Object Making with ArchiCAD

GDL for Beginners

by David Nicholson-Cole

GRAPHISOFT®

BUDAPEST • MUNICH • SAN FRANCISCO • TOKYO •
LONDON • MADRID • SAO PAULO • SANTIAGO DE CHILE

Object Making with ArchiCAD: GDL for Beginners

Back cover credits: Marks Barfield Architects, creators of the British Airways London Eye, http://www.marksbarfield.com. GDL model by David Nicholson-Cole.
Published by GRAPHISOFT R&D Rt., http://www.graphisoft.com

First printing.
Printed in Hungary.

The version of GDL described in this manual is compatible with ArchiCAD 6.5 and ArchiFM 2000.

ISBN 963 00 3726 2

About this Book

Object Making with ArchiCAD is designed to help you get more enjoyment and productivity from ArchiCAD by taking you beyond the confines of the Toolbox (the tools palette) into the realm of object making with GDL (Geometric Description Language).

Objects – such as furniture, doors and windows, lamps, building components – can be made using the existing tools – wall, roof, floor, mesh, etc. They can be made with native GDL, scripted from start to end; or they can be made with a combination of the ArchiCAD tools and scripting. ArchiCAD users have long asked for a book that will lead users from tool using to object making, and we hope that this little primer will open up new worlds for each user. The book is not just about GDL; it starts with object making using the existing tools, and by progressive stages, leads you into GDL, but we hope you will find the transition enjoyable.

Some of the examples used in the book will be available from the support website.

About the Author

David Nicholson-Cole is an architect and teaches at Nottingham University in the UK. He has been an enthusiastic evangelist for GDL and a prodigious producer of GDL objects since discovering the power (and the pleasure) of GDL. He made his mark in the GDL world as the author of the *GDL Cookbook,* is a director of The-Object-Factory.com, and a founder member of the GDL Alliance. The *GDL Cookbook* logically follows after the book you are now reading. It is an A-Z of GDL technique, and is usable in its own right as a GDL manual. David has also travelled around the Globe at the invitation of ArchiCAD dealers and their customers, teaching GDL.

Acknowledgements

Object Making with ArchiCAD: GDL for Beginners was developed in concept by Graphisoft as a means of filling the gap between the *GDL Cookbook* and the existing reference manuals. It contains easy to follow exercises in object making, and pulls together wisdom about object making from other sources, including the reference manuals, for which all the authoring team should be thanked.

Chapter 1

Introduction to Object Making

An introduction to the possibilities of object making in ArchiCAD, with or without GDL.

1.1 About Object Making

The real 3D world can be thought of as a vast assembly of objects. ArchiCAD's 3D environment can be thought of as a large theatrical stage where you assemble the cast (the objects), the set (the drawing) and the script (the design idea). You bring the objects together, organize them, and then – let the play begin!

Most of the objects in the ArchiCAD assembly area are building elements – walls, columns, floors, roofs, meshes, etc., brought together to form buildings. These objects are easily made with the tools given to you in ArchiCAD.

Whenever you need go beyond walls and floors, etc., you can use special objects such as furniture, windows and doors, lamps and components. In ArchiCAD manuals, these are also referred to extensively as Library Parts. These can be found in your library or made specially. So, why should we bother to make objects? Well, we can make:

- Structural elements that are right for their purpose
- Details of building construction that look authentic
- Furniture that is smart and elegant
- Components that conform to catalog numbers and manufacturers' specifications
- Lights that can transform the environment of a model
- Windows that open and swivel, and offer a choice of styles
- Doors that offer choices of ironwork and glazing styles
- Stairs that enable you to vary landings, risers and handrails
- Picture objects that can be placed in a model to look like people, trees, or even whole buildings
- 2D drawing objects and tools that can enhance your productivity and drawing accuracy.

In short, the pleasure and productivity of the ArchiCAD user can be greatly enhanced with objects. This book is intended to ease the ArchiCAD user safely into making objects. It is not entirely self contained; it needs to be read with the GDL Reference Manual and the main ArchiCAD Reference Guide. Dip into these when you need to. If you wish to take object making further, your next stage after this primer is to work with the *GDL Cookbook,* which takes you progressively into more advanced GDL.

Library Parts in ArchiCAD

When you build an ArchiCAD project, you can use the primary modeling tools – walls, floorslabs, columns, roofs, etc. The Toolbox also offers access library parts: furniture objects, lamps, windows, doors. In the Toolbox or the Tools menu, you may also find add-ons such as ArchiSITE, StairMaker, Profiler and RoofMaker. To place an object in the project you can use different actions, such as click and stretch, click and click again to indicate direction, or you may encounter an entire dialog box containing instructions and many fields in which to enter information.

In your use of ArchiCAD, you will have noticed that most library parts offer many chances for variation (called parameters) such as the number of mullions in a window or glazing styles in a door. They may also offer different ways (in materials, pens and fills) of representing themselves in the 3D view and in 2D.

This introduction to object making will make it possible to build objects yourself; it will introduce techniques for making more than just 3D or 2D shaped objects – you can try to make them capable of offering variations – thus they can be parametric.

Libraries must be loaded

Whatever you make, however you make it, ArchiCAD must be able to find the object. Every time you open ArchiCAD, it reads through the library directories (folders) and makes an index of what it finds. The existing ArchiCAD Library is a folder in your installed ArchiCAD directory. ArchiCAD's own library will be loaded (by default) but when you are running a project, or experimenting with GDL, you should make additional folders of your objects. Make sure you do not make objects or folders with the same names as ones that already exist in the libraries. When you are working with ArchiCAD, make sure that the folder you are saving your objects to is one of your loaded libraries.

You can either make a new library folder that relates to current projects, or your objects can be stored in a personal library in the ArchiCAD folder. The main point is that you know where they are, and you have loaded them. Subfolders to keep furniture objects, building components, window objects, textures, etc., should be logically organized.

Sources of Library Parts

The ArchiCAD Library

It is worth going through the ArchiCAD Library included with your package before you start creating library parts on your own. Investigate what already exists. This will give you an idea of what you can use and do not need to make; and the example in there will give you ideas of what is possible. You will be able to see how parameters change and this will make you think about what qualities you might like to include in objects that you make – such as user friendliness, clear organization and description of parameters, stretchiness, 3D authenticity and intelligible 2D symbols.

Complementary Libraries

Several specialist libraries are available on CD from Graphisoft or their partners (e.g., People and More, MasterLibrary, etc.). If you explore the Internet, you will find an increasing number of websites devoted to ArchiCAD objects including The Object Factory (www.the-object-factory.com) or Objects On Line (www.objectsonline.com). The Graphisoft website gives you an up to date list of addresses of such sites. You will find that, with the free GDL Object Web Plug-in, ArchiCAD objects can be viewed in a browser, changed parametrically, and rendered, almost as if you were examining them from within ArchiCAD.

DXF and DWG Libraries

Most building component manufacturers offer disks of DXF files although most of these are 2D only. It is always worth getting these, as DXF files can be used in ArchiCAD, and you will be able to make use of them. 2D DXF and DWG can be brought straight into the Project plan. Where you see the opportunity to make library parts out of DXF or DWG originals, you may be able to take advantage of the parametric power available to you in GDL.

For 3D, you can also make objects with other applications from 3rd parties – Zoom, Alias Wavefront, Design Workshop, AutoCAD – save them as a DXF, then bring them into ArchiCAD as a library part. The DXF/DWG Conversion Guide included in your ArchiCAD package explains this procedure in more detail.

When you bring the 3D DXF into ArchiCAD as a library part, you can opt for:

– Don't Process: The object does not get converted to 3D. Only the 2D parts will get imported. This is the default setting.

– Binary 3D: The object comes in as 3D, but it is encapsulated into an uneditable block of machine code that is neat, tidy and stretchy (and more reliable when rendering), but leaves you no opportunity for further editing or parametric modifications.

– GDL Script: With this option, all the 2D and 3D data will be imported, but you will be faced with a colossal quantity of script. Nothing brought in from DXF is solid; it is a vast mass of polygons and lines, and in no particular order. If you attempt to edit these you will need a lot of luck and guesswork. The most you can be sure of editing successfully are Material names, and a simplified 2D symbol.

You may be disappointed with the result. You will have to spend a lot of time learning the new applications. 3D objects made in other applications will leave you with little or no control over the materials and parametric properties. In particular, you may find that the lack of control over the number of polygons leaves you with a model that has so many polygons that your rendering times become unacceptable. With such an excess of polygons, you may find it difficult to export your ArchiCAD model to a 3rd party renderer such as Art•lantis Render.

If you take the time to learn GDL, you have the advantage over other CAD users, in that you can make a 2D or 3D library part in GDL that is parametrically variable, and save the resulting varied forms as separate DXFs.

Add-ons that make Objects

There are a number of third party programs that can create or edit ArchiCAD library parts, for example, StairMaker and ArchiSite. You will also find a growing number of object-making Add-ons provided by Graphisoft such as RoofMaker and Profiler.

Objects created by these Add-ons contain GDL scripts as well as other, application-dependent data.

1.2 Making your own Library Parts

Without GDL – using ArchiCAD's Tools

Walls, Floorslabs, Roofs and other ArchiCAD 3D tools are available in the Toolbox as custom building blocks, regardless of their originally intended purpose. You could, for example, model a dining table easily by using walls and slabs. The legs could be tall, small slabs, or short walls. The tabletop would be a slab. If the legs were splayed as in a trestle table, you could use the Roof tool. The resulting structure can be saved as a library part, thereby making it available for repeated use in other projects as well.

Library parts originating from the floor plan can be saved either as GDL scripts or in binary format. GDL scripts are editable, so you can enhance the library part's 3D appearance by modifying its script. You can do useful work on a GDL script, mainly in editing materials and pen values. Binary library parts offer lightning fast imaging speeds, but do not allow editing, and have no parameters other than ones enabling stretchiness – width, depth and height. If you want to perform other modifications, you must return to the original floor plan document, modify the model, and save it again as the same or as another binary library part.

If you save floor plan elements as a GDL script, the complexity of the resulting script will depend on the element types you have used. The Slab tool will produce a mass of `CPRISM`s, which are relatively easy to modify. The Wall tool will create `XWALL`s which are harder but not impossible to modify, but are easier to build with in the first place. The Roof tool creates `CSLAB`s which are manageable. Some of the most interesting possibilities are with the Mesh tool, which makes commands called `MASS`, allowing one to create rounded surfaces.

Sometimes the objects you make by assembling pieces of floor slab, wall and roof are more easily built on their side, rather than upright. Windows and doors can be built flat on the floor and ArchiCAD will convert them to be upright.

Sometimes you will not be able to make an object all at once. You may have to make the legs first and save those, then make the superstructure, save that, then make additional parts, and save them. There are primitives such as cylinders, cones and domes in the ArchiCAD Library that you might also use as part of your object. You can then bring all those subsidiary parts together in the

floor plan, and save the whole result as your final library part. Later in this book there is an exercise to demonstrate this procedure. Be warned though, that this final object will only work if all the subsidiary objects can be found in the loaded libraries.

For the more experienced object maker, it is possible to combine the Toolbox with a knowledge of GDL. Because you cannot rotate objects in the ArchiCAD floor plan (other than around the Z axis) you will quickly see the benefit of learning even a small amount of GDL, in which you can rotate shapes however you want. Shapes are drawn out in the floor plan with the wall, roof or slab tool, saved as little chunks of GDL and then added into a larger GDL project that the writer is working on – moved and rotated into position, amended or simplified.

Not all objects need to be 3D. It is possible to use the 2D tools of lines, text and fills to draw out standard symbols, select them, and 'save special' as a library part. If you already have a 3D object that you wish to use purely as 2D, you could select it, 'explode' it, and save the resulting lines as a new 2D library part. This does not damage the original 3D part.

All library parts are objects, but when you save them you can decide if they are to be an Object (a piece of furniture or a building component), a Window, or a Door. You can go on to re-open the object and resave it as a Lamp.

Assemblies built with the Toolbox do not have to be saved as library parts. They can be grouped or saved as modules and reused; but if you wish them to behave properly as objects and allow further editing, they must be saved as library parts.

Making Objects with GDL

GDL, even a small knowledge of GDL, allows you to go far beyond the possibilities offered by the simpler methods above. If you can describe what you want, in writing, you can make objects with GDL. That's why it's called Geometric Description Language.

- Because GDL has many 2D and 3D commands, you can make interesting objects that cannot be made with the Toolbox **– they can be shapely and elegant.**
- Because you can define diameters, spacing, thickness, materials and pens, your objects can allow variations **– they can be Parametric.**
- Because you specify elements in a GDL model by precise dimensions, angles or parameters, the objects will be exactly what you want them to be **– they can be accurate.**

– Because you can write IF statements, you can build rules of behavior into your objects, such as manufacturers' requirements, checks on incorrect parameters, self-sizing components etc. **– they can be Smart.**

– Because you can write routines that loop around and repeat commands many times, you can economically build large or repetitive structures that would be futile to build with the Toolbox **– your objects can be Tools.**

– Because you can rotate, slide and resize elements in the script, your objects can change their shape
– you can make Mechanisms.

As a bonus, GDL enables you to be utterly accurate – if you specify a cylinder to be 3.60 meters long, that is what it will be, not 3.597623. You also have the ability to decide which parts of a model can cast shadows, which pen color parts of the object should be drawn with. You can decide what resolution of curvature should be applied to individual components of the library part. You can generate your own materials – shiny, matte, transparent or glowing – so that they will work even with an ArchiCAD in a different language from yours. GDL objects are independent of dimensional settings; you can build something in metric that will work if the user works in imperial, or vice versa.

How do you make an Object?

Look around you, examine the 3D objects that surround you, as you read this book. Observe, if you can, the essential primitive forms that compose each object. Most objects can be seen as composed of blocks, prisms, cylinders, cones, tubes or surfaces. Ask yourself what detail needs to be included; discard detail that is not essential or will be impossible to model. Use a level of detail that is within your abilities to model either from the Toolbox or with GDL script. Impossible 3D tasks such as surface rippling or perforation can often be solved by using textures.

If you are not using GDL, view the object in terms of the familiar tools of wall, slab and roof.

If using GDL, you can see it in terms of blocks, spheres, cylinders and cones. As your GDL skills grow, you can add prisms, sweeps, plane and meshed surfaces and tubes.

Does it need to LOOK right, or BE right?

An object that is used to furnish the interior of a house may only be required to give a luxurious impression of the interior, an authentic

'lived in' look. It only needs to LOOK right. This could be built with ArchiCAD's own tools or with GDL, and the dimensions could be 'eyeballed' – that is, you use the grid as a guide and get the size as accurate as your eye tells you it needs to be.

Another object may be required to be totally accurate, so that you can derive accurately dimensioned sections and plans from it. It may be required to drive a CADCAM system (cutting tool) or a rapid prototyping machine. So you must be satisfied that it IS right. For this, you are better off working in GDL from the beginning.

The Idea of an 'Investment object'

Ask yourself if the object you want to make is to be used more than once, if it could be useful in other projects. You might be the one person in the office making objects for other people, and therefore you need to make the object usable by others. If you make it parametric and smart, it might take you longer to make an object that could more quickly be made with the Toolbox. But if the object has versatility and can be applied to many more projects, the time taken to make the object well is a good investment and your users will be grateful for the effort.

Furthermore, objects made with the Toolbox are no longer editable if the original floor plan file that made them has been lost.

A piece of GDL that is well written can be extended and made more powerful when your own knowledge of GDL improves or when a new edition of ArchiCAD offers more features. A well written object could be taken on by someone else in the office, years after you have left, and improved to take advantage of the features in a future version of ArchiCAD.

Can you learn GDL?

Many objects do not need to be built with GDL – the ArchiCAD tools may be enough. Experienced ArchiCAD users who are scared of GDL get very skilful in transforming the simple 3D tools of wall, slab and roof into most complex objects. But they will never have the pleasure of making their objects parametric.

If you are prepared to go beyond the Toolbox, GDL is an easy scripting language compared with those available in the other CAD packages. GDL is based on BASIC, the easiest of the programming languages. Millions of people learned BASIC during the pioneer years of microcomputing, and GDL remains the most useful and practical descendant from BASIC. Experts who used to say that BASIC was not powerful enough for commercial

applications can be confounded by a well written piece of GDL. ArchiCAD users who make the bold step of tackling GDL are frequently surprised at how easy it is compared with what they expected. Users can suddenly find a practical use for all that Mathematics they learned at school.

Total beginners can please themselves with rapid progress up the learning curve. They can make objects that impress their friends and colleagues, and genuinely improve the quality of their work.

Experts in GDL continue to find additional depths and techniques in GDL, and with energy and programming experience they can extend the possibilities. As soon as they begin to feel they have reached the limit, Graphisoft's next edition of ArchiCAD provides new areas of exploration and power.

Object Making in the ArchiCAD Manuals

Besides this book, you can read about object making in other manuals and guides:

ArchiCAD Reference Guide: tool and library descriptions (see table of contents and index of that volume).

DXF/DWG Conversion Guide: Bringing in objects from other CAD environments.

Library and Environment: Illustrated listing of most of the library parts supplied by Graphisoft in the standard ArchiCAD Library.

Step by Step Tutorial: Making library parts (see table of contents of that volume).

GDL Reference Manual: Syntax of GDL commands, in depth information on every aspect of GDL except programming.

The GDL Cookbook: Project-based approach to learning GDL that teaches both GDL and the technique of programming with GDL from beginner to expert level.

Project Framework: Discussion of how to get the most out of ArchiCAD in the professional office, with good explanations of organization of your libraries.

> **Note:** The GDL Cookbook and Project Framework are not part of the ArchiCAD package and can be purchased commercially. Consult your local reseller.

You could also consider keeping a personal notebook of your progress with object making. By noting your successes, you can build up a personal fund of expertise.

Chapter 2

Making Objects without GDL

Before learning GDL it is wise to become fully conversant with object making using ArchiCAD own tools.

Making Objects without GDL

Most ArchiCAD users become skilled at making objects using the normal building elements in the Toolbox before they tackle GDL. So, let us make sure you are fully conversant with object making before moving on to discover the power of GDL.

Make the object using the Slab, Wall and Roof tool, and select it in the floor plan window with the cursor. Then:

1. If the object has been made the right way up, choose Save Special/ArchiCAD Object in the ArchiCAD File menu. Save it into one of your loaded libraries.
2. If the object has been made the right way up, or on its side, view in the 3D window in plan or elevation, Save the resulting view as a library part – a .GSM object. Save it into one of your loaded libraries.
3. If the object is 2D, draw it into the 2D symbol window of a new library part.

ArchiCAD creates a written script for the new library part, including the 3D script, a 2D script or symbol, and routines to ensure stretchiness. This process is called Autoscripting. When you have made objects, you will be able to view the scripts – highly educational! The GDL it generates could be called 'Industrial GDL'; quite off putting and a lot more complicated than the 'Creative GDL' that you can attempt with the help of this book.

Getting Started

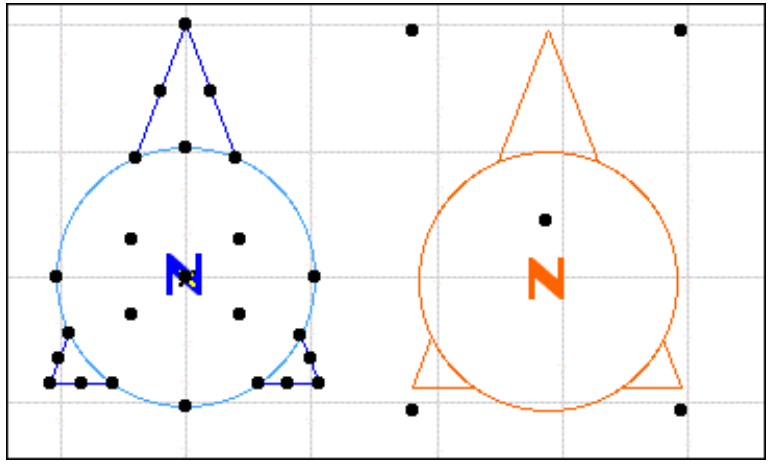
Before making your first object with the Toolbox, ensure that you are conversant with the use of the Magic Wand (for transforming a Fill to a Slab, or a Slab to a Roof), setting Grids, using the Coordinate Box, and with using the Settings boxes for the various tools in the Toolbox. Open a new ArchiCAD file, load a new folder to use as your library and save the new file to that folder or to somewhere convenient. Set the grid to 1'-0" or to 300 mm. Zoom in to somewhere near the Origin of the Floor Plan, so that you are viewing an area of the grid that is a little larger than the furniture you wish to build. Make sure this is the real Origin, not a temporary one.

If you have made a temporary origin somewhere else and cannot find the main origin, just click on the temporary origin and hit the delete key. The main origin will blacken.

Let's Make something in 2D

Try making this North point. As this is purely a 2D symbol, there is no scripting to do. You could use this for electrical symbols, kitchen or bathroom furniture, or as a complicated symbol for trees and plants.

Draw it in the Project Plan. You can use any of the 2D tools to do this, including Lines, Arcs, Fills, Hotspots and Text. To make it into a library part, use method 1: Select the drawing, then use File menu>Save Special>ArchiCAD Object. Save it into a loaded library. Then bring it straight back by selecting a library part from the Toolbox and placing it in the Floor plan next to the original drawing. You will find that the most recent library part saved is the first object that appears when you place an object in the plan.



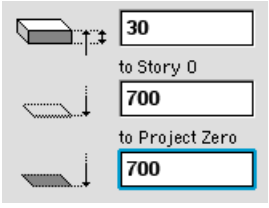
If you do not place any hotspots, you will get what is called 'the bounding box' around the object, as illustrated (right), and it will be stretchy. If you place hotspots, even as few as one, you get no bounding box, but the hotspot that you placed will be visible (it will no longer be stretchy). So if you place one hotspot, you should place more. If you use the Fill tool instead of the Line tool, you will find that ArchiCAD places the hotspots for you at all the significant nodes. If you want it to be stretchy, then place hotspots at the extreme left, right, top and bottom edges (unless the Fill tool reaches these extremities).

