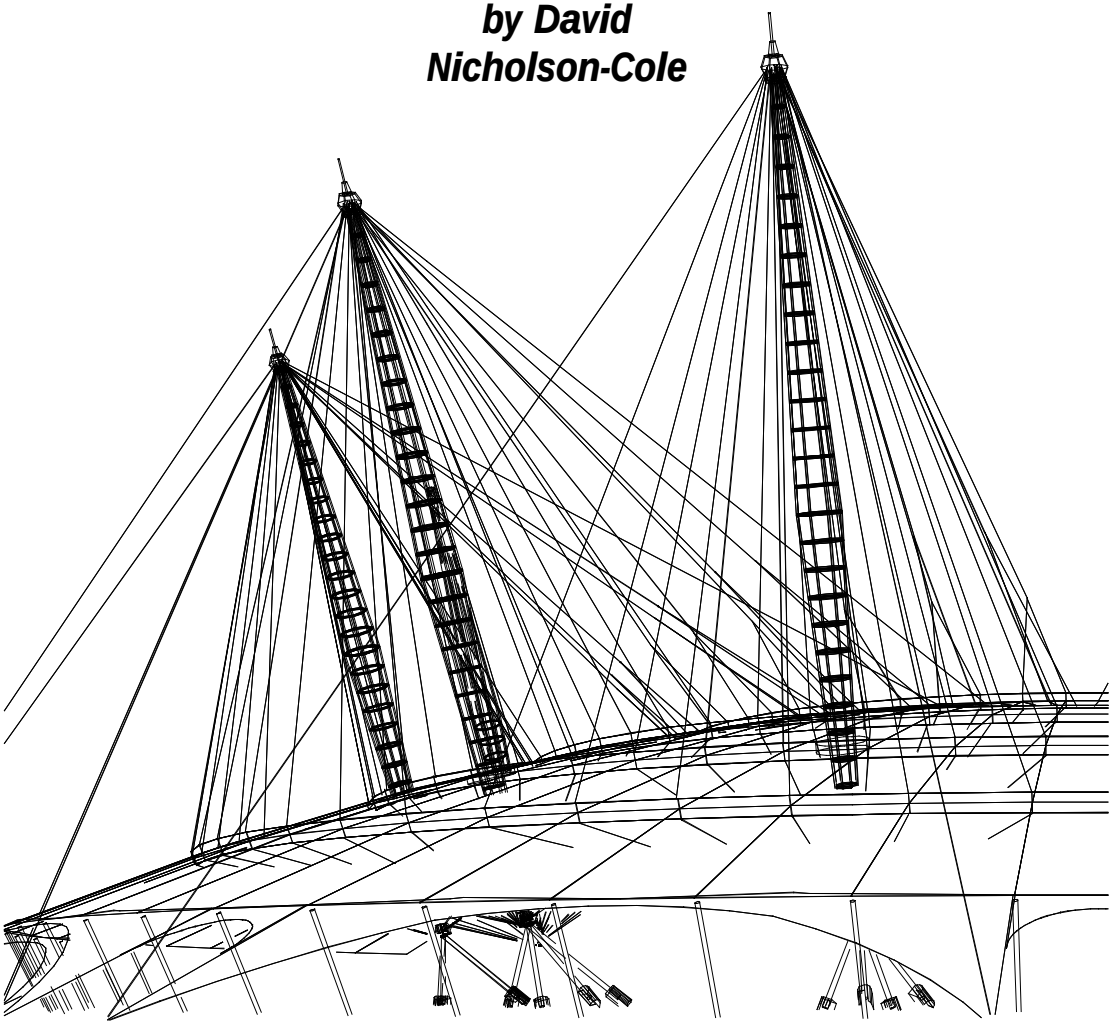
# GDL—Taster

## Introduce yourself to GDL

*by David Nicholson-Cole*



# GDL—Taster

## Teach yourself GDL by the fast track!

THE GDL Cookbook is a collection of models which have been produced to help teach GDL. Each model is accompanied by explanatory text and images. The models are graded and organised to give a 'Cookbook' style of progressive learning.

