



# Modeling the Real World

Procedural model generation

for training and mission rehearsal

## Executive Summary

Warfighters operate in the real world.

It has become essential that simulation systems provide these warriors with geospecific environments for both training and mission rehearsal. With the growing availability of geospecific data, the opportunity to provide these environments has become feasible. Partial solutions are currently offered through manual and semi-automatic processes, but these solutions fail to model the more complex features found in the real world.

What is needed is fully automatic or at least mostly automatic construction of these geospecific environments in near real-time. A critical component of this process is procedural generation of models for geospecific features. Accurate models provide our warfighters with information they need to succeed and survive in executing critical missions.

## Introduction

Two seemingly opposing forces are affecting the simulation industry. On one hand, there is a strong desire to decrease the cost of building simulation databases. On the other, better realism and higher fidelity is becoming an important factor in training and mission rehearsal effectiveness. The desired geographic area covered by simulation databases is increasing, and at the same time the use of game engines and high performance image generators for urban and indoor ground and small arms training is dramatically increasing the density of content. For many years the only solution has been to either accept a lower level of realism and/or decreased content, or accept a high labor cost to manually model features and even entire areas using tools such as 3D Studio or Creator.

Procedural modeling software is increasingly used to build 3D models of terrain and features including buildings, transportation features, vegetation, etc. Many terrain generation software packages have some capability to model these features using GIS data formats. The result is quick and (mostly) automatic generation of terrain databases.

However, most of the sophisticated procedural modeling engines do not aim to precisely reproduce the real world in synthetic form. Instead they use statistical models of what a city may look like, and then create an environment that is feasible given the available input data. While these data sets may be immersive and useful in some training and simulation environments, they are not truly geospecific and are not appropriate for mission rehearsal training. They only allow for training of the most common environments. While most road intersections involve two roads at near right angles, some intersections involve multiple roads at sharp angles. When the scenarios leave out the uncommon features that occur in the real world, the result is missed training opportunities.

When these systems work, they produce models quickly at far less expense than manually generating models. Some of these software systems can produce stunning datasets automatically, but only if they omit certain features, or take liberties with the realism and geospecific nature of the features. For example, a bridge with an unusual shape may be modeled as two separate bridges. A ramp that merges with an overpass may result in an unrealistic or strange looking model. Additionally, while many procedural modeling software packages support building models from GIS data formats such as ESRI Shapefiles, for best results they often require manipulation of the geometry and/or attributes. The manipulation of GIS data to clean it up and make it appropriate for procedural model generation can add substantial cost.

**An effective procedural modeling solution must be capable of generating models using GIS data as available from sources such as the US Census Bureau or the National Geospatial-Intelligence Agency (NGA). It must be capable of generating a variety of features, including features that interact with each other. It must allow for a large degree of customization of the generated models to allow the user to generate truly geospecific models. *Without these capabilities, opportunities for cost and training effectiveness will be missed.***

## Challenges

### Multi-Level Features

A model generator capable of building a bridge can also generate an overpass by building a ramp on each side of the road to be passed over, and building a bridge between the two ramps. While this may not be exactly how the overpass looks in the real world, it is often sufficient. The problem occurs when other features need to interact with the overpass. For example a cloverleaf ramp may merge with the overpass above ground (i.e. part of the overpass that is not a built-up ramp). A simple bridge generator that doesn't account for other roads intersecting or merging results in a feature that is not useful for ground training. In many cities roads are built on top of each other, sometimes with traffic moving in different directions for each level, other times each level may be a different road completely. While an elevated section of a road may have some resemblance to a bridge, it will need entrance and exit ramps that a bridge usually will not have.

An additional problem with multi-level features is the lack of information on which feature is above or below another feature. Common GIS data formats provide road network data as two-dimensional features. A model generator needs to determine which feature to put on top and which to put on the bottom. Ideally the model generator will make decisions that are realistic, but for true geospecific models, some user input may be necessary.

### Intersections

A common approach to procedural model generation of transportation features (including roads, railways, tunnels, etc.) is to use a system of building blocks to handle intersections. A model for each intersection type is created manually, so a four lane road that intersects at a right angle with a two lane road would use a specific pre-built model. This becomes a problem for truly geospecific scenarios as there are seemingly limitless combinations of intersections possible. Some intersections are not symmetrical, and some intersections are not right angles. Some intersections have more than one left or right turn lane, and some have none. Often the GIS data does not have enough information to automatically generate models with all of these characteristics precisely as they exist in the real world. However, a truly geospecific model generator should be capable of generating the intersections properly if the user provides the correct data. Further, the

### Intersection Examples



Figure 1 - Anomalies when roads change



Figure 2 - Merged roads without clipping



Figure 3 - Merged roads with clipping

cost of building all of the possible intersection models will discourage a variety of procedural models from being created.

