(2)

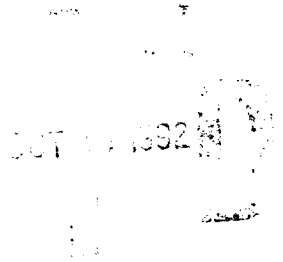
MEMORANDUM REPORT BRL-MR-4001

BRL

VULNERABILITY ANALYST'S GUIDE TO GEOMETRIC TARGET DESCRIPTION

CAROL A. ELLIS

SEPTEMBER 1992



U.S. ARMY LABORATORY COMMAND

BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

NOTICES

Destroy this report when it is no longer needed. DO NOT return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161.

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1992	Final -- August 1991-August 1992

4. TITLE AND SUBTITLE
VULNERABILITY ANALYST'S GUIDE TO GEOMETRIC TARGET DESCRIPTION

5. FUNDING NUMBERS
1L162618AH80

6. AUTHOR(S)
Carol A. Ellis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
BRL-MR-4001

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Target vulnerability assessment requires accurate information about physical target parameters. This information allows the analyst to draw experimental conclusions or model phenomena which are deemed predictable. At the Ballistic Research Laboratory (BRL), computerized descriptions capture the geometry of the physical target and are constructed using the Multi-device Graphics Editor (MGED), allowing interactive manipulation of solid geometric primitives and their boolean combinations. This approach, using Constructive Solid Geometry (CSG), allows the analyst to create shapes and refinements to the accumulated geometric data to fulfill the requirements for a wide range of analysis tasks. Geometry created using MGED can be interrogated using many of the tools incorporated in BRL-CAD software package. This handbook prepares novice analysts to appropriately address a number of geometric description problems. Details of the MGED editor are explained, manipulation of the primitives is explored with reproducible examples provided, and description database organization issues are addressed. A fully integrated approach to target description for use in vulnerability analyses is presented in a prescriptive format to facilitate any analysis task.

14. SUBJECT TERMS			15. NUMBER OF PAGES
CSG	Vulnerability Assessment	Targets	52
MGED	Solids Modeling	Vulnerability	16. PRICE CODE
BRL-CAD	Target Description		

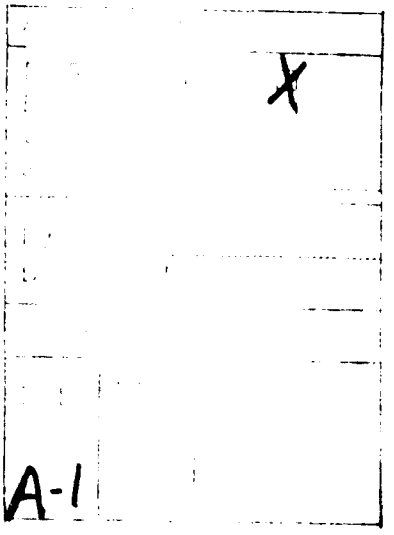
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

INTENTIONALLY LEFT BLANK.

CONTENTS

INTENTIONALLY LEFT BLANK.

LIST OF FIGURES



INTENTIONALLY LEFT BLANK.

1. INTRODUCTION

The mission of the Vulnerability/Lethality Division (VLD) of the Ballistic Research Laboratory (BRL) has traditionally included the assessment of combat vehicle vulnerability based on computerized geometric descriptions. In the early 1980's, as computer technology was rapidly changing and programmers found more sophisticated and powerful languages at their disposal, a new interactive target description tool was created at BRL. The package called MGED (Multi-device Graphics EDitor), the current evolved form of GED (Graphics EDitor), the first documented release of this software,[1] permits the creation of a target description on an interactively-controlled graphics screen, where solid geometric primitives and their boolean combinations can be viewed and edited to yield the shape and refinement necessary for the assessment task.

This approach, using Constructive Solid Geometry (CSG) to create a three-dimensional target model, yields a geometric data file that can be interrogated, using other analysis tools. The ultimate target information vulnerability analysts require is descriptive data delineating system performance in a specific scenario. Such vulnerability data allow analysts to act as designers, critics or supporters of a combat vehicle under study, with precise vehicle knowledge to back a studied opinion.

MGED and many interrogation tools for target descriptions are included in a large software package, written primarily at BRL, called BRL-CAD.[2] It should be noted that the BRL-CAD programs are not tailored specifically for combat vehicle vulnerability assessment work. MGED and other parts of this package are general purpose in nature and although they are perfectly suited for vehicle assessment they do not exclusively address this area of study. The BRL-CAD package, to include the MGED, is written in C programming language and functions in a UNIX-based operating environment. UNIX familiarity is quite helpful when learning the MGED, since some commands within the editor are terse, and syntactically similar to related UNIX commands.

The purpose of this report is to prepare novice combat vehicle target describers for some of the more difficult geometric problems facing them. The objective includes the preparation of analysts to understand the initial target

---

1. Michael J. Muuss, et.al., "GED: An Interactive Solid Modeling System for Vulnerability Assessments," ARBRL-TR-02480, March 1983.

2. Documentation for the BRL-CAD package is disseminated in release-specific form and resembles commercially available documents for other large-scale software. The documentation is not a BRL technical report, but can be obtained by contacting the laboratory.

analysis phase at which time descriptions are built. Although some documentation does exist for MGED and its use, this document presents the editor as the vulnerability analyst's tool. The handbook includes detailed examples that will clarify the use of various MGED commands, and suggest approaches to combat target description structure and organization.

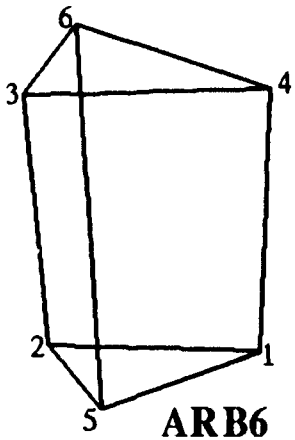
2. CONSTRUCTIVE SOLID GEOMETRY IN VLD

The essence of MGED is the ability to create complex, representative structures by combining primitive solids using boolean operations. It is, therefore, fundamentally necessary to understand the parametric definitions of the basic solids available to MGED users. CSG, as implemented in MGED, permits solid parameters to be specifically modified to define components, or groupings of solids to be manipulated as a whole. The distinction between types of editing procedures is facilitated by initial understanding of the solids themselves.

2.1 The Primitives There are four classes of solid types commonly used in building target descriptions. These are ARBitrary convex polyhedrons (ARB#) with 4 to 8 vertices, ELLipsoids (ELL), TORii (TOR), and Truncated General Conics (TGC). Each of these solid classes has specialized menu-driven parameter manipulation associated with it, permitting very refined definition of a solid's shape and size. Other solids, the ARbitrary triangulated Surface polyhedron (ARS), Half space (HAF), and ARBitrary N-sided convex polyhedrons (ARBN) are not commonly used, nor are they well supported for editing.

2.1.1 ARB# ARB4, ARB5, ARB6, ARB7, and ARB8 are the commonly used convex polyhedrons within the MGED framework. An "n" verticed convex polyhedron is defined as ARBN within MGED ($n > 8$) and is mostly used in military aerial target descriptions. ARBN will not be fully discussed here. Faces of each ARB must be planar and because this is true, parameter manipulation must be cautiously undertaken. A particular warning must be understood in this regard, since the MGED editor allows a user to create an "illegal" ARB. Editing of an ARB can result in a solid whose edges cross with the editor offering no complaint. Understand, however, subsequent utility codes balk at such errors, making visual and analytical validation of target solids a critical consideration. Figure 1 illustrates the five common ARBs and their associated edit menus. Note that certain ARB configurations permit edge editing while others permit point by point adjustment. Edge manipulation helps insure the planar nature of each face, changing other affected points within a solid to maintain this quality, however, the previous warning of crossing edges still applies.

2.1.2 ELL General ellipsoid definition permits manipulation of each of the determining vectors emanating from the solid's vertex. If each of these defining vectors are equal in magnitude, the resulting ELL is a sphere (SPH). The

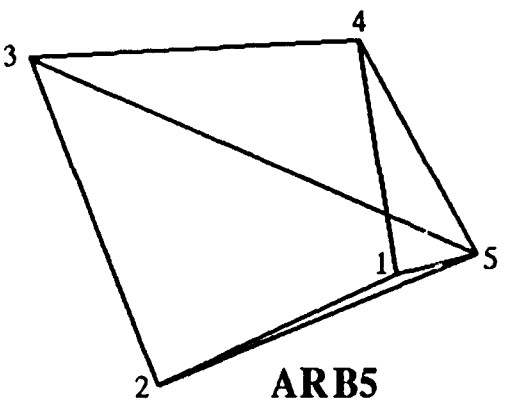


ARB6

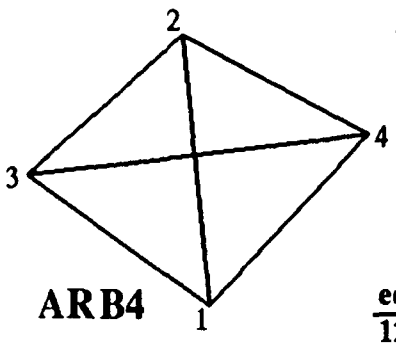
edges	faces
12	1234
23	2365
34	1564
14	125
15	346
25	
36	
46	
point 5	
point 6	

ARB Menu
move edges
move faces
rotate faces

edges	faces
12	1234
23	125
34	235
14	345
15	145
25	
35	
45	
point 5	

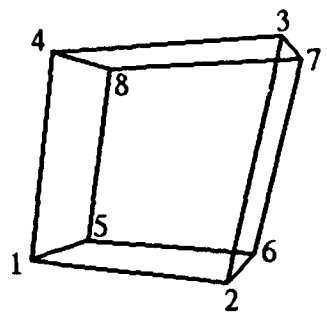


ARB5



ARB4

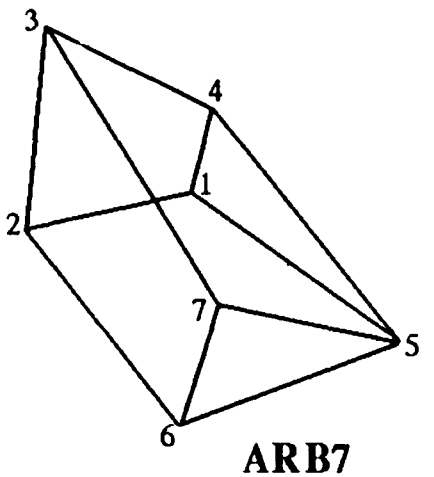
edges	faces
point 1	123
point 2	124
point 3	234
point 4	134



ARB8

edges	faces
12	1234
23	5678
34	1584
14	2376
15	1265
26	4378
56	
67	
78	
58	
37	
48	

edges	faces
12	1234
23	2376
34	
14	
15	
26	
56	
67	
37	
57	
45	
point 5	



ARB7

Figure 1. MGED Primitives - ARBs

concept of editing this solid type is less complex than the ARBs, and thus is readily and intuitively apparent. Figure 2 illustrates the ELL editing menu as well as the general definition of this solid type. Each of the vectors A, B and C may be scaled individually or as a group, as shown in the menu.

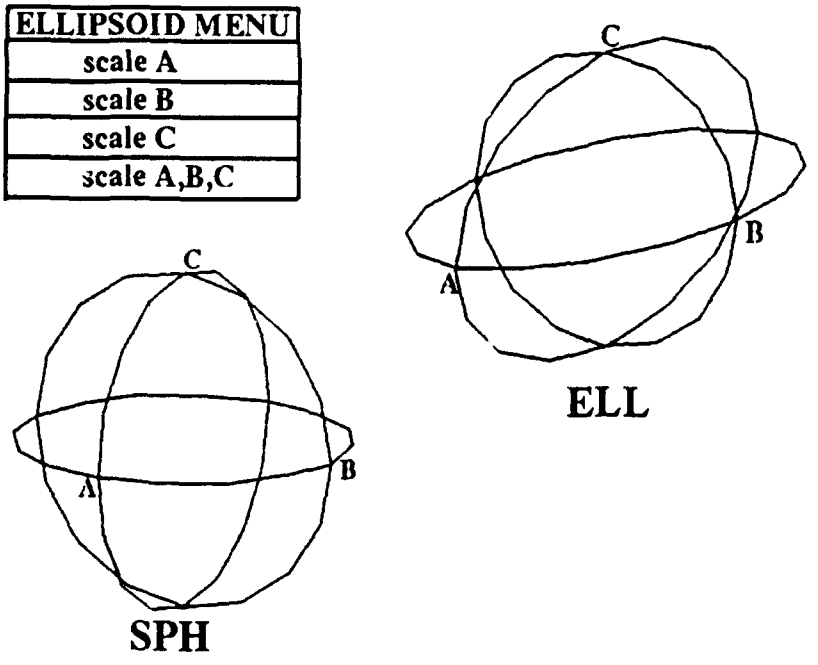


Figure 2. The MGED Ellipsoid Primitives

2.1.3 TOR The torus solid is more difficult to describe in text than to conceptually grasp. By definition, the TOR solid is a set of points encompassed by a circle spun in revolution about a point exterior to the circle. The only two parameters to be edited are the radius of the initial circle, r2, and the radius defined by the distance from the exterior point vertex to the locus of points determined by the centers of the revolved circle, r1. The definition is quite strict and the mathematics of the TOR is cumbersome, therefore, later discussion will describe some difficulties in viewing the TOR solid. Figure 3 illustrates the TOR

parameters and the edit menu available for this solid type. Note, N, is a normal vector to the plane determined by the locus of the centers of circular cross-sections of the torus. N, by definition, is required to define the orientation of the TOR solid.

