Getting Help

With non-classic MGED, right-clicking most labels and input fields will provide a description. Additionally, documentation is provided via the Help menu and on-line at `http://brlcad.org/`

obtaining help on all commands	**help**
obtaining help on a particular command	**help** *command*
search for commands that relate to keyword	**apropos** *keyword*
display command history for current session	**history**
record transcript of commands used to file	**journal** *file*
list subset of various simulatable GUI actions	**press** *help*

Geometry Information

list the top-level objects	**tops**
list the objects in currently open database	**ls**
get a table of contents for current database	**t**
display the information details for object(s)	**l** *obj ...* **cat** *obj ...*
get/set title of currently open database	**title**
get/set units of currently open database	**units**
print out CSG hierarchy for object(s)	**tree** *obj ...*
display combinations that reference object(s)	**dbfind** *obj ...*
display full paths that reference object(s)	**dbfindtree** *obj ...*
list all CSG paths under given object(s)	**paths** *obj ...*
show transformation matrices along a path	**showmats** *path*
list all regions referenced by object(s)	**get_regions** *obj ...*
display all regions with given air code(s)	**eac** *code ...*
display counts of primitives, regions, groups	**summary p r g**
save region identifier summary to file	**idents** *file obj ...*

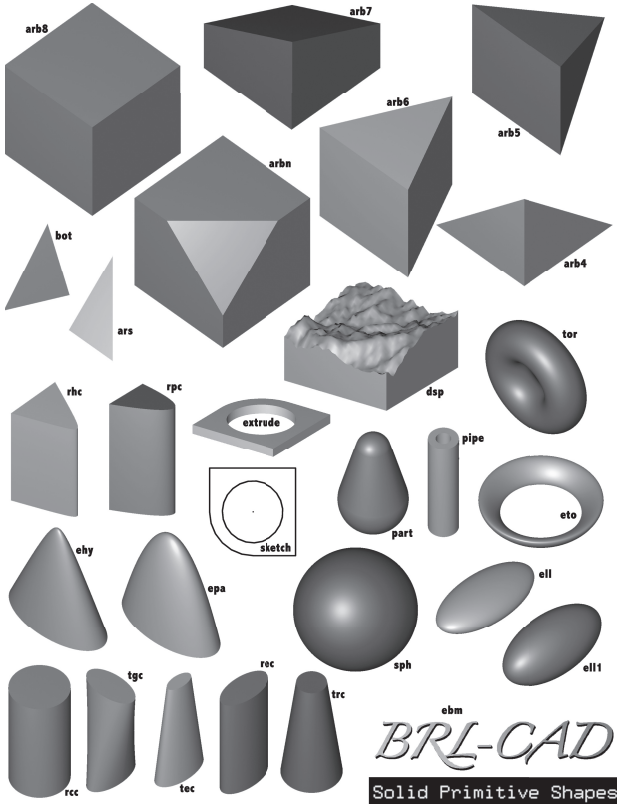
Creating Geometry

interactively type in new object parameters	**in**
create a prototypical primitive object	***make type name***
create a CSG combination object	**comb** *name op obj ...* **c** *name obj op obj ...*
create CSG region (aka "part") combination	**r** *name op obj ...*
create group (aka "assembly") combination	**g** *name obj1 obj2 ...*
create a region from a range of solids	**build_region** *prefix #*
create a shallow copy of an object	**cp** *obj objcopy*
create deep patterned copies of objects	**clone**
rename an object	**mv** *old new*
rename an object and all references	**mvall** *old new*
add a prefix to all references to an object	**prefix** *prefix obj*
create an arb8 with rotation and fallback	**arb** *rot fallback*
duplicate a cylinder, positioned at end or orig	**cpi** *cyl cylcopy*
make a bounding box around object(s)	**make_bb** *name obj ...*
mirror an object about the x, y, or z axis	**mirror** *obj new axis*
create arb given 3 points, 2 coords of 4th, and thickness	**3ptarb**

Deleting Geometry

MGED provides ***no direct means*** to recover deleted geometry, so **delete objects with caution**. Regularly performing geometry database backups (e.g. see the '`dump`' command) is recommended.

