Rod Fleming
9/5/78

Theseus

Reference Manual

Ingrid Carlbom

November, 1977

TABLE OF CONTENTS

PICK
~~BUTTON~~ – poll/wait
~~STACK~~ – ~~stack~~
KEYBOARD – poll/wait
LOCATOR – call func poll/wait

1 INTRODUCTION

Theseus is a high-level line-drawing graphics system implemented as a set of procedures callable from Algol W. Theseus generates code for the META4B/SIMALE display processors of the Brown University Graphics System (BUGS). The DPU code is processed for display on a Vector General Display (VG).

Theseus is a structured display file (SDF) system. It allows a user to build and manipulate a hierarchical graphical data structure. This data structure is analogous to a program structure consisting of procedures which may call other procedures. A graphical data structure consists of objects which may reference, or call, other objects.

An object consists of one or more segments. A segment is the smallest unit of data structure modification: it may be added to, or deleted from, an object. A segment may contain output primitives, such as lines and text, object calls, and attributes and transformations that apply to these object calls. The modelling transformations, or object construction transformations, specified as part of object calls allow objects to be defined in local coordinate systems and to be combined into new objects.

An object is displayed by creating a view of the object. This view has associated attributes and viewing transformations that determine how the object or a portion of the object is mapped to the display surface. When the graphical data structure is processed by the DPU, the viewing and modelling transformations are composed and applied to the local coordinate data of the objects. Similarly, the DPU composes the view, the object call and the segment attributes before applying them to the output primitives.

The elements of the graphical data structure may be manipulated interactively. Theseus provides support for all the interaction devices of BUGS.

Theseus is designed to contain a functionally complete set of features that can easily be extended. It will evolve as new experience is gained with graphics programming. Some facilities may be changed or removed if they do not prove to be useful, and others will be added. It is expected that Theseus will be enhanced in three major areas: interaction handling, real-time dynamics, and user-defined primitives.

Theseus consists of a lean set of functions. Only those features that practical experience deemed useful have been included. It is hoped that Theseus, even at this stage, will provide a flexible and useful set of functions.

In the description of Theseus that follows the routines are grouped by topic: first the routines that build and modify the hierarchy are defined, next the transformations, followed by the primitives, the attributes, and the interaction handling. The last section describes some miscelaneous features regarding the operation of Theseus.

The description of each routine is preceded by the syntax of the Algol W declaration for the routine. Upper case symbols denote Algol W and Theseus keywords; lower case symbols denote parameters to be specified by the user. Brackets, [ ], indicate optional paramenters. Optional procedure parameters can be omitted at a call, but the preceding comma must be specified. Optional integer parameters must be specified as non-positive integers, to indicate that they are to be ignored by Theseus.

Transformations, attributes, and extents are specified as procedure parameters of routines that define objects, segments, views, and object calls. An actual parameter corresponding to a procedure parameter can be any Algol W statement: a simple statement, a begin block, or a procedure statement. However, the only Theseus calls that can be used as part of any type of procedure parameters are Theseus routines of that type, e.g., a modelling transformation parameter may only contain Theseus modelling transformation routines, and attribute parameter only Theseus attribute routines.

2 HIERARCHY

The first three subsections below discuss those routines that are used to build and modify tha components of the hierarchical data structure: objects and object calls, view, and segments. Each of these constructs can have associated names for identification and modification purposes. Naming is discussed in the following subsection. Extents and conditional execution of objects are discussed at the end of this section.

2.1 OBJECTS

An object is defined by a hierarchical data structure. Each node in this data structure consists of an object, i.e., an object is defined in terms of other objects. When the DPU processes the data structure an object is referenced via an internal directory. At an object call the environment of the current object is saved, the transformation matrix of the call is composed with the current transformation matrix, and the attributes of the called object are composed with the current attributes. Upon the return from an object the environment of the calling object is restored. Instances of an object are hence easily added to and deleted from the data structure, but induce significant overhead in DPU processing time.

Theseus allows the user to create and delete objects, to modify objects by adding and deleting segments, and to create object calls. The attributes and the modelling transformations of the calls may be altered dynamically. An object may have an associated extent and can be conditionally executed depending on the size of this extent. Objects may be saved in picture files, and retrieved from these files for later use. Object creation, deletion and calling are discussed in this section; the discussion of picture files is deferred to section 7.

```
OPOBJ2D(INTEGER VALUE name; [PROCEDURE extent]; [PROCEDURE size])
OPOBJ3D(INTEGER VALUE name; [PROCEDURE extent]; [PROCEDURE size])
```

make the object "name" the open object for addition and deletion of segments. An object name must be unique within the data structure. If the specified object name exists, the existing object is modified; otherwise a new object is created. The object remains the open object until a call to CLOSOBJ.

An extent, i.e. a surrounding window of an object, can optionally be specified as part of the object definition. An existing extent can be modified when an object is altered. The extent is used to aid the clipping process, and is further discussed in section 2.5.

By specifying a minimum and maximum extent relative to the size of the clip window/volume, the object will be executed conditional upon its extent size. The procedure "size" defines the minimum and maximum extents (see section 2.5), and can be altered as the object is altered.

Error Conditions:
1) No more core available for graphical data structure.
2) Dimensionality mismatch.
3) An object is already open.
4) Invalid name; must be a positive integer.
5) Invalid extent routine.
6) Invalid size routine.

CLOSOBJ

The open object is closed, and can not be modified until it is again made the open object by a call to OPOBJ2D or OPOBJ3D.

Error Conditions:
1) No open object.
2) A segment was open; it has been closed.

DCLOBJ2D(INTEGER VALUE name)
DCLOBJ3D(INTEGER VALUE name)

declare a 2D and a 3D object, respectively, i.e. enter an object name in the directory. These procedures enable the user to generate an object call before the object is created, thereby allowing him to create a picture "top-down" instead of "bottom-up". The actual object may subsequently be defined by calling OPOBJ2D or OPOBJ3D.

Although these procedures may be invoked while there is an open object, this should generally be avoided since it causes inefficiency in DPU processing.

Error Conditions:
1) Object name already exists.
2) No more core available for graphical data structure.
3) Invalid name; must be a positive integer.
4) Dimensionality mismatch.

NULLOBJ(INTEGER VALUE name)

deletes the body of the named object, but leaves the object name in the directory. A new object with this name may subsequently be defined by OPOBJ2D or OPOBJ3D.

Error Conditions:
1) Object does not exist.
2) Invalid name; must be a positive integer.