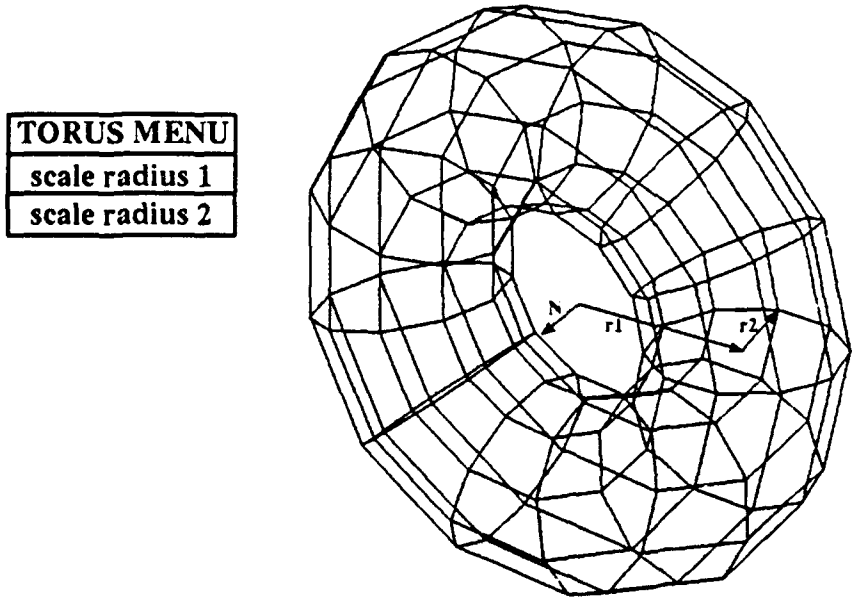


Figure 3. The Torus Primitive in MGED

2.1.4 TGC Truncated conics are commonly used in most MGED target descriptions. The generalized form of this solid is determined by two parallel bases, each defined by two perpendicular vectors, which are furthermore separated by a height vector. When the height vector is also perpendicular to the base determinant vectors, the result is a right circular cylinder (RCC) or a right elliptical cylinder (REC). The editing menu of the TGC is lengthy but intuitively clear. The only pertinent warning in dealing with the TGC is that the magnitude of any defining vector of the TGC must be positive, that is, a base vector cannot

collapse to a point, nor can a height vector. Figure 4 illustrates the TGC parameters and its edit menu, which permits scaling the height vector, scaling the base vectors individually, in pairs, or all four together. Bases may be rotated together, the height vector may be rotated as well, or the end of the height vector can be moved to a specified point, maintaining a right conic or skewing the presentation of the bases.

TGC MENU
scale H
scale A
scale B
scale C
scale D
scale A,B
scale C,D
scale A,B,C,D
rotate H
rotate AxB
move end H (rt)
move end H

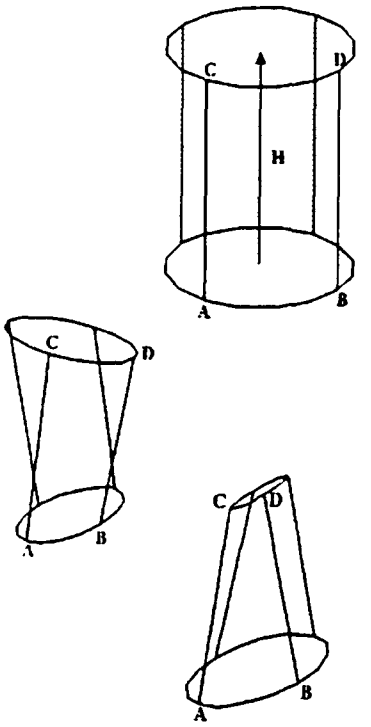


Figure 4. Conic Primitives in MGED

2.2 Regions A primitive solid alone is insufficient to describe the complex shapes encountered in target components. Combining solids using the three boolean operators permits a describer to artfully imitate the shape and form of complicated objects. The three boolean operators are: subtraction, intersection and union. Boolean operations are binary in nature, which means two solids are paired using the indicated boolean operator and the result is processed as a new

volume to be paired with the next solid and its specified boolean operator. Notably, however, a region can consist of a single solid, which implies the intersection of the solid with the universe. When boolean operators are used to define an object, the object is named, and becomes part of the database as a specialized object called a region. Regions reside on the second level of the tree, above the initial solid nodes, and begin the database hierarchy that is intrinsic to the MGED data file structure. A region can be as simple as a single positive solid or as complicated as hundreds of members combined with booleans. Within the BRL-CAD arena, regions are the lowest level of the data hierarchy that are evaluated for any post description creation processing. Understanding region creation and subsequent evaluation is a primary key to producing a useful. validated target model. Points of the following discussion regarding specific boolean operators are illustrated in Figure 5. Reference to this illustration will clarify the textual explanation of region creation.

2.2.1 Subtraction The result of subtracting two solids is all volume of the first solid less any common volume with the second solid. The operator, "-", signifies subtraction and is useful in hollowing a body, removing an odd shaped piece of a solid or accounting for edge intersections of walls, plates, piping or other "connected" solids. One important restriction of boolean use, that must be accommodated, is the convention that regions must begin with a positive body. If an initial solid within a region is associated with the "-" operator, the subtraction is ignored and the union operator is substituted.

2.2.2 Intersection The intersection operation, "+", combines two solids saving only their common volume. Unusual shapes can be attained using this operator and it is commonly used to "save" a piece of a shell as in, perhaps, a radar dish, or to use only a portion of a standard primitive in a component's definition. Intersection between two solids having no common points would, of course, be the null set which is a region having no evaluation potential.

2.2.3 Union The concept of union is the antithesis of intersection. The union operator, "u", joins solids so any volume in at least one is part of the resulting volume. The union operation allows several related parts of a single component, that overlap or trail one after another, to be defined in one region. Usefulness of the union operation is typified in creation of wiring harnesses, or even fuel or hydraulic lines. Some caution in using the union operator must be exercised, however. Regions created using this operation add to evaluation computer run times. Ramifications in post-creation editing of regions using union, also presents some problems. Further discussion of these issues will follow in subsequent sections.

2.3 The Nature of Regions Regions are typically the first-level hierarchy structure above the primitive solids. Region designation is a special kind of grouping mechanism not only in use of the booleans operators but also in

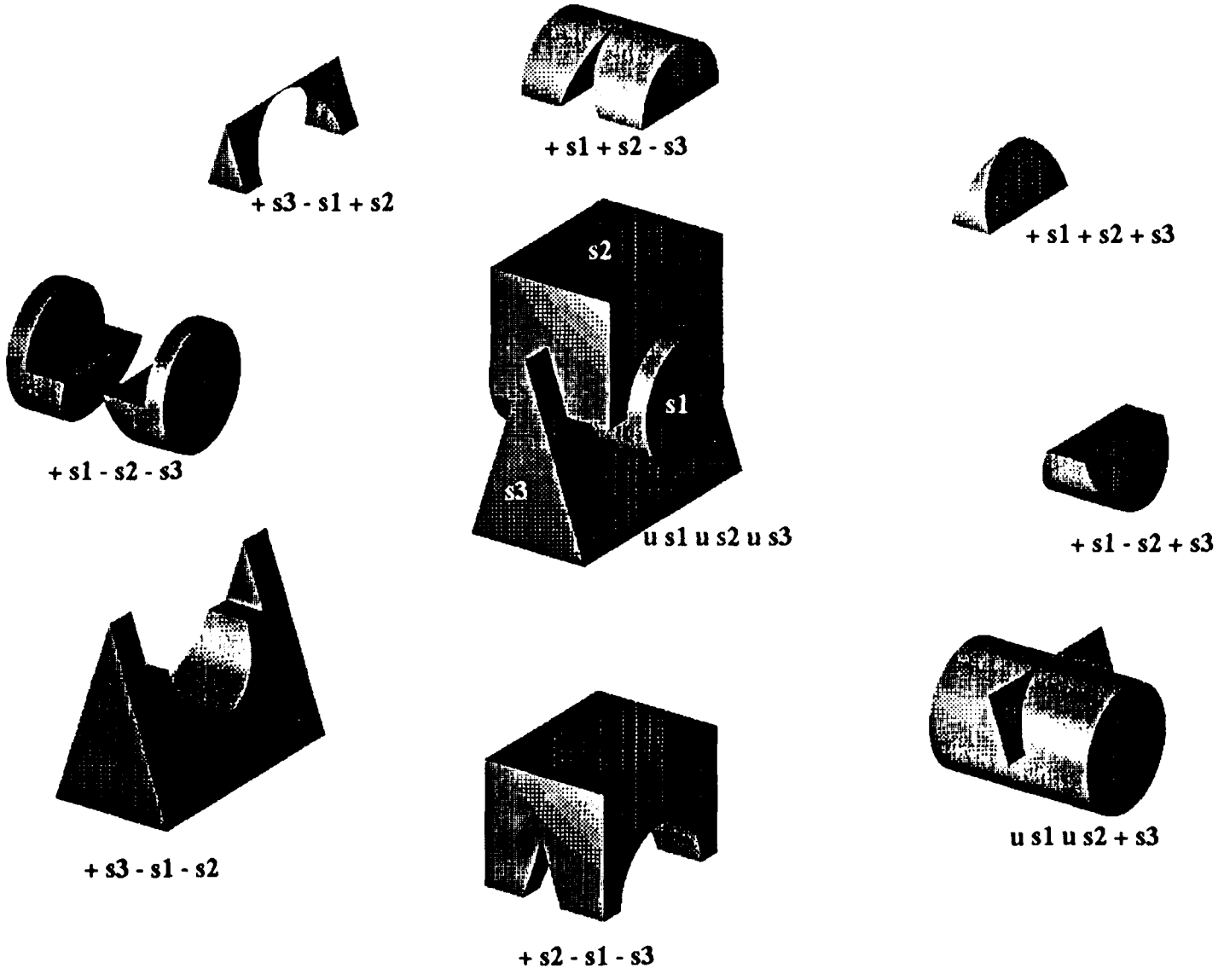


Figure 5. Defining Regions in MGED

allowing the describer to assign material properties to the group. Another "tag" that the describer can control in creating a region is the region identification number. Commonly called an item number, this designation within the region definition is closely linked to the subsequent analysis process for military target descriptions. The procedure for initial region creation is depicted in the following example:

Example - Creating a Region

> **Consider three solids, s1, s2, and s3. A combination of these solids will form region r1. The command** *r region-name op solid op solid op solid etc.* **generates a boolean combination of chosen solids which then represent a group labeled by** *region-name.* **The MGED editor's response to a new region's creation is:**
>
> **Defaulting item number to nnnn**
> **Creating region id=nnnn-1, air=0, los=100, GIFTmaterial=1**
>
> **So specifically,** *r r1 + s1 + s2 - s3* **, generates the region r1, includes it in the database, sets the identification number to nnnn-1 with the material type of 1 and line of sight density to 100%. In each editing session, nnnn, defaults to 1001 making the first item number in each session become 1000. The descriptive information defining regions is controlled by the describer, by setting default values for the descriptors. The effect of combining the solids as stated starts with the solid s1, intersects it with s2 and keeps the set of points describing the intersection. Next this intersection is further pared by subtracting points in common with solid s3.**

Later in this document, a discussion of material type and item number editing will give the describer an understanding of control in tagging the regions that are created. A final note on the creation of regions is that care must be taken to respect the blank spaces within this command. The operator symbols must be surrounded by blank spaces to be interpreted by MGED.

2.4 Creating Appropriate Regions Several issues must be considered in the initial construction of an MGED database. After describing initial solid parameters, region level combination is the natural second step. Being such special combinations, it is intuitively clear that complex regions can present special problems. In this section, some examples of region usage within the practical realm of military target description will help clarify the region combination genre.

2.4.1 Overlapping Regions A region is required by most application software to be wholely descriptive of the space it addresses, that is, it cannot overlap or be overlapped by any other region. Eliminating inadvertent overlaps within a target descriptions is part of description validation. Many software packages, using an MGED database as input, report the discovery of overlapping regions. Two regions reported as overlapping require the target describer to move solids, recreate solids or use booleans to resolve the problem area. An MGED command *rtcheck* , reports offensive regions from a group displayed on the screen. Overlap checking should occur at low levels of hierarchy, to insure a truly validated target. As low levels of the hierarchy yield clean checks, higher level groups up to the entire target must be evaluated for overlaps. Validation of the target description begins with this level of examination and continues to examine other aspects of the description package. Validation will be fully discussed later in this document.

2.4.2 Subtracting Regions When a very complex region is created, minor overlaps with adjacent regions can occur, but can be difficult to accommodate. Boolean subtraction of one region from another is sometimes the only reasonable solution. This approach to correction of overlap errors must be used judiciously and with the knowledge that added processing time is required for target descriptions containing such combinations. The following example demonstrates the need for region subtraction:

Example - Region Subtraction

Regions tray, tray.in and wall are defined as follows:

r tray u s4 - s7 - s5 + s8 u s6 - s5 + s8
r tray.in + s7 - s6 + s8
r wall + s9 - tray - tray.in

The definition of the region wall , is greatly simplified by the use of region subtraction. It is clear to see from Figure 6 that tray is an unusual shape and tremendous effort would be required to make a hole accommodating this structure without subtracting the previously defined region. Observe that both tray and tray.in are subtracted from the wall. Both subtractions are required in order to make a hole in wall, not just the cross-sectional imprint cut away from the region. Target describers are reminded that the use of region subtraction may add to the time needed to process the model, but in some cases, as this example depicts, clarity and simplicity dictate the use of the region subtraction tool. Users will see a message, when combining a region with a

region, reminding the user that the object paired with the operator is, indeed, a region. This note serves to check the describer's intention, as well as, denote the complexity inflicted on later processing of the description. Regions may also be combined using the intersection boolean but union combination is not permitted and, in fact, generates an error message.

