Asynchronous I/O

*This sample is compatible with the November 2015 Xbox One XDK or later*

# Description

This sample demonstrates the three major patterns for performing asynchronous file operations using overlapped I/O. The three major patterns are

1. Waiting on an event associated with the operation
2. Periodically querying the status of the operation
3. Having a callback function used when the operation completes

# Using the sample

Run the sample and it will iterate through each pattern. You can step through the code in the debugger to follow exactly what is happening. The code is heavily documented for what is happening and why.

# Implementation notes

All of the code for the demonstration is contained in the OverlappedSample class, the definition is in OverlappedSample.h. The implementation of the class is broken into four source files.

1. OverlappedSample.cpp
   1. Shared management of internal data structures used for each pattern. Some examples are the open file handles, thread management, and cache management for pending overlap structures.
2. WaitOverlapped.cpp – Contains two major functions
   1. EventTypeThreadProc – Thread procedure that creates the asynchronous request based on the wait pattern. It keeps up to c\_maxRequestsInFlight pending requests at a time. When it hits this number, it will stop creating new requests until some pending requests have completed.
   2. WaitForEventOverlappedToFinish – Procedure that performs the actual wait for a request to finish. It will wait simultaneously for at least one of the pending requests to finish and then cleanup that request.
      1. This function could be called from any other location in your code and any other thread. It is only called from EventTypeThreadProc for ease of implementation in the sample.
3. QueryOverlapped.cpp – Contains two major functions
   1. QueryTypeThreadProc – Thread procedure that creates the asynchronous request based on the query pattern. It keeps up to c\_maxRequestsInFlight pending requests at a time, it will stop creating new requests until some pending have completed.
   2. QueryForOverlappedFinish – Scans through all pending requests to see which ones are complete and processes them if they are complete.
      1. This function could be called from any other location in your code, any other thread, and at any frequency. It is only called from QueryTypeThreadProc with a rapid frequency for ease of implementation in the sample.
4. AlertableOverlapped.cpp – Contains three major functions
   1. AlertableTypeThreadProc – Thread procedure that creates the asynchronous request based on the alertable pattern. It keeps up to c\_maxRequestsInFlight pending requests at a time, it will stop creating new requests and sleep in an alertable state until some pending requests have completed.
   2. FileIOCompletionRoutine – Serves as a hook between the OS and the OverlappedSample class instance. This is the function passed to the OS as the callback for when the request has finished.
      1. This function can be called at any point in time and on any thread that has entered an alertable state.
   3. AlertableCompletionRoutine – Called by FileIOCompletionRoutine when a request has completed. Will go through the pending list and match up the completed request with a pending one.
      1. No thread safety has been added to this function because only the AlertableTypeThreadProc thread enters an alertable state. This means only that thread will receive the callback.
      2. Depending on your implementation, thread safety may need to be added to the callback routine.

The Wait and Query patterns are very similar. They both use an internal event to signal when the request has finished. They can also both wait for a configurable amount of time. The main difference is that the wait pattern can use WaitForMultipleObjects which means it can wait on all the pending requests at the same time. The query pattern with GetOverlappedResult can only check a single request at a time.

Since they both use a manual reset event for signaling it is possible to mix the patterns. For example, the main file loading system could use the wait pattern on all pending requests while another thread could use the query pattern for a critical request. If the patterns are mixed some form of thread safety should be used due to the manual reset event, they could both return complete at the same time.

It is strongly discouraged on using an auto reset event. It is very easy to get into an issue where the WaitFor###Object returns complete. However, the event then switches to not signaled so the next call to GetOverlappedResult fails because the event is no longer signaled.

# Update history

Initial release October 2016