Another common approach is to generate models one feature at a time, assuming each model does not interact or affect any other models. This is the simplest solution, and one that works for many bridges and overpasses. The problem with this approach is apparent when a ramp must merge with an elevated road such as an overpass. Intersections of simple roads will result in a weaving pattern, where some roads appear on top of the other at the intersection areas. If there are sidewalks or guard rails, the sidewalks and guard rails will cross the path of features it intersects with.

Transportation features such as roads, bridges, and tunnels often have a varying number of lanes. Most model generators do not have the ability to smoothly transition from two to four lanes without anomalies. In the real world, not all roads or lanes are precisely the same width, and the process of merging two lanes can take place over different distances. Model generators that do not have the ability to gradually change the width for roads will have anomalies like those illustrated in Figure 1.

### Workflow/Tool Specific Solutions

Most model generation capabilities are closely tied to a tool that does more than just build models. Terrain generation tools often include model generation capabilities. This is understandable and in many cases much simpler to implement because the models generally have to conform to the terrain. The downside is the user is not able to select the best of breed capabilities when choosing terrain and procedural model generation tools. Likewise, a tool that generates good procedural models may not be appropriate for a user if the terrain generation doesn't meet their requirements.

### Trafficable Roads and Dynamic Environments

A procedural model generator must do more than create textured geometry if the environment is intended to support constructive training systems. Model generators for these systems must include trafficability attributes, and have to generate features that will not 'kill' computer generated forces (CGF) due to violating a geometry constraint (e.g. a road with a short segment that exceeds a maximum turn angle might cause a CGF vehicle to turn over.)

## Complex Features

Complex features are features that intersect or otherwise interact in complicated ways, as well as features with abnormal geometry. Examples include:

- Multi-level bridges and overpasses
- Merging and splitting roads, bridges, and overpasses
- Multi-road (more than two) intersections
- Unusual angles in intersections
- Banked turns



Dynamic environments may require interactive features such as operable traffic lights. Because the traffic light configuration will depend on the road network it is controlling (e.g. the number of lanes, the direction of traffic, etc.), procedural model generators must offer support for dynamic environments.

## Approach

### Procedural Model Generator Requirements

In order to meet the requirements of modern simulation and mission rehearsal systems, and to avoid the problems identified previously in this paper, an effective procedural model generator must meet the requirements listed below. Clearly there are additional requirements for a model generator, but this list is highlighting additional requirements for generating complex features. This list is important because while some model generators meet some of these requirements, these requirements are generally missing from most procedural model generation systems.

#### Requirements for procedural model generation:

- Support user generated model templates
- Understand the topology of the GIS data used to create the models
- Determine the vertical layers using topology
- Manipulate and/or transform standard GIS data to its needs
- Use topology when building the models (to flatten intersections, and remove conflicting sidewalks, guard rails, etc.)
- Properly attribute surfaces and models to support constructive simulation and dynamic environments
- Use civil engineering standards to build realistic models

#### *Support user generated model templates*

In the real world, there is a seemingly limitless variety of architectural styles and construction materials used to build features. Many procedural model generators use hard coded designs for models. This means that a programmer is required to make any changes to the look or design of the model. Allowing the user to create their own model templates using industry standard modeling tools allows for limitless variety of models to be generated. This same capability allows a program to build models with the precise levels of detail they require. However, model templates must go beyond customizing a few parameters and textures. Allowing the user to generate model templates that define all aspects of the model generation is required to truly meet real-world requirements for geospecific models.

#### **Understand the topology of the GIS data used to create the models**

In order to make multiple models properly conform to each other, the model generator needs to understand the relationship between the features that are used to create the models. For example, a four lane road that splits into two separate two lane roads must properly connect without anomalies. GIS data typically would represent these as three different features with a common point where the split happens. A model generator that does not understand the topology will result in a feature that looks like the screenshot in Figure 2. Another example is a ramp that merges with a major highway. The

ramp must not change the grade of the highway while merging, or in any way interfere with the highway. In order to build the models so the ramp conforms to the highway, the model generator must know and understand the relationship.