DLTOBJ(INTEGER VALUE name)

deletes the body of the object and removes its name from the directory. If there are any remaining calls to this object in the data structure they will be ignored until the directory entry is used again. Since the programmer cannot directly control allocation of directory entries, this can cause unpredictable results.

Error Conditions:
1) Object does not exist.
2) Invalid name; must be a positive integer.

RNMOBJ(INTEGER VALUE oldname, newname)

renames an object "oldname" to "newname". All existing calls to object "oldname" will refer to "newname", and the object must subsequently be referenced by "newname".

Error Conditions:
1) Object name already exists.
2) Object does not exist.
3) Invalid name; must be a positive integer.

SWAPOBJ(INTEGER VALUE name1, name2)

exchanges the bodies of the objects "name1" and "name2". This routine can be used for double buffering. The object can be built as object "name2" while displayed as object "name1". When object "name2" is completed, the two objects are swapped and a new version of the object is again built as object "name2". No renaming is required!

Error Conditions:
1) Object does not exist.
2) Dimensionality mismatch.
3) Invalid name; must be a positive integer.

CALLOBJ(INTEGER VALUE name, [elemname], [dynamo]; [PROCEDURE attr]; [PROCEDURE mtrans])

adds a call of the object "name" to the open segment in the display data structure. The element name may be specified for identification of, and a dynamo name for modification of the object call. The attributes and transformations (see sections 5 and 3) of the object call are specified by "attr" and "mtrans", respectively.

If no attributes are specified the default attributes will be applied. If the transformations are omitted, a call without a transformation is generated.

In order to delete a call to an object, the surrounding segment must be deleted. Any component of the call may be altered dynamically by CHGCALL, which is described below.

An object may call itself recursively.

Error Conditions:
1) No more core available for graphical data structure.
2) Object does not exist.
3) Dynamo name already exists.
4) Dimensionality mismatch.
5) No open segment.
6) Invalid attribute routine.
7) Invalid modelling transformation routine.
8) Invalid name; must be a positive integer.
9) A menu object can not be referenced by an object call.

CHGCALL(INTEGER VALUE dynamo, [name]; [PROCEDURE attr]; [PROCEDURE mtrans])

replaces the called object, and modifies the attributes and transformations of an existing object call pointed to by the "dynamo" (see section 2.4). "name", "attr" and "mtrans" are defined as for CALLOBJ. If an optional parameter of the call is a non-positive integer or is omitted, the corresponding component of the object call is left unchanged. Note that if the object call was generated without a modelling transformation, such a transformation may not be added.

Error Conditions:
1) Dynamo does not exist.
2) Object does not exist.
3) Dimensionality mismatch.
4) Object call does not have an associated modelling transformation.
5) Dynamo does not refer to a correct type.
6) Invalid attribute routine.
7) Invalid modelling transformation routine.
8) A menu object can not be referenced by an object call.

2.2 VIEWS

An object is displayed by creating a view of the object. Each view has a unique name, and has associated attributes and viewing transformations. The views may be added to and deleted from the display data structure, and the image of the viewed object may be altered by changing the associated attributes and viewing transformations.

ADDVIEW(INTEGER VALUE objname, viewname, [dynamo]; [PROCEDURE attr]; [PROCEDURE vtrans])

adds a view with the name "viewname" of the specified object to the display data structure. A dynamo may be specified for modification of the view. The attributes and the viewing transformations are specified by "attr" and "vtrans", respectively, and are discussed in sections 3 and 5 below.

If no attributes or viewing parameters are specified the defaults will be applied.

Error Conditions:
1) Object does not exist.
2) Invalid name; must be a positive integer.
3) View name already exists.
4) Dynamo name already exists.
5) Invalid attribute routine.
6) Invalid viewing transformation routine. (The default transformation has been used.)
7) Inconsistent viewing parameters specified. (The default has been used.)

CHGVIEW(INTEGER VALUE dynamo, [objname]; [PROCEDURE attr]; [PROCEDURE vtrans])

replaces the image of one object with another by changing the name of the viewed object, and modifies the attributes and the viewing transformations of the view pointed at by "dynamo". "objname", "attr", and "vtrans" are defined as for ADDVIEW. If an optional parameter of the call is a non-positive integer or is omitted, the corresponding component of the view is left unchanged.

Error Conditions:
1) Dynamo does not exist.
2) Dynamo does not refer to a correct type.
3) Object does not exist.
4) Dimensionality mismatch.
5) Invalid attribute routine.
6) Invalid viewing transformation routine. (The default transformation has been used.)
7) Inconsistent viewing parameters specified. (The default has been used.)

DLTVIEW(INTEGER VALUE viewname)

deletes the specified view from the data structure.

Error Conditions:
1) View name does not exist.

2.3 SEGMENTS

An object consists of one or more segments that are made up of primitives and object calls. A segment may be added to and deleted from the open object, have associated attributes, and can be conditionally executed depending on the extent size of the object. A segment can not be reopened once it has been closed, but its attributes, its primitives and object calls may be modified.

Segments are cheaper in DPU processing time than are objects, but do not offer the same flexibility: they may not be referenced by other objects, and may not have associated extents.

BEGSEG(INTEGER VALUE name, [dynamo]; [PROCEDURE attr]; [PROCEDURE size])

makes the segment "name" the open segment for adding primitives and object calls. The segment name must be unique within the object, but need not be unique within the data structure. The segment remains the open segment until a call to ENDSEG.

If the attributes are omitted, a segment without attributes will be generated, object attributes alone will be applied to the primitives in the segment. If one or more attributes are specified, default attributes will be generated for the remaining attributes. Segments without attributes are more efficient in DPU processing time.

A dynamo name may be specified for modification of the segment attributes (see CHGSEG below), but should not be specified if the attributes are omitted. The attributes for the primitives are defined by "attr", and are futher discussed in section 5. "size", as for objects, defines the conditions under which the segment is displayed.

Error Conditions:
1) No more core available for graphical data structure.
2) No open object.
3) Segment name already exists in this object.
4) Invalid name; must be a positive integer.
5) Dynamo name already exists.

6) A segment is already open.
7) Invalid attribute routine.
8) Invalid size routine.
9) Attempt to associate a dynamo with a segment that does not have associated attributes.

ENDSEG

The open segment is closed, i.e., no more primitives and calls may be added to the segment.

Error Conditions:
1) No open segment.

DLTSEG(INTEGER VALUE name)

deletes the specified segment from the open object.

Error Conditions:
1) No open object.
2) Segment does not exist in the open object.

CHGSEG(INTEGER VALUE dynamo; PROCEDURE attr)

changes the attributes of the segment with the specified dynamo name. "attr" is defined as for BEGSEG.