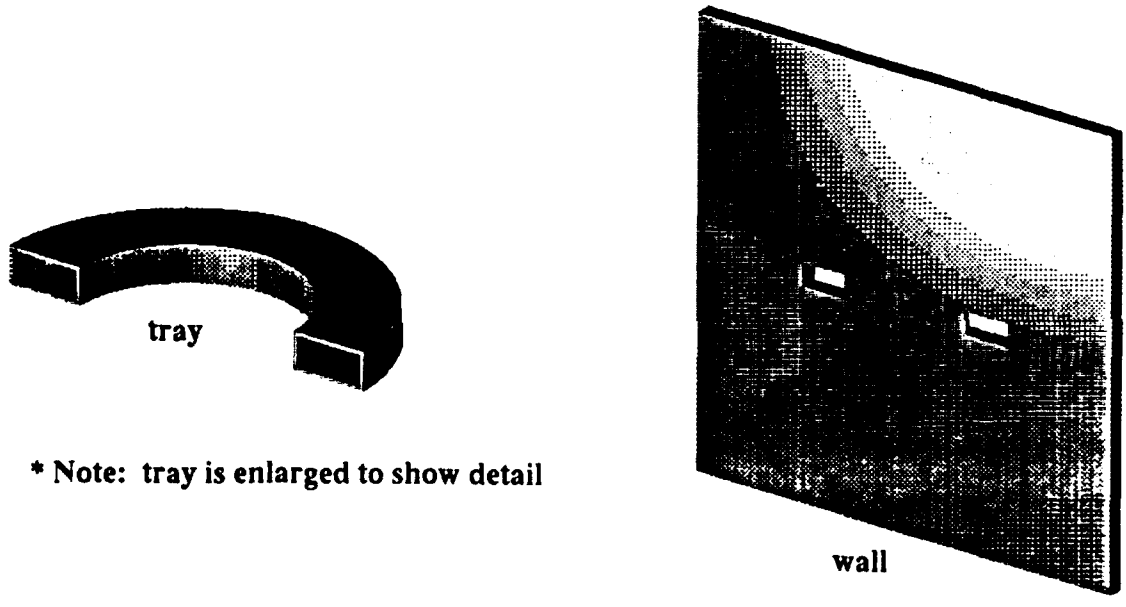


Figure 6. The Use of Region Subtraction

2.4.3 Using the Union Operator Another complicating factor of region creation is the use of the union operator. Though it is advisable to use this operator sparingly, some regions model such a complex structure that the union operator is the only sensible choice. Usually when union is invoked, that same structure can be described in two or more regions without losing the logic of the describer's approach. Occasionally, in the target description process, a difficult

component must be modeled and use of the union capability permits the describer to keep pieces lumped together that might not be obviously linked for the next user of the description. Precedence is an important factor in understanding the union operator. Consider:

r region1 u s1 - s2 u s3 - s4

This expression is evaluated as:

region1 = u (+ s1 - s2) u (+ s3 - s4)

This conclusion reiterates that regions described with the union operator can be expressed as two or more regions without that operation. In this example, the result is:

region1 = + s1 - s2 region2 = + s3 - s4

Another caution in using the union operator is that areas of a component identified as one region should be of reasonably homogeneous material type. Yet another issue, in the use of union, is the complication that occurs in correcting overlapping regions. If a solid from region r1 is violating a small portion of the space from region r2, and r2 uses the union operator, care must be taken to subtract the offending solid from the proper portion of r2. The following example illustrates the procedure currently used to decompose then recompose a region that requires editing in validation of the description. Note that non-standard MGED procedures exist to edit "nodes" of a target description as an ASCII file. The value of such a feature permits simpler corrective methods for union oriented descriptions by permitting the insertion of booleans within the existing region assignment. Look for a node editing feature in future BRL-CAD releases.

Example - Editing a Region

Consider two regions that have a small overlap.
The regions' definitions are:
r r1 u s1 - s3 - s5 - s6 + s8 u s2 - s3 + s5
r r2 + s9 - s4 - s2
Region r2 overlaps the first segment of r1.
After viewing the offending region, decision to subtract s9 is made.
Solution:
rm r1 s2 s3 s5
r r1 - s3 - s5 - s9 u s2 - s3 + s5
Notice that the removal of members s3 and s5 from region r1 removed their invocation in both unioned parts of r1. When the region is reconstructed, s3 and s5 must be subtracted from the first union segment of the region, then the second union segment is reintroduced.

2.5 The Hierarchical Structure When a target description is initiated, solids are manipulated in three dimensional space, regions are created using the solids to define more complicated volumes, then regions are grouped to establish further complexity leading, with more levels of grouping, to an ultimate group of groups that represents the entire target. The process of creating such a hierarchy is intuitive, but the methodology requires planning and forethought.

2.5.1 Building a Tree Grouping solids into representative regions is a special combination task that uses the boolean operations discussed in the previous sections. After sub-components or even components defined as regions exist, they must be further grouped to define sub-systems, then systems, of the target with a final apex group defining the entire target. Figure 7 demonstrates the basic idea of this kind of tree structure. Although the true leaves of the tree are the solids themselves, the describer must be aware that applications generally recognize the region level as the determinant level of hierarchy. This approach is a result of the qualitative, "special" information that tags the region combination but is not accessible in the more generic combination, group. Any solid formation that is to appear in the target, must be associated with a region. Careful consideration must be given to names chosen for groups within a target description, with an obvious focus on the functional breakdown of the target model.

2.5.2 Functional Grouping In creating a target description and especially adding new solids to build new regions, it makes most sense to group physically close regions to view potential overlaps and avoid surrounding structures. This sort of grouping should be regarded as temporary and useful only to the original describer. An important point to remember is that a solid, region or even a group can appear in more than one combination for the purposes of viewing a selected screen of objects pertinent to the editing currently at hand. Final organization of a target description, however, should be divided along functional boundaries. Increasingly complex levels of the hierarchy must be labeled clearly, and ideally represent a functional area of the target. Groupings, as commonly defined in U.S. military vehicle parts manuals, suggest a detailed listing of components that should be found in the target description. Adopting this or a similar approach to target organization, allows straightforward reference to a description and ease in using the description for subsequent analysis efforts.

3. MGED COMMANDS

MGED has a lengthy command list delineating the various powerful capabilities of the editor. During an editing session, typing a *?* or *help* will produce a listing of MGED commands. An experienced user will have little trouble locating the desired command from such a listing. A novice user, on the other hand, has a bigger problem deciphering the list since it is presented in

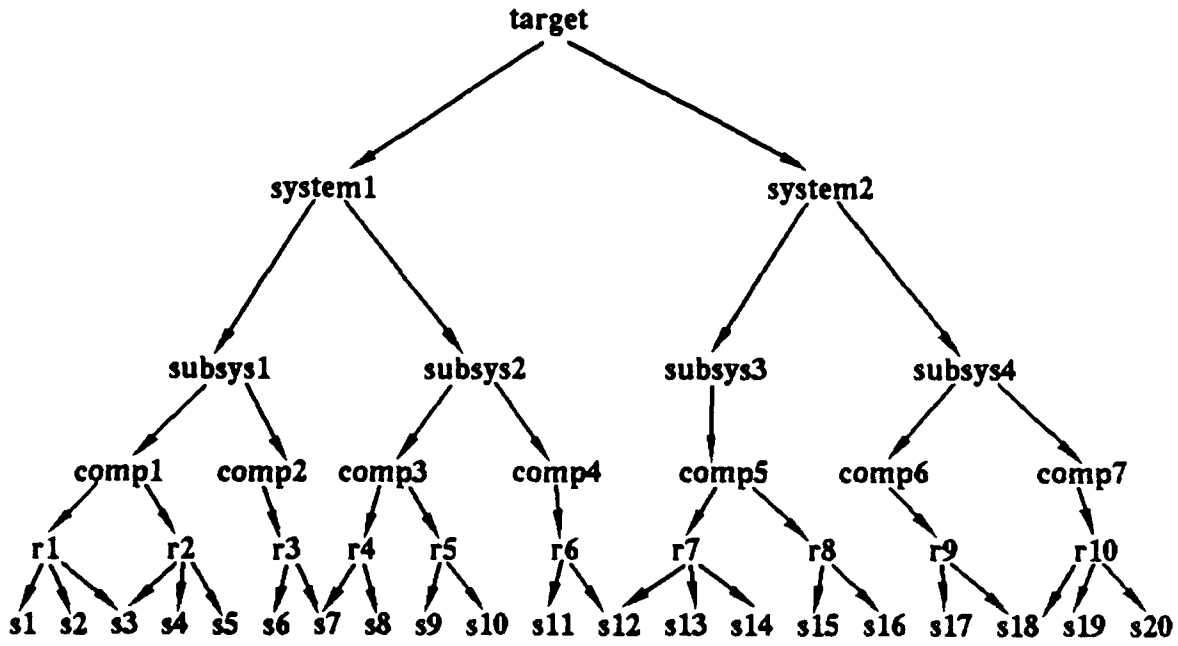


Figure 7. An Example Tree Structure

alphabetical order with no functional organization to the compilation. This section lists the MGED commands alphabetically within functional type and provides a usage message for each command. Following this compendium, a beginner MGED session will be offered, directing a new user through initial manipulations. A more advanced, but essential, section on troublesome MGED commands ends this portion of the handbook, with tips and warnings for users creating more complex targets as their skill improves.

3.1 Display Functions These commands control the user's screen by directing view aspect and actual assignment of objects to the display list. This group of commands also includes little used commands that simulate dial and button operations for hardware without this equipment.

adc (a1|a2|dst|dh|dv|dx|dy|dz|hv|xyz|reset|help) value(s)
control precisely the angle/distance cursor

ae azimuth elevation
rotate view using az and el angles

attach (nu|tek|tek4109|ps|plot|sgi|X)
attach new display processor

B obj1 obj2 ... objn
clear screen and display objects

center x y z
set view center in target coordinates

d obj1 obj2 ... objn
delete objects from the screen

e obj1 obj2 ... objn
display objects on the screen

E obj1 obj2 ... objn
display objects evaluating regions

fix
restart the display after hangup

in name type
directly enter new solid parameters; prompted for input

knob choice value
emulate knob twist

make name type
create and display a primitive

press button-label
emulate button press

refresh
send new control list

release
release current display processor; attach null processor

saveview file
saves current view in file for RT run

set [var=opt]
assign/display the MGED window control variables

size value
set view size

status
print view status concerning screen coordinate axes

vrot xdeg ydeg zdeg
rotate view

x
debug list of objects displayed

Z
clear the screen

3.2 Selection and Editing This set of commands allows the user to perform certain editing functions when manipulating a specific solid or altering a group of objects. This group. just as the previous set of functions, also includes commands that permit minimally configured hardware to use the MGED package.

arb name rot fb
make arb8 with specified rotation and fallback
copyeval
copy a solid evaluated down a specified path; prompted for input
cp oldobj newobj
copy old object to new object
cpi oldtgc newtgc
copy an old tgc to a new tgc while placing new base vertex at old tgc end
edgedir delta_x delta_y delta_z
define direction of an ARB edge being moved
eqn A B C
set planar equation coefficients
extrude #### distance
extrude a specified ARB face
facedef ####
define new face for an arb
i object instname
create an instance of an object
ill name
illuminate object without use of mouse
inside
find inside solid; prompted for input
mirface #### axis
mirror face #### about specified axis
mirror oldobj newobj axis
mirror an image of an object about an axis
p dx [dy dz]
precise parameter or delta definition for solid editing
rfarb
makes arb8 given a point, rotation and fallback angles
rotobj xdeg ydeg zdeg
precisely rotate an edited object
scale factor
precisely scale an edited object

sed solidname
solid edit a named solid
ted
text edit the currently selected solid's parameters
track
builds a "plate" track given wheel data; prompted for input
translate x y z
translate an edited object to place the key point or key solid at x y z
3ptarb
makes an arb8 given 3 points and thickness; prompted for input

3.3 Listing and Manipulating Hierarchy Some commands of MGED permit the user to assemble, destroy or extract from the data hierarchy, allowing the tree to be built representing the desired target. The following commands are useful beyond the initial solids' editing and help an analyst segregate and associate target components as necessary.

analyze solid
print copious information about a named solid
comb comb_name op1 comb1 op2 comb2 ... opn combn
create or extend a combination using booleans
concat file {prefix}
cats named file onto current file using optional prefix for unique naming
dup file {prefix}
lists duplicate names between file and current file using optional prefix
edcomb comb flag item air mat los
edit combination record information
find obj1 obj2 ... objn
finds all references to an object
g groupname obj1 obj2 ... objn
combines objects under a groupname
idents file obj1 obj2 ... objn
makes ASCII region identification summary in file
keep file.g obj1 obj2 ... objn
keeps objects in separate file.g; objects not removed from current file
kill obj1 obj2 ... objn
remove objects from the current file
killall obj1 obj2 ... objn
remove objects and any references from file
killtree obj1 obj2 ... objn
remove complete branches from hierarchy
l(cat) object
list object information

listeval
gives evaluated path summary; prompted for input
mv oldname newname
rename object
mvall oldname newname
rename object all references to an object
paths
lists all paths matching input path; prompted for input
prefix string obj1 obj2 ... objn
prefix listed object names with string
push obj1 obj2 ... objn
push object level transformations to solid parameters
r regname op1 obj1 ... opn objn
create or modify a region
regdef item [los air mat]
set default codes for next region created
regions file obj1 obj2 ... objn
make ASCII regions summary in file
rm comb mem1 mem2 ... memn
delete members from combination
solids file obj1 obj2 ... objn
make ASCII solids' parameter summary in file
summary [s/r/g]
list solid, region or group summary
t(ls) object
list table of contents
tab obj1 obj2 ... objn
list objects as stored in data file
title newtitle
change title of description
tops
list all top level hierarchy objects
tree obj1 obj2 ... objn
list hierarchy tree for objects
units [mm/cm/m/in/ft]
change output units of data file
whichid ident1 ident2 ... identn
list all regions with given ident

3.4 Pre-analysis checks and general commands As a target description project comes to an end, a final check of file organization is prudent. As previously mentioned, the target must also be validated to check for overlapping regions and visually surveyed for reality matching. This group of

commands facilitates these activities and includes a few commands of general use.

area [tolerance]
find presented area of displayed evaluated (E'd) objects

color low high r g b string
assign color to item range, low-high, and label with string

edcodes obj1 obj2 ... objn
screen edit region idents, material and los codes for listed objects

edcolor
text edit the color/item assignments

item region item air
change region item/air codes one region at a time

mater comb [material]
change combination material properties

nirt
trace a single ray from current view

overlay file.plot [name]
read a UNIX plot file as a named overlay

plot [-float] [-zclip] [-2d] [-grid] [out_file] [\|filter]
make 3-D UNIX plot of current view

prcolor
print the current item/color assignments

q
quit current session of MGED

rt [options]
raytrace current view onto framebuffer

rtcheck [-s res object]
check current view for region overlaps

sync
forces UNIX sync

%
escape to shell

?(help)
give short(long) list of MGED commands

3.5 Beginning use of MGED Actually describing a complex target using MGED may seem to be an overwhelming task. A simple breakdown of beginning procedures and a further explanation of some basic terms will make such an endeavor more plausible. Consider, first, the term object. Any creation, at any level of hierarchy, can be thought of as an object. Thinking of object, as the language in which MGED converses, allows more specific terms like solid and region to be more narrow in meaning. A solid is any basic primitive, a combination is any group of objects, but a region is a special combination that is the only evaluated object for raytracing. Every MGED path must contain a

region in order to be processed by the raytracing library. A solid at the bottom of a hierarchical path that has not been combined in a region, will by default be made into a region containing that one solid. Another important point concerning the creation of any objects in an MGED database is that they must be uniquely named. All references to objects are made by name and, as such, must be uniquely identifiable. The editor checks the database for uniqueness of name in most cases. A mistaken reference with an erroneous, but unique, name will not be denoted to the user since intention for the name call cannot be intuitively implied.

