Programs Written For SimNet Terrain Generation

Technical Report

M. Peri Cope
Michael Paul Czechanski
Tobi L.S. Sellekaerts

February 1998

Prepared for:

7th Army Training Command (7th ATC)
Grafenwoehr, Germany

Terrain Modeling Project Office (TMPO)
Washington, D.C.

Under Contract:
DAJA 22-96-D0069

Prepared by:

Logicon, RDA
Grafenwoehr, Germany

Table of Contents

1.0 Introduction **3**
1.1 Process overview 3
1.1.1 Elevation skin *3*
1.1.2 Features *3*
1.1.3 Database Compilation in S1000 *4*
1.1.4 Models *4*
1.1.5 Texturing *4*
1.1.6 Load module overload *4*
1.1.7 Paper maps *5*
1.2 Future actions 5
2.0 Programs **6**
2.1 Program overview 6
2.2 Individual program descriptions and source code 8
2.2.1 feat_for.aml (feat_alt.aml) *8*
2.2.2 grafplot.aml *18*
2.2.3 import_itin.aml *36*
2.2.4 import_s1kmodels.perl *38*
2.2.5 itin2vrml.prl (itin2vrml_sgi.prl) *40*
2.2.6 itin_conv.prl (itin_conv_sgi.prl) *46*
2.2.7 itin_script.prl *52*
2.2.8 lake_elev.aml *54*
2.2.9 model_format.perl *57*
2.2.10 remove_lm.perl *59*
2.2.11 stamp_conv.prl *61*
2.2.12 s1k_fence.aml *63*
2.2.13 s1k_land.aml *66*
2.2.14 s1k_net.aml *69*
2.2.15 s1k_texture.perl *73*
2.2.16 s1k_treeln.aml *77*
2.2.17 s1k_wtr.aml *80*
2.2.18 s1kc.prl (s1kc_sgi.prl) *83*
2.2.19 tin1.aml *112*
2.2.20 tin2.aml *116*
2.2.21 tin3.aml *118*
2.2.22 tin3perl *124*
3.0 Available SimNet Textures **128**
4.0 References **130**
4.1 Documents 130
4.2 Personal references 131
4.3 Presentations 131
5.0 Points of Contact **132**

1.0 Introduction

Terrain Simulation (TerraSim) is a Logicon lab contracted to the U.S. Army Europe (USAREUR) 7th Army Training Command (7th ATC) Battle Simulation Center (BSC). TerraSim rapidly creates simulation terrain, Geographic Information System (GIS) applications, and custom cartographic products for exercise and contingency operation support in the European Command (EUCOMM) area of responsibility (AOR).

1.1 Process overview

In the past, the generation of three dimensional terrain surfaces has proven to be a very labor and resource intensive process. Due to the many constraints imposed by the underlying simulator and image generation architecture, the limitations and inconsistencies among input data sources, and lack of standardized tools for creating such databases, the time gap between order and delivery of final products has been quite large. In an effort to narrow this gap significantly while working to meet current contractual obligations, the three dimensional terrain construction team of TerraSim has been combining a variety of available resources and tools with methos generated in-house to provide a more automated system of SimNet database production. This paper briefly describes these resources, tools, and methods, and describes and includes code for programs written in support of TerraSim's database generation efforts. A later revision of this paper will include a more thorough description of database construction methods with an IDEF0 process model description.

1.1.1 Elevation skin

Perhaps the most significant element of any SimNet terrain database is the underlying elevation structure, which is generally defined by a triangulated irregular network (TIN). A significant portion of our work to date has been focused on creating accurate and economic TIN structures for incorporation in SimNet. Initially, programs were written in the Arc Macro Language (AML) to generate optimized TINs using Arc/Info geographic information system (GIS) software. This work was based on research conducted by the Digital Mapping laboratory (DML) at Carnegie-Mellon University (CMU). Currently, our work has been set aside in favor of incorporating the iTIN software package being developed by DML / CMU. The iTIN suite allows the creation of optimized TINs from National Imagery and Mapping Agency (NIMA) digital terrain elevation data (DTED) or similar digital elevation models (DEMs). Where digital elevation data does not currently exist, we can create it from map contours, IFSAR data, or stereo imagery pairs. iTIN also provides tools for integrating linear features, such as roads, and polygonal features, such as lakes, into the terrain surface.

Optimized TINs contain the minimal number of polygons and points necessary to accurately portray a given landscape while at the same time maintaining adequate locational accuracy. Integrating linear features into these TINs adds to terrain database fidelity by ensuring that roads and lakebeds are level and that streams and rivers flow downhill.

TIN generation for SimNet is as much an art as it is a science. Because so many factors can contribute to or diminish from the overall effectiveness of a given TIN for representing a particular landscape, many subjective choices must be made to ensure that the output TIN is suitable for the projected need. For instance, one must decide beforehand which areas of the terrain playbox should receive special emphasis in the final TIN. Should the mountains and areas of high elevation be highly detailed or should the majority of the TIN polygon budget be used for defining low lying terrain? This is one example of the many questions that must be considered in building a suitable input TIN.

1.1.2 Features

In addition to elevation information, we incorporate the following features into our SimNet databases:

Point	Line	Polygon
buildings	highways	urban
vegetation stamps	other major roads	forest canopies
	airport runways and taxiways	swamps
	streams and rivers	lakes and oceans
	fences	surface vegetation
	walls	
	treelines	

Arc/Info is used for importing, processing, updating, and exporting feature data to S1000 (for point features, only the locations are exported to S1000). This gives us a great deal of flexibility in selecting data sources and the ability to update our data from imagery or photography. At TerraSim we build terrain for many different simulations. Feature data in processed in Arc/Info can be reused in other simulations such as JTS and Spectrum; databases built for these other simulations can be used in SimNet.

1.1.3 Database compilation in S1000

Before a database is loaded into SimNet, it must first be assembled and compiled in the appropriate format. This is done using S1000, a product developed and fielded by Lockheed Martin, Inc. In order to incorporate the many NIMA digital and analog mapping products that are readily available to us, we have also written a series of short programs in AML and Perl that will convert them to S1000 compatible text files. At the present time, variables within these programs are hard coded to meet a specific objective; however, they can be modified easily to support additional input data formats or to supply different output parameters as needed.

The primary disadvantage to using the S1000 package is the lack of reasonable documentation and instructions. Because this program is considered by some to be a 'legacy' product, at one time we thought that it might be necessary to find other software for compiling SimNet terrain databases. However, as SimNet is due to be phased out in USAREUR and replaced by CCTT, we currently plan to continue using S1000.

1.1.4 Models

We build all 3 dimensional models, such as buildings and tanks, in S1000's Model Tool. Each model is built to multiple levels of detail (LOD); most models in the GTA database have 3 LODs. We currently have solid colors applied to all model surfaces, not textures. If the same type of model occurs many times in the database, it only needs to be built once and then it is placed in multiple locations at the appropriate orientation and scale.

1.1.5 Texturing

Once a TIN or model has been built and is being prepared for use in SimNet, it will be necessary to apply one or more textures to its surface. Textures are images that represent what the surface of these objects would look like in the real world. For instance, a digital picture of a brick wall could be used as a texture for the model of a building. Initially we expected to create our own textures for our SimNet databases. However, now we know that the textures available for SimNet are predetermined and we cannot add to or change them. We experimented with the different available textures at length, determing which were most applicable to our database. See **3.0 Available SimNet Textures**.

1.1.6 Load module overload

Even though simulators are powerful visualization tools, they are not capable of loading and displaying an entire terrain database at one time. For this reason, the terrain is always broken up into smaller units called load modules. In the case of SimNet, load modules are squares, 500 meters on each side. In SimNet you can see 3500 meters - 7 load modules - away. Each load module has a polygon budget associated with it - this budget indicates the maximum number of polygons within a load module that can be reasonably displayed by the simulator image generator. We have found that we cannot exceed approximately 100 polygons per load module without the final database having holes.

1.1.7 Paper maps

In addition to providing digital terrain data for use in SimNet, a complete terrain includes paper maps to accompany the database. These maps are used for a variety of needs before, during, and after the actual operation of the simulator. We have written programs to support the automated generation of these maps. Working from the same data sources as those used to create the digital terrain database, we generate complete sets of standardized 1:50,000 scale maps as needed to meet the database requirements.

1.2 TerraSim future actions

TerraSim terrain builds are dictated by the 7th Army Training Command, and we will build SimNet databases over additional areas as requested. However, there are many refinements and improvements we can currently make. We need to complete the following tasks:

- Write an IDEF0 description of our terrain building process with appropriate graphics
- Use the database for an exercise
- Make tree canopies and road networks less slivery
- Improve land cover areas (i.e. urban) without a dramatic increase in polygon count
- Rewrite the map making AML (grafplot.aml) to be less geospecific
- Provide our database and models to TEC for distribution to other S1000 government users
- Build a database while better documenting the process
- Do a timed database build
- Build a database over an area with power lines

2.0 Programs

The Program Overview gives a short description of each program's purpose for quick reference. More thorough descriptions follow. For each program we have included a description of the purpose, history, usage, sample output, and program flow. Following this description is the source code for each program. The AMLs were written using Arc/Info for Solaris version 7.0.4. Our perl script binary executable resides in `/usr/local/bin/perl` on the Sun workstations and `/usr/local/sgi/bin` on the Silicon Graphics workstations.

2.1 Program Overview

feat_for.aml (feat_alt.aml)

- deletes channels and areas from a vegetation polygon coverage where roads, rivers,and other features pass through the vegetation polygons with (feat_for.aml) and without (feat_alt.aml) islands included.

grafplot.aml

- generates plot files for paper maps of all the features included in the final S1000 / SimNet database, including title and legend information.

import_itin.aml

- converts TIN addwams files into Arc/Info point and polygon coverages. It was written specifically to handle addwams files created by iTIN.

import_s1kmodels.perl

- converts an S1000 models text file into a text file readable by the Arc/Info GENERATE command. When thinning buildings in S1000, it is sometimes necessary to do the thinning in S1000 and not in Arc/Info. In order to include the buildings on the final paper maps, you have to import these final building locations back into Arc/Info.

Itin2vrml.prl (itin2vrml_sgi.prl)

- converts ITIN **.**addwams files into VRML 1.0 **.**wrl files for viewing in a VRML 1.0-compliant browser.

itin_conv.prl (itin_conv_sgi.prl)

- takes a set of incorrectly formated ADDWAMS files that have been generated as output by ITIN and convert them into properly formatted ADDWAMS files that can be correctly read and interpreted by S1000.

Itin_script.prl

- executes commands the CMU iTIN program in a sequence designed to create a complete SimNet TIN from start to finish.

lake_elev.aml

- finds the elevation of a polygonal water feature (the elevation at the label point / center of the polygon) so the polygon can be used as a hard return in TIN construction.

model_format.perl

- reads a text file that contains positional, size, and orientation information about models to be placed on the SimNet terrain, and converts it into a text file in the required S1000 model text file format.

remove_lm.perl

- removes load module edges from a set of addwams TIN files. It was written specifically to handle addwams files created by iTIN.

s1k_fence.aml

- converts Arc/Info line coverages (such as fences) into an ADDWAMS file that can be imported into S1000.

s1k_land.aml

- converts Arc/Info polygon (land cover) coverages (such as urban and swamp) into an ADDWAMS file that can be imported into S1000.

s1k_net.aml

- converts Arc/Info line (network) coverages (such as roads and railroads) into an ADDWAMS file that can be imported into S1000.

s1k_texture.perl

- creates a texture file called s1k.tlib.txt for use in creating a texture library in S1000.

s1k_treeln.aml

- converts Arc/Info line coverages (such as treelines) into an ADDWAMS file that can be imported into S1000.

s1k_wtr.aml

- converts Arc/Info polygon (hydro) coverages (such as lakes) into an ADDWAMS file that can be imported into S1000.

s1kc.prl (s1kc_sgi.prl)

- prompts the user to supply input data that is used to create a control file that can be used with the `s1kperfly` program to view and navigate through terrain databases created in S1000.

tin1.aml

- takes in an Arc/Info coverage and creates a new coverage - a polygon coverage covering the same extents as the input coverage with 500m square polygons (load modules).

tin2.aml

- thins a grid created from DTED by querying on a neighborhood basis in the Arc/Info GRID module. If a cell has the same value as all the cells touching it, or is within a certain slope value, that cell is given a value of NODATA.

tin3.aml

- takes a completed TIN (created manually) and an Arc/Info polygon coverage delineating load module boundaries (created with tin1.aml), and exports them to the appropriate S1000 addwams format.

tin3perl

- reformats MOSS format TIN elevation text files created by tin3.aml.

2.2 Individual program descriptions and source code

2.2.1 feat_for.aml

PURPOSE: A line or ploygon feature coverage is buffered and then cut out of the forest coverage.

HISTORY: This program was written by Michael Paul Czechanski in August 1997. An alternate version of the program, feat_alt.aml, is also used to produce the same coverage without any islands.

USAGE: The input coverage used by this program should reside in the Arc/Info workspace where the program is started, and the output should be placed in an Arc/Info workspace. Execute this AML from the Arc/Info prompt:

```
Arc:  feat_for <feature coverage> <forest coverage> <output forest>
<buffer distance> <LINE | POLY>
```

where `<feature coverage>` is the name of the feature coverage to be buffered and cut the input forest coverage, `<forest coverage>`. The resulting coverage will be the `<output forest>` name. The `<buffer distance>` is the distance in meters to buffer the feature coverage. The `<LINE | POLY>` must be entered as the type of feature coverage.

SAMPLE OUTPUT: N/A

PROGRAM FLOW: The program buffers the feature coverage. The input forest coverage is broken out into 1 to 3 polygon coverages depending on how many layers of islands there are. Then, these coverages are clipped(cut) with the buffered feature coverage. The resulting forest coverage is reassembled and extra polygons created by previous steps are removed. The forest polygons are attributed with a ‘22’ using the ‘WTR’ item. The rest are attributed with a ‘1’. The polygons having an area less than 250 are removed.

mpczech

```
&args feat forest forest2 dist type

/* feat_for.aml
/* ------------------------------------------------------------
/*
/*     Program:  feat_for.aml
/* Purpose:  AML to increase the area of the feature input, and
/* erase that area out of the forest coverage.  Assumes that you
/* have already generalized both the feature and forest coverages
/* (so that the feature in the final database will accurately follow
/*      the cuts through the forests.) Buffering the features is
       controled
/* by the distance entered as an arguement.  Examples of feature:
/* roads, hydro, lakes, urban, buildings.  Forest slivers with
/*      an area less than 10 are deleted.
/*
/* ------------------------------------------------------------
/*     Called By:  none
/*     Calls Made:  None
/* ------------------------------------------------------------
/*     Arguments:  feat forest newforest dist type
/*     Globals: none
/* ------------------------------------------------------------
/* Inputs:  arguments called above are the feature to erase from
/*      the forest, original input forest coverage, the name of the new
/*      final output coverage, and the distance to buffer.
/*
/* Outputs:  cover name called in the arguments.  The value '22'
/*      indicates a forest polygon.
/* ---------------------------------------------------------
/*     History: Initial writing only - December 1996 - Tobi Steinberg
/*     Terrain Simulations - Grafenwoehr, Germany
/*
/*      Modifications & bulk of work beginning on 30 January 1997 by
/*             Michael Paul Czechanski.
/*   Incoporated to the changes:  erase area around streams, lakes, urban
/*   areas, and buildings out of the forest also: delete forest slivers.
/*
/*         Last Updated:  23 August 1997
/* ------------------------------------------------------------
&severity &error &routine bail

&echo &on
display 0
&if [null %feat%] OR [null %forest%] OR [null %forest2%] OR ~
[null %dist%] &then &do
   &ty Usage: FEAT_FOR <feat_coverage> <forest_coverage>
       <output_forest> ~
<buffer_distance> {LINE | POLY}
   &return
&end

/*  CHECK TYPE
&if [null %type%] &then
    &s type = 'line'
&else &do
    &s type [locase %type%]
    &if %type% <> 'line' AND %type% <> 'poly' &then &do
        &type  \\Wrong type entered.
        &return
    &end  /* do
&end  /*  do

/* checking to see that both the coverages exist and that both
/* have the correct topology
&if ^ [exists %feat% -line] AND ^ [exists %feat% -poly] &then
       &do
   &ty %feat% does not exist with arc or polygon topology in
       this workspace.
   lc
   &return
&end


&if ^ [exists %forest% -polygon] &then &do
   &ty %forest% does not exist with polygon topology in this
       workspace.
   lc
   &return
&end

&if [exists %forest2% -cover] &then &do
   &ty %forest2% already exists.
   &return
&end


/* checking for coverage & file names which will be needed by
       this AML
&do cov &list xxrd xxbuf xxfor xxforold xxloop xxworking
       xxforest ~
    xxisland xxisland2 xxisl_for xxisl_for2 tmpforest tmpfor
   &if [exists %cov% -cover] &then kill %cov% all
&end

copy %feat% xxworking
&s feat xxworking
clean xxworking

/*  copy %forest% %forest2%
/*  &s forest %forest2%


&if %type% = line &then
  &do
```

```
/*************************************************************
/*     Used with jts
/*
/*    &do file &list xbadpts.txt
/*       &if [exists %file% -file] &then &s junk [delete %file%
       -file]
/*    &end

/*    /* prompting for which item in the feat attribute table
       represents the width
/*    items %feat%.pat
/*    &s att [response 'Which of these attributes represents
       the width? [wtw]' wtw]

/*************************************************************
    /* unsplitting feat network to get rid of any nodes that
       aren't needed
    ae
       ec %feat%
       ef arc
       sel all
       unsplit

/*                * FOR  JTS  *
/* adding an item (which will be removed later) which will
       represent the
/* width of the feat plus %dist% in meters - the width of the
       final buffer
/*       additem xxbuffer 10 10 i
/*       sel all
/*       calc xxbuffer = %dist%

       save
    quit

    /* xxbuf = buffered feat
    buffer %feat% xxbuf # # %dist% # line flat
  &end
&else
    buffer %feat% xxbuf # # %dist% # poly


/* 3 coverages are required for moveislands
/* xxfor = forest w/out island polys
/* xxisland = island polys
/* xxisl_for = polys in islands that are forest

copy %forest% xxisl_for
copy %forest% xxisland

/*  island and forest island coverages are made "empty"
ae
  ec xxisland
  ef arc
  sel all
  &if [show number selected] > 0 &then
    &do
      delete
      ed .1
      nodesnap closest .1
      build
      save

      ec xxisl_for
      ef arc
      sel all
      delete
      ed .1
      nodesnap closest .1
      build
      save
    &end
  &else
    &do
      &type \The feat coverage is not valid.
      &call exit
    &end
quit

tolerance %forest% nodesnap .1
tolerance %forest% edit .1
copy %forest% xxfor

set coverage names equivelant to moveislands routine
&s for xxfor
&s grass xxisland
&s for2 xxisl_for
&call moveislands
/*  set coverage names back to before the moveislands routine
&s xxfor for
&s xxisland grass
&s xxisl_for for2


/* assign boolean variables if data or no data

ae
  ec xxisland
  ef poly
  sel all
  &if [show number selected] > 0 &then
      &sv islandq = .true.
  &else
```

```
      &sv islandq = .false.

  ec xxisl_for
  ef poly
  sel all
  &if [show number selected] > 0 &then
      &sv islqfor = .true.
  &else
      &sv islqfor = .false.
quit

/* xxforest = buffered feat combined w/forest

union xxfor xxbuf xxforest
/*  dropitem xxforest.pat xxforest.pat xxfor#
dropitem xxforest.pat xxforest.pat xxbuf#
dropitem xxforest.pat xxforest.pat xxbuf-id


/* xxisland2 = buffered feat combined w/islands
/* if data exists

&if %islandq% &then
  &do
    union xxisland xxbuf xxisland2
    dropitem xxisland2.pat xxisland2.pat xxbuf#
    dropitem xxisland2.pat xxisland2.pat xxbuf-id
  &end


  /* xxisl_for2 = buffered feat combined w/forest islands
  /* if data exists

  &if %islqfor% &then
    &do
      union xxisl_for xxbuf xxisl_for2
      dropitem xxisl_for2.pat xxisl_for2.pat xxbuf#
      dropitem xxisl_for2.pat xxisl_for2.pat xxbuf-id
    &end


/*  delete input added to forest from buffer

ae
  ec xxforest
  ef poly
  /* select input polygons outside forest polygons
  sel xxfor# = 1
  &if [show number selected] > 0 &then
      delete
  sel inside = 100    /* selects input polygons in forest
       polygons
  &if [show number selected] > 0 &then
      delete
/*  sel npnts = 0
/*  &if [show number selected] > 0 &then
/*      delete
  sel all
  calc wtr = 22       /* attribute forest polygons
  build
  sel wtr = 22
  nsel
  &if  [show number selected] > 0 &then
      calc wtr = 1      /* attribute island polygons created by
       buffer
  build
  save

  /*  delete the input added to islands from buffer
  /*  if data exists

  &if %islandq% &then
    &do
      ec xxisland2
      ef poly
      sel xxisland# = 1 OR xxisland# = 0
      &if [show number selected] > 0 &then
          delete
      sel inside = 100
      &if [show number selected] > 0 &then
          delete
      sel all
      calc wtr = 1        /* attribute island polygon for
       deletion
      build
      save

      /*  delete the input added to forest islands from buffer
      /*  if data exists

      &if %islqfor% &then
        &do
          ec xxisl_for2
          ef poly
          sel xxisl_for# = 1 OR xxisl_for# = 0
          &if [show number selected] > 0 &then
              delete
          sel inside = 100
          &if [show number selected] > 0 &then
              delete
          sel all
          calc wtr = 22       /* attribute forest island
       polygon
          build
```

```
          save
        &end
    &end  /* do islandq
quit


/* extra items created from prevous steps are removed

dropitem xxforest.pat xxforest.pat inside
dropitem xxforest.pat xxforest.pat xxfor-id

dropitem xxforest.pat xxforest.pat xxfor#   /*  DEBUGGGGGG

&if %islandq% &then
  &do
    dropitem xxisland2.pat xxisland2.pat xxisland#
    dropitem xxisland2.pat xxisland2.pat xxisland-id
    dropitem xxisland2.pat xxisland2.pat inside


    &if %islqfor% &then
      &do
        dropitem xxisl_for2.pat xxisl_for2.pat xxisl_for#
        dropitem xxisl_for2.pat xxisl_for2.pat xxisl_for-id
        dropitem xxisl_for2.pat xxisl_for2.pat inside
      &end
  &end  /* do islandq

&else     /* no islands exist, program is finished
  &do
    copy xxforest %forest2%
    &type \\No islands exist, output coverage is %forest2%.
    &call exit
  &end


/* tmpforest = forest combined with islands

update xxforest xxisland2 tmpforest
build tmpforest

/*  remove island polys from forest

ae
  ec tmpforest
  ef poly
  sel wtr = 1
  delete
  build
  save
quit
```

```
&if %islqfor% &then
  &do
    update tmpforest xxisl_for2 tmpfor
    dissolve tmpfor %forest2% #all
  &end
&else
    dissolve tmpforest %forest2% #all
build %forest2%


/* Remove slivers from the forest coverage
ae
    ec %forest2%
    ef poly
    sel area < 250
    &if [show number selected] > 0 &then
      &do
        delete
        build
        save
      &end  /* do
quit


&call exit



/*************************************************************
/* exit routine
/*************************************************************
&routine exit
/* cleaning up coverage names
/*  &do cov &list xxrd xxbuf xxfor xxforold xxloop xxworking
       xxforest ~
/*    xxisland xxisland2 xxisl_for xxisl_for2 tmpforest
/*    &if [exists %cov% -cover] &then kill %cov% all
/*  &end

display 9999
&echo &off
&return &end

/*************************************************************
/* bail routine
/*************************************************************
&routine bail
&severity &error &ignore
&severity &warning &ignore
&type An error has occured in [show amlname]
&type Bailing out of [show amlname]
&return; &return &error
```

```
/*************************************************************
/* moveislands routine  WRITTEN BY ERIK PATTON
/*************************************************************

&routine moveislands

/* moveislands.aml
/* Removes islands from forest coverage and puts them into the
       grassland
/* and optionally forest2 coverages when building UCCATS/JTS
       terrain.
/*
/* It assumes that the forest, grassland, and forest2 coverages
       already exist.

/*  &args for grass for2
&severity &error &ignore

/* Check for valid input.
&if [null %for2%] &then
   &return Usage: MOVEISLANDS <forest_cov> <grassland_cov>
       <forest2_cov>

&do test &list %for% %grass% %for2%
   &if ^ [exists %test% -cover] &then
      &do
         &ty Coverage %test% doesn't exist.
         &return
      &end
&end

/*&s qes = [response 'Did you remember to BUILD? [n]' n]
/*&if %qes% = n &then
/*   &do
/*      &ty Program terminating.  Please use BUILD before using
       MOVEISLANDS.
/*      &return
/*   &end
/*&if %qes% = y &then

/* Update topology
build %for% poly
build %grass% poly
build %for2% poly

   &do

/* Update line topology if necessary because selection is done
       on aat.
&if ^ [exists %for% -line] &then
   build %for% line

&if ^ [exists %grass% -line] &then
   build %grass% line

display 0
ae

/* Delete any arcs that might be in grassland or forest2
&do delete &list %grass% %for2%
  ec %delete%
  ef arc
  sel all
  delete
&end

/* Select the islands in forest, put to grass and delete
ec %for%
ef arc
sel lpoly# ne 1 and rpoly# ne 1
put %grass%
y
delete

quit yes

/* Re-build to update arc id values to do re-select
clean %grass% # # 1.0 poly    /*******************************
build %grass% line

/* Select and put interior islands to forest2
display 0
ae
ec %grass%
ef arc
sel lpoly# ne 1 and rpoly# ne 1
put %for2%
y
delete
save
quit yes

/* Update topology
build %for% poly
build %grass% poly
build %for2% poly

&end   /* From response question above.
&severity &error &routine bail
&return
```

========== feat_alt.aml ==========

```
&args forest output dist

/****
/*  THIS PROGRAM HAS CRAPPY DOCUMENTATION, HAVE A NICE DAY
/****

&echo &on
display 0

&if [null %forest%] OR [null %output%] OR [null %dist%] &then
      &do
    &ty Usage: FEAT_ALT <forest_coverage> <output_forest>
      <buffer_distance>
    &call exit
&end

&if NOT [exists %forest% -poly]  &then  &do
    &type Forest polygon coverage does not exist.
    &call exit
&end

/*  &if [exists %output% -cover] &then &do
/*   &ty %forest2% already exists.
/*   &call exit
/*  &end

copy %forest% %output%
&call moveislands

&do feat &list paved lakes hydro rail building urban
    &if [exists %feat% -cover] &then &do
        &if [exists %feat% -line] &then
            &s type = line
        &else &if [exists %feat% -poly] &then
            &s type = poly
        &else &do
            &ty A coverage has wrong topology.
            &call exit
        &end

        &call thegoods
        kill %output% all
        kill xxworking all
        kill xxbuf all
        copy xxforest %output%
        kill xxforest all
    &end
&end  /* do

kill xxisland all
kill xxisl_for all

&ty PROGRAM IS COMPLETE\\
&ty THE PROGRAM TOOK THE %forest% COVERAGE AND CLIPPED A
      %dist%METER BUFFER
&ty OF THE FEATURE COVERAGES TO CREATE A FINAL COVERAGE CALLED
      *%output%*\\

&call exit


/*************************************************************
/*  THEGOODS  ROUTINE
/*************************************************************
&routine thegoods


copy %feat% xxworking
&s feat xxworking
clean xxworking

&if %type% = line &then
  &do
    /* unsplitting feat network to get rid of any nodes that
       aren't needed
    ae
       ec %feat%
       ef arc
       sel all
       unsplit
       save
    quit

    /* xxbuf = buffered feat
    buffer %feat% xxbuf # # %dist% # line flat
  &end
&else
    buffer %feat% xxbuf # # %dist% # poly


/* xxforest = buffered feat combined w/forest

union %output% xxbuf xxforest
/*  dropitem xxforest.pat xxforest.pat %output%#
dropitem xxforest.pat xxforest.pat xxbuf#
dropitem xxforest.pat xxforest.pat xxbuf-id

/*  delete input added to forest from buffer

ae
  ec xxforest
  ef poly
```

```
  /* select input polygons outside forest polygons
  sel %output%# = 1
  &if [show number selected] > 0 &then
      delete
  sel inside = 100    /* selects input polygons in forest
       polygons
  &if [show number selected] > 0 &then
      delete
/*  sel npnts = 0
/*  &if [show number selected] > 0 &then
/*      delete
  sel all
  calc wtr = 22       /* attribute forest polygons
  build
  sel wtr = 22
  nsel
  &if  [show number selected] > 0 &then
      calc wtr = 1      /* attribute island polygons created by
       buffer
  build
  save
quit

/* extra items created from prevous steps are removed

dropitem xxforest.pat xxforest.pat inside
dropitem xxforest.pat xxforest.pat %output%-id
dropitem xxforest.pat xxforest.pat %output%#   /*  DEBUGGGGGG


&return

/*************************************************************
/*  MOVEISLANDS   ROUTINE
/*************************************************************

&routine moveislands

/* moveislands.aml
/* Removes islands from forest coverage and puts them into the
       grassland
/* and optionally forest2 coverages when building UCCATS/JTS
       terrain.
/*
/* It assumes that the forest, grassland, and forest2 coverages
       already exist.

/*  &args for grass for2
&severity &error &ignore

/* Addition  *******************************************
/* 3 coverages are required for moveislands
/* %output% = forest w/out island polys
/* xxisland = island polys
/* xxisl_for = polys in islands that are forest

copy %forest% xxisl_for
copy %forest% xxisland

/*  island and forest island coverages are made "empty"

ae
  ec xxisland
  ef arc
  sel all
  &if [show number selected] > 0 &then
    &do
      delete
      ed .1
      nodesnap closest .1
      build
      save

      ec xxisl_for
      ef arc
      sel all
      delete
      ed .1
      nodesnap closest .1
      build
      save
    &end
  &else
    &do
      &type \The feat coverage is not valid.
      &call exit
    &end
quit

tolerance %forest% nodesnap .1
tolerance %forest% edit .1
copy %forest% %output%

set coverage names equivelant to moveislands routine
&s for %output%
&s grass xxisland
&s for2 xxisl_for


/* End Addition  ****************************************


/* Check for valid input.
```

```
&if [null %for2%] &then
   &return Usage: MOVEISLANDS <forest_cov> <grassland_cov>
       <forest2_cov>

&do test &list %for% %grass% %for2%
   &if ^ [exists %test% -cover] &then
      &do
         &ty Coverage %test% doesn't exist.
         &return
      &end
&end

/*&s qes = [response 'Did you remember to BUILD? [n]' n]
/*&if %qes% = n &then
/*   &do
/*      &ty Program terminating.  Please use BUILD before using
       MOVEISLANDS.
/*      &return
/*   &end
/*&if %qes% = y &then

/* Update topology
build %for% poly
build %grass% poly
build %for2% poly

   &do

/* Update line topology if necessary because selection is done
       on aat.
&if ^ [exists %for% -line] &then
   build %for% line

&if ^ [exists %grass% -line] &then
   build %grass% line

display 0
ae

/* Delete any arcs that might be in grassland or forest2
&do delete &list %grass% %for2%
  ec %delete%
  ef arc
  sel all
  delete
&end

/* Select the islands in forest, put to grass and delete
ec %for%
ef arc
sel lpoly# ne 1 and rpoly# ne 1
put %grass%
y
delete

quit yes

/* Re-build to update arc id values to do re-select
clean %grass% # # 1.0 poly    /*******************************
build %grass% line

/* Select and put interior islands to forest2
display 0
ae
ec %grass%
ef arc
sel lpoly# ne 1 and rpoly# ne 1
put %for2%
y
delete
save
quit yes

/* Update topology
build %for% poly
build %grass% poly
build %for2% poly

&end   /* From response question above.
&severity &error &routine bail
&return

/*************************************************************
/* exit routine
/*************************************************************
&routine exit
/* cleaning up coverage names
/*  &do cov &list xxrd xxbuf xxfor xxforold xxloop xxworking
       xxforest ~
/*    xxisland xxisland2 xxisl_for xxisl_for2 tmpforest
/*    &if [exists %cov% -cover] &then kill %cov% all
/*  &end

display 9999
&echo &off
&return &end

/*************************************************************
/* bail routine
/*************************************************************
&routine bail
&severity &error &ignore
&severity &warning &ignore
&type An error has occured in [show amlname]
```

```
&type Bailing out of [show amlname]
&return; &return &error
```

2.2.2 grafplot.aml

PURPOSE: This program generates plot files for paper 1:50,000 maps for the Grafenwoehr training area(GTA) and all the features included in the final S1000 / SimNet database, including title and legend information.

BACKGROUND: This program is specific to the GTA. To maintain the area extents of past SimNet maps and map aesthetics, the GTA is placed in the center of the map composition and minimal area is clipped on both edges. Coverages must be clipped prior to running this program. The legend text files are specific to the GTA. This program does not create adjacent map sheets.

HISTORY: This program was written by Michael Paul Czechanski in August 1997.

USAGE: This program is run in a directory at the same level of two other directories. The program doesn't have to be executed in an Arc/Info workspace. Execute this program at the Arc/Info prompt:

```
Arc: grafplot
```

This program requires a strict directory structure. There are three directories on an equal level. One directory doesn't have to be an Arc/Info workspace and this is where the program is initiated. This directory can be any name and here resides the text and graphics files for the legend. The two remaining directories mush be Arc/Info workspaces and have specific names. One must be called **COVERAGES** which contain all the coverages except the elevation coverages. The other must be called **DTED** and contain the elevation coverage and a generalized coverage for the legend elevation graphic. The name for this coverage must be **elev_sm**.

SAMPLE OUTPUT: This program ultimately can create a graphics file, plot file, and send to either of the TerraSim 24" plotters. The program prompts the user the option to continue at each of the above steps.

PROGRAM FLOW: The code for this program follows, along with the contents of the required accompanying text files. After the program is initiated, the user is prompted for lower left geographic coordinates and a map composition name. The entered coordinates are displayed and the user is prompted if the verification is correct. If it isn't, the user can re-enter them. Next, the user must choose one of the coverages in each of the pop-up menus. These are also displayed for and prompted if correct. The program creates the map composition, and then prompts the user to continue and create a graphics file. If no, the user is left with a map composition. The user can continue to a graphics file, plot file, and to a plotter.

NAMING CONV: Here is the naming convention for coverages, directories, and files. Names in **bold** are mandatory. The others are the user's choice.

Coverages	Elevation	Working
clip	elev_clip	**atc.gra**
forest_clp	**elev_sm**	**info_box.gra**
urban		**lft_info.txt**
lakes_clp		**rt_info.txt**
hydro_clp		**prj_info.txt**
building		**class1.leg**
roads_clp		**line.leg**
gta_clip		**scal.gra**
box_gta		
utmgrid		

The ***info.txt** files in the WORKING directory are specific for the GTA.

