Concurrency Report
==========

1. Thread Interaction

- The game engine launches multiple threads: one for each character (Harry, Hermione, Ron, Dementor), the event dispatcher, and the logger.
- Character threads move, pick up items, and interact with the game world concurrently.
- The event dispatcher thread processes game events from a BlockingQueue, coordinating actions and updates.
- The logger thread records game events asynchronously.

2. Potential Race Conditions & Solutions

- **HorcruxFragment Access**: Multiple characters may attempt to examine or solve a Horcrux at the same time.
  - *Solution*: Each HorcruxFragment uses a ReentrantLock to ensure only one character can access it at a time.
- **Room Item Pickup**: Characters may try to pick up items from the same room simultaneously.
  - *Solution*: Synchronized blocks on the room object prevent concurrent modification of the room's item list.
- **Character State Updates**: Health, alive status, and inventory may be updated by multiple threads.
  - *Solution*: Critical fields like `alive` are marked volatile for visibility; inventory updates are performed within synchronized blocks.

3. Thread Coordination Mechanisms

- **ExecutorService**: Manages the thread pool for character actions, ensuring efficient resource usage and lifecycle management.
- **BlockingQueue<Event>**: Used for event dispatching, allowing threads to safely communicate and process events in order.
- **ReentrantLock**: Applied to HorcruxFragments for exclusive access during examination/solving.
- **Synchronized Blocks**: Used for room and inventory access to prevent concurrent modification.
- **Volatile Fields**: Ensures visibility of state changes (e.g., character alive status) across threads.

These mechanisms collectively ensure safe, coordinated multithreaded gameplay and prevent data corruption or inconsistent game state.