What is MEC Threaded?

MEC Threaded is a small and light add-on module for More Effective Coroutines. It allows you to switch to an external thread using a simple yield return command. MEC Threaded uses thread pools to execute your coroutines as efficiently as possible, and takes care of the finicky work of sending your data to external threads in a thread safe manner.

The Limitations of Threading

This package is not for everyone. Threads are very difficult to use in Unity since **you cannot directly access any of the objects in your scene from any external thread**. Think about that for a moment, if you switch to another thread then the only things you can easily do on another thread are file reads/writes, data processing, and any networking that doesn't use the www object. That's not a very long list. If you're just looking to move an object across the screen **do not use threads to do that!** Unless your object is moving according to some *very* complicated logic threads will only slow things down and introduce potential errors.

In order to get around the limitation where you can't access objects in your scene from an external thread there are two main structures that you can implement. Saving the current state of the game to disk in the background is an excellent application for MEC Threaded. You can also use MEC Threaded for swarming.

In order to save your game in the background using this plugin you would

1. Start a MEC coroutine
2. Serialize the data that you need to save into a local variable
3. Switch to an external thread
4. Perform the file IO

In order to implement swarming you would

1. Run a MEC coroutine in a while loop that executes the following steps every frame
2. Read the current position of the object(s) that you are operating on into a local variable
3. Switch to an external thread
4. Perform calculations on the data that you read and save the results to a local variable
5. Switch back to the main thread
6. Update the position of the object(s) based on the value stored

An example implementation of both of these structures are included in the project.

Threading Syntax

MEC Threaded uses MEC. MEC Free is included in the project, but if you have MEC Pro or any other plugin from Trinary Software just import them over the top. It may also be possible that there is a more current version of MEC Free in the asset store, which is also ok to import over the top of the one included here.

The interface for MEC Threaded is very simple: You switch to another thread using a yield return statement within a coroutine. You can then end the coroutine or switch back to the main thread using another yield return statement.

```
yield return Threading.SwitchToExternalThread();
// Do stuff and then optionally use the following line to switch back
yield return Threading.SwitchBackToGUIThread;
```

Most of the time it's best to just run your threads at the default (normal) priority. Thread priority makes the thread take up more or less of the "attention" of the CPU. If you want to run at a different priority you can pass it into the SwitchTo function.

```
yield return Threading.SwitchToExternalThread(ThreadPriority.Low);
```

Normally you want to use the thread pool because it's more efficient, however if you expect your thread to take longer than one frame to execute then it's better to use a dedicated external thread. Also if you want to use Threading.Sleep to sleep for a long time inside your thread it's best to use a dedicated thread for that.

```
yield return Threading.SwitchToDedicatedExternalThread();
Threading.Sleep(number);
```

Threading.Sleep takes a "number of seconds" float. It's ok to use a pooled thread to sleep for short periods of time (if your thread is watching a network stream for example), just be sure to yield to other tasks between sleeps in case there's another thread waiting:

```
Threading.Sleep(0.016f);
yield return Threading.YieldToOtherTasksOnThisThread;
```