SENTINEL

MANUAL

Table of Contents

[Naming Conventions 3](#_Toc369695719)

[General Object Descriptions 3](#_Toc369695720)

[Wrappers 3](#_Toc369695721)

[References 3](#_Toc369695722)

[Inspectors 4](#_Toc369695723)

[Assets 4](#_Toc369695724)

[Model Exporter 5](#_Toc369695725)

[Level Editor Controls 6](#_Toc369695726)

[Sentinel Test Program Controls 7](#_Toc369695727)

[Create Custom Program 8](#_Toc369695728)

# Conventions

## Names

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. mGameWorld->Startup();

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

# General Object Descriptions

## 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** or **Release** to remove the internal variable from memory before C# does its garbage collection. All classes under ***Components*** or ***Systems*** require **Release** to free their memory.

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 deletion. 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** (i.e. ***Assets***) should call **Dispose** to ensure predictable counter decrement. Special 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. A ***get*** property operation always creates a new **m\_shared\_ptr** and must have **Dispose** called.

## 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. The textures automatically copy into the same folder as the exported model. Save the file as a native file format to *3ds Max 2012* in order to import the model.

# Level Editor Controls

The world 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

|  |  |
| --- | --- |
| **Keys** | **Function** |
| WASD  Space / C  ESC | Move  Up / Down  Exit |

# Create Custom Program

Open “**Sentinel\_Test.sln**”

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

Each header file contains instructions on its usage.

**Sentinel\_Test** tests components and the systems to ensure reliability and proper functionality.

The program is set up to load the **Default.MAP** file by default.