ABSTRACT: The Synthetic Environment Data Representation Interchange Specification (SEDRIS) is a powerful tool for promoting interoperability between environment representations of distributed simulation applications. The SEDRIS data model defines environment objects for all domains (land, air, ocean, space) and all types of simulation applications, including two- and three-dimensional visualization systems and computer generated forces (CGF) systems. This paper addresses the use of SEDRIS for generating CGF terrain databases, particularly Compact Terrain Database (CTDB) representations for ModSAF. We discuss our experience with generating CTDBs from various sources using the SEDRIS application programmer interfaces (APIs). We address the types of data required in SEDRIS transmittals for CGF terrain databases. Also, we discuss how the data could be organized in a SEDRIS transmittal to minimize memory usage and traversal time during compilation. We discuss the use of associations between geometry and feature objects in the SEDRIS hierarchy for efficient CGF terrain database processing. Finally, we describe our progress to date in generating CTDBs for the Army Stow A experiment and the conversion of the Close Combat Tactical Trainer (CCTT) SAF database into CTDB using SEDRIS.

1. Introduction

This paper describes the results of the effort to build a compiler for the Compact Terrain Database (CTDB) [1] that receives Synthetic Environment Data Representation Interchange Specification (SEDRIS) data as input. CTDB is the terrain database format for the ModSAF Computer Generated Forces (CGF) system. A previous paper [2] described the initial effort, which used SEDRIS data derived from the Lockheed Martin S1000 terrain database generation system. This paper focuses on work to expand the compiler to accept SEDRIS data from other sources.

The goal of this project was to be able to generate the CTDB compiler so that it could extract data from any valid SEDRIS transmittal. For a variety of reasons, as we will discuss in this paper, this is difficult to achieve. The SEDRIS project has produced a standard data model and application programmer’s interface, which have gone a long way in standardizing the types of Synthetic Natural Environment (SNE) data and how they are organized [3, 4]. However, the data producers, who implement the interface between the format in which the data is produced in and SEDRIS, must still make decisions about data organization. These decisions do have an effect on how the data can be extracted by data consumers, and have to be taken into account when designing applications like compilers that need to extract SNE objects from SEDRIS transmittals.

The CTDB compiler described in [2] adopted a design that was intended to minimize the effect of data organization decisions made by the data producers, as shown in the system block diagram in Figure 1. The compiler performs an initial traversal of the data to determine, among other things, what types of objects are present in the transmittal and how they may be organized. This metadata is not stored in the SEDRIS transmittal, and because of its specialized nature, is not intended to be included in future versions of SEDRIS. The second traversal performs the actual compilation. It was designed to focus on those SEDRIS objects that have counterparts in
the CTDB format and filter out objects in the hierarchy above that level. The data from the SEDRIS objects are extracted and stored in temporary files that are then used for the actual compilation. A mapping of SEDRIS to CTDB objects can be found in [5].

After successfully compiling terrain data derived from S1000 databases, the compiler was tested using SEDRIS transmittals derived from other database formats. One transmittal that the compiler was tested against was derived from a Lockheed Martin Data Base Generation System (DBGS) database [6]. A second transmittal used the data derived from an Evans and Sutherland EaSIEST General Data Format (GDF) database [7]. Although the compiler design was intended to minimize the effects of different data organizations, testing against these new SEDRIS transmittals showed that design decisions made by the data producers still affected the ability of the compiler to extract data. This paper will discuss what those decisions were, and how they affected the compiler.

2. SEDRIS Standardization
   Advantages

Without the SEDRIS effort, using the same compiler to extract data from more than one terrain database format would be extremely difficult. This section describes how SEDRIS makes this possible through standardization. Standardization at the syntactic level is described first; then standardization at the semantic level is discussed.

2.2 Syntactic Standardization

The syntactic level consists of the data structures and subroutine calls needed to access SNE data. Prior to SEDRIS, consumers of SNE data needed to be familiar with the details of the data format that they were using, e.g. S1000. Details varied with each data format depending upon its degree of optimization for either data storage size or for access speed. Application programmer’s interfaces (APIs) were also designed to work...
efficiently with the data format and tended to mirror their idiosyncrasies.

SEDRIS provides a standard library of data objects. The data in each object is organized in accordance with the SEDRIS standard, rather than the native format of the data. This allows a user to extract information about an object without knowing the details of the native data format. For example, a user can read the location of an object without knowing how the coordinate values are encoded in the native data format.

SEDRIS has also standardized how data objects are organized within a transmittal. The organization of the data for as simple an entity as a polygon can vary widely. For example, polygons can be encoded as lists of vertices, as lists of edges, as triangular irregular networks (TINs), where polygons can either list a vertex, or list a reference to a vertex in another polygon, or as even more elaborate structures. Decoding each of these schemes may be simple. However, requiring a consumer to recognize which scheme is being used and to provide an algorithm for each scheme adds considerable complication to the consumer’s task. By limiting data organization to one (or a few) standard scheme, SEDRIS makes it feasible for a consumer to provide code that can handle most of the data organizations that an application is likely to encounter.

2.3 Semantic Standardization

Standardizing the syntactic level is a necessary condition for a common interface. However, it is not sufficient. The user also needs to know what a SEDRIS data object represents in the environment. Without this information, the semantics of the data, the user is unable to incorporate the data into a simulation. The semantic level is concerned with what the SNE data represents in the real world. The user needs this information in order to model the synthetic environment accurately. In systems such as S1000, the method for determining what an object represents is specific to the application programmer’s interface (API). The access method might involve data returned with the object, the specific API sequence of calls used to access the object, or a combination of the two. This variety of methods forces the consumer to accommodate the API rather than the API accommodating the consumer.

SEDRIS has begun to standardize the semantics of the SNE data. A data object, classification data, can be associated with SEDRIS geometry and feature objects. The classification data object contains a code that describes what the SEDRIS object represents. The code values are defined in the SEDRIS Data Coding Standard (SDCS) [8]. SEDRIS Classification Codes (SCC), which are defined in the SDCS, tie SEDRIS objects to objects in the real world. SCCs are five character codes, and are a superset of the classification codes in the DIGEST Standard Feature and Attribute Coding Catalog (FACC) [9]. The SDCS has extended FACC to cover objects that need to be represented in synthetic environments but are not part of FACC, since FACC covers features that need to be represented on maps specifically. Thus the SCC in the classification data object describes what the SEDRIS object represents.

SEDRIS is also standardizing the semantics of how data is represented. Like the SCC, the SEDRIS Attribute Codes (SAC), which are also defined in the SDCS, extend the attribute codes described in FACC. The SAC allows data producers to store information about environmental features in a standard manner.

The CTDB compiler relies heavily on the use of the SCC to determine what an object represents. The compiler uses this information to decide how to process the object and where to store the resulting information in the CTDB. This reliance on classification data values is necessary because SEDRIS transmittals are not completely standardized. The same data structure can be used to encode a building or a tree, for instance. Since these two objects are treated differently by the CTDB compiler, the compiler must be able to distinguish between them. Some areas where an inability to distinguish between object types can cause problems for the data consumer are discussed in the following section.

3. SEDRIS Challenges

The standardization process begun by SEDRIS has come a long way and needs to continue. SEDRIS is allowing the simulation community to use SNE data in ways that were not anticipated when that data were originally generated. As a result, new standardization questions are emerging. This section will identify and discuss several of these challenges.
3.1 Data Organization Hierarchy

The SEDRIS data model organizes data hierarchically. Primitive objects are grouped into unions, called aggregates. These unions are in turn grouped into larger aggregates. At the top of this hierarchy is the synthetic environment object. The strategy for grouping primitive objects within aggregates and for grouping aggregates within the hierarchy is not defined by SEDRIS. Instead, data producers are allowed to use whatever strategy is most convenient or appropriate. SEDRIS objects may be grouped by location (spatially indexed, perimeter related, separating plane related, or quadtree unions), by object type (classification related unions), by state (state related, time related, or level of detail related unions), or by no strategy at all (generalized unions of features or geometry). Grouping strategies may also be mixed, with different strategies used at different levels of the SEDRIS hierarchy.

The variety of data organizations that are available under SEDRIS presents the data consumer with some important decisions in the design of a data extraction strategy. A highly efficient strategy can be developed by taking advantage of the data organization. For example, transmittals derived from S1000 data tend to be organized spatially first and then by classification code within the spatial aggregate. This permits the data consumer to extract data from a small area efficiently. However, other transmittals may and do use different data organization schemes. The strategy used for extracting data for the first transmittal might not be efficient or might even fail on the second transmittal’s data organization. A generic strategy that ignores the data hierarchy is less efficient and may still fail due to unconscious assumptions about the data hierarchy.

![Diagram](image.png)

Figure 2: Possible Locations for Storing Classification Data for Models
A possible solution to the variety of data organizations is to incorporate discriminators for the various types of unions into the SEDRIS search tools. This is already done for spatially organized data, and to a lesser extent, for state related data. Extending these discriminators to other types of unions would relieve the consumer of the burden of designing a search strategy that navigates the SEDRIS data hierarchy efficiently in all cases.

The flexibility of the SEDRIS object hierarchy is matched to a lesser extent by flexibility in where particular data objects are stored. For example, the classification data for an object can be stored as a component of the object, or as a component of one of the object’s aggregate objects. It can also be stored as a link object if one of the object’s aggregating objects is a classification related union. The classification data can also be stored in an associated object using any one of the above methods. Figure 2 illustrates this variety of storage locations for classification data when a geometry model instance is stored. In the figure, solid lines indicate a component-aggregate relationship, and dashed lines indicate an association. Link objects are identified by a perpendicular line joined to a component-aggregate relationship. In this example, a geometry model instance is associated with a point feature and a geometry model. Classification data can be stored as optional data in any of the locations indicated in the figure.

3.2 Alternate Data Representations

SEDRIS allows the data producer to encode synthetic environment data in different ways. For example, elevation data can be encoded as grids or as terrain polygons. Less obviously, a feature, such as a tree, can be encoded as a model, a point feature with attributes of height and width, or a group of polygons. This flexibility in encoding synthetic environment objects is convenient for data producers. However, each new encoding scheme requires the data consumer to develop a new decoding scheme. The flexibility in data encoding is limited to some extent by SEDRIS business rules [10]. The rules prohibit some encoding schemes and limit the use of others to particular applications. However, these rules could be expanded to place the emphasis on environmental objects and how they are encoded within SEDRIS, rather than just on the SEDRIS data objects.

The SEDRIS data model also contains many places where data is optional. Optional data can provide additional information about an object that is important to certain data consumers. In many cases, the data has been specified as optional because there are contexts in which requiring the data to be present does not make sense. There are other cases where the presence of the data would make sense and be useful to data consumers. However, the data is not necessary for a complete description of the synthetic environment, for instance a boundary volume for a geometry model.

Another type of optional data in SEDRIS is the association. Often an object in the synthetic environment will have characteristics of both a feature and geometry. In this case, the SEDRIS data model allows the data producer to encode the geometric aspects of the object in the geometry hierarchy and the feature aspects in the feature hierarchy. The SEDRIS data model also allows the data producer to create an association between the geometry and the feature objects so that the data consumer can process the two SEDRIS objects as a single object of the synthetic environment. However, the association between the two SEDRIS objects is optional and is not always provided.

When data is optional, a consumer must be ready for the case where the data has not been furnished. This is the general case, since it will work even if the data is present, when it can just be ignored. However, the consumer must then decide whether the extra code needed to handle the specific case of the data being present is worth the additional development effort. Not adding the extra code reduces the value of the data when it is present and makes it less likely that future data producers will provide the optional data. More work is needed to determine when the added value of an optional data item to the consumer is sufficient to justify requiring that data.

4. CGF Challenges

The SEDRIS challenges described in the preceding section are being addressed by the SEDRIS developers as more SEDRIS producer
and consumer interchange experiments are taking place. This section describes specific challenges we have found consuming SEDRIS transmittals for generation of Computer Generated Forces (CGF) terrain databases.

4.1 Merging Data

The challenge for Computer Generated Force applications is to be able to take advantage of SEDRIS to build synthetic environments using data sources that may not have been considered suitable previously. Many of the reasons for these databases not being suitable have been removed through the standardization that SEDRIS has accomplished. However, the fact that a CGF compiler can access a SEDRIS transmittal does not guarantee that all of the data needed to generate a CGF database is contained in the transmittal. Also, while the data may exist and may be accessible via SEDRIS, the data needed for a particular CGF object may reside in more than one data object, or even more than one SEDRIS transmittal.

One example of data which reside in multiple objects and which then need to be merged occurs when a high resolution elevation grid overlaps a lower resolution grid, where the terrain polygons are not provided as part of the SEDRIS transmittal. The elevation values of the lower resolution grid that are in the overlap region are intended to be superseded by the high resolution elevations. These superseded elevation readings must be identified and deleted from the resulting terrain database by the CGF application that is creating the database. Also, while the data may exist and may be accessible via SEDRIS, the data needed for a particular CGF object may reside in more than one data object, or even more than one SEDRIS transmittal.

Determining which values are needed for constructing the seam polygons from gridded elevation data is another task that may need to be performed by the data consumer without any assistance from SEDRIS. In cases like the one given above, where data from more than one SEDRIS object must be merged into a single CGF object, the CGF application must perform multiple tasks. It must first identify that such a merge is necessary. It must then locate the SEDRIS objects in their SEDRIS transmittals. Finally it must extract the appropriate data, and merge it.

A related problem occurs in the CTDB compiler, which stores elevation and mobility data together. Unfortunately, the source from which the CTDB is compiled may not store them together. For example, a DTED elevation grid may be the source of the elevation data, and a VPF theme may be the source of the mobility data. In this case, the two types of data will be in different SEDRIS transmittals. The CTDB compiler will need to be able to recognize that the elevation and mobility data must be merged from different objects.

4.2 Data Organization Standards

The CTDB compiler had several problems when reading SEDRIS transmittals derived from GDF and DBGS data. Several of these problems were described previously. This section will describe them in more detail and discuss what might be done to avoid such problems in the future.

Classification data was missing on some objects that the CTDB compiler needed to include in the CTDB. This occurred primarily with geometry models and geometry model instances. It is possible that the classification was included on a point feature but that the association between the point feature and the model instance was missing. Classification data must be extractable from all objects that the compiler needs to include in the CTDB.

Individual buildings were transmitted as point or areal features. The current CTDB compiler expects buildings to be geometry model instances. The models are then compiled as volume models in the CTDB. Additional code needs to be written so features can be included in the CTDB as volume models.

Forests were transmitted as areal features. The CTDB compiler also expects that the forest canopy will be transmitted as a set of polygons that contour the underlying terrain, which is the representation in S1000. An areal feature can be broken up into triangular polygons, but the resulting set of canopy polygons is not unique. The heights of points on the forest canopy will then depend on how the areal feature was broken up into triangles as well as the height of the terrain skin. (It is assumed that the forest has a constant height that is specified in a SEDRIS Attribute Code.)
Some terrain polygons lacked mobility information. This information was present in the transmittal, either as textures attached to the polygon or in areal features. In the first case, the data producer needs to map the soil textures to mobility data. In the second case, the areal features may not completely cover the terrain skin, so a complete mapping to terrain polygon mobility data may not be possible. In that case, the CTDB compiler must be modified to recognize that there is incomplete mobility data in the terrain polygons. The CTDB compiler might also be modified to perform the mobility data mapping from areal features. Whether the data producer or the data consumer should perform this mapping is an issue that needs further study.

Aggregate models were not part of the S1000-derived transmittals that were used for the original development of the CTDB compiler. The CTDB data format does include aggregate models but they have never been implemented. More work is needed to determine how to compile aggregate models.

The compilation of SEDRIS models into CTDB volume models needs additional work. The scheme for S1000 used bounding volumes of model components to build a roofline for the CTDB. Since bounding volumes are rectangular parallelepipeds, using them as the basis for the CTDB volume models limits the types of rooflines that can be compiled. However, the complexity of geometric surfaces that are possible in a model makes the development of a robust algorithm for compiling volume models very difficult. For any algorithm that can be devised, some geometric configuration will cause that algorithm to fail. Placing restrictions on the geometry of the models in SEDRIS transmittals will be necessary to insure reliable compilations into a CTDB.

5. Conclusion

SEDRIS has brought the representation of synthetic environments to the point where the use of different data sources in an application has become much easier. However, development of the SEDRIS standard and its products needs to continue to evolve. The flexibility of the SEDRIS data model is necessary for SEDRIS because it must accommodate many different types of synthetic environment data. However, this flexibility introduces challenges for an application, like the CGF compiler, to use SEDRIS. The number of possible ways that a particular piece of data may be encoded in the data model may become unwieldy. In that case, restricting the possible data encoding schemes through business rules in SEDRIS or additional standards may be necessary.

Some optional data may also need to be made mandatory for CGF applications. For example, classification data, while optional in SEDRIS, is necessary for CGF applications. How a CGF application uses a SEDRIS data object depends more on what that object represents in the synthetic environment than on how it is represented in SEDRIS. SEDRIS encodes what an object represents through its classification data. It may not make sense to make classification data mandatory for all SEDRIS transmittals but it may need to be made mandatory for CGF data.

While SEDRIS has removed many problems using data from multiple producers, it cannot solve all of the CGF terrain database generation challenges. In these cases, it is still up to the CGF application developer to develop the necessary software and technology.

6. References


Author Biographies

DR. VICTOR SKOWRONSKI is a Senior Member of the Technical Staff at TASC, where he investigates terrain representation for CGF systems. Prior to joining TASC in 1996, he did research in solid modeling at Rensselaer Polytechnic Institute, where he also earned a PhD in Computer Engineering. Victor earned a M.E. and a B.E. in Electrical Engineering from Stevens Institute of Technology. He is a licensed Professional Engineer in New York and Massachusetts.

THOMAS STANZIONE is the manager of the Computer Generated Forces Section at TASC. His interests include data representation for terrain reasoning and terrain database generation for simulation applications. Mr. Stanzione has a Masters of Science Degree in Photographic Science from the Rochester Institute of Technology.