You could try another method. You can use File menu>New Library Part>Object, and open the object master window, with a range of buttons allowing you to display scripts or graphical windows. You can draw in the 2D symbol window with the same

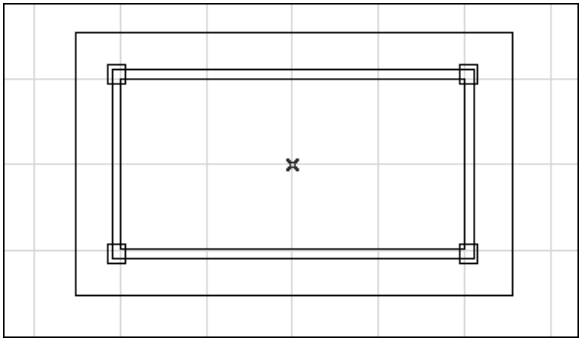
2D tools as in the main ArchiCAD floor plan. Save the object into your loaded library. Use this method only when you have become more familiar with this palette.

Let's Make a 3D Object – a Table

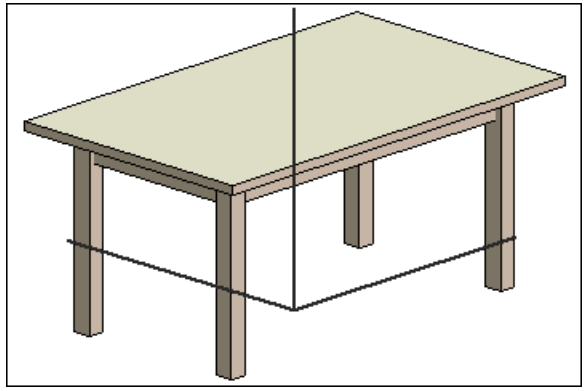
The next object, a table, is to be made 'the right way up' and for this, the Slab tool is most useful. Build up a table, starting with the table top, and add the legs and frame. For each part, use the settings dialog box to get the height of the slab correct. Remember that the height (altitude) of the slab is the top and the thickness is the amount of slab below the top. If you build from walls, the altitude of a wall is its base, and the height of the wall is above the base. Save it using File>Save Special>ArchiCAD Object.



Remember also that you need to set the materials of the tabletop and legs using the Slab Settings dialog box.



To make sure that each leg is consistent, make one correctly, then Edit>Drag a Copy to each corner of the table. If you use the grid, you can make the table reasonably accurate in dimensions. If you keep an eye on the Coordinate Box and type in your X and Y locations, you can make the table extremely accurate. You can make it even more accurate by setting Grid Snap to ON, and using a small grid of say 10 mm or 1/2".

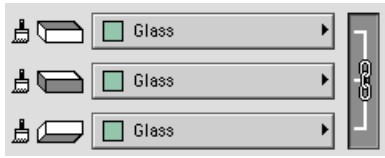


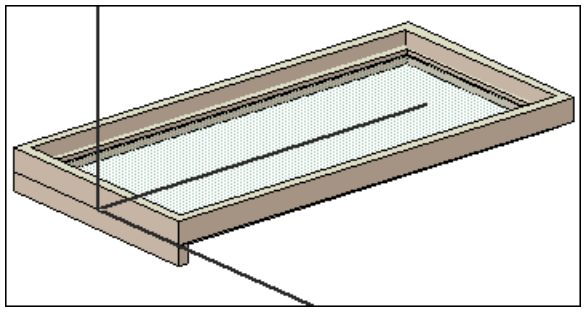
If you use the Arc/Circle tool's ellipse line option and then 'magic wand' a slab to it, you could make a more stylish and elegant table. This table will be stretchy, but if you stretch it, the section sizes will deform.

No matter how well you make it this way, it will never permit you to make parametric alterations to height, leg spacing or timber sizes. This is where a little knowledge of GDL comes in.

Let's Make a 3D Window

Windows have to be made flat on the floor, and are best made with the Slab tool. Again, it is best to build it over the origin, and the origin should be symmetrically at the centre of the lower sill of the window. The outside face of the frame must lie at zero height on the ground plane. Normally, slabs have their tops on the ground plane, so be careful to set your window frame at the right height. Draw out a rectangular slab first which defines the overall opening size. Then select this slab, and with the Slab tool selected draw another rectangle within the first slab. This draws a hole in the slab and effectively makes the frame hollow. Make sure that it is the right height – the depth of the frame. Set the material of this slab to frame material. You can now draw another slab into the window, set this to 10 mm (3/8") thick and set it to "Glass" for top bottom and sides. Make sure the height of the glass is well positioned in height, within the frame. For added realism, you can add a smaller frame to play the part of a casement, and a small slab below the XY ground plane to be a sill.

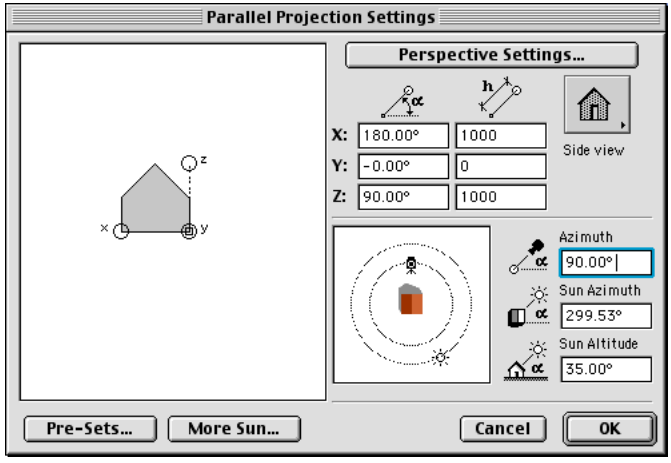




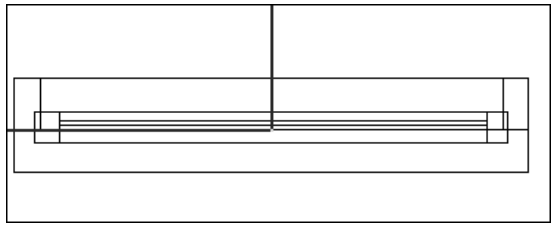
The illustration above shows how the window should sit on the XY ground plane. Check that the frame/casement/glass relationships are working correctly.

In this case, the project origin is neatly positioned at the centre of the sill. This is not absolutely necessary because ArchiCAD will make adjustments when it saves the window so that the origin finishes up in the right place. But it is tidy and more disciplined to get it right at the start.

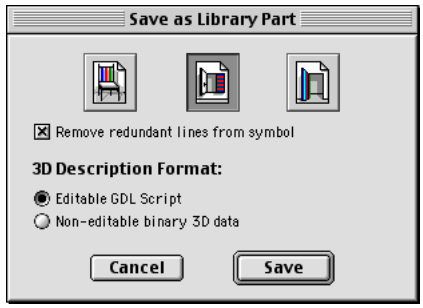
When you are done laying slabs, select all the slab elements of the window, and view them from the direction that you would need to see them if the finished window in the floor plan. Select Image>3D Projection Settings. You can use either shaded view or wireframe view.



The camera should be positioned at 90° and the view type should be Parallel Views>Side view.

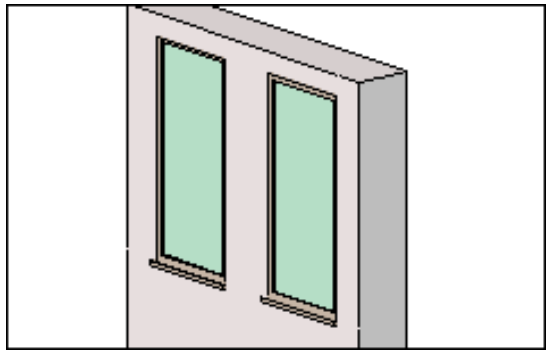


The 3D view will look like this, and if you ask for it to be Wireframe, you will be able to see the frame and glass outlines. This is as it will appear in the floor plan, when set into a wall.



Save that view as a Library Part and when asked, click on the Window icon. Check the box that hides redundant lines. Make sure you save it into a loaded library. If you have not already done so, make a folder in your personal library for windows and doors.

If you clicked the window or door icon, ArchiCAD forces the camera view to be 90° Parallel>Sideview even if you got the 3D view incorrect!



Now, back in the floor plan, build a wall, and then select a window to put into it. The window you have just saved will be the first one that ArchiCAD offers you. Place it into the wall, making sure that the Eyeball cursor is clicked to the outside face of the wall (the face with more hotspots). You can now view the result and change its parameters – of size only.

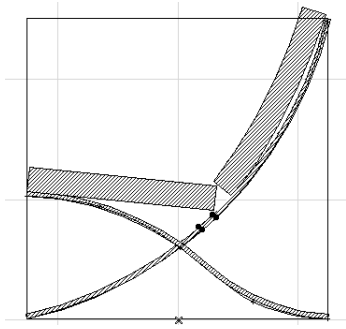
The technique of making windows is similar with door construction. But you would need GDL to change the door style, to make it open and shut and to select a choice of glazing or ironwork styles.

Let's Make a Furniture Object using Walls

Many objects are easier to make lying on their side, because of their complexity or shape. They may be far easier to build in profile than to build as a stack of objects assembled vertically.

You can make skillful use of the Wall tool. Walls are most useful as they can be set to a thickness and height and can be snapped to follow lines with the Magic Wand. They are more controllable at following curves than the Slab tool. Their only trouble is their annoying tendency to join together. In the Options menu, turn off Clean Wall Intersections.

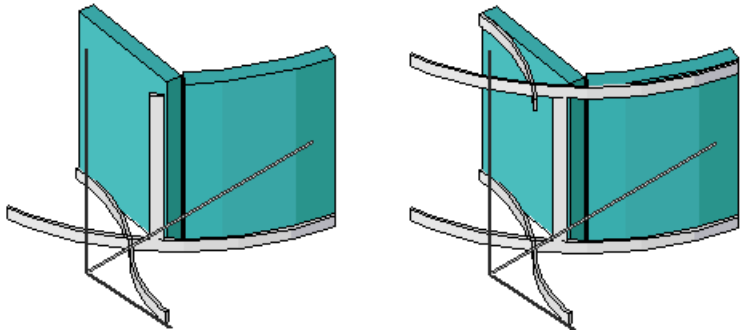
This Barcelona chair illustrates the use of the Wall tool perfectly – it could be made with the Slab tool but only with the greatest difficulty.



Making the Frame and Seat

First, over the Origin, within an outline box of 750x750 mm, draw the outline of the chair frame using the Spline line tool. Then set the Wall tool to be "Stainless Steel", 12mm thick and 50mm high.

Snap the Wall tool to the lines with the space bar depressed; this causes the little walls to follow the lines. Select and group the walls; drag a copy of the wall group to one side. Now use the Elevate command to raise that wall group by 700 mm. Now move that wall group back until it is exactly over the first. You now have the steel frame of the chair.



For the upholstery, change the wall settings to 750 mm high and 60 mm thick, and made of “Textile” (or “Leather” or “Asphalt”). Draw the walls for the upholstery yourself; the seat is flat, and the back is slightly curved using the curved Wall tool option.

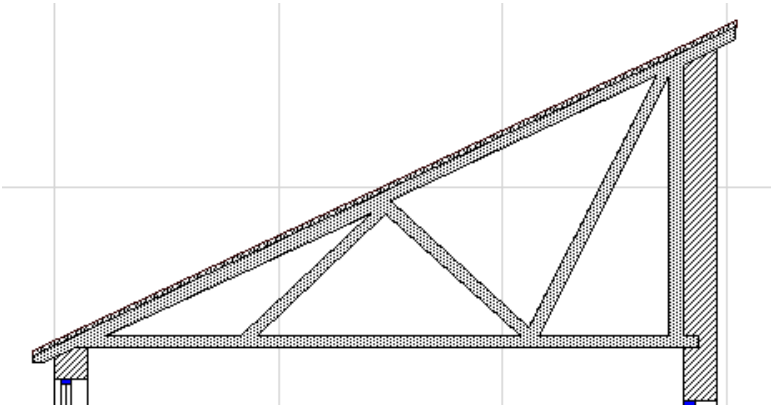
When you have something like the Barcelona chair in the diagram, set the cameras exactly as you did for the Window – 90° position for camera with Parallel Side view (elevation) selected. Save the 3D view as a library part into your loaded library; this time, click the object icon, not the window or door.

Place the chair into your project and enjoy!

Let's Make a Roof Truss

Try this exercise, which provides a link between the section and the plan windows.

Construct a small one-room building, and place a pitched roof over it. Make the walls higher than the roof and then use the Trim to Roof command to make the walls and roof fit each other. Insert a window and door if you like. Now using the Section/Elevation tool from your Toolbox, draw a section through the building, and view the section.



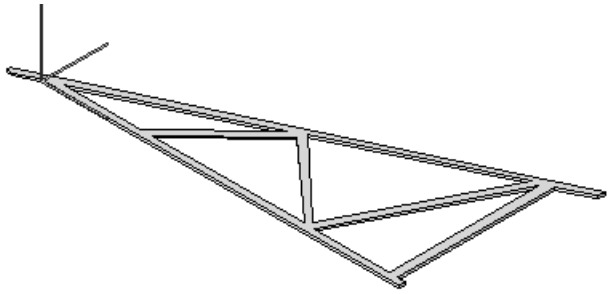
You can make use of the Line tool to set up guidelines to make sure that your timbers lie at the right angles and have the right depths

When you have the section in the foreground, the 3D tools go grey because you cannot use 3D tools in the section. But you can draw with the 2D tools, Text, Lines, Fills and Hotspots. Use the Fill tool to outline a small truss. This ensures that your future truss will fit the roof perfectly. Select the fill and continue to draw into it, and you will be able to cut holes. Thus you form the outline of the whole truss.

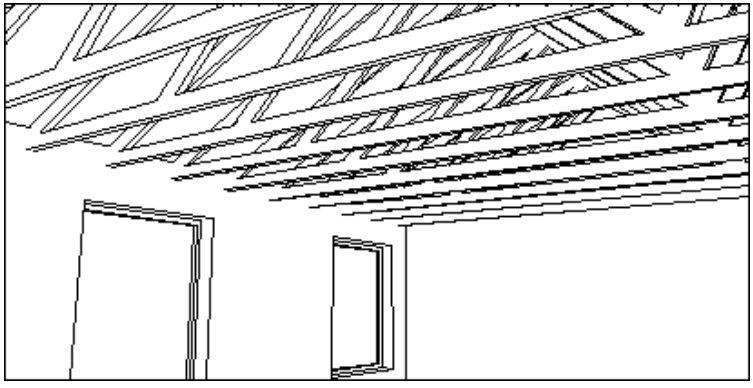
Now select the fill, copy it, move to the floor plan and paste it. Move it so that one end corner of the truss is over the Origin – preferably one of the truss bearings. The hotspots will help you. Now you can magic-wand-click the Slab tool to follow the outline of the Fill.

Make sure before you use the Slab tool that it is close to the ground – set the thickness to 40 mm, the top of the slab to 20 mm

and the material to timber. Thus the truss is nicely centered above and below ground zero.



You may have a bit of trouble with the Magic Wand. Snap-click to the outline of the fill first to get the outline of the truss. Then select the slab (making sure it is the slab and not the fill that you have selected) and then snap-click each of the holes in the fill. This will drill holes in the slab. You should finish with a small truss like the one above. If you get a lot of slab bits and no holes, delete the waste slabs and try again with the sequence above. Make yours more complicated if you want to, with more members, or more varied width of members.



Now view the truss from the same angle as you used for the window and the chair – 90˚ position for camera with parallel elevation view selected. Save the 3D view as a library part, click the object icon, and you have your truss.

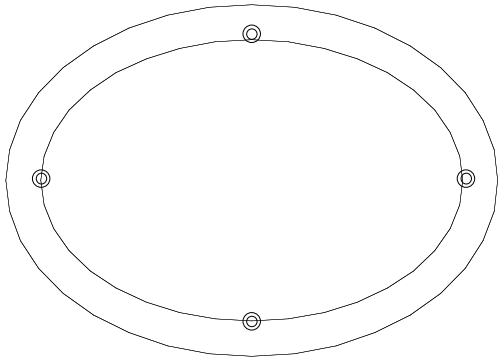
Now bring this truss into the floor plan, and space the new trusses along the building. Use the section view first, to make sure your truss aligns perfectly with the original fill pattern. The hotspots will help. Autoscripted objects are automatically stretchy so be careful

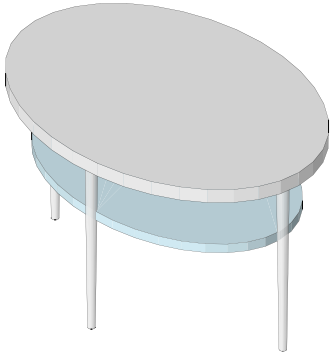
when handling them. You do not want to stretch them accidentally. Use Edit>Drag a Copy or Multiply to move and duplicate them, and use the Coordinate Box if you want exact spacing.

This same method can be used to make extraordinarily complex structures – the limitation being that they will not be parametric, and they will all be rectilinear in section.

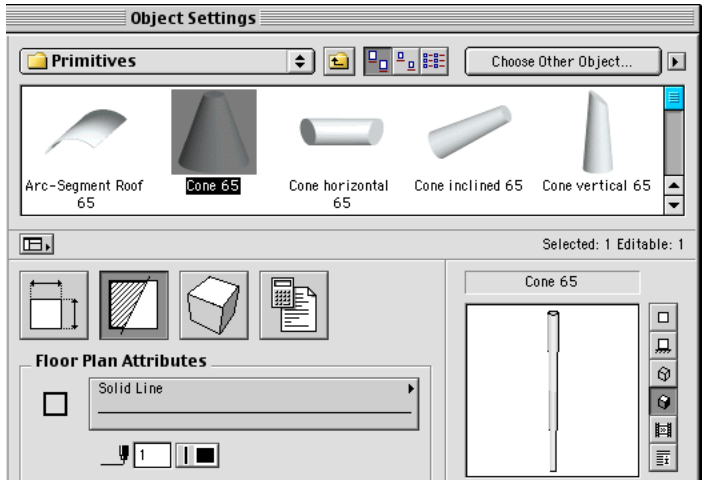
Let's Make an Object using existing Library Parts

You can make more complex or curvy objects using library parts that are in one of your loaded libraries. The standard ArchiCAD library contains a directory of special structures such as cones and vaults. Before you try to do this, just check which libraries are loaded (from the File menu) and if need be, load the default ArchiCAD Library.





This little oval cocktail table has been built entirely from the 'Cone65' object in the existing ArchiCAD Library. Try building something using the same or other objects in that directory. Some of these are somewhat confusing to manipulate, but the library parts settings dialog box makes it easy to play with the parameters if you display the 3D view of the object.



The settings box with visual icons makes it easier to select an object from the library

If you perceive that an object that you wish to make is to be made up of many components, then you can make them, one piece at a time, save them into a loaded library, then assemble the final object from the components and save that.

Remember that if you ever wish to view the final object, you will always have to be able to find the constituent parts in a loaded library. If you alter one of the components, it may result in an error when the final object is viewed. We have become used to seeing the 'Missing objects' dialog box coming up at some time or other. This is useful; if you read it, the warnings of duplicate library objects and of missing ones require action.

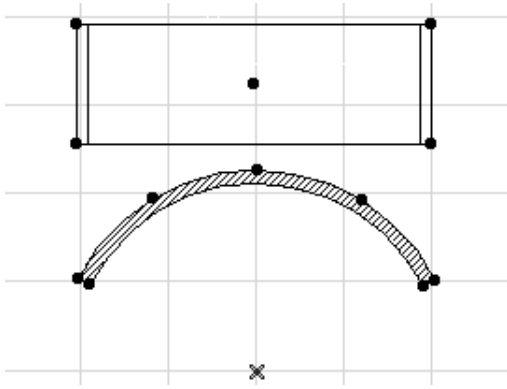
If the composite object never needs to be edited again, then you could view it in plan 3D (method 2) and then save as a binary. Thus all the subsidiary objects are included and you do not need to worry about missing the components.

Let's Make an Object that we can cut to shape

The oval table is an example of nested objects (objects within the larger object). As a further example of nested objects, we could make a major element of a building, a rounded vault, and further refine it with the ability to form a crossing - a groin vault.

This is something we cannot do with the Slab tool. Try using the more powerful Wall tool and draw the vault on its end. With the circular wall variant of the Wall tool, we can draw the outline of a vault. If you are more ambitious, follow this example making it as a Gothic vault (instead of a Romanesque) by having two sections of curved wall meeting each other.

Set the camera in Side view at 90°, view it in 3D, then save as a vault object. Call it "main_vault.GSM" if you like.



Picture of the basic wall in plan to form a flattened circular vault, above it, the resulting library part – stretchy in three dimensions

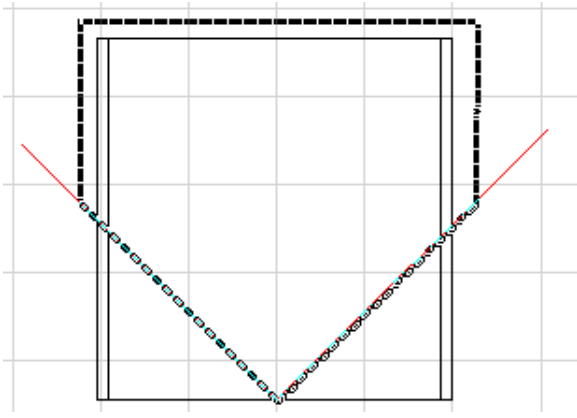
Bring it into the floor plan and you now have a stretchy vault. You can change its length, width and height. So it may have been round when you first built it, but it could now be elliptical if you wish. Drag a copy of your vault object and place it near the first one. Stretch the new one but keep it the same width. We are going to cut (or mitre) the vault at 45° angles to make it usable as a vault for the crossing.

Toolbox icon of the Marquee tool with the polygon and rectangle variants

Now you may have noticed that one function of the Marquee tool is to act as a cutter when you view in 3D. Everything within the marquee is displayed, and everything outside is ignored. It's a great way to do instant 3D sections.

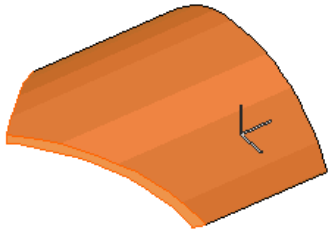
The marquee can be a rectangle or it can be a polygonal shape, like a fill. First draw one 2D line along the end edge of the vault, between the end hotspots. Now when you hover the cursor over the end of the vault, you get a little 'tick' mark to indicate the half width of the vault. Now draw two 2D lines from this half point at 45˚ angles to indicate the line you wish to cut along. By using the Shift key, you can ensure that these lines are at exactly 45˚.

Now select the Marquee tool and select the polygonal variant of the marquee. Draw along the 2D lines as accurately as possible and then encircle the rest of the vault that you intend to retain.



