Configuration

Graphics group
useSoftwareRenderer – enables the DirectX warp software renderer for use in VMs, with old GPUs, etc

SWG group
repositoryScript – name of the lua script which defines the files to look for for inclusion in your repository
baseDirectory – directory containing your trees/tocs
useFileOverrides - enables/disables a scan of your SWG directory for override files
additionalOverrideDirectories – specify additional override directories to be used in the repository, say if you want to work on a project in a separate folder (SWB will auto-apply such directories to your client for testing but this isn't available in SIE).
autoTemplateNamingOn - automatically generate and look for template file names based on the current chunk
autoTemplateApply - automatically apply templates on chunk selection. You might want to disable this if it is too slow
enableRenderer – allows you do disable the renderer entirely, might be useful if you don't want the heavy CPU usage that comes from using the software renderer on slow CPUs/VMs.

Templates

The template system supports the following primitive types:

- string, byte, bool, bool2 (2-byte wide bool), bool4 (4-byte wide bool), ushort, short, uint, be_uint (big endian), int, float, double, tag (char[4]), rtag(char[4] reversed)
- 8-bit per channel colours: rgb, bgr, rgba, argb
- 32-bit per channel colours: rgbf, rgbaf, argbf
- vec2, vec3, vec4, quat, mat3x3, mat4x3, mat4x4

The primitive types can be used to create structures and arrays.

- Arrays are defined like this Type[Name, ArraySize], where ArraySize is either a constant or a previously read scalar variable.
- Arrays can be scaled (for example if you need to read a number of floats but the chunk gives the size in bytes) by Type[Name, ArraySize, ScaleFactor]
- Arrays can read until the end of the chunk by Type[Name, inf]or Type[Name, -1]
- Structs are defined like this struct[StructName]{Variable1,Variabl1}
- Structs can be nested arrays can not

As an example of structs, vector types could have been implemented like so:

- struct[rgb]{byte[r],byte[g],byte[b]}
- struct[vec3]{float[x],float[y],float[z]}
- struct[quat]{float[w],float[x],float[y],float[z]}
- struct[mat3x3]{float[m00],float[m10],float[m20],float[m01],float[m11],float[m21],float[m02],float[m12],float[m22]}

As of version 3.2, you can define and use enumerations:

- enum[theAnimals]{CAT=1,DOG=2,FISH=4}
- int[animal](default=1,enum=theAnimals)

MISCELLANEOUS

As of version 3.0, you can create new forms from a template using the "Add Chunk from Template" button.

You can set the chunk type for the form to be added by adding a meta comment:

Code:

```
// META(CHUNKID:DATA)
```

You can append default values to POD types with the following syntax:

- float[x](0.43), string[y](default="hello_swg").

Default values for vector types are not supported (I can add them if there's any demand for it).

WARNING

The template system isn't perfect or foolproof - if you try to break it you will likely succeed!

REQUIREMENTS

64-bit operating system
DirectX 11 compatible GPU with Shader Model 4.1 or greater

http://www.microsoft.com/en-gb/download/details.aspx?id=40779
Microsoft .NET Framework 4.5

http://www.microsoft.com/en-gb/download/details.aspx?id=40784
Visual C++ Redistributable Packages for Visual Studio 2013 x64

https://www.microsoft.com/en-gb/download/details.aspx?id=14632
Visual C++ Redistributable Packages for Visual Studio 2010 x64

https://www.microsoft.com/en-gb/download/details.aspx?id=8109
DirectX End-User Runtimes (June 2010)