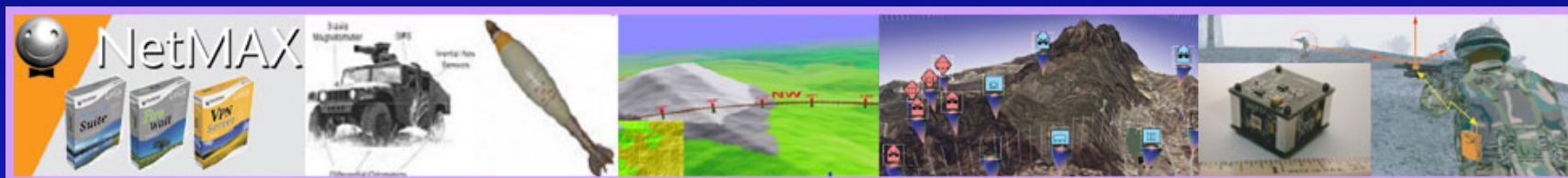


A String-Defined Symbol Construction Library

Douglas Haanpaa

Lead Programmer – Networked Simulation



Cybernet Systems Corporation

727 Airport Blvd

Ann Arbor, Michigan 48108

www.cybernet.com

(734) 668-2567

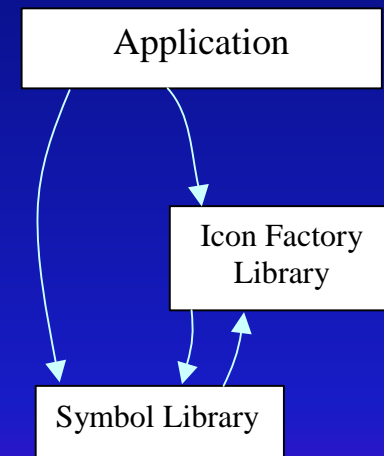
Introduction/Requirements

- ❖ **Tactical Terrain Viewer** – Cybernet’s 3D Mapping Software with force overlay needed a method for indicating the position of entities on the terrain. The force-overlay display needed symbology. Specific requirements were that it:
 - Support **MIL-STD-2525B** as it is the prevailing standard in the DoD and was therefore selected as the target symbol set.
 - Support the construction of alternate symbol sets
 - Support the creation of new symbol sets through human-editable configuration files.
 - Facilitate the incorporation of text and images.
 - Allow symbol components to be overlaid with transparent parts.
 - ❖ Library tentatively named the **Tactical View Symbol Library** (TVSL)
-
-

Architecture

The TVSL system is comprised of two interoperating components:

- The *Symbol Library*. The *Symbol Library* is responsible for the parsing of the input definition files that specify a particular symbol language.
- The *Icon Factory Library*. The *Icon Factory Library* is responsible for taking information produced by the Symbol Library and constructing an icon that can be used in the application.



Application Programmer Interface (API)

The symbology library API allows the application to create an icon as follows:

1. Application invokes the Symbol Library to create a symbol object by providing an input string

```
m_symbol = new CSymbol2525B("abcdefghijkl", 7, *symbology);
```

2. The application invokes the Icon Factory Library by passing it the symbol created above, whereupon the Icon Factory constructs an icon and passes it back to the application.

```
m_icon = ScenarioIconFactory.CreateIcon(m_symbol);
```

3. The application can then access the icon to move it around on the screen and change its dynamic values.

```
m_icon->SetPosition(100.0, 100.0);
```

Configuration Files

The symbology library is data driven by three files that specify a particular symbol language:

- **Field.ini** – defines the format/componentization of the input string
- **Primitives.ini** – defines the icon components (images, text, and their positions)
- **Specs.ini** – defines the logical link between the components of the input string (defined in Fields.ini) and the primitives that compose the symbol (defined in Primitives.ini)

Fields

The input string is simply a set of characters of arbitrary length. The **fields.ini** file breaks this string up into fields:

Name	Length	Description
Ae	6	"symbol ID - function"
Af	2	"symbol ID - modifier (AC, R)"
Ag	2	"symbol ID - country"
Ah	1	"symbol ID - order"
C	9	"quantity"

Primitives

The primitives in **primitives.ini** allow the developer to add images and text:

Frame – a texture (Bitmap or Targa file) that is added to the symbol at a certain position and with a specified scale factor.

```
Frame CircleBlueFill = "Circle Friend.tga" + (0.0,0.05) + (1.1,1.0)
```

Modifier and **Icon** – These two types are identical to Frame. They were created in order to conceptually separate the primary icon image from frame and decoration images.

Dtext – a text entity that is added to the symbol at a specified position and scale. This text is dynamic in that the application can change its value.

Stext – a text entity that is added to the symbol at a specified position and scale. The format of a SText line is identical to the format for DText. However, unlike DText, this text is static.