Plan of the shimmering marquee over the vault object. Note that the thin 2D lines ensure that your cutter is at exactly the right angle. Use the Coordinate Box if you like to type the angle in.

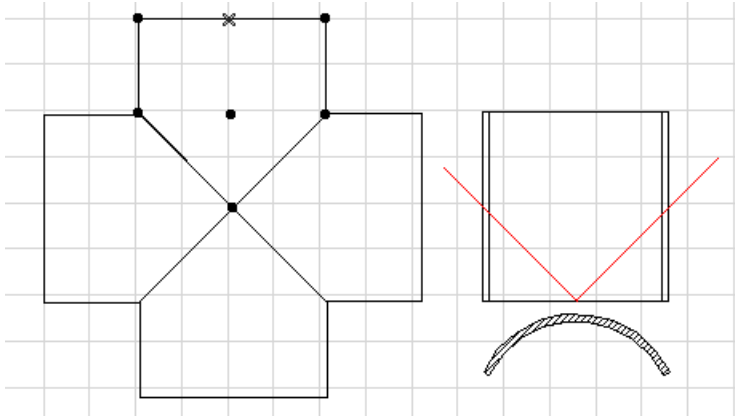
Now view this in 3D axonometric and you will be delighted to find that you have a groin vault mitred at 45˚. If the cutting effect does not occur, close the 3D window, use the Display>Rebuild command, and then try 3D again. Now use the 3D projections to set the camera position to 270˚ and the view type to Parallel>Plan. Using method 2, save this as a library part and you can call it "vault_groin.GSM". Return to the floor plan, click on the Object tool icon, click in the plan, and your new object arrives.



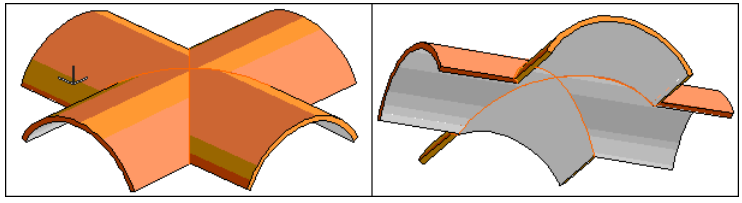
Picture of the single vault in 3D axonometric view. Go to 3D Plan view at 270˚ to save this as a new object.

Now this object has default 'Bounding Box' hotspots. You can customize your hotspots. Select it, and choose File>Open Library Part. Open the 2D Symbol window, and place your own 2D hotspots at the corners, including one at the pointy apex of the groin-vault and one in the middle. This will remove the bounding box altogether. The hotspot at the pointy corner can now be used as a hinge around which to Edit>Multiply>Rotate another three groin-vaults.

Do this multiplication, and you will get a plan view like this.



Plan View of the four vaults in plan view, alongside the original wall piece



Picture of the four vaults in 3D axo

You can now add in the uncut vault objects alongside these, stretch to appropriate lengths and before you know it, you have your church roof. You can use the same technique to take a sideways 'bite' out of a large vault and have a pair of smaller vaults intersecting. All this can be done without GDL, and yet, when they see it, your friends and colleagues will be convinced you have become a GDL expert.

Picture of the final structure, with arches and some columns

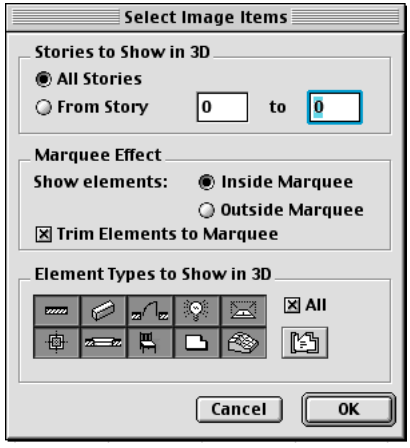
How far can we take Object Making without GDL?

Entire buildings can be library parts. ArchiCAD is a powerful tool for building modelling, and a finished building is an elaborate combination of layers, stories, library parts and building elements. In a city, every building has different story heights and your layer titles for a single building model are going to be concerned with city blocks, street furniture and landscape. In a building, your layers are concerned with differentiating internal walls from external, furniture from construction parts.

If you are doing City modelling, your project model will be the entire city area or a city block, and the elements in it will be the terrain, the roads and landscaping. The buildings are best brought in as individual library parts.

If you wish to include buildings you have already made, you should open the file of the building, try to make the windows of "Ice" or of a less transparent material, hide all layers to do with internal furnishing and small details (unless they can be seen

through windows) and hide the site on which it sits. Then select the entire building using the fat marquee, or view it in 3D in plan view. Either way, you will get all stories showing. If you had a basement that you wished to omit, then use Image>Select Image Items and you can decide which stories to omit. Save as a Library Part.



If you save it as editable, you are in danger in the long term of losing the subsidiary library parts (such as windows) that it was built from. It may also fail to render due to spurious errors. Save as binary, and you can safely export it to your City Model; place it anywhere and at any height and it will work. You would need a copy of the original if you wish to go back and modify any part of it, as the only editing you can do to a binary is to make it stretchy. If you do not want the binary building library part to be stretchy, open the building objects, open its 2D Symbol window, and place 2D hotspots where you think would be appropriate – corners, turrets, entrance canopy, ridges, etc., and then save again. That will turn off the bounding box hotspots.

If you only need facades, you can also build an entire facade on the ground, using the Slab, Wall and Roof tools. Then save it using the 3D view with the camera at 90˚ in the Parallel>Elevation view. Bring it back into your floor plan and assemble the building from facade library parts. Again, we advise that these are binary, to avoid any lost subsidiary components, and to reduce the risk of rendering errors.

Notes on the naming of parts

All saved objects end with the letters .GSM; windows end with .WIN; doors end with .DOR; and so on. Even Mac users work with this convention. Do not alter these suffixes either when saving or at a later time. Do not alter the names of objects in the libraries or your project file will not be able to find them next time it loads.

The best method is to think of a good name at the time of saving. Avoid using generic names for library parts – 'Table.GSM' sounds like a good name for a table – but the next time you open your project file, you may find a quite different table appearing because one of that name already exists in a different folder. 'Table_jdoe_02.GSM' is unlikely to exist already in the ArchiCAD Library. Use that, and your project file will be able to find it.

Summary of Object Making in this chapter

- You can make complex objects without GDL using the 3D tools or using library objects that already exist. They will be stretchy but not smart, and not parametric in anything except stretchiness.
- These can be 2D or 3D and can be saved from the floor plan, or from the 3D view.
- Windows and Doors can be made without using GDL.
- The Wall tool is powerful for object making without GDL.
- Using the Magic Wand, you can 'steal' shapes from ArchiCAD and turn them into Objects.
- Using the Marquee tool, you can cut library objects into smaller shapes.
- You can take object making to the extreme of having entire buildings or landscapes as objects.
- Take care with the naming of library parts – avoid using generic names for objects.

Chapter 3
Starting with GDL

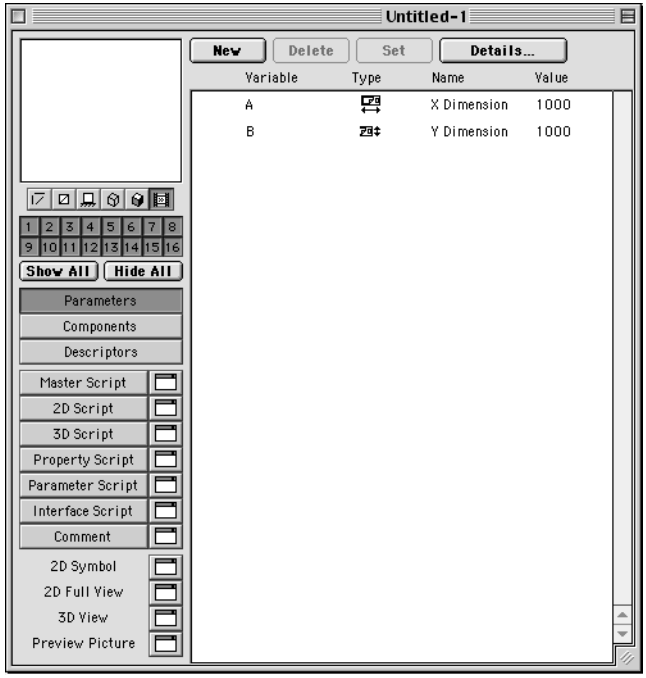
Now it's time to gather your courage and make a start with GDL. It's not as difficult as you expect!

3.1 Starting with GDL

Chapter 1 discussed reasons why you might want to build with GDL. Now it is time to give it a try. Do not be nervous. You will find it easy if you take it in stages.

Open the Object Making Master window

From ArchiCAD, select from the File menu>New Library Part> Object. You will get a Master window like this, with a vertical bar of buttons down the left-hand side. The wide buttons allow scripts to be displayed in the large white window. The small white buttons allow the scripts to be displayed in floating palettes.



When you have the Parameters button pressed, you get a 'New' button at the top – enabling you to create new parameters for length, materials, pen colors and more.

Before starting with some easy exercises, let's look at the function of the scripts and windows available to you. Bear in mind that for the beginner in GDL, the only scripts that you really need are the 3D and the 2D scripts.

The **Parameter table** is where you can build the Object Settings dialog box that the user will see when they use your object. This

allows the user to enter materials, diameters, angles, pen colors and much much more. By hitting the New button, and filling in the small details, you can create parameters.

The **Master Script** can be used for housekeeping tasks such as checking user errors and defining materials. This is read by the other scripts. If you type in a common piece of information like the radius of a tube or the width of a chair seat, it becomes available to all the other scripts. It saves you from repeatedly typing the same information into each script.

The **2D Symbol** is a window into which you can paste a 2D image, or draw using 2D tools, but it will only be displayed if there is no 2D script. With GDL knowledge, you are better off trying to write a 2D script.

The **2D Script** can be used simply to tell GDL to draw in 2D whatever it finds in the 3D script (`PROJECT2`). It can be used to force the 2D Symbol window to be the 2D symbol (`FRAGMENT2`). Its most creative use is for you to write a parametrically organized script to draw what the object will look like in 2D. By designating Hotspots, the 2D script can also make the object stretchy and easy to pick up. If the 2D script is left blank, the object will display whatever is drawn into the 2D Symbol window. If you leave both blank, the object will not show in the project.

The **2D Full View** shows what is generated by the 2D Script. This will not normally show what is in the 2D Symbol window.

The **3D Script** is the primary means of building parametric 3D objects. If the object is simple, almost all the work can done in the 3D Script. This is the engine room of most library parts.

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

The **Properties Script** enables you to write Components and Descriptor commands if the object is to be in a schedule.

The **Parameter Script** is the place where you can build pop-down menu selections to be used in the main parameters box.

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

The **Preview Picture** is a window containing a small bitmap image of a view of the object. It tells the user what the object will look like in its setting, and could come from Art•lantis Render or an ArchiCAD photorendering. This becomes the icon of the object in the settings dialog box.

The **User Interface Script** enables you to build a custom dialog box with your own text fields and images, with buttons and input fields for the user to enter parameters. It is great for complex objects where the user might need more explanation of the purpose of parameters.

Relax! When you start GDL, you only need the 2D and 3D scripts.

3D Entities

The easiest 3D elements to build with are Block, Cylinder, Sphere and Cone. Their syntax is very simple.

– `BLOCK x,y,z`
x being the width, **y** the depth and **z** the height.

– `CYLIND h,r`
h being the Height, and **r** the Radius.

– `SPHERE r`
r being the Radius.

– `CONE h,r1,r2,90,90`
h being the Height, and **r1** the Radius at the base and r2 the Radius at the top. The 90, 90 are the angles at which you cut off the top and bottom. You can vary these numbers to get interesting effects.

First steps in 3D GDL

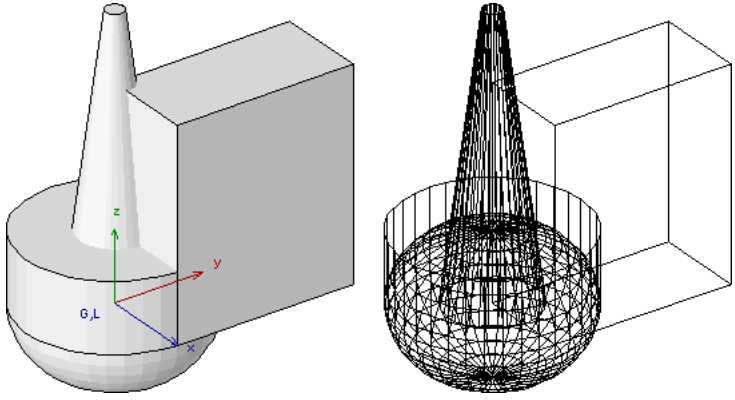
Open the 3D Script and try writing `BLOCK 1,2,3`. Look at the result in the 3D window – a tall cuboid. Now try writing, `CYLIND 1,2`. A short but very fat cylinder appears at the near left corner of the cuboid. Finally, try writing `SPHERE 1.5` – a large ball appears at the same corner, partially submerged in the fat cylinder. Finally, try the `CONE`. All these objects have been built at the same location, called the Global Origin. Even at this stage you will already want to improve the appearance of the object. It is all the same color, depending on the default `PEN`. So above the Block command, write `PEN 1`, and write `MATERIAL 'Whitewash'` (or use the name of a material that is known to you in your materials library).

```
PEN 1
MATERIAL 'Whitewash'
BLOCK 1.0, 2.0, 3.0
CYLIND 1.0, 1.0
SPHERE 1.0
CONE 4, 0.5, 0.1, 90, 90
```

The dimensions you are writing are in meters. The native dimensioning system of GDL is always meters, no matter what you set your main project to be using. In your project file, you could be working in feet and fractional inches, centimeters, millimeters. But GDL is 'hard coded' so you need to work to one dimensional system. If you wish to use feet or inches, you can, but you must write them with punctuation, e.g., `BLOCK 1″,2″,3′-0″`. It reduces a risk of confusion (for humans) if you write dimensions with a decimal form, as `2.0`, not as `2`.

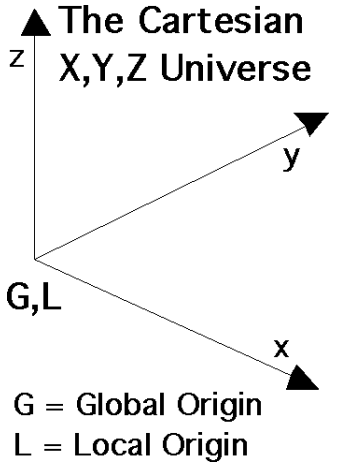
Click in the 3D window (of the library part, not the project) and you will see this. These objects all exist in the same space and overlap each other, so we need to find a way to separate them. The wireframe view shows each of the objects and the shaded view shows you how well GDL permits objects to 'collide' like this without harm.

If, when you click the 3D window, it refuses to draw because of an error, return to your script and click on the Check Script button. This will point to the line where the error occurred, usually because of an incorrectly placed comma, or a typo.



3D Space and the 3D Cursor

You need to move about in 3D space to be able to place elements such as the legs of a chair, or a tabletop. This is done with `ADD` commands. These work on the logical idea that the 3D world can be thought of in three rectilinear directions called X, Y and Z. You should be familiar with the idea of X, Y and Z from the normal ArchiCAD environment – the Coordinate Box allows you to enter a dimension in X or Y, and you can also enter radial dimensions and angles. In GDL, Z is used to provide vertical movement.



The three-pointed object that you can see in a GDL 3D view reminds you which ways are X, Y, or Z. This is like a 3D Cursor. When you word process, you are used to the idea of a cursor – wherever you place the cursor, you can begin typing, and whatever you type appears. In GDL, there are two cursor symbols. The one labelled 'G' is the Global cursor and shows you the origin of the model. The one labelled 'L' is the Local Cursor. Wherever this is, your next 3D command will happen. Using the ADD command, you can move the cursor about in 3D space. ADDX, ADDY, or ADDZ move the cursor in those axial directions. The ADD command moves the cursor in three axes at once.

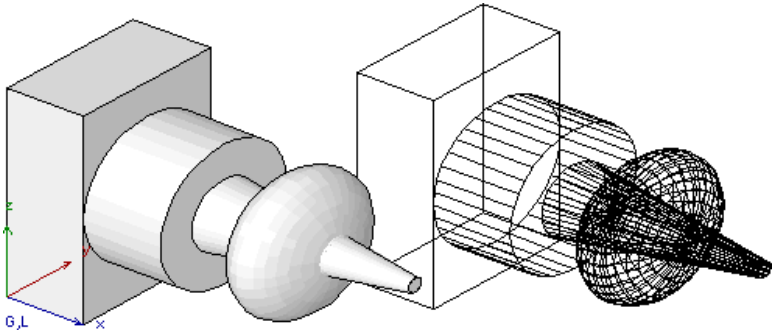
When you are writing GDL in 2D, you have a similar concept of X and Y based 2D space, and you do not have to worry about height. Using the ADD2 command, you can move about in 2D space providing you specify both the X and Y distances. Unfortunately, you do not have a visible cursor.

In reality, the 3D world is too complex to be defined ONLY in absolute terms of X, Y and Z. So you will be glad to know that you are also able to Rotate the cursor (using the ROT command) and move radially, and work at different angles from the pure horizontal and vertical. You are also able to shrink or expand the cursor (using the MUL command which multiplies) so you can easily deform or mirror elements of your model.

A few simple examples will illustrate how all of these work.

Take the previous script and play with it.

```
!Cursor Movement
PEN 1
MATERIAL 'Whitewash'
BLOCK 1.0, 2.0, 3.0
 ADD 1.0,1.0,1.5
 ROTy 90
CYLIND 1.0,1.0
 ADDz 2.0
 MULz 0.5
SPHERE 1.0
 ADDz -2.0
CONE 5, 0.5, 0.1, 90,90
 DEL 5
```



In this example, we have started by adding a Label. Using the exclamation mark, you can type anything you like after this and the machine will ignore it. But it is most useful to the human eye and brain. It is always good to label your scripts with a name and a date. It is also a good discipline to write GDL commands in upper case and to write labels and other text in lower case.

The Block is built as in the previous example. The cursor moves in the X direction, using the `ADDX` command. Write this as one word '`ADDX`' followed by the distance. Then using the '`ROTY`' command, the cursor can be rotated around the Y axis. Now the Cylinder gets drawn – although cylinders always grow vertically, the vertical axis is now laying on its side, so the cylinder grows sideways. Now you can `ADDZ` which raises the cursor vertically in its new orientation. We want to draw the Sphere, but as a bit of fun, use a `MUL` command to halve its height and make it elliptical. `MULZ 0.5` makes it half as high, but leaves width and depth unchanged.

The importance of DEL

Finally, it is a good discipline to return the cursor back to the origin. GDL remembers every cursor move you have made. The `DEL` command undoes or 'forgets' these movements, allowing the cursor to retreat back to the origin, leaving the 3D elements behind. As we have done four cursor commands, the `DEL 4` will neatly return the cursor to the origin.

You may be familiar with the difference between Absolute distance (distance from the Origin) and Relative distance (distance from where you are now to the next position). In GDL, you navigate around the 3D model making Relative jumps. Because you can also Rotate and Multiply, you are in danger of getting lost – after a few lines you could be 'deep in spaghetti'. After each set of moves, you should return to the origin using `DEL` before doing the next job. For example, in a chair model, you could build the legs, then return to the origin before starting work on the seat.

Do not forget the 2D script

Finally, when you save the GDL model, and bring it into the project floor plan, you will be dismayed to find that it does not display – it has no symbol, and at best you will have nothing more than a hotspot. So for the present moment, open the object again, open the 2D script window and write a single line:

```
PROJECT2 3,270,2
```

This is a 'killer' command that in one line, displays the 3D object in 2D, regardless of its complexity. Later, when we get onto 2D scripting, the `PROJECT2` command can be explained in more detail, along with other 2D GDL commands.

Summary of GDL, to this point

– Start with the easy 3D commands.
– Use UPPER case for commands, lower case everything else.
– Write (!)Labels to define a piece of work.
– Write GDL commands in UPPER case, and everything else in lower case.
– Set a Material and Pen color.
– Use Check Script to see if errors occur.
– Be careful with commas and spelling.
– Click in the 3D window to see the object build.
– Do one bit, return the cursor and then do next bit; and so on.
– Do not forget to make a short 2D script.

3.2 Let's build a 3D Object

From ArchiCAD, choose File>New Library Part>Object. We are going to build a simple chair. Remember that this tutorial is about GDL, not chair design, so please do not be concerned that it is a simple chair. It needs to be.

Start in the 3D Script with a BLOCK command for one leg of the chair. Let the chair have a general size of 50 mm (2") for legs and other thicknesses. Later we can introduce subtleties of form and size. Starting from the left-hand front corner, use the ADD commands and work your way around the chair, planting legs as you go.

```
!Legs
MATERIAL 'Pine': PEN 1
BLOCK 0.05,0.05,0.45
 ADDx 0.4
BLOCK 0.05,0.05,0.45
 ADDy 0.4
BLOCK 0.05,0.05,0.45
 ADDx -0.4
BLOCK 0.05,0.05,0.45
 DEL 3
```

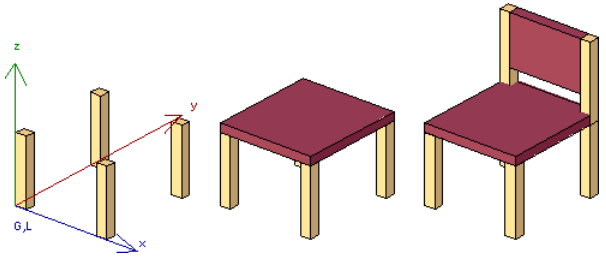
Notice that it is wise, even for such a simple object, to write a label, to define the material and pen color. These two can be included on the same line using a colon to separate the commands. You may not have the Material names used here, so put in the names of materials that you know to be loaded.

The operation to make the legs involves three ADD commands, so we can use DEL 3 to bring the cursor back to the origin. Then rest.

