while (! isTerminated ()
mutex . lock ();
manager -> notify ();
managerCondition . wait ( lock );
renderer ();
condition . timed_wait (lock , nextFrameTime );
mutex.unlock();
}

Rendering Parallel Views
- dual-core and multi-core processing systems have now become a standard
- multi-threading rendering takes advantage of this trend
- bad multi-threading approach will result in performance loss

Our approach
- support for a high number of parallel render threads
- limited only by the capabilities of the computer
- each render thread has own
  - camera
  - render state
  - refresh rate
- resources are shared among the render threads
  - textures
  - display lists
  - vertex buffer objects

Performance issues
- avoid context switches
- synchronize update and render threads
- no rendering may be active while updating

Straight forward approach
- update scene
- start render processes
- wait for next frame
- same refresh rate for all render threads

Our solution
- Render thread notifies update thread and performs the render process after the update (see Listing 1)
- The update thread (see Listing 2) waits for an update notification...
- ... locks each mutex of all render threads, ie waits until the last has finished rendering and prevents them from starting to render
- ... performs the update and unlocks the mutexes and notifies all render threads
- In case render threads notify the update thread during the update process, they can share the same update.

Rendering Stereo Pairs
- Quad-buffered stereo is not parallelizable
  - left and right view have to be rendered in one thread
- Stereo split view can be used with this approach
  - But: left and right view may display different frames
  - vertical sync may happen between buffer swaps.