Proceedings From The 1997 Fall Simulation Interoperability Workshop

Rapid Terrain Visualization: Prototyping Synthetic Natural Environments for CGF and Visualization

Alan Evans, Howard Lu, Tod Shannon
SAIC

Richard Rybacki
MetaVR

Victor Skowronski
TASC

ABSTRACT: *The mission of the Rapid Terrain Visualization (RTV) ACTD is to provide Digital Terrain Data (DTD) to the warfighter for intelligence and analysis, mission planning, and course of action analysis. RTV datasets will be archived and disseminated by the Battlefield Awareness and Data Dissemination (BADD) system. The applications using RTV data include ABCS applications such as MCS and the prototype visualization application BPV (Battlefield Planning Visualization). Low cost visualization applications are also currently under development for use in RTV. In addition, CGF simulation applications will be used for force-on-force modeling in support of COA analysis. RTV will provide complex data using semi-automated feature extraction techniques under tight time constraints. The RTV Transform Team is tasked with providing software tools to teams in the field that convert the data archived and disseminated by BADD into usable CTDB datasets for CGF simulation, as well as runtime datasets for low-cost visualization applications.*

1 Background

The mission of the Rapid Terrain Visualization (RTV) ACTD is to provide Digital Terrain Data (DTD) to the warfighter for intelligence, situation awareness, mission planning, and course of action analysis. These data will be derived from standard NIMA products as well as from collection assets tasked to respond to crisis situations in areas of interest around the world. RTV applications and datasets will be integrated into the Battlefield Awareness and Data Dissemination (BADD) system. The software using RTV data will be ABCS (Army Battle Command System) applications such as MCS and the prototype visualization application BPV (Battlefield Planning Visualization). CGF simulation applications will be used for force-on-force modeling in support of COA analysis. In addition, low cost visualization applications are also currently under development. RTV will provide complex data under very tight time constraints. The RTV Transform Team is tasked with providing software tools to the Topo teams in the field, which convert the data archived and disseminated by BADD into usable CTDB datasets for CGF simulation, as well as runtime datasets for low-cost visualization applications.

2 Statement of the Problem

Building terrain databases for visualization and simulation applications has historically been a time-consuming and costly process. Databases originate in NIMA products such as DTED, DFAD, ITD and PITD. Typically, an 18-24 staff-month effort has been required to produce a 90km x 90km playbox of STOW-like complexity. This timeline mostly consists of data fusion, generalization and enhancement performed in a COTS GIS environment. Currently, global coverage exists only at very coarse levels of resolution. Thus, exercises need to be planned far in advance to allow for the collection and processing of data at higher levels of resolution. Recent experiments, such as Topo Force XXI have made progress in shortening the timelines for certain types of database production. However, running simulations on short notice in areas of interest other than US military training sites is not yet a reality. The stated timeline requirements for the RTV ACTD are: 20km x 20km in 18 hours, 90km x 90km in 72 hours, and 300km x 300km in 12 days.

3 Technical Approach

An innovative approach is needed to meet the ambitious goals of the Transform Team. Put simply, the approach is to take the existing compiler for building CTDB databases, and replace the “front end” with software which directly imports elevation and feature data in standard NIMA format. In its place, we provide a software environment for generalization, enhancement, and application-specific value-added processing. A “back end” has been added as well which exports the database in the MDB format used by the VRSG (Virtual Reality Scene Generator) low-cost visualization application which has been developed by MetaVR. An underlying assumption is that cross-theme correlation and data fusion occur before feature data are ingested. In what follows, we explore the architectural decisions underlying the Transform Team technical approach.

3.1 Build on SEDRIS

The first architectural decision made by the Transform Team was to use SEDRIS (Synthetic Environment Data Representation Interchange Specification) for ingest of source data. The SEDRIS object model has been a guide throughout and SEDRIS software is used for ingest. Specifically, the SEDRIS Level 0 API for VPF format data is used to read feature data in the initial assembly phase. This Level 0 API has been designed to work with VPF products such as DTOP and VITD. The RTV feature extraction team will supply feature data in VPF format, as specified by MEDS (Mission Essential Data Set) For elevation data, a SEDRIS Level 0 API has been developed by the RTV Transform Team which provides DTED data at varying levels of resolution. This DTED SEDRIS interface is a prototype implementation of the SEDRIS object model for 2D gridded data sets. The RTV DTED SEDRIS Level 0 API currently is compatible with DTED Level I and II NIMA products, as well as DTED Levels III-V in ERDAS format.