Now for the Seat. This should be of a different material so start with the label and a new material statement. Lift the cursor and draw the block. Note that the block has to be high and big enough to cover the legs. When you use quote marks, it does not matter if they are double or single quotes as long as you are consistent.

```
!Seat
MATERIAL 'Textile'
 ADDz 0.45
BLOCK 0.45,0.45,0.05
 DEL 1
```

Now for the Back. Note that we are working with BLOCK on this chair. As your confidence increases, you will expand the range of GDL commands you use to achieve form. There is usually more than one way to make any shape.




```
!Back
MATERIAL 'Pine'
 ADD 0,0.4,0.45
BLOCK 0.05,0.05,0.45
   ADDx 0.4
   BLOCK 0.05,0.05,0.45
   DEL 1
MATERIAL 'Textile'
   ADD 0.05,0,0.15
   BLOCK 0.35,0.05,0.30
   DEL 1
DEL 1
```

To complete this, you will want to see this in your floor plan. You must write a 2D Script if it is to be visible. Open the 2D script window and write the single line `'PROJECT2 3,270,2'`. Close the object, return to the floor plan. Click the Object tool in the Toolbox, click in the plan, and your chair will appear, with its own hotspots provided by ArchiCAD.

The chair may not win a design award, but it is your first GDL object – so reward yourself and then try another task.

Let's make this chair parametric

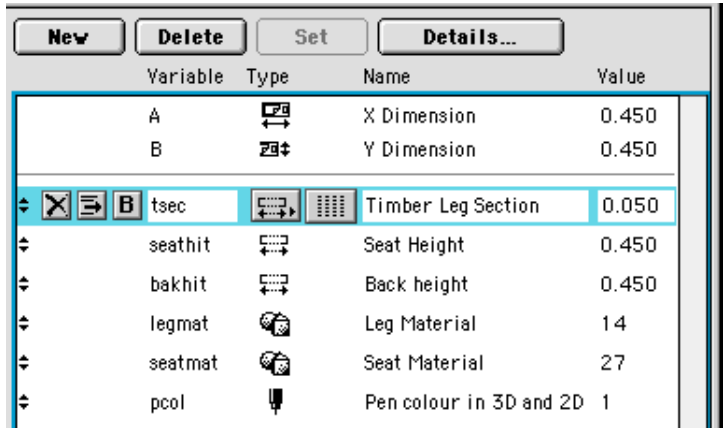
If you stretch a non parametric object, it deforms proportionally, but unrealistically. Square table legs become rectangular, cylindrical legs become elliptical. If you stretch a parametric object, it retains important properties such as diameters and thicknesses. The primary reason for using GDL has to be to enable objects to be parametric. So this simple chair needs to be upgraded.

First make some parameters

Click on the chair object and open it again for editing. In the parameter building box, click the 'New' button 5 or 6 times to make new lines in the building box. They are lettered C, D, E, F, and so on, and they always start with a value of zero. Click in the

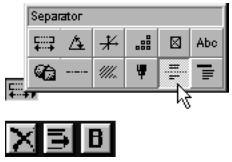
parameter name box of the first one and write in a name for the leg thickness parameter. Call it 'tsec' which is a 4-letter shorthand for timber section. Write in the larger field something like 'Timber sectional size' or equivalent. This is what the user will read when they open your object – so make your description a model of clarity. Then enter a starting value, such as 0.05 meters. If your system settings are in another unit system, write in 50 mm, or 2" or whatever suits you.

Add in more parameter names. Try 'seathit' for Seat height, 'legmat' for Leg material and so on. Think about the parameters you might want to change. You will find a button in the same parameter building box that you can pop-up to reveal a palette to tell GDL whether the parameter is a Dimension, a Material, Pen color or whatever. This palette will be covered in more detail as the exercises develop.



Rules for parameter naming

A and B at the top of the parameters box are 'Magic parameters' that govern the width and depth of objects. You do not have to use them. You could use 'wid' and 'dep' along with the other parameters. Let's use A and B here because they will enable the object to be stretchy. Change them both to the equivalent of 45 cm or 18".



The rules for names of parameters are simple. Use more than one letter so that you can remember what the parameter means. Single letters are best used as 'counters' or 'flags' when you get further into GDL. It's important to be able to read your script weeks later and still understand what you meant. 'Legmat' is far more expressive than 'L'. It takes more typing, but that is a small worry compared with the importance of understanding. It's easier to make typing errors with long parameter names, so avoid over-long

names like 'leg_cross_section'. The letters must be sequential (no spaces) but you can join words with an underscore e.g. 'leg_mat'. Parameters are not case sensitive, but as a discipline, stick exclusively to lowercase for parameters, upper case for commands.

The most important rule is that you must not use words that are already GDL commands (or archaic GDL commands). So names like 'block', 'strlen', 'open' and 'nod' cannot be used.

Now convert the 3D script

The next task is to convert the script from dimensions to parameters. Once you have written a few GDL scripts, it becomes second nature to think out the structure of an object in terms of parameters and write directly into the script using parameters. You develop a personal lingo of words like 'len', 'wid', and 'dep' that you can use in all objects. Start by changing all the leg section widths to 'tsec', then gradually change all the others. Notice that, because the block starts from the near-left corner, the distance it has to move to draw each leg is the dimension of the chair less the leg thickness, for example, width(A) minus tsec – and so on. Try it for yourself.

```
!Simple chair, parametric
PEN pcol
!Legs
MATERIAL legmat
   BLOCK tsec,tsec,seathit-tsec
    ADDX A-tsec
   BLOCK tsec,tsec,seathit-tsec
    ADDY B-tsec
   BLOCK tsec,tsec,seathit-tsec
    ADDX -(A-tsec)   !Note the use of brackets
   BLOCK tsec,tsec,seathit-tsec
    DEL 3
!Seat
   MATERIAL seatmat
    ADDZ seathit-tsec
    ADD 0.001,0.001,0 !Nudge upholstery by one pixel
   BLOCK A-0.002,B-0.002,tsec
    DEL 2
!Back
   MATERIAL legmat
    ADD  0, B-tsec, seathit-tsec
   BLOCK tsec,tsec,bakhit
         ADDX A-tsec
         BLOCK tsec,tsec,bakhit
         DEL 1
```

```
MATERIAL seatmat  !Upholstery
      ADD tsec,0,bakhit/3
      BLOCK A-tsec*2,tsec,bakhit*2/3
      DEL 1
DEL 1
```

Evolution of form

As part of the changes, we can improve the 3D appearance. In the simple chair, the rear upholstery of the seat was in exactly the same 3D space as the uprights for the back. If you nudge the upholstery forward and to the right by one 3D pixel (one millimeter) and reduce the seat size by two pixels, the chair will render better in 3D.

The idea of 3D 'systems analysis'

The back is a little more complicated. Previously, we used actual dimensions. Now we have to do a little systems analysis on the chair. It is logical that the upholstery of the back is equal to the width of the chair minus twice the leg thickness. We have to apply simple systems analysis to all 3D objects if parametric scripting is to work at all.

Now you can write the X,Y and Z values as arithmetic expressions. Star (*) means *multiply*, slash (/) means *divide*.

Error checking

As you work your way through the script, keep clicking in the 3D view window of the GDL object to see if the logic of your arithmetical expressions are making sense. If they are not, it may be a bad spelling. A wrongly spelled parameter will result in a value of zero, or could produce an error message. Click on the 'Check Script' button and read the error message – if you get one.

Fixed parameters

We made an assumption that the back upholstery would be 2/3 of the back height. This is what we call a "manufacturer's rule." So you fix this in the script without giving the user a choice. If you wanted it to be variable, you could provide additional parameters to enable the user to control the appearance.

3D is more powerful if it is parametric

You can see that it is possible to write 3D scripts that are 100% based on parameters and not on real dimensions. This is the key to making objects smart and powerful. Once ArchiCAD users have grasped this they realize that there was something useful in all that

algebra they learned so painfully at school. Later with GDL, you may find that your school trigonometry comes in handy, too!

Save the chair object, look at it in the floor plan, view its settings window, and play with it. Change the materials, height and leg thickness. If you have done everything correctly, it should behave parametrically.

3.3 First steps in 2D scripting

Use a script for a more effective 2D symbol

In the first version of the simple chair, you used the command of `PROJECT2 3,270,2` – a 'killer command' that does it all. To achieve this, it has to do a hidden-line 3D drawing of the object as seen in plan view. The result of this can be that your floor plan takes too long to load or regenerate if you have many such objects in the project. So we advise that you always use a 2D script unless the object is too changeable or complex to script. Let's try a small script for this chair.

The `RECT2` command is a 2D command that is similar to `BLOCK`, but it is more flexible as you can specify the X and Y of the opposing corners, e.g., `RECT2 x1,y1,x2,y2`. All 2D commands end in the letter '2'.

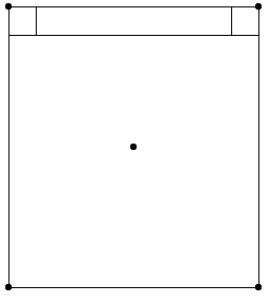
Let's try two ways of doing this chair. We could follow the same sequence as the 3D script, using the `ADD2` command to move around and plant each rectangle. Alternatively, we could stay at the origin and just issue all the `RECT2` commands without having to move anywhere.

The `ADD2` command is a move command in 2D, but requires both X and Y. For an X move, give a value for Y of zero.

```
!Simple chair, parametric
PEN pcol
!PROJECT2 3,270,2   !make this into a comment
RECT2 0,0,A,B !Seat
  ADD2 0,B-tsec
RECT2 0,0,tsec,tsec !Leg
  ADD2 A-tsec,0
RECT2 0,0,tsec,tsec !Leg
  DEL 2
!Back
RECT2 tsec,B-tsec,A-tsec,B
```

The commands for the seat and legs follow the logic of the 3D model. You can use `DEL` to cancel cursor movements, just as in 3D. Unfortunately, you do not have a visible 2D cursor to tell you where it is – you just have to keep a mental track of its location.

The command for the back upholstery sits in one place and issues the `RECT2` command by exact XY locations. We can keep the `PROJECT2` command in the script, but make it inactive by turning it into a comment line.



Save the object and place it on the floor plan. You will find that it generates the symbol instantaneously. It will also have hotspots. You will also find (to your delight?) that the chair is stretchy.

Write your own hotspots

If you wish to guarantee that hotspots are where you want them to be and that the object will be stretchy, you are better off writing them yourself. Open the chair object again, and add the following lines to the script you have written. Place the lines BEFORE the ones that contain the rectangle commands. Hotspots only need the X and Y location.

```
HOTSPOT2 0,0     !4 Corner stretch spots
HOTSPOT2 A,0
HOTSPOT2 A,B
HOTSPOT2 0,B
HOTSPOT2 A/2,B/2  !Pick up spot
```

If you have used ArchiCAD, you will know that hotspots have three main purposes: allowing objects to be picked up; making them stretchy; and giving them 'gravity' (helping them to snap against wall surfaces or each other). Design and place your hotspots as and where you need them.

Later in 2D GDL you can try commands like `CIRCLE2`, `LINE2` and `ARC2`.

3D made more curvy

Before long, you will want to break out of the rectilinear tyranny of `BLOCK` and `RECT2`. The syntax of cylinder (`CYLIND`), `CONE` and `ELBOW` are easy to use. With those three, you can build interesting tubular structures, such as the tubular frames of steel furniture. There isn't space to deal with them in this chapter. Now that you have got this far, you may find it an easy challenge to explore on your own.

Summary of GDL, in this section

– You can write GDL entirely in dimensions.
– It is better to write GDL in parameters.
– Use parameter names that look logical, like 'seatmat'.
– Remember to click the pop-up palette that tells you if a parameter is length, material or pen color.
– Describe the purpose of the parameters clearly.
– Try to make the entire 3D script parametric.
– Click in the 3D view window frequently.
– After each 3D task is achieved, DEL back to the origin before starting the next task.
– Writing a 2D script is easy and saves time in the long term.
– Hotspots are vital for making objects stretchy.
– Try cylinders, cones and elbows for extra quality and authenticity.
– Now, at last, you have a useful application for that algebra you learned at school.

Chapter 4
Practical Uses for GDL

Even in the early dawn of your GDL knowledge you can apply it to useful tasks. It's the best way to learn.

4.1 Practical Uses for GDL

The simple chair was useful for breaking the ice with GDL; let's now try two objects that could be genuinely useful to you in ArchiCAD. These objects will continue to widen your vision as to what is possible with GDL, and will provide you with more rules, techniques and advice on getting the best out of GDL.

GDL in 2D

It is possible to build objects entirely as 2D, but for many building components and furniture items, it is not so necessary. These can usually be built with the 2D tools and saved as objects – as previously demonstrated. In other cases, you can drop a 3D object from the ArchiCAD Library into the floor plan, and 'explode' it – making it a 2D group of lines. However, if you can think of something that offers parametric properties, or you are scripting something that you have already built in 3D, then there is a good reason to learn scripting in 2D.

Let's remind you of some of the common commands in 2D. Cross reference the corresponding section of the GDL Reference Guide.

HOTSPOT2: allows you to post hotspots where you want them to be. Normally GDL will give you 'bounding box' hotspots, but it is far more powerful to place them where they will be useful – to make objects stretchy, to mark out points along a circle, or to make objects easier to pick up.

LINE2 x1,y1, x2,y2: draws a line from point x1,y1 to x2,y2.

RECT2 x1,y1, x2,y2: draws a rectangle from point x1,y1 to x2,y2.

CIRCLE2 x, y, r: draws a circle at point x,y, of radius r.

ARC2 x, y, r, alpha, beta: draws part of a circle on point x,y, of radius r, starting with angle alpha and ending at angle beta – moving in an counterclockwise direction.

POLY2 and **POLY2_** (underscore) are more powerful than any of the above. You can draw any of the above shapes, plus many other shapes, including ones with drilled holes and with a variety of fill patterns.

ADD2, **MUL2**, **ROT2**, **DEL** are concerned with cursor movement, and you have already seen them in use with the simple chair example.

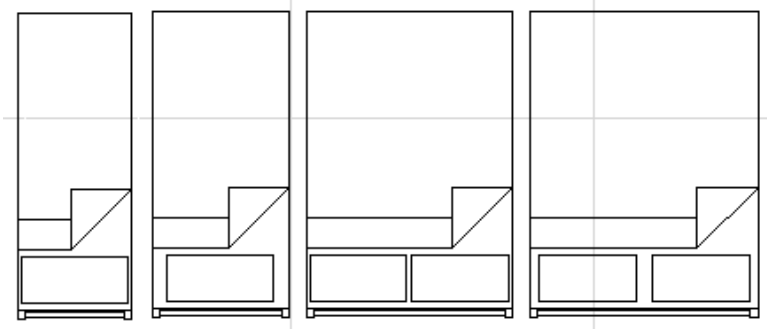
Let's build something in 2D – and make it snappy

From ArchiCAD, choose File>New Library Part>Object. Let us make a 2D Symbol for a bed. The only point in making it in GDL is for it to be parametric and to obey rules – to stretch intelligently, to 'snap' to certain sizes. You can make an object smart by setting up a number of 'IF statements' which help GDL decide what course to take. Some choices can be determined by the user playing with the parameters, other rules would be treated as "manufacturer's rules" and are written ('hard-coded') into the script.

Let us set up some simple rules. Beds, if single are 75 cm or 90 cm wide. If double, they are 135 cm or 150 cm wide. Beds are always 195 cm long. Beds may have headboards. Bedposts are 5 cm square. The beds will be stretchable, but we do not want the beds to be infinitely variable – that would be unrealistic. As each bed enlarges by pulling on its hotspots, we want it to snap to the correct size. 'A' is used as the parameter for width, and we do not need to use 'B' in this case.

In the 2D script, we start with a label and then state one of the main rules – bed length. Then the stretchy hotspots follow. These are always best written BEFORE you write any significant elements of script – because GDL is read in the order in which it is written, the stretchy hotspots tell the object how big to be, and the rest of the script does the job of drawing the object at the required size.

```
!2D Bed Symbol
len=1.95
HOTSPOT2 0,0
HOTSPOT2 A,0
HOTSPOT2 0,len
HOTSPOT2 A,len
```



The object can be 'smart' with IF... ENDIF statements

From the hotspots, the script knows the value of A and can decide what kind of bed it is. The IF statements use the 'greater than' and 'smaller than' expressions to decide on the correct bed width. IF statements can be written on a single line or, as in this script, can be written in the form shown here. The first line ends with the word THEN, followed on the next few lines with the things to be done as a result of the IF question. You round off the IF statement with the ENDIF statement. As long as you go in the right order (each IF statement checking in order of size), you finish up getting the right bed. You see here an example of a multi-statement line. You can use the colon to allow separate statements to exist on the same line (a space saving device).

The idea of Flags

The variable called 'bed' is created and this is a way for the script to remember what sort of bed it is. This variable is called a 'Flag', a shorthand way for GDL to remember which option has been requested. In this example, the first digit of twelve '1' tells GDL that it is a single bed and the second digit '2' tells it that the bed is of the wide variety. The flag is used here to decide on the pillows layout. In more complex objects, flags are vital as a way of remembering the configuration. In a car-model, flags can remember the type of car, left/right hand drive, lights on/off, doors open/shut, seats visible/hidden, passengers present, steering direction of the wheels.

```
   !Calculate widths and flags
   IF A<0.75 THEN
    wid=0.75:bed=11
    ENDIF
   IF A>0.76 THEN
    wid=0.90:bed=12
    ENDIF
   IF A>0.91 THEN
    wid=1.35:bed=21
    ENDIF
   IF A>1.36THEN
    wid=1.50:bed=22
    ENDIF
 HOTSPOT2 wid,0
 HOTSPOT2 wid,len
 HOTSPOT2 wid/2,len/2
```

These last few hotspots are placed at the actual corners of the bed as a result of the IF statements. Next, it is time to draw the actual bed object.

```
RECT2  0,0,  wid,len  !Main Bed shape
!Pillows
IF bed=11 OR bed=12 THEN  !Single
  ADD2  wid/2, 0.2
 RECT2  -0.35,-0.15,  0.35, 0.15 !pillow
  DEL 1
 ENDIF
IF bed=21 OR bed=22 THEN  !Double
  ADD2  wid/4, 0.2
 RECT2  -0.32,-0.15,  0.32, 0.15 !pillow
  ADD2  wid/2, 0
 RECT2  -0.32,-0.15,  0.32, 0.15 !pillow
  DEL 2
 ENDIF
```

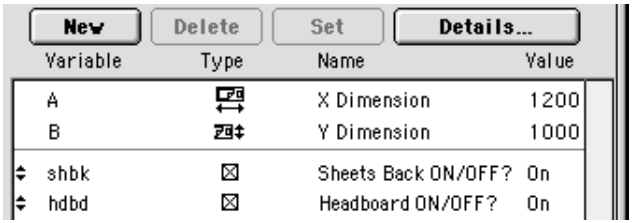
We use the flag (11, 12, 21, 22) to decide how to lay out the pillows. Notice that by using `OR`, we can make the `IF` statement more interesting. We use the parameter of 'wid' to know where to place the pillow. In this case, we have a fixed size for pillows. Check by clicking on the 2D view frequently to see how the script is working.

You will notice that some lines of the script are indented. This makes it easier to read. Two good rules are to indent cursor movements, and to indent the lines following an `IF... THEN` line.

If you haven't saved already, this is a good time to save.

Boolean choices – let the user decide

We could add sophistication and user friendliness such as some turned down sheets and a head board. So create two parameters, and make them of the sort 'Boolean'. This governs the making of a binary choice, an On/Off, Yes/No question. You get a tick box. Tick them in the ON condition. GDL sends the script a result of zero (OFF) or one (ON).



```
!Sheets
IF shbk=1 THEN
POLY2 4,1,        !Poly2 is an alternative
  wid-0.4,0.4,    !to the cluster of LINE
  wid,0.8,        !commands below
  wid-0.4,0.8,
  wid-0.4,0.4
! LINE2 wid-0.4,0.4,wid,0.8
! LINE2 wid-0.4,0.4,wid-0.4,0.8
! LINE2 wid-0.4,0.8,wid,0.8
 LINE2 0,0.4,wid-0.4,0.4
 LINE2 0,0.6,wid-0.4,0.6
ELSE
  LINE2 0,0.4,  wid,0.4
  LINE2 0,0.6,  wid,0.6
ENDIF
!Headboard and Hotspots
IF hdbd THEN
 RECT2 0,0,  0.05,-0.05
 RECT2 wid,0,  wid-0.05,-0.05
 RECT2 0.05,-0.01,wid-0.05,-0.04
 HOTSPOT2 0,-0.05
 HOTSPOT2 wid,-0.05
 ENDIF
```

In the case of the 'shbk' flag, you ask if it equals one. Since these two flags 'shbk' and 'hdbd' can only be 0 or 1 (false or true), you can write in a simpler way as shown for 'hdbd'. '`IF hdbd THEN`' means that anything other than zero will be regarded as true.

The outline of the turned sheet could either be written with a series of `LINE2` statements, or you could try your hand at a `POLY2` statement. You have to write the XY coordinates of the points of the polygon using parameters where possible, and you need to repeat the first point at the end of the XY list. This ensures that the polygon closes up. The `POLY2` statement is followed by the number of points in the list, and a control code of 1 (to draw the lines). A `POLY2` is cleaner code in that it can write many or all of the lines you need in the one command.

If curiosity has not already driven you to try this, save the bed object, place it in the floor plan, and practice stretching it, backwards and forwards. See how it snaps from one size to another. You could elaborate it further, giving the mattress rounded corners (with `ARC2`), adding a footboard, and perhaps

adding a tiny circle at the most important stretchy hotspot to make it easier to pick up. This is like a small joystick or 2D cursor.

```
CIRCLE2 A,0,0.01
```

Note that we have not set Pen color here as the user can do that in the settings box. The bed-object is transparent – you can see other objects below it – but you could try making opaque ones with fill patterns in later 2D scripts for which you certainly need `POLY2`.

Summary of GDL in this section

– Useful 2D objects can be written entirely in GDL.

– The range of 2D GDL commands is small and simple.

– `IF` statements are the first step to making an object 'Smart'.

– Not all rules and choices need be in the user's parameter box. Some rules can be "manufacturer's rules".

– If the answer to a user's question is ON or OFF, then use a Boolean parameter.

– Stretchy Hotspots should be written before the object.

– Flags can be used to store characteristics of the object.

– A Polygon is 'cleaner code' than a mass of lines and allows 2D objects to be opaque.

4.2 Let's make a 3D Door

We made a window earlier using the Slab tool. If this window is stretched, the frame will be distorted. You need GDL to build a window that will stretch but retain correct frame dimensions. With GDL you can also build in smart features like glazing bar options, ironwork, etc. A door is very similar to a window in the way it is made, so let's try one, and make it possible to open and shut the door.