In the COVERAGES directory there are four coverages that are GTA specific.

clip is the extent of the internal map elements

gta_clip is the GTA boundary that is clipped with the **clip** coverage

box_gta is the coverage used to shade the 'no data' area. It is a simple polygon coverage with the 'NODATA' attributed with the carto.shd symbol and the the GTA with '0'

utmgrid was created with the **utmgrid.aml**

elev_sm in the DTED directory is a polygon coverage with each polygon representing a third of the elevation. The attribute **elev_code** determines the shading for the map composition.

mpczech

```
/* grafplot.aml
/* ------------------------------------------------------------
/*      Program: grafplot.aml
/*      Purpose: This program generates plot files for paper
/*           maps of all the features included in the final
/*           S1000 / SimNet database, including title and
/*           legend information.
/* ------------------------------------------------------------
/*      Called By:  none
/*      Calls Made:  none
/* ------------------------------------------------------------
/*      Arguments:  none
/*      Globals:  none
/* ------------------------------------------------------------
/*      Inputs: Coverages of each of the individual features
/*     to be included in the final maps.
/*     Text files PRJ_INFO.TXT, LFT_INFO.TXT, RT_INFO.TXT
/*      Outputs: plot files for the paper maps
/* ------------------------------------------------------------
/*      History: written by Michael Paul Czechanski
/*              czech@email.grafenwoehr.army.mil
/*      Terrain Simulations - 7th ATC - Grafenwoehr, Germany
/* ------------------------------------------------------------
&ty [date -VFULL]

/**    ASSUMED
/**    THE NUMBER OF COORDINATE LABELS
/**    THE AREA IS 18 MINUTES EAST BY 15 MINUTES NORTH
/**    COVERAGES ARE CLIPPED PROPERLY TO THE GTA CLIP COVERAGE
/**    TEXT INFORMATION MUST EXIST: PRJ_INFO.TXT, LFT_INFO.TXT,
       RT_INFO.TXT

/*  &echo &on

/*  COLLECT COORDINATES AND ASSIGN VARIABLES  **************

&terminal 9999
display 9999

&s quit .FALSE.
&do &until %quit%
    &ty \\
    &s lngeodeg [RESPONSE 'Enter the lower left Latitude(y)
       Degree without the ~
Minutes']  /* lower northing geographic degree - lngeodeg
    &s lngeomin [RESPONSE 'Enter the Minutes']
    &s legeodeg [RESPONSE 'Enter the lower left Longitude(x)
       Degree without the ~
Minutes']  /* left easting geographic degree - legeodeg
    &s legeomin [RESPONSE 'Enter the Minutes']
    &s east %legeodeg% %legeomin% E
    &s north %lngeodeg% %lngeomin% N
    &ty \\%north%
    &ty %east%
    &ty \
    &s quit [Query 'Is this the correct northing and easting']
&end  /* until

&ty \'What would you like the map composition name to be?'
&s comp [Response 'Enter name with no extentions']


&ty \'Please choose the coverages from the popup windows.'\

&s quit .FALSE.
&do &until %quit%
    &s extentmap = [getcover ../coverages -polygon 'Please ~
select your map extent coverage: ']

/*  COVERAGES  *****
    &s forest = [getcover ../coverages -polygon 'Please select
       your forest ~
coverage:']
    &s treeline = [getcover ../coverages -line 'Please select
       your treeline ~
coverage:']
    &s urban = [getcover ../coverages -polygon 'Please select
       your urban ~
coverage:']
    &s lakes = [getcover ../coverages -polygon 'Please select
       your open water ~
coverage:']
    &s hydro = [getcover ../coverages -line 'Please select your
       hydro ~
coverage:']
    &s building = [getcover ../coverages -polygon 'Please
       select your ~
buildings coverage: ']
    &s road = [getcover ../coverages -line 'Please select your
       road coverage:']
    &s railroad = [getcover ../coverages -line 'Please select
       your railroad ~
coverages:']
    &s cliparea = [getcover ../coverages -polygon 'Please
       select your map area ~
(gta_clip_clp) coverage:']
    &s shadearea = [getcover ../coverages -polygon 'Please
       select your graf no ~
data (box_gta) coverage:']
    &s elevation = [getcover ../dted -line 'Please select your
       contour ~
coverage:']
    &s utmgrid = [getcover ../coverages -line 'Please select
       your utm grid ~
```

```
coverage:']
    &ty \map_extent->[entryname %extentmap%]
    &ty forest----->[entryname %forest%]
    &ty treeline--->[entryname %treeline%]
    &ty urban------>[entryname %urban%]
    &ty openwater-->[entryname %lakes%]
    &ty hydro------>[entryname %hydro%]
    &ty buildings-->[entryname %building%]
    &ty road------->[entryname %road%]
    &ty railroad--->[entryname %railroad%]
    &ty map_area--->[entryname %cliparea%]
    &ty no_data---->[entryname %shadearea%]
    &ty contour---->[entryname %elevation%]
    &ty utm_grid--->[entryname %utmgrid%]\
    &s quit [Query 'Are these the coverages you want']
&end  /* until

&s tempmin %legeomin% + 18
&if %tempmin% >= 60 &then &do
    &if %tempmin% = 60 &then
        &s regeomin 00
    &else
        &s regeomin %tempmin% - 60   /* right easting
       geographic minutes
    &s regeodeg %legeodeg% + 1
&end /* do
&else &do
    &s regeomin %tempmin%
    &s regeodeg %legeodeg%
&end /* do

&s tempmin %lngeomin% + 15
&if %tempmin% >= 60 &then &do
    &if %tempmin% = 60 &then
        &s ungeomin 00
    &else
        &s ungeomin %tempmin% - 60  /* upper northing
       geographic minutes
    &s ungeodeg %lngeodeg + 1
&end /* do
&else &do
    &s ungeomin %tempmin%
    &s ungeodeg %lngeodeg%
&end /* do

/*  CHANGE COORDINATES TO DECIMAL DEGREES

&s ledd [calc %legeomin% / 60 + %legeodeg%]  /* left easting
       decimal degrees
&s lndd [calc %lngeomin% / 60 + %lngeodeg%]  /* lower northing
       decimal degrees
&s redd [calc %regeomin% / 60 + %regeodeg%]  /* right easting
       decimal degrees
&s undd [calc %ungeomin% / 60 + %ungeodeg%]  /* upper northing
       decimal degrees

/*  CHANGE COORDINATES TO UTM
ap  /* arc plot is invoked early to maximize the use of the
       geo2utm routine
textset font
textsymbol 10
textjustification cc
textsize 1
move 4 2.75
text COCA-COLA
&s num 1
&do &while %num% < 20
    &s num %num% + 1
&end  /* do
clear
textsize .3
move 4 2.75
text 'STICK  AROUND'   /* this is just for fun


&s e %ledd%
&s n %lndd%
&call geo2utm
&s lowleftx %coord1%
&s lowlefty %coord2%


&s n %undd%
&call geo2utm
&s upleftx %coord1%
&s uplefty %coord2%

&s e %redd%
&call geo2utm
&s uprightx %coord1%
&s uprighty %coord2%

&s n %lndd%
&call geo2utm
&s lowrightx %coord1%
&s lowrighty %coord2%


/*  IF CHECKS ARE COMMENTED OUT DUE TO NON-INTEGERS WITH THE
       MOD
/*  ROUNDS THE lowleftx UP TO THE NEAREST 1000TH
/*  &if [MOD %lowleftx% 1000] <> 0 &then
```

```
    &s newllx [calc 1000 + [calc 1000 * [TRUNCATE [calc
      %lowleftx% / 1000]]]]

/*  ROUNDS THE lowrightx DOWN TO THE NEAREST 1000TH
/*  &if [MOD %lowrightx% 1000] <> 0 &then
    &s newlrx [calc 1000 * [TRUNCATE [calc %lowrightx% /
      1000]]]

/*  ROUNDS THE lowlefty UP TO THE NEAREST 1000TH
/*  &if [MOD %lowlefty% 1000] <> 0 &then
    &s newlly [calc 1000 + [calc 1000 * [TRUNCATE [calc
      %lowlefty% / 1000]]]]

/*  ROUNDS THE upleftx UP TO THE NEAREST 1000TH
/*  &if [MOD %upleftx% 1000] <> 0 &then
    &s newulx [calc 1000 + [calc 1000 * [TRUNCATE [calc
      %upleftx% / 1000]]]]

/*  ROUNDS THE uprighty DOWN TO THE NEAREST 1000TH
/*  &if [MOD %uprighty% 1000] <> 0 &then
    &s newury [calc 1000 * [TRUNCATE [calc %uprighty% / 1000]]]

/*  ROUNDS THE lowrighty UP TO THE NEAREST 1000TH
/*  &if [MOD %lowrighty% 1000] <> 0 &then
    &s newlry [calc 1000 + [calc 1000 * [TRUNCATE [calc
      %lowrighty% / 1000]]]]

/*  ROUNDS THE uplefty DOWN TO THE NEAREST 1000TH
/*  &if [MOD %uplefty% 1000] <> 0 &then
    &s newuly [calc 1000 * [TRUNCATE [calc %uplefty% / 1000]]]

/*  ROUNDS THE uprightx DOWN TO THE NEAREST 1000TH
/*  &if [MOD %uprightx% 1000] <> 0 &then
    &s newurx [calc 1000 * [TRUNCATE [calc %uprightx% / 1000]]]

/*  DIFFERENCES IN COORDINATES AND DIVIDED BY 1000  ********

&s ltdif [calc [calc %newuly% - %newlly%] / 1000]
&s rtdif [calc [calc %newury% - %newlry%] / 1000]
&s botdif [calc [calc %newlrx% - %newllx%] / 1000]
&s topdif [calc [calc %newurx% - %newulx%] / 1000]

/*  CALCULATE THE RISE OVER RUN FOR THE UTM EDGES  ************

&s rise [calc %lowleftx% - %upleftx%]
&s run [calc %uplefty% - %lowlefty%]
&s riserun_l [calc %rise% / %run%]       /* left

&s rise [calc %lowrightx% - %uprightx%]
&s run [calc %uprighty% - %lowrighty%]
&s riserun_r [calc %rise% / %run%]       /* right
&s rise [calc %lowrighty% - %lowlefty%]
&s run [calc %lowrightx% - %lowleftx%]
&s riserun_b [calc %rise% / %run%]       /* bottom

&s rise [calc %uprighty% - %uplefty%]
&s run [calc %uprightx% - %upleftx%]
&s riserun_t [calc %rise% / %run%]       /* top


/*  START MAP COMPOSITION  ********************************

clear /*  stick around

mape %extentmap%
pagesize 24 30
mapunits meters
mapscale 50000
map %comp%.map
box .5 .5 23.5 29.5
line .5 5.5 23.5 5.5
maplimits .5 5.5 23.5 29.5
mapposition cen cen
units map

arcs %extentmap%
arcs %utmgrid%


/* ##########################################################
/*  UTM EDGE LABELS  ************************************
/*  LEFT LABELS  *****

&s xdist [calc %lowleftx% - 293]
&s ydist %newlly%
&s num 1

textset font.txt
textsymbol 10
textjustification cc

&s textgross = [show textsize]


/*  ADDITIONAL TEXT FOR LOWER LEFT
mbegin

textsize .16
move %xdist% %ydist%
&s tempy [calc %ydist% / 1000]
&s label [MOD %tempy% 100]
&if %label% < 10 &then
    &call less
```

```
&else
    text %label%

&s tempx %xdist% - 150
&s bigy [TRUNCATE [calc %ydist% / 100000]]
textsize .07
move %tempx% %ydist%
text %bigy%

move [calc %xdist% + 225] %ydist%
text 000m

textsize .16
move [calc %xdist% + 400] %ydist%
text N
mend
mrotate 92
mmove .1 0 0 .15

&s num %num% + 1
&s xdist %xdist% - 32
&s ydist %ydist% + 1000


/*  CALCULATE THE INTERVAL TO ACCOUNT FOR THE SLANT
&s bottom_tmp1 [calc %newlly% - %lowlefty%]
&s bottom_tmp2 [calc %bottom_tmp1% * %riserun_l%]
&s bottom [calc %lowleftx% - %bottom_tmp2%]
&s top_tmp1 [calc %newuly% - %lowlefty%]
&s top_tmp2 [calc %top_tmp1% * %riserun_l%]
&s top [calc %lowleftx% - %top_tmp2%]
&s dif [calc %bottom% - %top%]
&s interval [calc %dif% / %ltdif%]

mbegin

&s times %ltdif% + 2

&do &while %num% < %times%
    move %xdist% %ydist%
    &s tempy [calc %ydist% / 1000]
    &s label [MOD %tempy% 100]
    &if %label% < 10 &then
        &call less
    &else
        text %label%
    &if [MOD %ydist% 10000] = 0 &then
      &do
        &s tempx %xdist% - 200
        &s bigy [TRUNCATE [calc %ydist% / 100000]]
        textsize .07
        move %tempx% %ydist%
        text %bigy%
        textsize .16
      &end  /* do
    &s num %num% + 1
    &s xdist %xdist% - %interval%
    &s ydist %ydist% + 1000
&end

mend

/*  RIGHT LABELS  *****

/*  CALCULATE THE INTERVAL TO ACCOUNT FOR THE SLANT
&s bottom_tmp1 [calc %newlry% - %lowrighty%]
&s bottom_tmp2 [calc %bottom_tmp1% * %riserun_r%]
&s bottom [calc %lowrightx% - %bottom_tmp2%]
&s top_tmp1 [calc %newury% - %lowrighty%]
&s top_tmp2 [calc %top_tmp1% * %riserun_r%]
&s top [calc %lowrightx% - %top_tmp2%]
&s dif [calc %bottom% - %top%]
&s interval [calc %dif% / %rtdif%]


mbegin

&s num 1
&s xdist [calc %lowrightx% + 275]
&s ydist %newlry%
&s times %rtdif% + 2

&do &while %num% < %times%

    &if [MOD %ydist% 10000] = 0 &then
      &do
        &s bigy [TRUNCATE [calc %ydist% / 100000]]
        textsize .07
        move [calc %xdist% - 100] %ydist%
        text %bigy%
        textsize .16
        &s tempx [calc %xdist% + 100]
      &end  /* do
    &else
        &s tempx %xdist%

    move %tempx% %ydist%
    &s tempy [calc %ydist% / 1000]
    &s label [MOD %tempy% 100]
    &if %label% < 10 &then
        &call less
    &else
        text %label%
```

```
    &s num %num% + 1
    &s xdist %xdist% - %interval%
    &s ydist %ydist% + 1000
&end

mend


/*  BOTTOM LABELS  *****

&s num 1
&s xdist %newllx%
&s ydist [calc %lowlefty% - 175]

/*  ADDITIONAL TEXT FOR LOWER LEFT
mbegin

move %xdist% %ydist%
&s newx [calc %xdist% / 1000]
&s label [MOD %newx% 100]
&if %label% < 10 &then
    &call less
&else
    text %label%

&s bigx [TRUNCATE [calc %xdist% / 100000]]
textsize .07
move [calc %xdist% - 125] %ydist%
text %bigx%

move [calc %xdist% + 225] %ydist%
text 000m

textsize .16
move [calc %xdist% + 400] %ydist%
text E
mend

mmove 0 .15 0 0

/*  CALCULATE THE INTERVAL TO ACCOUNT FOR THE SLANT
&s bottom_tmp1 [calc %newllx% - %lowleftx%]
&s bottom_tmp2 [calc %bottom_tmp1% * %riserun_b%]
&s bottom [calc %lowlefty% + %bottom_tmp2%]
&s top_tmp1 [calc %newlrx% - %lowleftx%]
&s top_tmp2 [calc %top_tmp1% * %riserun_b%]
&s top [calc %lowlefty% + %top_tmp2%]
&s dif [calc %top% - %bottom%]
&s interval [calc %dif% / %botdif%]

mbegin
&s num %num% + 1
&s xdist %xdist% + 1000
&s ydist %ydist% + %interval%
&s upper .FALSE.  /*  explained below

&s times %botdif% + 2
&call edge_long


/*  TOP LABELS  *****
&s num 1
&s xdist %newulx%
&s ydist [calc %uplefty% + 235]
&s upper .TRUE.   /*  boolean for allowing space for the geo
       label later

/*  CALCULATE THE INTERVAL TO ACCOUNT FOR THE SLANT
&s bottom_tmp1 [calc %newulx% - %upleftx%]
&s bottom_tmp2 [calc %bottom_tmp1% * %riserun_t%]
&s bottom [calc %uplefty% + %bottom_tmp2%]
&s top_tmp1 [calc %newurx% - %upleftx%]
&s top_tmp2 [calc %top_tmp1% * %riserun_t%]
&s top [calc %uplefty% + %top_tmp2%]
&s dif [calc %top% - %bottom%]
&s interval [calc %dif% / %topdif%]

&s times %topdif% + 2
&call edge_long
&s upper .FALSE.

mend


/*  CALCULATES THE X DISTANCE FOR UTM LABELS INSIDE THE MAP  **

textsize %textgross%
textmask rectangle .03
&s chk [calc [calc [calc %newlrx% - %newulx%] / 3] / 1000]
&s chkrnd [ROUND %chk%]
&if %chkrnd% > %chk% &then
    &s dist [calc 500 + [calc 1000 *[TRUNCATE %chk%]]]
&else
    &s dist [calc [calc 1000 * [TRUNCATE %chk%]] - 500]

/* LEFT INSIDE UTM LABELS
&s xdist [calc %newulx% + %dist%]
&s ydist [calc %newlly% + 1000]
&s times %ltdif% + 1
&call inside_long

/* RIGHT INSIDE UTM LABELS
&s xdist [calc %newlrx% - %dist%]
```

```
&s ydist [calc %newlly% + 1000]      /* update to a variable
&s times %rtdif% + 1
&call inside_long


/* CALCULATES THE Y DISTANCE FOR UTM LABELS INSIDE THE MAP
&s chk [calc [calc [calc %newury% - %newlry%] / 3] / 1000]
&s chkrnd [ROUND %chk%]
&if %chkrnd% > %chk% &then
    &s dist [calc 500 + [calc 1000 *[TRUNCATE %chk%]]]
&else
    &s dist [calc [calc 1000 * [TRUNCATE %chk%]] - 500]

/*  UPPER INSIDE LAT UTM LABELS
&s xdist %newulx%
&s ydist [calc %newury% - %dist%]
&s num 1
&s times %topdif% + 2
&call inside_lat

/*  LOWER INSIDE LAT UTM LABELS
&s xdist [calc %newulx% + 1000]
&s ydist [calc %newlly% + %dist%]
&s num 2
&s times %botdif% + 2
&call inside_lat

/*  DRAW THE COVERAGES  ***********************************

shadeset colornames
resel %forest% poly wtr = 22
polygonshades %forest% 67
polygonshades %urban% 25
arcs %urban%
polygonshades %lakes% 50
linesymbol 4
arcs %lakes%
arcs %hydro%
linesymbol 3
arcs %treeline%
linesymbol 85
arcs %railroad%
linesymbol 66
arcs %road%
polygonshades %building% 27
textset font.txt
textsymbol 10
textsize .1
linesymbol 2
textjustification cc
&s tmpelev [entryname %elevation%]
&s item_name %tmpelev%-id
resel %elevation% arc contour <> 400 AND contour <> 500
arcs %elevation%
nsel %elevation% arc
linesymbol 6
resel %elevation% arc length < 1000
arcs %elevation%
nsel %elevation% arc
resel %elevation% arc contour = 400 OR contour = 500
arctext %elevation% %item_name% # line # blank
linesymbol 1
arcs %cliparea%

shadeset plotter
polygonshades %shadearea% symbol
textsymbol 8 /*  try carto 137
/*  textmask rectangle .1
move 702480 5517420
text 'N O    D A T A'
move 702480 5499530
text 'N O    D A T A'

textmask none


/*  GEOGRAPHIC LABELS  ***************************************

&ty \\\\\'GET  SOME  COFFEE.    THIS  WILL TAKE  A  WHILE.'\\
&ty 'Calculating and drawing geographic coordinates.'\


/*  DEGREES AND MINUTES TEXT - LEFT EASTING
mbegin
textset font
textsymbol 1
textsize .12
textstyle typeset
&s mainx [calc %lowleftx% - 100]
&s mainy [calc %lowlefty% - 150]
move %mainx% %mainy%
text %legeodeg%!pat1713
&s minx [calc %mainx% + 290]
move %minx% %mainy%
text %legeomin%!pat1727
mend

mcopy graf.map
mmove .8 0 0 22.2


/*  DERGREES AND MINUTES TEXT - LOWER NORTHING
textsize .12
mbegin
```

```
&s mainx [calc %lowleftx% - 520]
&s mainy %lowlefty%
move %mainx% %mainy%
text %lngeodeg%!pat1713
&s minx [calc %mainx% + 300]
move %minx% %mainy%
text %lngeomin%!pat1727
mend

mcopy graf.map
mmove 0 0 17.7 .65


/*  DERGREES AND MINUTES TEXT - UPPER NORTHING
textsize .12
mbegin
&s mainx [calc %upleftx% - 520]
&s mainy %uplefty%
move %mainx% %mainy%
text %ungeodeg%!pat1713
&s minx [calc %mainx% + 300]
move %minx% %mainy%
text %ungeomin%!pat1727
mend

mcopy graf.map
mmove 0 0 17.65 .65


/*  DEGREES AND MINUTES TEXT - RIGHT EASTING
textsize .12
mbegin
&s mainx [calc %uprightx% - 100]
&s mainy [calc %uprighty% + 150]
move %mainx% %mainy%
text %regeodeg%!pat1713
&s minx [calc %mainx% + 290]
move %minx% %mainy%
text %regeomin%!pat1727
mend

mcopy graf.map
mmove 0 22.17 .85 0


/*  PLACE LABEL AT 5 MINUTE INCREMENTS - EASTING

&s newmin %legeomin% + 2  /* 2-> don't print the 1st one
&do &while %newmin% < %regeomin%
    &if [MOD %newmin% 5] = 0 &then &do
        &s e [calc %newmin% / 60 + %legeodeg%]
        &s n %lndd%  /* lower northing decimal degrees
        &call geo2utm
        &s quest %coord1%
        &s dist 200
        &call placecheck
        &if %bigmove% &then
            &s amt 350  /* amount is the same either way
        &else &if %move% &then
            &s amt 350  /* tied with the above comment
        &else
            &s amt 175

        &s mainx %coord1%
        &s mainy %coord2% - %amt%
        move %mainx% %mainy%
        textsize .12
        text %newmin%
        move [calc %mainx% + 110] [calc %mainy% + 75]
        textsize .1
        text ','
        &s n %undd%
        &call geo2utm
        &s quest %coord1%
        &s dist 200
        &call placecheck
        &if %bigmove% &then
            &s amt 350  /* amount is the same either way
        &else &if %move% &then
            &s amt 350  /* tied with the above comment
        &else
            &s amt 200

        &s mainx %coord1%
        &s mainy %coord2% + %amt%
        move %mainx% %mainy%
        textsize .12
        text %newmin%
        move [calc %mainx% + 110] [calc %mainy% + 75]
        textsize .1
        text ','

        &s newmin %newmin% + 4  /* increments to the next five
      minutes
    &end  /*do
    &s newmin %newmin% + 1
&end  /*while


/*  PLACE LABEL AT 5 MINUTE INCREMENTS - NORTHING
&s newmin %lngeomin% + 2  /* 2-> don't print the 1st one
&do &while %newmin% < %ungeomin%
    &if [MOD %newmin% 5] = 0 &then &do
        &s e %ledd%  /*  left easting decimal degrees
```

```
        &s n [calc %newmin% / 60 + %lngeodeg%]
        &call geo2utm
        &s quest %coord2%
        &s dist 200
        &call placecheck
        &if %bigmove% &then
            &s amt 700
        &else &if %move% &then
            &s amt 500
        &else
            &s amt 175
        &s mainx %coord1% - %amt%
        &s mainy %coord2%
        move %mainx% %mainy%
        textsize .12
        text %newmin%
        move [calc %mainx% + 110] [calc %mainy% + 75]
        textsize .1
        text ','
        &s e %redd%
        &call geo2utm
        &s quest %coord2%
        &call placecheck
        &if %bigmove% &then
            &s amt 700
        &else &if %move% &then
            &s amt 500
        &else
            &s amt 175
        &s mainx %coord1% + %amt%
        &s mainy %coord2%
        move %mainx% %mainy%
        textsize .12
        text %newmin%
        move [calc %mainx% + 110] [calc %mainy% + 75]
        textsize .1
        text ','

        &s newmin %newmin% + 4
    &end  /* do
    &s newmin %newmin% + 1
&end  /* while


/*  GEOGRAPHIC MARKERS  ********************************


&s e %ledd% + .016667    /* easting sent to geo2utm routine
&s n1 %lndd% + .0008      /* lower northing sent to geo2utm
       routine
&s re %ledd% + .3         /* right easting
&s n2 %undd% - .0008      /* upper northing sent to geo2utm
       routine

mbegin

markerset municipal.mrk
markersymbol 114
&do &while %e% < %redd%
    &s n %n1%
    &call geo2utm
    marker %coord1% %coord2%
    &s n %n2%
    &call geo2utm
    marker %coord1% %coord2%
    &s e %e% + .016667
&end


&s e1 %ledd% + .0009      /* left easting
&s e2 %redd% - .0009      /* right easting
&s n %lndd% + .016667    /* northing

markersymbol 112
&do &while %n% < %undd%
    &s e %e1%
    &call geo2utm
    marker %coord1% %coord2%
    &s e %e2%
    &call geo2utm
    marker %coord1% %coord2%
    &s n %n% + .016667
&end

mend

/*  NORTH ARROW  ****************************************

mbegin

&s orgx [calc %newlrx% - 3000]
&s newy [calc %newlly% + 6400]
line %orgx% %newlly%  %orgx% %newy%
&s e %orgx%
&s n %newlly%
&s hemi y  /* in northern hemisshere? yes
&call utm2geo
&s e %coord1%
&s n [calc %coord2% + .03]
&call geo2utm
line %orgx% %newlly%  %coord1% %coord2%

textjustification cc
```

```
markersymbol 69
markersize .1
marker %coord1% %coord2%
textsymbol 10
textsize .05
textmask rectangle .01
move %orgx% [calc %coord2% + 100]
text GN

mend 'North Arrow'

mmove 0 5 0 0

/*  ELEVATION  GUIDE   ***************************************

maplimits 20 .5 23.5 4
clipmapextent off
mapposition cen cen
mapscale 400000
linesymbol 1
arcs ../coverages/clip

/*  DRAW THE LATITUDE LINES  *****
&s yvar %newlry%
&do &while %yvar% <= %uplefty%
    &if [MOD %yvar% 10000] = 0 &then &do
        &s tmp1 [calc %yvar% - %lowlefty%]
        &s tmp2 [calc %tmp1% * %riserun_l%]
        &s leftx [calc %lowleftx% - %tmp2%]
        &s tmp1 [calc %yvar% - %lowrighty%]
        &s tmp2 [calc %tmp1% * %riserun_r%]
        &s rightx [calc %lowrightx% - %tmp2%]
        line %leftx% %yvar%  %rightx% %yvar%
    &end  /* do
    &s yvar %yvar% + 1000
&end  /* while

/*  DRAW THE LONGITUDE LINES  *****

&s xvar %newllx%
&do &while %xvar% <= %newlrx%
    &if [MOD %xvar% 10000] = 0 &then &do
        &s tmp1 [calc %xvar% - %lowleftx%]
        &s tmp2 [calc %tmp1% * %riserun_b%]
        &s boty [calc %lowlefty% + %tmp2%]
        &s tmp1 [calc %xvar% - %upleftx%]
        &s tmp2 [calc %tmp1% * %riserun_t%]
        &s topy [calc %uplefty% + %tmp2%]
        line %xvar% %boty%  %xvar% %topy%
    &end  /* do
    &s xvar %xvar% + 1000
&end  /* while

&s mvx [calc [calc %upleftx% + %uprightx%] / 2]
&s mvy [calc %uprighty% + 800]
move %mvx% %mvy%
textsymbol 12
textsize .15
text 'ELEVATION GUIDE'

/*  COORDINATE TEXT  *****
&s xdist [calc %lowleftx% - 1300]
&s rnd_label [calc 10 * [calc 1 + [TRUNCATE [calc %newlly% /
       10000]]]]
&s ydist [calc %rnd_label% * 1000]

/*  CALCULATE THE X DISTANCE
&s temp [calc %ydist% - %newlly%]
&s temp2 [calc %temp% / 1000]
&s temp3 [calc %temp2% * 32]
&s xdist [calc %xdist% - %temp3%]

&do &while %ydist% < %newuly%
    &s label [MOD %rnd_label% 100]
    textsymbol 10
    textsize .15 .15
    move %xdist% %ydist%
    &if %label% < 10 &then
        &call less
    &else
        text %label%

    &s tempx [calc %xdist% - 1500]
    &s bigy [TRUNCATE [calc %ydist% / 100000]]
    textsize .07 .07
    move %tempx% %ydist%
    text %bigy%


    &s xdist [calc %xdist% - 320]
    &s ydist [calc %ydist% + 10000]
    &s rnd_label [calc %rnd_label% + 10]
&end

&s ydist [calc %lowlefty% - 900]
&s rnd_label [calc 10 * [calc 1 + [TRUNCATE [calc %newllx% /
       10000]]]]
&s xdist [calc %rnd_label% * 1000]

/*  CALCULATE THE Y DISTANCE
&s temp [calc %xdist% - %newllx%]
&s temp2 [calc %temp% / 1000]
&s temp3 [calc %temp2% * 25]
```

```
&s ydist [calc %ydist% + %temp3%]

&do &while %xdist% < %newlrx%
    &s label [MOD %rnd_label% 100]
    textsize .15 .15
    move [calc %xdist% + 400] %ydist%
    &if %label% < 10 &then
        &call less
    &else
        text %label%

    &s bigx [TRUNCATE [calc %xdist% / 100000]]
    textsize .07 .07
    move [calc %xdist% - 800] %ydist%
    text %bigx%


    &s xdist [calc %xdist% + 10000]
    &s ydist [calc %ydist% + 400]
    &s rnd_label [calc %rnd_label% + 10]
&end

shadeset colornames
resel ../dted/temp_clip poly grid-code = 580
polygonshades ../dted/temp_clip 32
asel ../dted/temp_clip poly
resel ../dted/temp_clip poly grid-code = 520
polygonshades ../dted/temp_clip 33
asel ../dted/temp_clip poly
arcs ../dted/temp_clip


/*  ELEVATION BAR  **********************************

textsize .1
units page
mbegin
shadesymbol 32
patch 20.3 2 20.8 2.1
shadesymbol 33
patch 19.8 2 20.3 2.1
box 20.3 2 20.8 2.1
box 19.8 2 20.3 2.1
box 19.3 2 19.8 2.1
move 19.55 2.17
text Low
move 20.05 2.17
text Medium
move 20.55 2.17
text High
mend
mrotate 90


textsize %textgross%

/*  SCALE BAR & CONTOUR INFO  *******************************

plot scal.gra 8 4


move 11 3.5
textsymbol 12
textsize .15
text 'CONTOUR INTERVAL 20 METERS'

textcolor brown
move 11 3.7
text 'ELEVATIONS IN METERS'

/*  LEGEND  *********************************************

move 2.4 4.5
textcolor 1
text LEGEND
textsymbol 10
textjustification ll
keybox .5 .15
keyposition 3 4.05
keyshade class1.leg
keyposition .9 4.05
keyline line.leg nobox

plot atc.gra box 13.8 2 17.3 3.2

textsymbol 12
move .8 .8
text 'GRAFENWOEHR, GERMANY  EDITION ?-TERRASIM'


/*  TEXT IFORMATION  ************************************

plot info_box.gra 9 .65
textsymbol 10
textsize .1
textalignment right
move 5.5 2.5
textfile prj_info.txt block

/*  MAKE A GRAPHIC & PLOT FILE  ****************************

&if [Query 'Would you like to make a graphics or plot file']
      &then &do
    disp 1040
```

```
    %comp%  /* input graphic name required by arcplot
    plot %comp%.map
    q
    &if [Query 'Would you like to make a plot file'] &then &do
        rotateplot %comp%.gra %comp%r.gra
        hpgl2 %comp%r.gra %comp%.hpgl
        &if [Query 'Would like to send the file to the
       plotter'] &then
            &if [Query 'Enter "y" if you would like to send to
       hp650c2'] &then
                lp -dhp650c2 %comp%.hpgl
            &else
                lp -dhp650c1 %comp%.hpgl
    &end  /* do
&end  /* do


&echo &off
&return

/*************************************************************
/*  LABELS LESS THAN 10 ROUTINE
/*******************************************************
&routine less

&if %label% = 9 &then
    text 09
&else &if %label% = 8 &then
    text 08
&else &if %label% = 7 &then
    text 07
&else &if %label% = 6 &then
    text 06
&else &if %label% = 5 &then
    text 05
&else &if %label% = 4 &then
    text 04
&else &if %label% = 3 &then
    text 03
&else &if %label% = 2 &then
    text 02
&else &if %label% = 1 &then
    text 01
&else
    text 00

&return

/*********************************************************
/*  EDGE LABELS LONGITUDE ROUTINE
/**********************************************************
&routine edge_long

&do &while %num% < %times%

    &if [MOD %xdist% 10000] = 0 &then
      &do
        &s bigx [TRUNCATE [calc %xdist% / 100000]]
        textsize .07
        move [calc %xdist% - 70] %ydist%
        text %bigx%
        textsize .16
        &s tempx [calc %xdist% + 75]
      &end  /* do
    &else
        &s tempx %xdist%

    move %tempx% %ydist%
    &s newx [calc %xdist% / 1000]
    &s label [MOD %newx% 100]
    &if %label% < 10 &then
        &call less
    &else &do
        &if %label% = 12 AND %upper% &then
            move %tempx% [calc %ydist% + 270]  /*  create space
       for geo label
        &else &if %label% = 13 &then
            move %tempx% [calc %ydist% - 230]
        text %label%
    &end  /*do

    &s num %num% + 1
    &s xdist %xdist% + 1000
    &s ydist %ydist% + %interval%
&end

&return

/*********************************************************
/*  INTERNAL UTM LONGITUDE LABELS ROUTINE
/**********************************************************
&routine inside_long

&s num 1
mbegin
&do &while %num% < %times%
    move %xdist% %ydist%
    &s tempy [calc %ydist% / 1000]
    &s label [MOD %tempy% 100]
    &if %label% < 10 &then
        &call less
    &else
        text %label%
    &s num %num% + 1
```

```
    &s ydist %ydist% + 1000
&end
mend

&return

/*********************************************************
/*  INTERNAL UTM LATIITUDE LABELS ROUTINE
/*********************************************************
&routine inside_lat

mbegin
&do &while %num% < 23
    move %xdist% %ydist%
    &s tempx [calc %xdist% / 1000]
    &s label [MOD %tempx% 100]
    &if %label% < 10 &then
        &call less
    &else
        text %label%
    &s num %num% + 1
    &s xdist %xdist% + 1000
&end
mend

&return

/*********************************************************
/*  CHECK IF UTM LABEL PROXIMITY ROUTINE
/*********************************************************
&routine placecheck

&s move .FALSE.
&s bigmove .FALSE.
&s lwrnd [calc [TRUNCATE [calc %quest% / 1000]] * 1000]
&s uprnd [calc  %lwrnd% + 1000]

&if [calc %coord2% - [TRUNCATE %quest%]] = 0 &then
    &s move .TRUE.
&else &if [calc %quest% - %lwrnd%] < %dist% &then
    &if [MOD %lwrnd% 10000] = 0 &then
        &s bigmove .TRUE.
    &else
        &s move .TRUE.
&else &if [calc %uprnd% - %quest%] < %dist% &then
    &if [MOD %uprnd% 10000] = 0 &then
        &s bigmove .TRUE.
    &else
        &s move .TRUE.

&return
```

```
/*********************************************************
/*   GEO2UTM ROUTINE WRITTEN BY TOBI STEINBERG
/*********************************************************

&routine geo2utm

/* receiving input

/*   &label badlong
/*   &s e [response 'What is your longitude?']
/*   &if [null %e%] &then &return
/*   &if %e% gt 180 OR %e% lt -180 &then &do
/*      &ty Please enter a number between -180 and 180
/*      &goto badlong
/*   &end

/*   &label badlat
/*   &s n [response 'What is your latitude?']
/*   &if [null %n%] &then &return
/*   &if %n% gt 90 OR %n% lt -90 &then &do
/*      &ty Please enter a number between -90 and 90
/*      &goto badlat
/*   &end

/* projecting coords

   &if [exists xxxutm.txt -file] &then [delete xxxutm.txt -
       file]
   &if [exists xxxgeo.txt -file] &then [delete xxxgeo.txt -
       file]
   &s out [open xxxutm.txt os -write]
   &s space [unquote ' ']
   &s ne [quote %e%%space%%n%]
   &s junk [write %out% %ne%]
   &s junk [close %out%]

/* determining zone

   &if %n% gt 0 &then &do
      &s z [truncate [calc [calc [calc %e% + 6] / 6] + 30]]
      &ty UTM Zone %z%
      arc project file xxxutm.txt xxxgeo.txt
      input
         projection geographic
         units dd
         spheroid wgs84
         datum wgs84
         parameters
      output
         projection utm
         units meters
         zone %z%
```

```
         spheroid wgs84
         datum wgs84
         parameters
      end
   &end
   &else &do
      &s z [truncate [calc [calc [calc %e% + 6] / 6] + 30]]
      &ty UTM Zone %z%
      arc project file xxxutm.txt xxxgeo.txt
      input
         projection geographic
         units dd
         spheroid wgs84
         datum wgs84
         parameters
      output
         projection utm
         units meters
         spheroid wgs84
         datum wgs84
         parameters
         %e% 0 0
         %n% 0 0
      end
   &end

   &s junk [delete xxxgeo.prj -file]
   &s geo [open xxxgeo.txt os -read]
   &s geoin [read %geo% rs]
   &s junk [close %geo%]
   &s junk [delete xxxutm.txt -file]
   &s junk [delete xxxgeo.txt -file]
   &s geoin [trim %geoin%]
   &ty %geoin%
   /*  separates the two coordinates
   &s place [index %geoin% ' ']
   &s place %place% - 1
   &s coord1 [substr %geoin% 1 %place%]
   &s place %place% + 9
   &s coord2 [substr %geoin% %place%]



&return

/*************************************************************
/*   UTM2GEO ROUTINE WRITTEN BY TOBI STEINBERG
/*************************************************************
&routine utm2geo

/* receiving input
/*   &s e [response 'What is your easting?']
/*   &if [null %e%] &then &return
/*   &s n [response 'What is your northing?']
/*   &if [null %n%] &then &return

/*   &label badzone
/*   &s z [response 'What is your zone? <32>' 32]
/*   &if %z% gt 60 &then &do
/*      &ty Please enter a number less than 60.
/*      &goto badzone
/*   &end
/*   &if [truncate %z%] ne %z% &then &s z [truncate %z%]

/*   &s hemi [response 'Is this in the northern hemisphere?
       [y]' y]

/* projecting coords

   &if [exists xxxutm.txt -file] &then [delete xxxutm.txt -
       file]
   &if [exists xxxgeo.txt -file] &then [delete xxxgeo.txt -
       file]
   &s out [open xxxutm.txt os -write]
   &s space [unquote ' ']
   &s ne [quote %e%%space%%n%]
   &s junk [write %out% %ne%]
   &s junk [close %out%]

   &if %hemi% = y &then &do
      arc project file xxxutm.txt xxxgeo.txt
      input
         projection utm
         units meters
         zone %z%
         spheroid wgs84
         datum wgs84
         parameters
      output
         projection geographic
         units dd
         spheroid wgs84
         datum wgs84
         parameters
      end
   &end
   &else &do
      &s lat -10
      &s long [calc [calc [calc %z% - 30] * 6] + 3]
      arc project file xxxutm.txt xxxgeo.txt
      input
         projection utm
         units meters
```

```
         spheroid wgs84
         datum wgs84
         parameters
         %long% 0 0
         %lat% 0 0
      output
         projection geographic
         units dd
         spheroid wgs84
         datum wgs84
         parameters
      end
   &end

   &s junk [delete xxxgeo.prj -file]
   &s geo [open xxxgeo.txt os -read]
   &s geoin [read %geo% rs]
   &s junk [close %geo%]
   &s junk [delete xxxutm.txt -file]
   &s junk [delete xxxgeo.txt -file]
   &s geoin [trim %geoin%]
   &s place [index %geoin% ' ']
   &s place %place% - 1
   &s coord1 [substr %geoin% 1 %place%]
   &s place %place% + 10
   &s coord2 [substr %geoin% %place%]

&return
/*************************************************************
```

Necessary text files for grafplot.aml:

prj_info.txt

GRID . . . . . .1000 METER UTM ZONE 32
PROJECTION . . . . . TRANSVERSE MERCATOR
VERTICAL DATUM . . . . . .MEAN SEA LEVEL
HORIZONTAL DATUM . . . . . WORLD GEODETIC SYSTEM 1984
PRINTED BY . . . . . .TERRAIN SIMULATION

lft_info.txt

SAMPLE 1,000 METER GRID SQUARE

Sample
Point

100,000 M. SQUARE IDENTIFICATION

PA

GRID ZONE DESIGNATION

32U

rt_info.txt

100 METER REFERENCE

1. Read large numbers labeling the VERTICAL grid line left of point and estimate tenths (100 meters) from grid line to point. 12 3

2. Read large numbers labeling the HORIZONTAL grid line below point and estimate tenths (100 meters) from grid line to point. 45 6

Example: 123456

WHEN REPORTING ACROSS A 100,000 METER LINE, PREFIX THE 100,000 METER SQUARE IDENTIFICATION IN WHICH THE POINT LIES.