3.2 Push Vs. Pull

In STOW legacy CTDB compiler software, within each GCS cell, the top-level code construct is a main loop that iterates over all patches in the cell. Within the main loop, there is a “pull” of data from the available S1000 input, making calls to the S1K API. In sequence, function calls extract features by type, such as microterrain, roads and buildings. This is inherently inefficient, since the organization of data in the S1000 is not optimized

to support the types of queries made by the compiler. Data structures get processed over and over again, as filtering takes place to implement the S1K API calls in response to the "pull" mechanism. The RTV Transform Team believes that data should be "pushed" from available sources into a form that is organized to support the kinds of queries which will be made by the compiler software. We have designed and are built an Interim DataBase (IDB) format to implement this architectural belief. IDB is implemented as a C++ class library that allows for efficient retrieval of geospatial data by the compiler "middle end." Inheritance is used to make available the exact types of attributed data required by the CGF and visualization applications downstream. The Transform Team architecture will allow for easy extensibility to other forms of source data. By isolating the point in the data stream where code will be added for a pull from a new data source, the amount of coding will be kept to a minimum. The push of source data is especially well-suited to the use of SEDRIS. The SEDRIS model is flexible enough to allow features and attributes to be stored in many different ways. By opening a transmittal, then traversing and filtering the data based on classification and attribution, the need for a layer of software implementing source data queries on persistent store is removed.

3.3 Persistence of Intermediate Data

A second fundamental architectural principle underlying the work of the RTV Transform Team is to maintain data in an intermediate format that is persistent. IDB is a repository where all source data are assembled. An important requirement of the RTV program is to support incremental compilation of simulation databases. From an operational perspective, background elevation data at a coarse level of resolution and perhaps some feature data may be all that is available in a rapid deployment or mission rehearsal. Adding higher resolution data and more feature classes will inevitably be an iterative process as collection assets are tasked and feature extraction take place. The persistent store capability enables compilation that is incremental both spatially and thematically. The software that implements IDB processing is being built to be tolerant of system failure and will recover and restart processing from periodic checkpoints. This is required to insure system reliability when faced with large and complex data sets. The IDB persistent store enables this fault-tolerance.

3.4 3D Visualization and Modification of Geometry and Attributes

Our approach to building the Synthetic Natural Environment (SNE) starts with a complete triangulation of the terrain surface. Georeferenced data are converted to the Global Coordinate System (GCS) to preserve the geometry of the source data. All SNE data added to this framework are available in a 3D interactive modeling application called the Geospatial Workstation (GW). The GW opens a direct user viewport on the IDB. In other words, the user will be able to view the SNE as it will appear to target applications. Both a 3D and 2D viewport are available on the Geospatial Workstation. These views can be slaved or used independently. The GW has proven to be a valuable tool in software debugging and system test. One of the main functions of the GW will be the control of data flow into the IDB, with processes and filters spawned from the GW GUI.

IDB will provide a convenient interface to a master library of geotypical models. The geometry, textures, and other attributes of models, as well model topology, will be modifiable from the GUI of the Geospatial Workstation.