Error Conditions:
1) Dynamo does not exist.
2) Dynamo does not refer to a correct type.

2.4 NAMING

Names are used to identify or modify parts of the graphical data structure. Names may be associated with objects, segments,

views, object calls, and primitives. A name may be any positive integer.

An object name is used for the purpose of identification and modification of the object. It is defined when the object is defined or declared (see section 2.1). An object name must be unique within the display data structure.

A view name is used for identification and deletion of a view of an object. A view name is defined when a view is added to the display data structure, and must be unique within this data structure.

A segment name is used for the purpose of identification and deletion of a segment within an object. It is defined at segment definition time. A segment name must be unique within an object, but need not be unique within the entire graphical data structure. A segment is, however, always uniquely defined by the name pair: (object, segment).

Theseus provides one level of naming within segments: individual primitives and calls may be named for identification purposes (see section 6). Such element names are specified as part of procedures that generate calls (see section 2.1) and graphical primitives (see section 4). An element name need not be unique within a segment.

Dynamo names (or dynamos) are unique pointers to items such as segments, primitives, views, and object calls in the graphical data structure, and are used to modify these items dynamically. A dynamo is a shorthand for the pair (object, segment) or for the triplet (object, segment, element). A dynamo is either created when segments, object calls, views, and primitives are created, or when such items have been identified by a pick. (See sections 2, 4, and 6). Dynamos should be deleted by the user when the referenced item no longer is going to be modified.

DLTDYN(INTEGER VALUE dynamo)

deletes the specified dynamo.

Error Conditions:
1) Dynamo does not exist.

2.5 EXTENTS

An extent of an object is a rectangular window which always surrounds the object and whose edges are parallel to the axes of the local coordinate system. An extent is optionally specified as part of an object definition. When the object call is executed by the DPU, the extent of the called object is transformed by the current transformation matrix. A new extent which surrounds the transformed extent, but whose edges are parallel to the transformed coordinate axes, is calculated. The extent aids in the clipping process in that an object whose transformed extent lies entirely inside or outside the clip window can be trivially accepted or rejected, respectively.

An object can be conditionally executed depending on the size of its transformed extent. A segment can be conditionally executed depending on the size of the transformed extent of the surrounding object. This makes it possible to define "levels of detail" of an object each level in one segment.

EXTENT2D(REAL VALUE xcenter, ycenter, xsize, ysize)
EXTENT3D(REAL VALUE xcenter, ycenter, zcenter, xsize, ysize, zsize)

specify the center and size of the extent window in local coordinates. The dimensions of the extent window are xsize*2 by ysize*2 (by zsize*2) and should be defined such that they surround the entire object. This routine is optionally specified at object creation time. An existing extent of an object may be modified, but an extent can not be added to an object once the object is created.

Error Conditions:
1) Object has no extent.
2) This procedure can only be used as part of OPOBJ2D and OPOBJ3D.
3) Dimensionality mismatch.
4) Numeric data out of range.
5) No more core available for graphical data structure.

SIZE(REAL VALUE min, max)

is optionally specified at object and segment creation. The object or segment is displayed only if the size of the transformed extent of the object relative to the size of the clip window/volume is in the range "min" to "max". The object or segment is displayed if

$$2*min <= xsize^2/windxsize^2 + ysize^2/windysize^2 <= 2*max$$

in two dimensions and for orthographic projections in three dimensions, and if

$$2*min <= xsize^2/zsize^2 + ysize^2/zsize^2 <= 2*max$$

for perspective projections in three dimensions. xsize, ysize, and zsize are the dimensions of the transformed extent; windxsize, windysize are the dimensions of the clip window. Note that "min" and "max" are always in the range 0.0 to 1.0.

Error Conditions:

1) Numeric data out of range.
2) Min must be smaller than max.
3) This procedure can only be used as part of OPOBJ2D, OPOBJ3D, and BEGSEG.
4) Object has no extent. (An attempt was made to add a size to the object or one of its segments.)
5) An existing object without an associated SIZE was reopened with a non-null size routine.
6) No more core available for graphical data structure.

3 TRANSFORMATIONS

Theseus provides both modelling and viewing transformations. Modelling transformations allow objects to be defined in local coordinate systems and to be combined into new objects. These transformations include rotation, scaling, and translation. In addition, the user may specify an arbitrary 4x3 or 3x2 transformation matrix to obtain any nonstandard modelling transformation.

The viewing transformations determine how objects are mapped onto the display surface. In 2D, a window determines which portion of the world coordinate space is to be viewed, and a viewport controls the placement of the image of the object on the display surface. In 3D, a view volume determines which part of the world coordinate space is to be viewed, a method of projection defines how the object is projected onto a plane, and a viewport controls the placement of the image of the projected object on the display surface.

Theseus uses a left-handed coordinate system. The world coordinate system has the origin at (0.0,0.0,0.0) and a range specified by the RANGE function.

3.1 RANGE OF WORLD COORDINATES

The world coordinate data and the transformations are specified as real fullwords. The DPU, however, only accepts coordinate data in the range -1.0 to 1.0. The mapping from world coordinate space to the DPU coordinate space must be specified by the user.

RANGE(REAL VALUE xyzscale)

specifies the range of the x, y, and z coordinates: -xyzscale <= x,y,z <= xyzscale. The default is xyzscale = 1. This procedure should generally only be invoked once in each program, before the first invocation to a procedure that uses coordinate data. Note, if the range is changed after some coordinate data has been created, a locate (see section 6) in world coordinate space will return erroneous data.

Error Conditions:
1) Parameter must not be 0.0.

3.2 MODELLING TRANSFORMATIONS

Modelling transformations are specified at object call generation, and their scope is thus the called object. These transformations can be altered dynamically as was discussed in section 2.1. The modelling transformations are composed in the order in which they are specified by premultiplying the new transformation matrix with the existing matrix. To position a rotated, scaled subobject the transformations should appear in the order: translate, rotate, and scale.

```
SCALE2D(REAL VALUE xscale, yscale)
SCALE3D(REAL VALUE xscale, yscale, zscale)
```

specifies the x, y, and z scale factors of a subobject relative to its local coordinate system. The scale factors must be in the range 0.0 to 1.0. The default scale is xscale = yscale = zscale = 1.0.

Error Conditions:
1) Dimensionality mismatch.
2) This procedure can only be specified as part of an object call.
3) Numeric data out of range.

```
TRANS2D(REAL VALUE xtrans, ytrans)
TRANS3D(REAL VALUE xtrans, ytrans, ztrans)
```

specifies the x, y, and z translations of a subobject relative to its local coordinate system. The default translation is xtrans = ytrans = ztrans = 0.0.