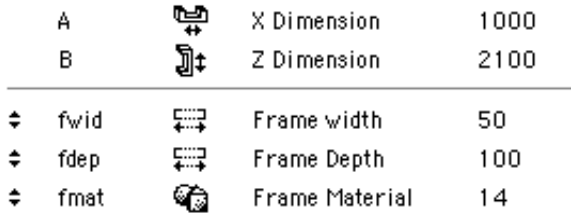
Remember, the primary rule about doors/windows is that they should be built flat on the floor, and that the floor surface – the XY plane – represents the external face of the wall that the window/door will fit into. 'A' represents the width of the door and 'B' will be the height of the door – when it is upright in a Wall.

From ArchiCAD, choose File>New Library Part>Door. (Note that you can build the door as an Object first and transform into a Door later.) We will build this door with `BLOCK`, because it is easy to use. We will not bother with rebates at this stage. Once you are more confident with GDL you can add in these refinements. One

thing we will do with this exercise is to make a simple pop-up menu – the key to making GDL objects VERY user friendly.

Build the door frame

First make some parameters for frame width, frame depth and frame material. In this example, the parameter box is working in millimeters. Use whatever suits you, but remember – scripts are 'hard coded'. If they are not parametric, they must be in meters (or in imperial with foot and inch signs). Then type in a simple script for the frame.




```
!Simple Door
!Frame
MATERIAL fmat
  ADDx -A/2
BLOCK fwid,B,fdep   !left frame stile
  DEL 1
  ADDx A/2-fwid
BLOCK fwid,B,fdep   !right frame stile
  DEL 1
  ADD -A/2,B-fwid,0
BLOCK A,fwid,fdep   !door head
  DEL 1
```

If you know precisely what door you want and how big it is, you may be tempted to write it out in dimensions, not parameters. But when you get more committed to the idea of your objects being 'investment objects', you will always prefer to write in parameters than in dimensions.

`BLOCK` is always built from the near left corner, so when you move the 3D cursor, you need to make adjustments, by deducting the frame width. It is easier if you make each part of the frame and use `DEL` to get back to the origin before doing the next part.

Now make the door

Now you can add in parameters for the door – thickness, material, and opening angles. It is useful to have 2D and 3D opening angles independent.


```
!Door
IF dang3<0  THEN dang3=0
IF dang3>90 THEN dang3=90
LET dwid=A-fwid*2  !Door width
LET dhit=B-fwid    !Door height
MATERIAL dmat
  ADDx dwid/2      !Move to right hand frame
  ROTy 180-dang3   !Rotate door
  ADDz -dthk       !Settle it into rebate
BLOCK dwid,dhit,dthk
  DEL 3
```

It is a good idea to keep the BLOCK command simple by defining the value of Door width and Door height beforehand. This is not important for the simple flush door – but it makes life much easier if you want to make a panelled door later.

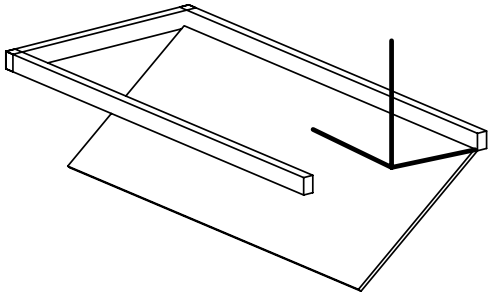
The use of the LET command is voluntary. If you write 'dwid=A-fwid*2' that is not a matter of fact – it is a Command for 'dwid' to have a new value. By putting LET in front of it, you are making it plain that you are issuing a command. The GDL interpreter is not worried whether you use LET or not – but it is easier for the human reader to understand it. If you were to read the script aloud, you would pronounce the equal sign (=) as 'become', as in 'LET dhit become B-fwid'.

The door needs to hinge off one frame. It is a house rule in GDL that doors hinge off the right-hand frame unless they are mirrored. For the door to hinge correctly, you must move the 3D cursor to the right hand frame and build the door from that point. This is much more complicated than doing it from the left frame because the BLOCK command always grows in a rightward direction. To hang correctly, this door needs to appear to be growing in a leftward direction. Therefore, rotate it by '-dang3', and it will grow out through the frame in diametrically the wrong direction. Now add 180° to the angle so it becomes a rotation of '180-dang3' and it

finishes in the right direction. You have to learn little tricks like this when using GDL. Play with the angles to understand this process. You need one small `ADDZ` to let the door 'sink' into the rebate.

When the door is in a wall, you will be able to use the Mirror or Rotate function to change the way it will turn. In a later chapter, we will make elements using `PRISM`, and you would not have to do such mental gymnastics with `PRISM` – because you can make it grow in a leftward direction from the hinging edge.

It is the `IF` statement that allows GDL to be smart, so for a door, the first smart thing we can do is to stop the door being turned so that the hinge would break. GDL models do not have gravity or collision protection, so some error correcting `IF` statements will save your door from 'breaking'. Below is the door in 3D, half open. It is still on the ground, but will be upright when placed in a wall.



The Door needs a 2D symbol

Before you can place it in the ArchiCAD environment, you need to give the door a 2D symbol. Firstly, it is easiest to use the `PROJECT2` command to generate the symbol. The syntax is `PROJECT2 view_type, camera_angle, drawing_method`.

Remember that `PROJECT2` tells GDL to draw in 2D whatever it finds in 3D. Please check in the corresponding chapter of the GDL Reference Guide and you will see that `PROJECT2` is remarkably versatile, allowing you to choose a viewtype that sees the object in a huge variety of angles – plan, elevation and various forms of axonometric. The camera angle is normally 270° for objects, and 90° for door/window objects. The drawing method is either wireline(1) or hidden line analytic(2).

The `PROJECT2` we used for the simple chair drew the chair in Plan. For a door or window we need to draw a view of the door as if the camera was placed at 90 degrees and viewing the doorset in

Elevation, looking at the head of the window. It also needs to view the doorhead so that the external wall is the top of the drawing.

```
!Simple Doorset 2D script
ROT2 180      !turns camera the right way up
PROJECT2 4,90,1 !views side view from 90˚
DEL 1         !back to origin
```

The ROT2 command turns the camera the right way up. Then the PROJECT2 command draws the doorset, and by doing it in wireframe, the frame is clearly visible.

Make sure the door works

If you haven't already saved, this is the time to save. The door will be saved as a .DOR file. If you built the door as a .GSM object file, go to File>OpenLibrary Part>Open Object as Door, and then save; it will now be a door.

Return to the ArchiCAD project floor plan, draw a wall. Click on the Door tool and place your new door. The one you have just saved will be the first door that gets placed. Check that it works in 3D view. Try altering its settings. You can move the frame in and out of the opening. You can open and shut the door. You will find that only the 3D doorswing is working. We haven't written the 2D yet.

You may also wonder how the hole in the wall is drilled. GDL cuts a rectangular hole in the wall, 'A' in width and 'B' in height. When you get further with GDL, you will find out how to cut holes of more interesting shapes using the WALLHOLE command.

Improve the 2D performance of the door

With PROJECT2, the 2D symbol can only show what the 3D object is doing, and for more powerful work, we would like to show the doorswing differently between 2D and 3D. In plan drawing, architects usually draw a doorswing in the open position, but in 3D modelling, it is often desired to leave the door shut.

```
!Simple Door
!ROT2 180         !these commands disabled
!PROJECT2 4,90,1  !but retained as comments
!DEL 1
!Frame
 ADD2 -A/2,0
RECT2 0,0,fwid,-fdep  !left stile
 DEL 1
 ADD2 A/2,0
RECT2 0,0,-fwid,-fdep !right stile
 DEL 1
```

```
!Door
IF dang2<0 THEN dang2=0
IF dang2>90 THEN dang2=90
 dwid=A-fwid*2 !Door width
 ADD2 dwid/2,0
 ROT2 180-dang2
RECT2 0,0,dwid,dthk
ARC2 0,0,dwid,0,dang2
 DEL 2
```

Whenever you can, use the same algorithm as in the 3D script – it may make the script longer, but it's easier to write. Start with the frames, and finish with the script for the door – using the same little 180° trick used to get the door to swing the right way.

Because we want to be sure this is correct, we keep the `PROJECT2` command working and visible. Indeed, if you write in a temporary Pen color (e.g., `PEN 1` in front of the `PROJECT2` and `PEN 10` in front of the rest of the script), the different pen colors will show the 2D symbol building up. Because we are looking at the door from the 90° camera angle, the frames end up in the 'negative-Y' side. So, most Y values in the script are minus. Where you had `BLOCK`, you replace it with `RECT2`. Where you had `ADD`, you replace it with `ADD2`. Where you had 'dang3' you replace with 'dang2'.

When you have finished working, you can convert the `PROJECT2` routine into a series of comments. Remove the `PEN` commands. Leave these to the user to supply in the settings box of the door.

Save the Door object, return to the floor plan and admire your handiwork. You should now be able to set different opening angles for 2D and 3D.

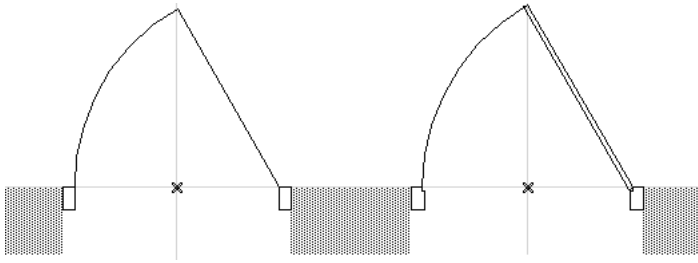
Scale Sensitive 2D symbol – with Global Variables

Architects also like 2D objects to be scale sensitive. At 1:100, the door may be a line, and the frame a simple shape, at 1:20, the door should be shown in detail, with a rebated doorframe. It is possible for the GDL object to know the current scale of the drawing and to change accordingly.

ArchiCAD keeps an active memory of everything that is going on in the current project. GDL can interrogate this memory and make use of it. The current Drawing Scale, the current Northpoint, the object's position in the Floor plan, the current Story, the current Wall thickness, etc. These 'Global Variables' are described in Appendix A of the GDL Reference Manual.

Let's just consider drawing scale. This is called GLOB_SCALE or A_ (A-underscore) for short. If the drawing is 1:100, A_ will be 100. If it is 1/8 inch to 1 foot, it will be 96. So we can design a better shape for the doorframe that will look detailed at 1:50 or larger. Let's assume that the rebate is 12 mm. You can now write a polygon to draw a realistic looking door frame. Polygons (and Prisms in 3D) are hard work, but if you sketch them out first, and think in terms of parameters, it is not so difficult.

```
!Simple Door - 2D script
!ROT2 180          !these commands disabled
!PROJECT2 4,90,1   !but retained as comments
!DEL 1
!Frame
 ADD2 -A/2,0
 LET rb=0.012 !Rebate
IF GLOB_SCALE>51 THEN
 RECT2 0,0,fwid,-fdep !left stile simple
 ELSE
 POLY2 7,5,           !left stile complex
  0,0,
  0,-fdep,
  fwid+rb,-fdep,
  fwid+rb,-dthk,
  fwid,-dthk,
  fwid,0,
  0,0
 ENDIF
 DEL 1
 ADD2 A/2,0
IF GLOB_SCALE>51 THEN
 RECT2 0,0,-fwid,-fdep !right stile simple
 ELSE
 POLY2 7,5,            !right stile complex
  0,0,
  0,-fdep,
  -fwid-rb,-fdep,
  -fwid-rb,-dthk,
  -fwid,-dthk,
  -fwid,0,
  0,0
 ENDIF
 DEL 1
```



As you work through this, have the 2D View window open, and keep checking that the frames are turned the right way. You can change the current scale by activating the scale button at the bottom left of the 2D View window. You can now finish by providing an alternative symbol for the door itself – the rectangle can now be used for 1:50 or larger, and a simple line for smaller drawing scales.

```
!Door
IF dang2<0 THEN dang2=0
IF dang2>90 THEN dang2=90
 dwid=A-fwid*2  !Door width
 ADD2 dwid/2,0  !Move to right hand frame
 ROT2 180-dang2 !Hinge the door
IF GLOB_SCALE>51 THEN
 LINE2 0,0,dwid,0     !Simple line
 ELSE
 RECT2 0,0,dwid,dthk  !Rectangle
 ENDIF
ARC2 0,0,dwid,0,dang2 !Door swing
 DEL 2
```

Master Script is useful here

Notice again that you have to tell it again what 'dwid' (door width) is. This is because 2D and 3D scripts do not talk to each other. If you placed values for 'rb' (rebate), 'dwid' (door width) and the error correcting routines into the Master Script, this could save you some typing, centralize some of your fixed parameters, and inform both the 2D and the 3D scripts of the correct values.

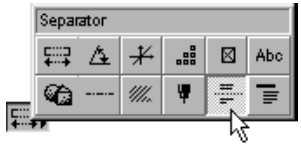
Build a pop-up menu with the VALUES command

For the 3D object door object, you would like to have a choice of door styles, but how does one ask the user? In the days of ArchiCAD 5.0, one had to have style 1, style 2, etc. Now you can create a pop-up menu that allows the user to choose by name. You can even have a visual display that shows each door as a

picture. You must have noticed this in the new Door objects in the ArchiCAD 6.5 door library.

We can make a start here by offering a simple choice of a solid or a panelled door.

First make a new parameter. Let's call it 'doortyp'. As the pop-up menu will offer a choice of door styles in words, select the parameter type 'Abc' which will be for text. Make a new parameter for panelling material, called 'panmat'.

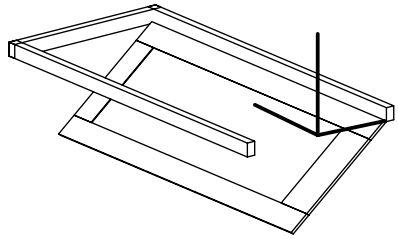


Now go to the Parameters Script. This is where you build pop-up menus. Type in a VALUES command, starting with the name of the parameter (in quotes), followed by a list of the menu choices (also in quotes). Separate the choices by commas.

```
VALUES 'doortyp' 'Flush Door', 'Panelled Door'
```

It may take a few seconds for GDL to notice, but after a while, 'doortyp' will realize that it has a pop-up menu and will allow you to choose one or the other. The little triangular symbol will appear at the right of the parameter entry. As the choice of door style is one of the most important parameters of the door, you could click on the 'Bold' button to make the font heavier, and you could use the little arrows at the left to push the parameter to the top of the list. Then you should make a parameter for Panelling Material.

This pop-up menu brings two major benefits – the object is more user friendly and it is impossible for the user to make a spelling error or choose a silly option. Before we get too confident, we have to put this pop-up menu to good use. Return to the 3D script and make the choice work.



You only need a simple IF statement to help the 3D Script decide which door to display. Let's take the original 3D text for the door and add in the IF-statements and build a script for a panelled door.

```
!Door - 3D Script
IF dang3<0 THEN dang3=0
IF dang3>90 THEN dang3=90
dwid=A-fwid*2  !Door width
dhit=B-fwid    !Door height
MATERIAL dmat
  ADDx dwid/2     !Move to right hand frame
  ROTy 180-dang3  !Rotate door
  ADDz -dthk      !Settle it into rebate
IF doortyp='Flush Door' THEN
 BLOCK dwid,dhit,dthk
 ENDIF
IF doortyp='Panelled Door' THEN
 BLOCK dwid/6,dhit,dthk !Left
 ADDx dwid*5/6
 BLOCK dwid/6,dhit,dthk !Right
 DEL 1
 ADD dwid/6,dhit-dwid/6,0
 BLOCK dwid*4/6,dwid/6,dthk !Toprail
 DEL 1
 ADDx dwid/6
 BLOCK dwid*4/6,dwid/4,dthk !Footrail
 DEL 1
MATERIAL panmat
 ADD dwid/6,dwid/4,dthk/4
 BLOCK dwid*4/6,dhit-dwid/4-dwid/6,dthk/2
 DEL 1
 ENDIF
  DEL 3
```

If you are sharp eyed, you will look at the script above and see that we could simply have used IF... ELSE... ENDIF, because we only have a choice of two styles. And you are right. This is a simple door; but when you build doors for actual use, you may wish to have five or six styles. Some of the door style choices in the ArchiCAD library run well into double figures. It is easier to have separate IF-statements for each choice.

You may also wonder how the script arrived at dividing the door width into sixths, to arrive at the sizes of the vertical door sections, compared with the panel size. This is another example of

Manufacturer's rules. One manufacturer may have several ways of proportioning their doors. You may need a style for each one.

Finally, a door is incomplete without its ironwork, but that is beyond the scope of this chapter. It could be scripted in with the door, or if you had a set of door handles already built, it could be brought in as another object, using the `CALL` command, and placed in the correct position on the door.

A first look at Property Scripting

When you do a listing of your building, an easy one line command will help you. The `DESCRIPTOR` command describes your object. Open the Properties script and try this easy script, using the parameter 'doortyp', which was the user's choice of door.

```
DESCRIPTOR doortyp
```

	Descriptor Key Name	Descriptor Code	Descriptor Short Text
1			Panelled Door

Let's finish the door

If you are serious about doors, you can see that there are many more things you could do. You could enter the manufacturer's serial codes for door type, introduce many variations on the panelling solution, offer choices of ironwork, specify fire resistance, kickplates or letter boxes and more.

If you want to build double doors or bifold doors, it would be possible to build these into the same object – but then you might make it less user friendly if it becomes excessively complex. It is better to Save As... the door under a new name. Save as a double door, then using the same frame details, provide the modified script for double doors – not forgetting that you need to update the 2D script, too.

Preview window – get yourself an icon

You must have noticed that most ArchiCAD Library parts have nice pictorial icons – how do you get these for yourself? Set up a nice 3D view of the object you have made, and photo-render this, preferably into a window sized 128x128 pixels. You can make it a bit larger, but keep it square. Select the image and Copy; from the Library part parameter building box, click on the button marked Preview Picture. Paste the image into that. Save and Close it. Now reload the object, and you will have a nice preview picture for the object, as well as an icon for it.

Summary of GDL in this section

– Doors and Windows must be built on the XY ground plane.
– The Origin should be at the center, lower sill or threshold.
– Hinging the door is not difficult – build the door from the hinging point.
– Build error-correcting statements into the script.
– The 2D symbol can be made with `PROJECT2`, in Side View. But a truly scripted 2D symbol may be more powerful.
– 2D symbols can adapt to the scale of the drawing if it is scripted.
– Global Variables – very interesting! Look at Appendix A of the GDl Reference Manual.
– `PRISM` is more powerful than `BLOCK`.
– Master Script can hold information and perform error correcting routines for both 2D and 3D.
– Pop-up menus are easy to build, user-friendly and reduce errors.
– Even simple objects can have a short Properties script.
– Make a Preview picture.

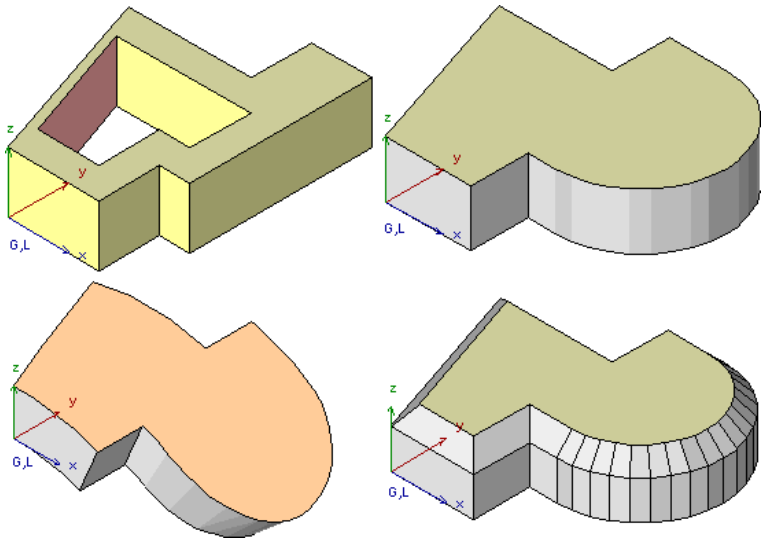
Chapter 5
The Power of PRISM

We discover a 3D command that adds real power to GDL.

5.1 The Power of PRISM

PRISM is the most versatile element in 3D GDL

We have found that BLOCK is acceptable for building elementary furniture and frames, but you rapidly discover that BLOCK is too primitive for serious work. Apart from its rigid shape, you have to move the cursor to the location of the block, and the block always grows from the near left corner. Many objects have recognizable shapes or curves in them that require you to draw their outline shape correctly. PRISM is the answer.



Prisms can be multi-colored, bent, rounded, cut, drilled, chamfered and hollowed out. They are quite useful.

You can define the outline of a prism by listing the XY locations of its nodepoints. Although 0,0 might be one of its points, this is not essential. Your cursor can remain at the origin and your script can calmly draw prisms that are hundreds of meters away because it is entering the precise XY locations of each node.

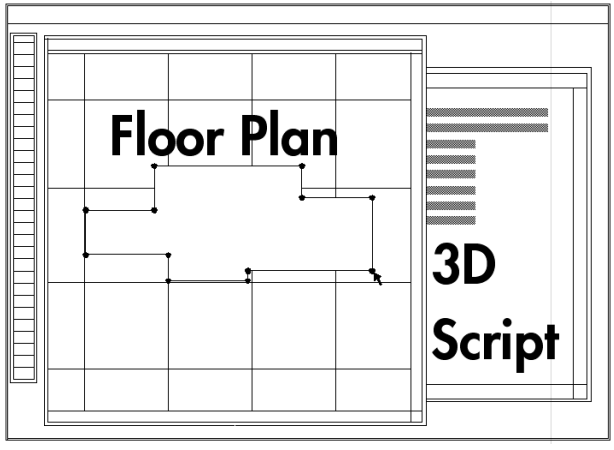
The code for many prisms can also be 'stolen' from ArchiCAD. Try this next idea.

This prism has been drawn out with the Slab tool in the ArchiCAD floor plan. To illustrate how prisms work, the nodes have been numbered in the order in which they were clicked. There are 13 points if you include the first point twice – the idea being that you fully close the prism by returning to the start.

Now draw out a prism yourself of roughly the same shape (no need to number the nodes). As a regular discipline, try to draw slabs in the counterclockwise direction, because GDL is happier working that way. Mathematically, counterclockwise is a 'positive' direction if you start trying to curve edges. Draw it so that one of the points coincides with zero – the global origin.

Instant GDL

Everything in ArchiCAD is in fact GDL (under the surface). You can click on anything in an ArchiCAD floor plan – wall, roof, floor, mesh, even library part – and find out what its GDL code is by dragging and dropping it into a GDL script window. 3D objects should be dragged into a 3D script window, and 2D into 2D.