### *Determine the vertical layers using topology*

When presented with GIS data that represents a complex interchange such as that shown in Figure 5, the model generator must make decisions on which roads overpass other roads, which intersect, etc. A road may overpass at different levels at different locations. The topological data derived by the model generator (and optionally enhanced by a human operator) can be used to determine the most realistic design for a complex interchange. This process must be efficient however, because even an interchange such as the one shown in Figure 5 can result in many billions of possible configurations. The model generator must choose a realistic configuration, and it must do it efficiently.

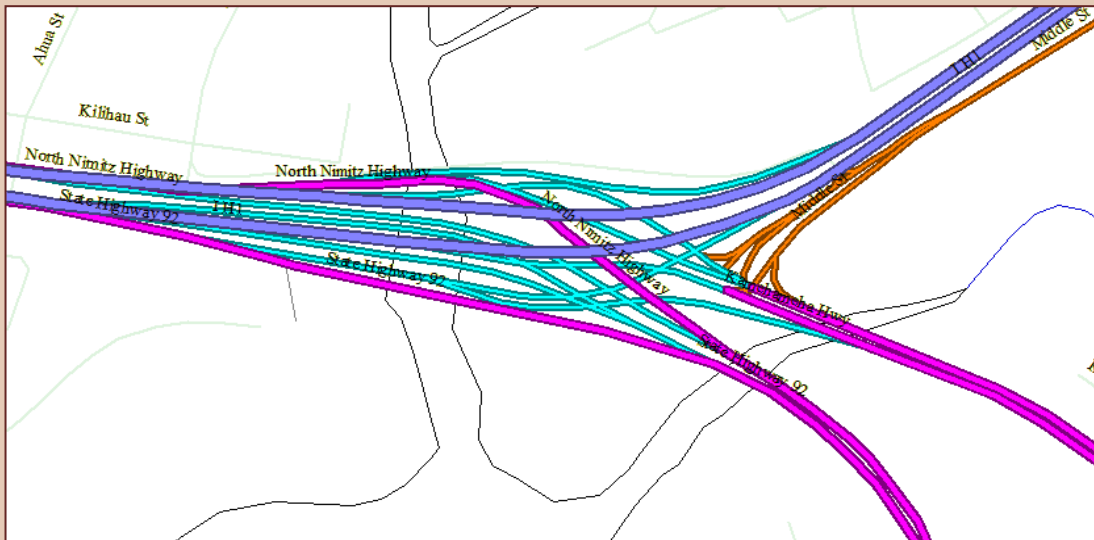


Figure 4 - Two-Dimensional GIS Data

### *Manipulate and/or transform standard GIS data to its needs*

To make procedural model generation on a large scale cost-effective, the GIS data must need little if any cleanup. A model generator that can process pre-existing data and manipulate it automatically will allow a great deal of content to be created automatically in a cost-effective manner.

### *Use topology when building the models*

When two roads, each with outside guard rails merge, the guard rails must end before the trafficable surfaces join. A similar situation occurs with sidewalks. Tunnels that intersect or merge must have the walls removed to allow traffic for the adjacent feature to pass. In order to support these types of features, the model generators need to understand the topology of not just the features, but also the models. A model generator cannot just place geometry without understanding what the geometry is (e.g. a sidewalk), and how it intersects with other geometries (e.g. a road).

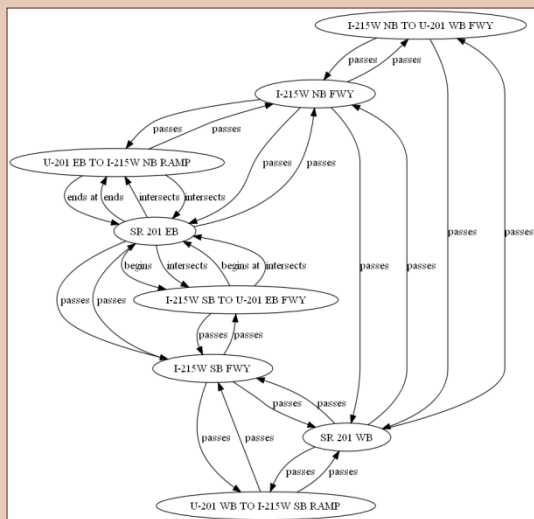
### *Constructive/Dynamic Environment Attribution*

Combined arms training requirements, and combined virtual/constructive simulation environments introduce additional parameters and model requirements. Besides visible light textures and geometries, models need material identifiers and textures for sensors. Additional footprint or trafficability data may be needed for SAF systems. In order to build models that support these environments, a flexible design for attribution and metadata must be considered.

### *Use civil engineering standards*

When building roads, ramps, bridges, overpasses, tunnels, etc., a model generator must take into account civil engineering standards. How sharp a turn can be for a given road is determined by the type of road, the speed limit, and other factors. Maximum grades, banked turns, safety barriers and drainage gradients are examples of parameters that must be considered in order to build detailed and realistic features. Civil engineering standards vary regionally, so a model generator cannot rely on hard-coded estimates of civil engineering standards if truly geospecific models are required.

## Feature Topology



Generating a model for a single feature without consideration for other features is easy. For real-world complex features, however, it is critical to consider the impact of other features on the feature to be modeled. A feature may overpass, underpass, or intersect with other features. A feature intersection may include crossing or merging. Multiple features may intersect at a single point.

These variations can be effectively represented in a mathematical graph. This image is a visual representation of a graph for a small feature set in Utah with a possible set of relationships.

## SPAWN Procedural Model Generation Toolkit

Cognitics is currently developing a model generator that has been designed to meet the requirements and solve the problems identified in this paper. Our solution, called SPAWN, and was designed from the outset to automatically produce high quality models of complex features.

SPAWN uses the concept of a Model Kit to represent all the information needed to procedurally create models. Model Kits at a minimum include geometry and texture information to describe the models. Material information is used to create models that work in a variety of visual and sensor environments. In addition they include a variety of parameters the model generator uses to build models of real-world features. The parameters allow the model generation to go far beyond extruding a shape into a 3D model. SPAWN uses this information to build features that vary in width and height (including merging and splitting lanes), and to place features such as lamp posts, traffic signs, etc. Each model kit includes a set of attachment points that are used to make sure features properly connect to one and other.

SPAWN includes a set pre-defined model kits to allow out of the box generation of a wide variety of transportation models. In addition, SPAWN provides a set of user friendly tools that allow model kits to be imported from industry standard modeling tools and configured for procedural model generation.

Model kits can be configured to represent several different levels of detail (LODs) of the same model, so the model generator can build a set of features that correlate, but have different LODs.

The SPAWN design includes a database of regionally specific civil engineering standards. When building models, SPAWN will automatically retrieve the regionally specific rules for the model being generated based on the geospecific location of the feature.

Cognitics has developed an implementation of the Open Geospatial Consortium (OGC) Simple Feature Access (SFA) specification. Our implementation supports a wide variety of GIS data formats. We have built on the SFA to create a set of feature processors that allow automatic attribute translation and geometric operations. Our library of feature processors can detect and in many cases automatically correct features so manipulation or cleanup of vector data is minimized. We have created bindings to the Lua scripting language, thus creating a simple to use and flexible pipeline. This pipeline allows users to specify the steps needed to process data and customize it to any dataset.

Commonly available GIS data does not include explicit topology information. Whether two crossing line features create an intersection or an overpass is often unclear. SPAWN includes a topology analyzer that examines clues in the geometry and the attribution and deduces the topology using a set of rules and probabilities. The topology is then saved to an intermediate file that can be edited by an end user to ensure the models generated from this topology is correct.

When two or more features cross but do not intersect, each feature must be at its own level. Many real world highway interchanges have features that cross at 4-6 different layers above ground. SPAWN uses an innovative system that assigns a cost to different road configurations, and then uses genetic

algorithm to determine which configuration is best suited to the real world. This system prevents roads from going over and under other features unnecessarily, and minimizes the use of raised sections of roads. The result is automatic configuration of highly complex features calculated in seconds.

In order to support a wide variety of terrain and model generation workflows, SPAWN is provided as a set of tools and software libraries that can be integrated as plug-in modules to virtually any software environment. Cognitics provides professional services to customize and integrate SPAWN to meet specific program needs.

## Specifications

Programming Language: C++, Lua

Platforms: Windows/Linux 32 and 64 bit.

- Input formats:
  - Models / Vector data
    - Shapefile
    - ESRI Geodatabase
    - GML
- Output formats:
  - OpenFlight
  - Correlated GIS data
    - Shapefile
    - ESRI Geodatabase
    - GML
  - Custom format support  
(developed to your requirements)



**For more information, please contact Cognitics, Inc:**

**Cognitics, Inc.**  
**6200 N Meeker Place**  
**Suite 220**  
**Boise ID 83713**

**Kevin Bentley**  
**kbentley@cognitics.net**  
**(208) 904-3780 (voice)**  
**(866) 922-2037 (fax)**