Error Conditions:
1) Dimensionality mismatch.
2) This procedure can only be specified as part of an object call.
3) Numeric data out of range.

ROT2D(REAL VALUE angle)

specifies a rotation in radians of a 2D subobject about the origin of its local coordinate system. The default rotation angle is 0.0.

Error Conditions:
1) Dimensionality mismatch.
2) This procedure can only be specified as part of an object call.

ROT3D(REAL VALUE xcos, ycos, zcos, angle)

specifies a rotation of "angle" radians of a 3D sub-object about the vector defined by the direction cosines (xcos, ycos, zcos). The default rotation angle is 0.0.

Error Conditions:
1) Dimensionality mismatch.
2) This procedure can only be specified as part of an object call.
3) Numeric data out of range.

TRANSMAT(REAL ARRAY mat(*,*))

allows the user to specify an arbitrary 3x2 (2D) or 4x3 (3D) transformation matrix to be composed with the matrix specifying the modelling transformation.

Error Conditions:
1) Dimensionality mismatch.
2) This procedure can only be specified as part of an object call.
3) Illegal array bound.

3.3 VIEWING TRANSFORMATIONS

The viewing transformations are specified when a view is added to or changed in the display data structure, and define how the object is mapped to the display surface.

The viewing transformations of Theseus are very similar to those of the Core Graphics System. In 2D the viewing transformation is defined by a window and a viewport. In 3D it is defined by a view volume, a view plane, and a viewport. The view volume is either a pyramid for perspective projections, or a parallelepiped for parallel projections.

VRP(REAL VALUE x,y,z)

defines the view reference point in world coordinates. All 3D viewing parameters are defined relative to this point. The default point is (0.0,0.0,0.0).

Error Conditions:
1) This procedure can only be specified as part of ADDVIEW and CHGVIEW.
2) Dimensionality mismatch.
3) Numeric data out of range.

VPN(REAL VALUE dx,dy,dz)

defines the view plane normal as a vector relative to the view reference point. The default view plane normal is (0.0,0.0,1.0).

Error Conditions:
1) This procedure can only be specified as part of ADDVIEW and CHGVIEW.
2) Dimensionality mismatch.
3) All elements of the direction vector are zero. (The default has been used.)
4) Numeric data out of range.

VPD(REAL VALUE dist)

The viewplane distance, i.e. the distance from the view reference point to the view plane, is specified. The view plane is defined by the view plane normal and "dist". It is perpendicular to the view plane normal, and is at a distance "dist" from the view reference point. Distances are measured in world coordinate units from the view reference point, with positive values in the direction of the view plane normal, and negative values in the opposite direction. The default view plane distance is 0.0.

Error Conditions:
1) This procedure can only be specified as part of ADDVIEW and CHGVIEW.
2) Dimensionality mismatch.
3) Numeric data out of range.

PARALLEL(REAL VALUE dx,dy,dz)

defines the direction of the projectors of a parallel projection, relative to the view reference point. If the projectors are normal to the view plane an orthographic projection will result; otherwise an oblique projection will result.

Error Conditions:
1) This procedure can only be specified as part of ADDVIEW and CHGVIEW.
2) Dimensionality mismatch.
3) All elements of the direction vector are zero. (The default has been used.)
4) Numeric data out of range.

PERSP(REAL VALUE dx,dy,dz)

defines the center of projection of a perspective projection, relative to the view reference point. The default projection is perspective, with the center of projection at (0.0,0.0,-1.0) relative to the view reference point.

Error Conditions:
1) This procedure can only be specified as part of ADDVIEW and CHGVIEW.
2) Dimensionality mismatch.
3) Numeric data out of range.

VIEWUP2D(REAL VALUE dx,dy)
VIEWUP3D(REAL VALUE dx,dy,dz)

VIEWUP2D defines the direction in the world coordinate system that will be "up" on the display surface. The vector (dx,dy) is defined relative to the origin of the world coordinate system. VIEWUP2D defines a rotation of the window about the origin.

VIEWUP3D defines a vector (dx,dy,dz) whose orthographic projection onto the view plane defines the direction in the viewplane that will be "up" on the display surface. The vector (dx,dy,dz) is defined relative to the view reference point. The view-up vector allows the definition of a new coordinate system in the viewplane called the UV-system. The orthographic projection of the view-up vector defines the V-axis. The U-axis is defined such that the U-axis, the V-axis and the viewplane normal form a left-handed coordinate system. The window in the viewplane is defined in the UV-system.

The default view-up vector is (0.0,1.0) or (0.0,1.0,0.0).

Error Conditions:
1) This procedure can only be specified as part of ADDVIEW and CHGVIEW.
2) Dimensionality mismatch.
3) All elements of the direction vector are zero. (The default has been used.)
4) Numeric data out of range.

WINDOW(REAL VALUE xcenter, ycenter, xsize, ysize)

defines the center and size of the clip window. The dimensions of the window are xsize*2 by ysize*2.

In 2D the center and size are defined in the world coordinate system. The window is rotated about the origin of the world coordinate system so that the vertical sides of the window become parallel to the view-up vector.

The window and the viewport define the mapping of the 2D world coordinate space to the display surface. If clipping is enabled, only those parts of the object inside the window will be mapped to the viewport. If clipping is disabled, the entire world coordinate space is mapped to the display surface in such a way that the window is mapped to the viewport.

In 3D, the center and size are defined in the UV-coordinate system in the view plane. xsize is defined along the U-axis, ysize along the V-axis.

If a perspective projection is selected, the center of projection and the window define a semi-infinite pyramid. If a parallel projection is selected, the window and the direction of the projectors define a semi-infinite paralellepiped. If clipping is enabled, only the contents of the view volume, the pyramid or the parallelepiped, are projected onto the window in the view plane. If clipping is disabled, the entire world coordinate space is projected, by a parallel or perspective projection, onto the viewplane. The viewplane is then mapped to the display surface in such a way that the window is mapped to the viewport.

The default window is (0.0,0.0,xyzscale,xyzscale), where xyzscale defines the range of the world coordinate system.

Error Conditions:
1) This procedure can only be specified as part of ADDVIEW and CHGVIEW.
2) Numeric data out of range.

VPORT(REAL VALUE xcenter, ycenter, xsize, ysize)

defines the center and size of the viewport in screen coordinates, i.e., fractional coordinates between -1.0 and 1.0. The dimensions of the viewport are xsize*2 by ysize*2. The default viewport is the entire screen, i.e. (0.0,0.0,1.0,1.0).