Raytracing is another basic concept of BRL-CAD and target vulnerability analysis at BRL. Raytracing is a target description interrogation technique that reveals information about objects encountered along a ray fired, from any aspect, at the target. Cumulative information from this process, implemented at some user-defined spacing across a chosen planar view, can be used for such applications as signifying penetration capabilities of a given threat or producing a color shaded image of objects first encountered on each ray. Figure 8 demonstrates the concept of raytracing. Notice the grid pattern of uniform height and width, which is superimposed at a standoff from the target surface. Generally, a random point is chosen in each cell from which a ray is fired through the target. These rays, representing interrogation from a given aspect, are parallel and are perpendicular to the aspect-determining plane. Another term used in the vulnerability community is shotlining. Shotlining is raytracing but carries the connotation of being a vulnerability analysis technique. Raytracing may also be referenced by a pixel resolution specification. Commonly, squared views of 128, 256, 512 and 1024 pixels demonstrate increasing resolution for a raytraced image without distortion.

When first invoking MGED to create a new database, the user types

MGED file.g,

and the editor responds with

file.g: No such file or directory
Create new database (y|n)[n]?

and the user types a *y* which generates a new file for compiling the target information. Several factors must be considered during the initial description session. The new database is untitled even though the file containing it is named. The *title* command allows the user to assign a proper label to the new description and even to identify the database with a particular describer or a certain date. Another prescriptive command that most users initially issue is *units*, which allows the describer to choose the target units that suit the new file. Numbers are

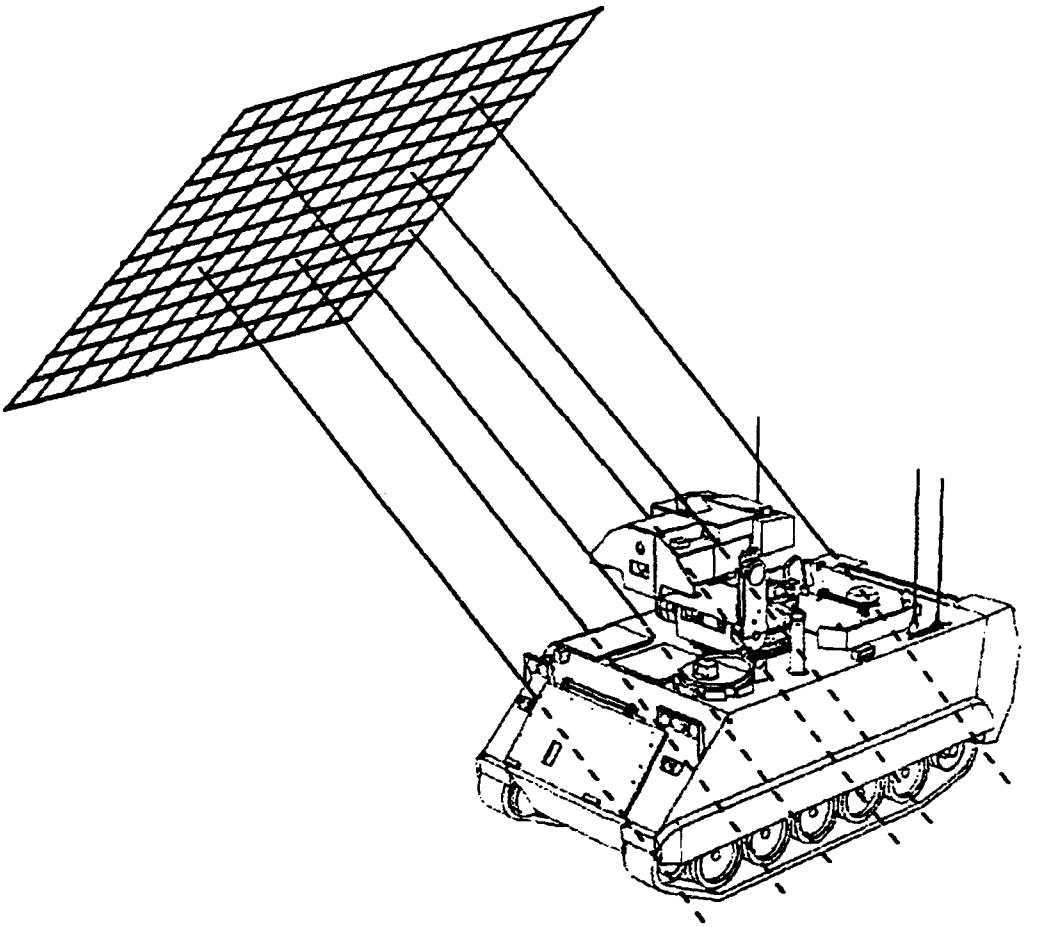


Figure 8. Raytracing a Target

always stored in the MGED database in millimeters, so the choice of units defaults to millimeters. If, however, the drawings and references are in inches, the describer can work in those units and the designation of this fact will filter the millimeter stored units to proper screen units.

When beginning a target description, primitive solids are the only objects that will be edited since the database contains little complexity at this point. In order to create a solid of a specified primitive type, the describer uses one of the

display function commands, *make, in, cp, inside, cpi, arb, copyeval, mirror, rfarb,* or *3ptarb,* depending on the desired new solid. The syntax of these commands is explained in previous sections. Practice and experimentation will give the user a better feel for the choosing "how" to create a solid. Solid editing, itself, is a reasonably intuitive experience. Controlling the magnitude and direction of chosen parameters gives the describer power to form the solid shapes desired. Choosing a solid to edit is accomplished by ensuring the solid is on the screen, then selecting it from the display list. The exact selection, on most hardware, involves selecting solid illumination then, sliding a mouse up and down a delineated pad where the size of the screen area devoted to each solid depends on the number of displayed objects. When the selection is made, an associated edit menu, as previously described, appears. Along with the edit menu, the graphics screen has a set of mouse selectable controls available for general functions. Figure 9 illustrates the general menu as displayed on current hardware used at BRL. One realization about the button menu is its imitation of a subset of button functions that reside on many MGED platforms. One button selection, called *sliders,* permits the describer to slew, zoom or rotate the view items by "sliding" the associated label across the screen. To stop the function, a *zero sliders* also exists. A final note about controlling the display; three keyboard function buttons, f1, f2, f3, control the depth-cueing, z-clipping and perspective of the image on Silicon Graphics platforms. These functions operate in a toggle manner and the first lets close items appear brighter than distant ones, the second determines whether parts of displayed solids will remain in view as the user zooms in, and the third draws the view in perspective with the closest edge appearing the largest. Upon first entering the editor, defaults for f1, f2, f3 are on, on, off.

As the target description is compiled, more sophisticated editing may be required. Within MGED, the ability to edit a hierarchy at any level, is indicative of the true power of the editor. Suppose, for example, a description is being compiled representing a developmental vehicle. In the process, the customer directing the description work, moves the engine and its accessories back four inches in the vehicle. At first glance, it seems that this situation will require a great deal of description modification. However, providing the engine has been grouped appropriately, the entire group can be translated back that four inches, leaving only revalidation of the description to complete the move. Further details concerning this concept, termed object editing, will appear in the next section.

3.6 Troublesome Commands Some MGED concepts are more difficult to grasp in first exposure. The basic thought of parametrically controlling geometric solids to achieve certain shapes and sizes is clear enough but the more interesting concepts are those that make target describing a challenge. In this section many complicated editing issues are addressed to help the describer and analyst handle a target description. An important feature of many MGED commands is the allowance of string matching for commands that require or

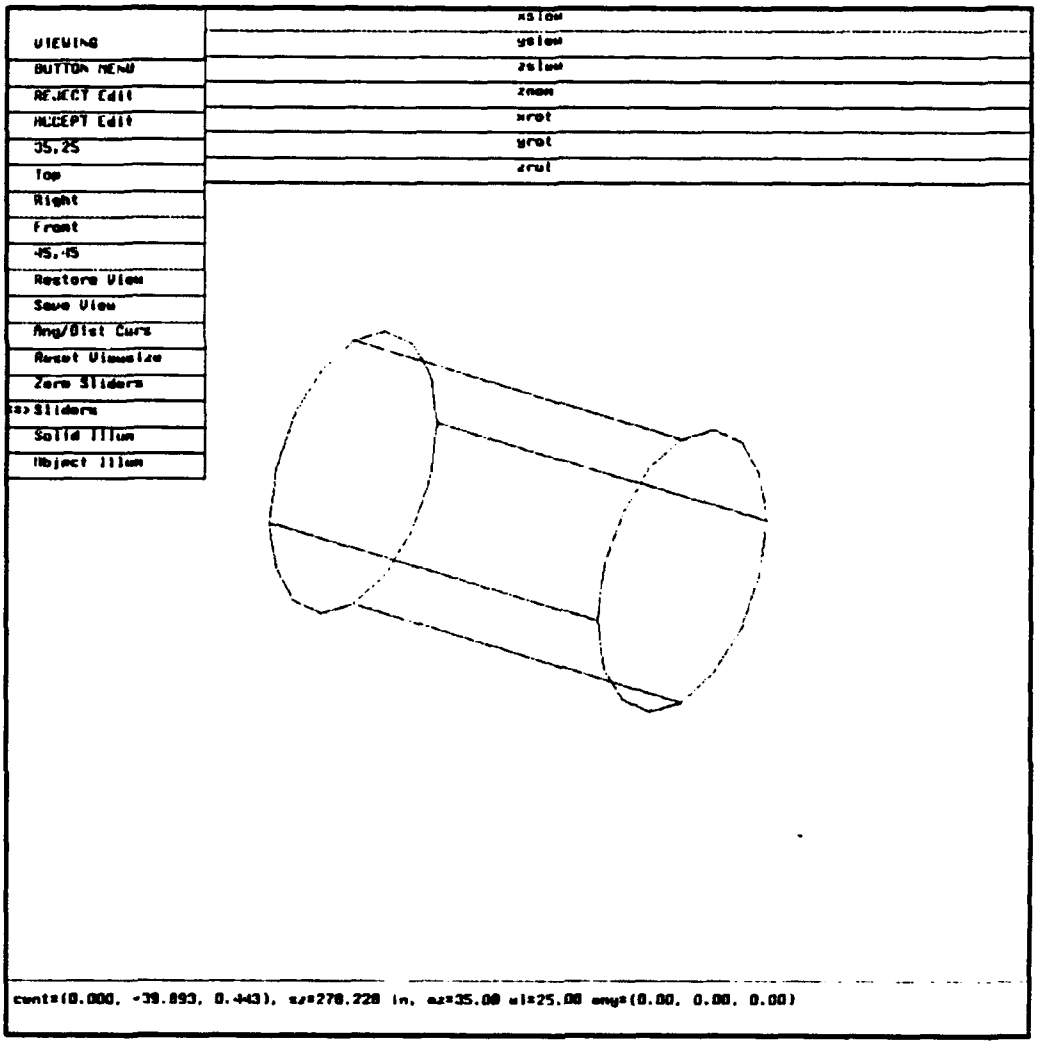


Figure 9. Sample MGED Display with Button Menu

accept more than one object in usage. While this facet of the command structure saves typing, care must be taken to avoid including stray objects on the command line.

3.6.1 Object Editing Displaying any object or group of objects for editing requires a choice between the *e* and *E* commands. The *e* command displays objects on the screen by traversing the paths in the object's hierarchy, accumulating transformation matrices at each level until the bottom of the path,

a solid, is reached. The solid's parameters are altered by the application of the accumulated transformation and the resultant solid is displayed. In the case of the *E* command, the transformations are similarly accumulated but only until a region is encountered on the path. The region then has the accumulated transformation applied to it then the *E* command further examines the booleans associated with the region. The region is displayed in an evaluated form, more closely resembling the actual object it represents, as a result of considering the booleans. Understanding the display process and the nature of the MGED hierarchy permits consideration of editing displayed groups in harmony. An important issue in object editing is the choice of path terminator as editing will occur about that focus. A path ending solid, when using *e* , takes on great importance since it becomes the key solid and its vertex or point 1 is the key point. With *E* , a region terminates the path and the physical center of that region becomes the key point. All object editing is done with respect to the key point. The following example depicts the concept of object path selection and editing.