Example: PA123456

WHEN REPORTING OUTSIDE THE GRID ZONE DESIGNATION AREA, PREFIX THE GRID ZONE DESIGNATION

Example: 32UPA123456

2.2.3 import_itin.aml

PURPOSE: This program will convert TIN **.addwams** files into Arc/Info point and polygon coverages. It was written specifically to handle addwams files created by iTIN.

HISTORY: This program was written in December 1997 by Tobi Sellekaerts.

USAGE: To run this program, first navigate to the directory where the addwams files you would like to convert are located. Next, at the Arc/Info prompt, issue the following command:

```
Arc: &r import_itin
```

The AML doesn't have to reside in the same directory as the addwams files, it just has to be run from there.

SAMPLE OUTPUT: A directory containing the following addwams files:

```
1-0-point.addwams  1-2-point.addwams  2-0-point.addwams  2-2-point.addwams
 1-0-poly.addwams   1-2-poly.addwams   2-0-poly.addwams   2-2-poly.addwams
1-1-point.addwams  1-3-point.addwams  2-1-point.addwams  2-3-point.addwams
 1-1-poly.addwams   1-3-poly.addwams   2-1-poly.addwams   2-3-poly.addwams
```

would have the following Arc/Info coverages after the program was complete:

```
1_0_POINT        1_0_POLY         1_1_POINT        1_1_POLY
1_2_POINT        1_2_POLY         1_3_POINT        1_3_POLY
2_0_POINT        2_0_POLY         2_1_POINT        2_1_POLY
2_2_POINT        2_2_POLY         2_3_POINT        2_3_POLY
```

PROGRAM FLOW: This program loops through each of the addwams files in the directory, figuring out when it has run out of files. It creates a polygon and point coverage for each pair of addwams files. There is a section at the end of the AML where the user can modify it in order to process the resulting point and polygon files (i.e. modify attributes, combine them, create a TIN, etc.)

tlss

```
/* import_itin.aml
/* ------------------------------------------------------------
/*      Program: import_itin.aml
/*      Purpose: To import addwams TIN files into Arc/Info.
       You must
/*      run this program in the same directory where your
       addwams
/*      files are located.  The program creates a point and
       polygon
/*      coverage for each addwams cell called <row>_<col>_poly
       and
/*      <row>_<col>_point.
/* ------------------------------------------------------------
/*      Called By:  none
/*      Calls Made:  none
/* ------------------------------------------------------------
/*      Arguments:  none
/*      Globals:  none
/* ------------------------------------------------------------
/*      Inputs: addwams files
/*      Outputs: arc/info coverages
/* ------------------------------------------------------------
/*      History: written by Tobi Sellekaerts
/*              steinbet@email.grafenwoehr.army.mil
/*      Terrain Simulations - 7th ATC - Grafenwoehr, Germany
/* ------------------------------------------------------------
&severity &error &routine bail
&echo &on
display 0
&ty [date -VFULL]

&s lp1test .TRUE.
&s lp2test .TRUE.

&s x 0
&s y 0

&do &while %lp1test%
  &do &while %lp2test%
    &if [exists ./%x%-%y%-poly.addwams -file] &then &do
        &if ^ [exists %x%_%y%_poly -cover] &then mossarc ~
          ./%x%-%y%-poly.addwams %x%_%y%_poly line
        &if ^ [exists %x%_%y%_point -cover] &then mossarc ~
          ./%x%-%y%-point.addwams %x%_%y%_point point
        &s y %y% + 1
      &end
    &else &do
        &s ymax %y% - 1
        &s lp2test .FALSE.
      &end
  &end
  &s y 0
  &s x %x% + 1
  &s lp2test .TRUE.
  &if ^ [exists ./%x%-%y%-poly.addwams -file] &then &do
      &s lp1test .FALSE.
      &s xmax %x% - 1
    &end
&end

&do x = 0 &to %xmax%
  &do y = 0 &to %ymax%

     /* if you want to put coverages together or do anything
      else
     /* to them, modify this section.  It is already set up to
     /* loop through all of the coverages.

  &end
&end

quit

&echo &off
display 9999
&ty [date -VFULL]
&return
/*************************************************************
/* bail routine
/*************************************************************
&routine bail
&severity &error &ignore
&severity &warning &ignore
&type An error has occured in import_itin.aml
&type Bailing out of import_itin.aml
&return; &return &error
```

2.2.4 import_s1kmodels.perl

PURPOSE: This program will convert an S1000 models text file into a text file readable by the Arc/Info GENERATE command. When thinning buildings in S1000, it is sometimes necessary to do the thinning in S1000 and not in Arc/Info. In order to include the buildings on the final paper maps, you have to import these final building locations back into Arc/Info. In order to place symbols on the map using the map building AML, do the following:

1. Run this program on the final text file
2. Import the output text file into Arc/Info using GENERATE
3. Build the coverage with the POINT option
4. Buffer the point coverage (6 meters worked well for our 1:50,000 scale maps) to create polygons for the paper map sheet.

HISTORY: This program was written in February 1998 by Tobi Sellekaerts.

USAGE: You can run this program from anywhere:

```
host% import_s1kmodels.perl
```

SAMPLE OUTPUT: The following models text file:

```
1  20454.19   9376.37 0.0   96.34 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
1  21319.48   9357.07 0.0    3.00 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
1  21343.89   9356.24 0.0    3.00 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
1  20453.36   9333.31 0.0   96.34 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
1  21148.42   9328.02 0.0   96.34 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
1  21146.49   9302.11 0.0   96.34 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
1  21340.62   9243.09 0.0    3.00 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
1  19247.99   8467.20 0.0  112.84 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
1  19459.60   8398.08 0.0  112.84 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
1  19451.56   8378.34 0.0  112.84 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
```

would produce the following GENERATE text file:

```
1, 710454.19, 5509376.37
2, 711319.48, 5509357.07
3, 711343.89, 5509356.24
4, 710453.36, 5509333.31
5, 711148.42, 5509328.02
6, 711146.49, 5509302.11
7, 711340.62, 5509243.09
8, 709247.99, 5508467.2
9, 709459.6, 5508398.08
10, 709451.56, 5508378.34
```

PROGRAM FLOW: This program asks the user for the name (and directory, if applicable) of the input and output text files. It then loops through the input text file, reformatting it to fit Arc/Info generate requirements. No information is retained other than the location of each model.

tlss

```
#! /usr/local/bin/perl

# ------------------------------------------------------------------
# This program imports a models.txt file which is also used by s1k
# so that you can turn it into a polygon coverage for use in the map
# creation process.  Import the resulting text file using GENERATE.
# ------------------------------------------------------------------

# get the name of the input and output files
print "What is the name of your s1000 models text file? ";
$file = <STDIN>;
chop($file);
open(IN, "$file");

print "\n";
print "If you specify an output file that already exists, the script \n";
print "will simply append any new lines to the existing file.\n";
print "\n";

system("ls\n");
print "What would you like to call your generate text file? ";
$output = <STDIN>;
chop($output);
open(OUT, ">>$output");

# get the lower left coordinates of the playbox in S1000 coordinates
print "What is the lower left corner of your playbox in UTM coordinates?\n";
print "Easting: ";
$east0 = <STDIN>;
chop($east0);
print "Northing: ";
$north0 = <STDIN>;
chop($north0);
$model_type = 0;

# read in each line of the input file, reformat, and output
while($inline = <IN>) {
    $model_type += 1;
    $space = index($inline, ' ') + 1;
    $east1 = substr($inline, $space, 10) * 1;
    $space += 10;
    $north1 = substr($inline, $space, 10) * 1;
    # recalculate the easting and northing for S1K
    $east = $east1 + $east0;
    $north = $north1 + $north0;

    # output the final line
    print OUT "$model_type, $east, $north\n";

}

print OUT "END\n";
close(OUT);
close(IN);

print "Now use the GENERATE command in Arc/Info as shown:\n";
print "     Arc:  GENERATE <cover>\n";
print "     Generate: input $output\n";
print "     Generate: points\n";
print "     Generate: quit\n";
print "\n";
print "Happy generating!\n";
```

2.2.5 itin2vrml.prl

PURPOSE: This program will convert ITIN **.addwams** files into VRML 1.0 **.wrl** files for viewing in a VRML 1.0-compliant browser. An SGI version exists called **itin2vrml_sgi.prl** - the only difference is the perl binary reference in the first line.

HISTORY: This program was originally coded in December 1997 by Peri Cope, and last updated on 12/17/97.

USAGE: To run this program, first navigate to the directory where the **.addwams** files you would like to convert are located. Next, at the shell prompt, issue the following command:

```
host% itin2vrml.prl <1st_outer_range> <1st_inner_range>
<2nd_outer_range> <2nd_inner_range> <output_path>
<vertical_exageration>
```

where the first four parameters are used to delineate the range of addwams files you would like to convert, “output_path” gives the full pathway where you would like the **.wrl** output files to be written, and “vertical_exageration” denotes the factor by which you would like your **.wrl** files to be exagerated in the VRML browser display.

For example, the command:

```
brazil% itin2vrml.prl 0 1 0 1 /strange/facial/hair/ 2
```

would convert the following **.addwams** files:

```
0-1-point.addwams 0-1-poly.addwams
```

into the following **.wrl** file:

```
/strange/facial/hair/0-1-itin2vrml.wrl
```

When displayed in a VRML 1.0 browser, this file would be exagerated by vertically by a factor of 2.

SAMPLE OUTPUT: n/a

PROGRAM FLOW: This program begins by looping through each line of the point file and writing each point’s x, y, and z coordinates to an appropriate array. The program then opens the polygon file and matches each polygon vertex with it’s corresponding point from the point file. Next, individual point z-values are given red, green, and blue color values based on their relative elevation. Lastly, the program writes out variables from each temporary array in the appropriate VRML 1.0 format.

mpcope

```
#!/usr/local/bin/perl
# [itin2vrml.prl] PERL script
#
# Purpose:  To take a set of ADDWAMS files that have been generated
# as output by ITIN and convert them into properly formatted
# VRML 1.0 files that can be viewed and examined by a VRML-compliant
# browser (such as CosmoPlayer or vrweb-mesa).
#
# Original Coding:  M. Peri Cope
#                   Software Engineer
#                   Terrain Simulations
#                   Logicon RDA
#                   Grafenwoehr, Germany
#
# Last Update:  12/19/97
#


print "-------------------------------------\n\n\n";

print "     Executing ITIN2VRML.PRL...\n\n\n";

print "-------------------------------------\n\n\n";

&test_args;


# BEGIN LOOPING THROUGH FILES

for ($i = $o_r1; $i <= $o_r2; $i++)  # LOOP FOR OUTER RANGE
{
  for ($j = $i_r1; $j <= $i_r2; $j++)  # LOOP FOR INNER RANGE
  {
    &read_point_file;
    &read_poly_file;
    &set_color_range;
    &print_out;
  }
}

#-------------------------------------
# SUBROUTINE test_args
#-------------------------------------
sub test_args
{

  if ((@ARGV[0] < 0) || (@ARGV[0] > 100))
    {
    print "USAGE:  itin2vrml.prl <1st_out_range> <1st_in_range> <2nd_out_range>\n";
    print "                      <2nd_in_range> <out_path> <vert_exag>\n\n";
    die "Value specified for <1st_out_range> invalid.\n";
    }

  if ((@ARGV[1] < 0) || (@ARGV[1] > 100))
    {
    print "USAGE:  itin2vrml.prl <1st_out_range> <1st_in_range> <2nd_out_range>\n";
    print "                      <2nd_in_range> <out_path> <vert_exag>\n\n";
    die "Value specified for <1st_in_range> invalid.\n";
    }

  if ((@ARGV[2] < 0) || (@ARGV[2] > 100))
    {
    print "USAGE:  itin2vrml.prl <1st_out_range> <1st_in_range> <2nd_out_range>\n";
    print "                      <2nd_in_range> <out_path> <vert_exag>\n\n";
    die "Value specified for <2nd_out_range> invalid.\n";
    }

  if ((@ARGV[3] < 0) || (@ARGV[3] > 100))
    {
    print "USAGE:  itin2vrml.prl <1st_out_range> <1st_in_range> <2nd_out_range>\n";
    print "                      <2nd_in_range> <out_path> <vert_exag>\n\n";
    die "Value specified for <2nd_in_range> invalid.\n";
```

```
    }

  if (!(@ARGV[4]))
    {
    print "USAGE:  itin2vrml.prl <1st_out_range> <1st_in_range> <2nd_out_range>\n";
    print "                      <2nd_in_range> <out_path> <vert_exag>\n\n";
    die "No output path specified.\n";
    }

  if ((!(@ARGV[5])) || ((@ARGV[5] < 1) || (@ARGV[5] > 10)))
    {
    print "USAGE:  itin2vrml.prl <1st_out_range> <1st_in_range> <2nd_out_range>\n";
    print "                      <2nd_in_range> <out_path> <vert_exag>\n\n";
    die "Value specified for <vert_exag> invalid.\n";
    }

  $o_r1 = @ARGV[0];
  $i_r1 = @ARGV[1];
  $o_r2 = @ARGV[2];
  $i_r2 = @ARGV[3];
  $o_p = @ARGV[4];
  $v_e = @ARGV[5];

}


#-------------------------------------
# SUBROUTINE read_point_file
#-------------------------------------
sub read_point_file
{

  $read_point_File = "$i-$j-point.addwams";  # SET POINT FILE PATH

# --------------- PARSE LINES OF POINT FILE AND SAVE VARIABLES --------------->

  print "\n\n-------------------------------------------------------\n";
  print "Reading and parsing $read_point_File.\n";
  print "-------------------------------------------------------\n\n";

  open(InPointFile, $read_point_File)  || die;

  $pt_count = -1;
  $min_elev{$i}{$j} = 0;
  $max_elev{$i}{$j} = 0;

  while ($line = <InPointFile>) # READ FROM POINT FILE
  {
    if ($line =~ /\s*(\d+\D\d+)\s*(\d+\D\d+)/)  # IT'S X,Y LINE
    {
      ($xval, $yval) = $line =~ /\s*(\d+.\d+)\s*(\d+.\d+)/;

      $xval{$i}{$j}{$pt_count} = $xval;
      $yval{$i}{$j}{$pt_count} = $yval;


      if (!($pt_count % 20))
      {
        print "Processing line:  $pt_count\n";
      }

    }
    else
    {
    $pt_count++;

      ($zval) = $line =~ /\s*\S*\s*(\d+.\d+)s*/;

      $zval{$i}{$j}{$pt_count} = $zval;

      if ($zval <= $min_elev)
      {
```

```
        $min_elev{$i}{$j} = $zval;
      }

      if ($zval > $max_elev{$i}{$j})
      {
        $max_elev{$i}{$j} = $zval;
      }
    }
  }

  $tot_points{$i}{$j} = $pt_count;
  $tot_pp1 = ($pt_count + 1);

  print "\nTotal number of points = $tot_pp1\n";
  print "Minimum elevation:  $min_elev{$i}{$j}\n";
  print "Maximum elevation:  $max_elev{$i}{$j}\n\n";

  close(InPointFile);
}


#-------------------------------------
# SUBROUTINE read_poly_file
#-------------------------------------
sub read_poly_file
{

  $read_poly_File = "$i-$j-poly.addwams";   # SET POLY FILE PATH

# ---------------- PARSE LINES OF POLY FILE AND SAVE VARIABLES ---------------->

  print "\n\n-------------------------------------------------------\n";
  print "Now reading and parsing $read_poly_File.\n";
  print "-------------------------------------------------------\n\n";
  open(InPolyFile, $read_poly_File) || die;
  $poly_count = -1;

  while ($line = <InPolyFile>) # READ FROM POLY FILE
  {
    if ($line =~ /\s*(\d+\D\d+)\s*(\d+\D\d+)/)  # IT'S X,Y LINE
    {
      $vrtx_count++;
      ($pxval, $pyval) = $line =~ /\s*(\d+.\d+)\s*(\d+.\d+)/;
      $pxval{$i}{$j}{$poly_count}{$vrtx_count} = $pxval;
      $pyval{$i}{$j}{$poly_count}{$vrtx_count} = $pyval;

      for ($k = 0; $k <= $pt_count; $k++)
      {
        if (($pxval == $xval{$i}{$j}{$k}) && ($pyval == $yval{$i}{$j}{$k}))
        {
          $face_set{$poly_count}{$vrtx_count} = $k;
        }
      }
    }
    else
    {
      $poly_count++;
      $vrtx_count = 0;

      if (!($poly_count % 20))
      {
        print "Processing polygon number:  $poly_count\n";
      }
    }
  }

  $tot_polys{$i}{$j} = $poly_count;
  $tot_polp1 = ($poly_count + 1);
  print "\nTotal number of polygons = $tot_polp1\n\n";
  close(InPolyFile);
}
```

```
#-------------------------------------
# SUBROUTINE set_color_range
#-------------------------------------
sub set_color_range
{
  print "\n\n-------------------------------------------------------\n";
  print "Now setting diffuseColor values for  $read_point_File.\n";
  print "-------------------------------------------------------\n\n";

  $elev_range{$i}{$j} = ($max_elev{$i}{$j} - $min_elev{$i}{$j});
  $red_factor{$i}{$j} = $elev_range{$i}{$j} / 60;
  $green_factor{$i}{$j} = $elev_range{$i}{$j} / 80;
  $blue_factor{$i}{$j} = $elev_range{$i}{$j} / 40;

  for ($l = 0; $l <= $tot_points{$i}{$j}; $l++)
  {
    $zval = $zval{$i}{$j}{$l};
    $difc_red_tmp1 = $zval / $red_factor{$i}{$j};
    $difc_red_tmp2 = (int $difc_red_tmp1 * 10);
    $difc_red_tmp3 = ($difc_red_tmp2 * .001);
    $difc_red{$i}{$j}{$l} = $difc_red_tmp3 + .20;
    $difc_grn_tmp1 = $zval / $green_factor{$i}{$j};
    $difc_grn_tmp2 = (int $difc_grn_tmp1 * 10);
    $difc_grn_tmp3 = ($difc_grn_tmp2 * .001);
    $difc_grn{$i}{$j}{$l} = $difc_grn_tmp3 + .20;
    $difc_blu_tmp1 = $zval / $blue_factor{$i}{$j};
    $difc_blu_tmp2 = (int $difc_blu_tmp1 * 10);
    $difc_blu_tmp3 = ($difc_blu_tmp2 * .001);
    $difc_blu{$i}{$j}{$l} = $difc_blu_tmp3 + .10;

    if (!($l % 20))
    {
      print "Calculating point number:  $l\n";
    }
  }
}

#-------------------------------------
# SUBROUTINE print_out
#-------------------------------------
sub print_out
{

# -------------- OPEN AND PREPARE THE OUTPUT FILE FOR WRITING ---------------->
  $write_File = "$o_p/$i-$j-itin2vrml.wrl";
  if (-e $write_File)  # OVERWRITE OUTPUT VRML FILE IF IT ALREADY EXISTS
  {
    print ('Overwriting ', $write_File, '.', "\n\n");
    system('rm ', $write_File);
  }

  $out_File = "+>$write_File";  # OPEN OUTPUT FILE FOR WRITING
  open(OutFile, $out_File)  || die;

# ---------------- WRITE THE CONVERTED DATA TO THE OUTPUT FILE ------------------>

  print "\n\n-------------------------------------------------------\n";
  print "Now writing output to $out_Path$i-$j-itin2vrml.wrl.\n";
  print "-------------------------------------------------------\n\n";

  select (OutFile);
  print "#VRML V1.0 ascii\n\n";
  print "Separator {\n\n";
  print "  ShapeHints {\n";
  print "    vertexOrdering\tUNKNOWN_ORDERING\n";
  print "    shapeType\tSOLID\n";
  print "    faceType\tCONVEX\n";
  print "  }\n\n";
  print "  DirectionalLight { direction\t0.0000\t2.000\t-3.000 } ";
  print "\n\n  Transform {\n";
  print "\tscaleFactor 1 1 $v_e\n";
  print "  }\n\n";
```

```
  print "  Coordinate3 {\n";
  print "    point [\n";

  for ($m = 0; $m <= $tot_points{$i}{$j}; $m++)  # LOOP FOR EACH POINT
  {
    print "\t-$xval{$i}{$j}{$m}";
    print "   $yval{$i}{$j}{$m}";
    print "   $zval{$i}{$j}{$m},\n";
  }

  print "\t]\n";
  print "  }\n\n";
  print "  Material {\n";
  print "  ambientColor 0 0 0\n\n";
  print "  diffuseColor [\n";

  for ($n = 0; $n <= $tot_points{$i}{$j}; $n++)  # LOOP FOR EACH VERTEX
  {
    print "\t$difc_red{$i}{$j}{$n} $difc_grn{$i}{$j}{$n} ";
    print "$difc_blu{$i}{$j}{$n},\n";
  }

  print "\t]\n  }\n";
  print "  MaterialBinding {\n";
  print "    value PER_VERTEX_INDEXED\n";
  print "  }\n\n";
  print "  IndexedFaceSet {\n";
  print "    coordIndex [\n";

  for ($o = 0; $o <= $tot_polys{$i}{$j}; $o++)  # LOOP FOR EACH FACE
  {
    for ($p = 1; $p < 4; $p++)
    {
      print "\t$face_set{$o}{$p},";
    }
    print "\t-1,\n";
  }

  print "\t]\n";
  print "  }\n";
  print "}\n";
  select (STDOUT);

close(OutFile);
}
```

2.2.6 itin_conv.prl

PURPOSE: This program takes a set of incorrectly formated ADDWAMS files that have been generated as output by ITIN and convert them into properly formatted ADDWAMS files that can be correctly read and interpreted by S1000.

HISTORY: This program was written by M. Peri Cope in December 1997. An SGI version exists called **itin_conv_sgi.prl** - the only difference is the perl binary reference in the first line.

USAGE: First, you must use iTIN to generate a set of ADDWAMS files representing a TIN. (For more information on iTIN, see the iTIN user's manual.) Second, determine how many rows and columns are in the cut up version of your TIN. To do this, look at your ADDWAMS files. They have names like `1-3-point.addwams`. In this case, 1 is the row and 3 is the column. Look for the biggest numbers that exist in these file names for the first and second number - these are your row and column totals. Don't add one to each number to accommodate the row and column counters beginning at 0, the program will account for that. Next move to the directory containing the ADDWAMS files. Run this perl script:

```
host% itin_conv.prl <outer_range> <inner_range> <output_path>
```

where `<outer_range>` is the highest of the first ADDWAMS number (row), `<inner_range>` is the highest of the second ADDWAMS number (column), and `<output_path>` is the destination where you want to put the output files. I highly advise that you put your output files in an alternate directory! (**Note**: Be **sure** to add a slash at the end of your designated output path. Type `../output/addwams/` instead of `../output/addwams.` The final slash is very important!)

SAMPLE OUTPUT: The change in file format is small but important. Excerpts from an original point file, called `1-3-point.addwams`:

```
  0          509.000000                                1
690000.00 5500000.00
  1          520.250000                                1
690500.00 5500500.00
  2          545.000000                                1
690000.00 5500500.00
```

...an original poly file, called `1-3-poly.addwams`:

```
  0          4-12 Land                                  4
690000.00 5500000.00
690500.00 5500500.00
690000.00 5500500.00
690000.00 5500000.00
  1          4-12 Land                                  4
690000.00 5500000.00
690500.00 5500000.00
690500.00 5500500.00
690000.00 5500000.00
```

...an output point file, called `001_003-point.addwams`:

```
 1                509.00     1
       690000.00        5500000.00
 2                520.25     1
       690500.00        5500500.00
 3                545.00     1
       690000.00        5500500.00
```

...an output poly file, called `001_003-point.addwams`:

```
1                004-012                        4
      690000.00        5500000.00
      690500.00        5500500.00
      690000.00        5500500.00
      690000.00        5500000.00
2                004-012                        4
      690000.00        5500000.00
      690500.00        5500000.00
      690500.00        5500500.00
      690000.00        5500000.00
```

PROGRAM FLOW: The program reads in and out each ADDWAMS file in turn, changing the spacing of the columns. Point files are processed first, followed by the polygon files.

tlss

```
#!/usr/local/bin/perl
# [itin_conv.prl] PERL script
#
# Purpose:  To take a set of incorrectly formated ADDWAMS files that have been
#           generated as output by ITIN and convert them into properly formatted
#           ADDWAMS files that can be correctly read and interpreted by S1000.
#
# Original Coding:  M. Peri Cope
#                   Software Engineer
#                   Terrain Simulations
#                   Logicon RDA
#                   Grafenwoehr, Germany
#
# Last Update:  12/10/97
#

if (((@ARGV[0] < 0) || (@RGV > 100)) || ((@ARGV[1] < 0) || (@ARGV[1] > 100)) ||
  (!@ARGV[2]))
  {
  die "USAGE: itin_conv_sgi.prl <outer_range> <inner_range> <output_path>\n";
  }

print "-------------------------------------\n\n\n";

print "   Now executing ITIN_CONV.PRL...\n\n\n";

print "-------------------------------------\n\n\n";

$out_Range = @ARGV[0];

$in_Range = @ARGV[1];

$out_Path = @ARGV[2];

#  THE FOLLOWING FORMAT WILL BE USED FOR DATA OUTPUT TO POINT FILES
format OutFile =
@>>>@>>>>>>>>>>>>>>>>>>.@>@>>>>>
$c_pt $zval  $zval_dec  $attr
@>>>>>>>>>>>>>>>>>@>>>>>>>>>>>>>>>>>
$xval             $yval
.


# THE FOLLOWING FORMAT WILL BE USED FOR DATA OUTPUT TO POLY FILES (FIRST LINES)
format FIRST_LINE =
@>>>@>>>>>>>>>>>>>>>>>>-@>>                      @>>
$c_ppt $patr1A         $patr2A              $patr3
.


# THE FOLLOWING FORMAT WILL BE USED FOR DATA OUTPUT TO POLY FILES (COORD LINES)
format OutFilePoly =
@>>>>>>>>>>>>>>>>>@>>>>>>>>>>>>>>>>>
$pxval             $pyval
.


for ($i = 0; $i < 2; $i++)  # LOOP FOR POINT (0) AND POLY (1) FILES
  {
    for ($j = 0; $j <= $out_Range; $j++)  # LOOP FOR OUTER RANGE
      {
        for ($k = 0; $k <= $in_Range; $k++)  # LOOP FOR INNER RANGE
          {

            if ($i == 0)  # SET INPUT/OUTPUT FILE TYPES
              {
                $F_type = 'point';
              }

            if ($i == 1)
              {
                $F_type = 'poly';
```

```
    }

  $read_File = "$j-$k-$F_type.addwams";  # SET INPUT FILE PATH VARIABLE

if (!(-e $read_File))  # INPUT FILE DOES NOT EXIST
    {
      print "WARNING:  The input files do not exist!!!";
   print "Try again!";
    die "\n";
  }
  else  # OPEN INPUT FILE FOR READING
    {
      open(InFile, $read_File)  || die;
  }


if ($j < 10)  # SET VALUES FOR $J_NEW TO INCLUDE IN OUTPUT FILE NAME
  {
    $j_new = "00$j";
  }
elsif (($j > 9) && ($j < 100))
  {
   $j_new = "0$j";
  }
else
    {
   $j_new = $j;
  }


if ($k < 10)  # SET VALUES FOR $K_NEW TO INCLUDE IN OUTPUT FILE NAME
    {
      $k_new = "00$k";
    }
  elsif (($k > 9) && ($k < 100))
    {
      $k_new = "0$k";
    }
  else
    {
      $k_new = $k;
    }

# SET OUTPUT FILE PATH VARIABLE
$write_File = "$out_Path$j_new"."_"."$k_new-$F_type.addwams";

if ($i == 0)  # WRITE OUT CORRECT TEXT FOR POINT FILES
  {

   if (-e $write_File)  # REMOVE OUTPUT POINT FILE IF IT ALREADY EXISTS
        {
          print ('Overwriting ', $write_File, '.', "\n\n");
          system('rm ', $write_File);
        }

   $out_File = "+>$write_File";  # OPEN OUTPUT FILE FOR WRITING
   open(OutFile, $out_File)  || die;

   $line_no = 0;
    while ($line = <InFile>)  # BEGIN READING IN LINES FROM INPUT FILE
      {
        $line_no++;

        if ($line_no % 2)  # LINE NUMBER IS ODD/FIRST LINE OF A POINT
         {
              ($pt, $zval, $zval_dec, $attr) = $line
                =~ /\s*(\d+)\s*(\d+)\W(\d+)\s*(\S+)/;
              $c_pt = $pt + 1;
         }
        else  # LINE NUMBER IS EVEN/SECOND (LAST) LINE OF A POINT
         {
              ($xval, $yval) = $line =~ /\s*(\d+.\d+)\s*(\d+.\d+)/;
```

```
            $~ = "OutFile";   # SET FORMAT TO OutFile FORMAT
               write OutFile;  # WRITE OUT FORMATTED DATA TO OUTFILE
              }
         }
      }

    if ($i == 1)  # WRITE OUT CORRECT TEXT FOR POLYGON FILES
        {

          if (-e $write_File)  # REMOVE OUTPUT POLY FILE IF IT ALREADY EXISTS
            {
              print ('Overwriting ', $write_File, '.', "\n\n");
              system('rm ', $write_File);
            }

          $out_File_Poly = "+>$write_File";  # OPEN OUTPUT FILE FOR WRITING
          open(OutFilePoly, $out_File_Poly)  || die;

        $ppt = 000;

        while ($line = <InFile>)  # BEGIN READING IN LINES FROM INPUT FILE
          {
              if ($line =~ /\s*(\d+\D\d+)\s*(\d+\D\d+)/)  # IT'S X,Y LINE
            {
                  $~ = "COORD_LINE";
                  ($pxval, $pyval) = $line =~ /\s*(\d+.\d+)\s*(\d+.\d+)/;
                  write OutFilePoly;  # WRITE OUT COLLECTED LINES
            }

           else  # IT'S AN ATTRIBUTE LINE
            {
              $~ = "FIRST_LINE";
                  ($ppt, $patr1, $patr2, $patr3) = $line
                    =~ /\s*(\S*)\s*(\d+)\D*(\d+)\s*\S*\s*(\S+)/;
                  $c_ppt = $ppt + 1;

              # SET FIRST ATTRIBUTE TO CORRECT FORMAT
              if ($patr1 < 10)
                {
                  $patr1A = "00$patr1";
                        }
              elsif (($patr1 > 9) && ($patr1 < 100))
                        {
                  $patr1A = "0$patr1";
                        }
              else
                        {
                  $patr1A = $patr1;
                        }

              # SET SECOND ATTRIBUTE TO CORRECT FORMAT
                  if ($patr2 < 10)
                    {
                      $patr2A = "00$patr2";
                    }
                  elsif (($patr2 > 9) && ($patr2 < 100))
                    {
                      $patr2A = "0$patr2";
                    }
                  else
                    {
                      $patr2A = $patr2;
                    }

              write "FIRST_LINE";  # WRITE OUT COLLECTED LINES
                }
         }
        }

# CLOSE INPUT AND OUTPUT FILES BEFORE SHUTTING DOWN PROGRAM

close(InFile);
```

```
            close(OutFile);
            close(OutFilePoly);
           }
       }
  }

print "\n\n\nITIN_CONV.PRL is now done with the ADDWAMS conversion process.\n\n\n";
```

2.2.7 itin_script.prl

PURPOSE: This Perl script executes commands the CMU iTIN program in a sequence designed to create a complete SimNet TIN from start to finish.

HISTORY: This script was written by Tobi Sellekaerts in January 1998, based on a script written by Michael Paul Czechanski in 1997.

USAGE: First, create your road, river, and lake coverages in Arc/Info and export them using the ARCMOSS command to text files. Second, create an iTIN format raster file containing your elevation information (and an accompanying mask if necessary). Third, create a directory to run the program in. It doesn't matter where just as long as it is empty, although it is better to have it on whichever machine is running iTIN. Fourth, execute the perl script:

```
host% ./itin_script.prl
```

Fifth, move into the ADW directory created by the program. Finally, run `itin_conv_sgi.prl` to change the format of the text files. Your ADDWAMS files are now ready for import into S1000.

SAMPLE OUTPUT: The program creates many working files in your directory, all of which can be discarded when it is finished. Most importantly, it creates a subdirectory called ADW into which it places the ADDWAMS files of the TIN.

PROGRAM FLOW: This script simply calls a series of iTIN commands, incorporating roads, rivers, and lakes into a TIN the size of your playbox and exporting this TIN to a set of ADDWAMS files. For more detailed information on the iTIN commands, see the iTIN User's manual.

tlss

```
#! /usr/local/sgi/bin/perl
# ----------------------------------------------------------------
#     Program: itin.perl
#     Purpose: to run itin
#  ----------------------------------------------------------------
#  Change the Arc/Info workspace, directory path, and playbox
#  extents as needed.
#  ----------------------------------------------------------------
#     History:  15 January 1998 - Tobi Sellekaerts
#            steinbet@email.grafenwoehr.army.mil
#     Terrain Simulations - 7th ATC - Grafenwoehr, Germany
#  ----------------------------------------------------------------
system ("adw2brep -i /net/italy/data1/graf/data/final2/roads.addwams -o roads.brep");

system ("adw2brep -i /net/italy/data1/graf/data/final2/hydro.addwams -o hydro.brep");

system ("tclBrep /usr/vdata/tindata/proc_roads.tcl roads.brep roads_proc.brep");

system ("tclBrep /usr/vdata/tindata/proc_water.tcl hydro.brep hydro_proc.brep");

system ("brep_addelev roads_proc.brep ../dem.img roads_z.brep");

system ("brep_addelev hydro_proc.brep ../dem.img hydro_z.brep");

system ("brep_lbreak roads_z.brep ../dem.img roads_break.brep -ss 1 -mf 20 -all");

system ("tclBrep /usr/vdata/tindata/check_roads.tcl roads_break.brep");

system ("ri_widen roads_break.brep roads_wide.brep -c roads_centers.brep");

system ("adw2brep -i /net/italy/data1/graf/data/final2/lakes.addwams -o lakes.brep");

system ("tclBrep /usr/vdata/tindata/proc_water.tcl lakes.brep lakes_proc.brep");

system ("brep_addelev lakes_proc.brep ../dem.img lakes_z.brep");

system ("brep_combine hydro_z.brep lakes_z.brep roads_wide.brep -o features.brep");

system ("brep_grid 500 690000 5500000 716000 5516000 grid.brep -img ../dem.img");

system ("brep_proxrm grid.brep features.brep grid_cull.brep -near 5 -in");

system ("fastitin -M 690000 5500000 716000 5516000 -d ../dem.img -o extent.mnf -sl");

system ("breptoman grid_cull.brep extent.mnf grid.mnf");

system ("brep_clip 690000 5500000 716000 5516000 features.brep features_clip.brep");

system ("brep_dice features_clip.brep 500 features_dice.brep");

system ("breptoman features_dice.brep grid.mnf features.mnf");

system ("fastitin -d ../dem.img -i features.mnf -m ../mask.img -a 150000 -o full.mnf -sf
10000 -so 20 51 -ss 500 -p 70000");

system ("mandice_tsg 500 full.mnf full.tsg");

system ("tsgrename full.tsg export.tsg land=Land");

system ("tsgrename export.tsg export2.tsg Primary\\ Highway=highway");

system ("mkdir adw");

system ("tsgtoaddwams 4 export2.tsg adw");

system ("mkdir s1000");
```

2.2.8 lake_elev.aml

PURPOSE: This aml finds the elevation of a polygonal water feature (the elevation at the label point / center of the polygon) so the polygon can be used as a hard return in TIN construction.