Error Conditions:

1) This procedure can only be specified as part of ADDVIEW and CHGVIEW.
2) The viewport extends outside the screen area. (The default has been used.)

When a view of an object has been specified, the viewing parameters are checked for consistency to ensure the validity of the viewing transformation. Two situations give rise to inconsistencies:

- View-up direction is parallel to the view plane normal,
- projectors for a parallel projection are parallel to the view plane.

If either of these situations arise an error message "inconsistent view parameters specified" is given and a valid viewing transformation is defined so that processing can continue.

4 PRIMITIVES

Theseus provides line, point and text primitives. The primitives are added to the open segment, and are specified in absolute coordinates of a coordinate system local to the object. Each primitive may have an associated element name for identification purposes, and an associated dynamo name for modification purposes.

The text primitives are part of two- or three-dimensional objects. Characters in the text strings are individually positioned, conforming to the user's specification as closely as the hardware character generator permits.

Individual primitives may not be deleted from a segment. They may, however, be altered if referenced by a dynamo. The type of the primitive may not be changed, but a point and an endpoint of a line may be replaced by another point, and a text string with another text string. Note that the length of a primitive, i.e. the number of lines or the number of characters, must remain the same.

Menu text is used for operator/machine interaction. Menus are objects that can be added to the display data structure by creating a view of the menu object. A menu is displayed in a user-defined viewport, but the user has no control over size and position of the menu text within the viewport.

4.1 LINE AND POINT PRIMITIVES

Theseus provides procedures for simple primitives such as points and lines, and for composite primitives such as polylines and polygons.

```
LINE2D(REAL VALUE x1, y1, x2, y2; INTEGER VALUE [elemname],
    [dynamo], linestyle)
LINE3D(REAL VALUE x1, y1, z1, x2, y2, z2; INTEGER VALUE
    [elemname], [dynamo], linestyle)
```

specify a line from (x1, y1) to (x2, y2) of a two-dimensional object, and a line from (x1, y1, z1) to (x2, y2,

z2) of a three-dimensional object, respectively. "linestyle" determines the type of line:

0 - solid
1 - dashed
2 - dotted

An element name may be specified for identification of, and a dynamo name for modification of the primitive.

Error Conditions:
1) No open segment.
2) No more core available for graphical data structure.
3) Dimensionality mismatch.
4) Numeric data out of range.
5) Invalid line style. (Solid has been used.)
6) Dynamo name already exists.

POINT2D(REAL VALUE x, y; INTEGER VALUE [elemname], [dynamo])
POINT3D(REAL VALUE x, y, z; INTEGER VALUE [elemname], [dynamo])

specify a point at (x, y) and (x, y, z), respectively. An element name may be specified for identification of, and a dynamo name for modification of the primitive.

Error Conditions:
1) No open segment.
2) No more core available for graphical data structure.
3) Dimensionality mismatch.
4) Numeric data out of range.
5) Dynamo name already exists.

POLYLINE(REAL ARRAY xyz(*,*); INTEGER VALUE [elemname], [dynamo])

defines a sequence of connected solid lines starting at the first coordinate point in xyz and ending at the last. The dimensionality is determined by the first array index, which is either 2 or 3 depending on whether a 2D or 3D polyline is to be generated.

An element name may be specified for identification of, and a dynamo name for modification of the primitive.

Error Conditions:
1) No open segment.
2) No more core available for graphical data structure.
3) Dimensionality mismatch.
4) Numeric data out of range.
5) Dynamo name already exists.

```
POLYGON(REAL ARRAY xyz(*,*); INTEGER VALUE [elemname], [dynamo])
```

is identical to POLYLINE except that the first and the last coordinate points in xyz are connected.

```
CHGPT2D(INTEGER VALUE dynamo, coordoffset; REAL VALUE x, y)
CHGPT3D(INTEGER VALUE dynamo, coordoffset; REAL VALUE x, y, z)
```

modify one coordinate in a two- and three-dimensional primitive, respectively. The primitive is referenced by the dynamo, and the actual point within the primitive is determined by "coordoffset", which is an offset into the set of coordinates that determine the primitive. "coordoffset" is always one for a point primitive, and the coordinateset number for the point in a line primitive that is to be modified.

Error Conditions:
1) Dynamo does not refer to a correct type.
2) Dimensionality mismatch.
3) Numeric data out of range.
4) Dynamo does not exist.
5) Coordinate set number out of range.

4.2 TEXT

```
TEXT2D(REAL VALUE x, y, dx, dy; STRING (120) VALUE chars; INTEGER
     VALUE length, [elemname], [dynamo])
TEXT3D(REAL VALUE x, y, z,  dx, dy, dz; STRING (120) VALUE chars;
     INTEGER VALUE length, [elemname], [dynamo])
```

defines a text string of length "length" with the center of the first character at (x, y) and (x, y, z), respectively. The inter-character distance is defined in world coordinate units by (dx, dy) or (dx, dy, dz) and defines the coordinate position for the center of each of the following characters. The coordinate positions are transformed by the composite transformation matrix, and the individual characters are displayed, in the screen plane, with the center of each character at the transformed character position. A character size appropriate for the transformed string length is chosen by Theseus.

An element name may be specified for identification of, and a dynamo name for modification of the primitive.

Error Conditions:
1) No open segment.
2) No more core available for graphical data structure.
3) Dimensionality mismatch.
4) Numeric data out of range.
5) Text length not in the range 0 to 120; 120 has been used.
6) Dynamo name already exists.

CHGTEXT(INTEGER VALUE dynamo, length; STRING (120) VALUE chars)

replaces the text string referenced by the specified dynamo. Note that if the new string is longer that the original string it will be truncated; if it is shorter it will be padded with blanks.

Error Conditions:
1) Dynamo does not refer to a correct type.
2) Dynamo does not exist.
3) Text length is either less that zero, or greater than length of string being modified. (It has been set to the length of the existing string.)

4.3 MENU TEXT

MENU(INTEGER VALUE name; STRING (120) ARRAY (*) chars; INTEGER VALUE length, elemname1)

defines a menu object with the name "name". A menu is a special object containing a text strings and can only be referenced from ADDVIEW or CHGVIEW. The text strings are displayed in the viewport defined by the viewing transformation. All other viewing parameters of the view are ignored.

"length" is the length of the longest string in "chars", and "elemname1" defines the element name of the first menu text string in "chars". The following strings have the element names "elemname1"+1, "elemname1"+2, etc.

Each string will be displayed preceeded by a bullet. A text string is identified by a pick on this bullet. Upon a pick of a menu string, the attention queue will contain the triplet (name,0,elemname), and the picked text string will be highlighted.