GDL-TakeAway will be a fast track route to GDL based on the Author's Lift Off courses. Early courses, with more time available for personal tuition, were based on the original Cookbook. But the Cookbook grew bigger and bigger and became the major work of reference and self teaching that it is now. The GDL–TakeAway is not yet published, but the author of the Cookbook is working on the Takeaway project.

THIS little GDL-Taster Document is a short taster giving you some of the material in the introductory pages of the GDL-Takeaway and the GDL Cookbook, and you will save time if you read this prior to coming on a GDL Lift Off course with David Nicholson-Cole, or anybody teaching GDL in your area.


The GDL–Cookbook is published by Marmalade Graphics, Nottingham

Marmalade, 15 Elmtree Ave, West Bridgford,
Nottingham NG2 7JU, England
Tel +44-115-945-5077 : Fax +44-115-945-5121: email:
davidnc@innotts.co.uk

## The Cooking Metaphor

THE ORIGINAL GDL manual is not well explained, and has too few examples. The 'Cookbook' approach to learning is designed to change what seems like a daunting task into something enjoyable and useful.

The cooking metaphor suggests that you will learn some GDL instantly. If you plough quite forcefully through the GDL Cookbook and the GDL Manual, you will learn it rapidly. However, you also need some real projects to test your skills and force you to try advanced techniques before you acquire proficiency to the level of a GDL Voyager.

The design environment is constantly changing – no sooner had the GDL Cookbook settled down to a level of completion in 1999 than I was having to adjust to the very welcome arrival of ArchiCAD 6.5 in January 2000.

## Smart, Parametric Models

YOU are truly privileged. You are the only members of the CAD community who have the ability to built PARAMETRIC objects without needing a degree in Computer Science. Quite simply, 'Parametric' means that you can change the parameters of an object. With GDL, it also means that the object can be 'smart' – change its own parameters according to rules or calculations. Thus it can display a level of 'artificial intelligence'. This is the power of GDL!

In other CAD software, if you stretch an object, it will stretch, but the elements of the object will be distorted proportionally. Whereas, with a parametrically built window, handrail, stair or structural truss tool, it can be written to recalculate the spacing of members, to resize members if necessary, correct user's errors, offer the user intelligent choices through pop-down menus in the object, and change the way it looks at different scales or distances. All these techniques are covered in GDL TakeAway and in the GDL Cookbook.

GDL is poised to go "Cosmic" in the year 2000 when Graphisoft make it available to other CAD environments. This is a useful skill to teach yourself.

# Reasons why you need not learn GDL

- You can already make library objects using the Wall, Slab and Roof tools.
- You are contented with the existing ArchiCAD library.
- You can get CDs of more objects.
- You can pay a bureau to make 3-D objects when the need arises.
- A lot of the work you do is 2-D and doesn't need 3D modelling.
- It looks difficult – the GDL manual makes you shudder, just looking at it.
- You can't remember trigonometry or circle geometry, and cannot program.
- None of the objects you make need to be 'smart' or elegant, or get repeated.

# Reasons why you should learn GDL

- You can make insanely great things that, for interest and complexity, go far beyond library objects made with wall, roof and slab tools.
- You can make 'tools' – such as a joist tool, louvre tool or a handrail tool – that increase your productivity.
- You can make parametric models – i.e. an object that offers you a dialog box with the means of changing its height, diameter, colour, frequency, style etc.
- It is very economical in disk space – GDL models only occupy the disk size of a script plus a small header and footer.
- You speed up rendering time; objects can be written with clean coding, unlike the dirty code produced by DXF or library objects made with wall, roof and slab, which needlessly calculate forms to a millionth of a metre accuracy.
- You can have really cool features like modestly intelligent objects which complain if you enter wrong parameters; offer you options in the language of your choice; offer you popdown menus to assist you with choosing parameters; which dynamically turn to face the camera; or change colour or shape as their position in the model, storey, or camera location changes. They can even consider whether to bother drawing themselves or not.
- You can do a useful cleaning up job on objects created with wall, roof and slab tools, if you know a bit of GDL.
- You can increase the level of detail in a model by being firmly in control of the number of polygons.
- It's fun, and easy once you get the hang. It is enjoyable to use simple mathematics and programming skills to see 3D forms popping out of the screen in front of your eyes!
- You can discover a new source of income, writing GDL objects for other users: this skill could be career changing if you get addicted.