HISTORY: This script was written in March 1997 by Tobi Sellekaerts. It is not currently used as we create most of our TINs in iTIN.

USAGE: First, create a polygon coverage of your polygonal water features (i.e. lakes, oceans). (This program was not intended to handle linear water features such as streams and rivers.) Second, make sure you have your elevation information in the form of an Arc/Info GRID. Third, run this AML at the Arc/Info prompt:

```
Arc:  &r lake_elev.aml <lake_cov> <grid>
```

where `<lake_cov>` is your polygon coverage and `<grid>` is your elevation information. The AML will work regardless of the resolution of your elevation information; however, the coverage and the grid must be in the same projection.

SAMPLE OUTPUT: The program creates one (and possibly two) output coverages with names similar to your initial polygon coverage. The output coverage of lakes with elevation information will be called

`<input_cov_name>_hr` (for 'hard return')

and if polygons with islands exist, they will be output in a coverage called

`<input_cov_name>_nhr` (for 'no hard return')

Polygons with islands cannot be included in a TIN as hard returns, and will be excluded from the 'hard return' coverage.

PROGRAM FLOW: First, the program determines if any of the polygons to be processed have islands by calling an atool called **moveislands.aml**. The polygons with islands are deleted from the original coverage and are not processed further. The program next determines the elevation of the remaining polygons and assigns it to an attribute called `SPOT`.

tlss

```
&args inlake elev

/* lake_elev.aml
/* ------------------------------------------------------------
/*    Program: lake_elev.aml
/*    Purpose: to find the elevation of lakes so they can be
      used
/*    as a hard replace item in the tin creation
/* ------------------------------------------------------------
/*    Called By:  none
/*    Calls Made:  the atool moveislands.aml
/* ------------------------------------------------------------
/*    Inputs: the generalized lake coverage
/*    Outputs: the same lake coverage, with a polygon attribute
/*    item called spot which represents the elevation of the
      lake
/*    Also, if any islands were present, a second lakes
      coverage
/*    which consists of lakes which can't be HARDREPLACEd into
/*    the TIN. Lake coverage name now appended with a _hr
/* ------------------------------------------------------------
/*    History:  17 March 1997 - Tobi Steinberg
/*    Terrain Simulations - 7th ATC - Grafenwoehr, Germany
/* ------------------------------------------------------------
&severity &error &routine bail
display 0
&ty [date -VFULL]

/* ==================================================
/* Pre-AML actions
/* ==================================================
/* Checking validity of entered arguments
&if [null %elev%] &then &do
   &ty Usage: lake_elev <lake_coverage> <elevation_grid>
   &return
&end

/* Checking to be sure that the lakes coverage is a polygon
      coverage
&if ^ [exists %inlake% -poly] &then &do
   &ty %inlake% is not a valid polygon coverage
   &return
&end

&if ^ [exists %elev% -grid] &then &do
   &ty %elev% is not a valid grid
   &return
&end

/* Checking on the existence of coverage names needed by the
      program
&do cov &list xxlakept %inlake%_nhr xxtemp1 %inlake%_hr
   &if [exists %cov% -cover] &then kill %cov% all
&end

/* Working on a copy...
copy %inlake% %inlake%_hr
copy %inlake% %inlake%_nhr
&s nhr %inlake%_nhr
&s inlake %inlake%_hr

countvertices %inlake% poly

/* Checking for the existence of an attribute item named spot
      in the
/* lakes coverage
&s exists 0
tables
   &s loop [show columns info %inlake%.pat]
   &do item &list %loop%
      &if %item% = SPOT &then &s exists 1
   &end
   &if %exists% = 0 &then additem %inlake%.pat spot 10 10 n
quit

/* ===================================
/* Main AML actions
/* ===================================

/* outputting the label points for each polygon to a different
      coverage
ae
   ec %inlake%
   ef label
   sel all
   put xxlakept
quit

dropitem xxlakept.pat xxlakept.pat spot
latticespot %elev% xxlakept

tables
   relate add
      spotelev
      xxlakept.pat
      info
      %inlake%-id
      xxlakept-id
      linear
      rw
      ~

sel %inlake%.pat
calc spot = spotelev//spot
```

```
relate drop
   spotelev
   ~
quit

/* Now that all the lakes have elevations, it is time to remove
       the lakes
/* which have islands so they will not be used as hard
       replaces.
copy %inlake% xxtemp1
moveislands %inlake% %nhr% xxtemp1
&s select 0
ae
   ec xxtemp1
   ef poly
   &if [show number selected] ne 0 &then &do
      put %inlake%
      y
      put %nhr%
      y
   &end
   ec %inlake%
      ef poly
      sel islands gt 0
      put %nhr%
      y
      delete
      save
   ec %nhr%
   ef poly
   sel all
   &if [show number selected] gt 0 &then &s select 1
quit
build %nhr%


/* ====================================
/* Post-AML actions
/* ====================================
/* Cleaning up...
&do cov &list xxlakept xxtemp1
   &if [exists %cov% -cover] &then kill %cov% all
&end

display 9999

&ty Final lakes coverage with elevations: %inlake%
&if %select% = 1 &then &do
&ty Lakes which are not to be used as hard replaces are in
&ty the coverage called %nhr%.
&ty Don't forget about these when making your TIN!
&end
&else kill %nhr% all

&return
/*************************************************************
      *****
/* bail routine
/*************************************************************
      *****
&routine bail
&severity &error &ignore
&severity &warning &ignore
&type An error has occured in lake_elev.aml
&type Bailing out of lake_elev.aml
&return; &return &error
```

2.2.9 model_format.perl

PURPOSE: This perl script reads a text file that contains positional, size, and orientation information about models to be placed on the SimNet terrain, and converts it into a text file in the required S1000 model text file format. A point coverage can be created in Arc/Info designating location, orientation, and type of model to be placed on the S1000 terrain. Create the point coverage, add an item for type designation (I used model_type), calculate angles, use ADDXY to add coordinates, and then UNLOAD it. The model_type should correspond to the ordering of models in your `.mlib` file in S1000 for correct models to be placed. In order for the spacing to work correctly in the script, use the following item definitions:

```
additem models.pat models.pat angle 10 10 n 2
additem models.pat models.pat model_type 2 2 I
```

`x-coord` and `y-coord` will be automatically formatted by ADDXY. The input file for the script is created in Arc/Info's TABLES module with the UNLOAD command. An example:

```
Enter Command: sel models.pat
Enter Command: unload input.txt model_type x-coord y-coord angle
columnar junk.txt
```

HISTORY: This script was written in December 1997 by Tobi Sellekaerts.

USAGE: The usage for this program is as follows:

```
italy% model_format.perl
```

The program will prompt you to enter the names of your input file and output files, and the lower left coordinates of your playbox in UTM meters.

SAMPLE OUTPUT: The program reads in an UNLOAD file like this:

```
 1      710454.18606     5509376.37282     96.34
 2      711319.47800     5509357.06719      3.00
 1      711343.88628     5509356.23553      3.00
 4      710878.29862     5509362.72917     96.34
 9      710453.52231     5509354.93068     96.34
```

and converts it to an S1000 format file:

```
1  20454.19   9376.37 0.0   96.34 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
2  21319.48   9357.07 0.0    3.00 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
1  21343.89   9356.24 0.0    3.00 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
4  20878.30   9362.73 0.0   96.34 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
9  20453.52   9354.93 0.0   96.34 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0
```

PROGRAM FLOW: The program changes the text file created by UNLOAD into a format that S1000 can read from to place models on the terrain. The location of the lower left corner of the playbox in UTM meters is subtracted from the coordinates for each point as S1000 uses relative (to the playbox origin), not absolute (to the earth), meter coordinates. The scale of each model is automatically set to 1; for changes, alter the program or the resulting text file. For more information on the S1000 model text format, see the S1000 User's Guide.

tlss

```
#! /usr/local/bin/perl

# model_format.perl
# ------------------------------------------------------------
#      Program: model_format.perl
#      Purpose: to take in an unload text file created in ARC
       and
#       reformat it to fit the S1000 model library ASCII file
#       requirements.
#       Note changes to be made when not working on the Graf
       file!
# ------------------------------------------------------------
#      Called By:  none
#      Calls Made:  none
# ------------------------------------------------------------
#      Inputs: an unload file from ARC, the lower left
       coordinates
#       of the playbox in S1000 coordinates
#      Outputs: a properly formatted S1K model library ASCII
       file
# ------------------------------------------------------------
#      History:  18 December 1997 - Tobi Sellekaerts
#            steinbet@email.grafenwoehr.army.mil
#      Terrain Simulation - 7th ATC - Grafenwoehr, Germany
# ------------------------------------------------------------

# get the name of the input and output files
print "What is the name of your unload text file? ";
$file = <STDIN>;
chop($file);
open(IN, "$file");

print "\n";
print "If you specify an output file that already exists, the
       script \n";
print "will simply append any new lines to the existing
       file.\n";
print "\n";
print "What would you like to call your output file? ";
$output = <STDIN>;
chop($output);
open(OUT, ">>$output");

# get the lower left coordinates of the playbox in S1000
       coordinates
print "What is the lower left corner of your playbox in UTM
       coordinates?\n";
print "Easting: ";
$east0 = <STDIN>;
chop($east0);
print "Northing: ";
$north0 = <STDIN>;
chop($north0);

# read in each line of the input file, reformat, and output
while($inline = <IN>) {
    $model_type = substr($inline, 0, 2) * 1;
    $east1 = substr($inline, 2, 18) * 1;
    $north1 = substr($inline, 20, 18) * 1;
    $angle = substr($inline, 38, 10) * 1;

    # recalculate the easting and northing for S1K
    $east = $east1 - $east0;
    $north = $north1 - $north0;

    # output the final line
      # note: the following specifications are all oriented
      toward the Graf terrain build!
      # change them to fit your own terrain build needs

    printf OUT "%1s %9.2f %9.2f %3.1f %7.2f", $model_type,
      $east, $north, 0, $angle;
    if($model_type == 1){
      print OUT " 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0\n";
    } elsif($model_type == 7) {
      print OUT " 0.0 0.0 1.2 1.2 1.2 1 0 0 1 0 0 0.0\n";
    } elsif($model_type == 2) {
      print OUT " 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0\n";
    } elsif($model_type == 16) {
      print OUT " 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0\n";
    } elsif($model_type == 15) {
      print OUT " 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0\n";
    } elsif($model_type == 13) {
      print OUT " 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0\n";
    } elsif($model_type == 17) {
       print OUT " 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0\n";
    } elsif($model_type == 18) {
       print OUT " 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0\n";
    } else {
      print OUT " 0.0 0.0 1.0 1.0 1.0 1 0 0 1 0 0 0.0\n";
    }
    # use this line instead of you want to change the
      percentages x y z
    # print OUT " 0.0 0.0 0.8 0.8 0.8 1 0 0 1 0 0 0.0\n";
}
close(OUT);
close(IN);
```

2.2.10 remove_lm.perl

PURPOSE: This perl script removes load module edges from a set of addwams TIN files.

HISTORY: This script was written in January 1998 by Tobi Sellekaerts.

USAGE: Use this script if you want to use iTIN to create road casings that are flat, then use your road casings in your Arc/Info TIN. First, concatenate all your addwams files into two files called

`total-poly.addwams` `total-point.addwams`

Second, execute the program in the directory where your addwams files are located. The usage for this program is as follows:

```
host% remove_lm.perl
```

The program creates two addwams files:

`0-0-poly.addwams` `0-0-point.addwams`

Third, use **import_itin.aml** to import these addwams files into Arc/Info coverages.

SAMPLE OUTPUT: n/a

PROGRAM FLOW: See program code which follows.

tlss

```
#! /usr/local/bin/perl

# remove_lm.perl
# ------------------------------------------------------------
#     Program: remove_lm.perl
#     Purpose: to remove the load module boundaries (points and
#       triangles) from a set of ADDWAMS files from iTIN
# ------------------------------------------------------------
#     Inputs: ADDWAMS files output from iTIN
#     Outputs: new ADDWAMS files without the LM boundaries
# ------------------------------------------------------------
#     History:  13 January 1998 - Tobi Sellekaerts
#            steinbet@email.grafenwoehr.army.mil
#     Terrain Simulation - 7th ATC - Grafenwoehr, Germany
# ------------------------------------------------------------

open(INPOLY, "total-poly.addwams");
open(INPOINT, "total-point.addwams");
open(OUTPOLY, ">0-0-poly.addwams");
open(OUTPOINT, ">0-0-point.addwams");

# process polygons first
while($in0 = <INPOLY>) {
    # read in four lines of coordinates
    $in1 = <INPOLY>;
    $in2 = <INPOLY>;
    $in3 = <INPOLY>;
    $in4 = <INPOLY>;
    $test = 1;      # will be changed if this includes lm edge
coords

    if(substr($in0, 19, 4) eq 'Land') { $test = 0; }

    # if the triangle is not part of the lm edges, write it to
the
    # output file
    if ($test == 1) {
       print OUTPOLY "$in0$in1$in2$in3$in4";
    }
}

# process points secont
while($in0 = <INPOINT>) {
    # read in the coordinate line
    $in1 = <INPOINT>;
    $test = 1;      # will be changed if this includes lm edge
coords

    # check against coordinates; yes, you have to change these
for each playbox
    if(index($in1, " 690000") != -1) { $test = 0; }
    if(index($in1, " 695000") != -1) { $test = 0; }
    if(index($in1, " 700000") != -1) { $test = 0; }
    if(index($in1, " 705000") != -1) { $test = 0; }
    if(index($in1, " 710000") != -1) { $test = 0; }
    if(index($in1, " 715000") != -1) { $test = 0; }
    if(index($in1, " 5500000") != -1) { $test = 0; }
    if(index($in1, " 5505000") != -1) { $test = 0; }
    if(index($in1, " 5510000") != -1) { $test = 0; }
    if(index($in1, " 5515000") != -1) { $test = 0; }

    # if the triangle is not part of the lm edges, write it to
the
    # output file
    if ($test == 1) {
       print OUTPOINT "$in0$in1";
    }
}
close(INPOLY);
close(INPOINT);
close(OUTPOLY);
close(OUTPOINT);
```

2.2.11 stamp_conv.prl

PURPOSE: This program generates an S1000 stamp text file from a MOSS format text file of points created by Arc/Info.

HISTORY: This AML was written by Michael Paul Czechanski in February 1998.

USAGE: Create a polygon coverage with the polygons not used for the canopy.
Run the CENTROIDLABELS command from the Arc prompt.
Add a label attribute item.
Calculate the value to correspond to the appropriate stamp.

```
Arc: arcmoss <infile> <outfile> utm <attribute name> # point
```

run the stamp_conv.prl:

```
host% stamp_conv.prl <file_name> <output_file>
```

where `<file_name>` is the name of the moss text file created by the ARCMOSS command and `<output_file>` is the name of the final s1000-compatible stamp file to be generated. `<output_file>` should not already exist.

SAMPLE OUTPUT: The figure below shows a sample of an ADDWAMS stamp file:

```
24  14862.96  13847.46 0.0    0.00 0.0 0.0 1.0 1.0 1.0 1 0 0 1 0 0 0.0
24  15307.55  13841.13 0.0    0.00 0.0 0.0 1.0 1.0 1.0 1 0 0 1 0 0 0.0
24  11264.11  13559.77 0.0    0.00 0.0 0.0 1.0 1.0 1.0 1 0 0 1 0 0 0.0
24  15410.96  13812.91 0.0    0.00 0.0 0.0 1.0 1.0 1.0 1 0 0 1 0 0 0.0
24  15457.97  13768.38 0.0    0.00 0.0 0.0 1.0 1.0 1.0 1 0 0 1 0 0 0.0
24  15566.98  13680.99 0.0    0.00 0.0 0.0 1.0 1.0 1.0 1 0 0 1 0 0 0.0
24  10765.28  13388.52 0.0    0.00 0.0 0.0 1.0 1.0 1.0 1 0 0 1 0 0 0.0
24   9309.54  13307.05 0.0    0.00 0.0 0.0 1.0 1.0 1.0 1 0 0 1 0 0 0.0
24   8574.14  13180.79 0.0    0.00 0.0 0.0 1.0 1.0 1.0 1 0 0 1 0 0 0.0
24  15722.07  13297.71 0.0    0.00 0.0 0.0 1.0 1.0 1.0 1 0 0 1 0 0 0.0
```

PROGRAM FLOW: See the following code for the program flow.

mpczech

```
#!/usr/local/bin/perl
#
#
# Purpose:  Take an arcmoss file and create a STAMPS model text file.
#
#
#
# Original Coding:  Michael Paul Czechanski
#                   Software Engineer
#                   Terrain Simulations
#                   Logicon RDA
#                   Grafenwoehr, Germany
#
# Last Update:  27 January 1998
#

if ((!@ARGV[0]) || (!@ARGV[1]))
  {
  die "USAGE:  stamp_conv.prl <file_name> <output_file>\n";
  }

print "-------------------------------------\n\n\n";
print "   Now executing STAMP_CONV.PRL...\n\n\n";
print "-------------------------------------\n\n\n";

$file_name = @ARGV[0];
$out_file = @ARGV[1];

#  THE FOLLOWING FORMAT WILL BE USED FOR DATA OUTPUT
format THE_LINE =
@>>>@>>>>>>.@>@>>>>>>.@> 0.0    0.00 0.0 0.0 1.0 1.0 1.0 1 0 0 1 0 0 0.0
$stmp $xval  $xdec $yval $ydec
.
$coord_line = "+>$out_file";
open (THE_LINE, $coord_line) || die "can't create";
open (COORDS,"$file_name") || die "cannot open addresses";

#  This will chew up the universal label from the initial coverage
$line = <COORDS>;
$line = <COORDS>;

while ($line = <COORDS>) {
    ($junk, $stmp, $num) = $line =~ /\s*(\d+)\s*(\d+)\s*(\S+)/;
#    ($junk, $num) = $line =~ /\s*(\d+)\s*(\d+)/;

    $line = <COORDS>;
    ($xmain, $xdec, $ymain, $ydec) = $line =~ /\s*(\d+.)(\d+)\s*(\d+.)(\d+)/;
    $xval = $xmain - 690000;
    $yval = $ymain - 5500000;

    write THE_LINE;   # WRITE OUT FORMATTED DATA TO OUTFILE

}

#  CLOSE INPUT AND OUTPUT FILES

close(THE_LINE);
close(COORDS);

print "\n\n\nProgram complete!!!!!!!!!.\n\n\n";
```

2.2.12 s1k_fence.aml

PURPOSE: This program will convert an Arc/Info fence (line) coverage into an ADDWAMS file that can be imported into S1000 as a treeline feature.

HISTORY: This AML was written by M. Peri Cope on 1/29/97, and was modified by Michael Paul Czechanski in January 1998.

USAGE: The input coverage used by this command should reside in the Arc/Info workspace where the program is run, and it should be named following TerraSim JTS naming conventions (the coverage used by this AML should be called FENCES). Execute this AML from the Arc/Info prompt:

```
Arc: s1k_fence <outname>
```

where `<outname>` is the name of the final s1000-compatible file to be generated (without an extension - the extension `.net` will automatically be added). `<outname>` should not already exist.

SAMPLE OUTPUT: The figure below shows the definition of a single fence from an ADDWAMS treeline file:

```
2250        L\75\309\3\Fences                                          5
  695203.44  5511269.00
  695213.13  5511313.00
  695207.31  5511361.50
  659190.12  5511111.32
  695221.13  5512345.67
```

PROGRAM FLOW: The program adds attribute items needed by the ADDWAMS file and populates them for a generic fence description (a height of 3 meters and a texture-id of 75). To change these default settings, either edit the AML or the output text file. Finally, the line coverage is exported to an ADDWAMS format text file via the ARCMOSS command. The program will produce the following coverages and file during execution:

```
XFENCE_CVR
X<outname>F_CVR
X<outname>F.NET
```

mpcope

```
/* s1k_fence.aml
/* ------------------------------------------------------------
/*     Program:  S1K_FENCE.AML
/*     Purpose:  To create an ADDWAMS/MOSS semi-compliant file
/*                 that will can be loaded and used to build
/*                 S1000 line feature terrain.
/* ------------------------------------------------------------
/*     Called By:  none
/*     Calls Made:  none
/* ------------------------------------------------------------
/*     Arguments:  outname
/*     Globals:  xxc
/* ------------------------------------------------------------
/*     Inputs:  FENCES
/*     Outputs:  XFENCESF_CVR X%OUTNAME%F_CVR X%OUTNAME%F.NET
/* ------------------------------------------------------------
/*     History:  1/10/97 - Original coding - M. Peri Cope
/*     Terrain Simulations - Grafenwoehr, Germany
/*
/*     Modified by Michael Paul Czechanski to be used only with
/*     tree lines.
/*       Last Update: 1/13/98
/* ------------------------------------------------------------
&echo &on
/*&severity &error &routine bail

&args outname

&if [null %outname%] &then
  &return &inform Usage: S1K_FENCE <out_file_name>

&if [exists x%outname%f_cvr -cover] &then
  &do
    &type Deleting x%outname%f_cvr...
    kill x%outname%f_cvr all
  &end

&if [exists x%outname%f.net -file] &then
  &do
    &type Now deleting x%outname%f.net...
    &s dump [delete x%outname%f.net -file]
    &type
  &end

&type S1K_FENCE.AML will now process the FENCES coverage found
       within
&type this workspace...
&type
&type

/*  Program will check for conflicting temporary files and
       delete them.
&type Checking for temporary files...
&type
&type


&s xxc = x%outname%f_cvr

&if [exists x%outname%f.net -file] &then
  &do
    &type Now deleting x%outname%f.net...
    &s dump [delete x%outname%f.net -file]
    &type
  &end
&if [exists %xxc% -cover] &then
  &do
    kill %xxc% all
    &type
  &end
&type

/* Program will make copies of coverages and modify them to
       create
/* an appropriate attribute item for ARCMOSS.

&if [exists %outname% -cover] &then
    &do
      copy %outname% %xxc%
      build %xxc% line
/*      additem %xxc%.aat %xxc%.aat WID2 2 2 i
      additem %xxc%.aat %xxc%.aat INDX 1 1 c
      additem %xxc%.aat %xxc%.aat TXTID 3 3 i
      additem %xxc%.aat %xxc%.aat PATID 3 3 i
      additem %xxc%.aat %xxc%.aat HYT 3 3 i
      additem %xxc%.aat %xxc%.aat DES 20 20 c

/* Program populates database for each cover.

      &call addf
      &call treeln
    &end

&call amoss

&return



/*************************************************************
/* addf routine
/*************************************************************
&routine addf
tables
sel %xxc%.aat
```

```
move 'T' TO INDX
quit
&return


/*************************************************************
/* treeln routine
/*************************************************************
&routine treeln
  tables
    sel %xxc%.aat
    calc TXTID = 75
    calc PATID = 390
    calc HYT = 3
    move 'Fences' TO DES
    quit

&return


/*************************************************************
/* amoss routine
/*************************************************************
&routine amoss

&if [exists %xxc%.net -file] &then &s junk [delete %xxc%.net -
       file]
arcmoss %xxc% %xxc%.net utm INDX\TXTID\PATID\HYT\DES # line

&return


/*************************************************************
/* bail routine
/*************************************************************
&routine bail
&severity &error &ignore
&severity &warning &ignore
&type An error has occured in s1000f.aml
&type Bailing out of s1000f.aml
&return; &return &error
```

2.2.13 s1k_land.aml

PURPOSE: This program will convert a Arc/Info land-cover (polygon) coverages into an ADDWAMS file that can be imported into S1000 as a polygon for texturing micro-terrain.

HISTORY: This AML was written by M. Peri Cope on 1/10/97, and was last updated on 1/23/98.

USAGE: The input coverage used by this command should reside in the Arc/Info workspace where the program is run, and it should be named following TerraSim JTS naming conventions. The coverages used by this AML should be called URBAN, LAKES, and SCRUB, and should be built polygon coverages, but they don't need any special attribute items. Execute this AML from the Arc/Info prompt:

```
Arc: s1k_land <outname>
```

where `<outname>` is the name of the final S1000-compatible file to be generated (without an extension - the extension `.net` will automatically be added). `<outname>` should not already exist.

SAMPLE OUTPUT: The figure below shows the definition of a single polygon from an ADDWAMS file created by this program:

```
2250        L\75\211\1\Residential                                    5
  695203.44  5511269.00
  695213.13  5511313.00
  695207.31  5511361.50
  659190.12  5511111.32
  695221.13  5512345.67
```

PROGRAM FLOW: The program adds attribute items needed by the ADDWAMS file and populates them as follows:

Coverage	Texture-id	PAT-id	Type	Description
lakes	43	211	3	Lake
scrub	102	211	2	Desert_Scrub
urban	75	211	1	Residential

To change these default settings, edit either the AML or the output text file. The polygon coverages are exported to an ADDWAMS format text file via the ARCMOSS command. The program will produce the following coverages and file during execution:

```
XURBAN_CVR
XSCRUB_CVR
X<outname>L_CVR
X<outname>L.NET
```

mpcope

```
/* s1k_land.aml
/* ------------------------------------------------------------
/*     Program:  S1K_LAND.AML
/*     Purpose:  To create an ADDWAMS/MOSS semi-compliant file
/*                 that will can be loaded and used to build
/*                 S1000 line feature terrain.
/* ------------------------------------------------------------
/*     Called By:  none
/*     Calls Made:  none
/* ------------------------------------------------------------
/*     Arguments:  outname
/*     Globals:  xxc
/* ------------------------------------------------------------
/*     Inputs:  URBAN SCRUB LAKES
/*     Outputs: XURBANL_CVR XSCRUBL_CVR XLAKESL_CVR
       X%OUTNAME%L_CVR
/*                X%OUTNAME%L.NET
/* ------------------------------------------------------------
/*     History:  1/10/97 - Original coding - M. Peri Cope
/*     Terrain Simulations - Grafenwoehr, Germany
/*
/*      Last Update:  1/23/98
/* ------------------------------------------------------------
&echo &on
/*&severity &error &routine bail

&args outname

&if [null %outname%] &then
  &return &inform Usage: S1K_LAND <out_file_name>

&if [exists x%outname%l_cvr -cover] &then
  &do
    &type Deleting x%outname%l_cvr...
    kill x%outname%l_cvr
  &end

&if [exists x%outname%l.net -file] &then
  &do
    &type Now deleting x%outname%l.net...
    &s dump [delete x%outname%l.net -file]
    &type
  &end


&type S1K_LAND.AML will now process all LAND coverages found
       within
&type this workspace...
&type
&type

/*  Program will check for conflicting temporary files and
       delete them.

&type Checking for temporary files...
&type
&type

&do cover &LIST urban scrub lakes
  &s xxc = x%cover%l_cvr

  &if [exists x%cover%l.net -file] &then
    &do
      &type Now deleting x%cover%l.net...
      &s dump [delete x%cover%l.net -file]
      &type
    &end
  &if [exists %xxc% -cover] &then
    &do
      kill %xxc%
      &type Killing %xxc%...
    &end
  &type

/* Program will make copies of coverages and modify them to
       create
/* an appropriate attribute item for ARCMOSS.

  &if [exists %cover% -cover] &then
    &do
      copy %cover% %xxc%
      additem %xxc%.pat %xxc%.pat INDX 1 1 c
      additem %xxc%.pat %xxc%.pat TXTID 3 3 i
      additem %xxc%.pat %xxc%.pat PATID 3 3 i
      additem %xxc%.pat %xxc%.pat TYP 1 1 i
      additem %xxc%.pat %xxc%.pat DES 20 20 c

/* Program populates database for each cover.

      &call addl
      &call %cover%
   &end
&end

&call combine   /* comment this out if one coverage is used
&call amoss

&return

/*************************************************************
/* addl routine
/*************************************************************
&routine addl
tables
sel %xxc%.pat
move 'L' TO INDX
```

```
quit
&return

/*************************************************************
/* urban routine
/*************************************************************
&routine urban
  tables
    sel %xxc%.pat
    calc TXTID = 75
    calc PATID = 211
    calc TYP = 1
    move 'Residential' TO DES
    quit

&return

/*************************************************************
/* scrub routine
/*************************************************************
&routine scrub
  tables
    sel %xxc%.pat
    calc TXTID = 102
    calc PATID = 211
    calc TYP = 2
    move 'Desert_Scrub' TO DES
    quit

&return

/*************************************************************
/* lakes routine
/*************************************************************
  &routine lakes
  tables
    sel %xxc%.pat
    calc TXTID = 43
    calc PATID = 211
    calc TYP = 3
    move 'Lake' TO DES
    quit

&return

/*************************************************************
/* combine routine
/*************************************************************
&routine combine

&s xcvr = x%outname%l_cvr

mapjoin %xcvr% poly all
  &do cov &LIST urban scrub lakes
    &if [exists x%cov%l_cvr -cover] &then x%cov%l_cvr
  &end

~
y
y

&return

/*************************************************************
/* amoss routine
/*************************************************************
&routine amoss

&s xcvr = x%outname%l_cvr
arcmoss %xcvr% x%outname%l.net utm INDX\TXTID\PATID\TYP\DES #
      poly

&return

/*************************************************************
/* bail routine
/*************************************************************
/*&routine bail
/*&severity &error &ignore
/*&severity &warning &ignore
/*&type An error has occured in s1000l.aml
/*&type Bailing out of s1000l.aml
/*&return; &return &error
```

2.2.14 s1k_net.aml

PURPOSE: This program will convert a Arc/Info line (roads, rivers, railroads) coverages into an ADDWAMS file that can be imported into S1000 as a network.

HISTORY: This AML was written by M. Peri Cope on 1/16/97, and was last updated on 1/21/98.

USAGE: The input coverage used by this command should reside in the Arc/Info workspace where the program is run, and it should be named following TerraSim JTS naming conventions. The coverages used by this AML should be called PAVED, HYDRO, LOOSE, TRACK, and RAIL, and should be built line coverages. Each coverage should have a special attribute describing the width of the feature in meters. (For PAVED, LOOSE, RAIL, and TRACK, the attribute is WTW. For Hydro, the attribute is WID.) Execute this AML from the Arc/Info prompt:

```
Arc: s1k_net <outname>
```

where `<outname>` is the name of the final S1000-compatible file to be generated (without an extension - the extension `.net` will automatically be added). `<outname>` should not already exist.

SAMPLE OUTPUT: The figure below shows what generated output will look like following this operation:

```
2250        N\15\791\3\2\Dirt_Road                                    5
  695203.44  5511269.00
  695213.13  5511313.00
  695207.31  5511361.50
  659190.12  5511111.32
  695221.13  5512345.67
```

PROGRAM FLOW: The program adds attribute items needed by the ADDWAMS file and populates them as follows:

Coverage	Texture-id	PAT-id	Type	Description
paved	91	790	1	Concrete_Road
hydro	70	802	5	Non_Perennial
loose	71	791	2	Gravel_Road
track	15	791	2	Dirt_Road
rail	15	790	0	Railroad

To change these default settings, edit either the AML or the output text file. The line coverages are exported to an ADDWAMS format text file via the ARCMOSS command. The program will produce the following coverages and file during execution:

```
XPAVED_CVR
XHYDRO_CVR
XLOOSE_CVR
XTRACK_CVR
XRAIL_CVR
X<outname>_CVR
<outname>.NET
```

mpcope

```
/* s1k_net.aml
/* ------------------------------------------------------------
/*     Program:  S1K_NET.AML
/*     Purpose:  To create an ADDWAMS/MOSS semi-compliant file
/*                 that can be loaded and used to build
/*                 S1000 line feature terrain.
/* ------------------------------------------------------------
/*     Called By:  none
/*     Calls Made:  none
/* ------------------------------------------------------------
/*     Arguments:  outname
/*     Globals:  xxc
/* ------------------------------------------------------------
/*     Inputs:  PAVED HYDRO LOOSE TRACK RAIL BRIDGE
/*     Outputs:  XPAVED_CVR XHYDRO_CVR XLOOSE_CVR XTRACK_CVR
/*                 XRAIL_CVR XBRIDGE_CVR X%OUTNAME%_CVR
/*                   X%OUTNAME%.NET
/* ------------------------------------------------------------
/*     History:  1/6/97 - Original coding - M. Peri Cope
/*     Terrain Simulations - Grafenwoehr, Germany
/*
/*      Last Updated:  1/21/98
/* ------------------------------------------------------------
&echo &on
/*&severity &error &routine bail

&args outname

&if [null %outname%] &then
  &return &inform Usage: S1K_NET <out_file_name>

&if [exists x%outname%_cvr -cover] &then
  &do
    &type Deleting x%outname%_cvr...
    kill x%outname%_cvr
  &end

&if [exists x%outname%.net -file] &then
  &do
    &type Now deleting x%outname%.net...
    &s dump [delete x%outname%.net -file]
    &type
  &end

&type S1K_NET.AML will now process all linear coverages found
       within
&type this workspace...
&type
&type

/*  Program will check for conflicting temporary files and
       delete them.
&type Checking for temporary files...
&type
&type

&do cover &LIST paved hydro loose track rail bridge
  &s xxc = x%cover%_cvr

  &if [exists %xxc% -cover] &then
    &do
      kill %xxc%
      &type
    &end
  &type

/* Program will make copies of coverages and modify them to
       create
/* an appropriate attribute item for ARCMOSS.

  &if [exists %cover% -cover] &then
    &do
      copy %cover% %xxc%
      additem %xxc%.aat %xxc%.aat WID2 2 2 i
      additem %xxc%.aat %xxc%.aat INDX 1 1 c
      additem %xxc%.aat %xxc%.aat TXTID 3 3 i
      additem %xxc%.aat %xxc%.aat PATID 3 3 i
      additem %xxc%.aat %xxc%.aat TYP 1 1 i
      additem %xxc%.aat %xxc%.aat DES 20 20 c
    &end

/* Program populates database for each cover.

  &if [exists %cover% -cover] &then
    &do
      &if %cover% = hydro &then
        &call round
      &if %cover% <> hydro &then
        &call round2
      &call addn
      &call %cover%
    &end

&end

&call combine
&call amoss

&return
/*************************************************************
/* round routine
/*************************************************************
&routine round
tables
sel %xxc%.aat
```

```
calc WID2 = WID
quit
&return

/*************************************************************
/* round2 routine
/*************************************************************
&routine round2
tables
sel %xxc%.aat
calc WID2 = WTW
quit
&return

/*************************************************************
/* addn routine
/*************************************************************
&routine addn
tables
sel %xxc%.aat
move 'N' TO INDX
quit
&return


/*************************************************************
/* paved routine
/*************************************************************
&routine paved
  tables
    sel %xxc%.aat
    calc TXTID = 91
    calc PATID = 790
    calc TYP = 1
    move 'Concrete_Road' TO DES
    quit
&return


/*************************************************************
/* hydro routine
/*************************************************************
&routine hydro
  tables
    sel %xxc%.aat
    calc TXTID = 70
    calc PATID = 802
    calc TYP = 5
    move 'Non_Perennial' TO DES
    quit
&return

/*************************************************************
/* loose routine
/*************************************************************
&routine loose
  tables
    sel %xxc%.aat
    calc TXTID = 71
    calc PATID = 791
    calc TYP = 2
    move 'Gravel_Road' TO DES
    quit
&return


/*************************************************************
/* track routine
/*************************************************************
&routine track
  tables
    sel %xxc%.aat
    calc TXTID = 15
    calc PATID = 791
    calc TYP = 2
    move 'Dirt_Road' TO DES
    quit
&return


/*************************************************************
/* rail routine
/*************************************************************
&routine rail
  tables
    sel %xxc%.aat
    calc TXTID = 15
    calc PATID = 790
    calc TYP = 0
    move 'Railroad' to DES
    quit
&return


/*************************************************************
/* bridge routine
/*************************************************************
&routine bridge
  tables
    sel %xxc%.aat
    calc TXTID = 000
    calc PATID = 000
    calc TYP = 0
    move 'bridgexxx' TO DES
    quit
&return
```

```
/*************************************************************
/* combine routine
/*************************************************************
&routine combine

&s xcvr = x%outname%_cvr

append %xcvr% line all
  &do cov &LIST paved hydro loose track rail bridge
    &if [exists x%cov%_cvr -cover] &then x%cov%_cvr
  &end

~
y
y

&return


/*************************************************************
/* amoss routine
/*************************************************************
&routine amoss

&s xcvr = x%outname%_cvr
arcmoss %xcvr% x%outname%.net utm INDX\TXTID\PATID\WID2\TYP\DES
       # line

&return


/*************************************************************
/* bail routine
/*************************************************************
/*&routine bail
/*&severity &error &ignore
/*&severity &warning &ignore
/*&type An error has occured in s1000.aml
/*&type Bailing out of s1000.aml
/*&return; &return &error
```

2.2.15 s1k_texture.perl

PURPOSE: This perl script creates a texture file called s1k.tlib.txt for use in creating a texture library in S1000.

HISTORY: This script was originally coded on 19 December 1997 by Tobi Sellekaerts. It is currently not useable as we are not able to pass texture libraries from S1000 to SimNet.

USAGE: The usage for this program is as follows:

```
italy% s1k_texture.perl
```

The program will prompt you to enter two pieces of information about each .tga image file you would like to include. The first is the name of the texture file without the .tga extension. The second is the size (width and height) of the image in pixels.

SAMPLE OUTPUT: The final page of this section shows a sample output from this program. The specifications for this file are taken from the S1000 users manual. All textures will be defined as wrapping in both the x and y direction, with no transparency. To change these settings, edit the text file after it has been created. The output file will always be called s1k.tlib.txt; you can change the name by moving or copying the file after it has been created.

PROGRAM FLOW: The following pages show the perl code for this program. The beginning steps for creating textures for S1000 / SimNet are:

First, select textures you would like to use in your SimNet database. Second, convert these textures into 32 bit Targa image files. Third, run this program. Fourth, put these image files in

```
/s1000/proj/<project>/image
```

Finally, follow the steps for creating a texture library as found in the S1000 user's manual.

tlss

```
#! /usr/local/bin/perl

# s1k_texture.perl
# ----------------------------------------------------------------
#      Program: s1k_texture.perl
#      Purpose: Creates a texture.tlib.txt file from image names
#        for creating a texture library for S1000.
#  ---------------------------------------------------------------
#      History:  29 December 1997 - Tobi L.S. Sellekaerts
#            steinbet@email.grafenwoehr.army.mil
#      Terrain Simulations - 7th ATC - Grafenwoehr, Germany
#  ---------------------------------------------------------------

open(OUT, ">s1k.tlib.txt");

print "The name of your output file will be s1k.tlib.txt\n";

# find out the name of the project in S1K
print "What is the name of your project?   ";
$project = <STDIN>;
chop($project);

print "\n";
print "You must make a new .tlib.txt file if you change your project name.\n";
print "\n";

print "All images must be in TARGA format.\n";
print "Type in the names of the images one at a time.\n";
print "Do not type the .tga extension!!!\n";
print "     Press return when finished.\n";

# printing out the header information
print OUT "# Version 1\n";
print OUT "# poly-texid map_type sw_map_size max_lod wrapx wrapy transbits color_filter\n";
print OUT "# ds_type        sw_map desc\n";
print OUT "\n";
print OUT "# poly-texid\n";
print OUT "# | map_type     sw_map_size\n";
print OUT "# | |            |   max_lod\n";
print OUT "# | |            |   |  wrapx\n";
print OUT "# | |            |   |  | wrapy\n";
print OUT "# | |            |   |  | | transbits\n";
print OUT "# | |            |   |  | | | color_filter\n";
print OUT "# | |            |   |  | | | |      ds_type\n";
print OUT "# | |            |   |  | | | |      |      trans_slope\n";
print OUT "# | |            |   |  | | | |      |      |   solid_transition\n";
print OUT "# | |            |   |  | | | |      |      |   | source_image
       sw_map                              desc\n";
print OUT "# | |            |   |  | | | |      |      |   | |
       |                                   |\n";


# requesting input of each of the image file names and writing
# the corresponding line to the .tlib.txt file
print "Image file name:";
$in = <STDIN>;
$count = 0;

while($in ne "\n") {
    chop($in);
    print "     Size (256, 128, 64, 32, 16, 8, 4, 2, 1):";
    $size = <STDIN>;
    chop($size);
    $file1 = "$project" . "\/image" . "\/" . "$in" . ".tga";
    $file2 = "$project" . "\/mip" . "\/" . "$in" . ".rgb";
    printf OUT "%3s", $count;
    print OUT " rgb        ";
    printf OUT "%3s %4s", $size, $size;
    print OUT " 1 1 0 none   linear 2   1 ";
    print OUT "$file1";
    $diff = 39 - length($file1);
    for($i = 1; $i < $diff; ++$i) {
```

```
        print OUT " ";
    }

    print OUT "$file2";
    $diff = 37 - length($file2);
    for($i = 1; $i < $diff; ++$i) {
       print OUT " ";
    }

    print OUT "\"             \"\n";
    ++$count;
    print "Image file name:";
    $in = <STDIN>;
    print "     Size (256, 128, 64, 32, 16, 8, 4, 2, 1):";
    $size = <STDIN>;
    chop($size);
}

close(OUT);

print "Remember to put your .tga files in /s1000/proj/$project/image !!!\n"
```

```
# Version 1
# poly-texid map_type sw_map_size max_lod wrapx wrapy transbits color_filter
# ds_type        sw_map desc

# poly-texid
# | map_type     sw_map_size
# | |            |   max_lod
# | |            |   |  wrapx
# | |            |   |  | wrapy
# | |            |   |  | | transbits
# | |            |   |  | | | color_filter
# | |            |   |  | | | |      ds_type
# | |            |   |  | | | |      |      trans_slope
# | |            |   |  | | | |      |      |   solid_transition
# | |            |   |  | | | |      |      |   | source_image                          sw_map                                desc
# | |            |   |  | | | |      |      |   | |                                     |                                     |
  0 rgb          64  64 1 1 0 none   linear 2   1 graf51/image/grass.tga               graf51/mip/grass.rgb                "             "
  1 rgb         128 128 1 1 0 none   linear 2   1 graf51/image/cement.tga              graf51/mip/cement.rgb               "             "
  2 rgb          64  64 1 1 0 none   linear 2   1 graf51/image/darkveg.tga             graf51/mip/darkveg.rgb              "             "
  3 rgb         128 128 1 1 0 none   linear 2   1 graf51/image/1460.tga                graf51/mip/1460.rgb                 "             "
  4 rgb         128 128 1 1 0 none   linear 2   1 graf51/image/water.tga               graf51/mip/water.rgb                "             "
  5 rgb         256 256 1 0 8 none   linear 2   1 graf51/image/conif.tga               graf51/mip/conif.rgb                "             "
```

2.2.16 s1k_treeln.aml

PURPOSE: This program will convert an Arc/Info treeline (line) coverage into an ADDWAMS file that can be imported into S1000 as a treeline.

HISTORY: This AML was written by M. Peri Cope on 1/16/97, and was last updated on 1/21/98.

USAGE: The input coverage used by this command should reside in the Arc/Info workspace where the program is run, and it should be named following TerraSim JTS naming conventions (the coverage used by this AML should be called TREELINE). Execute this AML from the Arc/Info prompt:

```
Arc: s1k_treeln <outname>
```

where `<outname>` is the name of the final S1000-compatible file to be generated (without an extension - the extension `.net` will automatically be added). `<outname>` should not already exist.

SAMPLE OUTPUT: The figure below shows what generated output will look like following this operation:

```
2250        L\29\390\20\Treeline                                        5
  695203.44  5511269.00
  695213.13  5511313.00
  695207.31  5511361.50
  659190.12  5511111.32
  695221.13  5512345.67
```

PROGRAM FLOW: The program adds attribute items needed by the ADDWAMS file and populates them for a generic treeline description (a height of 20 meters and a texture-id of 29). To change these default settings, either edit the AML or the output text file. Finally, the line coverage is exported to an ADDWAMS format text file via the ARCMOSS command. This AML is a variation of `s1k_fence.aml`, so the output files are similar. The program will produce the following coverages and file during execution:

```
XFENCE_CVR
X<outname>F_CVR
X<outname>F.NET
```

mpcope

```
/* s1k_treeln.aml
/* ------------------------------------------------------------
/*     Program:  S1K_TREELN.AML
/*     Purpose:  To create an ADDWAMS/MOSS semi-compliant file
/*                 that will can be loaded and used to build
/*                 S1000 line feature terrain.
/* ------------------------------------------------------------
/*     Called By:  none
/*     Calls Made:  none
/* ------------------------------------------------------------
/*     Arguments:  outname
/*     Globals:  xxc
/* ------------------------------------------------------------
/*     Inputs:  TREELN
/*     Outputs:  XTREELNT_CVR X%OUTNAME%T_CVR X%OUTNAME%T.NET
/* ------------------------------------------------------------
/*     History:  1/10/97 - Original coding - M. Peri Cope
/*     Terrain Simulations - Grafenwoehr, Germany
/*
/*     Modified by Michael Paul Czechanski to be used only with
/*     tree lines.
/*       Last Update: 1/19/98
/* ------------------------------------------------------------
&echo &on
/*&severity &error &routine bail

&args outname

&if [null %outname%] &then
  &return &inform Usage: S1K_TREELN <out_file_name>

&if [exists x%outname%t_cvr -cover] &then
  &do
    &type Deleting x%outname%t_cvr...
    kill x%outname%t_cvr all
  &end

&if [exists x%outname%t.net -file] &then
  &do
    &type Now deleting x%outname%t.net...
    &s dump [delete x%outname%t.net -file]
    &type
  &end

&type S1K_FENCE.AML will now process the FENCES coverage found
       within
&type this workspace...
&type
&type

/*  Program will check for conflicting temporary files and
       delete them.
&type Checking for temporary files...
&type
&type


&s xxc = xtreelnt_cvr

&if [exists x%outname%f.net -file] &then
  &do
    &type Now deleting x%outname%f.net...
    &s dump [delete x%outname%f.net -file]
    &type
  &end
&if [exists %xxc% -cover] &then
  &do
    kill %xxc% all
    &type
  &end
&type

/* Program will make copies of coverages and modify them to
       create
/* an appropriate attribute item for ARCMOSS.

&if [exists treeln -cover] &then
    &do
      copy treeln %xxc%
/*      additem %xxc%.aat %xxc%.aat WID2 2 2 i
      additem %xxc%.aat %xxc%.aat INDX 1 1 c
      additem %xxc%.aat %xxc%.aat TXTID 3 3 i
      additem %xxc%.aat %xxc%.aat PATID 3 3 i
      additem %xxc%.aat %xxc%.aat HYT 3 3 i
      additem %xxc%.aat %xxc%.aat DES 20 20 c

/* Program populates database for each cover.

      &call addf
      &call treeln

      &do covaat &LIST TRANSID OFC4 EXS LEN MCP MED WTC WTW
       BLANK NPNTS

           &s here .FALSE.
           tables
               &s loop [show columns info xtreelnt_cvr.aat]
               &do item &list %loop%
                    &if %item% = %covaat% &then
                        &s here .TRUE.
               &end
            quit

            &if %here% &then
            dropitem xtreelnt_cvr.aat xtreelnt_cvr.aat %covaat%
```

```
      &end
    &end

&call amoss

&return


/*************************************************************
/* addf routine
/*************************************************************
&routine addf
tables
sel %xxc%.aat
move 'T' TO INDX
quit
&return

/*************************************************************
/* treeln routine
/*************************************************************
&routine treeln
  tables
    sel %xxc%.aat
    calc TXTID = 29
    calc PATID = 390
    calc HYT = 20
    move 'treeline' TO DES
    quit

&return

/*************************************************************
/* amoss routine
/*************************************************************
&routine amoss

&s xcvr = xtreelnt_cvr
arcmoss %xcvr% x%outname%t.net utm INDX\TXTID\PATID\HYT\DES #
       line

&return

/*************************************************************
/* bail routine
/*************************************************************
/*&routine bail
/*&severity &error &ignore
/*&severity &warning &ignore
/*&type An error has occured in s1000f.aml
/*&type Bailing out of s1000f.aml
/*&return; &return &error
```

2.2.17 s1k_wtr.aml

PURPOSE: This program will convert an Arc/Info hydro (polygon) coverage into an ADDWAMS file that can be imported into S1000 as a feature for clipping micro-terrain.

HISTORY: This AML was written by M. Peri Cope on 9/25/97. It is not currently used as we can't get the micro-terrain clipping feature in S1000 to work. To get around this, we integrate the polygonal hydro features into the TIN before importing the TIN into S1000.

USAGE: The input coverage used by this command should reside in the Arc/Info workspace where the program is run, and it should be named following TerraSim JTS naming conventions (the coverage used by this AML should be called LAKES). Execute this AML from the Arc/Info prompt:

```
Arc: s1k_wtr <outname>
```

where `<outname>` is the name of the final S1000-compatible file to be generated (without an extension - the extension `.net` will automatically be added). `<outname>` should not already exist.

SAMPLE OUTPUT: The figure below shows what generated output will look like following this operation:

```
2250        C\131\29\1122\25\shoreline                                  5
  695203.44  5511269.00
  695213.13  5511313.00
  695207.31  5511361.50
  659190.12  5511111.32
  695221.13  5512345.67
```

PROGRAM FLOW: The program adds attribute items needed by the ADDWAMS file and populates them for a generic shoreline description. To change these default settings, either edit the AML or the output text file. The polygon coverage is exported to an ADDWAMS format text file via the ARCMOSS command. The program will produce the following coverages and file during execution:

```
XWTRC_CVR
X<outname>W_CVR
X<outname>.wtr
xhyrdow.wtr
xwaterw.wtr
```

mpcope

```
/* s1k_wtr.aml
/* ----------------------------------------------------------
/*     Program:  S1K_WTR.AML
/*     Purpose:  To create an ADDWAMS/MOSS semi-compliant file
/*                 that can be loaded and used to build
/*                 S1000 water feature terrain.
/* ----------------------------------------------------------
/*     Called By:  none
/*     Calls Made:  none
/* ----------------------------------------------------------
/*     Arguments:  outname
/*     Globals:  xxc
/* ----------------------------------------------------------
/*     Inputs:  PAVED HYDRO LAKES
/*     Outputs:  XHYDRO_CVR XAKES_CVR
/* ----------------------------------------------------------
/*     History:  9/25/97 - Original coding - M. Peri Cope
/*     Terrain Simulations - Grafenwoehr, Germany
/*
/*      Last Updated:  9/25/97
/* ----------------------------------------------------------
&echo &on
&severity &error &routine bail

&args outname

&if [null %outname%] &then
  &return &inform Usage: S1K_WTR <out_file_name>

/*  CHECK TO SEE IF x%OUTNAME%w_cvr EXISTS AND DELETE

&if [exists x%outname%w_cvr -cover] &then
  &do
    $type Deleting x%outname%w_cvr...
    kill x%outname%w_cvr
  &end

/*  CHECK TO SEE IF x%OUTNAME%.wtr EXISTS AND DELETE

&if [exists x%outname%.wtr -file] &then
  &do
    &type Now deleting x%outname%.wtr...
    &s dump [delete x%outname%.wtr -file]
    &type
  &end

&type S1K_WTR.AML will now process all water coverages found
       within
&type this workspace...
&type
&type

/*  CHECK FOR CONFLICTING TEMPORARY FILES AND DELETE THEME

&type Checking for temporary files...
&type
&type

&do cover &LIST hydro lakes
  &s xxc = x%cover%w_cvr

  &if [exists %xxc% -cover] &then
    &do
      kill %xxc%
      &type Killing %xxc%...
    &end
  &type

/*  MAKE COPIES OF COVERAGES AND MODIFY THEM TO CREATE
/*  AN APPROPRIATE ATTRIBUTE ITEM FOR ARCMOSS

  &if %cover% = hydro &then
     &do
       & call hydro_aat_set
     &end
  &else
     &do
       & call lakes_pat_set
     &end
 &end

&call combine

&return


/*************************************************************
/* hydro_aat_set routine
/*************************************************************
&routine hydro_aat_set

  &if [exists %cover% -cover] &then
    &do
      copy %cover% %xxc%
      additem %xxc%.aat %xxc%.aat WID2 2 2 i
      additem %xxc%.aat %xxc%.aat INDX 1 1 c
      additem %xxc%.aat %xxc%.aat TXTID 3 3 i
      additem %xxc%.aat %xxc%.aat PATID 3 3 i
      additem %xxc%.aat %xxc%.aat TYP 1 1 i
      additem %xxc%.aat %xxc%.aat DES 20 20 c

      tables
      sel %xxc%.aat
      calc WID2 = WID
      move 'N' TO INDX
      calc TXTID = 125
```

```
      calc PATID = 802
      calc TYP = 5
      move 'Non_Perennial' TO DES
      quit

      &do covaat &LIST HYDROID OFC5 WID BMC DVL DVR DW1 GW4
       BLANK NPNTS
        dropitem xhyrdrow_cvr.aat xhydrow_cvr.aat %covaat%
      &end

      arcmoss %xxc% x%cover%.wtr utm
       INDX\TXTID\PATID\WID2\TYP\DES # line

    &end

&return


/*************************************************************
/* lakes_pat_set
/*************************************************************
&routine lakes_pat_set

  &if [exists %cover% -cover] &then
    &do
      copy %cover% %xxc%
      additem %xxc%.pat %xxc%.pat INDX 1 1 c
      additem %xxc%.pat %xxc%.pat TXTID 3 3 i
      additem %xxc%.pat %xxc%.pat PATID 3 3 i
      additem %xxc%.pat %xxc%.pat TYP 1 1 i
      additem %xxc%.pat %xxc%.pat DES 20 20 c

      tables
      sel %xxc%.pat
      move 'L' TO INDX
      calc TXTID = 255
      calc PATID = 211
      calc TYP = 1
      move 'Lake' TO DES
      quit

      &do covpat &LIST TEMP NPNTS BLANK VEGID2 VEGID OFC3 BUD
       CCR DRW HLB ~
        SD1 SD2 STR TS1 TS2 VGC VH1 VR1 VR2 VR3 VS1 VS2 VS3 VW1
       VW2 VW3 WTR
        dropitem xlakesw_cvr.pat xlakesw_cvr.pat %covpat%
      &end

      arcmoss %xxc% x%cover%.wtr utm INDX\TXTID\PATID\TYP\DES #
       poly

    &end

&return


/*************************************************************
/* combine routine
/*************************************************************
&routine combine

&sys cat xhydrow.wtr xlakesw.wtr > x%outname%.wtr

&return
```

2.2.18 s1kc.prl

PURPOSE: This program will prompt the user to supply a series of input data that will be used to create a `<name>.s1k` control file. This file can then be used with the `s1kperfly` program to view and navigate through terrain databases that have been created in S1000. An SGI version exists called **s1kc_sgi.prl** - the only difference is the perl binary reference in the first line.

HISTORY: This program was originally coded in November 1997 by Peri Cope, and last updated 12/2/97.

USAGE: To run this program, first navigate to the directory where you would like the control file to be written. Next, at the shell prompt, issue the following command:

```
host% s1kc.prl output_file_name_root
```

where `output_file_name_root` is the root name of the control file you wish to create. For example, if you want to create a control file called `graf25.s1k`, enter

```
host% s1kc.prl graf25
```

The program will append the `.s1k` extension to the name you enter.

SAMPLE OUTPUT: The text below shows the sample output for a simple control file.

```
PROJECT graf24
S1KPROJ /net/belize/mike/is/crazy/s1000
ASSEMBLY_PREFIX graf24
UTM
ORIGIN 1000 1000
```

The text below shows the sample output for an advanced control file.

```
S1KPROJ /data4/s1000/proj/
PROJECT bangui1
ASSEMBLY_PREFIX bangui1
SEARCH_WINDOW 224000 478000 240000 489000
ORIGIN 224000 478000
UTM
LEAF_SIZE_MAX 500
MODEL_MAX_DISTANCE 25000
MULTISAMPLE
VIEWING_MODE OTW
DEFAULT_TERRAIN_COLOR 0.60 0.80 0.50 0.00
DEFAULT_NETWORK_COLOR 0.20 0.20 0.20 0.00
CANOPY_COLOR 0.00 0.40 0.10 0.00
TREELINE_COLOR 0.00 0.35 0.00 0.00
PAVED_SURFACE_COLOR 0.10 0.10 0.10 0.00
PACKED_SOIL_COLOR 0.15 0.25 0.05 0.00
SANDY_SOIL_COLOR 0.00 0.25 0.05 0.00
PASSABLE_H2O_COLOR 0.00 0.00 0.25 0.00
IMPASSABLE_H2O_COLOR 0.00 0.00 0.35 0.00
```

PROGRAM FLOW: As mentioned, this program will prompt the user to supply various pieces of input data that are required to create a new control file. The program also offers the option of creating simple or advanced control files. In most cases, a simple control file is all that will be needed to view and navigate a terrain database. In some instances, however, it may be necessary to specify further controls on the way the database is displayed. To specify these controls, use this program to create an advanced control file. For more information on eaach of these variables and their specific meaning in relation to a given database, please see the attached Texas Instruments documentation. (**Note**: With minor exceptions, this program does not validate data input by the user. In other words, the user is responsible for ensuring that data entered will be correct for the final control file.)

mpcope

```
#
#  Template for creating S1K Loader control files.
#
#  Author: Todd Pravata @ Texas Instruments 8/94
#  Modified to handle assigned polygon colors - Todd Pravata 11/94
#  Modified to handle MULTISAMPLE, VTX_NORMALS - Todd Pravata 1/95
#  Modified to handle TERRAIN_MODEL, ENABLE_PAGING - Todd Pravata, 9/95
#
#  PROJECT <project_name>
#    Mandatory. Name of the S1000 project to be loaded.  This directory should exist
#    under the directory pointed to by S1KPROJ.
#
#  S1KPROJ <project_directory>
#    Optional.  Specifies the path to the S1000 project directory (do not include
#    the project name).  The default is specified via environment variable S1KPROJ.
#
#  ASSEMBLY_PREFIX <assembly_prefix>
#    Optional.  Name of the assembly within the project to be loaded.  The
#    default value is the same as the PROJECT.
#
#  SEARCH_WINDOW <xmin> <ymin> <xmax> <ymax>
#    Optional.  Coordinates of the area to be extracted.  The default is the
#    entire database.
#
#  ORIGIN <x> <y>
#    Optional. User specified origin for the created scene graph.  The output
#    coordinates are relative to this origin.  Coordinates are assumed to given
#    in ASSY (S1000) unless UTM is selected.  The default is (0,0) in ASSY.
#
#  UTM
#    Optional. Keyword indicating that the coordinates are in UTM.  The default
#    is to interpret the input in S1000 ASSY coordinates.
#
#  INCLUDE_INACTIVE_POLYGONS
#    Optional.  Include polygons marked as inactive in the S1000 DB.  These are
#    typically those covered by canopies. The default is to exclude them.  Note
#    that these are only excluded if the S1000 database marked them as inactive.
#
#  MODEL_FILTER <filter_regexp>
#    Optional. Filter to pass model names thru.  This is used to limit the number
#    model references in the loaded database.  The default is to extract all
#    model referenes.  The filter regular expression syntax is the same as that
#    of the S1000 API (same as the editor ed/vi).
#
#  VTX_NORMALS
#    Optional.  Use S1000 database per vertex normals if available. Default
#    is to use per face normals or pfBuilder computed normals if none specified
#    in the S1000 database.
#
#  LEAF_SIZE_MAX <integer_size>
#    Optional.  Maximum size (in database units) of a leaf node in the quad tree.
#    The default is 1000.
#
#  TERRAIN_MAX_DISTANCE <distance>
#  CANOPY_MAX_DISTANCE <distance>
#  NETWORK_MAX_DISTANCE <distance>
#  TREELINE_MAX_DISTANCE <distance>
#    Optional.  Maximum viewing distance (LOD) for S1000 terrain and feature
#    geometry.  S1000 databases do not provide levels of detail for these
#    database components.  This directive allows the creation of an
#    "all-or-nothing" level of detail using the specified distance as the
#    range.  The default is to not create levels of detail for these database
#    components.
#
#  STAMP_MAX_DISTANCE <distance>
#    Optional.  Maximum viewing distance (LOD) for S1000 stamp geometry.
#    Default value is 1500.0 meters.  The S1000 database provides a mechanism
#    for specifying LOD ranges for stamp models, but they are typically not filled.
#
#  MODEL_MAX_DISTANCE <distance>
#    Optional.  Default maximum viewing distance (LOD) for S1000 static model
#    geometry if the S1000 database has not provided LOD distances.
```

```
#    Default value is 1500.0 meters.
#
#  SUPPRESS [polygon_type]
#    Optional.  polygon_type is one of TERRAIN, CANOPIES, TREELINES, or MODREFS.
#    Do not include the specified feature type in the output scene graph.  Default
#    is to include all of them.
#
#  CLONE_GEOMETRY
#    Optional.  Use pfClone on loaded model geometry rather than simple shared
#    instancing.  The default is to use shared instancing.  Note that this
#    option is required under 1.2 if you want to use pfFlatten on the returned
#    scene graph.
#
#  PFDECAL_BASE [ DISPLACE | STENCIL | FAST | HIGH_QUALITY ]
#    Optional.  Specifies the hardware mechanism to use in generating layered
#    geometry for the terrain.  The default is DISPLACE (see pfLayer).
#
#  MULTISAMPLE
#    Optional.  Use multisample transparency if it is available.  Default
#    is to not use it.
#
#  INCLUDE
#    Optional.  Include the referenced file.  This file should following S1K
#    control file syntax.
#
# TI_ONLY
#  VIEWING_MODE [ OTW | THERMAL | BOTH ]
#    Optional.  Specify the texture set to be using when extracting polygons
#    from the s1000 data base.  OTW alone creates a scene graph with an
#    "out-the-window" or optical view.  THERMAL creates a scene graph
#    resembling a FLIR view.  BOTH will provide a simulation which contains
#    both OTW and THERMAL views.  Set the channel's draw traversal mask to
#    select the view for the channel (see pfs1k.h).  The default is OTW.
#
# TI_ONLY
#  DEFAULT_TERRAIN_COLOR   <otw_rgba> [<thermal_rgba>]
#  DEFAULT_NETWORK_COLOR   <otw_rgba> [<thermal_rgba>]
#  CANOPY_COLOR            <otw_rgba> [<thermal_rgba>]
#  TREELINE_COLOR          <otw_rgba> [<thermal_rgba>]
#  RAILROAD_COLOR          <otw_rgba> [<thermal_rgba>]
#  PAVED_SURFACE_COLOR     <otw_rgba> [<thermal_rgba>]
#  PACKED_SOIL_COLOR       <otw_rgba> [<thermal_rgba>]
#  SANDY_SOIL_COLOR        <otw_rgba> [<thermal_rgba>]
#  PASSABLE_H2O_COLOR      <otw_rgba> [<thermal_rgba>]
#  IMPASSABLE_H2O_COLOR    <otw_rgba> [<thermal_rgba>]
#    Optional.  Red, green, blue, and alpha values for both out-the-window and
#    thermal views for textured polygons (S1000 specifies either a texture or
#    a color for a polygon - not both).  The values should be between 0.0 and
#    1.0.  The default is (1.0, 1.0, 1.0, 1.0).  The DEFAULT_TERRAIN_COLOR and
#    DEFAULT_NETWORK_COLOR is used on those terrain and network polygons for
#    which mobility type is not specified (ie. MOBILITY_TYPE_NONE).
#
# TI_ONLY
#  DEFAULT_TERRAIN_AD   <otw_ad>
#  DEFAULT_NETWORK_AD   <otw_ad>
#  CANOPY_AD            <otw_ad>
#  TREELINE_AD          <otw_ad>
#  RAILROAD_AD          <otw_ad>
#  PAVED_SURFACE_AD     <otw_ad>
#  PACKED_SOIL_AD       <otw_ad>
#  SANDY_SOIL_AD        <otw_ad>
#  PASSABLE_H2O_AD      <otw_ad>
#  IMPASSABLE_H2O_AD    <otw_ad>
#    Optional.  Ambient, diffuse values for the out-the-window view (thermal is
#    not lighted) for all polygons according to the S1000 mobility type.  The
#    values should be between 0.0 and 1.0.  The default values are taken from the
#    S1000 database.
#
# TI_ONLY
#  DB_MODEL_FILE <filename>
#    Optional.  This file defines an alternate set of models and/or textures to
#    be used in the place of the original S1000 models.  The format of this
```

```
#    file is specified in template.mdl.  Note that the loader uses LoadFile to
#    load models (use pfFilePath for alternate model locations).
#
# TI_ONLY
#  TERRAIN_MODEL <filename>
#    Optional.  This file (in one of the known Performer formats) defines the
#    terrain polyons used for planting features (so we don't have trees that
#    float).
#
# TI_ONLY
#  ENABLE_PAGING
#    Optional. Signals to the loader that the paging facilities will be used.
#    Initial load is then limited to shared objects (models, trees, etc).
#
```

```
#!/usr/local/bin/perl
# [s1kc.prl] PERL script
#
# Purpose:  This program will prompt the user for the information required to
#           to generate an S1K control file that will be used to view one of
#           our S1000 terrain databases in S1KPERFLY.
#
# Original coding:  M. Peri Cope
#                   Software Engineer
#                   Logicon RDA
#                   Grafenwoehr, Germany
#
#-----------------------------------------------------------------------------
# Arguments:  out_ctrl_file_name
#-----------------------------------------------------------------------------
#
# Last update:  12/2/97
#
#

if (!@ARGV[0])
  {
    print "USAGE: s1kc.prl <out_ctrl_file_name>\n";
    die;
  }

print "----------------------------------\n\n\n";

print "  Now executing S1KC.PRL...\n\n\n";

print "----------------------------------\n\n\n";

$out_ctrl_file_name = @ARGV[0];

&open_write;


# ----------- PROMPT USER FOR TYPE OF CONTROL FILE TO CREATE ---------->
# ------------------------------------------------------------------------>

print "\nWhich type of control file would you like to create?\n\n";

print "\t1\t- SIMPLE\n";
print "\t2\t- ADVANCED\n";
print "\t3\t- (More info please)\n\n";

print "Enter the correct choice (1-3):  ";

$choice = <STDIN>;
chop($choice);  # --- REMOVE \r FROM END OF $choice
$choice *= 1;  # --- CAST $choice AS IMPLICIT INTERGER



if ($choice == 1) # --- USER WANTS TO CREATE SIMPLE CONTROL FILE
  {
    &simple_ctrl;
    &simple_write;
    &house_keep;
  }
elsif ($choice == 2) # --- USER WANTS TO CREATE ADVANCED CONTROL FILE
  {
    &simple_ctrl;
    &adv_ctrl;
    &simple_write;
    &adv_write;
    &house_keep;
  }
elsif ($choice == 3) # --- USER WANTS FURTHER CLARIFICATION
  {
    &help_ctrl;
  }
else # --- USER DOESN'T KNOW WHAT THEY WANT
  {
    die "\n\n\nERROR: Invalid response.  Please start again.\n";
  }


# ------------------- DEFINE PRIMARY SUBROUTINES HERE ---------------->
# ------------------------------------------------------------------------>
```

```
#-----------------------------------
# SUBROUTINE open_write
#-----------------------------------
sub open_write
  {

    # ---- SET OUTPUT CONTROL FILE PATH VARIABLE ----->
    $write_File = "$out_ctrl_file_name.s1k";

    if (-e $write_File) # REMOVE OUTPUT CONTROL FILE IF IT ALREADY EXISTS
      {
         print ('Overwriting ', $write_File, '.s1k',  '.', "\n\n");
         system('rm ', $write_File, '.s1k');
      }

    $out_File = "+>$write_File";  # OPEN OUTPUT FILE FOR WRITING
    open(OutFile, $out_File) || die "Could not open output file for writing.\n";



#-----------------------------------
# SUBROUTINE simple_ctrl
#-----------------------------------
sub simple_ctrl
  {

     # ------- GET PROJECT PATH FROM USER ------->

     print "\n\n1)  Enter the full pathway to the S1000 project:\n\n";
     $proj_path = <STDIN>;
     chop($proj_path);

     if ($proj_path)
       {
         $simple{'S1KPROJ'} = $proj_path;
       }

     # ------- GET PROJECT NAME FROM USER ------->

     print "\n\n2)  Enter the S1000 project name (i.e. graf24):  ";
     $proj_name = <STDIN>;
     chop($proj_name);

     if ($proj_name)
       {
         $simple{'PROJECT'} = $proj_name;
       }

     # ------- GET ASSEMBLY PREFIX FROM USER ------->

     print "\n\n3)  Enter the assembly file prefix:  ";
     $assem_prfx = <STDIN>;
     chop($assem_prfx);

     if ($assem_prfx)
       {
         $simple{'ASSEMBLY_PREFIX'} = $assem_prfx;
       }


     # ------- GET COORDINATE BASE FROM USER ------->

     print "\n\n4)  Would you like to specify base coordinates in UTM? (y or n):  ";
     until (($utm eq "y") || ($utm eq "n"))
       {
         $utm = <STDIN>;
         chop($utm);
       }

     if ($utm eq "y")
       {
         $simple{'UTM'} = "UTM";
       }


     # ------- GET SEARCH WINDOW FROM USER ------->

     print "\n\n5)  Would you like to specify an initial search window? (y or n):  ";
     until (($sw_yn eq "y") || ($sw_yn eq "n"))
       {
```

```
        $sw_yn = <STDIN>;
        chop($sw_yn);
      }

    if ($sw_yn eq "y")
      {
        if ($utm eq "y")
          {

            # --- GET LL EASTING (IN UTM) FROM USER ---->

            print "\n\n5a)  Enter lower left Easting coordinate:  ";
            $ll_x = <STDIN>;
            chop($ll_x);  # --- REMOVE \r FROM END OF $ll_x
            $ll_x *= 1.0000000;  # --- CAST $ll_x AS IMPLICIT FLOAT WITH PRECISION

            # --- GET LL NORTHING (IN UTM) FROM USER ---->

            print "\n\n5b)  Enter lower left Northing coordinate:  ";
            $ll_y = <STDIN>;
            chop($ll_y);  # --- REMOVE \r FROM END OF $ll_y
            $ll_y *= 1.0000000;  # --- CAST $ll_y AS IMPLICIT FLOAT WITH PRECISION

            # --- GET UR EASTING (IN UTM) FROM USER ---->

            print "\n\n5c)  Enter upper right Easting coordinate:  ";
            $ur_x = <STDIN>;
            chop($ur_x);  # --- REMOVE \r FROM END OF $ur_x
            $ur_x *= 1.0000000;  # --- CAST $ur_x AS IMPLICIT FLOAT WITH PRECISION

            # --- GET UR NORTHING (IN UTM) FROM USER ---->

            print "\n\n5d)  Enter upper right Northing coordinate:  ";
            $ur_y = <STDIN>;
            chop($ur_y);  # --- REMOVE \r FROM END OF $ur_y
            $ur_y *= 1.0000000;  # --- CAST $ur_y AS IMPLICIT FLOAT WITH PRECISION

          }
        elsif ($utm eq "n")
          {

            # --- GET LL EASTING (IN METERS) FROM USER ---->

            print "\n\n5a)  Enter lower left X coordinate:  ";
            $ll_x = <STDIN>;
            chop($ll_x);  # --- REMOVE \r FROM END OF $ll_x
            $ll_x *= 1.0000000;  # --- CAST $ll_x AS IMPLICIT FLOAT WITH PRECISION

            # --- GET LL NORTHING (IN METERS) FROM USER ---->

            print "\n\n5b)  Enter lower left Y coordinate:  ";
            $ll_y = <STDIN>;
            chop($ll_y);  # --- REMOVE \r FROM END OF $ll_y
            $ll_y *= 1.0000000;  # --- CAST $ll_y AS IMPLICIT FLOAT WITH PRECISION

            # --- GET UR EASTING (IN METERS) FROM USER ---->

            print "\n\n5c)  Enter upper right X coordinate:  ";
            $ur_x = <STDIN>;
            chop($ur_x);  # --- REMOVE \r FROM END OF $ur_x
            $ur_x *= 1.0000000;  # --- CAST $ur_x AS IMPLICIT FLOAT WITH PRECISION

            # --- GET UR NORTHING (IN METERS) FROM USER ---->

            print "\n\n5d)  Enter upper right Y coordinate:  ";
            $ur_y = <STDIN>;
            chop($ur_y);  # --- REMOVE \r FROM END OF $ur_y
            $ur_y *= 1.0000000;  # --- CAST $ur_y AS IMPLICIT FLOAT WITH PRECISION

          }
    }

    # ---- GET DATABASE ORIGIN FROM USER ------->

    print "\n\n6)  Do you want to specify a starting origin? (y or n):  ";
    until (($or_yn eq "y") || ($or_yn eq "n"))
      {
        $or_yn = <STDIN>;
        chop($or_yn);
      }
```

```
      if ($or_yn eq "y")
        {
          print "\n\n6a)  Enter the origin X coordinate:  ";
          $or_x = <STDIN>;
          chop($or_x);  # --- REMOVE \r FROM END OF $or_x
          $or_x *= 1.0000000;  # --- CAST $or_x AS IMPLICIT FLOAT WITH PRECISION

          print "\n\n6b)  Enter the origin Y coordinate:  ";
          $or_y = <STDIN>;
          chop($or_y);  # --- REMOVE \r FROM END OF $or_y
          $or_y *= 1.0000000;  # --- CAST $or_y AS IMPLICIT FLOAT WITH PRECISION

        }

      # --- PUT DATABASE ORIGIN COORDINATES IN ARRAY ---->

      $simple{'ORIGIN_X'} = $or_x;
      $simple{'ORIGIN_Y'} = $or_y;

  }

#-------------------------------------
# SUBROUTINE adv_ctrl
#-------------------------------------
sub adv_ctrl
  {

  # ------- DOES THE USER WANT TO INCLUDE INACTIVE POLYGONS? ------->

  print "\n\n7)  Do you want to include inactive polygons? (y or n):  ";
  until (($ip_yn eq "y") || ($ip_yn eq "n"))
    {
      $ip_yn = <STDIN>;
      chop($ip_yn);
    }

  if ($ip_yn eq "y")
    {
      $adv{'INACTIVE_POLY'} = "INCLUDE_INACTIVE_POLYGONS";
    }

  # ------- DOES THE USER WANT TO INCLUDE A MODEL FILTER? ------->

  print "\n\n8)  Do you want to use a model filter? (y or n):  ";
  until (($mf_yn eq "y") || ($mf_yn eq "n"))
    {
      $mf_yn = <STDIN>;
      chop($mf_yn);
    }

  if ($mf_yn eq "y")
    {
      # ---- PROMPT USER FOR MODEL FILTER PATH ----->

      print "\n\n8a)  Specify the model filter path and filename:\n\n";
      $mf_name = <STDIN>;
      chop($mf_name);
      $adv{'MODEL_FILTER'} = $mf_name;
    }


  # ------- DOES THE USER WANT TO USE VERTEX NORMALS? ------->

  print "\n\n9)  Do you want to use vertex normals? (y or n):  ";
  until (($vn_yn eq "y") || ($vn_yn eq "n"))
    {
      $vn_yn = <STDIN>;
      chop($vn_yn);
    }

  if ($vn_yn eq "y")
    {
      $adv{'VTX_NORMS'} = "VTX_NORMALS";
    }


  # ------- DOES THE USER WANT TO SPECIFY A MAXIMUM LEAF SIZE? ------->

  print "\n\n10)  Do you want to specify a maximum leaf size? (y or n):  ";
  until (($ml_yn eq "y") || ($ml_yn eq "n"))
    {
```

```
      $ml_yn = <STDIN>;
      chop($ml_yn);
    }

  if ($ml_yn eq "y")
    {
      # ---- PROMPT USER FOR MAXIMUM LEAF SIZE ----->

      print "\n\n10a)  Specify the maximum leaf size (in meters):\n\n";
      $ml_size = <STDIN>;
      chop($ml_size);
      $adv{'LEAF_SIZE_MAX'} = $ml_size;
    }


  # ------- DOES THE USER WANT TO SPECIFY A MAXIMUM TERRAIN DISTANCE? ------->

  print "\n\n11)  Do you want to specify a maximum terrain distance? (y or n):  ";
  until (($mtd_yn eq "y") || ($mtd_yn eq "n"))
    {
      $mtd_yn = <STDIN>;
      chop($mtd_yn);
    }

  if ($mtd_yn eq "y")
    {
      # ---- PROMPT USER FOR MAXIMUM TERRAIN DISTANCE ----->

      print "\n\n11a)  Specify the maximum terrain distance (in meters):\n\n";
      $mtd_size = <STDIN>;
      chop($mtd_size);
      $adv{'TERRAIN_MAX_DISTANCE'} = $mtd_size;
    }


  # ------- DOES THE USER WANT TO SPECIFY A MAXIMUM CANOPY DISTANCE? ------->

  print "\n\n12)  Do you want to specify a maximum canopy distance? (y or n):  ";
  until (($mcd_yn eq "y") || ($mcd_yn eq "n"))
    {
      $mcd_yn = <STDIN>;
      chop($mcd_yn);
    }

  if ($mcd_yn eq "y")
    {
      # ---- PROMPT USER FOR MAXIMUM CANOPY DISTANCE ----->

      print "\n\n12a)  Specify the maximum canopy distance (in meters):\n\n";
      $mcd_size = <STDIN>;
      chop($mcd_size);
      $adv{'CANOPY_MAX_DISTANCE'} = $mcd_size;
    }


  # ------- DOES THE USER WANT TO SPECIFY A MAXIMUM NETWORK DISTANCE? ------->

  print "\n\n13)  Do you want to specify a maximum network distance? (y or n):  ";
  until (($mnd_yn eq "y") || ($mnd_yn eq "n"))
    {
      $mnd_yn = <STDIN>;
      chop($mnd_yn);
    }

  if ($mnd_yn eq "y")
    {
      # ---- PROMPT USER FOR MAXIMUM NETWORK DISTANCE ----->

      print "\n\n13a)  Specify the maximum network distance (in meters):\n\n";
      $mnd_size = <STDIN>;
      chop($mnd_size);
      $adv{'NETWORK_MAX_DISTANCE'} = $mnd_size;
    }


  # ------- DOES THE USER WANT TO SPECIFY A MAXIMUM TREELINE DISTANCE? ------->

  print "\n\n14)  Do you want to specify a maximum treeline distance? (y or n):  ";
  until (($mtld_yn eq "y") || ($mtld_yn eq "n"))
    {
      $mtld_yn = <STDIN>;
```

```
      chop($mtld_yn);
    }

  if ($mtld_yn eq "y")
    {
      # ---- PROMPT USER FOR MAXIMUM TREELINE DISTANCE ----->

      print "\n\n14a)  Specify the maximum treeline distance (in meters):\n\n";
      $mtld_size = <STDIN>;
      chop($mtld_size);
      $adv{'TREELINE_MAX_DISTANCE'} = $mtld_size;
    }


  # ------- DOES THE USER WANT TO SPECIFY A MAXIMUM STAMP DISTANCE? ------->

  print "\n\n15)  Do you want to specify a maximum stamp distance? (y or n):  ";
  until (($msd_yn eq "y") || ($msd_yn eq "n"))
    {
      $msd_yn = <STDIN>;
      chop($msd_yn);
    }

  if ($msd_yn eq "y")
    {
      # ---- PROMPT USER FOR MAXIMUM STAMP DISTANCE ----->

      print "\n\n15a)  Specify the maximum leaf size (in meters):\n\n";
      $msd_size = <STDIN>;
      chop($msd_size);
      $adv{'STAMP_MAX_DISTANCE'} = $msd_size;
    }


  # ------- DOES THE USER WANT TO SPECIFY A MAXIMUM MODEL DISTANCE? ------->

  print "\n\n16)  Do you want to specify a maximum model distance? (y or n):  ";
  until (($mmd_yn eq "y") || ($mmd_yn eq "n"))
    {
      $mmd_yn = <STDIN>;
      chop($mmd_yn);
    }

  if ($mmd_yn eq "y")
    {
      # ---- PROMPT USER FOR MAXIMUM MODEL DISTANCE ----->

      print "\n\n16a)  Specify the maximum model distance (in meters):\n\n";
      $mmd_size = <STDIN>;
      chop($mmd_size);
      $adv{'MODEL_MAX_DISTANCE'} = $mmd_size;
    }


  # ------- DOES THE USER WANT TO SUPPRESS A POLYGON TYPE? ------->

  print "\n\n17)  Do you want to suppress a particular polygon type? (y or n):  ";
  until (($spt_yn eq "y") || ($spt_yn eq "n"))
    {
      $spt_yn = <STDIN>;
      chop($spt_yn);
    }

  if ($spt_yn eq "y")
    {
      # ---- PROMPT USER FOR POLYGON TYPE TO SUPRESS ----->

      print "\n\n17a)  Specify polygon type to suppress:\n\n";
      print "\t1\t- TERRAIN\n";
      print "\t2\t- CANOPIES\n";
      print "\t3\t- TREELINES\n";
      print "\t4\t- MODREFS\n\n";

      print "Enter the correct choice (1-4):  ";
      until (($spt_choice > 0) && ($spt_choice < 5))
        {
          $spt_choice = <STDIN>;
        }
      chop($spt_choice);
      $spt_choice *= 1;
    }
```

```
if ($spt_choice == 1)
  {
    $adv{'SUPPRESS'} = 'TERRAIN';
  }
elsif ($spt_choice == 2)
  {
    $adv{'SUPPRESS'} = 'CANOPIES';
  }
elsif ($spt_choice == 3)
  {
    $adv{'SUPPRESS'} = 'TREELINES';
  }
else
  {
    $adv{'SUPPRESS'} = 'MODREFS';
  }


# ------- DOES THE USER WANT TO USE CLONE GEOMETRY? ------->

print "\n\n18)  Do you want to use clone geometry? (y or n):  ";
until (($cg_yn eq "y") || ($cg_yn eq "n"))
  {
    $cg_yn = <STDIN>;
    chop($cg_yn);
  }

if ($cg_yn eq "y")
  {
    $adv{'CLONE_GEOMETRY'} = "CLONE_GEOMETRY";
  }


# ------- DOES THE USER WANT TO USE A PF_DECAL BASE? ------->

print "\n\n19)  Do you want to use a PF_DECAL base? (y or n):  ";
until (($pfd_yn eq "y") || ($pfd_yn eq "n"))
  {
    $pfd_yn = <STDIN>;
    chop($pfd_yn);
  }

if ($pfd_yn eq "y")
  {
    # ---- PROMPT USER FOR SPECIFIC PF_DECAL BASE ----->

    print "\n\n19a)  Specify the PF_DECAL base type:\n\n";
    print "\t1\t- DISPLACE\n";
    print "\t2\t- STENCIL\n";
    print "\t3\t- FAST\n";
    print "\t4\t- HIGH QUALITY\n\n";

    print "Enter the correct choice (1-4):  ";
    until (($pfd_choice > 0) && ($pfd_choice < 5))
      {
        $pfd_choice = <STDIN>;
      }
    chop($pfd_choice);
    $pfd_choice *= 1;
  }

if ($pfd_choice == 1)
  {
    $adv{'PFDECAL_BASE'} = 'DISPLACE';
  }
elsif ($pfd_choice == 2)
  {
    $adv{'PFDECAL_BASE'} = 'STENCIL';
  }
elsif ($pfd_choice == 3)
  {
    $adv{'PFDECAL_BASE'} = 'FAST';
  }
else
  {
    $adv{'PFDECAL_BASE'} = 'HIGH_QUALITY';
  }


# ------- DOES THE USER WANT TO USE A MULTISAMPLE TRANSPARENCY? ------->
```

```
print "\n\n20)  Do you want to use a multisample transparency? (y or n):  ";
until (($mst_yn eq "y") || ($mst_yn eq "n"))
  {
    $mst_yn = <STDIN>;
    chop($mst_yn);
  }

if ($mst_yn eq "y")
  {
    $adv{'MULTISAMPLE'} = "MULTISAMPLE";
  }



# ------- DOES THE USER WANT TO INCLUDE ANOTHER CONTROL FILE? ------>

print "\n\n21)  Do you want to include another control file? (y or n):  ";
until (($iof_yn eq "y") || ($iof_yn eq "n"))
  {
    $iof_yn = <STDIN>;
    chop($iof_yn);
  }

if ($iof_yn eq "y")
  {
    # ----- PROMPT USER FOR CONTROL FILE PATH ----->

    print "\n\n21a)  Specify the control file path and filename:\n\n";
    $iof_name = <STDIN>;
    chop($iof_name);
    @adv{'INCLUDE'} = $iof_name;
  }



# ------- DOES THE USER WANT TO SPECIFY THE VIEWING MODE TEXTURE SET? ------->

print "\n\n22)  Do you want to specify a viewing mode texture set? (y or n):  ";
until (($tivm_yn eq "y") || ($tivm_yn eq "n"))
  {
    $tivm_yn = <STDIN>;
    chop($tivm_yn);
  }

if ($tivm_yn eq "y")
  {
    # ---- PROMPT USER FOR SPECIFIC VIEWING MODE TEXTURE SET ----->

    print "\n\n22a)  Specify the viewing mode texture set:\n\n";
    print "\t1\t- OTW\n";
    print "\t2\t- THERMAL\n";
    print "\t3\t- BOTH\n\n";

    print "Enter the correct choice (1-3):  ";
    until (($tivm_choice > 0) && ($tivm_choice < 4))
      {
        $tivm_choice = <STDIN>;
      }
    chop($tivm_choice);
    $tivm_choice *= 1;
  }

if ($tivm_choice == 1)
  {
    $adv{'TI_ONLY_VIEWING_MODE'} = 'OTW';
  }
elsif ($tivm_choice == 2)
  {
    $adv{'TI_ONLY_VIEWING_MODE'} = 'THERMAL';
  }
else
  {
    $adv{'TI_ONLY_VIEWING_MODE'} = 'BOTH';
  }

# ---- DOES THE USER WANT TO SPECIFY A DEFAULT TERRAIN COLOR? ----->

print "\n\n23)  Do you want to specify a default terrain color? (y or n):  ";
until (($dtc_yn eq "y") || ($dtc_yn eq "n"))
  {
```

```
      $dtc_yn = <STDIN>;
      chop($dtc_yn);
    }

  if ($dtc_yn eq "y")
    {

    # --- PROMPT USER TO ENTER RED LEVEL FOR DEFAULT TERRAIN COLOR ----->
    print "\n\n23a)  Enter a red level (0.00 - 1.00) for the default terrain color:  ";
    $dtc_rl = -5;
    until (($dtc_rl > -0.01) && ($dtc_rl < 1.01))
      {
        $dtc_rl = <STDIN>;
        chop($dtc_rl);
      }
    $adv {'DTC_RED'} = $dtc_rl;


    # --- PROMPT USER TO ENTER GREEN LEVEL FOR DEFAULT TERRAIN COLOR ----->
    print "\n\n23b)  Enter a green level (0.00 - 1.00) for the default terrain color:  ";
    $dtc_gl = -5;
    until (($dtc_gl > -0.01) && ($dtc_gl < 1.01))
      {
        $dtc_gl = <STDIN>;
        chop($dtc_gl);
      }
      $adv {'DTC_GREEN'} = $dtc_gl;


    # --- PROMPT USER TO ENTER BLUE LEVEL FOR DEFAULT TERRAIN COLOR ----->
    print "\n\n23c)  Enter a blue level (0.00 - 1.00) for the default terrain color:  ";
    $dtc_bl = -5;
    until (($dtc_bl > -0.01) && ($dtc_bl < 1.01))
      {
        $dtc_bl = <STDIN>;
        chop($dtc_bl);
      }
      $adv {'DTC_BLUE'} = $dtc_bl;


    # --- PROMPT USER TO ENTER ALPHA LEVEL FOR DEFAULT TERRAIN COLOR ----->
    print "\n\n23d)  Enter an alpha level (0.00 - 1.00) for the default terrain color:  ";
    $dtc_al = -5;
    until (($dtc_al > -0.01) && ($dtc_al < 1.01))
      {
        $dtc_al = <STDIN>;
        chop($dtc_al);
      }
      $adv {'DTC_ALPHA'} = $dtc_al;

    }


  # ---- DOES THE USER WANT TO SPECIFY A DEFAULT NETWORK COLOR? ----->

  print "\n\n24)  Do you want to specify a default network color? (y or n):  ";
  until (($dnc_yn eq "y") || ($dnc_yn eq "n"))
    {
      $dnc_yn = <STDIN>;
      chop($dnc_yn);
    }

  if ($dnc_yn eq "y")
    {

    # --- PROMPT USER TO ENTER RED LEVEL FOR DEFAULT NETWORK COLOR ----->
    print "\n\n24a)  Enter a red level (0.00 - 1.00) for the default network color:  ";
    $dnc_rl = -5;
    until (($dnc_rl > -0.01) && ($dnc_rl < 1.01))
      {
        $dnc_rl = <STDIN>;
        chop($dnc_rl);
      }
      $adv {'DNC_RED'} = $dnc_rl;


    # --- PROMPT USER TO ENTER GREEN LEVEL FOR DEFAULT NETWORK COLOR ----->
    print "\n\n24b)  Enter a green level (0.00 - 1.00) for the default network color:  ";
    $dnc_gl = -5;
    until (($dnc_gl > -0.01) && ($dnc_gl < 1.01))
      {
```

```
        $dnc_gl = <STDIN>;
        chop($dnc_gl);
      }
      $adv {'DNC_GREEN'} = $dnc_gl;


    # --- PROMPT USER TO ENTER BLUE LEVEL FOR DEFAULT NETWORK COLOR ----->
    print "\n\n24c)  Enter a blue level (0.00 - 1.00) for the default network color:  ";
    $dnc_bl = -5;
    until (($dnc_bl > -0.01) && ($dnc_bl < 1.01))
      {
        $dnc_bl = <STDIN>;
        chop($dnc_bl);
      }
      $adv {'DNC_BLUE'} = $dnc_bl;


    # --- PROMPT USER TO ENTER ALPHA LEVEL FOR DEFAULT NETWORK COLOR ----->
    print "\n\n24d)  Enter an alpha level (0.00 - 1.00) for the default network color:  ";
    $dnc_al = -5;
    until (($dnc_al > -0.01) && ($dnc_al < 1.01))
      {
        $dnc_al = <STDIN>;
        chop($dnc_al);
      }
      $adv {'DNC_ALPHA'} = $dnc_al;
    }



  # ---- DOES THE USER WANT TO SPECIFY A CANOPY COLOR? ----->

  print "\n\n25)  Do you want to specify a canopy color? (y or n):  ";
  until (($can_yn eq "y") || ($can_yn eq "n"))
    {
      $can_yn = <STDIN>;
      chop($can_yn);
    }

  if ($can_yn eq "y")
    {

    # --- PROMPT USER TO ENTER RED LEVEL FOR CANOPY COLOR ----->
    print "\n\n25a)  Enter a red level (0.00 - 1.00) for the canopy color:  ";
    $can_rl = -5;
    until (($can_rl > -0.01) && ($can_rl < 1.01))
      {
        $can_rl = <STDIN>;
        chop($can_rl);
      }
      $adv {'CAN_RED'} = $can_rl;


    # --- PROMPT USER TO ENTER GREEN LEVEL FOR CANOPY COLOR ----->
    print "\n\n25b)  Enter a green level (0.00 - 1.00) for the canopy color:  ";
    $can_gl = -5;
    until (($can_gl > -0.01) && ($can_gl < 1.01))
      {
        $can_gl = <STDIN>;
        chop($can_gl);
      }
      $adv {'CAN_GREEN'} = $can_gl;


    # --- PROMPT USER TO ENTER BLUE LEVEL FOR CANOPY COLOR ----->
    print "\n\n25c)  Enter a blue level (0.00 - 1.00) for the canopy color:  ";
    $can_bl = -5;
    until (($can_bl > -0.01) && ($can_bl < 1.01))
      {
        $can_bl = <STDIN>;
        chop($can_bl);
      }
      $adv {'CAN_BLUE'} = $can_bl;


    # --- PROMPT USER TO ENTER ALPHA LEVEL FOR DEFAULT TERRAIN COLOR ----->
    print "\n\n25d)  Enter an alpha level (0.00 - 1.00) for the canopy color:  ";
    $can_al = -5;
    until (($can_al > -0.01) && ($can_al < 1.01))
      {
        $can_al = <STDIN>;
```

```
        chop($can_al);
      }
      $adv {'CAN_ALPHA'} = $can_al;
    }


  # ---- DOES THE USER WANT TO SPECIFY A TREELINE COLOR? ----->

  print "\n\n26)  Do you want to specify a treeline color? (y or n):  ";
  until (($tlc_yn eq "y") || ($tlc_yn eq "n"))
    {
      $tlc_yn = <STDIN>;
      chop($tlc_yn);
    }

  if ($tlc_yn eq "y")
    {

    # --- PROMPT USER TO ENTER RED LEVEL FOR TREELINE COLOR ----->
    print "\n\n26a)  Enter a red level (0.00 - 1.00) for the treeline color:  ";
    $tlc_rl = -5;
    until (($tlc_rl > -0.01) && ($tlc_rl < 1.01))
      {
        $tlc_rl = <STDIN>;
        chop($tlc_rl);
      }
      $adv {'TLC_RED'} = $tlc_rl;


    # --- PROMPT USER TO ENTER GREEN LEVEL FOR TREELINE COLOR ----->
    print "\n\n26b)  Enter a green level (0.00 - 1.00) for the treeline color:  ";
    $tlc_gl = -5;
    until (($tlc_gl > -0.01) && ($tlc_gl < 1.01))
      {
        $tlc_gl = <STDIN>;
        chop($tlc_gl);
      }
      $adv {'TLC_GREEN'} = $tlc_gl;


    # --- PROMPT USER TO ENTER BLUE LEVEL FOR TREELINE COLOR ----->
    print "\n\n26c)  Enter a blue level (0.00 - 1.00) for the treeline color:  ";
    $tlc_bl = -5;
    until (($tlc_bl > -0.01) && ($tlc_bl < 1.01))
      {
        $tlc_bl = <STDIN>;
        chop($tlc_bl);
      }
      $adv {'TLC_BLUE'} = $tlc_bl;


    # --- PROMPT USER TO ENTER ALPHA LEVEL FOR TREELINE COLOR ----->
    print "\n\n26d)  Enter an alpha level (0.00 - 1.00) for the treeline color:  ";
    $tlc_al = -5;
    until (($ltc_al > -0.01) && ($tlc_al < 1.01))
      {
        $tlc_al = <STDIN>;
        chop($tlc_al);
      }
      $adv {'TLC_ALPHA'} = $tlc_al;
    }


  # ---- DOES THE USER WANT TO SPECIFY A RAILROAD COLOR? ----->

  print "\n\n27)  Do you want to specify a railroad color? (y or n):  ";
  until (($rrc_yn eq "y") || ($rrc_yn eq "n"))
    {
      $rrc_yn = <STDIN>;
      chop($rrc_yn);
    }

  if ($rrc_yn eq "y")
    {

    # --- PROMPT USER TO ENTER RED LEVEL FOR RAILROAD COLOR ----->
    print "\n\n27a)  Enter a red level (0.00 - 1.00) for the railroad color:  ";
    $rrc_rl = -5;
    until (($rrc_rl > -0.01) && ($rrc_rl < 1.01))
      {
        $rrc_rl = <STDIN>;
```

```
        chop($rrc_rl);
      }
      $adv {'RRC_RED'} = $rrc_rl;


    # --- PROMPT USER TO ENTER GREEN LEVEL FOR RAILROAD COLOR ----->
    print "\n\n27b)  Enter a green level (0.00 - 1.00) for the railroad color:  ";
    $rrc_gl = -5;
    until (($rrc_gl > -0.01) && ($rrc_gl < 1.01))
      {
        $rrc_gl = <STDIN>;
        chop($rrc_gl);
      }
      $adv {'RRC_GREEN'} = $rrc_gl;


    # --- PROMPT USER TO ENTER BLUE LEVEL FOR RAILROAD COLOR ----->
    print "\n\n27c)  Enter a blue level (0.00 - 1.00) for the railroad color:  ";
    $rrc_bl = -5;
    until (($rrc_bl > -0.01) && ($rrc_bl < 1.01))
      {
        $rrc_bl = <STDIN>;
        chop($rrc_bl);
      }
      $adv {'RRC_BLUE'} = $rrc_bl;


    # --- PROMPT USER TO ENTER ALPHA LEVEL FOR DEFAULT RAILROAD COLOR ----->
    print "\n\n27d)  Enter an alpha level (0.00 - 1.00) for the railroad color:  ";
    $rrc_al = -5;
    until (($rrc_al > -0.01) && ($rrc_al < 1.01))
      {
        $rrc_al = <STDIN>;
        chop($rrc_al);
      }
      $adv {'RRC_ALPHA'} = $rrc_al;
    }


  # ---- DOES THE USER WANT TO SPECIFY A PAVED SURFACE COLOR? ----->

  print "\n\n28)  Do you want to specify a paved surface color? (y or n):  ";
  until (($psc_yn eq "y") || ($psc_yn eq "n"))
    {
      $psc_yn = <STDIN>;
      chop($psc_yn);
    }

  if ($psc_yn eq "y")
    {

    # --- PROMPT USER TO ENTER RED LEVEL FOR PAVED SURFACE COLOR ----->
    print "\n\n28a)  Enter a red level (0.00 - 1.00) for the paved surface color:  ";
    $psc_rl = -5;
    until (($psc_rl > -0.01) && ($psc_rl < 1.01))
      {
        $psc_rl = <STDIN>;
        chop($psc_rl);
      }
      $adv {'PSC_RED'} = $psc_rl;


    # --- PROMPT USER TO ENTER GREEN LEVEL FOR PAVED SURFACE COLOR ----->
    print "\n\n28b)  Enter a green level (0.00 - 1.00) for the paved surface color:  ";
    $psc_gl = -5;
    until (($psc_gl > -0.01) && ($psc_gl < 1.01))
      {
        $psc_gl = <STDIN>;
        chop($psc_gl);
      }
      $adv {'PSC_GREEN'} = $psc_gl;


    # --- PROMPT USER TO ENTER BLUE LEVEL FOR PAVED SURFACE COLOR ----->
    print "\n\n28c)  Enter a blue level (0.00 - 1.00) for the paved surface color:  ";
    $psc_bl = -5;
    until (($psc_bl > -0.01) && ($psc_bl < 1.01))
      {
        $psc_bl = <STDIN>;
        chop($psc_bl);
      }
```

```
      $adv {'PSC_BLUE'} = $psc_bl;


    # --- PROMPT USER TO ENTER ALPHA LEVEL FOR DEFAULT TERRAIN COLOR ----->
    print "\n\n28d)  Enter an alpha level (0.00 - 1.00) for the paved surface color:  ";
    $psc_al = -5;
    until (($psc_al > -0.01) && ($psc_al < 1.01))
      {
        $psc_al = <STDIN>;
        chop($psc_al);
      }
      $adv {'PSC_ALPHA'} = $psc_al;
    }


  # ---- DOES THE USER WANT TO SPECIFY A PACKED SOIL COLOR? ----->

  print "\n\n29)  Do you want to specify a packed soil color? (y or n):  ";
  until (($sc_yn eq "y") || ($sc_yn eq "n"))
    {
      $sc_yn = <STDIN>;
      chop($sc_yn);
    }

  if ($sc_yn eq "y")
    {

    # --- PROMPT USER TO ENTER RED LEVEL FOR PACKED SOIL COLOR ----->
    print "\n\n29a)  Enter a red level (0.00 - 1.00) for the packed soil color:  ";
    $sc_rl = -5;
    until (($sc_rl > -0.01) && ($sc_rl < 1.01))
      {
        $sc_rl = <STDIN>;
        chop($sc_rl);
      }
      $adv {'SC_RED'} = $sc_rl;


    # --- PROMPT USER TO ENTER GREEN LEVEL FOR PACKED SOIL COLOR ----->
    print "\n\n29b)  Enter a green level (0.00 - 1.00) for the packed soil color:  ";
    $sc_gl = -5;
    until (($sc_gl > -0.01) && ($sc_gl < 1.01))
      {
        $sc_gl = <STDIN>;
        chop($sc_gl);
      }
      $adv {'SC_GREEN'} = $sc_gl;


    # --- PROMPT USER TO ENTER BLUE LEVEL FOR PACKED SOIL COLOR ----->
    print "\n\n29c)  Enter a blue level (0.00 - 1.00) for the packed soil color:  ";
    $sc_bl = -5;
    until (($sc_bl > -0.01) && ($sc_bl < 1.01))
      {
        $sc_bl = <STDIN>;
        chop($sc_bl);
      }
      $adv {'SC_BLUE'} = $sc_bl;


    # --- PROMPT USER TO ENTER ALPHA LEVEL FOR PACKED SOIL COLOR ----->
    print "\n\n29d)  Enter an alpha level (0.00 - 1.00) for the packed soil color:  ";
    $sc_al = -5;
    until (($sc_al > -0.01) && ($sc_al < 1.01))
      {
        $sc_al = <STDIN>;
        chop($sc_al);
      }
      $adv {'SC_ALPHA'} = $sc_al;
    }


  # ---- DOES THE USER WANT TO SPECIFY A SANDY SOIL COLOR? ----->

  print "\n\n30)  Do you want to specify a sandy soil color? (y or n):  ";
  until (($ssc_yn eq "y") || ($ssc_yn eq "n"))
    {
      $ssc_yn = <STDIN>;
      chop($ssc_yn);
    }
```

```
  if ($ssc_yn eq "y")
    {

    # --- PROMPT USER TO ENTER RED LEVEL FOR SANDY SOIL COLOR ----->
    print "\n\n30a)  Enter a red level (0.00 - 1.00) for the sandy soil color:  ";
    $ssc_rl = -5;
    until (($ssc_rl > -0.01) && ($ssc_rl < 1.01))
      {
        $ssc_rl = <STDIN>;
        chop($ssc_rl);
      }
      $adv {'SSC_RED'} = $ssc_rl;


    # --- PROMPT USER TO ENTER GREEN LEVEL FOR SANDY SOIL COLOR ----->
    print "\n\n30b)  Enter a green level (0.00 - 1.00) for the sandy soil color:  ";
    $ssc_gl = -5;
    until (($ssc_gl > -0.01) && ($ssc_gl < 1.01))
      {
        $ssc_gl = <STDIN>;
        chop($ssc_gl);
      }
      $adv {'SSC_GREEN'} = $ssc_gl;


    # --- PROMPT USER TO ENTER BLUE LEVEL FOR SANDY SOIL COLOR ----->
    print "\n\n30c)  Enter a blue level (0.00 - 1.00) for the sandy soil color:  ";
    $ssc_bl = -5;
    until (($ssc_bl > -0.01) && ($ssc_bl < 1.01))
      {
        $ssc_bl = <STDIN>;
        chop($ssc_bl);
      }
      $adv {'SSC_BLUE'} = $ssc_bl;


    # --- PROMPT USER TO ENTER ALPHA LEVEL FOR SANDY SOIL COLOR ----->
    print "\n\n30d)  Enter an alpha level (0.00 - 1.00) for the sandy soil color:  ";
    $ssc_al = -5;
    until (($ssc_al > -0.01) && ($ssc_al < 1.01))
      {
        $ssc_al = <STDIN>;
        chop($ssc_al);
      }
      $adv {'SSC_ALPHA'} = $ssc_al;
    }


  # ---- DOES THE USER WANT TO SPECIFY A PASSABLE H2O COLOR? ----->

  print "\n\n31)  Do you want to specify a passable water color? (y or n):  ";
  until (($ph2o_yn eq "y") || ($ph2o_yn eq "n"))
    {
      $ph2o_yn = <STDIN>;
      chop($ph2o_yn);
    }

  if ($ph2o_yn eq "y")
    {

    # --- PROMPT USER TO ENTER RED LEVEL FOR PASSABLE H2O COLOR ----->
    print "\n\n31a)  Enter a red level (0.00 - 1.00) for the passable water color:  ";
    $ph2o_rl = -5;
    until (($ph2o_rl > -0.01) && ($ph2o_rl < 1.01))
      {
        $ph2o_rl = <STDIN>;
        chop($ph2o_rl);
      }
      $adv {'PH2O_RED'} = $ph2o_rl;


    # --- PROMPT USER TO ENTER GREEN LEVEL FOR PASSABLE H2O COLOR ----->
    print "\n\n31b)  Enter a green level (0.00 - 1.00) for the passable water color:  ";
    $ph2o_gl = -5;
    until (($ph2o_gl > -0.01) && ($ph2o_gl < 1.01))
      {
        $ph2o_gl = <STDIN>;
        chop($ph2o_gl);
      }
      $adv {'PH2O_GREEN'} = $ph2o_gl;
```

```
    # --- PROMPT USER TO ENTER BLUE LEVEL FOR PASSABLE H2O COLOR ----->
    print "\n\n31c)  Enter a blue level (0.00 - 1.00) for the passable water color:  ";
    $ph2o_bl = -5;
    until (($ph2o_bl > -0.01) && ($ph2o_bl < 1.01))
      {
        $ph2o_bl = <STDIN>;
        chop($ph2o_bl);
      }
      $adv {'PH2O_BLUE'} = $ph2o_bl;


    # --- PROMPT USER TO ENTER ALPHA LEVEL FOR PASSABLE H2O COLOR ----->
    print "\n\n31d)  Enter an alpha level (0.00 - 1.00) for the passable water color:  ";
    $ph2o_al = -5;
    until (($ph2o_al > -0.01) && ($ph2o_al < 1.01))
      {
        $ph2o_al = <STDIN>;
        chop($ph2o_al);
      }
      $adv {'PH2O_ALPHA'} = $ph2o_al;
    }


  # ---- DOES THE USER WANT TO SPECIFY AN IMPASSABLE H2O COLOR? ----->

  print "\n\n32)  Do you want to specify an impassable water color? (y or n):  ";
  until (($iwc_yn eq "y") || ($iwc_yn eq "n"))
    {
      $iwc_yn = <STDIN>;
      chop($iwc_yn);
    }

  if ($iwc_yn eq "y")
    {


    # --- PROMPT USER TO ENTER RED LEVEL FOR IMPASSABLE H2O COLOR ----->
    print "\n\n32a)  Enter a red level (0.00 - 1.00) for the impassable water color:  ";
    $iwc_rl = -5;
    until (($iwc_rl > -0.01) && ($iwc_rl < 1.01))
      {
        $iwc_rl = <STDIN>;
        chop($iwc_rl);
      }
      $adv {'IWC_RED'} = $iwc_rl;


    # --- PROMPT USER TO ENTER GREEN LEVEL FOR IMPASSABLE H2O COLOR ----->
    print "\n\n32b)  Enter a green level (0.00 - 1.00) for the impassable water color:  ";
    $iwc_gl = -5;
    until (($iwc_gl > -0.01) && ($iwc_gl < 1.01))
      {
        $iwc_gl = <STDIN>;
        chop($iwc_gl);
      }
      $adv {'IWC_GREEN'} = $iwc_gl;


    # --- PROMPT USER TO ENTER BLUE LEVEL FOR IMPASSABLE H2O COLOR ----->
    print "\n\n32c)  Enter a blue level (0.00 - 1.00) for the impassable water color:  ";
    $iwc_bl = -5;
    until (($iwc_bl > -0.01) && ($iwc_bl < 1.01))
      {
        $iwc_bl = <STDIN>;
        chop($iwc_bl);
      }
      $adv {'IWC_BLUE'} = $iwc_bl;


    # --- PROMPT USER TO ENTER ALPHA LEVEL FOR IMPASSABLE H2O COLOR ----->
    print "\n\n32d)  Enter an alpha level (0.00 - 1.00) for the impassable water color:  ";
    $iwc_al = -5;
    until (($iwc_al > -0.01) && ($iwc_al < 1.01))
      {
        $iwc_al = <STDIN>;
        chop($iwc_al);
      }
      $adv {'IWC_ALPHA'} = $iwc_al;
    }
```

```
# ---- DOES THE USER WANT TO SPECIFY DEFAULT TERRAIN AMBIENT/DIFFUSE VALUES? ----->

print "\n\n33)  Do you want to specify default";
print " terrain ambient/diffuse values? (y or n):  ";
until (($dtad_yn eq "y") || ($dtad_yn eq "n"))
  {
    $dtad_yn = <STDIN>;
    chop($dtad_yn);
  }

if ($dtad_yn eq "y")
  {

  # --- PROMPT USER TO ENTER DEFAULT TERRAIN AMBIENT VALUE ----->
  print "\n\n33a)  Enter the ambient value (0.00 - 1.00) for the default terrain:  ";
  $dt_av = -5;
  until (($dt_av > -0.01) && ($dt_av < 1.01))
    {
      $dt_av = <STDIN>;
      chop($dt_av);
    }
    $adv {'DT_AV'} = $dt_av;


  # --- PROMPT USER TO ENTER DEFAULT TERRAIN DIFFUSE VALUE ----->
  print "\n\n33b)  Enter the diffuse value (0.00 - 1.00) for the default terrain:  ";
  $dt_dv = -5;
  until (($dt_dv > -0.01) && ($dt_dv < 1.01))
    {
      $dt_dv = <STDIN>;
      chop($dt_dv);
    }
    $adv {'DT_DV'} = $dt_dv;
  }


# ---- DOES THE USER WANT TO SPECIFY DEFAULT NETWORK AMBIENT/DIFFUSE VALUES? ----->

print "\n\n34)  Do you want to specify default";
print " network ambient/diffuse values? (y or n):  ";
until (($dnad_yn eq "y") || ($dnad_yn eq "n"))
  {
    $dnad_yn = <STDIN>;
    chop($dnad_yn);
  }

if ($dnad_yn eq "y")
  {

  # --- PROMPT USER TO ENTER DEFAULT NETWORK AMBIENT VALUE ----->
  print "\n\n34a)  Enter the ambient value (0.00 - 1.00) for the default network:   ";
  $dn_av = -5;
  until (($dn_av > -0.01) && ($dn_av < 1.01))
    {
      $dn_av = <STDIN>;
      chop($dn_av);
    }
    $adv {'DN_AV'} = $dn_av;


  # --- PROMPT USER TO ENTER DEFAULT NETWORK DIFFUSE VALUE ----->
  print "\n\n34b)  Enter the diffuse value (0.00 - 1.00) for the default network:   ";
  $dn_dv = -5;
  until (($dn_dv > -0.01) && ($dn_dv < 1.01))
    {
      $dn_dv = <STDIN>;
      chop($dn_dv);
    }
    $adv {'DN_DV'} = $dn_dv;
  }


# ---- DOES THE USER WANT TO SPECIFY CANOPY AMBIENT/DIFFUSE VALUES? ----->

print "\n\n35)  Do you want to specify canopy ambient/diffuse values? (y or n):  ";
until (($canad_yn eq "y") || ($canad_yn eq "n"))
  {
    $canad_yn = <STDIN>;
    chop($canad_yn);
  }
```

```
if ($canad_yn eq "y")
  {

  # --- PROMPT USER TO ENTER CANOPY AMBIENT VALUE ----->
  print "\n\n35a)  Enter the ambient value (0.00 - 1.00) for the canopy:   ";
  $can_av = -5;
  until (($can_av > -0.01) && ($can_av < 1.01))
    {
      $can_av = <STDIN>;
      chop($can_av);
    }
    $adv {'CAN_AV'} = $can_av;


  # --- PROMPT USER TO ENTER CANOPY DIFFUSE VALUE ----->
  print "\n\n35b)  Enter the diffuse value (0.00 - 1.00) for the canopy:   ";
  $can_dv = -5;
  until (($can_dv > -0.01) && ($can_dv < 1.01))
    {
      $can_dv = <STDIN>;
      chop($can_dv);
    }
    $adv {'CAN_DV'} = $can_dv;
  }


# ---- DOES THE USER WANT TO SPECIFY TREELINE AMBIENT/DIFFUSE VALUES? ----->

print "\n\n36)  Do you want to specify treeline ambient/diffuse values? (y or n):  ";
until (($tlad_yn eq "y") || ($tlad_yn eq "n"))
  {
    $tlad_yn = <STDIN>;
    chop($tlad_yn);
  }

if ($tlad_yn eq "y")
  {

  # --- PROMPT USER TO ENTER TREELINE AMBIENT VALUE ----->
  print "\n\n36a)  Enter the ambient value (0.00 - 1.00) for the treeline:   ";
  $tl_av = -5;
  until (($tl_av > -0.01) && ($tl_av < 1.01))
    {
      $tl_av = <STDIN>;
      chop($tl_av);
    }
    $adv {'TL_AV'} = $tl_av;


  # --- PROMPT USER TO ENTER TREELINE DIFFUSE VALUE ----->
  print "\n\n36b)  Enter the diffuse value (0.00 - 1.00) for the treeline:   ";
  until (($tl_dv > -0.01) && ($tl_dv < 1.01))
    {
      $tl_dv = <STDIN>;
      chop($tl_dv);
    }
    $adv {'TL_DV'} = $tl_dv;
  }


# ---- DOES THE USER WANT TO SPECIFY RAILROAD AMBIENT/DIFFUSE VALUES? ----->

print "\n\n37)  Do you want to specify railroad ambient/diffuse values? (y or n):  ";
until (($rrad_yn eq "y") || ($rrad_yn eq "n"))
  {
    $rrad_yn = <STDIN>;
    chop($rrad_yn);
  }

if ($rrad_yn eq "y")
  {

  # --- PROMPT USER TO ENTER RAILROAD AMBIENT VALUE ----->
  print "\n\n37a)  Enter the ambient value (0.00 - 1.00) for railroads:   ";
  $rr_av = -5;
  until (($rr_av > -0.01) && ($rr_av < 1.01))
    {
      $rr_av = <STDIN>;
      chop($rr_av);
    }
```

```
      $adv {'RR_AV'} = $rr_av;


    # --- PROMPT USER TO ENTER RAILROAD DIFFUSE VALUE ----->
    print "\n\n37b)  Enter the diffuse value (0.00 - 1.00) for railroads:   ";
    $rr_dv = -5;
    until (($rr_dv > -0.01) && ($rr_dv < 1.01))
      {
        $rr_dv = <STDIN>;
        chop($rr_dv);
      }
      $adv {'RR_DV'} = $rr_dv;
    }


  # ---- DOES THE USER WANT TO SPECIFY PAVED SURFACE AMBIENT/DIFFUSE VALUES? ----->

  print "\n\n38)  Do you want to specify paved surface ambient/diffuse values? (y or n):  ";
  until (($psad_yn eq "y") || ($psad_yn eq "n"))
    {
      $psad_yn = <STDIN>;
      chop($psad_yn);
    }

  if ($psad_yn eq "y")
    {

    # --- PROMPT USER TO ENTER PAVED SURFACE AMBIENT VALUE ----->
    print "\n\n38a)  Enter the ambient value (0.00 - 1.00) for paved surfaces:   ";
    $ps_av = -5;
    until (($ps_av > -0.01) && ($ps_av < 1.01))
      {
        $ps_av = <STDIN>;
        chop($ps_av);
      }
      $adv {'PS_AV'} = $ps_av;


    # --- PROMPT USER TO ENTER PAVED SURFACE DIFFUSE VALUE ----->
    print "\n\n38b)  Enter the diffuse value (0.00 - 1.00) for paved surfaces:   ";
    $ps_dv = -5;
    until (($ps_dv > -0.01) && ($ps_dv < 1.01))
      {
        $ps_dv = <STDIN>;
        chop($ps_dv);
      }
      $adv {'PS_DV'} = $ps_dv;
    }


  # ---- DOES THE USER WANT TO SPECIFY PACKED SOIL VALUES? ----->

  print "\n\n39)  Do you want to specify packed soil  ambient/diffuse values? (y or n):  ";
  until (($pksad_yn eq "y") || ($pksad_yn eq "n"))
    {
      $pksad_yn = <STDIN>;
      chop($pksad_yn);
    }

  if ($pksad_yn eq "y")
    {

    # --- PROMPT USER TO ENTER PACKED SOIL AMBIENT VALUE ----->
    print "\n\n39a)  Enter the ambient value (0.00 - 1.00) for packed soil:   ";
    $pks_av = -5;
    until (($pks_av > -0.01) && ($pks_av < 1.01))
      {
        $pks_av = <STDIN>;
        chop($pks_av);
      }
      $adv {'PKS_AV'} = $pks_av;


    # --- PROMPT USER TO ENTER PACKED SOIL DIFFUSE VALUE ----->
    print "\n\n39b)  Enter the diffuse value (0.00 - 1.00) for packed soil:   ";
    $ps_dv = -5;
    until (($ps_dv > -0.01) && ($ps_dv < 1.01))
      {
        $ps_dv = <STDIN>;
        chop($ps_dv);
      }
```

```
      $adv {'PS_DV'} = $ps_dv;
    }


  # ---- DOES THE USER WANT TO SPECIFY SANDY SOIL AMBIENT/DIFFUSE VALUES? ----->

  print "\n\n40)  Do you want to specify sandy soil ambient/diffuse values? (y or n):  ";
  until (($ssad_yn eq "y") || ($ssad_yn eq "n"))
    {
      $ssad_yn = <STDIN>;
      chop($ssad_yn);
    }

  if ($ssad_yn eq "y")
    {

    # --- PROMPT USER TO ENTER SANDY SOIL AMBIENT VALUE ----->
    print "\n\n40a)  Enter the ambient value (0.00 - 1.00) for sandy soil:   ";
    $ss_av = -5;
    until (($ss_av > -0.01) && ($ss_av < 1.01))
      {
        $ss_av = <STDIN>;
        chop($ss_av);
      }
      $adv {'SS_AV'} = $ss_av;


    # --- PROMPT USER TO ENTER SANDY SOIL DIFFUSE VALUE ----->
    print "\n\n40b)  Enter the diffuse value (0.00 - 1.00) for sandy soil:   ";
    $ss_dv = -5;
    until (($ss_dv > -0.01) && ($ss_dv < 1.01))
      {
        $ss_dv = <STDIN>;
        chop($ss_dv);
      }
      $adv {'SS_DV'} = $ss_dv;
    }


  # ---- DOES THE USER WANT TO SPECIFY PASSABLE H2O AMBIENT/DIFFUSE VALUES? ----->

  print "\n\n41)  Do you want to specify passable";
  print " water ambient/diffuse values? (y or n):  ";
  until (($pwtr_yn eq "y") || ($pwtr_yn eq "n"))
    {
      $pwtr_yn = <STDIN>;
      chop($pwtr_yn);
    }

  if ($pwtr_yn eq "y")
    {

    # --- PROMPT USER TO ENTER PASSABLE WATER AMBIENT VALUE ----->
    print "\n\n41a)  Enter the ambient value (0.00 - 1.00) for passable water:   ";
    $pwtr_av = -5;
    until (($pwtr_av > -0.01) && ($pwtr_av < 1.01))
      {
        $pwtr_av = <STDIN>;
        chop($pwtr_av);
      }
      $adv {'PWTR_AV'} = $pwtr_av;


    # --- PROMPT USER TO ENTER PASSABLE WATER DIFFUSE VALUE ----->
    print "\n\n41b)  Enter the diffuse value (0.00 - 1.00) for passable water:   ";
    $pwtr_dv = -5;
    until (($pwtr_dv > -0.01) && ($pwtr_dv < 1.01))
      {
        $pwtr_dv = <STDIN>;
        chop($pwtr_dv);
      }
      $adv {'PWTR_DV'} = $pwtr_dv;
    }


  # ---- DOES THE USER WANT TO SPECIFY IMPASSABLE H2O AMBIENT/DIFFUSE VALUES? ----->

  print "\n\n42)  Do you want to specify impassable";
  print " water ambient/diffuse values? (y or n):  ";
  until (($iwtr_yn eq "y") || ($iwtr_yn eq "n"))
    {
```

```
      $iwtr_yn = <STDIN>;
      chop($iwtr_yn);
    }

  if ($iwtr_yn eq "y")
    {

    # --- PROMPT USER TO ENTER IMPASSABLE WATER AMBIENT VALUE ----->
    print "\n\n42a)  Enter the ambient value (0.00 - 1.00) for impassable water:   ";
    $iwtr_av = -5;
    until (($iwtr_av > -0.01) && ($iwtr_av < 1.01))
      {
        $iwtr_av = <STDIN>;
        chop($iwtr_av);
      }
      $adv {'IWTR_AV'} = $iwtr_av;


    # --- PROMPT USER TO ENTER IMPASSABLE WATER DIFFUSE VALUE ----->
    print "\n\n42b)  Enter the diffuse value (0.00 - 1.00) for impassable water:   ";
    $iwtr_dv = -5;
    until (($iwtr_dv > -0.01) && ($iwtr_dv < 1.01))
      {
        $iwtr_dv = <STDIN>;
        chop($iwtr_dv);
      }
      $adv {'IWTR_DV'} = $iwtr_dv;
    }


  # ------- DOES THE USER WANT TO INCLUDE A DATABASE MODEL FILE? ------->

  print "\n\n43)  Do you want to include a database model file? (y or n):  ";
  until (($dmf_yn eq "y") || ($dmf_yn eq "n"))
    {
      $dmf_yn = <STDIN>;
      chop($dmf_yn);
    }

  if ($dmf_yn eq "y")
    {
      # ---- PROMPT USER FOR DATABASE MODEL FILE PATH ----->

      print "\n\n43a)  Specify the database model file path and filename:\n\n";
      $dmf_name = <STDIN>;
      chop($dmf_name);
      $adv{'DB_MODEL_FILE'} = $dmf_name;
    }


  # ------- DOES THE USER WANT TO INCLUDE A TERRAIN MODEL? ------->

  print "\n\n44)  Do you want to include a terrain model? (y or n):  ";
  until (($tm_yn eq "y") || ($tm_yn eq "n"))
    {
      $tm_yn = <STDIN>;
      chop($tm_yn);
    }

  if ($tm_yn eq "y")
    {
      # ---- PROMPT USER FOR TERRAIN MODEL PATH ----->

      print "\n\n44a)  Specify the terrain model path and filename:\n\n";
      $tm_name = <STDIN>;
      chop($tm_name);
      $adv{'TERRAIN_MODEL'} = $tm_name;
    }


    # ------- DOES THE USER WANT TO ENABLE PAGING? ------->

    print "\n\n45)  Would you like to enable paging? (y or n):  ";
    until (($page_yn eq "y") || ($page_yn eq "n"))
      {
        $page_yn = <STDIN>;
        chop($page_yn);
      }

    if ($page_yn eq "y")
      {
```

```
        $adv{'ENABLE_PAGING'} = "ENABLE_PAGING";
      }

  }


#-----------------------------------
# SUBROUTINE help_ctrl
#-----------------------------------
sub help_ctrl
  {
     print "\n\n";
     print "-------------------------------------------------------------------\n";
     print " This program is used to generate S1K control files that can be\n";
     print " used to view and navigate three dimensional terrain databases\n";
     print " that were created using S1000 modelling software.  Use the SIMPLE\n";
     print " option to create a simple control file for a quick look at a given\n";
     print " terrain database.  Use the ADVANCED option to create a detailed\n";
     print " database complete with terrain colors, clipped viewing areas, etc.\n\n";

     print " For more information, see the S1KPERFLY on-line documentation\n";
     print " under /net/france/data1/tis1k/template.s1k.\n";
     print "-------------------------------------------------------------------\n";
     print "\n\n";
  }



#-----------------------------------
# SUBROUTINE simple_write
#-----------------------------------
sub simple_write
  {

    # ---- WRITE OUT USER INPUT FROM ASSOCIATIVE ARRAY ----->

    print OutFile "S1KPROJ $simple{'S1KPROJ'}\n";

    print OutFile "PROJECT $simple{'PROJECT'}\n";

    if ($simple{'ASSEMBLY_PREFIX'})
      {
        print OutFile "ASSEMBLY_PREFIX $simple{'ASSEMBLY_PREFIX'}\n";
      }

    if ($simple{'UTM'})
      {
        print OutFile "$simple{'UTM'}\n";
      }

    if ($simple{'SEARCH_WIN_X1'})
      {
        print OutFile "SEARCH_WINDOW $simple{'SEARCH_WIN_X1'}";
        print OutFile " $simple{'SEARCH_WIN_Y1'}";
        print OutFile " $simple{'SEARCH_WIN_X2'}";
        print OutFile " $simple{'SEARCH_WIN_Y2'}\n";
      }

    if ($simple{'ORIGIN_X'})
      {
        print OutFile "ORIGIN $simple{'ORIGIN_X'}";
        print OutFile " $simple{'ORIGIN_Y'}\n";
      }

  }



#-----------------------------------
# SUBROUTINE adv_write
#-----------------------------------
sub adv_write
  {

    # ---- WRITE OUT USER INPUT FROM ASSOCIATIVE ARRAY ----->

    if ($adv{'INACTIVE_POLY'})
      {
        print OutFile "INCLUDE_INACTIVE_POLYGONS\n";
      }
```

```
    if ($adv{'MODEL_FILTER'})
      {
        print OutFile "MODEL_FILTER $adv{'MODEL_FILTER'}\n";
      }

    if ($adv{'VTX_NORMALS'})
      {
        print OutFile "VTX_NORMALS\n";
      }

    if ($adv{'LEAF_SIZE_MAX'})
      {
         print OutFile "LEAF_SIZE_MAX $adv{'LEAF_SIZE_MAX'}\n";
      }

    if ($adv{'TERRAIN_MAX_DISTANCE'})
      {
         print OutFile "TERRAIN_MAX_DISTANCE $adv{'TERRAIN_MAX_DISTANCE'}\n";
      }

    if ($adv{'CANOPY_MAX_DISTANCE'})
      {
         print OutFile "CANOPY_MAX_DISTANCE $adv{'CANOPY_MAX_DISTANCE'}\n";
      }

    if ($adv{'NETWORK_MAX_DISTANCE'})
      {
         print OutFile "NETWORK_MAX_DISTANCE $adv{'NETWORK_MAX_DISTANCE'}\n";
      }

    if ($adv{'TREELINE_MAX_DISTANCE'})
      {
         print OutFile "TREELINE_MAX_DISTANCE $adv{'TREELINE_MAX_DISTANCE'}\n";
      }

    if ($adv{'STAMP_MAX_DISTANCE'})
      {
         print OutFile "STAMP_MAX_DISTANCE $adv{'STAMP_MAX_DISTANCE'}\n";
      }

    if ($adv{'MODEL_MAX_DISTANCE'})
      {
         print OutFile "MODEL_MAX_DISTANCE $adv{'MODEL_MAX_DISTANCE'}\n";
      }

    if ($adv{'SUPPRESS'})
      {
         print OutFile "SUPPRESS $adv{'SUPPRESS'}\n";
      }

    if ($adv{'CLONE_GEOMETRY'})
      {
         print OutFile "CLONE_GEOMETRY\n";
      }

    if ($adv{'PFDECAL_BASE'})
      {
         print OutFile "PFDECAL_BASE $adv{'PFDECAL_BASE'}\n";
      }

    if ($adv{'MULTISAMPLE'})
      {
         print OutFile "MULTISAMPLE\n";
      }

    if ($adv{'INCLUDE'})
      {
         print OutFile "INCLUDE $adv{'INCLUDE'}\n";
      }

    if ($adv{'TI_ONLY_VIEWING_MODE'})
      {
         print OutFile "VIEWING_MODE $adv{'TI_ONLY_VIEWING_MODE'}\n";
      }

    if ($adv{'DTC_RED'})
      {
         print OutFile "DEFAULT_TERRAIN_COLOR $adv{'DTC_RED'}";
         print OutFile " $adv{'DTC_GREEN'}";
         print OutFile " $adv{'DTC_BLUE'}";
         print OutFile " $adv{'DTC_ALPHA'}\n";
```

```
      }

    if ($adv{'DNC_RED'})
      {
         print OutFile "DEFAULT_NETWORK_COLOR $adv{'DNC_RED'}";
         print OutFile " $adv{'DNC_GREEN'}";
         print OutFile " $adv{'DNC_BLUE'}";
         print OutFile " $adv{'DNC_ALPHA'}\n";
      }

    if ($adv{'CAN_RED'})
      {
         print OutFile "CANOPY_COLOR $adv{'CAN_RED'}";
         print OutFile " $adv{'CAN_GREEN'}";
         print OutFile " $adv{'CAN_BLUE'}";
         print OutFile " $adv{'CAN_ALPHA'}\n";
      }

    if ($adv{'TLC_RED'})
      {
         print OutFile "TREELINE_COLOR $adv{'TLC_RED'}";
         print OutFile " $adv{'TLC_GREEN'}";
         print OutFile " $adv{'TLC_BLUE'}";
         print OutFile " $adv{'TLC_ALPHA'}\n";
      }

    if ($adv{'RRC_RED'})
      {
         print OutFile "RAILROAD_COLOR $adv{'RRC_RED'}";
         print OutFile " $adv{'RRC_GREEN'}";
         print OutFile " $adv{'RRC_BLUE'}";
         print OutFile " $adv{'RRC_ALPHA'}\n";
      }

    if ($adv{'PSC_RED'})
      {
         print OutFile "PAVED_SURFACE_COLOR $adv{'PSC_RED'}";
         print OutFile " $adv{'PSC_GREEN'}";
         print OutFile " $adv{'PSC_BLUE'}";
         print OutFile " $adv{'PSC_ALPHA'}\n";
      }

    if ($adv{'SC_RED'})
      {
         print OutFile "PACKED_SOIL_COLOR $adv{'SC_RED'}";
         print OutFile " $adv{'SC_GREEN'}";
         print OutFile " $adv{'SC_BLUE'}";
         print OutFile " $adv{'SC_ALPHA'}\n";
      }

    if ($adv{'SSC_RED'})
      {
         print OutFile "SANDY_SOIL_COLOR $adv{'SSC_RED'}";
         print OutFile " $adv{'SSC_GREEN'}";
         print OutFile " $adv{'SSC_BLUE'}";
         print OutFile " $adv{'SSC_ALPHA'}\n";
      }

    if ($adv{'PH20_RED'})
      {
         print OutFile "PASSABLE_H2O_COLOR $adv{'PH2O_RED'}";
         print OutFile " $adv{'PH2O_GREEN'}";
         print OutFile " $adv{'PH2O_BLUE'}";
         print OutFile " $adv{'PH2O_ALPHA'}\n";
      }

    if ($adv{'IWC_RED'})
      {
         print OutFile "IMPASSABLE_H2O_COLOR $adv{'IWC_RED'}";
         print OutFile " $adv{'IWC_GREEN'}";
         print OutFile " $adv{'IWC_BLUE'}";
         print OutFile " $adv{'IWC_ALPHA'}\n";
      }

    if ($adv{'DT_AV'})
      {
         print OutFile "DEFAULT_TERRAIN_AD $adv{'DT_AV'}";
         print OutFile " $adv{'DT_DV'}\n";
      }

    if ($adv{'DN_AV'})
```

```
      {
         print OutFile "DEFAULT_NETWORK_AD $adv{'DN_AV'}";
         print OutFile " $adv{'DN_DV'}\n";
      }

    if ($adv{'CAN_AV'})
      {
         print OutFile "CANOPY_AD $adv{'CAN_AV'}";
         print OutFile " $adv{'CAN_DV'}\n";
      }

    if ($adv{'TL_AV'})
      {
         print OutFile "TREELINE_AD $adv{'TL_AV'}";
         print OutFile " $adv{'TL_DV'}\n";
      }

    if ($adv{'RR_AV'})
      {
         print OutFile "RAILROAD_AD $adv{'RR_AV'}";
         print OutFile " $adv{'RR_DV'}\n";
      }

    if ($adv{'PS_AV'})
      {
         print OutFile "PAVED_SURFACE_AD $adv{'PS_AV'}";
         print OutFile " $adv{'PS_DV'}\n";
      }

    if ($adv{'PKS_AV'})
      {
         print OutFile "PACKED_SOIL_AD $adv{'PKS_AV'}";
         print OutFile " $adv{'PKS_DV'}\n";
      }

    if ($adv{'SS_AV'})
      {
         print OutFile "SANDY_SOIL_AD $adv{'SS_AV'}";
         print OutFile " $adv{'SS_DV'}\n";
      }

    if ($adv{'PWTR_AV'})
      {
         print OutFile "PASSABLE_H20_AD $adv{'PWTR_AV'}";
         print OutFile " $adv{'PWTR_DV'}\n";
      }

    if ($adv{'IWTR_AV'})
      {
         print OutFile "IMPASSABLE_H20_AD $adv{'IWTR_AV'}";
         print OutFile " $adv{'IWTR_DV'}\n";
      }

    if ($adv{'DB_MODEL_FILE'})
      {
         print OutFile "DB_MODEL_FILE $adv{'DB_MODEL_FILE'}\n";
      }

    if ($adv{'TERRAIN_MODEL'})
      {
         print OutFile "TERRAIN_MODEL $adv{'TERRAIN_MODEL'}\n";
      }

    if ($adv{'ENABLE_PAGING'})
      {
         print OutFile "ENABLE_PAGING $adv{'ENABLE_PAGING'}\n";
      }

  }

#-----------------------------------
# SUBROUTINE house_keep
#-----------------------------------
sub house_keep
  {

    close(OutFile);

    print "\n\n=====================================================";
    print "\n\nS1KC.PRL has successfully created the control file";
    print " $out_ctrl_file_name.\n\n";    print "Execution of S1KC.PRL now complete.\n\n";
```

```
    print "=====================================================\n\n";

  }
}
```

2.2.19 tin1.aml

PURPOSE: This AML takes in a clip coverage in UTM and creates a new coverage - a polygon coverage covering the entire playbox area with 500m squares. The overall extents are queried and if they do not match the original extents, a new playbox is created with the same lower left but different upper right. This is the second in a many step process to create s1000 terrain in Arc/Info:

1. Create a playbox coverage consisting of a single box in UTM. Be sure to build.
2. Run this AML, tin1, to create a 500m grid covering the playbox area and create a new clip coverage if necessary.
3. Create the TIN
4. Run tin3.aml to split the Tin up into smaller pieces and export it into the appropriate S1000 addwams format.

HISTORY: This program was originally coded in August 1996 by Tobi Sellekaerts, and last updated February 1997. We now make almost all of our S1000 TINs in iTIN so we don't use this program very often.

USAGE: To run this program, first create a playbox coverage consisting of a single box in UTM. Execute the program at the Arc/Info prompt:

```
Arc:  &r tin1.aml <extents_cover> <output_cover>
```

where `<extents_cover>` is the coverage whose extents match those of the SimNet playbox you are trying to create and `<output_cover>` is the 500 meter load module polygon coverage you are creating. The `<output_cover>` should not already exist in the current workspace.

SAMPLE OUTPUT: n/a

PROGRAM FLOW: The program tests the extents coverage to see if it covers an area divisible by 500 meters in each direction. If not, the upper right corner is adjusted accordingly. It then creates a 500m x 500m load module grid coverage and cleans it. Finally, it assigns an attribute to each polygon in the coverage: the S1000 load module name which is its row and column designator. The attribute is called LMPAIR.

001-000	001-001	001-002	001-003	001-004
000-000	000-001	000-002	000-003	000-004

tlss

```
&args clip 500mgrid

/* tin1.aml
/* ----------------------------------------------------------------
/*    Program: tin1.aml
/*
/*    Purpose: FOR USE WITH S1000 TERRAIN BUILDS
/*    This AML takes in a clip coverage in UTM and creates a new
/*    coverage - a polygon coverage covering the entire
/*    playbox area with 500m squares.  The overall extents are queried
/*    and if they do not match the original extents, a new playbox
/*    is created with the same lower left but different upper right.
/*    This is the second in a many step process to create s1000
/*    terrain in Arc/Info.
/*
/*    1. Create a playbox coverage consisting of a single box in UTM.
/*       (you can use the atool createclip.aml)
/*    2. Run this AML, tin1, to create a 500m grid covering the playbox
/*       area and create a new clip coverage if necessary.
/*    3. Create the TIN, using tin2.aml (still to be written).
/*    4. Run tin3.aml to split the Tin up into smaller pieces and export
/*       it into the appropriate S1000 addwams format.
/* ----------------------------------------------------------------
/*    Called By:  none
/*    Calls Made:  none
/* ----------------------------------------------------------------
/*    Arguments:  name of your extents coverage, name of the output
/*     load module polygon coverage
/*    Globals:  none
/* ----------------------------------------------------------------
/*    Inputs: coverage whose extents match those of the SimNet playbox
/*    Outputs: 500 meter load module polygon coverage
/* ----------------------------------------------------------------
/*    History:  August 1996 - Tobi Steinberg
/*    Terrain Simulations - CUBIC Inc. - Grafenwoehr, Germany
/*
/*    Revised February 1997, as the procedure for TINning is
/*    radically changing.
/* ----------------------------------------------------------------
&severity &error &routine bail

&if [null %500mgrid%] &then &do
   &ty Usage:TIN1 <clip_cover> <out_polycov>
   &return
&end

display 0

/* value of tester will determine whether or not a new clip coverage
/* will be needed.  If tester is set to 1 at any point then the new
/* necessary extents don't match the original extents.
&s tester 0

&describe %clip%
&s xmin %dsc$xmin%
&s ymin %dsc$ymin%
&s xmax %dsc$xmax%
&s ymax %dsc$ymax%
create %500mgrid% %clip%

&s xtot [calc [calc %xmax% - %xmin%] / 500]
&if %xtot% ne [truncate %xtot%] &then &do
   &s xtot [calc [truncate [calc [calc %xmax% - %xmin%] / 500]] + 1]
   &s tester 1
&end
&s xmaxnew [calc [calc %xtot% * 500] + %xmin%]

&s ytot [calc [calc %ymax% - %ymin%] / 500]
&if %ytot% ne [truncate %ytot%] &then &do
   &s ytot [calc [truncate [calc [calc %ymax% - %ymin%] / 500]] + 1]
   &s tester 1
&end
```

```
&s ymaxnew [calc [calc %ytot% * 500] + %ymin%]

ae

/* Construct 500m by 500m grid beginning at the lower left of the playbox

   ec %500mgrid%
   ef arc
   coordinates keyboard
   add

   &do x = %xmin% &to %xmaxnew% &by 500
      2, %x%, %ymaxnew%
      2, %x%, %ymin%
   &end

   &do y = %ymin% &to %ymaxnew% &by 500
      2, %xmin%, %y%
      2, %xmaxnew%, %y%
   &end

   9
   nodesnap closest 5
   clean
   save

/* add row and column designators to the grid, starting with 0,0
/* in the lower left corner

   ef poly
   additem row 3 3 I
   additem column 3 3 I
   additem lmpair 7 7 C

   &do x = 0 &to [calc %xtot% - 1]
      &do y = 0 &to [calc %ytot% - 1]
         sel
         [calc %xmin% + 250 + [calc %x% * 500]], ~
         [calc %ymin% + 250 + [calc %y% * 500]]
         calc column = %x%
         calc row = %y%
         &if %x% lt 100 &then &do
           &if %x% lt 10 &then &s tempx 00%x%
           &else &s tempx 0%x%
         &end
         &else &s tempx %x%
         &if %y% lt 100 &then &do
           &if %y% lt 10 &then &s tempy 00%y%
           &else &s tempy 0%y%
         &end
         &else &s tempy %y%
         moveitem [quote %tempy%-%tempx%] to lmpair
      &end
   &end
   save

/* If the extents of the 500m grid are different than those of the
/* original playbox area, the program will create a new playbox extents cover

   &if %tester% = 1 &then &do
   &s newclip [response 'What would you like to call ~
                        the new clip cover?' newclip]
   &if [exists %newclip% -cover] &then kill %newclip% yes
   create %newclip% %500mgrid%
   ec %newclip%
   ef arc
   add
   2, %xmin%, %ymin%
   1, %xmin%, %ymaxnew%
   1, %xmaxnew%, %ymaxnew%
   1, %xmaxnew%, %ymin%
   2, %xmin%, %ymin%
```

```
   9
   build
   save
   &end

   coordinates mouse
quit

display 9999
&return

/*****************************************************************
/* bail routine
/*****************************************************************
&routine bail
&severity &error &ignore
&severity &warning &ignore
&type An error has occured in tin3.aml
&type Bailing out of tin3.aml
&return; &return &error
```

2.2.20 tin2.aml

PURPOSE: This AML thins a grid created from dted by querying on a neighborhood basis in grid. If a cell has the same value as all the cells touching it, or is within a certain slope value, that cell is given a value of NODATA.

HISTORY: This program was originally coded in August 1996 by Tobi Sellekaerts. It is no longer used in the TIN generation process.

USAGE: To run this program, first import your DTED or elevation data into Arc/Info so that it exists as a GRID. Execute the program at the Arc/Info prompt:

```
Arc: &r tin2.aml <in_grid> <out_grid> <out_point>
```

where `<in_grid>` is your elevation data in GRID format, `<out_grid>` is the name of your new thin grid (`<out_grid>` should not already exist), and `<out_point>` is the new grid converted to a point coverage (`<out_point>` should not already exist).

SAMPLE OUTPUT: n/a

PROGRAM FLOW: The program thins the GRID by querying on a neighborhood basis. If a cell has the same elevation value (within 5 meters) as all the cells touching it, or is within 3 degrees of the same slope oriented within 3 degrees of the same direction, that cell is given a value of NODATA. The thinned GRID is converted to a point coverage with GRIDPOINT, and all cells with the NODATA value are excluded from the point coverage.

tlss

```
&args ingrid outgrid point
&if [null %outgrid%] &then &do
   &ty Usage: tin2 <in_grid> <out_grid> <out_point>
   &return
&end
&echo &on; display 0
/* ============================================
/* AML (used) to be used with S1000 terrain generation.
/* ============================================
/* This AML thins a grid created from dted by querying on a
/* neighborhood basis in grid.  If a cell has the same value as
/* all the cells touching it, or is within a certain slope value,
/* that cell is given a value of NODATA.
/* All NODATA cells will not be converted to points and won't add
/* unneccesary data to the final TIN.
/* Created August 1996 by Tobi Steinberg.
/* ============================================
/* October - changing slope values to within 3 degrees on Ernie's advice
/* February, 1997 - this AML is no longer applicable to creating S1000
/* terrain as we are not using the entire DTED anymore.
grid

/* If a cell has the same value as all the cells touching it, within
/* a tolerance of 5 meters, that cell is given a value of NODATA.
/* 5 meters is the rise over 100 meters distance that would lead
/* to an angle variation of 3 degrees
   temp1 = focalrange (%ingrid%)
   tempval = con (temp1 > 5, %ingrid%)
   kill temp1 all

/* If a cell has the same slope as all the cells touching it, within
/* a tolerance of 3 degrees, that cell is given a value of NODATA.
/* It will later be tested for consistency of direction of slope.
   temp1 = slope (%ingrid%)
   temp2 = focalrange (temp1)
   tempslo = con (temp2 > 3, %ingrid%, -100)
   kill temp1 all
   kill temp2 all

/* If a cell has the same aspect as all the cells touching it, within
/* a tolerance of 3 degrees, that cell is given a value of NODATA.
   temp1 = aspect (%ingrid%)
   temp2 = focalrange (temp1)
   tempasp = con (temp2 > 3, %ingrid%, -100)
   kill temp1 all
   kill temp2 all

/* Now comparing the cells with consistent slopes and aspects as those
/* surrounding them.  If both are consistent, the value will be set to
/* NODATA and left out of the final point coverage used in building the TIN.
   tempsa = con (con(tempslo ne -100, %ingrid%, tempasp) ne -100, %ingrid%)
   %outgrid% = max (tempval, tempsa)
   kill tempval all
   kill tempsa all
   kill tempasp all
   kill tempslo all
quit

&if ^ [null %point%] &then
   gridpoint %outgrid% %point% spot
&echo &off
display 9999
&return
```

2.2.21 tin3.aml

PURPOSE: This program takes a completed TIN (created manually) and an Arc/Info polygon coverage delineating load module boundaries (created with tin1.aml), and exports them to the appropriate S1000 addwams format. See the **PURPOSE** section under **tin1.aml** for a more specific description of where this program falls in the TIN generation procedure.

HISTORY: This program was originally coded in August 1996 by Tobi Sellekaerts, and was last updated in February 1997. This program is rarely used as we create most of our S1000 TINs in iTIN.

USAGE: Execute the program at the Arc/Info prompt:

```
Arc: TIN3 <in_tin> <lm-grid_cover> <out_name> {resolution}
```

where `<in_tin>` is your TIN (with the load module boundaries incorporated into the TIN), `<lm-grid_cover>` is the name of the polygon coverage created with tin1.aml, `<out_name>` is the name of the directory that will be created for the output text files (it should not already exist), and `{resolution}` is an optional item specifying the distance in meters into which the TIN will be cut (see **PROGRAM FLOW**).

