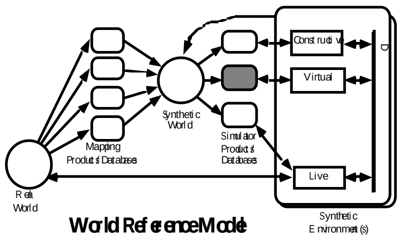
ARPA War Breaker

World Reference Model



WRM Entity Flight Specification

Entity Models for Distributed Interactive Simulation (DIS) Interoperability

Version 1 Draft 10
September 26, 1994

Prepared for
The Advanced Research Projects Agency
War Breaker Program
Systems Engineering and Modeling Contract

By
Science Applications International Corporation
Dan Brockway
dbrockway@wb.com

With most of the work done by:

Michael Bienvenu Len Granowetter Carl Suttle

bien@paradigmsim.com
(214) 960-2301

lengrano@mak.mak.com
(617) 876-8085

multigen!carl@uunet.UU.NET
(408) 247-4326

WRM Entity Flight Specification:

Entity Models
for
Distributed Interactive Simulation (DIS)
Interoperability

WRM Entity Flight Specification:

Entity Models

for

Distributed Interactive Simulation (DIS)

Interoperability

I. Purpose

Historically, control of models (e.g., motion, articulation, and special effects) requires supporting software familiar with the model's structure. For closed applications and few models, this is not a problem; but in DIS environments with hundreds of model types, it would be more convenient to adopt uniform conventions for various features of models (like the origin and orientation of the local coordinate system), and to embed as much other supporting information as possible (like the DIS entity code), so that models can be selected and controlled by software with little or no knowledge of the model itself.

A further particular feature of DIS entities(models) is that they assume various appearances based on states; a tank may have its hatch in any of several positions, may appear in various camouflages, and may incur various types of damage. All of these states are invoked by data in DIS Protocol Data Units (PDUs) in accordance with published specifications (Refer to Section 2 Applicable Documents, DIS Standards). Managing the various geometries and textures required to create these varying appearances should be possible, again, without the need for software having any particular knowledge of the model.

This document describes a methodology for the creation of three-dimensional models for use in the War Breaker WRM Distributed Interactive Simulation (DIS) environment. Compatible operation is achieved with models created using **MultiGen Inc.**'s (*née* Software Systems) ***MultiGen*** model development system, and rendered using either **Paradigm Simulation's *Vega*** simulation environment or **Mäk Technologies *Stealth*** renderer. Both Vega and MäK Stealth are Silicon Graphics Inc. Performer-based applications. Vega uses a custom loader, while MäK Stealth uses the existing Performer loader, which MultiGen maintains for Silicon Graphics. Compatible operation thus requires a cooperative effort among Paradigm, MäK, and MultiGen; this document is a reflection of that effort, as well as guidance from SAIC and significant contributions by others. The document is maintained by Paradigm, who welcomes comments and suggestions.

The approach described considered three primary criteria:

1. The MultiGen Flight format version 14.1 file (*.flt) structure hierarchy must be modular, flexible, and suitable for conversion to an optimal scene graph in Performer.
2. The file structure must provide a means for control of the model's *representation* (its textures and material properties) and its *state* (geometry).
3. The file structure must be self documenting regarding conformance to DIS standards.

I. Applicable Documents

SAIC War Breaker

World Reference Model (WRM) Architecture Release 2 Draft 8, July, 1994
WRM Entity Architecture, Release 1 Draft 10, September, 1994

DIS Standards:

Tactical Warfare Simulation & Technology Information Analysis Center, Institute for Simulation and Training, University of Central Florida (TWSTIAC-IST):

Application Protocols:
- IST-CR-94-50 *Draft Standard* (Version 2.0 Fourth Draft, Revised)
- IST-CR-94-18 *Draft Standard* (Version 3.0, Working Draft)
- IST-CR-93-03 *Rationale Document* (accompanies Version 2.0)
- IST-CR-93-46 *Enumeration & Bit Encoded Values*

Communication Architecture Requirements:
- IST-CR-94-15 *Draft Standard*
- IST-CR-93-42 *Rationale Document*
- IST-CR-94-17 *Guidance Document*

Exercise Control and Feedback Requirements:
- IST-CR-94-12 *Draft Standard*
- IST-CR-94-10 *Rationale Document*

Fidelity Description Requirements:
- IST-CR-94-13 *Draft Standard*
- IST-CR-93-34 *Rationale Document*

Paradigm Simulation , Inc.

Vega User's Manual, Version 1.0
Vega LynX User's Manual, Version 1.0

MäK Technologies, Inc.

VR-Link, The Virtual Reality Networking ToolKit, Release 2.2.3, 1994

MultiGen, Inc.

ModelGen Modeler's Guide, Revision 13.0, April 1993
MultiGen Modeler's Guide, Revision 14.0, March 1994
MultiGen Flight Format Modeler's Guide, Revision 12.0, December 1992

Other

Department of Defense World Geodetic System 1984, Its Definition and Relationships with Local Geodetic Systems; Defense Mapping Agency Technical Report 8350.2, 1987

I. Background

A. Overview of the MultiGen Flight^{TM} format

1. Groups, Objects, Faces, Levels of Detail, and Degree of Freedom beads

MultiGen's Flight file format uses a straightforward system to organize geometry within the file, consisting several types of *beads*.

- *Face beads* represent individual polygons.
- *Object beads* represent face beads which are logically or geographically related. Object beads may have only face beads (polygons) as children.
- *Level of detail (LOD) beads* provide a means to select alternate (simpler) geometries based on a user-definable range. (In MultiGen, LOD switch ranges are set manually by eye. The Vega *Automatic LOD* feature, if enabled, recomputes switch ranges based on object size in the image plane [accounting for field of view, model size, model range, and image resolution] overriding the ranges set in MultiGen.)
- *Degree of Freedom (DOF) beads* are placed in the data base to specify translational and/or rotational freedom for portions of a model. They can be used for DIS articulated parts or animated special effects. They specify a type of transformation and its limits, such as "rotate about the x axis between 45 and 135 degrees". MultiGen's approach to degrees of freedom is hierarchial. Consider the example of a fingertip modeled as part of a hand. The fingertip is attached to the second segment of a finger, the second segment connected to the first segment, and the first segment attached to the hand. The fingertip is free to move within limits expressed with respect to a local coordinate system. This means you can set limits on the fingertip without regard to those imposed on the rest of the finger, or on other items attached above it in the hierarchy. Conversely, transformations accumulate as you move down the tree, so rotations and transformations on the hand as a whole are automatically taken into account before those affecting the fingertip are considered.
- *Group beads* represent logically-related or geographically-related object beads. Group beads are the most general, and may have any type of beads, except face beads, as children.

1. External references and instancing

Two MultiGen constructs provide a means to avoid duplicating massive amounts of geometry:

- An *external reference* (or *xref*) is a pointer to another Flight file containing additional geometry. A transformation matrix is usually attached to the xref to locate the position and orientation of the referenced object's coordinate system.
- *Instances* are used when two or more groups have children which share the same geometry. Each instance may have a transformation matrix controlling the position and orientation of the child's coordinate system. This allows, for example, the same house to be used at various locations and with various orientations on the terrain. (If flattened as a result of the selected optimization level, (See Section 3.2.4), the database size increases as the geometry is duplicated, transforming the vertices at each instance; this is faster at runtime, but may result in a much larger database.)

1. Bead name and comment fields

Each bead may be given a name, currently limited to seven characters.

Most beads also contain a comment field (the comment is created within the bead dialog box, though it actually creates a separate *comment bead* which is not graphically represented in the hierachy mode) for user-definable data. The comment field is used to provide an implementation of the features described in this document.

A. An overview of the Performer Scene Graph

In contrast to the Flight format's *beads*, the Performer scene graph consists of various types of *nodes*.

1. DCSs and Articulations

- A d*ynamic coordinate system* (DCS) node is a transformation applied to all children of the node. It allows, for example, the real-time articulation of moving parts. A model's top-level DCS controls the position and orientation of its coordinate system.
- An *articulation* is any movable or detachable part of a model which lends itself to representation by a DCS node. Each articulated part should be represented by a single DCS node, which in turn will be the parent of that part's LOD nodes and representation or state switch nodes. Because only a single DCS may be used to represent an articulation, an articulation DCS should not be a descendant of an LOD or geometry switch node higher in the model. An articulation DCS is a descendant of the DCS associated with the part of the model to which the articulation is attached. The parent part may be the root of the model (e.g., the turret's parent is the tank), or another articulated part (e.g., the barrel's parent is the turret).

2. LOD nodes

LOD nodes allow Performer to provide alternate representations of objects, and are commonly used to switch in simpler geometries at further ranges. LOD nodes, like LOD beads, may have group nodes, geodes (objects), and other LOD nodes as its children.

Performer uses range checking to switch models from high to low resolution based on the ranges set in the MultiGen LOD beads.

1. Switch nodes

Performer uses switch nodes to select among children of the node. Vega uses these nodes specifically to select from among alternate representations of the object. Each entity or part requires a switch node for each type of attribute capable of multiple appearances.

1. Tree optimization

When MultiGen Flight files are loaded into Performer, the loader optimizes the files for run time based on the optimization level set in the file. Two general levels of optimization are *clean* and *partition*. *Clean* culls all empty or unnecessary groups and geodes out of the tree. *Partition* creates a user specified rectangular grid system. The two dimensional grids contain a list of all the polygons in a particular grid. This is used to optimize intersection routines. A third optimization is *flatten*. *Flatten* only occurs for geometry for which there is no DCS; it then replicates the geometry of the object at each instance and transforms all object coordinates to world coordinates.

A. Functions of the Vega or Performer Loader

Both the Vega and Performer loaders translate MultiGen Flight files to Performer scene graphs by creating Performer nodes from MultiGen beads. The table below summarizes the relationship between the Performer nodes and the MultiGen beads.

MultiGen Flight format	Performer Node Type
group bead	pfGroup
animation sequence in group bead	pfSequence
combination of LOD beads	pfLOD
polygon bead with children (subfaces)	pfLayer
draw as light flag in polygon bead	pfLightPoint
object bead	pfGeode
axis rotation flag in polygon bead	pfBillboard
group bead with @dis inherit	pfSCS or pfDCS
DOF bead	pfDCS
group bead with @dis switch	pfSwitch

I. Modeling for DIS Interoperability

Techniques for modeling for the DIS environment are divided into three groups:

- Adoption of uniform standards for locating and orienting the model coordinate system
- Techniques which assist in the mapping from DIS codes to associated models and parts;
- General methods for eliminating individual support software for models

These features will be implemented within the existing Flight file structure, primarily using the bead comment fields.

A. DIS Attribute Lexicon (DAL)

The following lexicon is used to embed DIS attributes in the MultiGen Flight file format when creating a model for DIS Interoperability:

1. DAL Syntax

The DAL syntax is simple and straightforward:

@dis	**keyword**	**value**
Main keyword	Particular DIS attribute being modeled	specific attribute information either a single entry, or a comma separated list, or a range of values (state keyword). Note the motion keyword requires min and max values to follow the keyword value.

1. DAL Keywords

@dis - main keyword that precedes all other keywords. It also denotes the termination of a free form comment.

model_spec_version - identifies the version level of this document used to create the models.

entity_type - identifies the model with a unique DIS Entity code.

attachment_station - identifies the location on the parent model where externally referenced child models may be placed via a unique DIS Attached Part code.

attach_entity - identifies the model(s) that may be attached to an attachment_station via a unique DIS Entity code(s).

attachment_point - identifies the location where the externally referenced child model will be attach to the parent model’s attachment stations via x, y, z offsets.

articulated_part - identifies the moveable parts of a model with a unique DIS Articulated Part code.

motion - identifies the type and range of motion for an articulated part.

switch - identifies a model appearance attribute with a DIS keyword.

state - determines the actual state of the appearance attribute with a DIS appearance state code.

inherit - identifies an instance where texture, material, or LODs should be inherited from the parent model.

weapon_effect - identifies the location of a weapon fire special effect.

dcs - forces Performer to create a dynamic coordinate system node.

scs - forces Performer to create a static coordinate system node.

# - delineates the end of an **@dis** field and the start of a free form comment that continues until the next **@dis** keyword. Comment fields will be interpreted sensitive to case.

1. DAL Keyword Values

@dis - takes another DAL keyword as its value.

model_spec_version - takes the version number on the front cover as its value.

entity_type - takes the numeric values in DIS Standard IST-CR-93-46 Enumerations and Bit Encoded Values Section 4.2.3 Comprehensive Entity-Type tables separated by colons (:) as its value.

attachment_station - takes a numeric value in DIS Standard IST-CR-93-46 Enumerations and Bit Encoded Values Section 4.7.1 Attached Parts as its value. It may optionally take the default specifier -1 as its value to indicate that any non exact match is to be attached to this location.

attach_entity - takes the numeric values in DIS Standard IST-CR-93-46 Enumerations and Bit Encoded Values Section 4.2.3 Comprehensive Entity-Type tables separated by colons (:) as its value.

attachment_point - takes the x, y, and z offsets from the model's origin that uniquely identify the attachment point.

articulated_part - takes the numeric value in DIS Standard IST-CR-93-46 Enumerations and Bit Encoded Values Section 4.7.2 Articulated Parts as its value.

motion - takes a value from Table 4.3.3.2 to identify the kind of motion followed by minimum and maximum range values.

switch - takes the name value in DIS Standard IST-CR-93-46 Enumerations and Bit Encoded Values Section 4. 3 Entity Appearance (General and Specific), joined by the underscore character (_) if the name is multiple words, as its value. Note that the following names (frozen_status, power_plant_status, and state) are preceded by an abbreviation for their domain as follows: gm_ for guided munitions and lf_ for life forms. Refer to Table 4.3.5.

state - takes the numeric value defined in DIS Standard IST-CR-93-46 Enumerations and Bit Encoded Values Section 4. 3 Entity Appearance (General and Specific) Purpose field, as its value. Refer to Table 4.3.5. and Section 4.3.5 for a description of a value list for this keyword.

inherit - takes one of the following reserved names as its value: LOD, material, or texture.

weapon_effect - takes the numeric values in DIS Standard IST-CR-93-46 Enumerations and Bit Encoded Values Section 4.2.3.2 Munitions separated by colons (:) as its value. Note that more than one munition entity type maybe specified in a comma separated list.

dcs - none required.

scs - none required.

# - takes user defined alphanumeric text as its value.

1. DAL Implementation

@dis:

The keyword "*@dis*" is used at the beginning of each line in the comment field to signal the Vega and Performer loaders of a feature specific to DIS modeling. Other key words and values follow.

model_spec_version:

This keyword is placed in the parent model header as well as in any externally refereneced models' headers. Refer to Section 4.3.1.

entity_type:

This keyword is placed in the parent model header as well as in any externally refereneced models' headers. Refer to Section 4.3.1.

attachment_station:

This keyword is placed in a Group bead in the parent model. Refer to Section 4.3.2.1.

attach_entity:

This keyword is placed in the same Group bead as the attachment station in the parent model. Refer to Section 4.3.2.1.

attachment_point:

This keyword is placed in a Group bead in the parent model if it is an attachable model. Refer to Section 4.3.2.2.

articulated_part:

This keyword is placed in a Group bead that is a child to the DOF bead that identifies an articulated part. This keyword is always followed by the motion keyword described below. Refer to Section 4.3.3.2.

motion:

This keyword is placed in a Group bead that is a child to the DOF bead that identifies an articulated part. Refer to Section 4.3.3.2.

switch:

This keyword is placed in a Group bead in the parent model as well as in a Group bead in any externally refereneced models. Refer to Section 4.3.4.

state:

This keyword is placed in the child of a Group bead in the parent model as well as in a child of a Group bead in any externally refereneced models. Refer to Section 4.3.5.

inherit:

This keyword is placed in the state's comment bead as appropriate. Refer to Section 4.3.6.

weapon_effect:

This keyword is placed in a Group bead that is a child to the DOF bead that identifies the special effect location. Refer to Section 4.4.1.

dcs:

This keyword is placed in a Group bead in the parent model or in a Group bead in an externally refereneced model whenever a DCS node must explicitly be used. Refer to Section 4.5.

scs:

This keyword is placed in a Group bead in the parent model or in a Group bead in an externally refereneced model whenever a SCS node must explicitly be used. Refer to Section 4.5.

#:

The pound sign "#" is used to stop the interpretation of an *@dis* field, and allows a free-form comment to follow. Any new line not starting with "*@dis*" will be ignored, and may be used for comments. Examples:

"*@dis* model_spec_version 001 # designed to version 001 of the model spec"

"*@dis entity_type* 1:1:225:3:1:1:0 # M88A1 Armored Utility Vehicle,

only armored vehicle in this exercise"

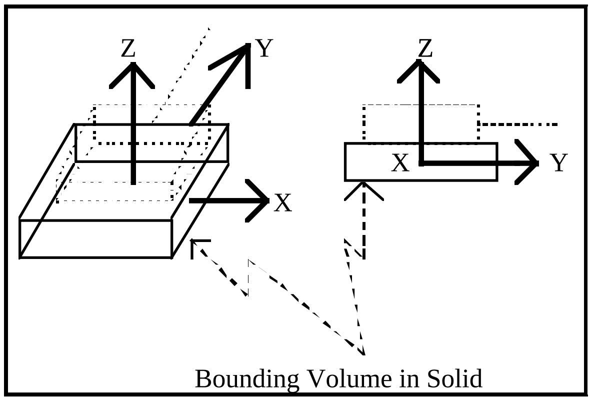
A. Model Coordinate System

1. Orientation of the model coordinate system

Performer and MultiGen, use a *different* right-hand coordinate system from that of the DIS Standard, namely positive z up, x right, and y forward. **The following is an intentional, specific deviation from the DIS standard intended to make the modeler's use of MultiGen intuitive with respect to a model's coordinate system:**

Ignore the DIS model coordinate system. Construct models using the Performer/MultiGen coordinate orientations (z up, x right, y forward), with a one meter unit length.

WRM MODEL COORDINATE SYSTEM



The real-time software will perform the translation:

$$x_{model} = y_{DIS}$$
$$y_{model} = x_{DIS}$$
$$z_{model} = -z_{DIS}$$

1. Origin of the model coordinate system

Locate the origin of vehicles as appropriate from the following table:

Domain	***Location of model origin***
Land	*On the plane of ground contact, at the axis of yaw rotation*
Air	*At the center of gravity*
Surface	*On the axis of the center of buoyancy, in the plane of the waterline for an unloaded ship*
Subsurface	*On the axis of the center of buoyancy, in the plane of the waterline of a surfaced submarine*
Space	*At the center of gravity*

A. Modeling for Efficient Mapping

1. Comment the parent model header

The parent model is the base geometry without any external references. The parent model header is comprised of two sequential comments. The first is "**@dis model_spec_version value**", where value is the version number of this document to which the model was designed. The second is "**@dis entity_type code**", where code is found in the DIS Standard IST-CR-93-46 *Enumeration & Bit Encoded Values* Section 4.2.3 Comprehensive Entity-Type tables. Note each field is separated by colons. Entity is the DIS terminology for a model.

For example, comment the parent model header of an M88A1 as follows:

> *"@dis model_spec_version 001*
> *@dis entity_type 1:1:225:3:1:1:0"*

The entity type is derived from the DIS specifications:

1	=	Entity kind	=	'Platform'
1	=	Domain	=	'Land'
225	=	Country	=	'USA'
3	=	Category	=	'Armored utility vehicle'
1	=	Sub-category	=	'M88 Medium Recovery Vehicle'
1	=	Specific	=	'M88A1'
0	=	Extra	=	'other'

1. Attached parts

In a DIS environment, an entity may have one or more attached parts. Attached parts are separate entities that are "attached" to another entity. For instance, a missle launcher may have an attached missle prior to launch, an aircraft may have bombs attached prior to its bombing run, etc.

a) Comment attached parts stations in primary models

An attached part is placed at a particular attachment station on the primary entity. One may think of an attachment station as a hanger, from which an attached part may be hung.

When modeling an entity which may have other entities attached to it, (e.g. a missle launcher), one must indicate where the attachment stations are in the entity's geometry, and at what orientation the attached part model should be placed.

Stations are numbered sequentially starting with one. The order of station numbering is from top to bottom, then back to front, then left to right, except for aircraft wing stations, which are numbered from inboard to outboard. The attached part stations are defined in DIS Standard IST-CR-93-46 *Enumeration & Bit Encoded Values* Section 4.7.1 Attached Parts.

Attachment station information is encoded in the primary model as follows:

*Place a Group bead in the primary model for each attachment point. Create a minimal polygon at 0, 0, 0. Translate this polygon to the attachment station's x, y, z offset from the origin of the model. Rotate the polygon in the yaw, pitch, and roll axes such that if any geometry were to be placed below this group, it would have the proper orientation for the attached part. Comment the Group bead in the primary model for each attachment point as "**@dis attachment_station value list**", where* value list *is of the form* ***{number[, number]}*** *where* number *is the four-digit DIS code found in the* DIS Standard IST-CR-93-46 *Enumeration & Bit Encoded Values* Section 4.7.1 Attached Parts *for the station you are identifying. If the optional default specifier* -1 *follows an attachment_ station number, it will be used as the attachment stationt for any attached part which does not have an exact match.* For example, comment the Group bead for the fuselage centerline attachment point:

"@dis attachment_station 512, -1 # fuselage centerline station; also use for anything else that doesn't match"

For a missle launcher, the point on the launcher's geometry where a missle can be attached has a group bead with the following comment field:

"@dis attachment_station 1 # station 1"

A comment of the form "**@dis attach_entity code**", *where* code *is found in the* DIS Standard IST-CR-93-46 *Enumeration & Bit Encoded Values* Section 4.2.3 Comprehensive Entity-Type tables, *is placed below the @dis attachment_station comment in the group bead. Additional codes may be placed in a comma separated list. Note each field is separated by colons*. For the aircraft example, comment the Group bead:

"@dis attach_entity 2:9:225:1:15:3:0 #Mk-84 bomb PAVEWAY II laser-guided"

For the missle launcher:

"@dis attach_entity 2:1:225:1:15:1:0, 2:1:225:1:16:0:0 #FIM-92 Stinger A/B/C or MIM-104 Patriot"

a) Comment the attachment point in attachable models

When modeling an entity that can be attached to another entity, one must indicate what point on its geometry should coincide with the location of the attachment station on the entity to which this entity may attach. Think of this point as the point to be hung from the hanger (attachment station) defined above in section 4.3.2.1.

The attached part will by default be oriented parallel to the main model's longitudinal (y) axis. If any rotation or transformation is required to attach the attaching point to the attachment point station, it will be done by the primary model.

The location of the attatchment point is specified by a comment in the root bead of the model in a line following the "@dis entity_type ... " comment of the following format:

"@**dis attachment_point x y z**" *where* x, y, and z *are the offsets from the model's origin that uniquely defines the attachment point.* For the aircraft example, comment the root bead in the bomb model:

"@dis attachment_point 0 0 0.75 # bomb attachment point x = 0m, y = 0m, z = 3/4m"

For the missle launcher, comment the root bead in the missle model:

"@dis attachment_point 0 0 -0.5 #missle attaching point"

This indicates that when this missle type is attached to a primary model, the point of attachment is half a meter below the entity's origin.

1. Articulated Parts

Articulated parts are the moveable parts of a model, such as the turret on a tank, the periscope on a submarine, or the landing gear on an aircraft. They may be attached to the model or to another articulated part.

a) Representing Articulations in MultiGen's Hierarchy

DOF beads are used to represent an articulated part.

There must be exactly one DOF bead for each articulated part in the model. If there are geometry switches, or multiple levels of detail associated with a particular articulated part, these beads must appear below the DOF bead for the articulated part.

If the articulated part is attached to another articulated part rather than to the base geometry of the model (e.g. a gun is attached to a turret, whereas a turret is attached to the tank body [base geometry]) then that articulated part's DOF must be a descendant of the parent articulated part's DOF. In this case, the gun's DOF must be a descendant of the turret's DOF.

The articulated part and DOF should be positioned such that the articulated part is in its neutral or initial position when the DOF bead attributes represent the identity transformation.

b) Embedding DIS Articulation Data in the Model

The DOF bead does not have a comment field. Thus in order to identify which articulated part the DOF bead controls and to define the articulated part's motion parameters, a

Group bead is placed as a child to the articulated part's DOF bead. All of the geometry that defines the articulated part is placed under this Group bead. The DIS specific model features are identified by comments in this Group bead as defined below:

The articulated part's Group bead comment field may consist of several lines, each starting with the "@dis" keyword. The first line is a DIS Articulated Part Type Specification. The additional lines are DIS Articulated Part Motion Type Specifications, each assigned to a single line. There must be a DIS Articulated Part Motion Type Specification for each motion that is allowable for the articulated part.

The DIS Articulated Part Type Specification takes the form ***"@dis articulated_part type",*** *where* type *is the four digit DIS enumeration for the Articulated part. Refer to* DIS Standard IST-CR-93-46 *Enumeration & Bit Encoded Values* Section 4.7.2 Articulated Parts *for* type *values. For example, comment the group bead for the turret of an M1 tank:*

"@dis articulated_part 4096 # primary turret"

A DIS Articulated Part Motion Type Specification takes the form ***"@dis motion kind min max",*** *where* kind *is the type of motion allowed for an articulated part (see Table 4.3.3.2 below);* min *is the minimum position/orientation deviation from the neutral position/orientation; and* max *is the maximum position/orientation deviation from the neutral position/orientation. The min/max constraints on translational parameters are in units of meters, and they are in units of degrees for rotational parameters. Refer to the following examples:*

```
1) "@dis articulated_part 4096       # primary turret M1 tank"
   "@dis motion azimuth 0.0 360.0 # primary turret can only rotate in azimuth;
                                  # within the range of 0 to 360 degrees"

2) "@dis articulated_part 4416         # primary gun M1 tank"
   "@dis motion elevation -30.0 90.0 # gun may change its elevation within  a range
                                     # of -30 to +90 degrees only"
   "@dis motion Y -0.6 0.3            # gun may change its position along the
                                     # Y axis for recoil within a range
                                     # of -0.6 to +0.3 meters only"

3) "@dis articulated_part 3072 # left main landing gear F15E"
   "@dis motion Y  0.0 1.0        # gear may change itsY, X, and Z positions
   "@dis motion X -1.0 1.0       # with the indicated constraints
   "@dis motion Z 0.0 2.0        # in meters
```

Table 4.3.3.2 DIS Articulated Part Motion Type Specification Kind Values

Kind Value	**DIS cross reference**	**Description**
Y	DIS X	Forward / Backward motion (Model Y axis)
X	DIS Y	Left / Right motion (Model X axis)
Z	-DIS Z	Up / Down motion (Model Z axis)
Azimuth	-Azimuth	Heading (rotation with respect to Model Z axis)

Elevation	Elevation	Pitch	(rotation with respect to Model Y axis)
Rotation	Rotation	Roll	(rotation with respect to Model X axis)

Table 4.3.3.2

Note that the DIS coordinate system differs from the MultiGen/Performer coordinate system such that in DIS, x is forward, y is right, and z is down

1. **Use commented group beads to control model appearance**

*Use Group beads to allow multiple variations in a model's appearance. Comment the Group bead as "**@dis switch attribute**" where* attribute *is the DIS Entity Appearance Record attribute the switch controls, e.g., "@dis switch paint_scheme". Refer to Table 4.3.5 DIS Appearance Attributes and States for the complete list of appearance attributes.*

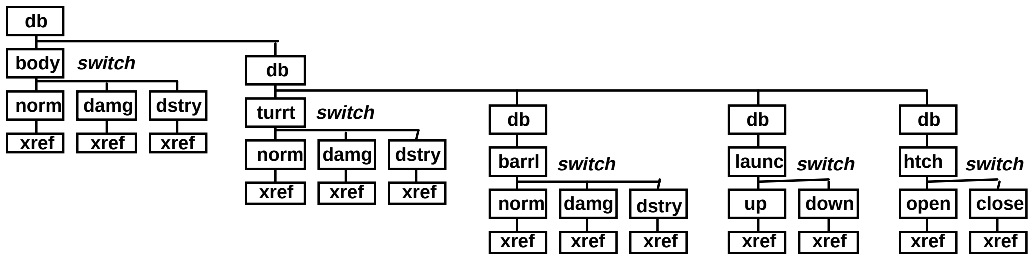
Group beads identified with the @dis switch keyword allow different model appearances without resulting in an explosion of beads or an uncontrollable proliferation of files, and can be used to switch geometry, texture, or material:

- A *geometry switch* is an attribute switch whose children are different states of a part (e.g., a normal, damaged, or destroyed turret). If more than one type of representation of the part entity is also possible, texture switch (see below) may be used which share the same children. (e.g., desert or forest camouflage applied to a normal, damaged, or destroyed turret).
- A *material switch* allows one set of geometry to have multiple alternate material properties (e.g., a forest green vs. desert tan base coat). This avoids duplicating the model and its corresponding parts for each material.
- A *texture switch* allows selection from among multiple textures (e.g., desert vs. forest camouflage) for a single geometry.

The Vega and Performer loaders will translate Group beads with the @dis switch keyword to Performer pfSwitch nodes.

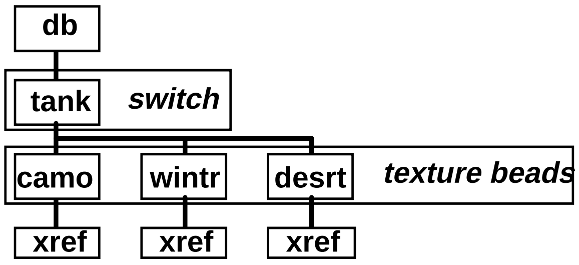
The reserved names for the switch attributes are taken from the DIS Entity Appearance Record specification. The switch attribute names are shown in **Table 4.3.5**.

The following example shows how a tank with multiple appearance attributes, states, LODs, and parts should be structured.

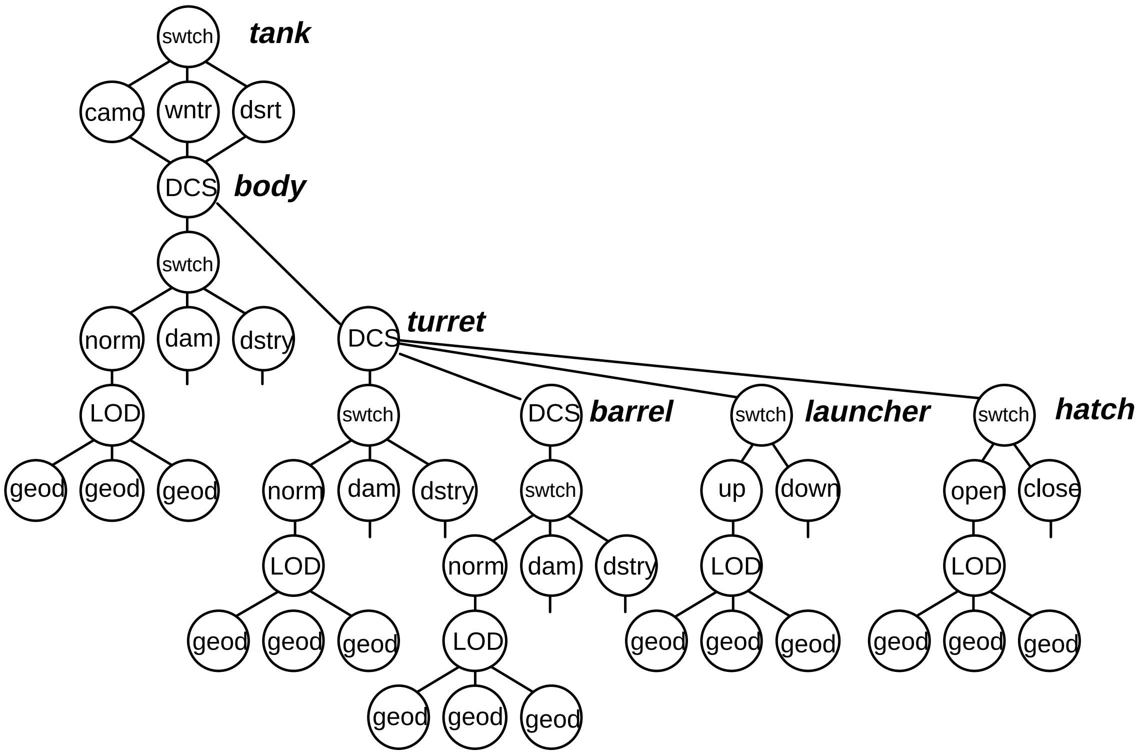


In this figure, the turret switch file is attached to the body switch file, while the barrel, launcher, and hatch switch files are attached to the turret. This allows the turret to be positioned in the coordinate system of the body, and the barrel, launcher, and hatch to be positioned in the coordinate system of the turret. In this example, all external references inherit the texture and material tables of their parents.

If multiple color schemes are needed to represent camouflage, winter, and desert versions, another switch is needed. The previously defined tank would be referenced three times, with each reference attached to a texture attribute bead. The following figure illustrates the hierarchy needed to accomplish this:



By using the concept of switch files and inheritance, the database consists of thirteen unique data files referenced by six switch nodes. This enables the tank to assume 324 different appearances with a minimum of duplication. When converted to a Performer scene graph, the tank would have the following representation:



While inheritance and attribute switches provide a higher level of functionality, they also require the modeler to be conscious of the interaction of all branches of a hierarchy. For example, if a part of a model does not inherit its parents' attributes, attribute switches higher in the tree will also go unnoticed and may create undesired effects. By the same token, if a part contains different textures than its parent but inherits texture data, a texture switch may leave the part unaffected.

In order to reduce the number of existing permutations it is suggested that all of a model's state files (e.g., normal, damage, destroyed) use the same texture data. Additional textures and different materials may be added to reflect degrees of damage, but the base texture should remain the same. With this structure, a texture change may occur at the highest level in the hierarchy and be reflected in all representations, states, and parts.

1. **Comment the children of attribute switches the DIS state represented**

*Comment the child of a Group bead identified by the @dis switch keyword, which represents one of two or more alternate states as "***@dis state value list",** *where* value list *is one or more of the defined DIS states in the Entity Appearance Record, or the default child indicator "-1".* Value list *is of the form {m[,m]}, where m is the value -1, a single non-negative integer* i, *or an integer range* i-j. *In addition, set the LOD, texture, and material inheritance flags via comments in the bead as appropriate (See section 4.2.6.).*

Geometries can be combined appropriate to the application. For example, not all hatch states may need to have unique visual states:

Parent switch commented *"@dis switch hatch"*

Child commented *"@dis state -1,1"* (used for the closed state and any other states not explicitly defined by child bead comments)

Child commented "@dis state 2-3" or *"@dis state 2,3"* (used for either popped state)

Child commented *"@dis state 4,5"* (used for either open state)

If a switch does not have a match for a DIS PDU state, and no default is explicitly defined, the child with the lowest defined value shall be used.

The reserved names and values for these states are taken from the DIS specification for Entity Appearance Records, and shown in **Table 4.3.5.** The appearance attributes are used as the attribute switch values, and the state numbers are used as described above to comment the children.

Table 4.3.5. DIS Appearance Attributes and States.

Domain	Appearance Attribute	State	Description
General	paint_scheme	0	uniform color
		1	camouflage
	mobility	0	no mobility kill
		1	mobility disabled
	fire_power	0	no firepower kill
		1	fire power disabled
	damage	0	no damage
		1	slight
		2	moderate
		3	destroyed
	smoke	0	not smoking

		1	smoke plume
		2	engine smoke
		3	smoke plume and engine smoke
	trailing_effects	0	none
		1	small
		2	medium
		3	large
	hatch	0	(not applicable)
		1	closed
		2	popped
		3	popped with visible person
		4	open
		5	open with visible person
		6	(not used)
		7	(not used)
	lights	0	none
		1	running
		2	navigation
		3	formation
		4	(not used)
		5	(not used)
		6	(not used)
		7	(not used)
	flaming	0	none
		1	flames
Specific	**All platforms, regardless of domain:**		
	frozen_status	0	not frozen
		1	frozen [do not dead reckon]
	power_plant_status	0	power plant off
		1	power plant on
	state	0	active
		1	deactivated
Land	launcher	0	not raised
		1	raised
	camouflage_type	0	desert
		1	winter
		2	forest
		3	(unused)
	concealed	0	not concealed
		1	concealed
Air	afterburner	0	off
		1	on
Surface	(none currently defined)		
Subsurface	(none currently defined)		
Space	(none currently defined)		
Guided Munitions	launch_flash	0	no launch flash

		1	launch flash
	gm_frozen_status	0	not frozen
		1	frozen [do not dead reckon]
	gm_state	0	active
		1	deactivated
Life Forms	life_form_state	0	not defined
		1	upright, standing still
		2	upright, walking
		3	upright, running
		4	kneeling
		5	prone
		6	crawling
		7	swimming
		8	parachuting
		9	jumping
		10-15	(not used)
	lf_frozen_status	0	not frozen
		1	frozen [do not dead reckon]
	lf_state	0	active
		1	deactivated
	weapon_1	0	no primary weapon present
		1	primary weapon stowed
		2	primary weapon deployed
		3	primary weapon in firing position
	weapon_2	0	no secondary weapon present
		1	secondary weapon stowed
		2	secondary weapon deployed
		3	secondary weapon in firing position
Environmentals	density	0	clear
		1	hazy
		2	dense
		3	very dense
		4	opaque
		5	(not used)
		6	(not used)
		7	(not used)
Cultural Features	(none currently defined)		
Supplies	(none currently defined)		
Radios	(none currently defined)		

1. **Use inheritance flags to capture LOD, texture, or material from the parent**

In the example shown previously, the turret and barrel should inherit the textures, color, and material of the body of the tank.

Set the associated inheritance flags in the external reference to the parent for each attribute (LOD, texture, or material) that the child is to assume from the parent:

"@dis inherit texture"
"@dis inherit material"
"@dis inherit LOD".

A. Features for Independent Models

1. Tag the location of weapon fire special effects

Place a DOF bead at the location of the weapon fire special effect. Comment the Group bead which is a descendant of the DOF bead ***"@dis weapon_effect code list****", where* code list *consists of the DIS munition IDs fired from this location.*

For example:

- Place a DOF bead at the end of the 50 caliber machine gun barrel on the turret of an M1A2, and label the descendant Group bead

  "@dis weapon_effect 2:2:225:2:1:-1:-1, 2:8:225:2:5:-1,-1 # Munition, Anti-Armor, US, Ballistic, 50 cal, any type of round; also Munition, Anti-Personnel, US, Ballistic, 50 cal, any type of round"

- Place a DOF bead at the end of the main gun barrel, and label the descendant Group bead

  "@dis weapon_effect 2:2:225:2:13:1:-1:-1 # Munition, Anti-Armor, US, Ballistic, 120 mm, any type of round"

1. Tag the location of marking text (future)

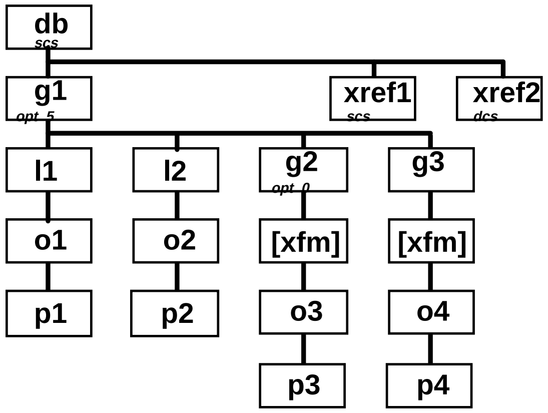
[need a tag in the file which specifies the location, orientation, text font, and text size for PDU marking text; e.g., on the ship's stern and bow]

1. Model infrared or radar signatures (future)

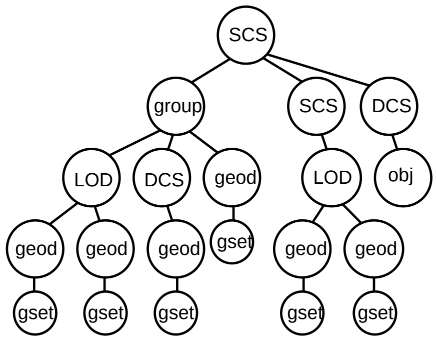
[need methodology to handle Infrared & Radar Signatures in the model geometry]

A. Managing Optimization of the Performer Scene Graph

The structures shown in the figure below show the correspondence between a typical MultiGen file and the Performer tree which is created when it is loaded.



MultiGen Hierarchy



Performer Tree

Note that the group bead labeled *"opt 0"* becomes a DCS while the other group, g3, is flattened and its group is culled from the tree by optimization. The modeler may force the creation of a DCS in three different ways:

- Comment the group bead *"@dis dcs"* or *"@dis scs"* to force the creation of a DCS or SCS node respectively.
- If the model is attached to another model as an external reference, the loader will create either an SCS or a DCS, depending on the xref's header
- If the loader finds a transformation matrix in a MultiGen bead whose optimization level is such that its hierarchy is not flattened, it will create a DCS

The following rules should govern DCS and SCS nodes in the construction of a tree:

1. A parent object or parent part always exists in its *own* coordinate system, whether an SCS or a DCS.
2. Flags and properties (*@dis* fields) may be placed anywhere in the MultiGen file hierarchy
3. If a property is set in a bead in the MultiGen hierarchy, it will only effect the bead's children, not its parent(s) or siblings.
4. A DCS is never removed or flattened by the Performer or Vega loaders, no matter what level of optimization is set.
5. An SCS may be flattened (culled from the Performer tree) if a high enough optimization level (clean, flatten) is specified in the MultiGen comment field where the SCS is defined.