We can transform this slab to GDL immediately. From the File menu, choose New Library part>Object, and arrange the 3D script window in a narrow column on the right of the screen. Rearrange the project floor plan so that it is occupying 2/3 of the screen and is to the left, and overlapping the 3D script window slightly. Now click on the slab and, holding it, simply Drag it across and Drop it into the 3D script window. There is a slight knack to this, so try it a few times if it does not work the first time.

```
BODY  -1
MODEL SOLID
RESOL    36
!!Slab-018
PEN    1
ADDZ    4.1
BODY  -1
cPRISM_ "Whitewash", "Whitewash", "Whitewash",
       13,   0.1,
       0.0,   0.0,   15,
       4.28548,   0.0,   15,
       4.28548,   0.949196,    15,
       5.279549,   0.949196,   15,
       5.279549,   2.155116,   15,
       3.405484,   2.155116,   15,
       3.405484,   3.067704,   15,
      -0.000425,   3.067704,   15,
      -0.000425,   2.00845,    15,
      -2.444858,   2.00845,    15,
      -2.444858,   0.949196,   15,
      0.0,   0.949196,   15,
      0.0,   0.0,   -1
BODY  -1
DEL    1
```

This resembles what you will get in the 3D script window. Instant GDL! All Slab tool objects are converted into '`CPRISM_`'; this is a prism in which the top, bottom and side surfaces are specified by name or number. The next line contains the number of nodes and the thickness, and then follows the list of XY locations. Each one ends in the number 15, and the final node ends in a -1. The 15 is a 'masking code' and simply tells the slab to be drawn as a Solid and tells all the edge lines to be drawn. The -1 tells GDL that the slab is finished (and also that the end point is the same as the start point). This is the default, so leave these as they are unless you want to omit nodepoints.

Note that the slab was started from the main origin of the project, so that the first node listed is 0,0. Note also that the XY locations are accurate to a millionth of a meter. This is a lot of reading for the GDL parser, and if you write creative GDL, you can write in much cleaner numbers.

You can also do 'Instant GDL' in 2D

Try drawing a polygon with the Fill tool. Drag and Drop the result into a 2D Script window – another example of instant GDL!

About Prisms...

Prisms come in several forms.

```
PRISM number of nodes, thickness,
X1,Y1,..... Xn,Yn
```

For a simple prism, you do not need materials or the underscore, and you do not even need to close the prism. The default is for it to be a closed solid. By adding an underscore and 15 at the end of each line, you get much the same thing, but with added potential.

```
PRISM_ number of nodes, thickness,
X1,Y1,mask,..... Xn,Yn,mask, X1,Y1,-1
```

Normally, the value of the masking code is always 15. The power of masking in the underscore version of PRISM is that you can omit faces and lines of the prism and you can curve corners, and drill holes through the prism. The curved corners are made using the technique of the 'Polyline'. It is too early to lead you through all these possibilities now, but you need to know that they are available.

```
CPRISM_ topmat,botmat,sidemat,
number of nodes, thickness,
X1,Y1,mask,..... Xn,Yn,mask, X1,Y1,-1
```

`CPRISM_` is what you get when you make 'instant GDL' with the Slab tool. This has all the powers of `PRISM_` but it also allows you to change the colors of the top, bottom and side faces.

You also have the following types of prism which are suitable for later study:

- `BPRISM_` is an extension of `CPRISM_` but allows you to bend the prism by adding a radius of curvature. It bends downwards, but if you precede it with a mirroring routine (`MULZ -1`), it will bend upwards.
- `FPRISM` is an extension of `CPRISM_` but allows you to have curved or chamfered top edges (hills) by specifying the angle, height and material of the hill. This is useful for suggesting softness, e.g., in furniture. However, it creates a lot of polygons and lines in 3D view.
- `SPRISM_` is an extension of `CPRISM_` but allows you to have a cut plane taken through the prism. As it is possible to use an actual command `CUTPLANE`, this version of the Prism is not so frequently used.

Clean up the Prism

Return to the cPrism we dragged and dropped earlier. If you just want a pure prism, clean away the statements before and after the prism. The function of these is explained more in chapter 6. If you find an ADDZ command like the one in this example, it is a leftover from a previous Slab tool setting, so you can delete that, along with DEL. You are left with a simple CPRISM_ command:

```
CPRISM_ "Whitewash", "Whitewash", "Whitewash",
  13, 0.1,
  0.0, 0.0, 15,
  4.28548, 0.0, 15,
  4.28548, 0.949196, 15,
  5.279549, 0.949196, 15,
  5.279549, 2.155116, 15,
  3.405484, 2.155116, 15,
  3.405484, 3.067704, 15,
   -0.000425, 3.067704, 15,
   -0.000425, 2.00845, 15,
   -2.444858, 2.00845, 15,
   -2.444858, 0.949196, 15,
  0.0, 0.949196, 15,
  0.0, 0.0, -1
```

If you are doing a very simple solid one-color prism, you do not need all the 15s and the -1, so you can delete those. You can also remove the 'c', the underscore and the material definitions. One single material definition is enough for the whole prism. You can also use the GDL text editor's Find and Replace function to remove some of the surplus spaces to make the code look more compact. So you are left with an even simpler PRISM command:

```
MATERIAL "Whitewash"
PRISM  13, 0.1,
  0.0, 0.0,
  4.28548, 0.0,
  4.28548, 0.949196,
  5.279549, 0.949196,
  5.279549, 2.155116,
  3.405484, 2.155116,
  3.405484, 3.067704,
   -0.000425, 3.067704,
   -0.000425, 2.00845,
   -2.444858, 2.00845,
   -2.444858, 0.949196,
  0.0, 0.949196,
  0.0, 0.0
```

In creative GDL, you have to decide how your model is to be written.

– Do you have a single prism shape that is not going to change much even if the user changes parameters? If so, use this method, always working from the main origin of the project, to ensure that at least one node of the prism is zero.

– If the shape is truly parametric, such as a window or door frame, then you should do a sketch on paper of the prism, number each node in an anti clockwise direction, then start writing the prism methodically using parameters, or significant dimensions where appropriate.

Have a look in the GDL Reference Manual (by now you should find it a lot easier to browse) and read the section on `PRISM`s in the chapter dealing with 3D Shapes. If it is still hard going, you will find it easier after you have been through this chapter.

Comma confusion?

In some countries, it is common to write decimal points as commas, e.g., 1,25. In other countries it is common to write thousands with commas for example, a million is 1,000,000. If you have a localized ArchiCAD, this is how numbers may appear in the parameters box. But in GDL, the scripts are hard coded so only one convention can be followed. Dots are used to form the decimal point, and commas are used to separate numbers. Many of the errors in your early efforts in creative GDL will be with misplaced dots and commas. Pay strict attention to them.

5.2 Let's make a 3D Window

Now that you know about prisms, we can build a window using a prism for the window, drilling a hole through to form the frame, and even allowing the possibility of a curved top to the window. We can also form a sill using a prism.

Remember the basic rule of making a window (this is the third wall-opening-filler in the book now) which is that it is made flat on the XY ground plane, viewed from 90° and saved. Then, when it is brought back into ArchiCAD and placed in a wall, it is the right way up and able to cut a hole. The second rule is that 'A' governs the width and 'B' governs the height – ideally, the origin should be centrally positioned at the sill of the window.

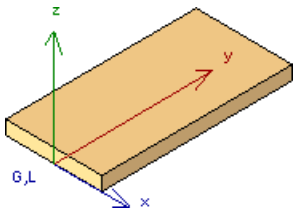
First, create a new window by choosing File>New Library Part>Window. Enter some parameters to get started with. These here are in mm, but you can work your chosen dimension system.

	A		X Dimension	600
	B		Z Dimension	1200
◆	fwid		Frame width	50
◆	fdep		Frame Depth	90
◆	silproj		Sill Projection	100
◆	frammat		Frame Material	14
◆	glasmat		Glass Material	29

The first task is to make a prism for the outline of the window, using A and B. You have to use `PRISM_` (underscore) because you are planning to drill a hole in the prism to form the window frame. Work in a counterclockwise direction. Click in the 3D View window when you have finished. Do not click until you have finished – an incomplete Prism statement will result in an error.

Start with a Label and Material statement.

```
!Window Frame
MATERIAL frammat
PRISM_ 5,fdep,
-A/2,0,15,
 A/2,0,15,
 A/2,B,15,
-A/2,B,15,
-A/2,0,-1
```

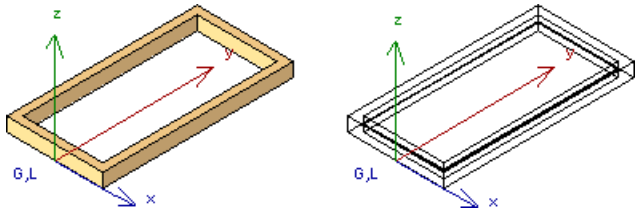


Now you add the XY locations of the hole – based on the frame width. End each XY with a 15. The number of nodepoints will be increased because the number that make up the hole are added to the number for the outline. You need 5 points to form a 4-sided prism. So you will have to add another 5 points for the hole. We recommend that you write the number of nodes as an expression '5+5'. This way, you can always remember how you arrived at the final number. Here, this might seem elementary, but when you get onto more complex prism shapes you will find this a useful tip.

As you work your way round the prism, make appropriate additions and subtractions of the frame width 'fwid'. If you finish with something silly, then you may have put your plus and minus signs in the wrong place. If you get an error you may have

forgotten to keep the commas continuous to the end, or you may have forgotten to change the nodepoints from 5 to 5+5.

```
PRISM_ 5+5,fdep,
-A/2,0,15,
 A/2,0,15,
 A/2,B,15,
-A/2,B,15,
-A/2,0,-1,            !end of outline
-A/2+fwid,0+fwid,15, !start of hole
 A/2-fwid,0+fwid,15,
 A/2-fwid,B-fwid,15,
-A/2+fwid,B-fwid,15,
-A/2-fwid,0+fwid,-1  !end of hole
```



Now you can add the Glass. For a fixed light, this will be the same outline as the inside line of the hole in the frame. Therefore, to build your glass easily, you can simply copy and paste the list of points for the hole. Assume that the glass is at the midpoint of the frame so lift the glass, and assume that the glass is 6 mm thick.

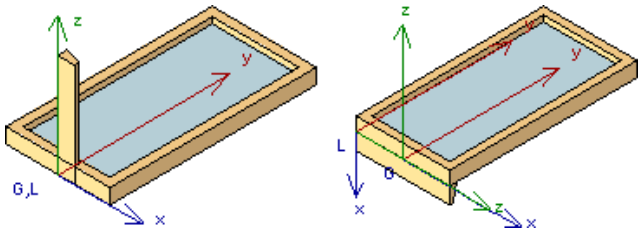
```
!Glass
MATERIAL glasmat
ADDz fdep/2
PRISM_ 5,0.006,
-A/2+fwid,0+fwid,15,
 A/2-fwid,0+fwid,15,
 A/2-fwid,B-fwid,15,
-A/2+fwid,B-fwid,15,
-A/2+fwid,0+fwid,-1
DEL 1
```

We have assumed that the frame is rectangular in section. This is quite adequate for general 3D visuals and general arrangement drawings for a building. If you wanted to include highly detailed information about the frame profile, including rebates, etc., you would have to use another command called TUBE. With TUBE, you define a profile and pull that profile through a series of points – the outline of the window – to form the frame. TUBE is beyond the scope of this book, but when you have more experience with

GDL, try it! For construction drawing purposes, you could put this level of detail into the 2D symbol script, including scale sensitivity.

Add the Sill

This is more difficult. So far, the whole frame has been safely on the ground, and we have only had to define its outline, and it was produced in the right place. Prisms only grow upwards from the XY ground plane. So the sill has to be made on end with the `PRISM` command; then turned over and aligned with the window. Make it at the origin first.




```
!Sill
MATERIAL frammat
PRISM 5,A,
 0,0,
 silproj,0,
 silproj,fwid/2,
 0,fwid,
 0,0
```

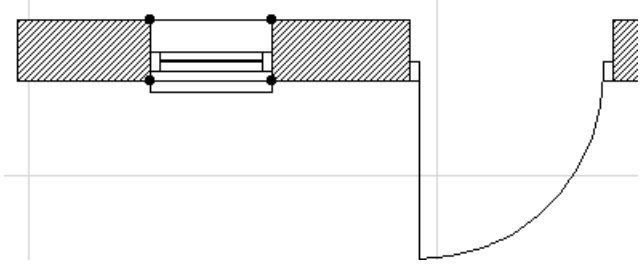
This basic sill sticks up in the air. But you can see that by rotating it 90˚ around Y and recessing it to the corner of the main frame, it will be an easy job. Put the `ROT` and `ADD` commands in front of the Sill prism, and remember to `DEL` afterwards. The right-hand image shows the sill finally positioned, just before the `DEL` command is issued.

```
!Sill
MATERIAL frammat
ROTy 90
ADDz -A/2
PRISM 5,A,
 0,0,
 silproj,0,
 silproj,fwid/2,
 0,fwid,
 0,0
DEL 2
```

Don't forget the 2D Symbol

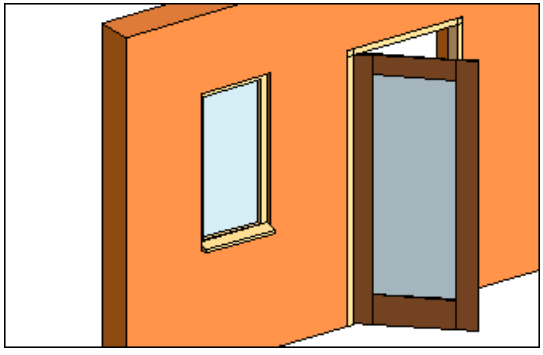
For an easy life, you can use this regular do-it-all script. You might, however, wish for a high performance symbol that has scale sensitivity. In this way, you can keep the 3D object simple, but allow the 2D drawings to display rebated frames and casements. Chapter 4 used the Door example to demonstrate the procedure.

```
!Simple Window 2D script
ROT2 180
PROJECT2 4,90,1
DEL 1
```



The window is seen here with the door from chapter 4

Save the window file, return to the ArchiCAD floor plan, draw a wall and drop this window into it. Place a section line in front of it. Practice stretching the window. Notice that the frame remains the correct size, even though you can stretch the window greatly. If it was a smart window, you could get it to work out when it was wide enough to need intermediate mullions.



You can see at a glance that the use of `PRISM` leads to a clean 3D image, whereas the use of `BLOCK` (as for the door) left a number of untidy lines on the 3D view.

Try WALLHOLE

We all know that GDL windows and doors automatically cut a rectangle based on A (width) and B (height). If this is too simplistic for you, it is possible to dictate exactly what shape the window should cut in the wall, irrespective of the actual window frame that will fit. WALLHOLE is a command that is like a cookie cutter – punches a hole in the wall, yet is easy to use - having the same syntax as PRISM_. In fact, if you take the outline of the window frame, that perfectly defines the list of XY locations for the hole. It is not required for a simple window like this, but it is valuable to know about WALLHOLE so that you are able to provide a complex window shape when you need it.

The major limitation of WALLHOLE is that you cannot have indents in the shape. A 'D' shape is acceptable, but an 'L' shape is not. If you wanted to make a twin arch window, which requires a 'B' shape, you could overlap two 'D'-shaped WALLHOLEs together. Have a look through the Windows and Doors Specials chapter of the GDL Reference Manual for more detailed explanation. Copy and paste the original outline prism command, and then change its label and the title.

```
!Cut the Hole!
WALLHOLE 5,1,
-A/2,0,15,
 A/2,0,15,
 A/2,B,15,
-A/2,B,15,
-A/2,0,-1
```

In place of the Prism thickness, you have a status code. This can be either 0 or 1. If you make it 0, the reveal to the window will adopt the color of the window frame (or whatever was the most recent Material statement.) If you make it 1, the reveal will display the characteristics of the wall.

Give it a curvy window head – Flags and Polylines

This is too early to go into the matter of polylines, but you might like a glance at the future – and decide you like it. If you add a masking value of 1000 to the 15 you can make the PRISM_ outline follow a tangential curve. Let's try this.

First you need to make a small pop-up menu. Make a new parameter 'winshape', and write a short VALUES statement in the Parameter Script.

```
VALUES 'winshape' 'Rectangular','Round-topped'
```

Now, at the top of the 3D script write a short routine to parse the result. This generates a value for a flag, 'ws'.

```
IF winshape='Rectangular'  THEN ws=0
IF winshape='Round-topped' THEN ws=1
```

We introduced the idea of flags earlier, in chapter 4, to remember which sort of bed was required.

Final 3D script – we see the power of WALLHOLE

Let's run through the whole of the 3D script from start to end.

```
!Window Frame

IF winshape='Rectangular'  THEN ws=0
IF winshape='Round-topped' THEN ws=1

!Cut the Hole!
WALLHOLE 5,1,
-A/2,0,15,
 A/2,0,15,
 A/2,B,13,
-A/2,B,13+1000*ws,
-A/2,0,-1
```

Now you see why you need WALLHOLE. When you want a non-rectangular window opening, WALLHOLE is essential. If you add 1000 to the mask value at the END of the curve, it will fly round from the previous location forming a tangential curve. By multiplying 1000 by 'ws' you can switch the curve ON and OFF. Do the same thing to the actual frame.

```
!Window Frame
MATERIAL frammat
PRISM_ 5+5,fdep,
-A/2,0,15,
 A/2,0,15,
 A/2,B,13,
-A/2,B,13+1000*ws,
-A/2,0,-1,

-A/2+fwid,0+fwid,15,
 A/2-fwid,0+fwid,15,
 A/2-fwid,B-fwid+fwid*ws,13,
-A/2+fwid,B-fwid+fwid*ws,13+1000*ws,
-A/2+fwid,0+fwid,-1
```

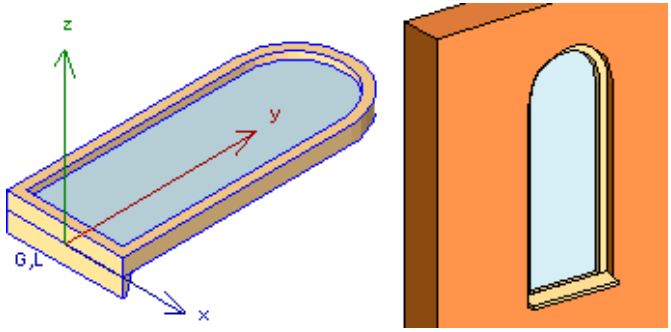
There is a problem with the inside line of the hole. Because the XY points at the inside head of the rectangular window are where they are, a pure tangential curve will not run smoothly parallel to the external outline. So you have to raise those two points to the same height as the highest point on the external outline. Use the flag 'ws' again to do this. The changes in the script are marked in bold. If you are typing this, try it without the modification and you will see the difference in quality.

Another point is that you do not want the curve to be interrupted by many edge lines. Change the 15 to 13 at the start and end of the curve; do this for the `WALLHOLE` and for the `PRISM`s for the Frame and the Glass. If you check the section on prisms in the 3D Shapes chapter of the GDL Reference Manual you will see that a masking value of 13 removes the vertical lines. Leave them at 15 first, then change them to 13, and compare the result in 3D.

```
!Glass
MATERIAL glasmat
ADDz fdep/2
PRISM_ 5,0.006,
-A/2+fwid,0+fwid,15,
 A/2-fwid,0+fwid,15,
 A/2-fwid,B-fwid+fwid*ws,13,
-A/2+fwid,B-fwid+fwid*ws,13+1000*ws,
-A/2+fwid,0+fwid,-1
DEL 1
```

As the internal line of the frame is the same as the outline of the glass, you are able to copy and paste the list of XY locations. The Sill does not change.

```
!Sill
MATERIAL frammat
ROTy 90
ADDz -A/2
PRISM 5,A,
 0,0,
 silproj,0,
 silproj,fwid/2,
 0,fwid,
 0,0
DEL 2
```



This is just a start. Now you have worked through this example, you may be willing to try more adventures on your own with PRISM.

Summary of GDL in this section

– Prisms are the most useful command in creative GDL – they can be bent, rounded, cut, drilled, chamfered and hollowed out.

– You do not need to move the cursor to build prisms, because the XY of each of their nodepoints can be defined. You may only have to lift them into position.

– You can shorten the time of building prisms with 'Instant GDL' using Drag and Drop.

– 'Instant GDL' is best built from slabs placed on the main origin. It is best used if prism shape is not going to change much. Otherwise, define the prism with parameters.

– The most common cause of errors when writing prisms is confusion with commas.

– Wall holes for openings are normally rectangular (A and B), but you can define a special shape with the WALLHOLE command.

– Flags can be set as a result of a pop-up menu, and then used to modify the shape of the object.

Chapter 6

Looking into Autoscripted GDL

It is valuable to look into the scripts of objects generated without GDL and know how to modify them.

6.1 Looking into Autoscripted GDL

There is a middle road in Object Making

Earlier, we practised making objects without GDL. We have now tried making objects with GDL. It may seem to you that the message of this book is that making objects without GDL should make you feel guilty. This is not so! You can make excellent objects without GDL, but you know that you are rejecting the chances of them being parametric. By contrast, if you make objects with GDL, you will feel virtuous – but you may become so enthusiastic about GDL that you try to do all the work yourself. You can suffer from exhaustion putting in the hours, and frustration if the results are not as good as hoped for.

However, there is a middle road that you can follow. You can get ArchiCAD to do much of the work for you and then come in at the half way point and add some of the qualities that make the object parametric. You can also make individual components using the 'Instant GDL' method, then move them into the GDL script, and use `ADD` and `ROT` to position the component.

First, let's look at an object completely made with ArchiCAD's own tools and saved as an object.

Making objects in the ArchiCAD environment

Anybody who uses ArchiCAD eventually has the curiosity to open the 3D script of an object made in the ArchiCAD environment – and has been thoroughly scared. The auto-scripted object contains a mass of complex 'industrial' code that is off putting to look at, and unstructured. It is good enough for the GDL interpreter – but it is not for humans to fully understand. By comparison, 'creative' GDL that has been composed by a human looks clean and uncluttered – if it is written with parameters, it is capable of extension into something more powerful from just a humble beginning.

If you can become familiar with the look and organization of autoscripted code and make use of it in easy steps, it is not as scary as you first thought.