Specs

The `specs.ini` file defines the rules that invoke the primitives based on the fields in the input language. :

```
Ae = --cdef , Ah = k: CircleBlueFill
```

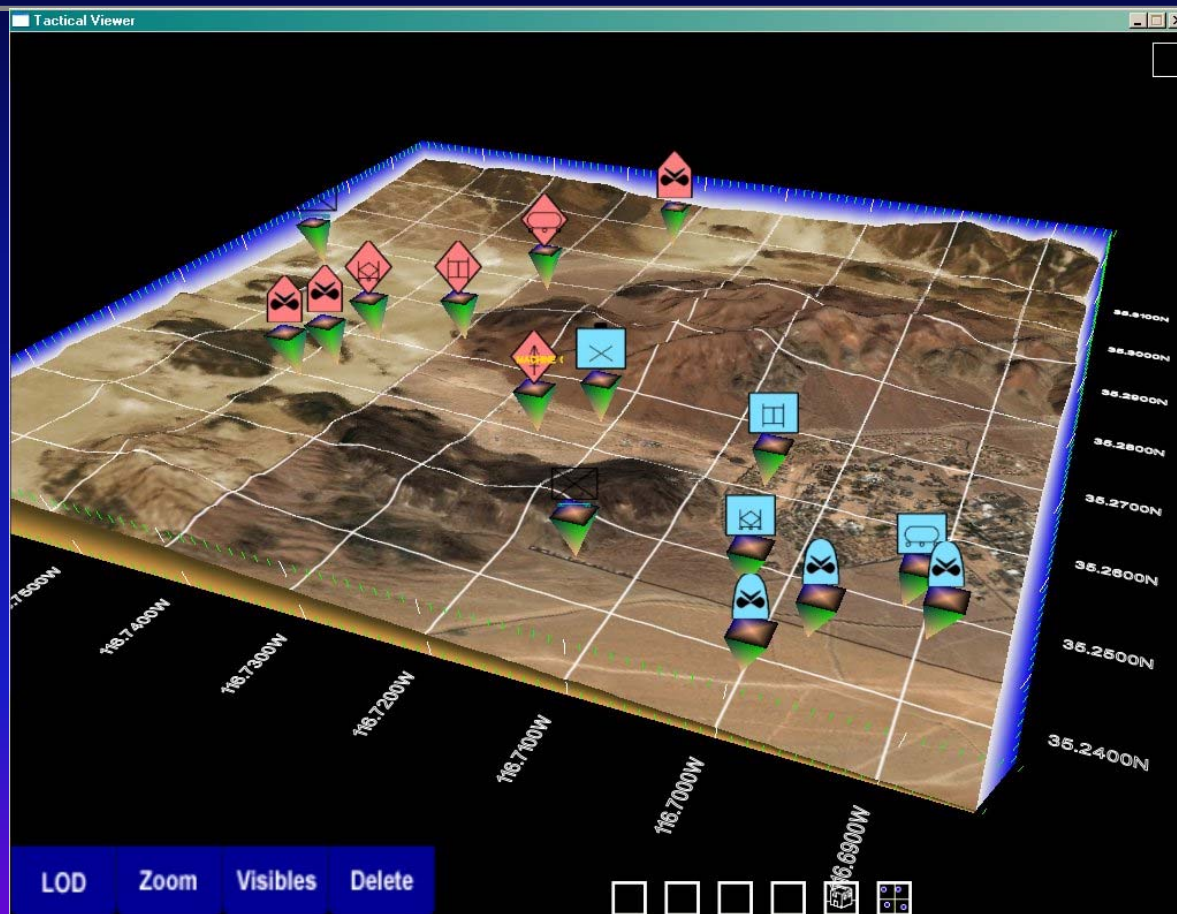
If the symbol substring described by Ae ends in ‘cdef’ AND the symbol substring described by Ah is the letter ‘k’, then invoke the primitive called “CircleBlueFill”. (Note: the dash “-” is a one-character wildcard)

Example of Use: Overseer



Amplifying Human Performance Through Advanced Technology

Examples of Use: Tactical Terrain Viewer



Amplifying Human Performance Through Advanced Technology

Network Transport and Misc.

- Because the representation is string-based, network transportation of the symbols is easy and cross platform (i.e. no endian issues)
 - We have demonstrated a system that transports these symbols using **HLA**.
- System currently supports dynamic updating of text primitives from the application.
- Icon construction/rendering uses Cybernet's OpenGL-based **GOOEY** library.

Future Direction

- Graphical Interface to create the Ini files.
 - Import, position and scale images.
 - GUI for rule (spec.ini) construction.
- Dynamic image manipulation by application.
- More content



Questions? & Contact Info

Douglas Haanpaa

Lead Programmer

Networking and Simulation

dhaanpaa@cybernet.com