

MGED Quick Reference Card

(for version 7.x)

Starting & Stopping MGED

To start MGED
Run in classic console mode
Run a single MGED command
To quit MGED

```
mged
mged -c file.g
mged -c file.g cmd
exit or quit or q
```

Files

Geometry database files in MGED are always automatically saved to disk after an edit is made. As such, the concept of performing a "Save" manually does not apply. Files use the ".g" extension.

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

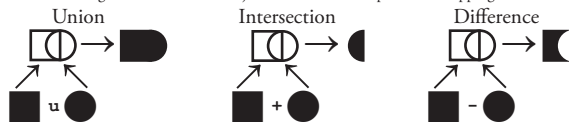
Getting Help

With non-classic MGED, right-clicking most labels and input fields will provide a description. Additionally, documentation is provided via the Help menu.

obtaining help on all commands	<code>help</code>
obtaining help on a particular command	<code>help command</code>
search for commands that relate to keyword	<code>apropos keyword</code>
display command history for current session	<code>history</code>
record transcript of commands used to file	<code>journal file</code>

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 "-" annotation. Consider the example of combining two primitive object shapes, \blacksquare and \bullet . The example below shows the resulting CSG combination object when the two shapes are overlapping.



Creating Geometry

interactively type in new object parameters
create a prototypical primitive object
create a CSG combination object

```
in
make type name
comb name op obj ...
c name obj op obj ...
r name op obj ...
g name obj1 obj2 ...
group name obj ...
```

create CSG region (aka "part") combination
create group (aka "assembly") combination

create a region from a range of solids
create a shallow copy of an object
create deep patterned copies of objects
rename an object
rename an object and all references
add a prefix to all references to an object
create an arb8 w/ rotation and fallback
duplicate a cylinder, positioned at end or orig
make a bounding box around object(s)
mirror an object about the x, y, or z axis
create arb given 3 points, 2 coords of 4th, and thickness

```
build_region prefix #
cp obj objcopy
clone
mv old new
mval old new
prefix prefix obj
arb rot fallback
cpi cyl cylcopy
make_bb name obj ...
mirror obj new axis
3ptarb
```

Deleting Geometry

MGED provides no 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	<code>kill obj ...</code>
delete object(s) and all references	<code>killall obj ...</code>
delete object(s), all sub-objects, all references	<code>killtree obj ...</code>

Geometry Information

list the top-level objects
list the objects in currently open database
get a table of contents for current database
display the information details for object(s)

```
tops
ls
t
l obj ...
cat obj ...
dbfind obj ...
tree obj ...
pathlist obj ...
paths pattern
showmats path
eac code ...
summary prg
idents file obj ...
title
units
geomtree
```

display combinations that reference object(s)
print out CSG hierarchy for object(s)
list all CSG paths that reference object(s)
list all CSG paths that match a pattern
show transformation matrices along a path
display all regions with given air code(s)
display counts of primitives, regions, groups
save region identifier summary to file
get/set title of currently open database
get/set units of currently open database
hierarchical geometry browser GUI tool

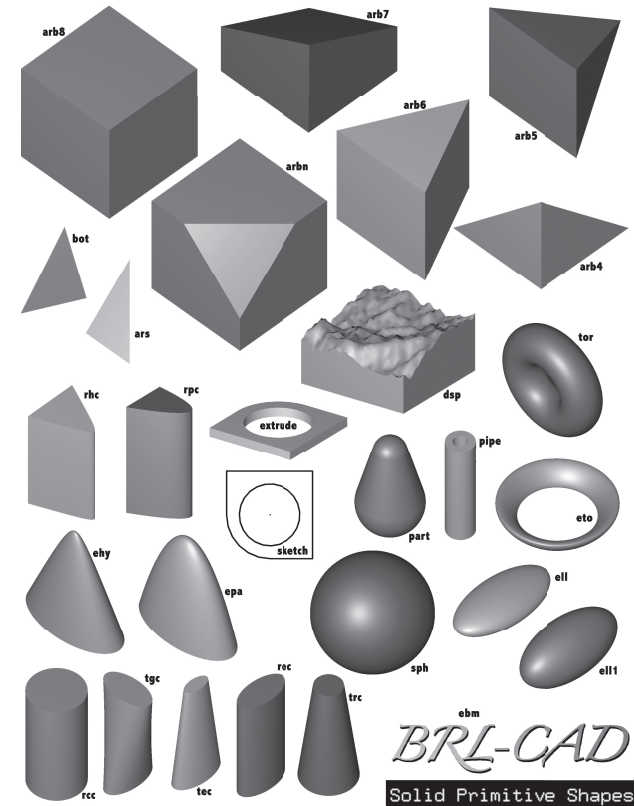
Displaying Geometry

display object(s) for editing
remove object(s) from the display
remove object(s) referencing specified object
remove all objects from the display
remove all objects and (re)display object(s)
mark object(s) as hidden
unmark object(s) as hidden

```
e obj ...
draw obj ...
d obj ...
erase obj ...
dall obj ...
erase_all obj ...
Z
B obj ...
hide obj ...
unhide obj ...
```