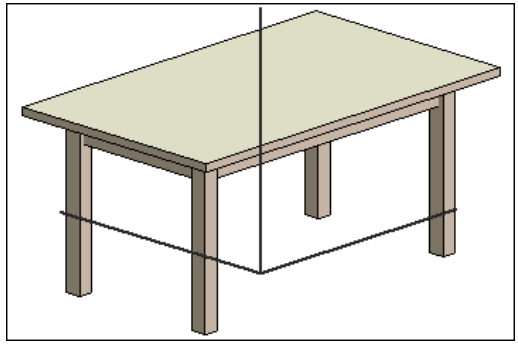
Let's remind you again how to make objects using ArchiCAD's own tools. We have two main methods. Make the object using the Slab, Wall and Roof tools, select it in the floor plan window. Select

with the pointer tool the object you have made from these 3D tools. Then:

1. If the object has been made the right way up, use the ArchiCAD File menu to Save Special as an ArchiCAD Object. Save it into one of your loaded libraries.
2. If the object has been made the right way up, or on its side, view in the 3D window in plan or elevation, Save the resulting view as a library part – a .GSM object. Save it into one of your loaded libraries.

Examine the 3D script of a typical autoscripted object

Open the table that was used as an example in Chapter 2, or make yourself a quick table now using method 1 above. We will run through explaining what is contained in the script, and then look at ways of shortening and modifying it. Try to make it approx 1.5 m wide by 0.9 m deep.



The 3D script starts out with some labelling. It's good to have date and title and author at the start of the script. Do likewise when you write creative scripts.

```
!   Name     : table_01.GSM
!   Date     : Monday, May 1, 2000
!   Version  : 6.50
!   Written by ArchiCAD
MULX  A/    1.521409
MULY  B/    0.919806
MULZ  ZZYZX/    0.7
ADD         0.771409,     0.45,    -0.0
BODY    -1
MODEL SOLID
RESOL       36
```

The three `MUL` commands enable the object to be stretchy. When it was first made, the table was drawn by eye in the floor plan, so it was approximately 1.5 meters by 0.9 meters, and the height was 0.7 meters.

ZZYZX is what GDL denotes for the height. This rather curious name arises from the need to call it something related to Z. But it could not be called Z as many older objects from ArchiCAD 4.5 and earlier used Z for other purposes and might no longer work in 5.0 and later. The GDL development team chose 'zzyzx' after a small town in Arizona, the last town in the US telephone directory.

Because the autoscript is accurate to a millionth of a meter, you get a vast number of trailing decimals.

Now you can see how stretchiness work in autoscripted objects. If you stretch the table to 2.0 meters, the `MULX` command will 'multiply everything in the X direction by A/1.521409'. As 'A' has a newly stretched value of 2.0, the multiplication will be 2.0/1.521409. This will make it slightly wider. If you stretch the height to be 1.2 meters, the vertical components will all be multiplied by 1.2/0.7. The `MULY` command works in the same way.

The `ADD` command is an offset – this is in case you built the object a long distance from the origin of the main floor plan. Some people build their objects hundreds, sometimes thousands of meters from the project origin. If you build very close to the project origin (it must be the main project origin, not a temporary one) then this offset will be very small. This offset will place the object's own origin within its own curtilage, usually, the bottom left-hand corner.

As this table was built centrally over the project origin, you can see that the offset is approximately half the width and depth of the table – so that it is putting the table's own origin at one corner.

The `BODY -1` command is one that autoscripts always carry. These are a command to the GDL interpreter to 'ensure the 3D integrity of each object'. They are not necessary in creative GDL until you get into more complex tasks such as the `TUBE` command.

`MODEL SOLID` just tells the object to be solid. This is the default condition, so if you did not write it, your model would still be solid even without the command. `RESOL 36` tells it that all rounded surfaces are to have 36 polygons to make up a full circle. 36 is the default value for this, so you do not really need this either.

```
GLOB_SCRIPT_TYPE =      3
GLOB_CONTEXT =      3
GLOB_SCALE =    100
GLOB_NORTH_DIR =      90.0
GLOB_FRAME_NR =     -1
GLOB_EYEPOS_X =    -0.780905
GLOB_EYEPOS_Y =    -0.557944
GLOB_EYEPOS_Z =     0.3
GLOB_TARGPOS_X =    -1.26576
GLOB_TARGPOS_Y =    -0.292552
GLOB_HSTORY_HEIGHT =     4.7
```

These 'GLOB' items are called Global Variables. They are not doing anything here except recording the conditions that prevailed when the object was made. At the time the table was made the North point was in the 90° direction, there was no animation taking place (so the Frame number is set to -1), the camera eye and target positions were noted, the drawing scale was 1:100. These are not commands, they are a historical record.

You can find out more about Global Variables in Appendix A of the GDL Reference Manual. Meanwhile, you can cheerfully delete all lines beginning with 'GLOB'.

```
!!Slab-016
PEN          1
ADDZ        0.67
GLOB_LAYER = " Library Constructions"
GLOB_ID = "Slab-016"
GLOB_INTID =     14
BODY    -1
CPRISM_ "Limestone, shiny", "Limestone, rough",
  "Limestone, rough",
             5,     0.03,
           -0.771409,     0.469806,     15,
            0.75,     0.469806,     15,
            0.75,    -0.45,     15,
           -0.771409,    -0.45,     15,
           -0.771409,     0.469806,     -1
BODY    -1
DEL      1
```

This is the first real 3D element in the script. It is obviously the tabletop as it is only 3 cm thick and is 0.67 meters off the floor. The tabletop was made with a Slab tool, and this always translates into a 'CPRISM_'. The syntax for CPRISM_ was explained in chapter 5. The prism is lifted by 0.67 and drawn; then the DEL command returns the cursor to the origin.

```
!!Slab-016
ADDZ        0.57
GLOB_INTID =     15
BODY    -1
CPRISM_ "Limestone, shiny", "Limestone, rough",
  "Limestone, rough",
            10,     0.1,
           -0.642659,     0.337009,     15,
            0.615665,     0.337009,     15,
            0.615665,    -0.322618,     15,
           -0.642659,    -0.322618,     15,
           -0.642659,     0.337009,     -1,
           -0.616168,     0.307869,     15,
            0.583876,     0.307869,     15,
            0.583876,    -0.288179,     15,
           -0.616168,    -0.288179,     15,
           -0.616168,     0.307869,     -1
BODY    -1
DEL      1
```

This is the frame under the tabletop. It is obviously the frame because the height of the prism is 0.57 metres, and the thickness of the prism is 0.1 meters. The frame is in fact a prism with a hole through it. The masking values shows you how a drilled hole works in GDL – the 15s are points on the outline, the first -1 means that the outline is complete, and the second occurrence of -1 means that the outline of the hole is complete.

The following commands are obviously table legs as they are all 0.67 meters in thickness, and are not lifted above the XY ground plane. If you analyze the XY points, they are all approximately square in a sectional size of 65 mm.

```
!!Slab-017
GLOB_ID = "Slab-017"
GLOB_INTID =     16
BODY    -1
CPRISM_ "Limestone, shiny", "Limestone, rough",
  "Limestone, rough",
             5,     0.67,
           -0.658554,     0.355553,     15,
           -0.597624,     0.355553,     15,
           -0.597624,     0.286676,     15,
           -0.658554,     0.286676,     15,
           -0.658554,     0.355553,     -1
BODY    -1
!!Slab-017
GLOB_INTID =     17
BODY    -1
CPRISM_ "Limestone, shiny", "Limestone, rough",
  "Limestone, rough",
             5,     0.67,
            0.567981,     0.355553,     15,
            0.62891,     0.355553,     15,
            0.62891,     0.286676,     15,
            0.567981,     0.286676,     15,
            0.567981,     0.355553,     -1
BODY    -1
!!Slab-017
GLOB_INTID =     18
BODY    -1
CPRISM_ "Limestone, shiny", "Limestone, rough",
  "Limestone, rough",
             5,     0.67,
           -0.658554,     -0.269636,     15,
           -0.597624,     -0.269636,     15,
           -0.597624,     -0.338512,     15,
           -0.658554,     -0.338512,     15,
           -0.658554,     -0.269636,     -1
BODY    -1
!!Slab-017
GLOB_INTID =     19
BODY    -1
CPRISM_ "Limestone, shiny", "Limestone, rough",
  "Limestone, rough",
             5,     0.67,
            0.567981,     -0.269636,     15,
            0.62891,     -0.269636,     15,
            0.62891,     -0.338512,     15,
```

```
                0.567981,     -0.338512,      15,
                0.567981,     -0.269636,      -1
BODY    -1
```

What about the 2D Script?

Look at the 2D script – it may well look just as complicated, but made of LINE2 and POLY2 commands. If you modify the shape or size of the 3D object significantly, you may have to delete the whole 2D script and replace it with a PROJECT2 command and some Hotspots at the corners.

This can be tuned up – to be parametric

If you have the object open, save a copy of it under a new name – 'Table_02.GSM' perhaps. Now make new parameters for Materials, Pen colors and Frame depth.

	Variable	Type	Name	Value
	A		X Dimension	1500
	B		Y Dimension	900
	ZZYZX		Table Top height	700
B	tthik		Tabletop thickness	30
	legmat		Legs Material	14
	framat		Frame Material	6
	topmat		Veneer on Table top material	17
	pcol		Pen colour 3D and 2D	1

Simplify A and B to exactly 1.5 m and 0.9 m, the dimensions we wanted in the first place when building the inaccurate table. Here, the parameter box is in millimeters. Do not forget to keep strictly to meters when writing in the actual script.

Rebuilding the 3D Script

Modify the labelling at the top to remind yourself when you altered the object. Retain the first two MUL commands so that the table will remain stretchy, but clean them up with the exact values of A and B. Delete the MULZ command – this will be done parametrically. Delete the offset, but remember to delete the equivalent offset command in the 2D script – or the symbol and the actual table will not coincide in location.

Put in a parametric PEN command, provide a label for the tabletop, and remove all the redundant GLOBs and BODYs.

```
! Name : table_01.GSM
! Date : Monday, May 1, 2000
! Version  : 6.50
! Written by ArchiCAD
! modified June 2000 by DNC
```

```
MULX  A/  1.50
MULY  B/  0.90
PEN  pcol
!Tabletop
ADDZ  zzyzx-tthik
cPRISM_ topmat, framat, framat,
   5, tthik,
 -0.75, 0.45, 15,
  0.75, 0.45, 15,
  0.75,-0.45, 15,
 -0.75,-0.45, 15,
 -0.75, 0.45, -1
DEL  1
```

For the tabletop, change the ADDZ command to be the total table height minus the tabletop thickness. Then change the tabletop thickness to our new parameter of 'tthik'. Then clean up all the 'millionth of a meter' decimals to clean rounded dimensions. As the CPRISM allows you to specify top material, bottom material and side material within the command, replace all that Limestone with your new material parameters of 'framat' and 'topmat'.

Although it takes a bit more thinking about, you can do the same thing to the frame under the table top, simplifying all the dimensions down to two decimal places, and making it so that the frame has a thickness of 30 mm. Accepting that the height of the frame is 0.1 meters, you can work out how high it has to be raised by the ADDZ command.

Notice that after the tabletop and the table frame commands are complete, the GDL returns the cursor to the origin. This is good.

```
!Tableframe
ADDZ  zzyzx-tthik-0.1
cPRISM_ framat, framat, framat,
  10, 0.1,
 -0.64, 0.34, 15,
  0.64, 0.34, 15,
  0.64,-0.34, 15,
 -0.64,-0.34, 15,
 -0.64, 0.34, -1,
 -0.61, 0.31, 15,
  0.61, 0.31, 15,
  0.61,-0.31, 15,
 -0.61,-0.31, 15,
 -0.61, 0.31, -1
DEL  1
```

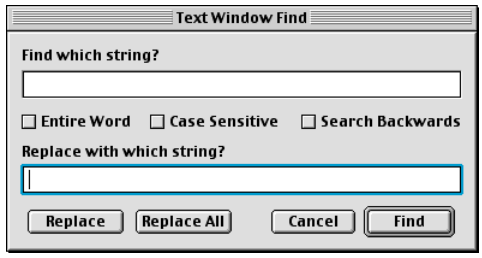
If you have followed thus far, you have momentum and you can push this process to completion. Label and change the table legs in the same way. Alter the height of the legs to the height of the table minus the tabletop thickness. Change the materials to 'legmat'.

```
!Legs
CPRISM_ legmat, legmat, legmat,
   5, zzyzx-tthik,
   -0.658554, 0.355553, 15,
   -0.597624, 0.355553, 15,
   -0.597624, 0.286676, 15,
   -0.658554, 0.286676, 15,
   -0.658554, 0.355553, -1
CPRISM_ legmat, legmat, legmat,
   5, zzyzx-tthik,
  0.567981, 0.355553, 15,
  0.62891, 0.355553, 15,
  0.62891, 0.286676, 15,
  0.567981, 0.286676, 15,
  0.567981, 0.355553, -1
CPRISM_ legmat, legmat, legmat,
   5, zzyzx-tthik,
   -0.658554,  -0.269636, 15,
   -0.597624,  -0.269636, 15,
   -0.597624,  -0.338512, 15,
   -0.658554,  -0.338512, 15,
   -0.658554,  -0.269636, -1
CPRISM_ legmat, legmat, legmat,
   5, zzyzx-tthik,
  0.567981,  -0.269636, 15,
  0.62891,  -0.269636, 15,
  0.62891,  -0.338512, 15,
  0.567981,  -0.338512, 15,
  0.567981,  -0.269636, -1
DEL TOP
```

To finish, you can issue a `DEL TOP` command. This is similar to the `DEL` command, but it DELs everything, including all the `MUL` commands at the beginning. If you write `DEL 2`, this will work just as well. `DEL 2` is better because it requires you to count the cursor commands at the start of the script, which means that you are in full control of the process – `DEL TOP` is just a guess.

Find and Replace, to clean up the script

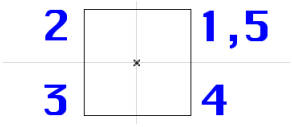
Finally, you could use the Find and Replace command to weed out all those spaces in the original script. Type 3 spaces in the Find field and one space in the Replace field. Place your text cursor at the top of the script, and hit the 'Replace All' button. Your text will be cleaned up nicely.



Add the final touches of quality

This Table could now be used, but there is an extra level of quality we can give it. The tabletop and frame were 'tidied up' – their eyeballed dimensions were converted to exact clean ones. The table legs could now appear somewhat uneven where they join the frame. If you use the BLOCK command, or better, use CPRISM, you could provide much better table legs. As the legs were square sections, approx 6-7 cm wide, let's make them exactly 7 cm wide, and modify the CPRISM_ so that it is axially symmetrical, as follows:

```
!One Leg
CPRISM_ legmat, legmat, legmat,
   5, zzyzx-tthik,
    0.035, 0.035, 15,
   -0.035, 0.035, 15,
   -0.035,-0.035, 15,
    0.035,-0.035, 15,
    0.035, 0.035, -1
```



Copy the prism command that produces one of the legs, and adapt the XY locations as shown. Do a little sketch like this, and number each corner. You can see that the prism starts from the top right, and works its way anti-clockwise around the tableleg. It is important to make sure the start point [1] is the same as the end point [5]. Use the diagram to make sure you get your minus signs in the right place.

You can simplify it even further. The only point in using CPRISM is if the materials of top, bottom and sides are different. For creative GDL with a solid prism that is all one color, you can use plain PRISM, and declare the Material value beforehand. You can remove the 15s and the underscore. Make sure you also remove the trailing comma.

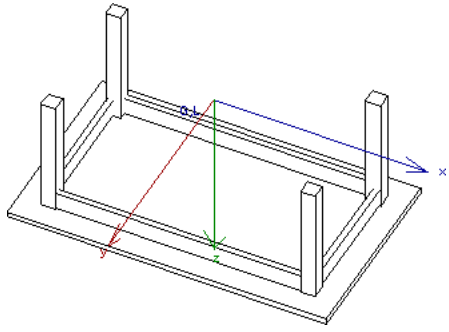
```
!One Leg - simplest statement
MATERIAL legmat
PRISM 5, zzyzx-tthik,
    0.035, 0.035,
   -0.035, 0.035,
   -0.035,-0.035,
    0.035,-0.035,
    0.035, 0.035
```

Now that you have a single leg, you can use the ADD command and post a leg to each corner of the frame. Now that the frame and leg are defined precisely, you will have a precise connection of

frame and leg that will look good close up. If you remember, the internal line of the table frame was defined in meters. So you can use the same dimensions.

```
!All Leg - simpler statement
MATERIAL legmat
ADD -0.61, 0.31, 0
PRISM 5, zzyzx-tthik,
    0.035, 0.035,
   -0.035, 0.035,
   -0.035,-0.035,
    0.035,-0.035,
    0.035, 0.035
DEL 1
ADD  0.61, 0.31, 0
PRISM 5, zzyzx-tthik,
    0.035, 0.035,
   -0.035, 0.035,
   -0.035,-0.035,
    0.035,-0.035,
    0.035, 0.035
DEL 1
ADD  0.61,-0.31, 0
PRISM 5, zzyzx-tthik,
    0.035, 0.035,
   -0.035, 0.035,
   -0.035,-0.035,
    0.035,-0.035,
    0.035, 0.035
DEL 1
ADD -0.61,-0.31, 0
PRISM 5, zzyzx-tthik,
    0.035, 0.035,
   -0.035, 0.035,
   -0.035,-0.035,
    0.035,-0.035,
    0.035, 0.035
DEL 1
```

As each leg is built, you use `DEL 1` to get back to the origin, and wait for the next command.



The table legs now have a clean relationship with the frame – as in a mortice and tenon joint

You are probably getting worried by this time about the amount of typing involved. If you make the maximum use of Copy and Paste, the amount of typing is not huge as most of your effort is in modifying commands rather than typing from new.

You might also want the option to change the leg shape from square to cylinder or cone. For this you would need to reorganize the script so that the legs are capable of having different shapes. The key to this is the SUBROUTINE. This is where a group of code is used often. Instead of retyping it again and again, it is better to give it a number, and just call it by number with a GOSUB command when a leg needs to be drawn. This tableleg is the ideal opportunity to illustrate the use of the subroutine. See chapter 7 for more on subroutines.

```
!All Leg - simpler statement
MATERIAL legmat
 ADD -0.61, 0.31, 0
GOSUB 100:!Draw one leg
 DEL 1
 ADD  0.61, 0.31, 0
GOSUB 100:!Draw one leg
 DEL 1
 ADD  0.61,-0.31, 0
GOSUB 100:!Draw one leg
 DEL 1
 ADD -0.61,-0.31, 0
GOSUB 100:!Draw one leg
 DEL 1
```

Somewhere, further down in the script, after an END command, you can have a subroutine numbered 100 that contains the PRISM statement.

Cautionary note for modifying objects in this way

If you make objects using the Wall tool, and try this method of modification, you need to be more experienced. The CPRISM is really a very friendly command when you get used to it – so it is easy to modify objects made with the Slab tool. The Wall tool generates XWALL commands in the autoscript, and these are more resistant to human intervention. As the Wall tool is frequently used for objects that are captured in side view from 90°, there are some additional ROT commands at the start of the 3D script which are best left untouched.

6.2 Come back to 'Instant GDL'

This is the method whereby you make a component in the ArchiCAD floor plan, then drag and drop it into a script window. (See chapter 5.) This method is not advisable for entire objects for which methods 1 and 2 are perfectly suited. But it is the ideal way to make components such as fretwork, curvy shapes, outlines for furniture legs and a hundred other uses. These can be stitched into the structure of your creatively scripted object. It is possible to do the same for 2D objects, such as fills and lines.

Use 2D to help your 3D model

Surprisingly, the greatest assistance to creative 3D GDL is the 2D Fill tool. Drag and drop a 2D fill into the 2D Script window. Then copy and paste the 2D fill (which by this time is a `POLY2_B` command) into the 3D window and convert it to a 3D command.

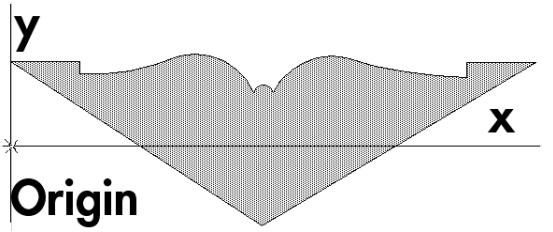
A lot of 3D commands involve drawing out a 2D profile and then doing something with it: `EXTRUDE` extrudes, `REVOLVE` lathes the section around an axis; `SWEEP` pulls a profile through 3D space, twisting or rescaling the section as it goes; `TUBE` also draws a 2D profile through 3D space, allowing it to be twisted and mitred as it goes; `PYRAMID` pulls the nodes on a 2D profile to one point; `RULED` takes the points on a 2D profile and allows you to form a 3D solid by repeating the same number of nodes at different heights. These all use the same syntax for XY points as `POLY2_B`.

All of these profiles support the use of Polylines. These are outside the scope of this book. If you use the 2D Fill tool and force lines to be curves, and then copy them to GDL, you will find many lines with 900 and 4000 in them. These define centres and radii of curvature. If you copy and paste these 2D profiles and use them in this way, you don't entirely have to know how they work. They just do.

Try instant GDL with a `REVOLVE`

There is not time to go into all of these, but let's try one – `REVOLVE`. Look up `REVOLVE` in the GDL Reference Manual. This method makes it easy beyond your expectations to make a lathed object. Basically you only have to draw a 2D profile in the Positive-Y zone – that means all point above the X axis – and you can revolve it. When you draw the profile, you have to close it. The easiest way is to select a 'silly' point somewhere well away from the main profile. If it is in the negative-Y zone, you will easily identify it when it is converted into GDL code.

Create a New Library Part from the File menu, and position the 2D window to the right of the screen, so that it is next to the project floor plan. Drag and Drop the 2D fill into the 2D Script window.