Example - Object Editing

Consider a roadwheel on a tracked vehicle. Suppose the roadwheel has been misplaced along the side of the vehicle, and it must be moved -2.0 inches along the x-axis in order to match the pivot arm assembly. Moving each solid would be tedious and time consuming, but by using object editing this problem is relieved. The *e* display command, in this case, would be most effective since exact placement of a well selected key solid would be appropriate. Also required for display on the screen are any connecting solids that would confirm the correctness of the editing session. In the case of the misplaced roadwheel, the connecting piece that must match is a pivot arm assembly that holds the end of the torsion bar and connects to the roadwheel support arm. The support arm is a part of the roadwheel assembly. The first step in editing a group is the selection of the proper path to accomplish the editing task. Looking at rw.l.1, the assembly that must be moved, consider the listing that follows:

```
rw.l.1: rw.l.1 (len 5)
  u lwheel.in.1
  u lwheel.out.1
  u lhub.1
  u lhubcap.1
  u lroadarm.1
```

A tree of this group also follows:

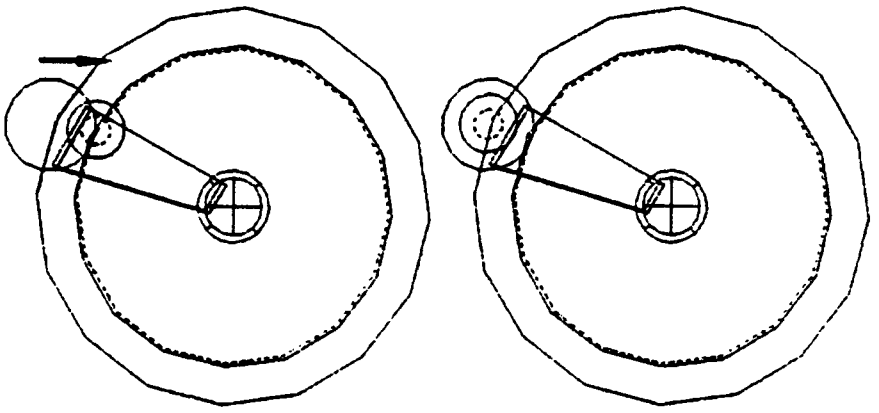
```
rw.l.1/
    u lwheel.in.1/
        u lrubber.in.1/R
            + lt1.in
            - s540
        u lrim.in.1/R
            u lt1.wh.in
            - s580
            u s540
            - rim.lt1
            + lt1.in
    u lwheel.out.1/
        u lrubber.out.1/R
            + lt1.out
            - s540
        u lrim.out.1/R
            u lt1.wh.out
            - s580
            u s540
            - rim.lt1
            + lt1.out
    u lhub.1/R
        + s580
    u lhubcap.1/R
        + lthub1.cov
        - s580
    u lroadarm.1/
        u larm.1/R
            + s581
            - s580
            - s579
        u ltorhub.1/R
            + s579
            - s505
            - tor10
```

The invocation of object editing is a two step procedure. After displaying the necessary objects, select "object edit" from the button menu. The initial "object pick" state allows the user to edit based on an appropriate key solid or key point. Selection of the desired path is followed by choosing the level of editing. Observe the movement of the matrix label as the mouse is moved

up and down the screen. When the matrix resides at the hierarchical level that will affect the entire group, select that level. The viewing screen will display the chosen path in the bottom margin and editing can begin. Object editing includes scaling, translation in x only, translation in y only, translation in all axes, rotation, scale in x only, scale in y only, and finally scale in z only. An important realization is that movement in X and Y, as object editing choices, are based on screen X and screen Y movements. This means that X and Y truly represent the "horizontal" and "vertical" with respect to the current view. If the current view is a RIGHT view, for example, X move would affect translation in x and Y move would alter the translation in z. Experimentation with this concept will clarify this precept. For this editing task, choose a LEFT side view then select "X move", as the roadwheel assembly needs only translation in x when displayed from a LEFT view. Figure 10 demonstrates the results of the editing. The roadwheel can be placed precisely by typing *translate x-coordinate, y-coordinate, z-coordinate* **and the movement will be based on the absolute change of the key solid's point 1 or vertex and the relative translation of all other solids with respect to the key solid. One definitive note, however, must be taken with regard to the impact on the actual solid parameters. There is no change in the solid record database as a result of any object editing. If the describer requires the resultant display to be reflected in the solid parameters, using the** *push* **command can make this change to the database. The** *push* **command has some restriction and will be discussed in later text.**

It should be noted that the *E* command does not evaluate regions whose members are other than solids nor can it display an evaluated TOR, ARBN or ARS.

3.6.2 Push With a full understanding of the hierarchy structure, the notion that a matrix resides at each branch location and affects the sub-tree beneath it, is apparent and intrinsic with respect to MGED hierarchies. When a section of the hierarchy is affected by a non-identity matrix, it is displayed on screen at a location removed from that prescribed in its member solid records. In general, the containment of the transformation information within such matrices has no effect on the accuracy or assessability of the target. However, when simple movement of a group into the desired location is not reflected in the solid parameters, it makes the description more complicated for follow-on users. It is generally suggested that the *push* command be used following object editing sessions. Some warning and caution should be noted when using this command, however. In the event that the edited group contains a copy or instance of



Before editing **After editing**

Figure 10. Object Editing

another group, a conflict occurs because the solid records are utilized at least twice. The same situation occurs when a solid appears in two different regions that have been edited separately and result in two different transformation matrices attempting to manipulate the same solid. A complication of this command can occur if a solid record is transformed by push but that same solid is mentioned outside the group. The result here is an unexpected shift or movement in the external mention of the solid since the solid record appears in the database exactly once, under the given name. Target describers should be aware of the impact of the push and use in carefully. A recommendation would be creating only a "pure" group to push, that is, a group that has no external solids in any member region and donates none to external groups until the "pushing" is completed. Instancing is a method of referencing a prototype object that is different from its neighbors only in its placement within the description. Instancing will be discussed later in the text. Another issue that must be understood to clarify the push command is that the push must be invoked at a group above the point where matrices have been altered. Figure 11 illustrates this point.

3.6.3 cpi This command is very useful in adding wires, fluid lines or any repetitive conic to a target description. The essence of the command is to take the immediately previous conic, oldtgc, copy it to newtgc, placing the vertex of newtgc at the previous solid's end. The final step of *cpi* puts the editor into solid edit with newtgc illuminated and selected for editing. This command speeds the

If subsys1 is selected for editing as a member of system1, the push command must be executed at the system1 level to relocate the editing results at the solid parameter level. The ** shows which matrix has been changed.

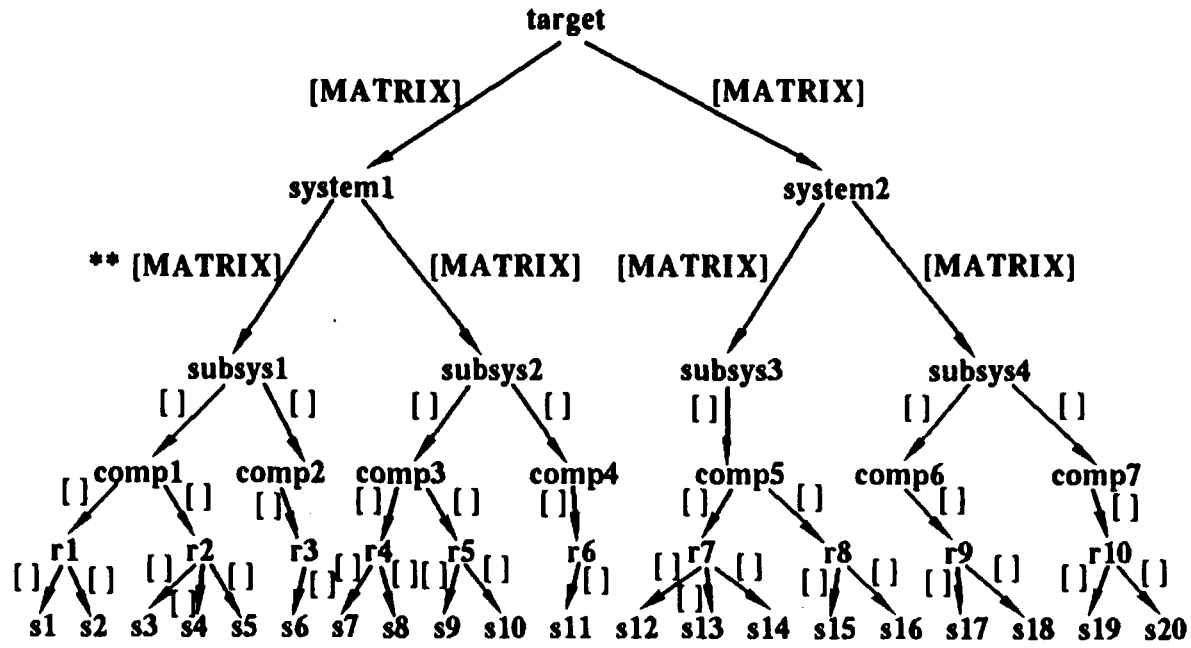


Figure 11. Pushing Object Editing to Solid Level

description process as the describer winds wires or lines around existing target objects, carefully avoiding overlapping regions. Frequently, solid manipulation can be done with the mouse and efficiency of this modeling task is maximized.

3.6.4 r The command that creates the special combination record, region, is *r*. As was previously explained, regions are records, whose members are combined using booleans, and are the combinations that are interpreted by the raytracer. Regions also contain other tagging information useful in analyzing target descriptions. When a target description is created, default values are given to the item number, air code, material type and percentage associated with a region. The user controls the defaults, but in practice, the describer usually edits a group of regions within a larger group by using the *edcodes* command. The invocation of *edcodes combination* displays each region's information in a screen editor format. The cursor appears on the first line of region information, poised over the item number. The describer can then change the number by typing a new item number for the region whose name appears at the end of the editing line. A space bar skips to the next data piece, the air code, which can also be changed. The space bar cyclically runs through the item, then air code, then material type, and finally percentage, also termed los (line of sight), until the describer is happy with that line. The carriage return lets the next region in the group be listed for editing and the process continues until all regions within the group have been listed. The one drawback to this editing process is that the user cannot back up a line during the editing process. If an error is made, invoking the edcodes command with just the one region name allows the user to edit just one region's information. A important point regarding changes in item numbers when regions or higher groups have been copied or instanced is that changes in item numbers or material characteristics may be recorded, but a singular region record is changed time and again since the same region record is multiply referenced. The latest editing will remain recorded in the region information. This situation and its drawbacks merits further discussion in the section on instancing.

3.6.5 kill and rm The concept of eliminating or removing something from an MGED database is as natural as understanding the need for creating the object originally. Especially when inheriting an MGED database from a previous analyst, the need for molding the database to the use at hand requires the ability to remove, or recombine, portions of the original. Killing a piece of the description mentioned in more than one group is accomplished by the *killall* command. This command searches the database for all references to the specified object, which can be a solid or combination record, and permanently deletes it from the database. If a region, for example is destroyed using *killall*, that combination and all references will be wiped from the database. It should be noted, however, the solid records that comprised the region will not be killed. To complete an action like this, the *killtree* command is used. This command

destroys the combination specified, and all records below it. Extreme caution should be used before invoking this command. The *tree* command will list for the describer all records that will be killed with the *killtree* action. More simply, the *kill* command destroys the record specified, permitting the user to recreate it or continue without it. The *kill* command does not remove any reference to a "killed" record, so again, caution is advised. The *rm* command allows a user to delete a reference to an object but the object remains in the database. This command is especially useful when editing regions where order of booleans is important (union).

Example - Killing and Removing

Imagine the tree structure in Figure 7. If the describer wanted to kill subsys1, the *tree* command would demonstrate that solid s7 would be killed using *killtree*. In the combination r3, use of the *rm* command will permit the solid s7 to be deleted from reference in r3 but will not alter its reference in r4. After this check, subsys1 can be safely removed from the database using *killtree*.

3.6.6 Summary Tables MGED enables the describer to summarize the production of solids, regions or regions ordered by item numbers. The obligatory note of care for this command is that the syntax requires the user to specify a file within which the summary will be written. Care should be taken in choosing the file name as these commands will overwrite an existing file by the specified name. Special caution should be taken to avoid use of the mged database filename as certain horror would face the user unaware of the filename's use in these commands. The three commands *solids, regions* and *idents* give ASCII summary of the information requested for the groups specified.

3.6.7 mirface, extrude and move face in edit menu ARBs present the most plentiful editing options to the mged user. When creating a target description, it is difficult to imagine a more useful solid than the ARB. This discussion will focus on this solid class. The *mirface* command allows a describer to create a configuration on one face of an ARB then mirror that configuration to the opposite face about the appropriate axis. The ARB being currently selected for solid editing is the solid affected. This command is especially useful when adding interior air to a symmetric target. Internal air spaces are commonly defined for analyses where penetration into a certain volume or compartment of the model is of interest. The *mirface* command allows the describer to use symmetry to account for every portion of the space being defined. A face of an ARB is defined and the opposite face will have opposite values on the axis specified. When defining air regions, it is important to remember that any region with an item number is permitted to overlap a region with a non-zero air code.

The only overlaps that are reported for air regions are regions with different non-zero air codes that conflict. The reason for reporting differing air regions as overlap errors stems from the need to keep separate compartments of a vehicle uniquely and concisely defined. This type of consideration is important in the sort of analysis, mentioned above, that keys on changes in air type to recognize the type of damage that could occur versus a specified threat. Similar to *mirface* is the *extrude* command, that keys on the current solid being edited. The *extrude* command allows the describer to project a specified face some normal distance thus creating a new arb. Note the entire face is projected at a normal distance, thus a rotated or skewed face will lose its orientation using this command. If the result of "extruding" a face yields the opposite of expectations, *extrude* the same face using the opposite distance originally specified. Repetitive solids of the same thickness can be created using the *extrude* command. Copying the previously edited solid, then projecting a face to a desired location makes, for instance, armor pl .e creation quite easy. Finally, moving a face selected on the button menu under ARB editing, allows the describer to place the desired face by specifying an appropriate intersecting plane. The current arb selected for editing will maintain its relative face rotation but the manipulated skewed face may seem projected as if in a perspective drawing. An example of face manipulation follows to clarify the text. Another point to remember is that the ARB editing features maintain face planarity, and alter faces containing affected points when completing an editing task. Notably, however, a face can be rotated to cross another or collapse two edges into one and the describer is not warned of the infraction. Such errors are usually detected in simple perusal of the display but follow-on software may not handle a misconstructed ARB in the way the user would expect. Care should be taken to prevent such situations.

Example - ARB editing

Consider an ARB8.
Figure 12 will demonstrate the effect of *mirface, extrude* and moving a face using the edit menu. These commands can be used to achieve identical results but each has its special usage.