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 mode states or their behavior changes based on the state (e.g. the 'p' command).



visually illuminate & select combination
visually illuminate & select solid primitive
enter object-illuminate mode
get the current editing state
edit a primitive (enter solid edit mode)
edit a matrix (enter object edit mode)
add object reference to existing combination
remove object reference(s) from combination
set/get the center of editing transformation
manipulate an object's matrix or material
copy the matrix on one object to another
select matrix path when in pick mode
set a matrix on a given path
mirror arb face about the x, y, or z axis
apply all matrix transformations down to the primitives
same as push but creates new primitives as needed

```
ill comb
sill prim
press oill
status state
sed prim
oed lpath rpath
i obj comb
rm comb obj ...
keypoint x y z
arced path cmd
copymat path1 path2
matpick path1 path2
putmat path m0 ... m16
extrude face axis
push obj ...
xpush obj ...
```

The geometry editing commands below require you to be in an edit mode before they can be utilized. These commands implicitly apply to the objects currently selected for editing.

set parameter(s) for current edit operation	<code>p val ...</code>
return to viewing mode, rejecting any edits	<code>reject</code>
return to viewing mode, accept any edits	<code>accept</code>
edit the face of an arb interactively	<code>facedef face</code>

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 dx 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**
permute the vertices of an arb **permute 8vertices**

TRANSLATING GEOMETRY

move object being edited (relative position) **tra dx dy dz**
move object being edited (absolute position) **translate x y z**

SCALING GEOMETRY

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

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=epico

edit a combination using a text editor **red comb ...**
edit a primitive using a text editor **ted 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 ...**

Attributes

In BRL-CAD, "attributes" may be used to store arbitrary information, i.e. metadata, about an object. Attributes may be applied to any object (e.g. combinations or primitives) 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 an object's color **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 ...**

Manipulating the View

MGED uses a right-hand 3D Cartesian coordinate system where "up" is defined as the positive z-axis (+Z), "right" is the positive x-axis (+X), and "front" is towards the positive y-axis (+Y).

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**
turn perspective mode off (i.e. orthogonal) **set perspective -1**
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 current wireframe to a Postscript file **ps file.ps**
save the current view to a file **saveview file.rt**
load a saved view from a file **loadview file.rt**
save current wireframe to a UNIX plot file **plot file.pl**
overlay a UNIX plot file onto the display **overlay file.pl**

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**

Analyzing Geometry

analyze the faces of an ARB **analyze arbrname**
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 **query_ray x y z**

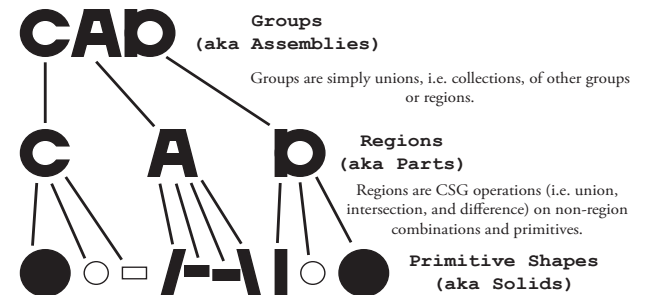
get/set query_ray behavior settings **vnirt x y**
check for overlaps (aka interferences) **vquery_ray x y**
compute view-dependent surface areas **qray**
get/set MGED calculation tolerances **rtcheck**
rtarea
tol

Naming Conventions

MGED imposes minimal restrictions on how 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.

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

autoview
refresh
ae az el tw
center x y z
eye_pt x y z
lookat x y z
size size
zoom scale
set perspective angle
set perspective -1
tra dx dy dz
sca factor
rot x y z
mrot x y z
vrot xdeg ydeg zdeg
qvrot dx dy dz angle
setview xdg ydg zdg
sv x y
orientation quat
knob params
adc
ps file.ps
saveview file.rt
loadview file.rt
plot file.pl
overlay file.pl



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.

Tcl Scripting New Commands Inside MGED

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**
run an external command **exec command**

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.

```
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"
}
```

Shell Scripting Outside MGED

Example shell script that displays the contents of all the top-level objects in the specified geometry database(s). each top-level object is ray-traced to a file and the visible surface area is computed.

```
#!/bin/sh
for db in $* ; do
    objjs=""mged -c $db tops -n -g 2>&1`
    echo "PROCESSING $db"
    for obj in $objjs ; do
        mged -c $db cat $obj 2>&1
        echo "Rendering $obj to ${db}.${obj}.pix"
        rt -o ${db}.${obj}.pix $db $obj 2>/dev/null 1>&2
        echo "Computing visible surface area"
        rtarea ${db} ${obj} 2>&1 | grep Area | tail -1
    done
done
```

Copyright (c) 2006 United States Government
MGED Reference Card version 1 for BRL-CAD version 7, April 2006
designed by Christopher Sean Morrison

Permission is granted to make and distribute copies of this card provided the copyright notice and this permission notice are preserved on all copies.