Ethan Waldo September 25, 2025

I compared how long it takes to copy a Player versus move a Player between different scenarios:

- Small inventory: 10x10 items (100 total items)
- Large inventory: 2000x2000 items (4 million total items)

Small Inventory (10x10)

- **Copy time:** 5 microseconds
- **Move time:** 0 microseconds
- **Difference:** Move is slightly faster

Large Inventory (2000x2000)

- **Copy time:** 56,118 microseconds
- **Move time:** 1 microsecond
- **Difference:** Move is 56,000 times faster

Analysis:

When working with small amounts of data, copying and moving take about the same time. But when dealing with large amounts of data, moving becomes much more efficient.

Why moving is faster:

- Copying creates a brand new copy of all the data
- Moving just transfers ownership without copying anything
- With 4 million items, copying takes a lot more work than just moving pointers around

Observations:

1. **Size matters:** The bigger the data, the more move operations help with performance
2. **Small data:** For small inventories, the difference isn't very noticeable
3. **Large data:** For large inventories, move operations are essential for good performance
4. **Memory usage:** Moving also saves memory since you're not creating duplicates

Conclusion:

This experiment shows why move semantics are important in C++. When working with large amounts of data, using move operations instead of copy operations can make a program run thousands of times faster. For small data, it doesn't matter much, but for big data structures, move semantics are crucial for performance.