Rethinking Game Architecture with Immutability

JACOB DUFAULT

SPONSER: DR. BERNHARD

Milestone 6 Progress Summary

- Work on XNA/MonoGame bindings
- Sample game demo
  - Demonstrates how to achieve key concepts in forge (entity creation, destruction, systems, data, etc)
- User manual documentation
  - Generated via both Sandcastle and Doxygen
  - Doxygen preferable; more usable output (Windows help files vs pure HTML)

Future Work

- Revisiting a key design decision
  - Systems (game logic) apply globally
  - Extra work in content editor was done to ameliorate this issue
- Instead of global systems, instead:
  - An entity requests that a system process it
    - Naturally, this will include all systems contained in the dependency graph of the requested system
  - Each system can store local data per entity
  - Entity data is explicitly meant to be shared across all systems for communication, etc
    - So the *TemporarySystem* cannot access the data the *SpawningSystem* uses in the entity

Lessons Learned

- Immutability is extremely useful
- Favor simplicity over performance
- Don't optimize until profiling
- Multithreading introduces lots of subtle bugs that take significant amounts of time to fix
- Unit tests are awesome
  - Handy for catching multithreading bugs by repeating them hundreds of times and debug/breaking on error
- Dual content/runtime implementations tricky
  - Future: Try an explicit serialization protocol with completely separate content/runtime libraries

Demo

- Runtime Implementation – Unity
- Runtime Implementation – XNA
- Content Implementation – Unity

Questions?

Thanks!