],[1,1,0]
, [1,0,0]]);
```



produces a square but

```
plots[polygonplot3d]([v1,v2,v3,v4]);
```



doesn't know what to do with since the items in the list its given do not represent points in space, i.e. are not themselves lists of three numbers. The solution to this is to tell Maple how to interpret v1, v2, v3, v4 as points in space. This amounts to assigning these names to specific points (or more precisely to sets of coordinates for the points relative to some reference frame). This assignment is totally up to us. We might, for instance have decided that our abstract box is actually the "unit cube" in three dimensions relative to some coordinate system. In that case our sketch might have been.

Plot: ch1box1.eps

This can be interpreted as assigning values to our previously abstract quantities v1, v2, v3, v4, ... – something we now know how to do.

```
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];
```

$$v1 := [1, 0, 1]$$

$$v2 := [1, 1, 1]$$

$$v3 := [1, 1, 0]$$

$$v4 := [1, 0, 0]$$

$$v5 := [0, 0, 1]$$

$$v6 := [0, 1, 1]$$

$$v7 := [0, 1, 0]$$

$$v8 := [0, 0, 0]$$

The computer will assign these meanings whenever it encounters the symbols v1..v8. Since they occur in our previously defined names for the faces, those names now represent specific polygons in space.

```
top;
```

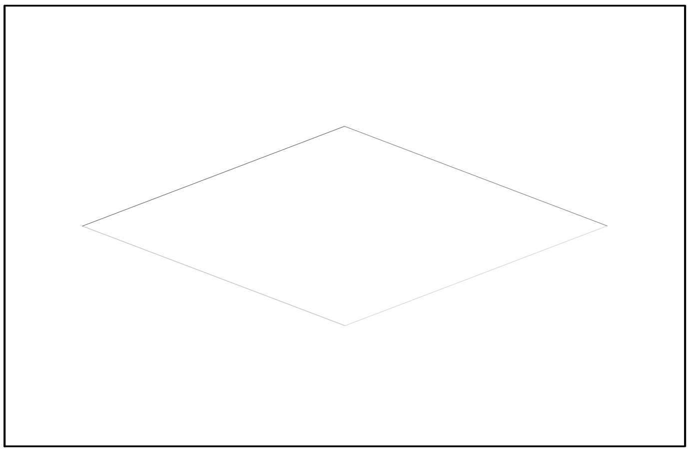
$$[[2, 0, 3], [2, 1, 3], [1, 1, 3], [1, 0, 3]]$$

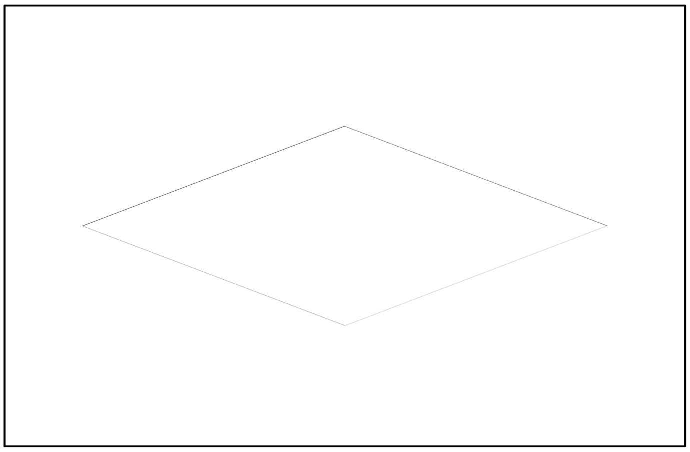
```
bottom;
```

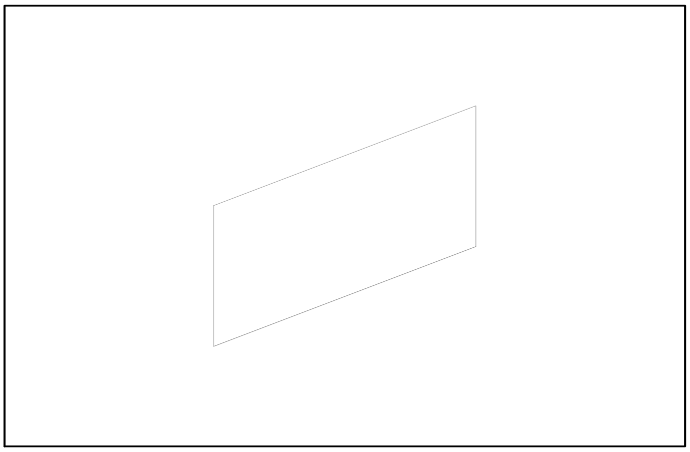
$$[[2, 1, 2], [2, 0, 2], [1, 0, 2], [1, 1, 2]]$$

Now we can ask Maple to render visualizations of these faces:

```
plots[polygonplot3d](top);
plots[polygonplot3d](bottom);plots[polygonplot3d](right);
```

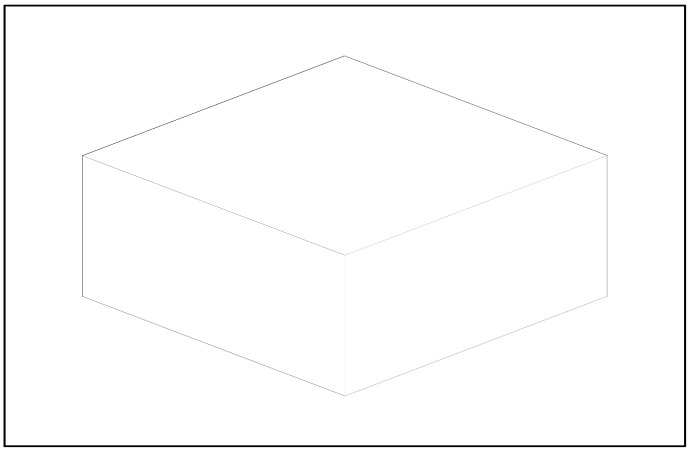






Of course this isn't satisfactory, we don't want to see the faces separately, we want them all together. To do this we simply give the command the complete list of faces which directs the program to render all of them in the same image:

```
plots[polygonplot3d]([top, bottom, left,
right, front, back]);
```



This is better but we can't identify front, back, etc. nor does the picture have the "wireframe" format of our sketch. This can be remedied in by asking for axes and altering the plotting style. Left clicking on the image encloses it in a box and creates a "tool bar" of manipulations of the image. The reader should explore the effect of each of the toolbar buttons and should note that the image can be rotated by draging the cursor across it. In addition there are "command line" options corresponding to each of the buttons on the tool bar. We will now explore several of them: style, scaling, and color which will be of immediate use. Each of these is an *option for the plots[polygonplot3d] command. The way to learn about a command and its options is through the help system. As noted earlier this can be invoked in a number of ways. When we know exactly what we are looking for the command line approach is the most efficient:*

```
?plots[polygonplot3d];
```

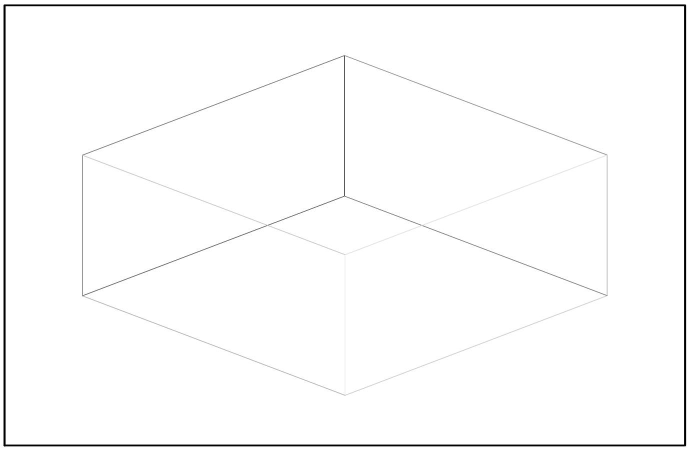
Note that the "help" page this takes us to describes the command and refers us to other help sheets for further information. We are interested in the options which we are told can be reached by the command ?plot3d[options]; or by clicking on that hyperlink at the bottom of the sheet. In either case we reach a page which lists a large number of options. Our main objective at this point is to know that the page exists and how to reach it. A few examples of the use of some of the options generally suffices to give us enough information to work out for ourselves how others are used. Note that the primary `plots[polygonplot3d]` help sheet contains some examples. These can be copied and pasted into our work session and executed to see what they do. For now we will apply some of these options to our example. First lets take advantage of our ability to assign short names:

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

$$box := [[[2, 0, 3], [2, 1, 3], [1, 1, 3], [1, 0, 3]], [[2, 1, 2], [2, 0, 2], [1, 0, 2], [1, 1, 2]], [[2, 1, 2], [2, 1, 3], [1, 1, 3], [1, 1, 2]], [[2, 0, 2], [1, 0, 2], [1, 0, 3], [2, 0, 3]], [[2, 0, 3], [2, 1, 3], [2, 1, 2], [2, 0, 2]], [[1, 0, 3], [1, 1, 3], [1, 1, 2], [1, 0, 2]]]$$

Now we can simply have the computer draw "box" - but we want to do this with some of the options. The first changes the plotting style from the default "hidden line" to the "wireframe" format indicated in our rough sketch.

```
plots[polygonplot3d](box,
style=wireframe);
```



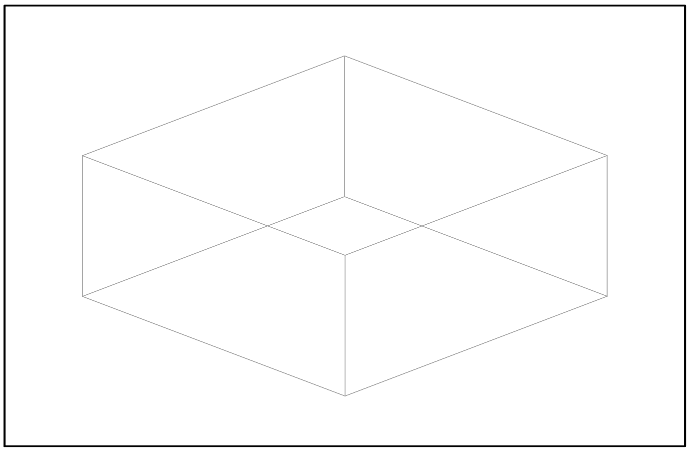
We can color the edges of the frame as we wish. The response to ?plot3d[options];refers us to plot[color]

```
?plot3d[options]
```

```
?plot[color]
```

Which gives us the following list : aquamarine, black , blue, navy, coral, cyan, brown, gold, green, gray, grey, khaki, magenta, maroon, orange, pink, plum, red, sienna, tan, turquoise, violet, wheat, white, yellow - as well as instructions for making our own colors. For now we will stuck with the "off-the-shelf" colors.

```
plots[polygonplot3d](box,
style=wireframe,color = tan);
```



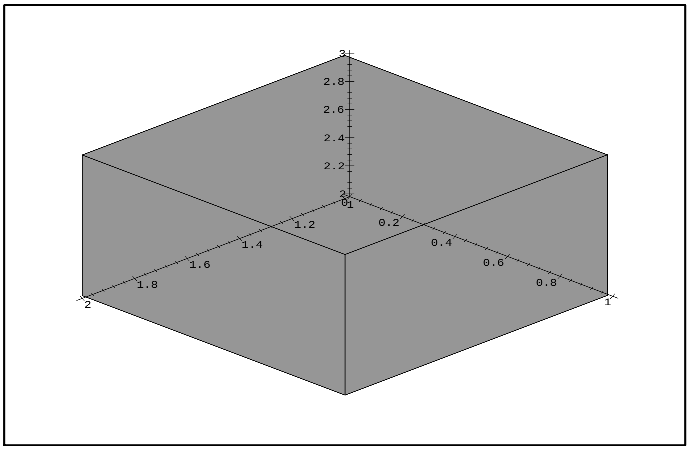
The "patch" style allows us to specify a color for the entire box.

```
?plot3d[options]

plots[polygonplot3d](box, style=patch,color =
tan);
```

The axes option allows us to include coordinate axes in different ways:

```
plots[polygonplot3d](box, style=patch,color =
green, axes= normal);
```

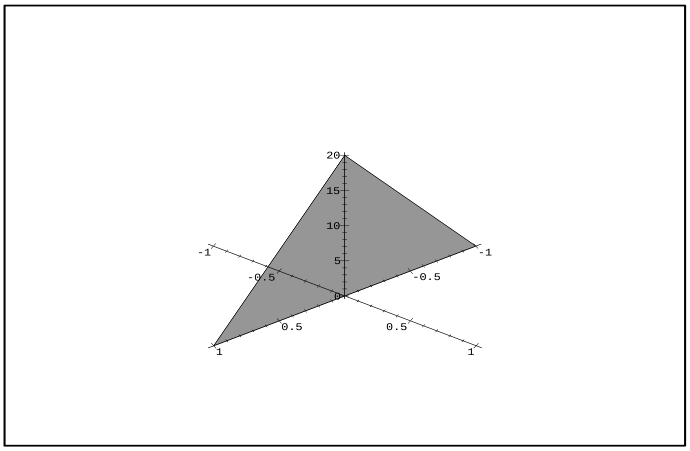


When creating a general plot Maple attempts to do the best job it can of placing the entire picture in the space available. In doing so it often scales the coordinate axes so that the image produced does not appear "to scale". The option which turns this off or on is "scaling" which is "constrained" (i.e. to scale) or "unconstrained". The symmetry of our box makes it a poor choice to illustrate this but we can readily define other objects such as the following

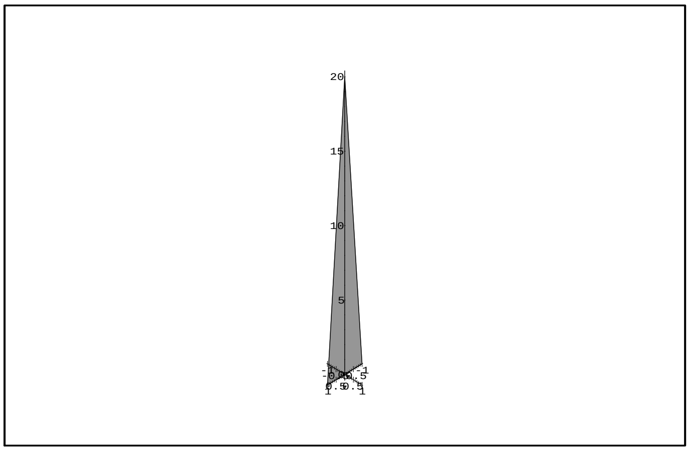
```
skinnytriangle:=[ [-1,0,0],
[1,0,0],[0,0,20]];
```

$$skinnytriangle := [[-1, 0, 0], [1, 0, 0], [0, 0, 20]]$$

```
plots[polygonplot3d](skinnytriangle,
style=patch,color = green, axes= normal);
```




```
plots[polygonplot3d](skinnytriangle,
style=patch,color = green, axes= normal, scaling=constrained);
```



An obvious disadvantage of the coloring option as used on our box to this point is that we can specify only one color in a command and therefore our box can be only one color. The solution to this is to "draw" each of the faces separately with whatever color we want and plot them together. To do this we need another command, plots[display] which displays a list of individual plots in the same picture, relative to the same coordinates. Suppose we want our box to have a blue top, yellow sides, and a purple bottom. We can separately construct the pieces with different colors:

```
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);
```

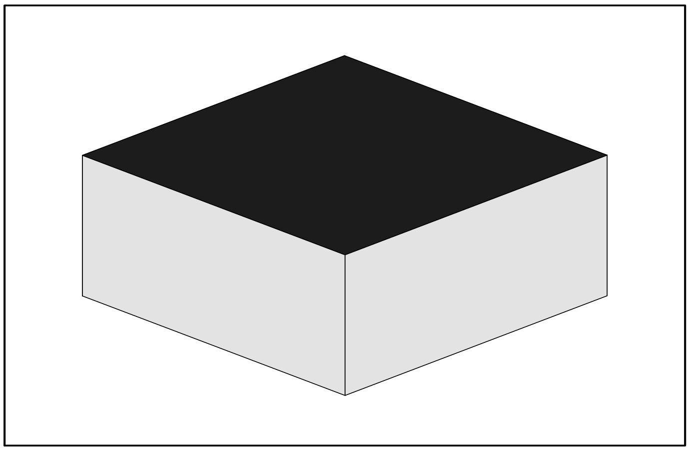
bluetop := **PLOT3D**(**POLYGONS**([[2., 0, 3.], [2., 1., 3.], [1., 1., 3.], [1., 0, 3.]]),
STYLE(***PATCH***), **COLOUR**(***RGB***, 0, 0, 1.00000000))

purplebottom :=
PLOT3D(**POLYGONS**([[2., 1., 2.], [2., 0, 2.], [1., 0, 2.], [1., 1., 2.]]), **STYLE**(***PATCH***))

yellowsides := **PLOT3D**(**POLYGONS**([[2., 0, 2.], [1., 0, 2.], [1., 0, 3.], [2., 0, 3.]],
[[2., 1., 2.], [2., 1., 3.], [1., 1., 3.], [1., 1., 2.]], [[2., 0, 3.], [2., 1., 3.], [2., 1., 2.], [2., 0, 2.]],
[[1., 0, 3.], [1., 1., 3.], [1., 1., 2.], [1., 0, 2.]]), **STYLE**(***PATCH***),
COLOUR(***RGB***, 1.00000000, 1.00000000, 0))

We have not told Maple to draw these items. What we have done is assigned names to the "plot structures", the sets of information needed by Maple to display the images. The command plots[display] accepts a list of such images and displays them simultaneously:

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

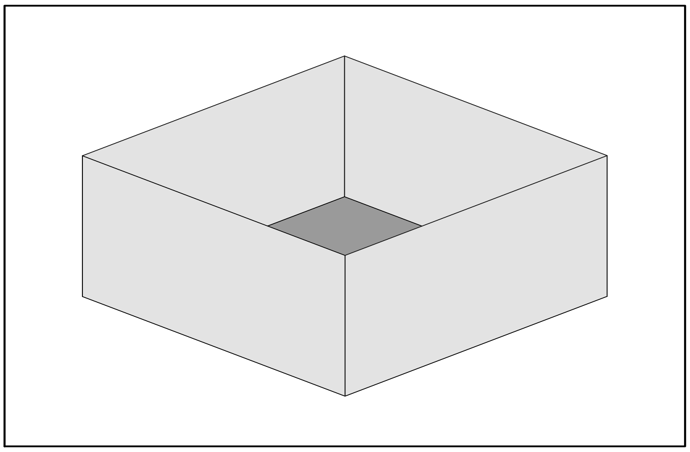


If we would like to be able to "see through" the top of our box we could draw the top in the "wireframe" style.

```
bluetop:=plots[polygonplot3d](top,
style=wireframe,color = blue );
```

bluetop := **PLOT3D**(**POLYGONS**([[2., 0, 3.], [2., 1., 3.], [1., 1., 3.], [1., 0, 3.]]),
STYLE(***LINE***), **COLOUR**(***RGB***, 0, 0, 1.00000000))

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



Note that the second assignment of a value to "bluetop" redefines "bluetop" to be a wireframe style plot structure. It does not change the previous image (which still has its solid blue top) since that is an image computed from the previous meaning. However if we now go back and re-evaluate the command which produced that image the computation will be done with the current meaning of "bluetop" and the resulting picture will be identical to the one immediately above.

1.2.4 Exercise (drawing colored cubes): Have Maple draw a cube on which each face has a different color. Have it draw a cube on which each pair of opposite faces have the same color but no two faces which share an edge have the same color.

We have seen that we can make new boxes from old ones by "repainting" the faces of a single box or making some of the faces transparent. These "new" boxes will, however, occupy exactly the same position in space as the original. Suppose we want to "stack" a copy of the original box on top of it, or create a duplicate some distance away. This is extremely easy to do because of the general nature of our approach. Our original sketch could have any box at all - it didn't become a specific box until we assigned values to the vertices v1..v8. Thus if we simply go back and assign different values to the v1..v8 and execute all of the commands which followed we will duplicate everthing we have done for a new box which can then be displayed along side the old one. However at the outset of an investigation it is usually not a good idea to go back and change commands which we have working the way we want. It is MUCH better to copy the working commands that we want into the same or another worksheet and modify the copies. That way when something doesn't work we have a definite point at which to start "debugging". We will use the option of copying the commands we want into a new worksheet within the same Maple session.

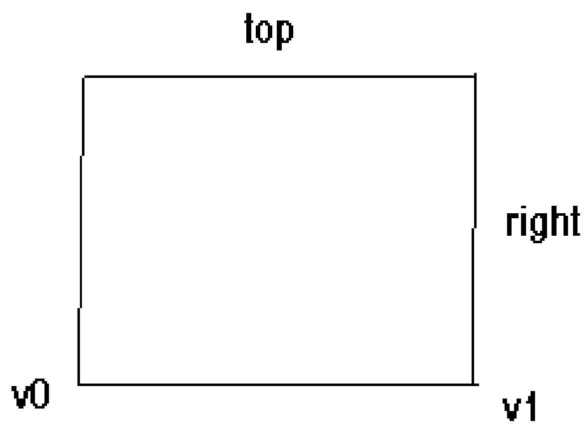
1.3 Introductory Exercises: Drawing in the Plane

The following exercise set is a brief, "get started" introduction to Maple. Follow the instructions in order.

1.3.1 *1. Open a new Maple worksheet by clicking on "new" in the "file menu". Practice "toggeling" between this and the new worksheet by selecting the appropriate worksheet in the "Window" menu. Save the new worksheet using the "Save As" option at the "File" menu with an appropriate name such as "EX_CH1".*

*

1.3.2 *2. "Select" this (entire) exercise set with the mouse and copy and paste it into the new worksheet.*



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

PASTE THE DIAGRAM IN THE SPACE ABOVE

*

1.3.4 *4. Working in the new worksheet, break this exercise into distinct paragraphs (execution groups) by placing the cursor at each asterisk to the left and pressing "F3". This can also be done at at the "Edit" menu at "split or join" execution group.*

*

1.3.5 *5. Delete each of the asterisks.*

*

1.3.6 *6. Open two command lines (execution groups) immediately below the each of your newly created cells by use of the CONTROL J or at the "Insert" Menu or by use of the large ">" (greater than sign) on the toolbar. That is, place the cursor in the cell and then press CONTROL J or use the menu or toolbar (twice). The result should be that each of the cells you created is followed by two new cells (execution groups) containing only a ">" prompt .*

*

1.3.7 *7. Follow each of the following instructions by entering the requested command on the first of the command lines you opened. Commands for "arithmetic": Addition, multiplication, division, exponents, parenthesis, etc. follow the familiar syntax used with most calculators. For instance one would enter 3*(5+3^12)^(2^(11)); in order to calculate* $3\,(5+3^{12})^{(2^{11})}$.

1.3.8 *8. Add the numbers 1,3,17,-21,35,12*

*

1.3.9 *9. Multiply the numbers 1,3,17,-21,35,12*

* The "ifactor" command allows you to factor integers, e.g ifactor(24); should return the prime factorization of 24.

*

1.3.10 *10. factor your social security number; factor your birth year.*

*

1.3.11 *11. Calculate* $2^{(2^j)}+1$ *and factor* $2^{(2^j)}+1$ *for j = 0,1,2,3,4,5,6,7. Do so by entering both commands on the same command line below (each with its own semicolon) and subsequently by modifying the commands, changing the "j" to the next integer. (Until the turn of the century it was thought that all such numbers were prime.)*

*

Maple will add, subtract, and multiply polynomial expressions. Typically a line like (x+y)*(x-y); will return the "unexpanded" version. The "expand" command tells the computer to actually multiply out a polynomial expression in any number of variables, e.g. expand(x*(a+b)); and expand((a+b)^2); will produce just what one would expect. One can do things like assign a name to an expression like joe:=expand((a+b)^2); . The command "factor" does for polynomials what "ifactor" does for integers. Thus following the previous command one could factor(joe);

1.3.12 *12. Have the computer do the polynomial multiplication (1-3*x+5*x^3)*(23-19*x+12*x^2);.*

*

1.3.13 *13. Have the computer "multiply out"* $(a+b)^{99}$*. **

1.3.14 *14. Tell the computer to assign the value "kate" to the result of expanding* $(2\,x+3)^{50}$ *out "completely".*

*

1.3.15 *15. Have the comuter to factor "kate".*

*

1.3.16 *16. Release 4 of Maple will "add" lists (earlier versions will not) and multiply a number times a list 2*[a,4]; returns [2a,8]) . If you have Release 4 or higher , tell Maple to add the lists [1,2,3], [4,5,6], and 7 times [a,b,c].*

*

Maple knows all of the functions found on a graphics calculator and many more. The command plot(sin(3*x),x=-2*Pi..2*Pi); will plot sin(3*x); from $-2\,\pi$ to $2\,\pi$. The command plot({sin(3*x),sin(2*x), sin(x)} ,x=-2*Pi..2*Pi); will calculate the three functions sin(3x), sin(2x) and sin(x) at the same time.

*

1.3.17 *16. Have Maple plot cos(x) from -5 to 3.*

*

1.3.18 *17. Have Maple plot 1, x, 2*x, and 3*x^2 from -2 to 2 at the same time.*

*

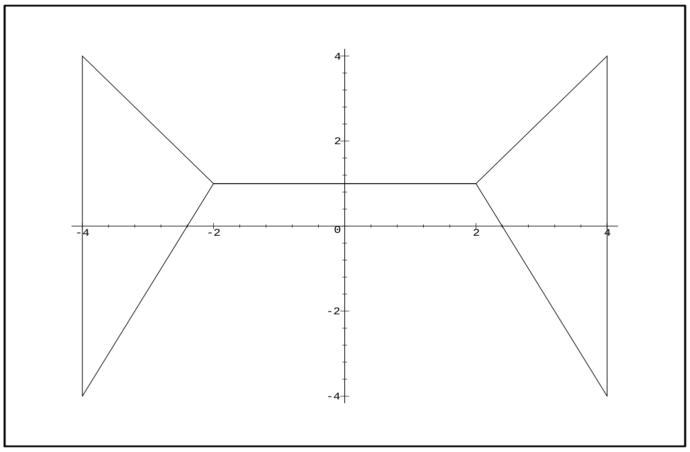
The plots[polygonplot] command will plot lines and polygons in the plane. The command plots[polygonplot]([[1,2],[3,7]]); will draw the line segment connecting the point [1,2] to the point [3,7] while

plots[polygonplot]([[1,2],[3,-4], [4,0]]); will draw a triangle with the specified vertices.

plots[polygonplot]({ [[1,2],[3,7]], [[1,2],[3,-4], [4,0]]}); will draw a line segment and a triangle.

*

1.3.19 *18. Have Maple draw the following figure:*
1.3.20



*

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:

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=3.

Multiple options can be invoked in the same plot. For instance, plots[polygonplot]([[1,2],[3,-4], [4,0]],color=red, style=patch,axes=none, scaling=constrained);

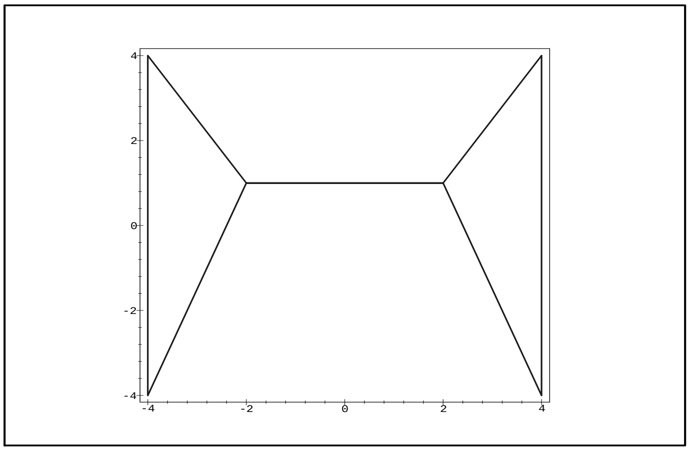
will draw the polynomial "to scale", with no axes, and will "color it in" in red. Specifying a color will automatically set the style to "patch" so if one wants just the edges of a polygon colored one invokes the LINE style.

*

1.3.21 *19. Have Maple draw a "solid" blue triangle with vertices at [0,0], [3,0], and [0,10] - do not specify any scaling (this automatically invokes the default which is "unconstrained"). Draw the same triangle "to scale" with "framed" axes. Draw the same triangle with no axes, do not fill it in, unconstrained, with the edges as "thick" (thickness=3) red lines.*

*

1.3.22 *20. Have Maple draw this picture.*



*

The plot options can also be used with the "plot" command (although in some cases they may have no effect). For instance the following will yield the same results: plot(sin(x), x=-Pi..Pi); and plot(sin(x), x=-Pi..Pi, style = patch); . On the other hand other options will change the graphs.

*

1.3.23 *21. Execute the following, changing the varions options among those listed above and observe the differences:*

1.3.24 *plot({sin(x),cos(x)}, x=-Pi..Pi, style = POINT, scaling=constrained, axes=framed, thickness=2, color=blue); **

1.4 More Introductory Exercises and Drawing in Space

1.4.1 *1. Open a new Maple worksheet by clicking on "new" in the "file menu". Practice "toggeling" between this and the new worksheet by selecting the appropriate worksheet in the "Window" menu.*

*

1.4.2 *2. "Paint" this exercise with the mouse and copy and paste it into the new worksheet.*

*

1.4.3 *3. Use the "paint" program to draw a (square-based) pyramid, label each of its vertices with the symbols v1,..v5, and paste it into your new Maple worksheet in the space below.*

PASTE THE DIAGRAM IN THIS SPACE
*

1.4.4 *4. Working in the new worksheet, break this exercise into distinct paragraphs (execution groups) by placing the cursor at each asterisk to the left and pressing "F3". This can also be done at at the "Edit" menu at "split or join" execution group.*

*

1.4.5 *5. Delete each of the asterisks.*

*

Open two command lines (execution groups) immediately below the each of your newly created cells by use of the CONTROL J or at the "Insert" Menu or by use of the large ">" (greater than sign) on the toolbar. That is, place the cursor in the cell and then press CONTROL J or use the menu or toolbar (twice). The result should be that each of the cells you created is followed by two new cells (execution groups) containing only a ">" prompt .

Follow each of the following instructions by entering the requested command on the first of the command lines you opened.

Maple does arithmetic with integers (whole numbers: no decimal point) or fractions whose numerator and deniminator are integers with no round off. If a single number in such a calculation has a decimal (i.e. is a floating point number) then Maple carries the number of decimal digits specificd by the variable Digits . The default value of digits is 10. The user can change the value of Digits by assigning a (positive integer) value to it.

1.4.6 *6. Have Maple calculate 2^100 and 100!*

*

1.4.7 *7. The Maple command "sqrt" can be used to calculate square roots, To calculate the square root of 19 use sqrt(19); Note the difference between sqrt(19); and sqrt(19.0);*

Have Maple display the first thousand digits of the square root of 2. Start with the command Digits:=1000; When done be sure to reset the digits to 10.

*

1.4.8 *8. Sometimes Maple's accuracy obscures what we really want to know. For instance, is the fraction $\frac{13^{99}}{123^{97}}$ greater than 1 or less than 1? Enter joe:=13^99/123^97; and see if you can tell. In such cases it is convenient to convert such experssions to floating point (decimal) approximations from which we can readily answer such questions. The Maple command evalf (evaluate floating point) does this. Try evalf(joe);*

1.4.9 *9. Maple stores 10000 digits of Pi but if you ask it to return Pi by entering the command Pi; you will just get a symbol. Use "Digits" and "evalf" to have Maple print out 5000 digits of Pi. Note that Pi is what Maple thinks is the ration of the circumference of a circle to its diameter; pi is simply another symbol with no independent interpretation. Note the difference between evalf(Pi); and evalf(pi);*

*

1.4.10 *10. We have seen that Maple can plot functions of one variable with plot . It can plot functions of two variables with plot3d . To plot "sin(x^2-y^2)" for x between -2 and 3 and y between -1 and 2.5 use the command plot3d(sin(x^2-y^2),x=-1..3,y=-1..2.5);*

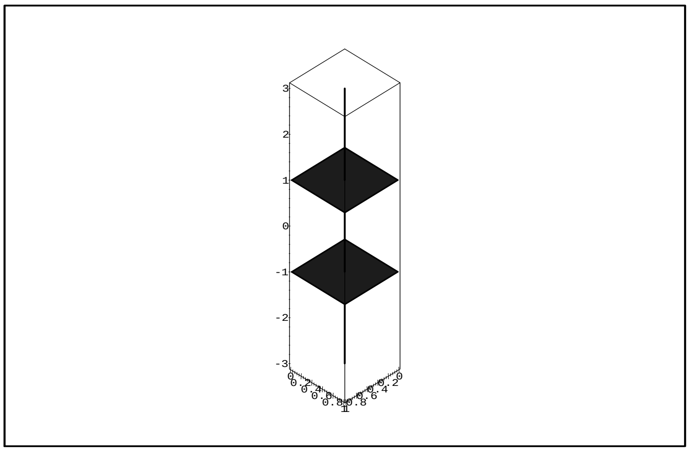
*

The plots package contains many interesting three dimensional plotting tools. Among the most useful is plots[polygonplot3d], the analog of plots[polygonplot] in two dimensions. One must keep in mind that plots[polygonplot] expects lists of points in the plane while plots[polygonplot3d] expects lists of points in space. plots[polygonplot3d] will plot lines and polygons in the plane. The command plots[polygonplot]([[1,2,0],[3,7,0]]); will draw the line segment connecting the point [1,2,0] to the point [3,7,0] while

plots[polygonplot]([[1,2,0], [3,-4,0], [4,0,0]]); will draw a triangle with the specified vertices.

plots[polygonplot]({ [[1,2,0],[3,7,0]], [[1,2,0],[3,-4,0], [4,0,0]]}); will draw a line segment and a triangle.

```
plots[polygonplot3d]({[[0,0,1],[1,0,1],[1,1,
1],[0,1,1]],
[[0,0,-1],[1,0,-1],[1,0,-1],[1,1,-1],[0,1,-1]],[[1/2,1/2,3],[1/2,1/2,-
3]]},
thickness=3,color=blue, scaling=constrained, style=patch,
axes=boxed);
```



1.4.11 *11. Make changes in the options invoked and observe the changes. Note that you can rotate three dimensional plots by "clicking" on them with the mouse, holding the mouse button down and "dragging" the reference box (which represents the figure) around. Double clicking on the figure or selecting "R" for (redraw) at the tool bar will draw the rotated figure.*

1.4.12 Exercise (drawing colored cubes): Have Maple draw a cube on which each face has a different color. Have it draw a cube on which each pair of opposite faces have the same color but no two faces which share an edge have the same color.