SENTINEL

MANUAL

Table of Contents

[Naming Conventions 3](#_Toc367279126)

[Wrappers 3](#_Toc367279127)

[References 3](#_Toc367279128)

[Inspectors 3](#_Toc367279129)

[Assets 4](#_Toc367279130)

[Model Exporter 5](#_Toc367279131)

[Level Editor Controls 6](#_Toc367279132)

[Sentinel Test Program Controls 7](#_Toc367279133)

[Create Custom Program 8](#_Toc367279134)

# Naming Conventions

All member class variables start with m, e.g. ColorRGBA\* mRef;

All properties start with capital letters, e.g. Inspector.TreeStyle;

All defines, macros, and static variables are in all capitals, e.g. double DESIRED\_FRAME\_RATE;

All member functions start with capital letters, e.g. GameWorld::Inst()->Startup();

All local variables start with lowercase letters, e.g. MeshBuilder meshBuilder;

## Wrappers

Classes named with W represent managed Wrapper classes created within the CLR. They each possess an instantiation of a variable, and free automatically through the Finalizer. Call Dispose to remove the internal variable from memory before C# does its garbage collection.

Components are the exception to the Wrapper classes in that they create a new GameComponent, but they do not free the memory automatically, except through their associated GameObject during Shutdown. Call Dispose to free the memory of a GameComponent if not attached to a GameObject. This behavior also occurs with GameWorld and their GameObject(s).

Classes that possess m\_shared\_ptr (e.g. WMesh, WModel) should call Dispose to ensure predictable counter decrement. Certain classes (e.g. WMeshComponent) have properties that get a new m\_shared\_ptr possessing class (e.g. WMesh). Doing so creates a new variable that increments the counter. Take note that assigning a WMesh to another WMesh does not affect the counter in any way.

## References

Classes named with R represent Reference classes that reference a variable. The Reference class inherits from the base Wrapper class. Calling Dispose has no effect on the variable as it is located elsewhere in memory. This creates a safety mechanism that prevents variables from unintentional deletion. Many functions return a Wrapper class, but they may actually be a Reference class instead. This allows the variable to become a function parameter without worry of casts, and without worry of deletion from Dispose.

In addition, if a class starts with RP, it references a pointer. Assigning to this variable causes the internal variable to point to the newly assigned one.

## Inspectors

Classes named with I represent Inspector classes for the Editor. Each Inspector class is a TreeViewItem designed for placement within the Inspector interface.

## Assets

Classes named with A represent Asset classes. They are broken down into groups called Texture, Shader, Mesh, and Model.

# Model Exporter

The custom 3ds Max exporter resides within the Sentinel\_Exporter folder. The resulting Sentinel\_Exporter.dle outputs to the local 3ds Max 2012 folder, e.g. C:\Program Files\Autodesk\3ds Max 2012\plugins, through the “ADSK\_3DSMAX\_x64\_2012” environment variable. An additional environment variable, “MAX2012SDK”, references the SDK. Generally, the file folder for 3ds Max resides within a protected folder, therefore, to compile the program, Visual Studio should be opened in Administrator mode.

The exporter only exports the model within the scene, i.e. no camera, lights, etc. Copy the textures into the same folder as the model. Save the file as a native file format to 3ds Max 2012 in order to import the model.

# Level Editor Controls

The scene viewing area can rotate its view by holding the middle mouse button / wheel.

Move forward and backward by scrolling the middle mouse wheel.

To create a hierarchy with the Objects, drag and drop them onto each other.

All values within the Inspector modify the objects immediately, i.e. no need to reload the scene or object to see the changes.

# Sentinel Test Program Controls

WASD – Move

Space / C – Up / Down

ESC – Exit

# Create Custom Program

Open “Sentinel\_Test.sln”

Use “Sentinel\_Test.cpp” as a reference.

Each header file contains instructions on its usage.