delete object(s) from database	**kill** *obj ...*
delete object(s) and all references	**killall** *obj ...*
delete object(s), all sub-objects, all references	**killtree** *obj ...*



Displaying Geometry

display object(s) for editing	**e** *obj ...* **draw** *obj ...*
erase object(s) from the display	**d** *obj ...* **erase** *obj ...*
erase any objects that reference object(s)	**dall** *obj ...* **erase_all** *obj ...*
"zap": clear all objects from the display	**Z**
"blast": clear all objects & display object(s)	**B** *obj ...*
mark object(s) as "hidden" to hide from '`ls`'	**hide** *obj ...*
unmark object(s) as "hidden"	**unhide** *obj ...*
hierarchical geometry browser GUI tool	**geometree**

Rendering Geometry

raytrace current view to a lingering window	**rt** *-F/dev/X1*
raytrace current view to 2048x2048 file	**rt** *-s2048 -o file.pix*
raytrace white background hidden-line image	**rtedge** *-W -o file.pix*
abort any raytraces started within mged	**rtabort**

Customization

MGED will process a "`.mgedrc`" initialization file in your home directory as a sourced Tcl script. This file generally contains defaults set by the GUI but may also include your own customizations including new commands, shortcuts, loadable plugin modules, and custom key bindings.

Text File & Table Editing

Several commands in MGED utilize an external text editor, determined from your environment EDITOR setting, to edit object values. Depending on your shell, you may need to set your EDITOR environment variable before invoking MGED. Bash example: `export EDITOR=pico`

edit a combination using a text editor	**red** *comb ...*
edit the region identifier codes for object(s)	**edcodes** *comb ...*
edit the combination/region materials	**edmater** *comb ...*
print the color table	**prcolor**
edit the color table codes	**edcolor**
read/import region identifier codes from file	**rcodes** *file*
write region identifier codes to file	**wcodes** *file obj ...*
read combination materials from file	**rmater** *file*
write combination materials to file	**wmater** *file obj ...*
write report of primitive solids to file	**solids** *file obj ...*

Manipulating the View

get/set the various view parameters	**view**
automatically resize/recenter the view	**autoview**
redraw the current view	**refresh**
set the azimuth, elevation, and twist	**ae** *az el tw*
set/get the view center	**center** *x y z*
set/get the eye point	**eye_pt** *x y z*
set/get the viewing direction	**lookat** *x y z*
set/get the view size	**size** *size*
zoom the view by specified scale factor	**zoom** *scale*
set the perspective viewing angle	**set perspective** *angle*
translate/move the view relative to current	**tra** *dx dy dz*
scale the view size by given factor	**sca** *factor*
rotate the view by x, y, z degrees	**rot** *x y z*
rotate view about a specified model vector	**mrot** *x y z*
rotate viewpoint by specified degrees	**vrot** *xdeg ydeg zdeg*
set view using direction and twist angle	**qvrot** *dx dy dz angle*
set view using x, y, z angles in degrees	**setview** *xdg ydg zdg*
pan the view	**sv** *x y*
set the view orientation from quaternion	**orientation** *quat*
emulate a knob twist	**knob** *params*
control the angle/distance cursor	**adc**
save the current view orientation to a file	**saveview** *file.rt*
load a saved view orientation from a file	**loadview** *file.rt*
save current wireframe to a Postscript file	**ps** *file.ps*
save current wireframe to a UNIX plot file	**plot** *file.pl*
overlay a UNIX plot file onto the display	**overlay** *file.pl*

Analyzing Geometry

analyze the faces of an ARB	**analyze** *arbname*
rough estimate of presented area	**area**
trace single ray from current view or x, y, z	**nirt** *x y z*
trace single ray from x, y position	**vnirt** *x y*
get/set query_ray behavior settings	**qray**
check for overlaps (aka interferences)	**rtcheck**
compute view-dependent surface areas	**rtarea**
get/set MGED calculation tolerances	**tol**

Editing Geometry

MGED is a modal editor (akin to "vi") meaning that you have to enter and exit various editing modes. The primary mode states related to editing are VIEWING (default), SOLEDIT, and OBJEDIT. Some commands are only valid in certain modes or change behavior based on mode.