Error Conditions:

1) Object name already exists.
2) No more core available for graphical data structure.
3) Invalid name; must be a positive integer.
4) Text length not in the range 0 to 120; 120 has been used.

5 ATTRIBUTES

Attributes allow the user to manipulate the images of objects and segments. These attributes are intensity, pickability, blink, vector mode, clip enable, extent enable, size enable, and invertibility. The attributes are specified at generation of object calls, views, and segments, and can be altered by the procedures CHGCALL, CHGVIEW, and CHGSEG. If an attribute specification is omitted when a view or an object call is defined, a default is applied; if it is omitted at segment definition, a segment without attributes is generated. The attributes of views, object calls, and segments are composed by the DPU as the hierarchical data structure is processed in a manner described below.

INTENS(REAL VALUE intens)

defines the intensity at which the image of a object or a segment is displayed. "intens" can range from 0.0 to 1.0; 0.0 is the dimmest, 1.0 the brightest. When attributes are composed at a subobject call or at the beginning of a segment, the maximum of the two intensities is chosen as the new attribute. The default is intens = 0.5.

Error Conditions:
1) Attributes can only be part of an object call, a view, or a segment.
2) Numeric data out of range.

PICKABLE(LOGICAL VALUE pick)

This attribute, when true, enables the object or segment for picking (see section 6). When attributes are composed, the or of the two attributes is chosen as the new attribute. This implies that when an object is pickable, so are all its segments and subobjects. The default is pick = true.

Error Conditions:

1) Attributes can only be part of an object call, a view, or a segment.

BLINK(LOGICAL VALUE onoff)

This attribute, when true, causes the image of an object or segment to blink until the attribute is set to false. When the attributes are composed the or of the two attributes is chosen as the new attribute. This implies that when an object is blinking, so are all its segments and subobjects. The default is onoff = false.

Error Conditions:
1) Attributes can only be part of an object call, a view, or a segment.

VECMODE(LOGICAL VALUE onoff)

This attribute, when true, changes the line style for the image of an object or a segment. Solid lines will change to dashed, dashed to dotted, and dotted lines to endpoints. When attributes are composed the or of the two attributes is chosen as the new attribute. The default is onoff = false.

Error Conditions:
1) Attributes can only be part of an object call, a view, or a segment.

CLIP(LOGICAL VALUE onoff)

allows the user to enable and disable clipping of an object or segment against the window or view volume. When the attributes are composed the or of the two attributes is chosen as the new attribute. The default is clipping enabled, i.e. onoff = true.

Error Conditions:
1) Attributes can only be part of an object call, a view, or a segment.

EXTCON(LOGICAL VALUE onoff)

allows the user to enable and disable extent processing. If disabled, the DPU will process each primitive in an object for clipping. This attribute is mainly used for debugging. When the attributes are composed, the or of the two attributes is chosen as the new attribute. The default is extent processing enabled, i.e. onoff = true.

Error Conditions:
1) Attributes can only be part of an object call, a view, or a segment.

SIZECON(LOGICAL VALUE onoff)

allows the user to enable and disable the conditional test that determines if an object is too small or too large relative to the object extent size, to be processed for display. If disabled, the segment or object will be processed regardless of its size. This attribute is used for debugging and for creating hard-copy output of a picture with all its detail. When the attributes are composed, the or of the two attributes is chosen as the new attribute. The default is size processing enabled, i.e. onoff = true.

Error Conditions:
1) Attributes can only be part of an object call, a view, or a segment.

INVERT(LOGICAL VALUE onoff)

This attribute when true, enables locator data (see section 6) in a viewport to be returned in world coordinates instead of screen coordinates. This attribute can only be specified by ADDVIEW and CHGVIEW that refer to 2D objects, and applies to the viewport of the associated viewing transformation. Locators are discussed in section 6. The default attribute is onoff = false.

Error Conditions:

1) The INVERT routine can only be an attribute of a view of a 2D object.

6 INTERACTION

Theseus supports five logical interaction devices:

PICK - identify object, segment, or primitive
BUTTON - select function
KEYBOARD - provide alphanumeric information
LOCATOR - provide coordinate information
VALUATOR - provide values

These logical devices may be implemented by one or more physical devices, and the programmer may choose which physical interaction device on BUGS most conveniently implements a logical device for his application.

The logical devices are divided into two classes: event causing devices and sampling devices. Each event causing device has an associated one-element event queue. Upon an event, an event report with data related to the event is placed on this queue. The event report must be removed from the queue by the application program before a new event report may be placed on the queue. The application program may poll any queue, or wait for an event from one or more devices. Sampling devices have values that may be sampled by the application program. The pick, button, and keyboard are event causing devices, the valuator a sampling device. There are two kinds of locators: a sampling device that returns screen coordinates and an event causing device that returns world coordinates.

Each logical device and its physical implementation is described below. The event handling functions for polling and waiting, and for accessing event reports are described last.

6.1 PICK

A pick device is an event causing device that may identify a view, object, segment or primitive in the data structure. The event report contains the DPU object call stack of (object, segment, elemname) triplets at the time of the pick. The programmer may obtain any triplet in the stack for identification purposes, or assign a dynamo to any item in the stack for modification purposes. If viewports overlap, a pick will occur in the first viewport with a pickable item.

The position of the pick window may be explicitly controlled by the programmer or implicitly controlled by attaching it to the joystick or to the tablet. Theseus only supports one cursor/pickwindow, so a pick device and a locator cannot both be enabled.

PICKEXPL(REAL VALUE x, y, xscale, yscale)

enables the logical pick device for explicit picking. A pick-window of dimensions xscale*2 by yscale*2 is initially positioned at (x,y). x, y, xscale, and yscale are all specified in screen coordinates.

Error Conditions:
1) A pick device is already enabled.
2) Numeric data out of range.
3) A locator device is already enabled.

POSPICK(REAL VALUE x, y)

moves the center of the pick window to (x, y).

Error Conditions:
1) Explicit pick device has not been enabled.
2) Numeric data out of range.

PICKIT

performs an explicit pick at the current position of the pick window. If the pick is successful, the event report is placed on the pick event queue. This function and POSPICK allow the programmer to use any combination of sampling and event causing devices to perform a pick. Note, this function may be used to perform a pick when the joystick or the tablet is enabled as the pick device.

Error Conditions:
1) A pick device is not enabled.

PICKJOY(INTEGER VALUE fnkey, REAL VALUE xscale, yscale)

enables the joystick as a pick device. A pick-window of dimensions xscale*2 by yscale*2 is positioned at the current (x,y) joystick coordinates. The function key specified by "fnkey" is implicitly enabled and serves as a trigger. If the pick is successful, the event report is placed on the pick event queue. "xscale" and "yscale" are specified in screen coordinates.

Error Conditions:
1) A pick device is already enabled.
2) Function key is already enabled.
3) Numeric data out of range.
4) A locator device is already enabled.
5) Invalid function key number.

PICKTAB(REAL VALUE xscale, yscale)

enables the tablet as a pick device. A pick window of dimensions xscale*2 by yscale*2 is positioned at the current (x,y) tablet coordinates. The pick occurs when the tip-switch is pressed. If the pick is successful, the event report is placed on the event queue. "xscale" and "yscale" are specified in screen coordinates.

The user should be aware that the tip-switch cannot be disabled, and will cause a pick to occur when it is pressed no matter which physical device is enabled as a pick device.

Error Conditions:
1) A pick device is already enabled.
2) Numeric data out of range.
3) A locator device is already enabled.

DISPICK

disables the current pick device, and flushes the queue for the pick device.

Error Conditions:

1) A pick device is not enabled.

6.2 BUTTON

A button is an event causing device that allows the programmer to select a function. The event report contains the number of the button. The buttons are implemented as function keys.

BUTKEY(INTEGER VALUE number)

enables the specified function key as a button, and lights the corresponding function key light.

Function keys 0 - 31 give one interrupt when pressed, whereas function key 32 is a continous interrupt key. All keys are enabled and disabled as buttons in the same manner, but the light for key 32 is not lit.

Error Conditions:
1) Function key is already enabled.
2) Invalid function key number.

BUTKEYS

enables all function keys as buttons, exept for any function key that is already enabled as a trigger. The corresponding function key lights (exept that for key 32) are lit.

DISBUT(INTEGER VALUE number)

disables the specified button, and flushes the button queue of its event report.

Error Conditions:
1) Button is not enabled.
2) Invalid function key number.

DISBUTS

disables all buttons, and flushes the queue for the buttons.

6.3 KEYBOARD

The keyboard is an event device that allows the user to enter alphanumeric text. The keyboard event occurs when the carriage return is typed, and the event report contains the text string that was entered and the number of characters in that string.

Theseus maintains and displays a one-line character buffer for prompt text and for input from the alphanumeric keyboard. The prompt text and the input text are displayed in a programmer-defined viewport at a programmer-defined cursor position. The input string may be edited with left arrow to delete a character and HOME to delete a line.

KEYVIEW(REAL VALUE xcenter, ycenter, xsize, ysize)

specifies the center and size of the text input viewport in screen coordinates. The dimensions of the viewport are xsize*2 by ysize*2. The default viewport is (0.0,0.0,1.0,1.0), i.e. the entire screen.

Error Conditions:
1) Numeric data out of range.
2) The viewport extends outside the screen area. (The default has been used.)

KEYBUF(INTEGER VALUE length)

specifies the total buffer size for the combined prompt message/input line. The maximum allowable buffer size is 120, and the default is 80 characters. The buffer size and the viewport size determine the size of the characters; the character size is chosen so that the entire buffer will fit on one line in the viewport. If the user tries to input beyond the length of the buffer the input is ignored.

Error Conditions:
1) Parameter out of range.

PROMPT(REAL VALUE x, y; STRING (120) VALUE message; INTEGER VALUE length)

enables the alphanumeric keyboard for input. A prompt message "message" of length "length" characters followed by a cursor is displayed at the coordinate position (x,y). (x,y) are specified in the range -1.0 to 1.0, and are coordinates in the input viewport defined by KEYVIEW. Upon carriage return the event report is placed on the event queue, and the keyboard is disabled for input.

Error Conditions:
1) Numeric data out of range.
2) Text length is either less than zero, or greater than the maximum buffer size. (It has been set to the maximum buffer size.)

6.4 LOCATOR

A locator is either a sampling device that provides coordinate information in screen coordinates, or an event device that provides coordinate information in world coordinates. The locator event report contains the world coordinates of the locator position in the first viewport with the invert attribute enabled. It also contains the view name corresponding to this viewport. When the locator position is mapped to world coordinates it is assumed that the viewing transformation has its view-up vector along the y-axis.

The locator may either be controlled by the joystick or the tablet. Theseus only supports one cursor/pick-window, so a locator and a pick device cannot both be enabled.

LOCJOYS

enables the joystick as a sampling locator. A cursor is displayed at the current (x,y) joystick position.

Error Conditions:
1) A locator device is already enabled.
2) A pick device is already enabled.

LOCTABS

enables the tablet as a sampling locator. A cursor is displayed at the current (x,y) tablet position.

Error Conditions:
1) A locator device is already enabled.
2) A pick device is already enabled.

READLOC(REAL RESULT x, y)

reads the current locator. (x,y) is returned in screen coordinates.

Error Conditions:
1) A locator device is not enabled.

LOCJOYE(INTEGER VALUE fnkey)

enables the joystick as an event locator. A cursor is displayed at the current (x,y) joystick position. The function key specified by "fnkey" is implicitly enabled and serves as a trigger. If a locate is successful, i.e., if the locator is positioned in a viewport with the invert attribute enabled when the function key was pressed, the event report is placed on the locator event queue.

Error Conditions:
1) A locator device is already enabled.
2) A pick device is already enabled.
3) Function key is already enabled.
4) Invalid function key number.

LOCTABE

enables the tablet as an event locator. A cursor is displayed at the current (x,y) tablet position. The tipswitch is used as the trigger device. If a locate is successful, i.e., if the locator is positioned in a viewport with the invert attribute enabled when the tipswitch is pressed, the event report is placed on the locator event queue.

Error Conditions:
1) A locator device is already enabled.
2) A pick device is already enabled.

DISLOC

disables the locator, and flushes the queue for the locator.

Error Conditions:
1) A locator device is not enabled.

6.5 VALUATOR

A valuator is a sampling device that allows the user to read numeric values in the range 0.0 to 1.0. There are 10 one-dimensional valuators: the 10 dials; and 2 three-dimensional valuators: the joystick and the tablet.

READDIAL(INTEGER VALUE number)

is a real procedure that returns the value of the dial specified by "number".

Error Conditions:
1) Parameter out of range.

READJOY(REAL RESULT x, y, z)

returns the current (x,y,z) values of the joystick.

READTAB(REAL RESULT x, y, INTEGER RESULT z)

returns the current (x,y,z) values of the tablet. z indicates the position of the tablet pen:

0 - out of range of tablet
1 - in range of tablet
2 - touching tablet.