Clearly, the reasons to use GDL far outnumber the reasons not to(:-) This GDL-TakeAway makes it easy to progress through the learning process, and to enjoy the benefit of GDL.

Geometric Description Language

# Introducing GDL

IN THE EARLY DAYS of ArchiCAD (before Mac or PC) the prototype of ArchiCAD was on the Apple 2. Not having a mouse or windows, ArchiCAD *was* GDL – a scripting language to produce 3D form. The arrival of the Macintosh enabled programmers to put the building plan into one window, the building tools into another (to form a 'pallette'), coordinates in another, and so on. ArchiCAD as we know it now became visible. Designers could now drop walls and other building elements into a plan window, and stretch them to fit and change their properties; view the 3-D results in another window. GDL survived as a speciality, evolving in its own way.

| | |
|---|---|
| **GDL commands**<br>Produce 3D entities – like BLOCK, SPHERE, CONE, TUBE. | **3D organisation**<br>Look at the object analytically and understand what 3D shapes it contains. |
| **Programming**<br>The organisation of the script in a structured way, correctly sequenced. | **Interfacing with A'CAD**<br>Enable the object to respond to conditions in the main project. |

Working with GDL consists of four main areas of knowledge:

## GDL commands

GDL commands in their raw state are available to you in the GDL manual. It is difficult to understand their use and syntax until you have to apply them for real. Therefore, the cookbook approach in this book gives you an easier way of learning them by doing small projects, a bit at a time. The Cookbook method – being based on small projects growing in complexity – is analogous to the process of architectural education.

## 3D Organisation

SCRIPTING is easier if you understand the 3-D nature of the object. You must analyse the object accurately, regardless of the language you are using or thinking in. For architects and designers who are in the business of 3-D, this should be the easiest part.

You may not be able to build the object in every detail, but you should look at it, and decide what parts of it need to be included. This is called the 'level of detail' (LOD). In an urban model, buildings might be represented as simple slabs. At city block level, you would show roofs, but not windows. At single building level, you might show window openings, but not gutters or sills. At roof detail level, you would decide whether to show individual tiles and battens.

The universal language of 3D is the 3D primitive – the Block, Cylinder, Cone, Sphere, Extrusion etc. Almost everything can be made with combinations of these. At the next level of 3D interpretation, you have slabs which curve, holes drilled, surfaces rounded or chamfered, extrusions taken through curved or angular pathways, saddle shapes, lathed objects, elements repeated around axes, and many such variations.

You also decide how the object behaves if is to be stretched or hinged. You must identify elements which might change, like lights which come on, parts that must be able to slide or rotate, or to disappear if they are too detailed, or which might be generated by random numbers.

Think these out on paper. If you cannot draw the object freehand in pencil, then you probably cannot script it.

## Programming

A PROGRAM is just a sequence of instructions. You can read instructions to someone over the telephone, you can struggle through the instructions for setting the controls on your video, you can explain a cookery recipe to a friend, or plan the most efficient way of mowing the lawn. There is even a program in making a cup of tea! In all these cases you use a program, even if you don't think of it as a program. Suppose you write down the sequence of tasks, you would be sure to get them into the right order, to avoid doing tasks you had just done, or doing tasks that would undo ones you had done. You can write the sequence as a series of pictures, or as written commands. That's all there is to it!