3.5 Generalization, Enhancement and Value-Added Processing

The source data of interest to the RTV program will largely not have available color and texture data suitable for simulation visualization applications such as a Stealth. Part of the Transform Team effort will be directed at providing geospecific textures that can be applied to the terrain surface and feature coverages drawn from available source data. At the present time, attribution algorithms exist at two levels of complexity. At a coarse level, TIN polygons overlapping features such as soil defrags and areal hydrology are attributed based on the FACC values in these features. At a more complex level, the TIN is retriangulated, based on clipping against these same features before attribution. In addition to decoration of features with color and texture, algorithms will be added to IDB processing which make generalizations of features, using techniques such as thinning and geometric relaxation, which are necessary for scene complexity management in the target visualization application. Processing has also been added which makes features polymorphic in ways meaningful to a visualization application. For example, roads, are described locally in IDB as a sequence of vertices and edges,

with width attribution of the edges. For visualization, a more complex representation is needed, with a 3D polygonal representation of the roadway, complete with consistent texture coordinates. IDB maintains both local representations, as polymorphic views of a common base class instance. From a global point of view, road segments fit together to form a topological network. The Transform Team intends to include value-added processing for linear network defragmentation and reassembly, and encapsulate this processing in the IDB. Formally speaking, an IDB road network is an aggregation of road segment instances, with subclassing of the segments as required by the application semantics.

Previously, value-added processing such as the cut and fill of roadbeds on hillsides to preserve correctness of camber and grade, has been performed in a front-end environment such as a GIS. The Transform Team is analyzing the requirements for this kind of value-added processing in RTV, given the high level of resolution of available elevation data and the tactical level of detail of most of the feature data. It may be necessary to provide semi-automated cut and fill in areas of coarse resolution elevation data.

3.6 Common Data Path

Another fundamental architectural principle underlying the work of the RTV Transform Team is to maintain a common data path, from standard sources to applications, until the latest point possible. The IDB provides a common data path to the SAF CTDB and the VRSG MDB for all geometric data. This is a best-effort attempt to maintain correlation between these views of the SNE.

4 Details of Design

Some details of the data ingest process, and of the IDB class library itself, are spelled out in this section.

4.1 Data Flow

Figure 1 illustrates the data flow through from source data to SAF CTDB and VRSG MDB products. Input may either be a subset of the full extents of the end product, or a subset of the feature content of the desired output. The eventual inclusion of the SEDRIS Write API may be required for SNE interchange with C4ISR application environments.

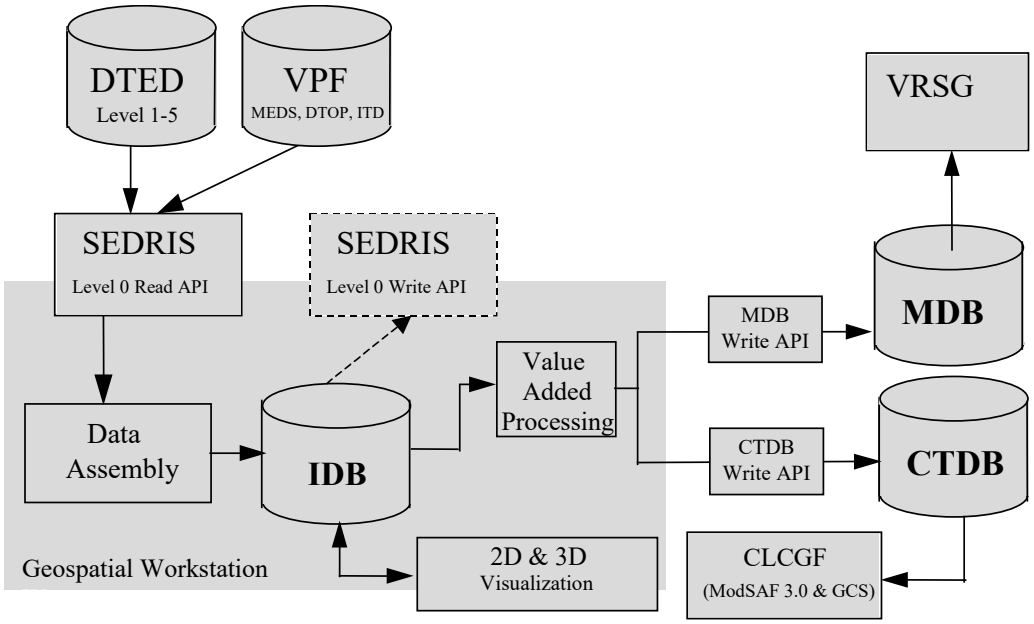

Figure 1

4.2 Software Layering

Figure 2 shows the layering of software services provided by the IDB class library. The Geospatial Workstation is an IDB application that uses portions of the IDB library as depicted in Figure 3 below.

4.3 Interim DataBase (IDB) format and class library

The Interim Database (IDB) is an intermediate terrain database format and associated class library which allows for persistent store, incremental compilation, and augmentation, either theme-based or geographic. IDB design is intended to facilitate the assembly of several different data sources for terrain and feature information. The format is application neutral, supporting both computer generated forces applications as well as visual system applications. Written in C++, the IDB class library adheres to C++ coding guidelines, consistent with industry standards, which have been developed over a number of SAIC software projects. Note that all of the class interfaces have been edited for this paper to concentrate on important structures and methods.

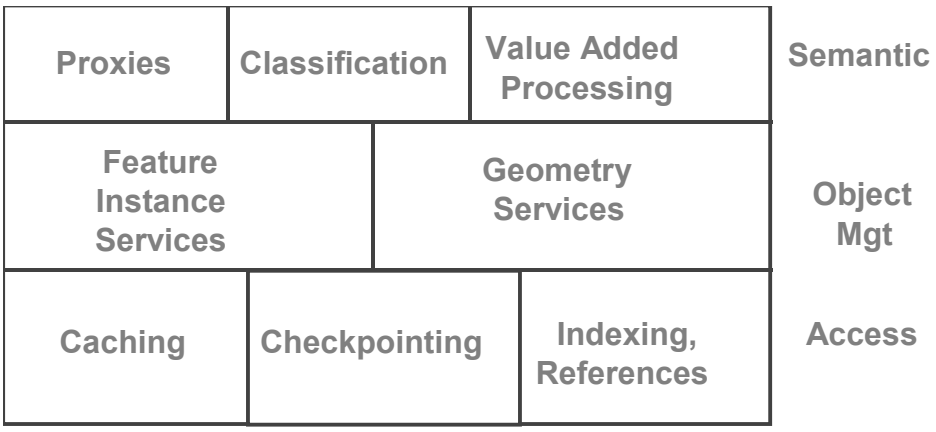

Figure 2

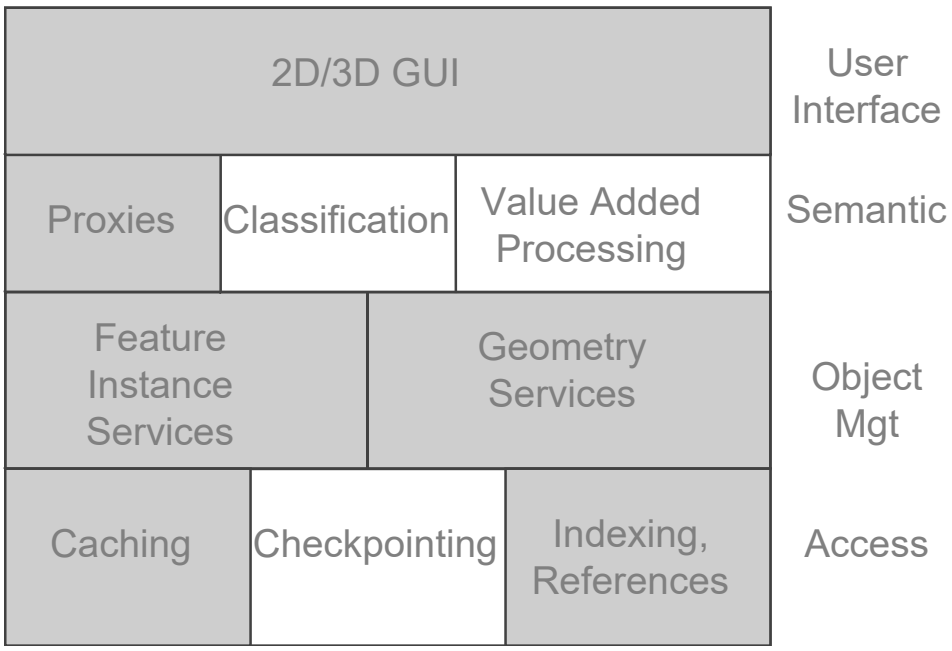

Figure 3

4.4 Metadata and Top Level Classes

An IDB is highly GCS aware and supports the creation of databases of arbitrary size and resolution. For background on GCS (the Global Coordinate System) see [1] and [2]. At the top level, the class `Idb` is an aggregation of metadata, cells, textures and models. For now, IDB metadata

describes the extents in geographic coordinates and lists the GCS cells tiling the database.

```
struct idbMetaData
{
    float64 swLat;
    float64 swLon;
    float64 neLat;
    float64 neLon;
    int32   numCells;
    int32  *cells;
};
```

The fundamental class holding the features and geometry and providing application methods for insertion, deletion and extraction of data is the `IdbCell`:

```
class IdbCell
{
  public:
    IdbCell( const char *path,
             int32 cell,
             float32 patchSize );
    IdbCell( const char *path,
             int32 cell );
    ~IdbCell();

    int32 insertData(
          IdbPrimitiveGeometry
          &theData);

    int32 insertData(
          IdbTerrainGeometry
          &theData);

    int32 insertData(
          idbTerrainPolygon
          &theData);

    int32 replaceData(
          IdbPrimitiveGeometry
          &oldData,
          IdbPrimitiveGeometry
          &newData);

    int32 extractData(
          float64 swX,
          float64 swY,
          float64 neX,
          float64 neY,
          Extraction Method meth,
          List
          <IdbPrimitiveGeometry>
          &data);

    int32 extractData(
          int32 numVerts,
          const float64
          bounds[][XY],
          ExtractionMethod method,
          List
          <IdbPrimitiveGeometry>
          &data);
```

The methods `insertData()`, `replaceData()` and `extractData()` at the `IdbCell` level are the primary interface for adding, replacing and extracting geometric data. Each cell has associated metadata:

```
struct idbCellMetaData
{
    int32   cellNumber;
    float32 patchSize;
    int32   patchesWide;
    int32   patchesHigh;
    int32   numPatches;
    float64 swLat;
    float64 swLon;
    float64 neLat;
    float64 neLon;
    ...
};
```

Associated with each cell are instances of subclasses of the `Accessor` class. These objects are responsible for managing the flow of data to and from persistent store. The possible subclasses are: `VertexAccessor`, `PolygonAccessor` and `SkinAccessor`. Each `IdbCell` is an aggregation of instances of the class `IdbPatch`.

4.5 Geometry Structures and Classes

This section lists some of the basic data types used by IDB. Geometric data are built from `Coord3D` structures

```
struct
    {
      float64 x;
      float64 y;
      float64 z;
      int32   cell;
    } gcs;
} Coord3D;
```

and are instances of subclasses of the class `idbPrimitiveGeometry`. This class has subclasses `idbLinearGeometry`, `idbSurfaceGeometry`, `idbVolumeGeometry` and

idbTerrainGeometry. There is also a geometric class idbPostedGeometry.

Persistent store of data is managed at the IdbPatch level. The most primitive persistent geometric structure is the vertexLocal. A vertexLocal specifies a point in a GCS cell's Cartesian coordinate system.

```
typedef struct
{
      uint16      patchNumber;
      unit16      x;
      uint16      y;
      float32     z;
} vertexLocal;
```

For image generation applications, texture coordinates and color information are also provided.

```
typedef struct
{
      vertexLocal vertex;
      // Actual spatial data
      float32           u;
      // Texture coordinate
      float32           v; ;
      // Texture coordinate
      uint8 char r, g, b, a;
      // Color
} vertexLocalVisual;
```

4.6 Feature Classes

Features are represented as instances of a particular feature class. All feature classes are derived from the abstract base class IdbFeature. This class contains a member variable of the type idbFeatureType which indicates which particular subclass a given instance belongs to. This base class stores a vertex pool through which geometry of the derived classes are stored.

```
typedef enum
{
    IDB_FEATURE_TYPE_LAID_LINEAR,
    IDB_FEATURE_TYPE_LINEAR,
    IDB_FEATURE_TYPE_CANOPY,
    IDB_FEATURE_TYPE_ABSTRACT,
      ...
} idbFeatureType;

class IdbFeature
{
public:
      IDBFeature();
      ~IDBFeature();
      ...
private:
      idbFeatureType
      featureType;
      idbLocalVertex
      *vertexPool;
      int         numVerts;
      ...
};
```

Linear features such as roads, rivers, and railroads, which form topological networks conformal to the terrain surface, are represented locally by the IdbLaidLinearFeature class, a subclass of IdbFeature. Coordinates of a linear feature are stored in GCS cell coordinates. For this reason, linear features may span terrain patches but not GCS cells. Linear features are stored as segments (two or more vertices) of like width and material type.

```
enum idbLaidLinearFeatureType
{
      IDB_LINEAR_FEATURE_ROAD,
      IDB_LINEAR_FEATURE_RAILROAD,
      IDB_LINEAR_FEATURE_RIVER,
      ...
};

class IdbLaidLinearFeature :
public IdbFeature
{
public:
      IdbLinearFeature();
      ~IdbLinearFeature();
      IdbLinearFeatureType
getLinearType();
      void setLinearType(
      IdbLinearFeatureType );
      double getWidth();
      void setWidth( double );
      short getMaterialType();
      void setMaterialType( short
);
      ...
private:
      ...
};
```

5 Implementation

The RTV Transform Team consists of staff of the Burlington office of SAIC's Technology Research Group and MetaVR of Brookline, MA.

Visualization is being provided by the MetaVR VRSG application with common software development efforts for data ingest and integration. The mission of the Transform Team is to develop more robust processes for production of SAF and visualization terrain databases, using only GOTS and open commercial software. Release 1.0 is scheduled for mid-July, An iterative spiral of design, build and test will follow, as resource usage and required enhancements become better understood. Due to the complex nature of the work, it is expected that some compromises will have to be made in early releases.

6 Acknowledgment

The work reported in this paper was performed under U.S. Government contract DACA76-93-D-0007. The authors would like to thank our sponsor, Chris Moscoso, and our technical adviser, Jeffrey Turner, both of the US Army Topographic Engineering Center, for their support and guidance during this program.

7 References

[1] Buettner, C. et al., "Global Coordinate System in the Improved Computer Generated Forces Terrain Database", Proceedings of the 6th Conference on Computer Generated Forces and Behavioral Representation, Orlando, Florida, July 1996.

[2] Evans, A. and Stanzione, T., "Coordinate Representations for CGF Systems", 13th Workshop on Standards for the Interoperability of Distributed Simulations, Orlando, Florida, September 1995.

Author Biographies

ALAN EVANS is the technical lead of the RTV Transform Team. He holds a Ph.D. in Mathematics from Michigan State University and an M.S. in Computer Science from New York University. Alan is the manager of the Burlington MA branch of SAIC.

HOWARD LU is a Senior Software Engineer at SAIC. Since joining in August 1995, Howard has been involved in the ICTDB project and the Synthetic Environment Data Representation Interchange Specification (SEDRIS) program. He graduated from the Massachusetts Institute of Technology with an M.S. in Computer Science in 1995.

TOD SHANNON is a consultant to SAIC. He has previously worked at Reality by Design, Loral Advanced Distributed Simulation and BBN Advanced Simulation Division. Tod has a B.S. in Computer Science from Carnegie Mellon University.

RICHARD RYBACKI is a Senior Scientist with MetaVR Inc. Rich previously worked at TASC, and is an Air Force veteran. He holds both a B.S. and an M.S. in Computer Science from the University of Texas at San Antonio.

VICTOR SKOWRONSKI is a senior member of the technical staff at TASC. Since joining TASC in June 1996, Victor has been involved in the ICTDB project. Prior to joining TASC, he did research in solid modeling at RPI. Victor has a Ph.D. in Computer Engineering from RPI, and an M.E. and B.E. in Electrical Engineering from Stevens. Institute of Technology. He is also a licensed Professional Engineer in New York and Massachusetts.