SlimMath

# Goal

The goal of the SlimMath project is to produce a managed library which allows for the batch processing of mathematical operations on Vectors, Matrixes, Quaternions, Planes, and other 3D primitives using hardware accelerated functionality such as SSE2 through the use of unmanaged libraries such as XNA Math.

By providing a suitably abstracted managed library SlimMath will allow a user to construct batches of operations to be performed on provided 3D primitives which can then be fed to an underlying unmanaged layer that will perform all of the translation and operations before returning back to the client. Thus minimizing the impact of managed 🡪 unmanaged and unmanaged 🡪 managed marshalling, along with using advanced CPU instruction sets, such as SSE2.

# Architecture

The core component of the SlimMath library will be the SlimMath.Batch object, which will provide methods for constructing and queuing up operations to be batch processed.

**Batch** sampleBatch = new **Batch**();  
**Handle** m = sampleBatch.Add(**MatrixOp**.Mul, rotMatrix, tranMatrix, **ResultType**.Ignore);  
**Handle** result = sampleBatch.Add(**VectorOp**.Mul, lookVector, m, **ResultType**.Output);  
sampleBatch.Process();  
**Vector** resultingLook = result.ToVector();

Each batch is separate from all other batches, and should be able to execute in parallel without requiring synchronization, except in that referencing the same object in multiple batches can cause contention and will require synchronization on the part of the user.

Adding operations to a batch should be fairly lightweight, with minimal marshalling required to access and manipulate the operations in unmanaged code. The Process method shall iterate over the batch and perform all queued operations, storing in the appropriate handles those results that are indicated to be of interest (ResultType.Output) and optionally discarding those results that are not of interest (ResultType.Ignore).

Handles will be used to reference the results of batch operations for both internal use in a batch, and for retrieving the results of a completed batch of operations. Each Handle will reference the result of but a single operation, but that operation can be the sum result of any number of prior operations (see example above). Handles will not marshal results back from the unmanaged side until requested to minimize the amount of marshalling necessary. Handle objects will provide the appropriate conversion routines to retrieve the type referenced by the handle, these operations will be checked and an exception thrown for invalid cases (such as matrix 🡪 vector, or vector 🡪 bool).