3.6.8 Material Properties It is difficult to discuss target description without emphasizing the importance of the region combination. Another use of the item code that tags regions is the assignment of color to a group of associated regions. The *color* command allows the describer to assign an rgb color to a range of item numbers. As briefly mentioned in previous text, having a range of item numbers representing a functional group within the description can be useful in analysis but for creating color-shaded images, other properties of a range of items is of greater significance. The syntax of the *color* command allows the describer to pick an item number range low to high and define a color to

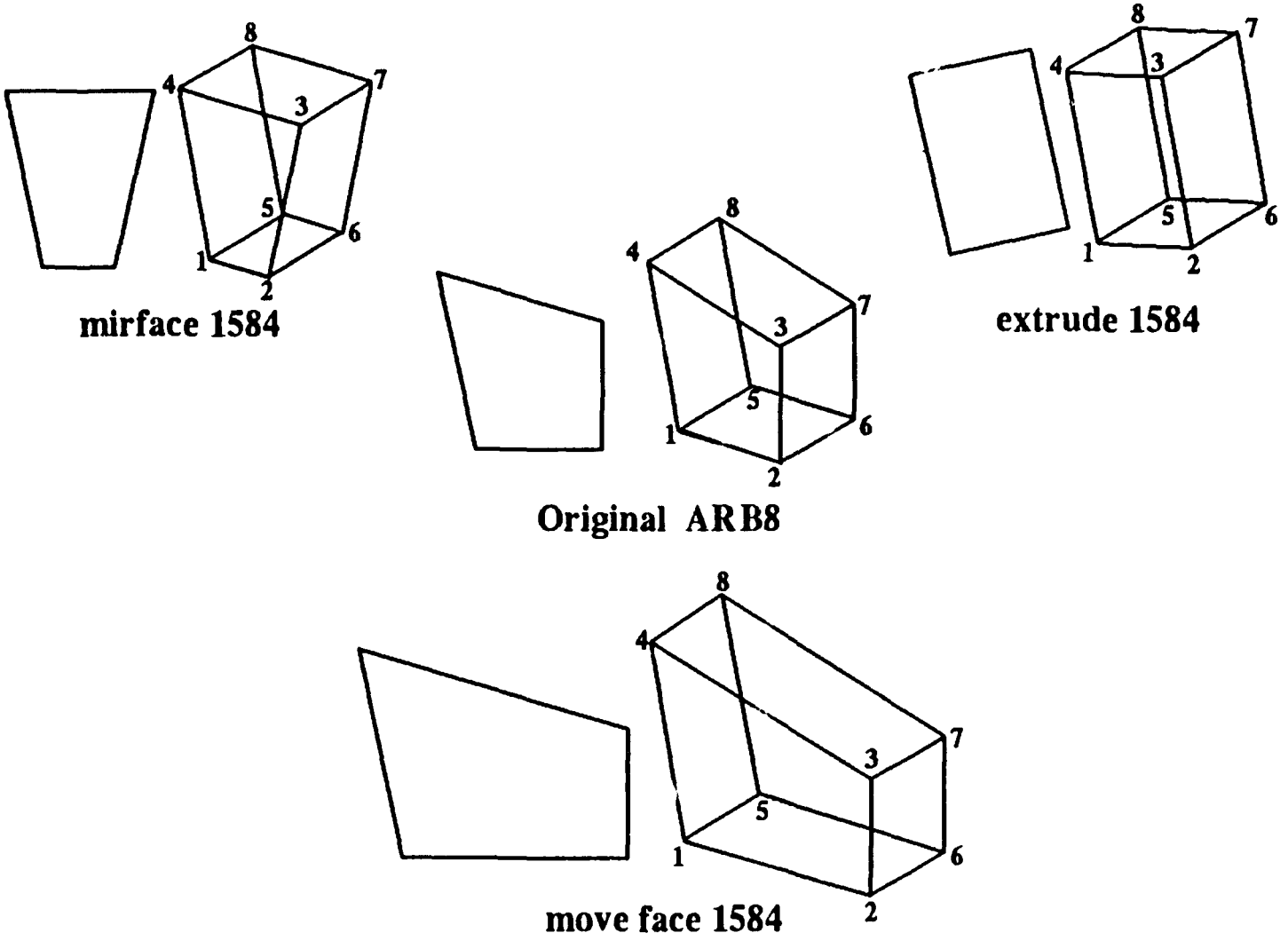


Figure 12. Editing ARB8 Faces

represent it. The color choices for an entire database can be edited by invoking the *edcolor* command. This command puts the user into a text editor and displays the color choices, line by line, that have been defined for the database. If a range of item numbers have not been entered for a specified color, the raytracer associates these items with a grey scale for image production purposes. The *color* command assignments are interpreted by *rt,* the raytracer and MGED. Beyond color, an important function of the material types defined for each itemized region is the acknowledgement of the actual make-up of the real world component. The material type and percentage are, in fact, interpreted only by certain follow-on programs that calculate mass and moment of inertia data for the target. Information validated through the calculation of mass and moment reminds the describer of the importance of physical size and accurate approximation of internal features. Material designation of internal features corresponds to ability of opponent threats to damage or kill a component. More information on validation of target descriptions resides in the next section on pre-analysis treatment of target descriptions, and in the subsequent text on overlap correction using rtcheck.

3.6.9 rtcheck When a significant portion of the target is described and representative volumes are located in juxtaposition, the need to establish, with some certainty, that regions are not overlapping is apparent. The command *rtcheck* raytraces the displayed objects to a desired refinement and generates contrasting rays that are displayed to illustrate overlap problems. Figure 13 provides an example of the output to be expected from *rtcheck* if overlaps are detected. Furthermore, *rtcheck* lists problems encountered, shotline by shotline, in the control window for MGED. Target description validation is a must to ensure accurate representation of the real system being studied. The *rtcheck* refinement is controlled by the -s# option where the # is replaced by some number up to 1024. As the number of rays examined in *rtcheck* increases, the potential accuracy of the validated description increases.

3.6.10 3ptarb and rfarb The two methods of creating ARB8 input to a target description are *3ptarb* and *rfarb.* These methods are not commonly used but might be handy in defining armor plate for a vehicle's shell. *3ptarb* prompts the user for input and requires 3 points and 2 coordinates of the fourth for one face of the ARB8. Furthermore a thickness is required leaving the editor then to calculate the third coordinate of the fourth point and the other face, a normal distance equal to the thickness, away from the original face. The *rfarb* command requires a point on the ARB8 face and the rotation and fallback angle for that face. Two coordinates of the three remaining points of that face are also required as well as a thickness. The third coordinates of the three points are calculated and the opposite face is generated a normal "thickness" away from the original. Again, this command prompts the user for required information.

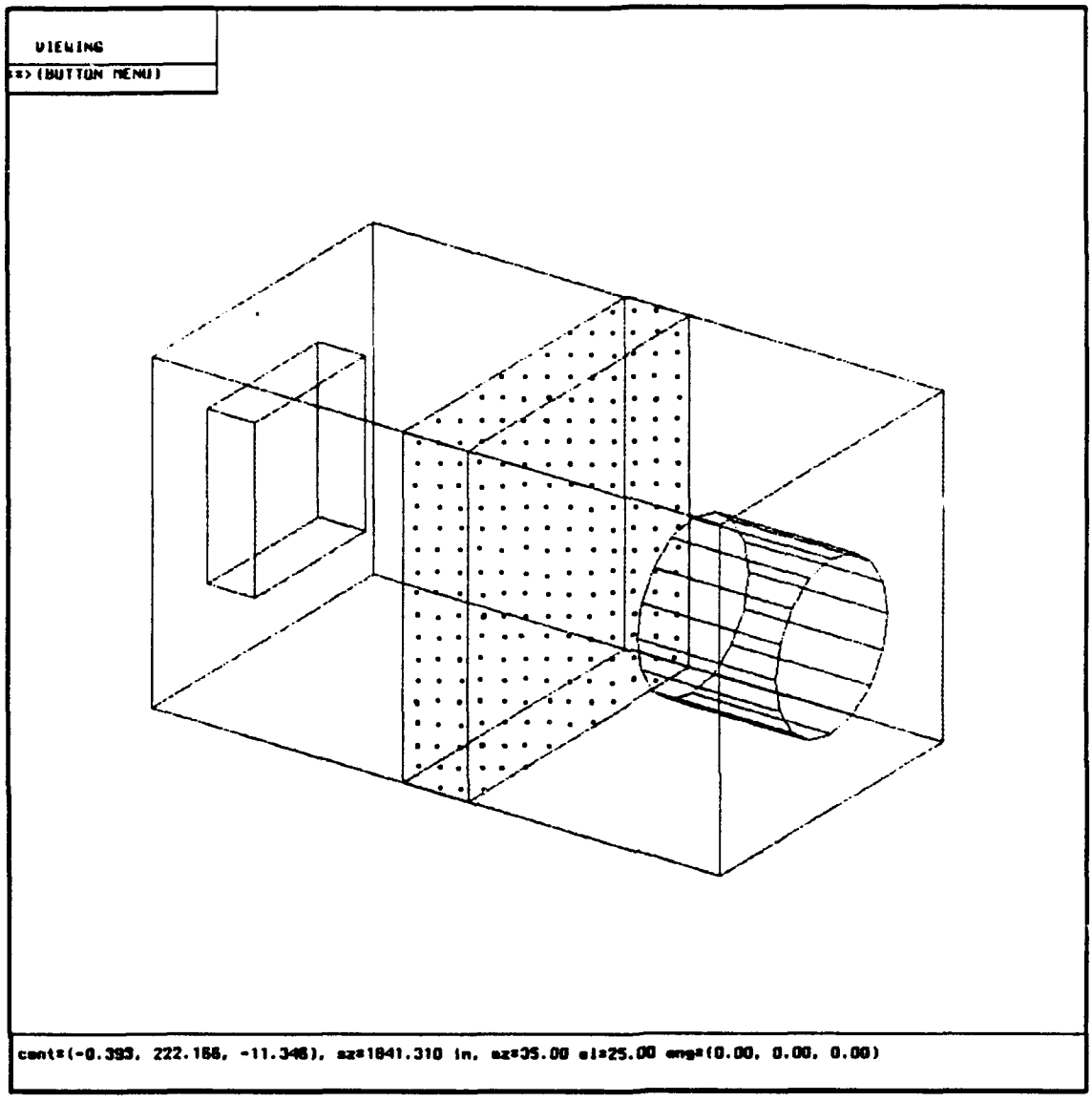


Figure 13. Sample rtcheck Output

3.6.11 listeval and copyeval In validating a target description as the database becomes complex, it is possible that overlaps will be revealed between regions that have only identity matrices in their path and those whose transformation information cannot be pushed. It is unfruitful to attempt to subtract, for instance, a solid within a non-pushed segment of the description and a region unaffected by upper level matrices. In order to circumvent this problem, MGED allows a describer to list a solid as evaluated down a specified path using

the *listeval* command. If the desired solid is one required to ameliorate an overlap problem, the solid can be copied as evaluated down a chosen path using the *copyeval* command. The following example will help clarify the issue.

Example - Using copyeval

Consider a roadwheel assembly that has been instanced (see instancing later in this section) five times down the left side of the vehicle. The prototype has been appropriately translated to create five unique groups displayed on the screen but this is of little help in subtracting the torsion bar of each assembly from the lower part of the vehicle hull. Using *listeval,* each torsion bar can be verified to coincide with the screen image that incorporates the translation of the roadwheel groups. The *copyeval* command prompts the user for the appropriate path information and allows the describer to name the new solid that will solve the overlap problem.

3.6.12 dup and concat Keeping parts of previously described targets into a separate file using the *keep* command signifies the efficiency and ease of the database system in MGED. Adding solids, regions or groups from other targets to a current database may pose problems if object names have been duplicated. The *dup* command allows a describer to investigate the occurrence of duplications in the object names without violating the current database. If duplications are located, the describer can end the current editing session and rename the violating pieces within the "kept" database, or use the *dup* command which allows the user to remain in the current edit session and test an added prefix for all object names. When the user is satisfied that the keep file can be merged with the current database, the *concat* command allows the user to accomplish this using the optional prefix attached to entering objects if it is necessary. Caution is, as always, required to prevent surprises in adding portions of files into a current file. In the event the user overlooks a duplicated name, the editor will not add the second occurrence and will announce its deletion. With *dup* and *concat,* however, this oversight should rarely occur.

3.6.13 inside Commonly in target describing, a component is represented as a casing holding an interior volume that is characterized as a homogeneous mass. On the average, this is sufficiently representative of most situations where subcomponentry is densely packed into the component interior or at very least uniformly. Though this does not remedy all description problems, it handles a great many. To facilitate the creation of interior volumes, the *inside* command allows the user to define the new solid which will represent the interior. The *inside* command prompts the describer for details of desired thicknesses for all

surfaces of the casing solid. If the user is currently in solid edit mode, the currently edited solid is assumed to be the chosen outside solid. In the event the user has not selected solid editing, the command allows specification of the outside solid by name. The following example shows the creation of two inside solids. It must be noted that the thickness parameter can be positive or negative, positive yielding the intuitive result of a remaining thickness of the outside solid. A negative value, howe er, is of great use when "cutting" a hole in one side of the casing for some other description purpose.

Example - inside command

Consider an ARB8 and a TGC that each require definition of an inside solid. Suppose, also, that the TGC inside should protrude beyond the bases. Figure 14 illustrates the desired results. The describer types: inside
outside solid name:
s1
inside solid name:
s1.inside
thickness 1234
0.5
thickness 5678
0.5
thickness 1485
0.5
thickness 2376
0.5
thickness 1265
0.5
thickness 3478
0.5
The resulting inside ARB8 will have created a uniform casing of 0.5 thickness. It should be noted that the thicknesses will be in current description units. For the TGC, the *inside* command will yield the following session. inside
outside solid:
s2
inside solid:
s2.inside
thickness AxB
-0.1

thickness CxD
-0.1
thickness side:
0.5
The result in this example will perforate both bases of the TGC and leave a shell thickness around the conic. It is important to note that the inside solid must yet be combined with the original solid using the boolean operators. Most commonly, the inside solid will be a positive body within its own region as well as a solid subtracted from the outside casing.