6.6 EVENT HANDLING

This section will discuss how the event queues can be polled, how an application program can wait for an event, and how to access the event reports.

POLL(LOGICAL expr)

is an integer procedure that polls the event queues for one or more devices. "expr" is a boolean expression of the form:

BUTTON OR PICK OR KEYBOARD OR LOCATOR.

Theseus will poll the event queues of the devices in the boolean expression in the order in which they are specified. If an event occurred, an integer equal to the index of the event device in the expression will be returned. If no event occurred, POLL will return the number of devices in the boolean expression plus one. For example, POLL(BUTTON OR LOCATOR OR PICK) would return:

1 for a button event
2 for a locator event
3 for a pick event
4 if no event occured.

This function removes the corresponding event report from the queue. The dequeued report become the current event report for that device, and can be accessed by the functions described below.

Error Conditions:
1) Invalid routine in boolean expression.

WAIT (LOGICAL expr)

works much in the same fashion as POLL, except that WAIT will not return until an event has occurred. The queues will continuously be polled in the order in which the devices are specified in the boolean expression.

Error Conditions:
1) Invalid routine in boolean expression.

STACK

is an integer procedure that returns the depth of the object call stack in the current pick event report.

Error Conditions:
1) Current event report for the designated device does not exist.

GETPICK(INTEGER VALUE depth; INTEGER RESULT object, segment, elemname)

returns the triplet name (object, segment, elemname) at level "depth" from the top of the object call stack in the current pick event report. The top (level1) item of the stack contains (0,viewname,0). The bottom element contains the (object, segment, elemname) triplet for the primitive that was picked. Intermediate levels contain triplets for the intervening calls. If an element name was not specified for an object call or a primitive, an elemname of 0 is returned.

Error Conditions:
1) Current event report for the designated device does not exist.
2) Parameter out of range. (Depth exceeds number of items on the stack.)

DYNELEM(INTEGER VALUE depth, dynamo)
DYNSEG(INTEGER VALUE depth, dynamo)

create a dynamo with the name "dynamo", referring to the item corresponding to the triplet (object, segment, elemname) and the pair (object, segment) at level "depth" from the top of the object call stack in the current pick event report. DYNELEM creates a dynamo for a primitive, a view, or an object call, whereas DYNSEG creates a dynamo for a segment, if this segment has attributes associated with it.

Error Conditions:
1) Current event report for the designated device does not exist.
2) Parameter out of range. (Depth exceeds number of items on the stack.)
3) Dynamo name already exists.
4) Invalid name; must be a positive integer.
5) No more core available for graphical data structure.
6) Object does not exist. (It has been deleted since the pick.)
7) Segment does not exist. (It has been deleted since the pick.)
8) Attempt to associate a dynamo with a segment that does not have associated attributes.

GETBUT

is an integer procedure that returns the number of the function key that caused the button event.

Error Conditions:
1) Current event report for the designated device does not exist.

GETTEXT(INTEGER RESULT number; STRING (120) RESULT text)

returns the length of input character string in the keyboard event report and the characters in this string.

Error Conditions:
1) Current event report for the designated device does not exist.

GETLOC(REAL RESULT x, y; INTEGER RESULT viewname)

returns the locator data of the current locator event report. If "viewname" is zero, the locate was unsuccesful.

Error Conditions:
1) Current event report for the designated device does not exist.

7 CONTROL

This section discusses a set of unrelated issues regarding the operation of Theseus.

7.1 INITIALIZATION

A series of Theseus programs using the same graphical data structure can be executed under GMS. Theseus is invoked by executing the following command:

THESEUS <filename> .... <filename> [(<options>]

<filename> is a file with filetype MODU. The files are executed in the order in which they are specified, and operate on the same graphical data structure. The options can be specified in any order, separated by blanks. Theseus has the following options (in addition to the ALGOL W ones):

SD=x is the maximum depth of the object call stack. The stack depth must not be greater than 10; the default is SD=5.

D=xxxx specifies the dimensionality of the objects as either 2D, 3D or BOTH. The default is DIM=2D.

PCF=xx specifies the amount of core in K-bytes that Theseus uses for the largest object program. Theseus will utilize the remainder of core for the graphical data structure and the Algol W stack frame.

GCF=xx specifies the amount of core in K bytes that Theseus should use for its graphical data structure.

Note that only one of PCF and GCF should be specified. If both are specified then the latter will be used, and if both are omitted the default is GCF=4.

7.2 TERMINATION

Theseus is terminated and the space for the graphical data structure freed up at the end of the execution of the last program on a single command line.

7.3 CLOCK

The "frame" clock is not treated as an interaction device and is not accessible by Theseus. However, Algol W provides access to the CPU clock.

7.4 DEFAULTS

The attributes and the transformations have defaults defined by Theseus. These defaults are applied at view and object call generation if the attributes or transformations are not explicitly specified by the programmer. The defaults may not be altered by the programmer.

7.5 PICTURE FILES

Theseus allows a user to save an object on disk and to retrieve it for later use. Each object is saved in a separate disk file.

```
SAVEOBJ(INTEGER VALUE name, STRING(8) VALUE file)
```

saves the object "name" and all its subobjects in a disk file named "file". Note that the object is not deleted from the data structure.

Error Conditions:
1) Object does not exist.
2) File with the specified name already exists.

LOADOBJ(STRING(8) VALUE file) loads the object in the file with the name "file". Care must be taken that there are no duplicate object names in the disk file and the graphical data structure.

Error Conditions:
1) File with the specified name does not exist.
2) Object name already exists.

7.6 INQUIRY

Theseus does not provide any functions to inquire about the contents of the graphical data structure. At some later date, when facilities for higher-level interaction and dynamics are added, it is expected that functions to inquire about primitives and transformations will be available.

7.7 ERROR HANDLING

Runtime errors in Theseus are handled as runtime errors in Algol W. The error number and the coordinate of the Theseus statement that caused the error are printed on the operator's console. FUDD may be used just as for Algol W.

There are two types of errors in Theseus: warnings and fatal errors. At a non-fatal error the procedure call will be ignored unless other action is indicated in the error message, and execution will continue. If an error occurs in a Theseus procedure that is passed as a parameter, only that particular procedure will be ignored. For example, if an error occurs in an attribute routine, only that attribute routine is ignored. The other attribute routines as well as the segment, view, or object call definition will be processed.

Note that procedures may return unpredictable values if the procedure generated an error.

There is only one fatal error: "No more core available for graphical data structure."

7.8 HARD COPY