visually illuminate & select combination	**ill** ***comb***
visually illuminate & select solid primitive	**sill** ***prim***
enter object-illuminate mode	**press oill**
get the current editing state	**status state**
edit a primitive (enter solid edit mode)	**sed** ***prim***
edit a matrix (enter object edit mode)	**oed** ***l*****path** ***rpath***
add object reference to existing combination	**i** ***obj comb***
remove object reference(s) from combination	**rm** ***comb obj ...***
set/get the center of editing transformation	**keypoint** ***x y z***
manipulate an object's matrix or material	**arced** ***path cmd***
copy the matrix on one object to another	**copymat** ***path1 path2***
select matrix path when in pick mode	**matpick** ***path1 path2***
set a matrix on a given path	**putmat** ***path m0 ... m16***
apply all matrix transformations down to the primitives	**push** ***obj ...***
same as push but creates new primitives as needed	**xpush** ***obj ...***

The geometry editing commands below including the commands related to translation, scaling, and rotation require that MGED be in an edit mode before they can be utilized. The commands implicitly apply to the objects currently selected (e.g. with '`sed`' or '`oed`') for editing.

set parameter(s) for current edit operation	**p** ***val ...***
return to viewing mode, accept any edits	**accept**
return to viewing mode, rejecting any edits	**reject**
edit selected primitive using a text editor	**ted**
edit the face of selected arb interactively	**facedef** ***face***
mirror selected arb face across x, y, or z axis	**mirface** ***face axis***
permute the vertices of selected arb	**permute 8***vertices*

Translating or Moving Geometry

move object being edited to relative position	**tra d***x dy dz*
move object being edited to absolute position	**translate** ***x y z***

Scaling or Resizing Geometry

scale primitive being edited	**sca** ***factor***
scale combination object being edited	**oscale** ***factor***
extrude arb face by some absolute distance	**extrude** ***face dist***

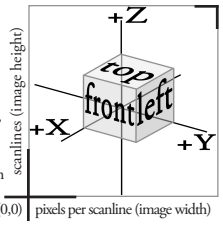
Rotating Geometry

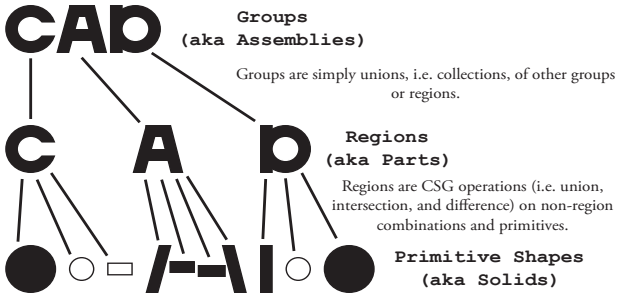
rotate primitive being edited	**rot** ***x y z***
rotate combination object being edited	**orot** ***x y z***
rotate angle degrees about an arbitrary axis	**arot** ***x y z angle***
incrementally rotate combination object	**rotobj -i d***x dy dz*
rotate combination about vector	**qorot** ***x y z dx dy dz angle***
use provided planar coefficients when rotating arb face	**eqn** ***A B C***

BRL-CAD Coordinate Systems

BRL-CAD uses a ***right-hand*** **3D** Cartesian coordinate system with real number addressing where "up" is in the positive z-axis (+Z) direction, "left" and "right" are perpendicular to the y-axis, and "front" is towards the positive x-axis (+X) direction.

BRL-CAD uses a ***first-quadrant*** **2D** Cartesian coordinate system with integer addressing where `(0,0)` is the lower-left corner pixel and `(width-1,height-1)` is the top-right pixel in an image.





Groups (aka Assemblies)

Groups are simply unions, i.e. collections, of other groups or regions.

Regions (aka Parts)

Regions are CSG operations (i.e. union, intersection, and difference) on non-region combinations and primitives.