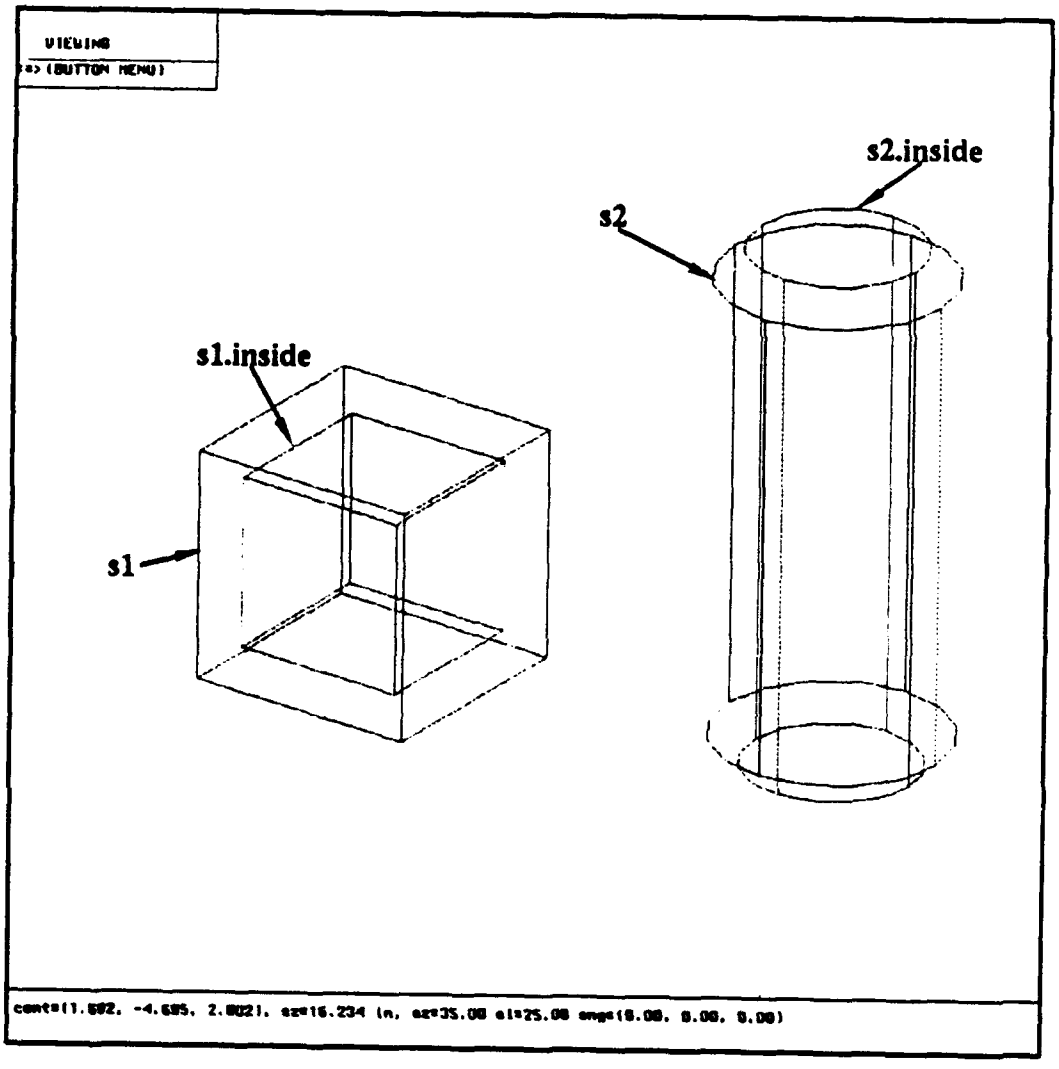


Figure 14. Using the inside Command

3.6.14 Instancing For ease in understanding the essence of instancing, it can be thought of as cloning a prototype group to replicate it a desired number of times. Instancing is one of the most powerful tools in the MGED command set. The use of instancing begins with the creation of a prototype group that will be "instanced" throughout the target. Commonly instanced objects are roadwheel assemblies and rounds of ammunition. Reference of a prototype group first saves storage space as the prototype solid, region and group levels appear only once in the database. The prototype is referenced as desired and manipulated with only the transformation matrix and combination name for each instance being added to the database. Another advantage of instancing is the ability to change the prototype and have all changes automatically reflected in the instances. This useful feature stems from the basic fact that the solid records and regions are only referenced for instanced occurrences. Though instancing creates economy of storage it has a major drawback in its inability to store unique item number identifiers for the region level combinations. In some analyses, this drawback does not except the usefulness of instancing, in others, however, it renders the command completely ineffective. Future releases of BRL-CAD can be expected to remedy this shortcoming of MGED. For the present, if the objects that are appropriate for instancing within the current description are not separately critical to the analysis at hand, use instancing at will. If the analysis effort currently being performed requires unique identifiers for all regions, or at very least, those appropriate for instancing, avoid this useful tool. The following example demonstrates the use of instancing.

Example - Instancing

A prototype roadwheel is presented as proto.rw. Five instances are required of this group on the vehicle left side. Figure 15 shows this prototype and its replications. Note that copyeval may be required to solve some overlaps in the description.

4. A PRE-ANALYSIS LOOK AT TARGET DESCRIPTION

When starting a new target description effort, an analyst must gather available information sources concerning the target. If information is sketchy or detailed, the purpose of the analysis and the time requirement enter into the decision of complexity of the target description effort. Studies that must have quick turnaround require innovation in giving accurate estimates without becoming bogged in details that will have little effect on results. In such cases, perhaps a surrogate target can represent the weapon system of interest or using pieces of existing descriptions and placing them in a newly defined shell might yield adequately sensitive data. In any event, a target describer certainly prefers a technical data package, assembly drawings, parts manuals, the vehicle itself or

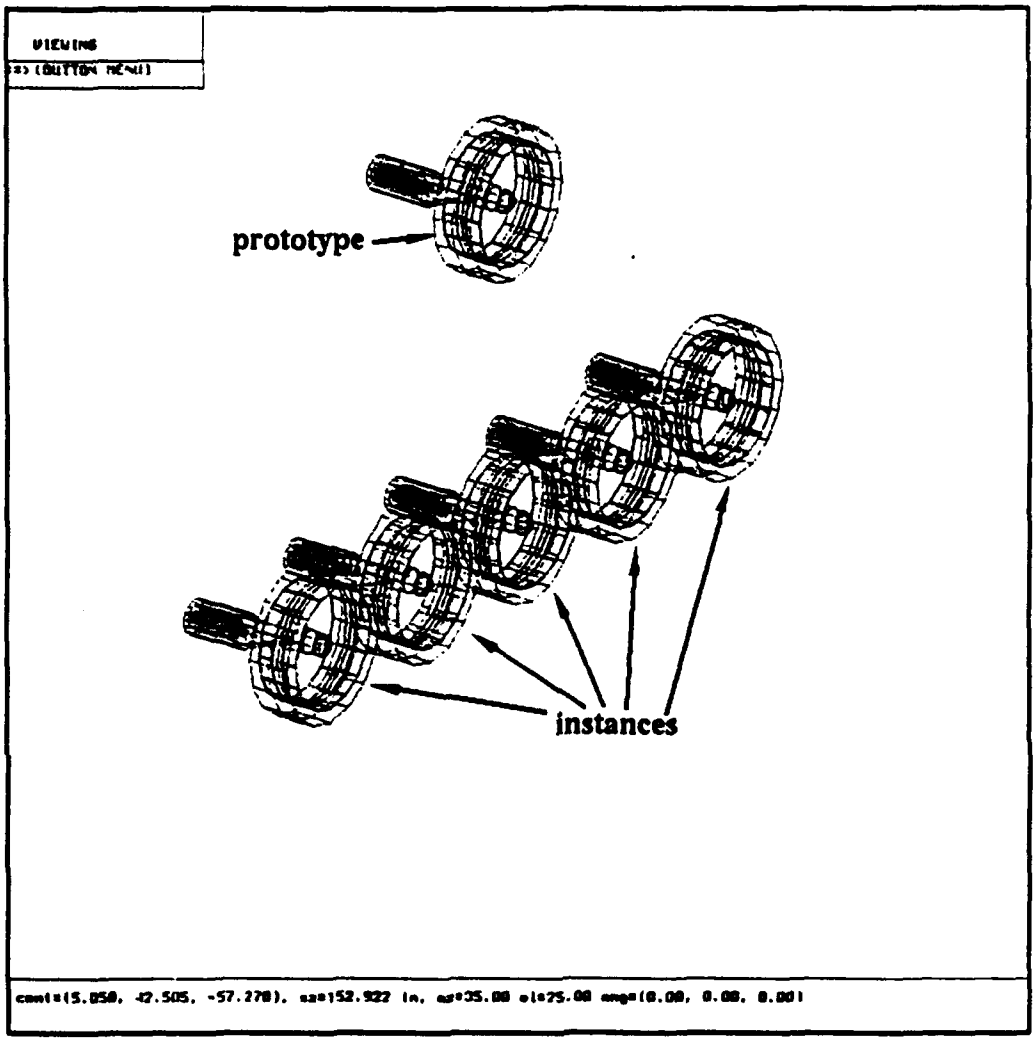


Figure 15. Using Instancing in a Target Description

accurate intelligence data when making a decision on detail. Some targets must be described under less than ideal circumstances so a firm understanding of vulnerability analysis requirements is essential to the production of usable data. A prerequisite to performing any analysis is a thorough understanding of the target description to be used as its basis.

4.1 The Coordinate System In target description convention, a right-handed coordinate system is used. In this arena, a vehicle can still be placed in that three-space in many orientations. For ground vehicles, the positive x-axis points toward the front of the vehicle, the positive y-axis to the describer's left as though the analyst is perched within the target and the positive z-axis points upward. The intersection of the three axes, the origin, can be placed anywhere within those guidelines. Typically, it is prudent in a new description effort to place the origin wisely. Bearing in mind that translation of objects can permit a describer to take advantage of the symmetry of near origin description, it is easy to see that the MGED user can be quite flexible in this regard. Ultimately, however, the origin of the description must be known for follow-on analysis tools that typically require, at least, knowledge of the location of the ground plane. The ground plane is the minimum z-coordinate in the description, or the set of points where the vehicle touches the ground. Bear in mind that this generalization applies to the description of ground vehicles. In common practice, a ground vehicle is basically described approximately symmetric about the x-axis. The x zero can be located at any convenient point, such as the center of the drive sprocket, center of the idler, or the center of the turret ring for armored vehicles. The z zero can also be conveniently located for the describer/analyst at ground, the center of the turret ring or another desirable location for that target. Illustration of the coordinate system and the origin definition is shown in Figure 16.

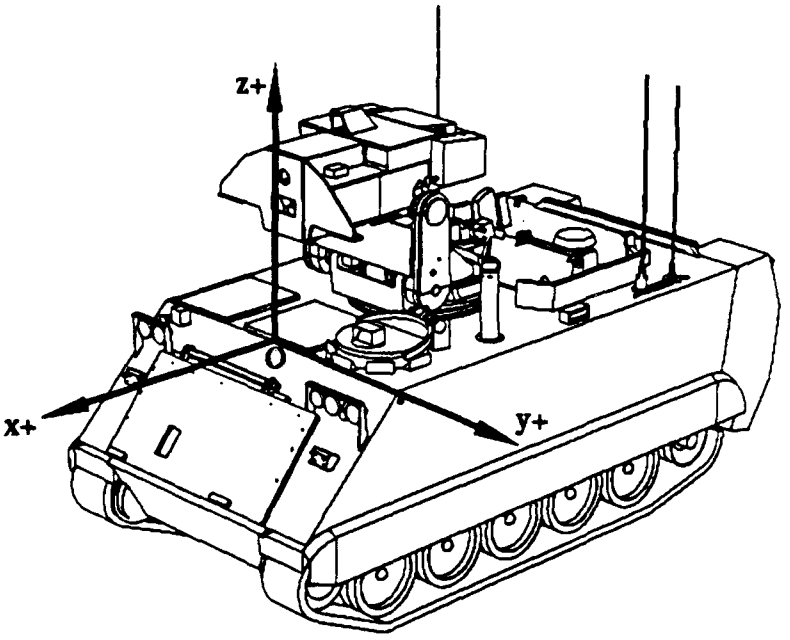


Figure 16. Locating the Origin

4.2 Description Organization As mentioned before, a describer must plan the organization of a new target description effort. Top level groups must be defined to reflect a reasonable hierarchy for that vehicle type. Some indication should exist in the top level groups of a description to indicate the level of detail for the analysis in progress. A suggested approach for a tracked ground vehicle might be to organize top groups of suspension, hull, turret, engine and accessories, crew, hull interior, turret interior, main weapon. The system mission may alter this listing but the intention of such organization is clear. Ranges of item numbers should be easily associated with a certain vehicle function. Figure 17 give an example of a suggested set of item number ranges. An important benefit of adherence to this concept is that it makes further analysis efforts less prone to error. An illustration of this approach is, for instance, defining all engine components in the same identification number range, followed by the engine accessories. This scheme adopts its structure from a parts manual hierarchy and is insurance that parts important to critical function are rarely overlooked. An important note on planning a target description effort is that the describer should use all the MGED tools to speed the description process. Taking advantage of symmetry, instancing, description at origin then translating into place are all useful tools in completing a project in a timely manner.

4.3 Complexity As increasing detail is required in a target description, it is important that the describer respect the real world that is being modeled. It is a challenge to realize that live-fire testing, for instance, examines selected shotlines through a target in pre-test evaluation and looks for near identical results from the target description and the real system. No two vehicles are wired identically, so small, yet critical, ones may be bundled in different places, stowed into place differently, and so on. The describer is tasked, however, with creating the best representation possible. If a component is not in a description, it cannot be evaluated. In the same vein, large, complicated components must be described carefully so that identified regions that vary in material type or percentage are sufficiently indicated to be true to the physical target. As a description becomes complex, the union boolean operator should be used sparingly, as it increases processing time, complicates debugging and validation, and generally makes the description less readable. Complexity cannot be avoided in many analysis tasks but management of the description effort, especially where more than one describer is involved, alleviates most problems.

4.4 Naming Conventions It is almost certainty that a target description will have more than one analytical use in its lifespan. Upper level naming has been discussed but a word about the lower levels should help describers make more useful data files. Solid and region levels within a description rarely require a great deal of consideration in naming. Short names are certainly expedient in the first creation efforts and it is noteworthy that most of the description time will be spent dealing with these levels. At component level, however, a target

1-999	Personnel
1000-1999	System Exterior
2000-2999	Engine & Accessories
3000-3999	Suspension
4000-4999	Main Armament
5000-5999	Powertrain
6000-6999	Electrical System
7000-7999	Hydraulics
8000-8999	Communications
9000-9999	Special Equipment

Figure 17. Sample Item Number Designations

description should be easily readable to the next user of the database. Easy perusal of the database should permit an analyst to grasp the level of detail within a target and determine its usefulness to a current or future analysis. Another point of organization must be mentioned. If the current analysis effort requires the creation of similar vehicle prototypes or a several vehicle system with common chassis or carrier, the describer can create a single database to take advantage of like upper groups and save storage space required for the database. Figure 18 illustrates this point.