Which language would you use? For cookery, you would speak in English, or German or Spanish, but you would use similar food ingredients. For GDL, you use the language of BASIC, and your ingredients are commands like BLOCK and CONE and TUBE. BASIC is the easiest of the major programming languages, devised in the seventies, and predominant on micros during the eighties. BASIC is easy to learn. Graphisoft must be thanked for choosing BASIC as the model for their programming language. We can all write a few lines of code, and see 3-D objects springing up into existence. It is fun! One's capabilities are vastly increased – in the 'old days' of programming, BASIC could build menu driven programs for dull things like accounts and calculations, many of which were far easier to do in spreadsheets. Now, towers, chairs, bridges, trains, space structures, buildings and people can all grow from your keyboard.

## Interfacing

GDL objects can behave slightly intelligently in that they can know current wall thicknesses, storey, scale, frame number, location, current pen and so forth, and behave accordingly.


# So, you think you can't program...

WELL YOU CAN! – squirrels can program, birds can program, spiders do it, small children do it – in that they plan a series of actions, they try to get them in the right order, they correct their own mistakes when they make them, and they achieve a result – perhaps a hoard of nuts, a nest full of eggs, a web, or a Lego model.

The British Standards Institute recently (Sept '99) received the 'Ig Nobel' Prize for Literature in that they wrote BS.6008: *Method for Preparation of a Liquor of Tea*, and took 5000 words to do it, going into the minutest detail! We will not go quite so far, but let's use tea making as an example of human programming. Let's write down a series of 'Operations' involved in making Tea. Without going into too much detail, list the sequence of operations to someone who knows the obvious things like how to turn on taps. Anybody reading this (above the age of 7) should be able to recite the following list in the right order.

```
1. Fill Kettle with water
2. Boil water
3. Get teapot
4. Fill teapot with bags
5. Fill with water
6. Get cups
7. Wait until ready
8. Pour out
9. Add Milk
10.Serve Tea
```

1

```
100: !Fill Kettle with water
200: !Boil water
300: !Get teapot
400: !Fill teapot with bags
500: !Fill with water
600: !Get cups
700: !Wait until ready
800: !Pour out
900: !Add Milk
1000:!Serve Tea
```

2

Now because we are going to teach a computer how to make tea, you need to rewrite the list giving each Operation a number(with a colon), and use an exclamation mark to make each Operation name into a Label. For a human, you might number each task 1, 2, 3, etc, but for computers its easier to use larger numbers (because later, you could insert extra things you hadn't thought of.)

Next, you look at each Operation and realise that none of them are single Actions. Each Operation consists of a number of Actions which might include Error Checking. For example, if the teapot contains last night's teabags, then Wash Teapot. This might require quite a lot of IF statements, and then the Actions taken as a result of the IF statements. Depending on the level of knowledge or stupidity of the machine, you instruct it appropriately. For example we assume here that the machine knows how to turn a tap or a kettle on. In the same way, GDL knows that Cylinders are round and that Cones are tapered *and* round without us having to define 'roundness' mathematically.

## The final program for making tea

3

```
100:!Fill Kettle with water
IF kettle (empty) THEN (Fill kettle with water)
IF kettle (filled) THEN GOTO 200:!Boil water

200:!Boil water
IF water (boiling) THEN (turn kettle off) ELSE (boil water)

300:!Get teapot
IF (number of people) less than 2.01 GET smallteapot
IF (number of people) greater than 2 GET largeteapot
IF teapot (dirty) THEN (wash out teapot) UNTIL (clean)
IF teapot (clean) THEN (swirl hot water) UNTIL (warm)
IF teapot (warm)  THEN (Fill teapot with bags)  !continue

400:!Fill teapot with bags
LET teatype=USERCHOOSE(EarlGrey,Lapsang,Camomile,Tetleys)
IF teapot (large) THEN add 2 teabag USING teatype
IF teapot (small) THEN add 1 teabag USING teatype

500:!Fill with water
POUR water INTO teapot UNTIL (full)
IF (not enough water) THEN GOTO 100:
PLACE teacosy UPON teapot
START timeclock USING seconds

600:!Get cups
REPEAT (in cupboard)
 GET (a cup)
 IF (cup dirty) THEN (wash cup)
 PLACE cup ONTO teatray
  UNTIL (enough cups for everybody)

700:!Wait until ready
IF TIMECLOCK less than 120 seconds THEN WAIT

800:!Pour out
REPEAT (on the teatray)
 IF fullness=0 THEN POUR (a cup) UNTIL fullness=7/8
   UNTIL (enough cups for everybody)

900:!Add Milk
LET milkcondition=USERCHOOSE(with milk,without milk)
  REPEAT (for each cup on the teatray)
   IF milkcondition=(with milk) THEN (add milk to cup)
     UNTIL (each cup tested)

1000:!Serve Tea
```