SAMPLE OUTPUT: The coverages created by the AML use the following naming structure:

```
xtpy_0_0    xtpt_0_0
xtpy_0_1    xtpt_0_1
xtpy_0_2 (etc.)
```

These coverages are exported to text files called:

```
xtpy_0_0.moss           xtpt_0_0.moss
xtpy_0_1.moss           xtpt_0_1.moss
xtpy_0_2.moss (etc.)
```

PROGRAM FLOW: If there are more than 10,000 points or polygons, then the playbox area must be cut up into smaller squares for addwams export. The program tests for more than 10,000 points or polygons, and if they exist, it will go ahead and cut up the TIN (if no resolution was specified, the program will calculate one). If the TIN has fewer than 10,000 points or polygons, then it will not be cut, even if a resolution was specified in the arguments. The program converts the TIN into polygon and point coverages with TINARC - multiple sets if the TIN is being split. Arc exports these coverages to text files with ARCMOSS. Finally, a perl script called tin3perl is called to format the text files.

tlss

```
&args intin clip name res

/* tin3.aml
/* ------------------------------------------------------------
/*
/*     Program: tin3.aml
/*     Purpose: The final AML in the tin generation series; takes
/*     a completed tin and the LM poly coverage, and exports them to
/*     the appropriate S1000 addwams format.
/* ------------------------------------------------------------
/*     Called By:  none
/*     Calls Made:  accesses a perl script called tin3perl
/* ------------------------------------------------------------
/*     Arguments:  Need to enter in the name of the Tin, the name
/*     of the load module polygon coverage, the output name you
/*     would like the group of files to be called, and the
/*     resolution of the tiles into which the grid will be cut.
/*     (A suggestion - try starting with 5.)
/* ------------------------------------------------------------
/*     Inputs: the completed TIN, the load module boundary coverage,
/*     and the slicing resolution.
/*     Outputs: properly formatted ADDWAMS files for import into
/*     S1000.
/* ------------------------------------------------------------
/*     History: August 1996  - Tobi Steinberg
/*     Terrain Simulations - Grafenwoehr, Germany
/*     Parts rewritten in February 1997; changing the slow section
/*     at the end and attempting to use perl to speed it up.
/*     Also, have to avoid using an identity because it is not
/*     accurate enough.
/*     Final comment, 17 March 1997- this seems to run perfectly when
/*     a very low tolerance is used in the CREATETIN.  Finished today.
/* ------------------------------------------------------------
&severity &error &routine bail

&if [exists t3_[date -tag].wat -file] &then
   &s delwat [delete t3_[date -tag].wat]
&watch t3_[date -tag].wat; &ty [date -VFULL]

&echo &on
display 0

/* Checking validity of entered arguments
&if [null %clip%] &then &do
   &ty Usage: TIN3 <in_tin> <lm-grid_cover> <out_name> {resolution}
   &return
&end

&if ^ [exists %clip% -cover] &then &do
   &ty Grid coverage %clip% does not exist.
   &ty Usage: TIN3 <in_tin> <lm-grid_cover> <out_name> {resolution}
   &return
&end

&if ^ [exists %intin% -tin] &then &do
   &ty Tin %intin% does not exist.
   &ty Usage: TIN3 <in_tin> <lm-grid_cover> <out_name> {resolution}
   &return
&end

&s autocalcres 0
&if [null %res%] &then &do
   /* automatically calculating the optimal cell size for dividing the
   /* area up into smaller cells for output
   &s autocalcres 1
   &describe %intin%
   &s x [calc %tin$xmax% - %tin$xmin%]
   &s y [calc %tin$ymax% - %tin$ymin%]
   &s res [truncate[calc [sqrt [calc [calc [calc %x% * %y%] * 9999] ~
   / %tin$ntris%]] / 1000]]
&end

&label badres
&if [show program] ne ARC &then quit
&if %autocalcres% = 2 &then &do
   /* comes here to repair itself when it overestimated the cell size
   &s autocalcres 1
   &if %res% gt 1000 &then &s res [calc %res% / 1000]
   &if %res% = 1 &then &do
      &ty Automatic calculation of the cell size resolution
      &ty stopped at 1.  Make a TIN with better point distribution.
   &end
   &s res [calc %res% - 1]
&end

/* Querying for a destination for the output files.
&if [exists ../export -directory] &then
   &s dir ../export
&else &do
   &label baddir
   &s dir [response 'Where would you like to put your output files?']
   &if ^ [exists %dir% -directory] &then &do
      &ty Directory %dir% does not exist.
      w
      &goto baddir
   &end
&end

/* Creating a directory to put the output files in if it does not exist
&if ^ [exists %dir%/%name% -directory] &then &sys mkdir %dir%/%name%

/* Checking on the existence of coverage names needed by the program
&do cov &list xtpoly xtpy xtpoint xtpt xxlabel xxlabel2
   &if [exists %cov% -cover] &then kill %cov% all
&end

/* ==============================================
/* Determine whether or not the TIN must be split up.
/* ==============================================

&describe %intin%
&if %tin$ntris% lt 10000 &then &do
   &s noclip .TRUE.
   &if [exists xtpy_0_0 -cover] &then kill xtpy_0_0 all
   &if [exists xtpt_0_0 -cover] &then kill xtpt_0_0 all
   &s xreps 1
   &s yreps 1
   &goto endclip
```

```
&end

/* ============================================
/* Query the playbox size and determine to what extent
/* it must be split into pieces.
/* ============================================

/* Making sure the resolution is a number divisible
/* by 500 meters.
&if [calc %res% / .5] ne [truncate [calc %res% / .5]] &then &do
   &s res [calc [truncate [calc %res% / .5]] * .5]
   &ty Changing the grid size to [truncate %res%]
&end

/* Obtain the relative dimensions of the playbox.
&describe %clip%
&s xdiff [calc %dsc$xmax% - %dsc$xmin%]
&s ydiff [calc %dsc$ymax% - %dsc$ymin%]

/* Convert the resolution to meters instead of kilometers.
&s res [calc 1000 * %res%]

/* If the chosen resolution is greater than the overall
/* dimensions of the playbox in both directions,
/* skip the clipping section of the program.
&if %xdiff% lt %res% AND %ydiff% lt %res% &then &do
   &ty Your resolution of %res% is less than both your
   &ty x dimension of %xdiff% and your y dimension of %ydiff%.
   &ty Skipping the clip section of the program.
   &s noclip .TRUE.
   &s xreps 1
   &s yreps 1
   &goto endclip
&end

/* Determine how many boxes there will be in each direction.
&s xreps [truncate [calc %xdiff% / %res%]]
&s yreps [truncate [calc %ydiff% / %res%]]
&s xremain [calc %xdiff% - [calc %xreps% * %res%]]
&s yremain [calc %ydiff% - [calc %yreps% * %res%]]

/* If there is only going to be one box in each direction,
/* skip the section which cuts the playbox into smaller boxes.
&if %xreps% = 1 AND %yreps% = 1 &then &do
   &ty There is only going to be one box in each direction.
   &ty Skipping the clip section of the program.
   &s noclip .TRUE.
   &goto endclip
&end

/* If the width of the remaining swath is less than
/* 30% of the width of the resolution, tack the extra
/* on to the remaining square.
&if %xremain% gt [calc %res% * .3] &then
   &s xreps [calc %xreps% + 1]
&if %yremain% gt [calc %res% * .3] &then
   &s yreps [calc %yreps% + 1]

&label endclip
&if [show program] ne ARC &then quit

/* ============================================
/* Export the TIN to both poly and point Arc coverages.
/* Clip the poly and point coverages into the appropriate pieces.
/* Completely changing the way I do this section on 13 Feb 1997.
/* Trying to use perl to speed up the whole text file process.
/* ============================================

tinarc %intin% xtpoly poly
tinarc %intin% xtpoint point
tolerance xtpoly fuzzy .0001
tolerance xtpoint fuzzy .0001

/* Putting out the label points from the triangles to assign
/* load module row and columns to them.  Doing an identity to
/* determine which row and column they fall into, the using a
/* relate to assign these values back into the polygon coverage.

ae
   ec xtpoly
   ef label
   sel all
   delete
   build
   sel all
   put xxlabel
   nodesnap closest .0001
   ed .0001
   grain .0001
   wt .0001
   save
   ec xtpoint
   ef point
   nodesnap closest .0001
   ed .0001
   grain .0001
   wt .0001
   save
quit

identity xxlabel %clip% xxlabel2 point
relate add
   rowcol
   xxlabel2.pat
   info
   xtpoly-id
   xxlabel-id
   ordered
   rw
   ~
additem xtpoly.pat xtpoly.pat row 3 3 I
additem xtpoly.pat xtpoly.pat column 3 3 I
additem xtpoly.pat xtpoly.pat lmpair 7 7 C

tables
   sel xxlabel2.pat
      sort xxlabel-id
   sel xtpoly.pat
```

```
      calc row = rowcol//row
      calc column = rowcol//column
      move rowcol//lmpair TO lmpair
   sel xxlabel2.pat
      sort xxlabel2#
   relate drop
      rowcol
      ~
quit

/* Relic from an older time...
rename xtpoly xtpy
rename xtpoint xtpt

/* Clipping the polygon and points coverages into the smaller
/* coverages necessary for dividing up the tin to go into S1000.
/* Do the whole thing twice, once for poly and once for points.
/* It is more code but runs much faster because it is the "ec xtpy"
/* and "ec xtpt" that make it slow!

&if %xreps% ne 1 OR %yreps% ne 1 &then &do
   ae
      &describe %clip%
      coordinates keyboard
      ec xtpy
      ef poly
      &do x = 0 &to [calc %xreps% - 1]
         &s tmpxmin [calc [calc %x% * %res%] + %dsc$xmin%]
         &s tmpxmax [calc [calc [calc %x% * %res%] + %res%] + %dsc$xmin%]
         &if %x% = [calc %xreps% - 1] &then &s tmpxmax %dsc$xmax%
         &do y = 0 &to [calc %yreps% - 1]
            &if [exists xtpy_%x%_%y% -cover] &then kill xtpy_%x%_%y% yes
            &if %y% = [calc %yreps% - 1] &then &s tmpymax %dsc$ymax%
            &s tmpymin [calc [calc %y% * %res%] + %dsc$ymin%]
            &s tmpymax [calc [calc [calc %y% * %res%] + %res%] +
       %dsc$ymin%]

            /* First, extracting from the polygon coverage
            sel box
            %tmpxmin%, %tmpymin%
            %tmpxmax%, %tmpymax%
            &s totpy [show number selected]
            &if %totpy% gt 9999 &then &do
               &if %autocalcres% = 0 &then &do
                  &ty This square has %totpy% polygons and
                  &ty exceeds the total number allowed, 9999.
                  &ty Please restart with a smaller cell size.
                  quit
                  &goto cleanup
               &end
               &else &do
                  &s autocalcres 2
                  &goto badres
               &end
            &end
            &else &do
               put xtpy_%x%_%y%
            &end
         &end
      &end

      ec xtpt
      ef point
      &do x = 0 &to [calc %xreps% - 1]
         &s tmpxmin [calc [calc %x% * %res%] + %dsc$xmin%]
         &s tmpxmax [calc [calc [calc %x% * %res%] + %res%] + %dsc$xmin%]
         &if %x% = [calc %xreps% - 1] &then &s tmpxmax %dsc$xmax%
         &do y = 0 &to [calc %yreps% - 1]
            &if [exists xtpt_%x%_%y% -cover] &then kill xtpt_%x%_%y% yes
            &if %y% = [calc %yreps% - 1] &then &s tmpymax %dsc$ymax%
            &s tmpymin [calc [calc %y% * %res%] + %dsc$ymin%]
            &s tmpymax [calc [calc [calc %y% * %res%] + %res%] +
       %dsc$ymin%]

            /* Second, extracting from the point coverage
            /* Yeah, this is redundant but the above was easier to copy
            &s tmpxmax [calc %tmpxmax% + .5]
            &s tmpymax [calc %tmpymax% + .5]
            &s tmpxmin [calc %tmpxmin% - .5]
            &s tmpymin [calc %tmpymin% - .5]
            sel box
            %tmpxmin%, %tmpymin%
            %tmpxmax%, %tmpymax%
            &s totpt [show number selected]
            &if %totpt% gt 9999 &then &do
               &if %autocalcres% = 0 &then &do
                  &ty This square has %totpt% points and
                  &ty exceeds the total number allowed, 9999.
                  &ty Please restart with a smaller cell size.
                  quit
                  &goto cleanup
               &end
               &else &do
                  &s autocalcres 2
                  &goto badres
               &end
            &end
            &else &do
               put xtpt_%x%_%y%
            &end
         &end
      &end
   quit
&end
&else &do
    &if [exists xtpy_0_0 -cover] &then kill xtpy_0_0 all
    &if [exists xtpt_0_0 -cover] &then kill xtpt_0_0 all
    copy xtpy xtpy_0_0
    copy xtpt xtpt_0_0
&end

/* Efficiency in one section requires additions elsewhere...
/* Building the coverages so they are recognized as point and poly
&do x = 0 &to [calc %xreps% - 1]
   &do y = 0 &to [calc %yreps% - 1]
      build xtpt_%x%_%y% point
      build xtpy_%x%_%y%
   &end
```

```
&end

/* Reorder the point and poly coverages in anticipation
/* of later being able to meet S1000 ordering requirements.

/* Sorting first by column then by row
tables
   &do x = 0 &to [calc %xreps% - 1]
      &do y = 0 &to [calc %yreps% - 1]
         additem xtpy_%x%_%y%.pat count 5 5 I
         sel xtpy_%x%_%y%.pat
/*         sort row column
      &end
   &end
quit

/* ============================================
/* Export the clipped poly coverages to MOSS files.
/* ============================================

&do x = 0 &to [calc %xreps% - 1]
   &do y = 0 &to [calc %yreps% - 1]
      &if [exists xtpy_%x%_%y%.moss -file] &then
         &sys rm xtpy_%x%_%y%.moss
      &if [exists xtpt_%x%_%y%.moss -file] &then
         &sys rm xtpt_%x%_%y%.moss
      arcmoss xtpy_%x%_%y% xtpy_%x%_%y%.moss utm lmpair
      arcmoss xtpt_%x%_%y% xtpt_%x%_%y%.moss utm spot # point
   &end
&end

/* ****************************************************
/* Okay, now begins the fun with Perl.
/* ****************************************************

&if [exists xxrowcol.txt -file] &then
   &s junk [delete xxrowcol.txt -file]
&if [exists xxtest.txt -file] &then
   &s junk [delete xxtest.txt -file]

&do x = 0 &to [calc %xreps% - 1]
   &do y = 0 &to [calc %yreps% - 1]
      &s output [open xxrowcol.txt os -write]

/* testing that files needed by the perl script don't exist
      &s junk [delete xtpy_%x%_%y%.txt -file]
      &s junk [delete xtpt_%x%_%y%.txt -file]
      &s junk [delete xtpt.moss -file]
      &s junk [delete %x%_%y%perl.txt -file]

/* creating a text file needed for passing the row and column
/* values to the perl script
      &s junk [write %output% %x%]
      &s junk [write %output% %y%]
      &s junk [close %output%]

      arcmoss xtpt xtpt.moss utm spot # point

/* Calling the perl script
/* The perl script is going to sort and reorder the poly and point
/* moss files, then makes a copy and renames it
      &sys tin3perl > %x%_%y%perl.txt
      &s junk [delete xxrowcol.txt -file]
      &s junk [delete xxtest.txt -file]
      &s junk [delete xxtest2.txt -file]
      &s junk [delete xtpt.moss -file]

/* renaming and copying the output files with the required S1000
/* import naming structure to the specified output destination
      &if %x% lt 10 &then &do
         &if %y% lt 10 &then &do
            &sys cp xtpy_%x%_%y%.txt %dir%/%name%/00%x%_00%y%-
        poly.addwams
            &sys cp xtpt_%x%_%y%.txt %dir%/%name%/00%x%_00%y%-
        point.addwams
         &end
         &else &do
            &sys cp xtpy_%x%_%y%.txt %dir%/%name%/00%x%_0%y%-poly.addwams
            &sys cp xtpt_%x%_%y%.txt %dir%/%name%/00%x%_0%y%-
        point.addwams
         &end
      &end
      &else &do
         &if %y% lt 10 &then &do
            &sys cp xtpy_%x%_%y%.txt %dir%/%name%/0%x%_00%y%-poly.addwams
            &sys cp xtpt_%x%_%y%.txt %dir%/%name%/0%x%_00%y%-
        point.addwams
         &end
         &else &do
            &sys cp xtpy_%x%_%y%.txt %dir%/%name%/0%x%_0%y%-poly.addwams
            &sys cp xtpt_%x%_%y%.txt %dir%/%name%/0%x%_0%y%-point.addwams
         &end
      &end
   &end
&end

&sys cat *perl.txt > perl[date -tag].txt
&sys rm *perl.txt

/* If the program bails out it will come here and clean up
/* before ending.
&label cleanup
&ty xxxxxxxx
&ty Cleaning up...
&do x = 0 &to [calc %xreps% - 1]
   &do y = 0 &to [calc %yreps% - 1]
      &if [exists xclip_%x%_%y% -cover] &then kill xclip_%x%_%y% all
      &if [exists xclipt_%x%_%y% -cover] &then kill xclipt_%x%_%y% all
      &if [exists xtpy_%x%_%y% -cover] &then kill xtpy_%x%_%y% all
      &if [exists xtpt_%x%_%y% -cover] &then kill xtpt_%x%_%y% all
      &if [exists xtpy_%x%_%y%.moss -file] &then &sys rm
        xtpy_%x%_%y%.moss
      &if [exists xtpy_%x%_%y%.txt -file] &then &sys rm xtpy_%x%_%y%.txt
      &if [exists xtpt_%x%_%y%.moss -file] &then &sys rm
        xtpt_%x%_%y%.moss
      &if [exists xtpt_%x%_%y%.txt -file] &then &sys rm xtpt_%x%_%y%.txt
      &if [exists xtpt_%x%_%y%.info -info] &then kill xtpt_%x%_%y%.info
        info
```

```
   &end
&end

&do cov &list xtpoly xtpy xtpoint xtpt xxlabel xxlabel2
   &if [exists %cov% -cover] &then kill %cov% all
&end

&ty [date -VFULL]
&watch &off
&echo &off
display 9999
&return

/*****************************************************************
/* bail routine
/*****************************************************************
&routine bail
&severity &error &ignore
&severity &warning &ignore
&ty xxxxxxxx
&if [show program] = ARCEDIT &then &do
   quit
   y
&end

&if [show program] ne ARC &then quit
&do x = 0 &to [calc %xreps% - 1]
   &do y = 0 &to [calc %yreps% - 1]
      &if [exists xclip_%x%_%y% -cover] &then kill xclip_%x%_%y% all
      &if [exists xclipt_%x%_%y% -cover] &then kill xclipt_%x%_%y% all
      &if [exists xtpy_%x%_%y% -cover] &then kill xtpy_%x%_%y% all
      &if [exists xtpt_%x%_%y% -cover] &then kill xtpt_%x%_%y% all
      &if [exists xtpy_%x%_%y%.moss -file] &then &sys rm
        xtpy_%x%_%y%.moss
      &if [exists xtpy_%x%_%y%.txt -file] &then &sys rm xtpy_%x%_%y%.txt
      &if [exists xtpt_%x%_%y%.moss -file] &then &sys rm
        xtpt_%x%_%y%.moss
      &if [exists xtpt_%x%_%y%.txt -file] &then &sys rm xtpt_%x%_%y%.txt
      &if [exists xtpt_%x%_%y%.info -info] &then kill xtpt_%x%_%y%.info
        info
   &end
&end

&type An error has occured in tin3.aml
&type Bailing out of tin3.aml
&ty [date -VFULL]
&return; &return &error
```

2.2.22 tin3perl

PURPOSE: This perl script reformats ARCMOSS exported files created by tin3.aml. For more specific information, see **tin3.aml**.

HISTORY: This program was originally coded in February 1997.

USAGE: This program is executed from within tin3.aml and is not intended to be used alone.

SAMPLE OUTPUT: A MOSS polygon file:

```
    2                 029-000                       4
  690000.00 5515000.00
  690500.00 5515000.00
  690000.00 5514500.00
  690000.00 5515000.00
    3                 029-000                       4
  690000.00 5514500.00
  690500.00 5515000.00
  690500.00 5514500.00
  690000.00 5514500.00
```

would be converted into an S1000 ADDWAMS polygon file:

```
1              029-000                         4
         690000.00        5515000.00
         690500.00        5515000.00
         690000.00        5514500.00
         690000.00        5515000.00
2              029-000                         4
         690000.00        5514500.00
         690500.00        5515000.00
         690500.00        5514500.00
         690000.00        5514500.00
```

A MOSS point file:

```
    1                            526.67               1
  690000.00 5515000.00
    2                            520.96               1
  690500.00 5515000.00
    3                            554.98               1
  690000.00 5514500.00
```

would be converted into an S1000 ADDWAMS point file:

```
1               526.67     1
         690000.00        5515000.00
2               520.96     1
         690500.00        5515000.00
3               554.98     1
         690000.00        5514500.00
```

PROGRAM FLOW: The program reads in every line of each MOSS file, reformatting them to fit the S1000 ADDWAMS standard. For more information about the S1000 ADDWAMS standard, see the S1000 users manual.

tlss

```
#! /usr/local/bin/perl

# tin3perl
# ----------------------------------------------------------------
#     Program: tin3perl
#     Purpose: to convert moss text files to S1000 ADDWAMS format
#     text files.
#  ----------------------------------------------------------------
#     Called By:  tin3.aml
#     Calls Made:  none
#  ----------------------------------------------------------------
#     Locals: Coordinate names used during the script:
#     pt - point
#     py - polygon
#     hdr - header
#     crd - coordinate
#     $__in - input from a text file
#     a, b, and n are tests or loop counters
#  ----------------------------------------------------------------
#     Inputs: a file called xxrowcol.txt must exist before the
#     script begins; it passes in the row and column of the file
#     to be reformatted.  This file is created by tin3.aml.
#     Outputs: Reformatted poly and point files ready for S1000
#  ----------------------------------------------------------------
#     History:  18 February 1997 - Tobi Steinberg
#     Terrain Simulations - 7th ATC - Grafenwoehr, Germany
#  ----------------------------------------------------------------

# Determine the row and column of the cell we need to work on.
# This file is created in the AML unique for each execution of this
# perl script.
open (ROWCOL, "xxrowcol.txt");
$x = <ROWCOL>;
$y = <ROWCOL>;
chop($x);
chop($y);
close (ROWCOL);

# Open the files needed for input.  PTIN is the input point moss file.
# PYIN is the input polygon moss file.
open (PYIN, "xtpy_$x\_$y.moss");
open (PTOUT, ">xtpt_$x\_$y.txt");
open (PYOUT, ">xtpy_$x\_$y.txt");
open (TEST2, ">xxtest2.txt") || die "Couldn't open TEST2";

# setting variables which will be used in the script
$one = "1";
$four = "4";
$ptcount = 0;
$rejcount = 0;
$nomatch = 0;

# Reading in line from the input polygon file.  If the end of the file
# has not been reached, continue the program.  Otherwise, end.
# Reading in the polygon header.
while ($pyhdrin = <PYIN>) {
   chop($pyhdrin);

   # Reformat the input poly header
   $pycount = substr($pyhdrin, 0, 5);
   $pycount = $pycount - 1;
   $lmpair = substr($pyhdrin, 15, 7);

   # Write it to the poly output file
   if ($pycount < 10) {
      printf PYOUT "%4s %20s %25s\n", $pycount, $lmpair, $four;
   } elsif ($pycount < 100) {
      printf PYOUT "%5s %20s %25s\n", $pycount, $lmpair, $four;
   } elsif ($pycount < 1000) {
      printf PYOUT "%6s %20s %25s\n", $pycount, $lmpair, $four;
   } elsif ($pycount < 10000) {
      printf PYOUT "%7s %20s %25s\n", $pycount, $lmpair, $four;
```

```
   }

   # Reading in the poly coordinates; loop of 4 times
   $n = 1;
   while ($n < 5) {

      # Read in the first poly coordinate
      $pycrdin = <PYIN>; # entire line from input moss file
      chop($pycrdin);

      # Reformat the coordinates
      $pyxcrd = substr($pycrdin, 0, 11); # x coordinate
      $pyycrd = substr($pycrdin, 12, 11); # y coordinate
      chop($pyycrd);

      # Write the coordinates to the output poly file
      printf PYOUT "%18.2f %17.2f\n", $pyxcrd, $pyycrd;

      # only go through the loop if it is not the last coordinate
      # in the file; the fourth is always a duplicate of the first
      if ($n < 4) {
      # Check it against all values in the test file
      # Using the value of b for the tester
      close(TEST2);
      # system("head xxtest2.txt");
      system("cp xxtest2.txt xxtest.txt");
      open(TEST, "xxtest.txt") || die "Couldn't open TEST";
      $testx = "0";
      $testy = "0";
      $b = 0;
      while ($b < 1) {
         if ($testin = <TEST>) {
            $testx = substr($testin, 0, 18);
            $testy = substr($testin, 18, 18);
            unless ($testx != $pyxcrd) {
               unless ($testy != $pyycrd) {
                  $b = 1;
                  # print "Duplicate point: $testx, $testy\n";
                  ++$rejcount;
               }
            }
         } else {
         $b = 2;
         }
      }
      close(TEST);
      open(TEST2, ">>xxtest2.txt");

      # If not present in the test file, then
      if ($b > 1) {

         # Find the coordinates in the input point file
         # First, test to see if the coordinate even exists in the file
         open(PTIN, "xtpt_$x\_$y.moss");
         $a = 0; # the tester for this inside while loop
         while ($a < 1) {
            if ($pthdrin = <PTIN>) {
               $ptcrdin = <PTIN>;

               # pull out x and y coordinates
               $ptxcrd = substr($ptcrdin, 0, 11);
               $ptycrd = substr($ptcrdin, 12, 11);
               chop($ptycrd);

               unless ($ptxcrd != $pyxcrd) {
                  unless ($ptycrd != $pyycrd) {
                     $a = 2; # This signifies a match was found
                     # print "Match: $ptxcrd, $ptycrd\n";
                  }
               }
            } else {
            $a = 1;
```

```
            }
         }
         close(PTIN);

         # if the coordinate does exist, do the following:
         if ($a > 1) {
            # Reformat both the coordinates and header
            ++$ptcount;
            $spot = substr($pthdrin, 6, 21);

            # Write the coordinates to the test file
            printf TEST2 "%18.2f %17.2f\n", $ptxcrd, $ptycrd;

            # Write the point header and coordinates
            # to final output point file

            # Write it to the poly output file
            if ($ptcount < 10) {
               printf PTOUT "%4s %21.2f %5s\n", $ptcount, $spot, $one;
            } elsif ($ptcount < 100) {
               printf PTOUT "%5s %21.2f %5s\n", $ptcount, $spot, $one;
            } elsif ($ptcount < 1000) {
               printf PTOUT "%6s %21.2f %5s\n", $ptcount, $spot, $one;
            } elsif ($ptcount < 10000) {
               printf PTOUT "%7s %21.2f %5s\n", $ptcount, $spot, $one;
            }
            printf PTOUT "%18.2f %17.2f\n", $ptxcrd, $ptycrd;
         }

         if ($a < 2) {
         print STDOUT "No match: $pyxcrd, $pyycrd\n";
         ++$nomatch;
         }
      }
      }
   ++$n;
   }
}

print STDOUT "Report for file xtpy_$x\_$y.txt:\n";
print STDOUT "$ptcount points were written to the output file\n";
print STDOUT "$rejcount points were duplicate\n";
print STDOUT "Matches were not found for $nomatch points\n";
print STDOUT "\n";
close (PYIN);
close (PTOUT);
close (PYOUT);
```

3.0 Available SimNet Textures

Rivers

```
14 rgb          32  32 1 0 8 none   linear 2   1 bellevue/image/water_river.tga
70 rgb          32  32 1 0 8 none   linear 2   1 bellevue/image/water_river_b.tga
123 rgb          64  64 1 0 8 none   linear 2   1 bellevue/image/water_river_c.tga
124 rgb          64  64 1 0 8 none   linear 2   1 bellevue/image/water_river_d.tga
125 rgb          64  64 1 1 8 none   linear 2   1 bellevue/image/water_river_e.tga
```

Lakes

```
43 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/water_lake.tga
126 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/water_lake_b.tga
127 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/water_lake_c.tga
128 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/water_lake_d.tga
```

Roads

```
4 rgb          64  64 1 0 8 none   linear 2   1 bellevue/image/imprv_highway.tga
25 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/imprv_runway.tga
36 rgb          64  64 1 0 8 none   linear 2   1 bellevue/image/imprv_freeway.tga
90 rgb          64  64 1 0 8 none   linear 2   1 bellevue/image/imprv_dirtrd_b.tga
91 rgb          64  64 1 0 8 none   linear 2   1 bellevue/image/imprv_dirtrd_c.tga
92 rgb          64  64 1 0 8 none   linear 2   1 bellevue/image/imprv_dirtrd_d.tga
93 rgb          64  64 1 0 8 none   linear 2   1 bellevue/image/imprv_dirtrd_e.tga
152 rgb          64  64 1 0 8 none   linear 2   1 bellevue/image/imprv_dirtrd_f.tga
153 rgb          64  64 1 0 8 none   linear 2   1 bellevue/image/imprv_dirtrd_g.tga
```

Railroads

```
15 rgb          64  64 1 0 8 none   linear 2   1 bellevue/image/imprv_rr_track.tga
```

Vegetation
Treelines

```
29 rgb         128 128 1 0 8 none   linear 2   1 bellevue/image/veg_treeln5_trns.tga
40 rgb         128 128 1 0 8 none   linear 2   1 bellevue/image/veg_treeln_opaq.tga
```

Tree Stamps

```
30 rgb          64  64 0 0 8 none   linear 2   1 bellevue/image/veg_new_oak.tga
31 rgb          64  64 0 0 8 none   linear 2   1 bellevue/image/veg_beechtree.tga
32 rgb          64  64 0 0 8 none   linear 2   1 bellevue/image/veg_hemlock.tga
```

Vegetation Lines

```
44 rgb          64  64 1 0 8 none   linear 2   1 bellevue/image/veg_brushline.tga
```

Vegetation Stamps

```
33 rgb          64  64 0 0 8 none   linear 2   1 bellevue/image/veg_tall_bush.tga
42 rgb          64  64 0 0 8 none   linear 2   1 bellevue/image/veg_short_bush.tga
```

Canopies

```
57 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/veg_cnp_opaq.tga
64 rgb          64  64 1 1 8 none   linear 2   1 bellevue/image/veg_cnp_trns.tga
121 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/veg_cnp_opaq_b.tga
122 rgb          64  64 1 1 8 none   linear 2   1 bellevue/image/veg_cnp_trns_b.tga
130 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/veg_cnp_opaq_c.tga
131 rgb          64  64 1 1 8 none   linear 2   1 bellevue/image/veg_cnp_trns_c.tga
157 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/veg_cnp_opaq_d.tga
```

Land

```
1 rgb          64  64 1 1 0 none   linear 2   1  bellevue/mip/land_rf37_hds1.rgb
24 rgb         128 128 1 1 0 none   linear 2   1 bellevue/image/land_crop.tga
52 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_grd_dig.tga
54 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_b.tga
60 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_c.tga
72 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_d.tga
77 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_e.tga
78 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_f.tga
79 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_g.tga
80 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_crop_b.tga
81 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_crop_c.tga
82 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_crop_d.tga
83 rgb         128  64 1 1 0 none   linear 2   1 bellevue/image/land_crop_e.tga
84 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_crop_f.tga
98 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_h.tga
```

```
99 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_i.tga
100 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_j.tga
101 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_k.tga
102 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_l.tga
103 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_m.tga
104 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_n.tga
105 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_o.tga
150 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_p.tga
151 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_rf37_hds1_q.tga
154 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_crop_g.tga
156 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/land_grd_dig_b.tga
```

Buildings

```
73 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/bldg_brick.tga
74 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/bldg_paneling.tga
75 rgb          16  16 1 1 0 none   linear 2   1 bellevue/image/bldg_window.tga
76 rgb          16  16 1 1 0 none   linear 2   1 bellevue/image/bldg_door.tga
108 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/bldg_paneling_b.tga
109 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/bldg_paneling_c.tga
110 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/bldg_paneling_d.tga
111 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/bldg_paneling_e.tga
112 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/bldg_paneling_f.tga
113 rgb          16  16 1 1 0 none   linear 2   1 bellevue/image/bldg_window_b.tga
114 rgb          16  16 1 1 0 none   linear 2   1 bellevue/image/bldg_window_c.tga
115 rgb          16  16 1 1 0 none   linear 2   1 bellevue/image/bldg_door_b.tga
116 rgb          16  16 1 1 0 none   linear 2   1 bellevue/image/bldg_door_c.tga
119 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/bldg_brick_b.tga
120 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/bldg_brick_c.tga
129 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/bldg_brick_d.tga
155 rgb          64  64 1 1 0 none   linear 2   1 bellevue/image/bldg_paneling_g.tga
```

Other

```
68 rgb          32  32 1 1 8 none   linear 2   1 bellevue/image/misc_rock2.tga
87 rgb          32  32 1 1 8 none   linear 2   1 bellevue/image/misc_rock2_b.tga
88 rgb          32  32 1 1 8 none   linear 2   1 bellevue/image/misc_rock2_c.tga
89 rgb          32  32 1 1 8 none   linear 2   1 bellevue/image/misc_rock2_d.tga
```

4.0 References

4.1 Documents

Banchero, Jay. TEC Visual Database Construction - Process Summary, Loral Advanced Distributed Simulation, Bellevue WA, for LNK Corporation, Riverside MD, July 1994.

DeHaemer, Michael J. Jr. And Zyda, Michael J, "Simplification of Objects Rendered by Polygonal Approximations," *Computer & Graphics*, Vol. 15 No. 2, pp. 175-184, 1991.

Earth Systems Resource Institute (ESRI), Arc/Info Online Help / Manual, Arc/Info version 7.0.4, 1996.

Farsai, Steve and Gafford, Durwood. S1000 Data Base Application Programmer's Interface (API) User's Manual, Loral Advanced Distributed Simulation, Bellevue WA, for LNK Corporation, Riverside MD, 93040 v.1; August 1995.

Hiatt, Steven, ED. IRIS Performer Programmer's Guide. Silicon Graphics Computer Systems, Mountain View CA, Document number 007-1680-030, 1995.

Hubbard, Mike, "Exporting Arc/Info Data to Support Interoperable Synthetic Environments," *Proceedings: 1997 Arc/Info Users Conference*, 1997.

LNK Corporation. Bosnia SIMNET Terrain Database Design Document, for the U.S. Topographic Engineering Center (TEC), October 1993.

Lockheed Martin Federal Systems. S1000 Tool Set User's Manual, Version 1.6.2.4.

McKeown, David M. Jr. and Chi Tau Lai, Robert. Integrating Multiple Data Representations for Spatial Databases, Carnegie-Mellon University Department of Computer Science, Pittsburgh PA.

McKeown, David M. Jr. And Lukes, George E. Building a Cartographic Infra-Structure: Simulation and Cartography, Carnegie-Mellon University Department of Computer Science, Pittsburgh PA, and the U.S. Army Topographic Engineering Center Research Institute - Autonomous Technologies Division, Fort Belvoir VA, for the U.S. Army Topographic Engineering Center (TEC), Ft. Belvoir VA, and Wright Research and Development Center, Aeronautical Systems Division (AFSC), Wright-Patterson AFB OH, 1991.

McKeown, David M. Jr., et. al. Progress in Automated Virtual World Construction, Carnegie-Mellon University Department of Computer Science - Digital Mapping Laboratory, Pittsburgh PA, January 1996.

McKeown, David M. Rapid Construction of Virtual World - Interim Program Review, Carnegie-Mellon University Department of Computer Science - Digital Mapping Laboratory, Pittsburgh PA, for the Terrain Modeling Project Office (TMPO), November 1996.

Novak, Jason and Jennings, Joe. Use of Computer Image Generators in Distributed Simulation Exercises. The MITRE Corporation.

Papelis, Yiannis E. Terrain Modeling on High-Fidelity Ground Vehicle Simulators, University of Iowa Center for Computer Aided Design, Iowa City IA, 1994.

Patton, Erik and Baumgartner, David. The Joint Tactical Simulation (JTS) Terrain Building Model (JTBM), Cubic Technical Services International - Terrain Simulation, Grafenwoehr Germany, for 7th Army Training Command, Grafenwoehr Germany, and the Terrain Modeling Project Office (TMPO), Washington D.C., June 1996.

Polis, Michael F., Gifford, Stephen J., and McKeown, David M. Jr. Automating the Construction of Large Scale Virtual Worlds, Carnegie-Mellon University School of Computer Science - Digital Mapping Laboratory, Pittsburgh PA, for the U.S. Army Topographic Engineering Center (TEC), Ft. Belvoir VA, and Wright Research and Development Center, Aeronautical Systems Division (AFSC), Wright-Patterson AFB OH.

Polis, Mike and McKeown, David M. Jr. Iterative TIN Generation from Digital Elevation Models, Carnegie-Mellon University School of Computer Science - Digital Mapping Laboratory, Pittsburgh PA, for the U.S. Army Topographic Engineering Center (TEC), Ft. Belvoir VA, and Wright Research and Development Center, Aeronautical Systems Division (AFSC), Wright-Patterson AFB OH, October 1991.

Polis, Michael F., Gifford, Stephen J., and McKeown, David M. Jr. Automating the Construction of Large Scale Virtual Worlds, Carnegie-Mellon University School of Computer Science - Digital Mapping Laboratory, Pittsburgh PA,

Polis, Mike and McKeown, Dave. Integrated TIN Generation User's Manual, Carnegie-Mellon University School of Computer Science - Digital Mapping Project, Pittsburgh PA, MAPS Document M100 revision 1.1, November 1996.

Schroeder, Will and Citriniti, Tom, "Decimating Polygon Meshes," *Dr. Dobb's Journal*, July 1997.

Seventh Army Training Center Battle Simulations Support Branch. USAREUR Simulations Primer, United States Army Europe, May 1995.

4.2 Personal References

Caldwell, Doug. U.S. Topographic Engineering Center (TEC), Fort Belvoir VA.
Donaldson, LTC Bruce. National Imagery and Mapping Agency (NIMA), Fairfax VA.
Gardner, Nancy. U.S. Topographic Engineering Center (TEC), Fort Belvoir VA.
Haes, Steve. BDM International, Inc., McClean VA.
Lasch, Tom. Simulation, Training and Instrumentation Command (STRICOM), Orlando FL.
Polis, Michael. Carnegie-Mellon University School of Computer Science - Digital Mapping Laboratory, Pittsburgh PA.
Powers, John. Raytheon TI Systems, Plano TX.
Reith, Ernie. National Imagery and Mapping Agency (NIMA) Terrain Modeling Project Office (TMPO), Bethesda MD.
Salemann, Leo. Lockheed Martin Advanced Distributed Simulation (LADS), Bellevue WA.

4.3 Presentations

Foley, Paul. Rapid Preparation of Simulation Databases, MITRE for the Defense Modeling and Simulation Office (DMSO), for Terrain Week 1997, San Antonio TX, January 1997.

McKeown, Dave, and Polis, Mike. Recent CMU TIN Developments, Carnegie-Mellon University School of Computer Science - Digital Mapping Laboratory, Pittsburgh PA, for Terrain Week 1997, San Antonio TX, January 1997.

Miller, Dale, Miller, Timothy, and Cornish, Charlie. Development of Large GCS Terrain Data Bases in Arc/Info and S1000, Lockheed Martin Advanced Distributed Simulation (LADS), Bellevue WA, for Terrain Week 1997, San Antonio TX, January 1997.

Miller, Timothy. DCR400 Terrain Database, Lockheed Martin Advanced Distributed Simulation (LADS), Bellevue WA, for Terrain Week 1997, San Antonio TX, January 1997.

Olson, Warren, Caldwell, Doug, and Clover, Bob. Operation Kirby Review, Institude for Defense Analysis (IDA) and the U.S. Army Topographic Engineering Center (TEC), Fort Belvoir VA, for Terrain Week 1997, San Antonio TX, January 1997.

Sull, Young Suk. Rapid Mapping / Simulation Requirements, National Imagery and Mapping Agency (NIMA) Terrain Modeling Project Office (TMPO), Bethesda MD, for Terrain Week 1997, San Antonio TX, January 1997.

Witzgall, Christoph, Damron, James J., and Miller, Timothy. Combined High Resolution Delta Corridor and Range 400, National Institute of Standards and Technology (NIST), U.S. Topographic Engineering Center (TEC), Fort Belvoir VA, and Loral Advanced Distributed Simulation (LADS), Bellevue WA, for Terrain Week 1997, San Antonio TX, January 1997.

5.0 Points of contact

Terrain Simulation (TerraSim), a Logicon lab
Dave Knox, TerraSim manager

World Wide Web Homepage	http://usa.7atc.army.mil/
Telephone Number	DSN 474-3111 Civilian 011 49 9641 833111
Fax Number	Civilian 011 49 9641 550
U.S. Mailing Address	TerraSim attn: Dave Knox Logicon TSI Unit 28130 APO AE 09114
Overseas Mailing Address	TerraSim attn: Dave Knox Gebaeude 1460 Camp Aachen Truppenuebungsplatz 92655 Grafenwoehr, Germany

Paper Authors (can also be reached at the above numbers and addresses)

M. Peri Cope	copem@email.grafenwoehr.army.mil
Michael Paul Czechanski	czech@email.grafenwoehr.army.mil
Tobi L.S. Sellekaerts	steinbet@email.grafenwoehr.army.mil

W. Bruce Bollinger, the SCOR for TerraSim at USAREUR 7th Army Training Command

Telephone Number	DSN 474-2396 or 2460 Civilian 011 49 9641 83 2396 or 2460
Fax Number	DSN 474-2541 Civilian 011 49 9641 832541
U.S. Mailing Address	HQ 7th ATC Attn: AEAGC-TS-F Unit 28130 APO AE 09114
Email Address	bollingb@hq.7atc.army.mil