Primitive Shapes (aka Solids)

Attributes

In BRL-CAD geometry database files, "attributes" may be used to store arbitrary information, i.e. metadata, on an object. Attributes may be applied to ***any*** object in the database.

display current attributes for object(s)	**attr show** ***obj ...***
set the specified attribute on an object	**attr set** ***obj atr val***
append the specified attribute value	**attr append** ***obj a v***
modify an object attribute(s)	**adjust** ***obj atr nval***
delete an object attribute	**attr rm** ***obj atr***
interactively set visual material properties	**mater** ***comb***
set object color (red, green, and blue values)	**comb_color** ***obj R G B***
get region identifier code for specified region	**whatid** ***region***
list all regions using particular shader(s)	**which_shader** ***shdr ...***
identify regions with specified air code(s)	**whichair** ***code ...***
identify regions with specified region id(s)	**whichid** ***id ...***
incrementally set region id on all regions referenced by object	**reid** ***obj #***
set material id on all regions referenced by object	**remat** ***obj #***

Scripting New Commands in MGED with Tcl

echo, i.e. display or print, the provided text	**echo** ***text***
pause for the specified amount of time	**delay** ***sec usec***
get combination CSG structure as a Tcl list	**lt** ***object***
use shell-style name globbing	**set glob_compat_mode 1**
use Tcl shell syntax evaluation	**set glob_compat_mode 0**

Here is an example of writing a custom command called '`get_primitives`' that traverses over all objects in a given combination, printing a list of all primitives encountered. For this example, glob_compat_mode is disabled (i.e. set to 0, not the default value of 1) so that there is no need to escape various characters with a preceding "\" slash.

```
set glob_compat_mode 0
proc get_primitives {object} {
  set children [lt $object]
  set prims ""
  if { $children != "" } {
    foreach node $children {
      set name [lindex $node 1]
      set data [db get $name]
      if { [lindex $data 0] != "comb" } {
        set prims [concat $prims $name]
      } else {
        set prims [concat $prims [get_primitives $name]]
      }
    }
  }
  return "$prims"
}
```


MGED Quick Reference Card version 4 for BRL-CAD version 7, June 2006

designed by Christopher Sean Morrison

MGED Quick Reference Card

(for version 7.x)

Starting & Stopping MGED

start MGED with default graphical user interface (GUI)	**mged**
run MGED in classic console mode	**mged -c**
open geometry database file creating new if necessary	**mged** ***file.g***
run a single MGED command on database	**mged -c** ***file.g cmd***
quit MGED	**exit** or **quit** or **q**

Files

Geometry database files in MGED are always ***automatically saved*** to disk after an edit is made. As such, performing a file "Save" operation manually is not necessary and is not provided by MGED.

open a new or existing geometry database	**opendb** ***file.g***
close any open geometry database	**closedb**
save a copy of the currently open database	**dump** ***newfile.g***
export objects from currently open database	**keep** ***newfile.g obj ...***
check if file contains duplicate object names	**dup** ***file.g***
combine a geometry database into existing	**dbconcat** ***file.g***
eliminate unused space from open database	**garbage_collect**
display version of currently open database	**dbversion**
upgrade currently open database to the latest	**dbupgrade**
import data file as a binary object	**wdb_binary -i u c** ***obj file***
export binary object to a data file	**wdb_binary -o u c file** ***obj***

BRL-CAD File Name Conventions

binary BRL-CAD geometry database files	**.g**
ascii BRL-CAD geometry database files (deprecated)	**.asc**
raw binary headerless 3-channel color image data files	**.pix**
raw binary headerless 1-channel grayscale image files	**.bw**
extended UNIX 2D/3D color plot format files	**.pl**
raytrace command saveview shell script (text) files	**.rt**

Geometry Naming Conventions

MGED imposes minimal restrictions on how geometric objects are named. It is up to the individuals and organizations to utilize consistent naming conventions when creating geometry. The below object naming suffix convention is frequently utilized and recommended.

groups / assemblies	**no suffix or .g**
regions / parts	**.r**
non-region combinations	**.c**
primitive solid shapes	**.s**

Constructive Solid Geometry Operations

Constructive Solid Geometry (aka Combinatorial Solid Geometry) is based on three mathematical boolean operations: union, intersection, and difference (aka subtraction). These operators are applied to primitives to form compound objects in MGED using the "u", "+" and "-" notation. Consider the example of combining two primitive object shapes, ■ and ●. The example below shows the resulting CSG combination object when the two shapes are overlapping.