Goes round until kettle filled: 2 IF statements

A single line way of writing two IF statements

People are 'integer' quantities; but this makes doubly sure that you can make tea for 2.

For the 'parameter' of tea, you could let the user select from a list of choices.

In the event of a major problem you might have to jump back to the start.

This is a repeating 'Loop', do it until all cups are selected and clean and in place.

This was going on while you were getting the cups.

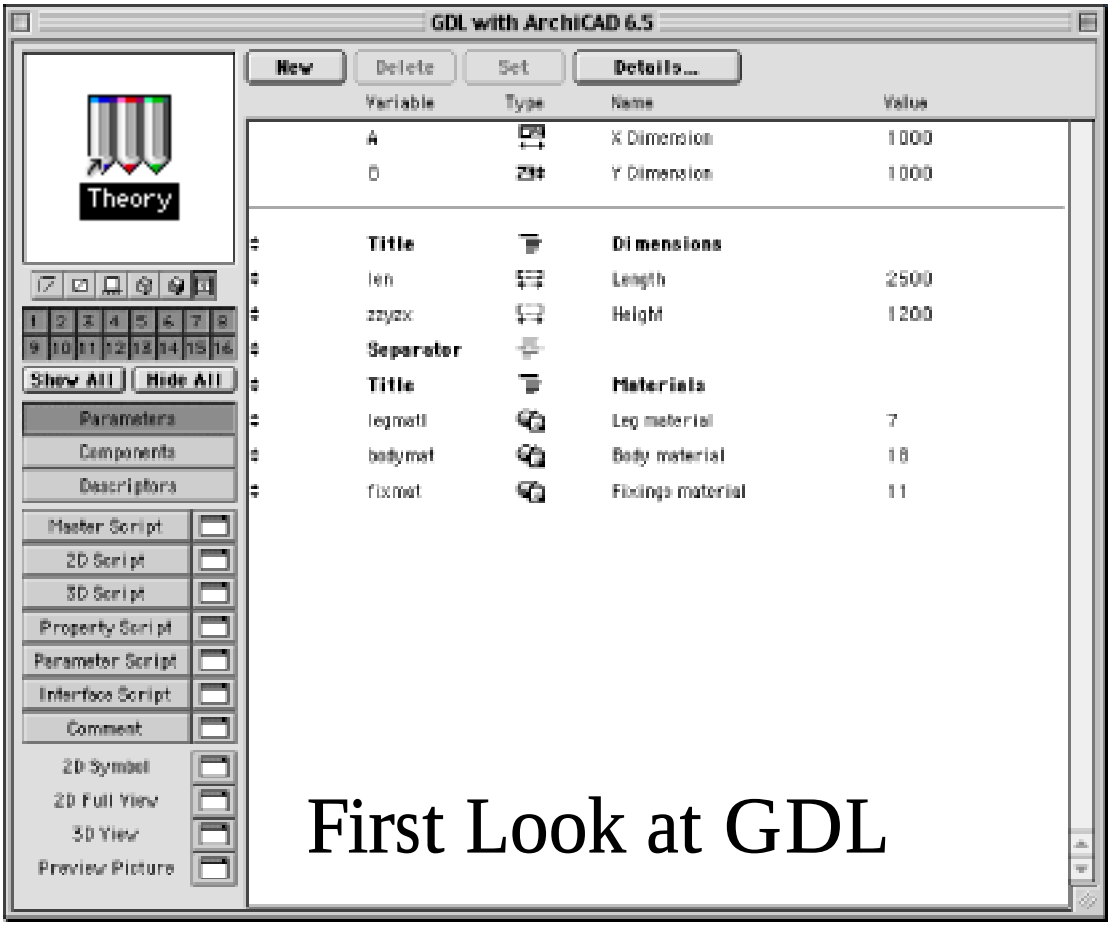
Another repeating 'Loop'

Another example of getting the parameter from the user's choice.

*Now this isn't written in BASIC, and it isn't written in GDL, but it's close to both of those two, and shows that programming can be like writing in English in a slightly mathematical style.*

OK, we can't all be perfect. The British Standard says put the milk in the cup first, and of course, I haven't tested to see if the users have chosen sugar, or want spoons, saucers or biscuits. And what happens if there isn't enough tea for everyone? Then you have to loop back and top up the teapot, or even reboil the kettle. This could be built into the program. That level of detail and error correction would make the program more professional and user friendly.

I hope this shows the sort of thing you can do, and has persuaded you that you can indeed write a program.




FROM the File menu, open any 3D object and have a quick look at the way the Parameter window is organised. A Pop down menu (in AC 6.0) and a vertical array of Buttons (in AC 6.5) allows you to view different script and window types. The main scripts and windows we are concerned with are:

Master Script :– there are many housekeeping tasks such as reading in Value lists, setting up Parameters, checking user Errors, defining Materials, setting Flags and so on. These are read by the 3D Script, the 2D Script and the Properties Script. If you didn't use a Master Script, you would have to write all these things time and time again.


2D Symbol:- is a window into which you could paste a 2D image, or draw using 2D tools, but it will only be displayed if there is no 2D script. With GDL knowlege, you are better to write 2D script.

2D Script:– this can be used simply to tell GDL to draw in 2D whatever it finds in 3D (Project2), or whatever is in the 2D Symbol window (Fragment2). You can (and should) use it for writing a parametrically organised script to draw what the object will look like (using RECT2 and LINE2). By designating Hotspots, the 2D script can also govern how stretchy the object can be. By leaving this script blank, the GDL object will display what ever is drawn into the 2D Symbol window.

2D Full View:– is generated by the 2D Script.

3D Script:– the primary means of building parametric 3D objects. If the object is simple, almost all the work can done in the 3D Script.

3D View:– is generated by the 3D Script – not to be confused with the 3D window of the main project.

Properties Script :– in which you could write Components and Descriptor commands if the object is included in a schedule.

Value List (now called Parameter Script in 6.5):– this is the means of creating Pop-Down menu selections in the main parameters box. It is read first, even before the Master Script.

Comment:- is a small text field in which you could write a small set of instructions to your user on how to use the object, or could record a log of the development of the object.

Preview Picture:- is a window containing a pasted in bitmap image of a view of the object. It gives the user an idea of what the object will look like in its setting, and could come from Artlantis Render or an ArchiCAD photorendering.

Parameter table:– we also fill out the table of parameters by hitting the New button, and filling in the small details. A and B are 'obligatory parameters' – they already exist – but we can make many more.


# Working in 3D space

JUST how do you write in GDL in 3D? Well, when you are word processing, you move your cursor to a location and start typing. Move the cursor, and type again, and the new words appear at the new position.

With GDL, you have a 3D cursor; when you issue a 3D GDL command like BLOCK or CONE, you will get a 3D object wherever the 3D cursor happens to be. Move the 3D cursor to a new position and issue the same command and now it appears in the new position.

