Data Design #1

Continued…

Performance Matters

i.e. ***Optimizability*** matters

“How much does performance matter here?”

Wrong question!

How well can you predict how much it's going to matter?

- Understand the implications of change.
  - Refactoring is ***not*** cheap.

Use case: Serialize/De-serialize

Generate based on *known* data

- e.g. DDL

Cache decisions based on *known* data.

- e.g. Headerparsed data
- All branches use data (i.e. the condition)
- That data can be cached, analyzed and sorted.

Call for all/many, never for one.

- e.g. MentalRayExportVisitor
- Usually safe to assume that one is a degenerate case of many.

Cache likely lookups and relationships

- e.g. XML (and SAX vs DOM)
- Even w/ unreliable, changing format,
- 20x speedup.

Use inter-request coherency information

- e.g. Picking
- Camera location to cache likely objects.
- Objects already sorted by comparison type.

PickVisitor, etc.

- Object cache
- Objects already sorted by comparison type.
- Analyze likelihood for comparison order
  - E.g. Next selected. Avg search space reduced.
- 1::Many for Transform::Data, not 1::1
  - E.g. Frustum versus Spheres (not singular)
- Leave room for broad-phase vs. narrow-phase