Threading Model

The C# Client is designed to operate in high-frequency, real-time systems like VR rendering. It can only be used asynchronously.

The C# client uses worker threads to perform operations like downloading and decompressing data in the background. Before anything else, the application must initialize this pool of worker threads by calling:

```
VoxelFarm.VoxelFarmWorkPool.Start(NumberOfThreads);
```

Where “NumberOfThreads” is an integer specifying how many background threads should be used.

The “VoxelFarmWorkPool” object can be used to start new tasks either in background threads or in the main thread by calling the “RunInBackground()” or “RunInMainThread()” respectively. These functions take an action as a parameter, which will be queued and will be executed as soon as possible. Since the functions only insert the action into a queue, they return immediately.

The application is responsible for triggering the processing of pending actions in the main thread. To achieve this, the application must call the “RunNextMainThreadJob()” method in the “VoxelFarmWorkPool” object. Applications like client-side report programs will likely do so in loops that wait until the work is done.

It is up to the application coder to enforce thread safety for the behavior inside these actions. In general is it possible to achieve thread safety without using expensive critical sections by making background threads produce its own copy of the result data, and then passing the copy to an action in the main thread that will collate the copy with the final result. Since it is only the main thread accessing the results, all operations remain lock-free.

This code shows an example of this approach:

```
VoxelFarmWorkPool.Start(10);
string finalResult = "";
const int Iterations = 100;
int completedIterations = 0;
for (int i = 0; i<Iterations; i++)
{
   VoxelFarmWorkPool.RunInBackground(() =>
```

```
    {
        string partialResult = Guid.NewGuid().ToString();
        VoxelFarmWorkPool.RunInMainThread(() =>
        {
            finalResult += partialResult;
            completedIterations++;
        });
    });
}
while (completedIterations<Iterations)
{
    VoxelFarmWorkPool.RunNextMainThreadJob();
}
```

This example will initialize the work pool with 10 threads. Next, it will compute 100 random strings (using GUIDs) in parallel. Whenever a thread has computed the new string, a new task is scheduled to run in the main thread so the string can be added to a single final string which will eventually contain the 100 random strings

---

Revision #1
Created 17 March 2025 20:11:22 by admin
Updated 17 March 2025 20:13:19 by admin