4.5 Describing Armor Commonly, in ground target descriptions, an armored shell of the target is described as a first step in simulating the current weapon system. There can be a number of approaches to creating this shell, but one important issue must be addressed. System armor packages may change as a part of development of the vehicle, vehicle mission adjustment or prescribed protection enhancement. Keeping this possibility in mind, armor should be described in plates rather than outside box, inside box, creating varying thicknesses for each face. Many vulnerability analysis tools that use target description information, such as shotlines, as input, permit the analyst to adjust a

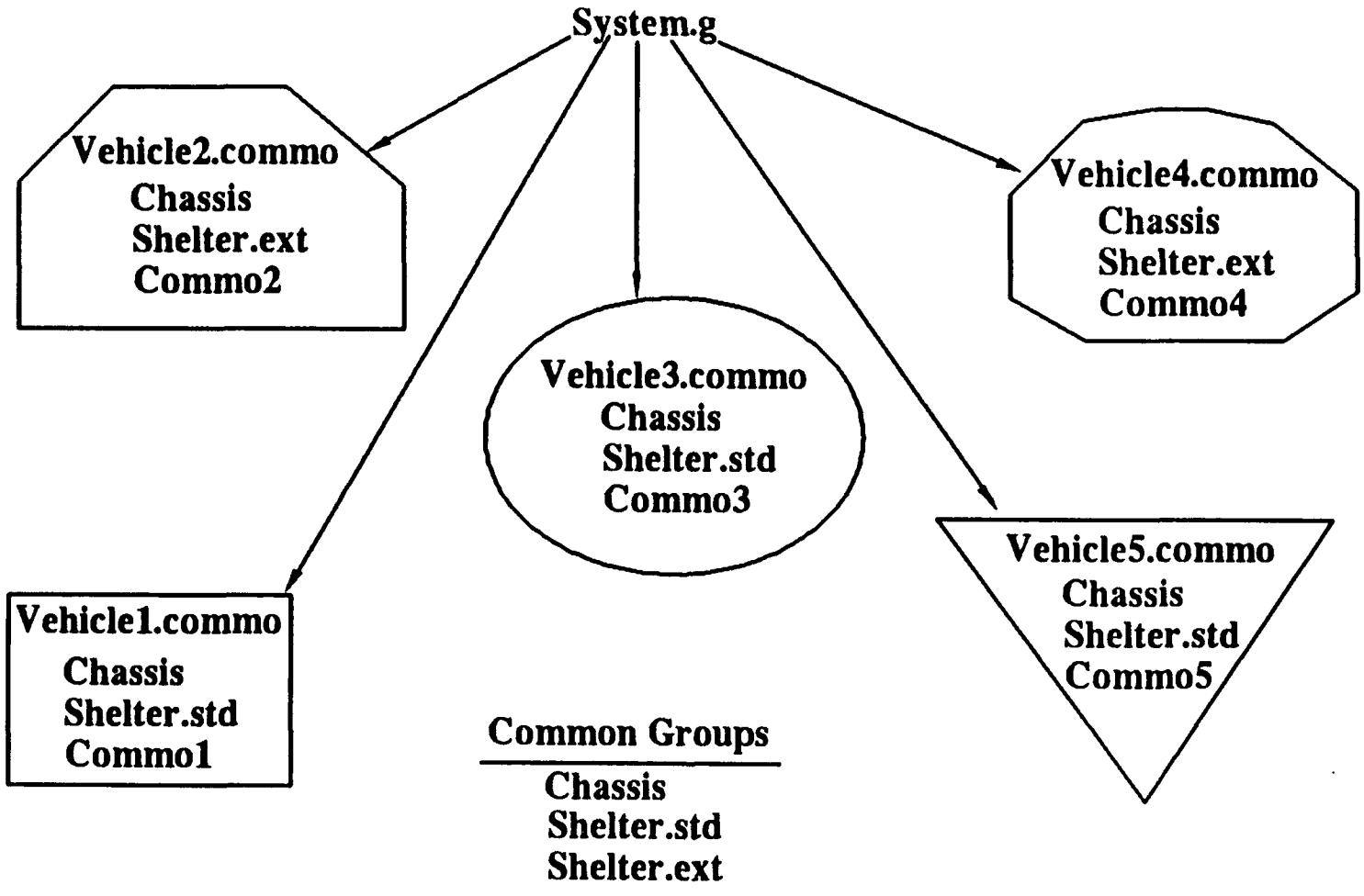


Figure 18. Storage Reduction with Common Groups

line of site thickness by item number encountered. To permit, say, top armor to increase in protection but leave side armors unchanged, requires that each plate reside separately in the database and not as one item chiseled from a single solid. It is important that this guideline not be violated, as an increase in data production time will result if geometry must be changed for each trial iteration.

4.6 Validation Validating a completed target description has been unspecified in VLD. Certain traditions are common knowledge among describers but definition of proper validation will be compiled here for new users. Initial validation involves correction of region overlap errors as reported by rtcheck, or any of the raytracing package. Correction of overlap errors may be accomplished by using booleans to combine existing solids with offending regions, typically by subtraction, moving an offending region or a part of it, shrinking or otherwise editing the solid members of offending regions, or deleting the offensive region. Please note that it is categorically inappropriate to simply subtract a problem away. Strange holes in components can unexpectedly render them almost volumeless and not yielding accurate target representation. Once detected overlap errors are corrected, using the raytracing library to create color shaded images of the target or hidden line drawings help visually confirm target accuracy. Next, validation of the component masses and vehicle moments of inertia help establish the accuracy of the material and percentage designation of the vehicle's components. For the purposes of image creation, boolean operations can be performed within groups to allow cut-away views to be produced from target descriptions. Used only for such special purposes, combining a large ARB with an entire target to make, for instance, a mid-line cut, would allow the describer to examine internal component placement in a shaded image format. Examining raytracing results for first component identification on each shotline, is another useful validation tool. Applications that peruse first item thicknesses and line-of-sight placements can help distinguish problems of missing or thinned armor due to a geometry error. Finally, extensively shotlining the target and reviewing follow-on analysis results stands out as the true test of description accuracy - production of credible analysis results.

5. THE ANALYST'S TOOLS

Approaching a target description as a necessary piece of an analysis helps focus the describer, and prevents simplifying solely the description effort. This focus prevents infliction of awkward details on a later analysis chore, with the sole benefit of creating a convenient description structure. Granting that the task at hand may require months of description or a week of adapting a surrogate, target description is a critical part of the vehicle vulnerability tools.

5.1 Using Other's Descriptions Initial examination of an inherited target description is commonly a confusing or overwhelming task. Truly understanding another's choices in creating a target takes time and insight. Adherence to many of the precepts suggested in this document will help make this eventuality more reasonable. Within VLD, description efforts have been strictly describer dependent with little concern for format or standardization. Recently created target descriptions are less likely to be troublesome but older ones may be poorly organized and on first glance, randomly itemized. The one-to-one sequential nature of MGED's predecessor code, GIFT, Geometric Information For Targets, leaves a library of older descriptions that are troublesome, at best, to convert and utilize in MGED. The code, comgeom-g, takes an ASCII listing of solids, regions and region identification information and loosely defines an MGED database with a gross scale hierarchy. The upper groups are defined g1, g2, g3... depending on the item number range membership of the region being converted. All information on the idents table is lost with the exception of the integer air, item, material and percentage data. Placeholder regions, fostering a one-to-one solid to region scheme, are frequently defined in such old databases and, as such, are converted as regions with no member solids. Part of the conversion effort is the renaming of all components and the creation of a useful hierarchy. An example of comgeom-g usage follows:

Example - comgeom-g usage

There are two possible formats for GIFT files, GIFT4 and GIFT5. The user must identify which style is being converted by examining the region identification table, the last table, in the description. GIFT4 format has material and percentage information at the end of each identification line where GIFT5 has material and percentage information immediately following the item number and air code at the beginning of each line. A version flag, -v, permits the user to specify GIFT4 format since GIFT5 is the default. Solids from the Gift listing are converted to solid records in order, as they appear in the solids table. Solids are named s*i* where *i* is the sequential number of the solid in the GIFT listing. Like solids, regions are named sequentially, r*i*. Groups are formed by combining regions within certain item number ranges to create g00 - regions with item number 0, g0 - regions with item numbers 1-99, g1 - regions with item numbers 100-199, and so on. An optional suffix flag, -s, permits the user to append a suffix to each object included in the MGED database. Possible invocations of this conversion routine are:

comgeom-g m99.cg5 m99.g

comgeom-g -s.m99 m99.cg5 m99.g
comgeom-g -v4 m99.cg4 m99.g
comgeom-g -s.m99 -v4 m99.cg4 m99.g

Though plausible to convert, it is sometimes more time effective to begin a description from scratch in MGED. Taking pieces of existing MGED databases is usually less complicated but still rely heavily on the wisdom of the original describer. Commands such as *whichid* and *find* help locate pieces that must be deleted or included in a group of interest. The keep command, as previously discussed, is very useful in this case. Another type of target description scheme, currently used by the Air Force and the Navy, is the PATCH format. This type of target description relies on surface modeling to enclose a volume in a triangulated web, thus defining it. Conversion tools for PATCH to MGED exist, but again create many problems in the effort. Use of PATCH descriptions is cumbersome at best.

5.2 Input to Vulnerability Codes Shotline analysis of a target allows the identification of vulnerable locations on the target surface and within its exterior shell. Through use of penetration equations for various conventional threats versus many material types, target descriptions visualize important target vulnerability concepts and thus are supremely important. Detail and accuracy of information within a target description drives the analytical process, as has been discussed. Analysis of a target ultimately depends on the accuracy of its geometry. Care must be taken to include all critical components that are sizeable enough to be detected in a typical shotline analysis. A component with presented area that is less that one quarter of the interrogation cell size is usually safely omitted but this decision is dependent on the type of analysis being performed and the number of cells through which a component may travel, like wires or fluid lines. Redundancy of function within a target system must also be considered and adequate representation of shielding components must be included.

5.3 Surrogacy Sometimes, so little information is available about a target that it is impossible to adequately represent it. In such cases, where some representative data must be delivered, a presumed similar target can be chosen to substitute for the one of interest. This problem frequently arises in dealings with foreign targets. Selection of a surrogate target is a task that requires expert decision making and cannot be handled without current intelligence system approval. Commonly, in choosing a surrogate, similar armor and armament, as well as, related profile and function are considered.

5.4 Specialized Targets Target description efforts requiring special consideration are those of aircraft and those for signature prediction. Aircraft targets are highly detailed models with refined cell size considerations when

analyses are performed. Air systems critically rely on fuel and hydraulics, as well as small diameter control rods where armored ground vehicles are not commonly driven by such refined details. Considerations for signature studies focus on identifying glints returned by bends and creases in every detail of external geometry. Line-of-sight thicknesses of protecting armor are of standard interest in armored vehicle studies but hold little concern for those attempting to identify opponent vehicles by radar, heat or other images gained in stealth. MGED can readily be used to create a diversity of targets but special considerations for the description's application must drive the user's attention to the details of the geometry.

5.5 Using Other Platforms An MGED database is stored in a binary format on the platform hardware as the description is built. On occasion, another user or location, requires a copy of the target description. In order to move MGED files across hardware types, two BRL-CAD utilities were created. The first program, *g2asc,* converts an mged database to an ASCII file. The other utility, *asc2g,* restores the file to a binary format on the new platform. To illustrate, the following scheme is presented:

Example - file conversions

Consider platform A and the file *target.g*
Execute the following -
g2asc < target.g > target.asc
which converts the file target.g into an ASCII impression that can be transported across machine types.
Consider platform B and the file *target.asc*
Execute the following -
asc2g < target.asc > target.g
which creates the file target.g, which is a usable MGED file on platform B.

6. CONCLUSIONS

MGED is a useful tool for creating target descriptions for vulnerability analyses. Many features can be used to manipulate these descriptions, create images from them, and answer many questions about the real systems the descriptions represent. This document provides basic guidelines for ground vehicle descriptions as they are used in the VLD. The precepts of solid modeling, compiled here, are applicable, however, to any description tasks that require detailed geometry and specialized post-processing.

INTENTIONALLY LEFT BLANK.

No. of Copies	Organization
2	Administrator Defense Technical Info Center ATTN: DTIC-DDA Cameron Station Alexandria, VA 22304-6145
1	Commander U.S. Army Materiel Command ATTN: AMCAM 5001 Eisenhower Ave. Alexandria, VA 22333-0001
1	Commander U.S. Army Laboratory Command ATTN: AMSLC-DL 2800 Powder Mill Rd. Adelphi, MD 20783-1145
2	Commander U.S. Army Armament Research, Development, and Engineering Center ATTN: SMCAR-IMI-I Picatinny Arsenal, NJ 07806-5000
2	Commander U.S. Army Armament Research, Development, and Engineering Center ATTN: SMCAR-TDC Picatinny Arsenal, NJ 07806-5000
1	Director Benet Weapons Laboratory U.S. Army Armament Research, Development, and Engineering Center ATTN: SMCAR-CCB-TL Watervliet, NY 12189-4050
(Unclass. only) 1	Commander U.S. Army Rock Island Arsenal ATTN: SMCRI-TL/Technical Library Rock Island, IL 61299-5000
1	Director U.S. Army Aviation Research and Technology Activity ATTN: SAVRT-R (Library) M/S 219-3 Ames Research Center Moffett Field, CA 94035-1000
1	Commander U.S. Army Missile Command ATTN: AMSMI-RD-CS-R (DOC) Redstone Arsenal, AL 35898-5010

No. of Copies	Organization
1	Commander U.S. Army Tank-Automotive Command ATTN: ASQNC-TAC-DIT (Technical Information Center) Warren, MI 48397-5000
1	Director U.S. Army TRADOC Analysis Command ATTN: ATRC-WSR White Sands Missile Range, NM 88002-5502
1	Commandant U.S. Army Field Artillery School ATTN: ATSF-CSI Ft. Sill, OK 73503-5000
(Class. only) 1	Commandant U.S. Army Infantry School ATTN: ATSH-CD (Security Mgr.) Fort Benning, GA 31905-5660
(Unclass. only) 1	Commandant U.S. Army Infantry School ATTN: ATSH-CD-CSO-OR Fort Benning, GA 31905-5660
1	WL/MNOI Eglin AFB, FL 32542-5000
	Aberdeen Proving Ground
2	Dir, USAMSAA ATTN: AMXSY-D AMXSY-MP, H. Cohen
1	Cdr, USATECOM ATTN: AMSTE-TC
3	Cdr, CRDEC, AMCCOM ATTN: SMCCR-RSP-A SMCCR-MU SMCCR-MSI
1	Dir, VLAMO ATTN: AMSLC-VL-D
10	Dir, USABRL ATTN: SLCBR-DD-T

INTENTIONALLY LEFT BLANK.

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. BRL Report Number BRL-MR-4001 Date of Report September 1992

2. Date Report Received _____

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) _____

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) _____

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. _____

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) _____

CURRENT ADDRESS

Name

Organization

Address

City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the New or Correct Address in Block 6 above and the Old or Incorrect address below.

OLD ADDRESS

Name

Organization

Address

City, State, Zip Code

(Remove this sheet, fold as indicated, staple or tape closed, and mail.)

DEPARTMENT OF THE ARMY
Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

OFFICIAL BUSINESS

BUSINESS REPLY MAIL
FIRST CLASS PERMIT No 0001, APG, MD

Postage will be paid by addressee.

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES