Parallel Collections and Actor Threads

9-19-2011
Minute essay comments

- Using threads in projects.
  - AI
  - Needs significant workload.
  - Splitting up drawing.
  - Threading input.
Parallel Data Structures

- BlockingQueue
- ConcurrentHashMap
- CountDownLatch
- CyclicBarrier
- Exchanger
- PriorityBlockingQueue
- Semaphore

Scala provides some support for basic collections.
Locks

- More flexible than synchronized.
- Provides extra power when needed. Particularly for locking across method calls.
Atomics

- Data values with atomic access.
- Faster and easier than doing your own synchronization.
Parallel Collections

- Scala 2.9 introduced scala.collection.parallel.
- The methods of these collections do their work in parallel.
- Covert from regular collections to parallel ones by calling the “par” method.
- Convert back with “seq”.
- Not all collections convert efficiently.
The scala.actors library provides an alternative threading model we will explore in depth later on.

For now there are two methods that simplify launching threads.

- Actor.actor(body: => Unit):Actor
- Futures.future[T](body: => T):Future[T]

Use the first to launch code in a thread. Use the second if you want a return value.
Let's write some.
Questions about parallel before we move on to streams?