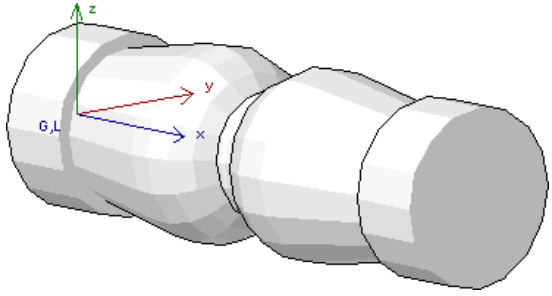

```
PEN          1
SET FILL "25 %"
POLY2_B      22,      3,      1,     91,
            -0.000071,      0.06026,       1,
             0.048921,      0.06026,       1,
             0.048921,      0.051348,      1,
             0.055658,      0.217545,    900,
             0.0,       25.2526,   4001,
             0.120465,      0.064357,      1,
             0.13148,     0.020407,    900,
             0.0,      -80.7543,   4001,
             0.173089,      0.03834,       1,
             0.180185,      0.036271,    900,
             0.0,     -147.4796,   4001,
             0.18728,     0.03834,       1,
             0.229512,      0.016782,    900,
             0.0,      -74.8857,   4001,
             0.239313,      0.063174,      1,
             0.327746,      0.331473,    900,
             0.0,       17.8153,   4001,
             0.325639,      0.048983,      1,
             0.325639,      0.059626,      1,
             0.374715,      0.059626,      1,
             0.179133,     -0.056834,      1,
            -0.000071,      0.06026,      -1
HOTSPOT2     -0.000071,      0.06026
HOTSPOT2      0.048921,      0.06026
HOTSPOT2      0.048921,      0.051348
HOTSPOT2      0.120465,      0.064357
HOTSPOT2      0.173089,      0.03834
HOTSPOT2      0.18728,     0.03834
HOTSPOT2      0.239313,      0.063174
HOTSPOT2      0.325639,      0.048983
HOTSPOT2      0.325639,      0.059626
HOTSPOT2      0.374715,      0.059626
HOTSPOT2      0.179133,     -0.056834
HOTSPOT2     -0.000071,      0.06026
```

Copy only the list of XY points across to the 3D script window, and delete the hotspots and all the rest.

Delete the last point in the list (which is simple a repeat of the first point). Delete the second last point (which is the ‘silly’ point).

Now add the REVOLVE command above the list of points. The word REVOLVE is followed by the number of points (was 22, now 20), the angle through which the profile should be revolved (360°) and finally a masking value, which you can assume to be 63.

```
PEN 1
MATERIAL 'Whitewash'
RESOL 16
REVOLVE 20,360,63,
 -0.000071, 0.06026,  1,
  0.048921, 0.06026,  1,
  0.048921, 0.051348,  1,
  0.055658, 0.217545,  900,
  0.0,  25.2526, 4001,
  0.120465, 0.064357,  1,
  0.13148, 0.020407,  900,
  0.0, -80.7543, 4001,
  0.173089, 0.03834,  1,
  0.180185, 0.036271,  900,
  0.0,  -147.4796, 4001,
  0.18728, 0.03834,  1,
  0.229512, 0.016782,  900,
  0.0, -74.8857, 4001,
  0.239313, 0.063174,  1,
  0.327746, 0.331473,  900,
  0.0,  17.8153, 4001,
  0.325639, 0.048983,  1,
  0.325639, 0.059626,  1,
  0.374715, 0.059626,  1
```



If this lathed shape is a stone baluster, you only have to precede the command with a ROTY -90 command and it will stand upright.

The RESOL command is most important. The RESOL command tells GDL that you only want 16 polygonal faces to the shape. You can ruin your model with too many polygons. Multiply these stone balusters by a few hundred and your computer may not have enough memory to render, and you may have an unrealistic number of polygons if you wish to export to DXF or Art•lantis. The benefit of being a GDL user is that you can control the number of polygons with a simple RESOL command.

Summary of GDL in this section

– The table example is so simple that you may feel that, by now, you are quite capable of scripting this in native GDL without needing the autoscript. But it has been a good means of demonstrating the contents of an autoscripted object.

– The easiest things to change into parameters are the names of materials – they can be replaced with variable names.

– The methods used in this chapter are fine for complete objects. But if you are working at creative GDL and simply want a quick solution, try 'Instant GDL'.

– Many 3D objects use the same syntax as the 2D Fill tool, so you can make interesting 3D objects easily.

– It is important to control the number of polygons in the model – if you are building any objects that are repeated in great numbers you must use GDL to define the number of polygons in each solid.

Chapter 7
GDL Roundup

There is a lot more to GDL than we have been able to cover in this slim volume. This section runs through some of the further possibilities in a brief form.

7.1 GDL Roundup

This book is an introductory tutorial for GDL, a tour through the easy foothills. If your interest has been aroused, there is further reading for you. No doubt, the best way to learn GDL is to attempt some practical tasks, which will be both useful to you in your work and will teach you additional techniques in GDL.

In GDL, there are four main areas of knowledge

– We have to analyze the 3D nature of the object we wish to build. See how much of it can be done with simple blocks or cylinders, and then how much of it will require more complex commands such as `PRISM`, `REVOLVE` and `TUBE`.

– The GDL itself. The syntax of each command, which command to use under the circumstances, how to control polygons, how to define text style and materials – and much more. You need to become conversant with 3D and 2D syntax.

– The programming language of GDL. This has not been covered in detail in this tutorial, but it is possible to write long complex scripts in GDL using Loops and Subroutines. GDL is based on BASIC, a programming language that was devised in the 70s, became universally known to users of the Apple II, Commodore Pet and Tandy TRS-80 machines of the early eighties; and now it has been rather forgotten. In GDL, it lives on! GDL can read and write data/text files, include complex calculation routines with a GDL script, such as structural calculation or predicting the outline points of a surface in 3D space.

– GDL can react according to conditions in the main ArchiCAD model. We saw in Chapter 4 how a doorframe could respond to the current drawing scale. With knowledge of the thickness and materials of the current Wall, a Window can be more powerful. With knowledge of the object's position in 3D space and the location of the current camera, the object knows how far away the camera is, and in which direction. Thus, it can turn to face the camera if it's a flat tree; or simplify itself if its a complex object that is too far away to require drawing in detail.

See also....

The **GDL Cookbook** covers all of these topics in greater detail, and provides practical working examples to help you learn.

The programming language of GDL

You have discovered that you can write GDL without being a programmer. If you can issue a few GDL commands, move the cursor around, write some `IF` statements that make the object smarter and safer, then you have been programming. In fact, programming is an activity all of us do – ants do it when looking after their eggs, squirrels do it when hoarding their winter food, lions do it when hunting zebra, children do it when playing with their toys and developing social relationships. Adults do it whenever they get in their car and drive to work.

Programming simply means the organization of a set of actions conditioned by a number of `IF` statements that decide which actions shall be taken. It also implies many repetitive actions – in real life, you could stack hundreds of books on a shelf and make decisions for each one as you proceeded to sort them in alphabetical order, by ISBN or size.

To get further with GDL you will need to learn about **Subroutines** and **Loops**.

7.2 Subroutines

Subroutines are used to avoid repetitive typing

You can take a piece of text that will have to be repeated frequently, give it a number, and then call the subroutine with a `GOSUB` command. For example, in the case of our basic chair in Chapter 3, if your chair has an elaborate leg that takes a hundred lines to describe and you have four legs to do, you would not want to type that all out again, in fact you would not even want to copy and paste the leg four times.

```
!All the legs
GOSUB 100:!One leg
   ADDX A
GOSUB 100:!One leg
   ADDY B
GOSUB 100:!One leg
   ADDX -A
GOSUB 100:!One leg
   DEL 3
```

Subroutines are used to give structure to a long script

The main script can consist of a number of `GOSUB` commands to a series of subroutines that build elements of the model. These are logically based on your 3D analysis.

```
!Chair
GOSUB 200:!All the Legs
GOSUB 300:!Draw the Seat
GOSUB 400:!Build the Back
END !----------------------------
100:!One single leg
 BLOCK tsec,tsec,seathit-tsec
RETURN
200:!All the legs
 GOSUB 100:!One leg
   ADDx A
 GOSUB 100:!One leg
   ADDy B
 GOSUB 100:!One leg
   ADDx -A
 GOSUB 100:!One leg
   DEL 3
RETURN
```

In this case, you would refer to the first part of the script as the 'executive script', with most of the actual work being done by the subroutines that follow the `END` command. Let's summarize some easy to remember 'must-do' rules.

The subroutines must all be written **after** the `END` command. Subroutines must start with a **number** and a label, and end with the command **`RETURN`**. Subroutines may call other subroutines. Subroutines must be **self contained**; the number of DELs must exactly equal the number of cursor movements.

Subroutines are used to help with decision making

When you have defined a flag, like 'ws' for window shape, to be valued 0, 1 or 2, you can issue a `GOSUB` command that includes the flag in the line number. For example:

```
GOSUB 100+ws !Window
END:!---------------------------------
100:!Rectangular window
...........
 RETURN
101:!Round topped window
...........
RETURN
102:!Gothic window
...........
RETURN
```

7.3 Loops

Machines never tire of doing repetitive work. With a GDL object, you may have many spokes in a wheel, many shelves in a furniture system, many balusters in a handrail, many rungs on a ladder. If we have a means of telling GDL how many we want, and what spacing we want it at, we can get the work done for us.

The FOR... NEXT Loop

We can do this with the FOR... NEXT loop. We can do it in several ways, but the most common is to do it by counting numbers, by distance or by angle. Let's show three simple examples.

Loop by counting numbers

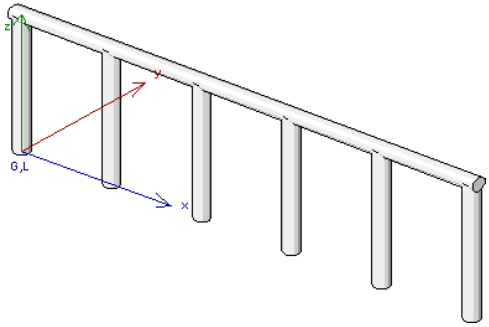
We want to plant a handrail at the racecourse and want to plant 6 x 1 meter high poles at 0.6 meter intervals, then cap it with a long rod. We assume an imaginary counter called 'k', and we increment it by 1 each time the program goes around the loop. So it starts with a value of 1, does one pole, advances to the position of the next pole, and waits. Then it meets the NEXT k command. As the job is not finished, it returns to the FOR line, increments 'k' again, and plants another pole. Eventually, it completes the task, and NEXT k recognizes that it has finished and allows the program to continue, whereby it DELs back to the origin.

```
!Long fence using numbers
RESOL 10
MATERIAL 'Whitewash': PEN 1
pspac=0.6  !Pole spacing

FOR k=1 TO 6 STEP 1
 CYLIND 1.0,0.05
 ADDX pspac
 NEXT k
 DEL 6

!Capping piece
ADD  -0.05,0,1.0
ROTY 90
CYLIND 3.1,0.05
DEL 2
```

Note that a trick is used to improve the look of the capping rail. By making it slightly longer (by one diameter) than the sum of the distances of the poles, it looks more authentic.



We can improve the way the loop is coded in one small way. The first one worked fine, but we had to DEL all the cursor moves AFTER the loop was complete. This method includes the DEL inside the loop, so that the loop is cleanly self-contained.

```
FOR k=1 TO 6 STEP 1
 ADDX pspac*(k-1)
 CYLIND 1.0,0.05
 DEL 1
 NEXT k
```

Now we place the ADDX command before the cylinder. The first time, the value of (k-1) is zero, so if you multiply 'pspac' by zero, the distance added is zero. So it plants the first pole. Then it DELs back to the origin. When 'k' is equal to 2, the distance moved is 'pspac*1' so it draws the next pole, then returns to the origin. Notice that in both cases, we specified the stepping rate to be '1'. If it is '1', you can omit that STEP command, but if it is anything else at all, you must specify the STEP, or GDL will assume a value of 1 – which could be misinterpreted as 1.0 meters, or 1.0 degrees!

Loop by Distance

The next method is when you do not know how many will be needed, but you know their spacing. This says, 'starting from point zero, advance a distance of 'd' (which is zero) and plant a pole. Then return to zero. Then increment the value of 'd' by 'pspac', then advance 'd' (which by now has been incremented), plant another pole, then return to zero. Do this until you have covered a total distance of 3.0 metres.'

```
pspac=0.6  !Pole spacing
FOR d=0 TO 3.0 STEP pspac
   ADDx d
   CYLIND 1.0,0.05
   DEL 1
   NEXT d
```

Loop by Angle

There are many occasions when your object has interesting circle geometry. Here is the same handrail, but now it's on a curved track. We use the FOR... NEXT loop to distribute the posts at an angular spacing of 15°.

This brings the opportunity to introduce you to two more GDL commands: ELBOW and RADIUS. ELBOW is like a cylinder but it curves at a defined radius, to a defined angle.

```
ELBOW curve radius, alpha angle, tube radius
```

In this case, it is a 180° curve angle. Because elbows always grow upwards, you have to precede the command with a ROTX -90 and lift it to the height of the top of the posts.

```
RADIUS small radius, large radius
```

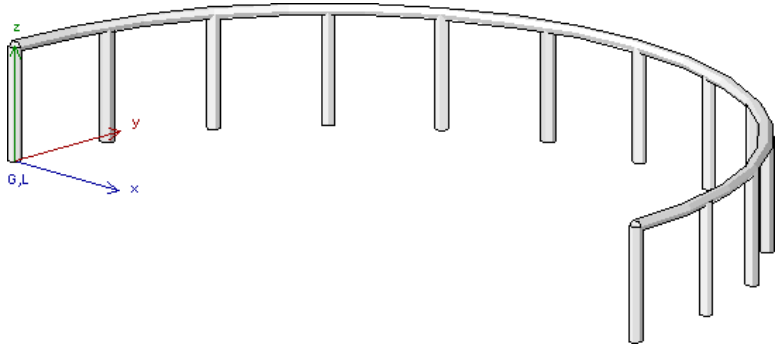
RADIUS does something similar to the RESOL command. However, RESOL is too simple. In this elbow, we have two curves, the main curve and the tubing curve. If we have a large resolution so that the rail looks right, the tubing will have too many polygons, and the computer could be crippled trying to shade all those surfaces.

If you issue a RADIUS command, all radii equal to or less than the small radius will have a RESOL 6, and all radii equal to or greater than the larger one will have a RESOL of 36. All radii with in between values will have in between values of RESOL. It is a very useful command.

```
!Circular racecourse rail
RADIUS 0.05,0.3
crad=3.0 !Curve Radius
 ADDz 1.0
 ROTx -90
ELBOW crad,180,0.05
 DEL 2

ADDx crad
FOR ang=0 TO 180 STEP 15
 ROTz ang
 ADDx crad
 CYLIND 1.0,0.05
 DEL 2
 NEXT ang
DEL 1
```

We have to move to the centre of the circle first. Turn to face in one direction using the ROTZ command, move out to the rail with an ADDX command, and draw the post. Withdraw to the centre again, increment the value of 'ang' by 15°, then do the next one. One final DEL returns you to the starting place.



Other ways of Looping

You have other methods of looping,

- `DO... WHILE`: do an action while a condition applies, for example a dimension or an angle will eventually be equal to less than zero; until then, keep doing it.
- `WHILE condition DO... ENDWHILE`: does the same as above.
- `REPEAT... UNTIL`: go on doing something until a condition is met – for example, when a dimension or angle is equal to or less than zero, or the numbers in a memory buffer have been used up.

The `FOR... NEXT` loop meets almost all your requirements. Most loops are based on counters, or incrementing until a distance or angle are achieved. You will know when a situation suitable for `REPEAT... UNTIL` is the best way for you to loop.

Look at the GDL Reference Manual

This primer has been written in the hope that as your approach to GDL becomes more confident, you will be more willing to dip into the actual manual. The explanations may not be as dry or technical as you first thought.

Other possibilities

There are many more capabilities in GDL than we have had space for in this primer. There are just too many to cover in one tutorial. But you can find out more by reading further or taking a course in GDL. If you made it this far the hard way – by reading and doing some of the exercises – you may be prepared to try some of these yourself.

Capabilities of GDL that this tutorial leaves for future investigation:

– Organizing the parameter box with titles and 'cascading' parameters.

– More powerful pop-up menus – with numbers and pictures.

– Numeric arrays (tables) available to the user as parameters.

– Making curvy skin-based surfaces using `COONS` or `MASS`.

– Planar 3D GDL objects when they do not need to be solid.

– Smart 2D symbols using `FRAGMENT2`, which combine drawn symbols with 2D scripting.

– Applying 'Polyline' curves to the outlines of prism, poly, extrude, revolve and other objects.

– Using 'masking' to control the way objects and surfaces look in 3D drawings and in photorendering.

– Using a `CUTPLANE`, `CUTSHAPE` and `CUTPOLY` to carve and drill solid shapes, to make them more complex.

– Attaching PICTUREs to surfaces – for trees, people, facades.

– Calling other library objects (such as taps or door handles).

– Making Lamp objects with controllable light sources.

– Using trigonometry to calculate the pathway of tubes and edges through curved and parabolic pathways.

– Reading and displaying the system date, time, project name, and much more, using the `REQUEST` command.

– Defining your own materials, line types and fill patterns.

– Storing numbers in memory and bringing them out later (Arrays and `PUT` & `GET`).

– Building a graphic User Interface that replaces the normal parameter box and makes the GDL object look more professional.

– Self-labelling objects – the font and text size can be defined to be parametric and scale sensitive.

– Setting up animation possibilities by changing the object's shape according to the frame number in the animation.

– Knowing where the camera is, and changing accordingly.

– Knowing which story the object is on.

– Reading and writing text based datafiles.

Index

2D 13
2D Script 38, 49
  First steps with ~ 44
  "killer command" 38
3D 14, 98
  ~ Cursor 35
  ~ Entities 34
  ~ Pixel 43
  ~ Projections 16
  ~ Script 34
  ~ Space 35
  ~ View 36

A

ADD 36
Add-ons 5
add-ons 3
ADD2 44, 48
ARC2 45, 48
ArchiSite 5
Art•lantis Render 5, 95
Autoscripted GDL 82

B

BASIC 9, 98
Binary 5
Binary 3D 23
BLOCK 34, 39
Boolean choices 51
BPRISM_ 69

C

Chair 18, 39
CIRCLE2 45, 48
Commas 71
Comment 33

CONE 34
Cones 22
CPRISM 6
CPRISM_ 69
CSLAB 6
CUTPLANE 69
CYLIND 34, 102

D

DEL 38
Door 53
DWG 4
DXF 4, 95
DXF/DWG Conversion 4

E

ELBOW 46, 103
Error checking 43

F

Fill tool 13
Find and Replace 70, 89
Flags 50, 76
FPRISM 69

G

GDL Cookbook 2, 98
GDL Object Web Plug-in 4
Global Variables 58, 85

H

hotspot 13
HOTSPOT2 48
Hotspots 45

I

IF... ENDIF 50
IF… ENDIF 62
Instant GDL 67, 93
Internet
  Objects on the ~ 4
Investment object 9

L

Label 37
Lamp 7
LET 55
Libraries
  ArchiCAD ~ 4
  Complementary ~ 4
  DXF and DWG ~ 4
  listing ~ parts 10
  loading ~ 3
LINE2 45, 48
Location awareness 58
Loops 101
  DO... WHILE 104
  FOR... NEXT 104
  REPEAT... UNTIL 104
  WHILE 104

M

Magic Wand 12, 21
Manufacturers' Rules 43, 63
Marquee 25
Masking codes 68
MASS 6
Master Script 33, 60
Master window 32
MATERIAL 34
MUL 36, 83
MUL2 48

O

Object Factory 4
Objects On Line 4

P

Parameter Script 33
Parameter table 32
Parametric 7, 40, 87
POLY2 48, 52
POLY2_ 48
Polylines 76
Pop-up menu 60, 76
Preview Picture 33
Preview picture 63
PRISM 66, 69
PRISM_ 69
Prisms 66, 69

Profiler 5
PROJECT2 38, 44, 56, 75
Properties Script 33, 63

R

RADIUS 103
RECT2 48
RESOL 95, 103
REVOLVE 93
Roof Truss 20
Roof vault 25
RoofMaker 5
ROT 36
ROT2 48

S

Scale
  drawing ~ 58
Slab tool 14, 18
SPHERE 34
SPRISM_ 69
StairMaker 5
Status code 76
Stretchiness 41, 49, 83
stretchiness 12
SUBROUTINE 92
Subroutines 99

T

Table 14, 83

U

User Interface Script 34

V

VALUES 60, 76

W

Wall tool 18
WALLHOLE 57
Window 15, 71

X

XWALL 6

Z

Zoom 4
ZZYZX 84

Contents

Chapter 1: Introduction to Object Making 1

1.1 About Object Making 2
Library Parts in ArchiCAD 3
Sources of Library Parts 4
1.2 Making your own Library Parts 6
Without GDL – using ArchiCAD's Tools 6
Making Objects with GDL 7
How do you make an Object? 8
Does it need to LOOK right, or BE right? 8
The Idea of an 'Investment object' 9
Can you learn GDL? 9
Object Making in the ArchiCAD Manuals 10

Chapter 2: Making Objects without GDL 11

Making Objects without GDL 12
Getting Started 12
Let's Make something in 2D 13
Let's Make a 3D Object – a Table 14
Let's Make a 3D Window 15
Let's Make a Furniture Object using Walls 18
Let's Make a Roof Truss 20
Let's Make an Object using existing Library Parts 22
Let's Make an Object that we can cut to shape 24
How far can we take Object Making without GDL? 27
Notes on the naming of parts 29
Summary of Object Making in this chapter 29

Chapter 3: Starting with GDL 31

3.1 Starting with GDL 32
3D Entities 34
First steps in 3D GDL 34
3D Space and the 3D Cursor 35
Do not forget the 2D script 38
Summary of GDL, to this point 38
3.2 Let's build a 3D Object 39
Let's make this chair parametric 40
3.3 First steps in 2D scripting 44
Summary of GDL, in this section 46

Chapter 4: Practical Uses for GDL 47

4.1 Practical Uses for GDL 48

GDL in 2D . . . 48

Let's build something in 2D – and make it snappy . . . 49

Summary of GDL in this section . . . 53

4.2 Let's make a 3D Door 53

Summary of GDL in this section . . . 64

Chapter 5: The Power of PRISM 65

5.1 The Power of PRISM 66

PRISM is the most versatile element in 3D GDL . . . 66

Instant GDL . . . 67

About Prisms... . . . 69

5.2 Let's make a 3D Window 71

Don't forget the 2D Symbol . . . 75

Try WALLHOLE . . . 76

Summary of GDL in this section . . . 79

Chapter 6: Looking into Autoscripted GDL 81

6.1 Looking into Autoscripted GDL 82

This can be tuned up – to be parametric . . . 87

6.2 Come back to 'Instant GDL' 93

Try instant GDL with a REVOLVE . . . 93

Summary of GDL in this section . . . 96

Chapter 7: GDL Roundup 97

7.1 GDL Roundup 98

The programming language of GDL . . . 99

7.2 Subroutines 99

7.3 Loops 101

The FOR... NEXT Loop . . . 101

Other ways of Looping . . . 104

Look at the GDL Reference Manual . . . 104

Other possibilities . . . 104

Index 107