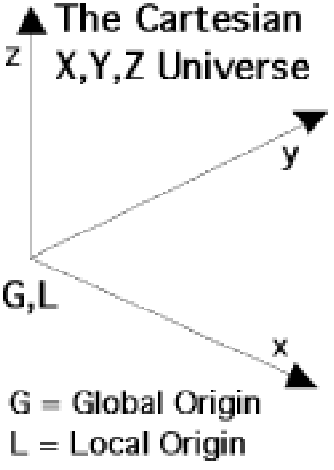
Thus to make a chair, you can move to each corner, plant a chair leg, then raise the cursor up to plant the seat and finally, move the cursor to plant the back of the chair.

When you have achieved a group of things (like a bunch of chair legs), you have completed a task. So now return the cursor to the global origin. Now you can depart and do another task, for example the seat of the chair.

When you have finished, you can do a small 2D script so that a 2D symbol will appear in the project plan, save the file, and you are done.

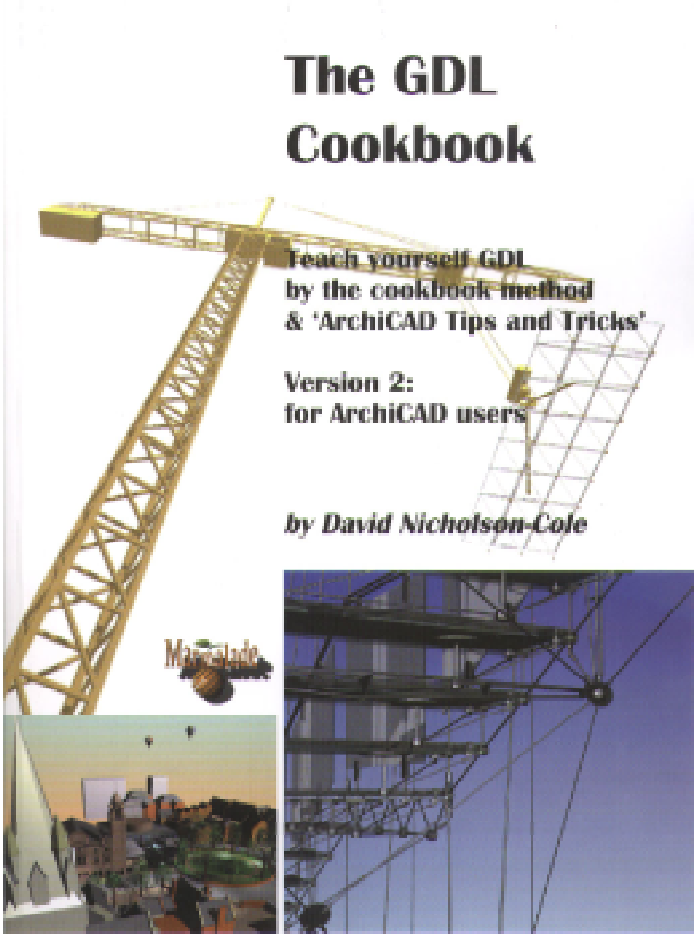
## The 3D world in which you move your cursor

FIRST get acquainted with the idea of the X, Y, Z universe that you have to work in. All locations are defined in these coordinates. If you want to move sideways, you 'add' in the X direction. If you move forward, you 'add' in a Y direction. If you move up or down, you 'add' in a Z direction. You can Rotate the cursor and the XYZ world gets rotated too.




- The 'G' (global) coordinates remain fixed at the Origin of the model. When you bring an object into the project plan, this origin is what decides the height of the object in the project. The origin should be planned carefully – preferably at the base of the object, and at the axis of any symmetry or rotational axis that you perceive.
- The 'L' (local) coordinates are like the moving cursor in your word processor – they travel with you, wherever you are drawing an component. Always try to return the GDL cursor to the global origin before you start out with another element of the model.

# Go on!
# Treat yourself to the GDL Cookbook this week!




## The GDL-Cookbook 2

**Available from selected ArchiCAD resellers for $39 (plus local taxes and postage)**