

The Dynamic Component System

Terrance Cohen & Doug Sheahan

4/30/2008

1. What's a *Component System* anyway?
2. Prior work – RCF
3. The DCS
4. Use cases
 1. Head
 2. Explosion
 3. FlockEffector
 4. JobSerializer & Conversation
5. Discussion

- Ask any type of Host for any type of Component
- A step toward *construction by composition*

```
virtual AI::ComponentBase*  
MobyBaseUpdate::ProcessComponentRequest  
(u32 component_type);
```

- AI::ComponentBase [empty]
- Subclasses override, call base, and switch on type

- Satisfies basic requirement
- Easy to understand, use, & maintain
- Fast to request component of type from host

- Components are “baked” into their hosts *
 - Static allocation: hosts *always* have their components
 - Component types must be headerparsed (because hosts are)
- Can't query for component by base type
- No straightforward way of updating all of type
- Need custom code per type per host
- Can't report on utilization

- Terms
 - Type
 - REGISTER_COMPONENT_TYPE(name, max)
 - Handle
 - 16-bit
 - Chain
 - 32-bit – one handle and a next index
 - Host
 - Unique 32-bit handle
 - Has a Component Chain

- Hosts' API

Component*

Allocate(type, host_handle, chain)

Resolve(type, host_handle, component_handle)

Get(type, host_handle, chain)

GetThatImplements(type, host_handle, chain)

- Systems' API

```
i32* GetTypesThatImplement (type, count&)  
bool TypeImplements (type, interface)  
Component** GetComponent (type, count&)  
void UpdateAll<type> ()
```

- Components are allocated when (& while) needed
 - Max may be much lower than potential hosts
- Components are *not* headerparsed
 - Prius can be headerparsed, then used to initialize a component instance
 - Only holds data to *initialize* component, not dynamic runtime data
 - Doug's dynarray trick...
- Can query for component that implements interface

- UpdateAll<>() can happen independent of host
 - Including on SPU
- Straightforward utilization reports (debug screen)
- System handles all component requests without additional code

- Type Registry
 - Per code segment, e.g. Systems, Gameplay
 - Size, max, name, constructor, peak for each type
- For each component type
 - Array of max instances
 - Free/alloc list of same length
 - Free indices and Allocated indices are kept *contiguous*
 - Partition value holds index into free/alloc list of first allocated

- 29 Component types created since November
- Does *not replace* the Classic Component System
 - DCS has Static Max instances per type
 - DCS Handle resolve is small constant, but query host for component of type has small *linear* cost
- Particularly good for
 - On-demand components
 - SPU update
 - Polymorphism
 - Iterating over Damage Modifiers, easy to create new types

- API methods for looking, blinking, expression, and lipsync
- Abstracts fixed/swappable heads so gameplay doesn't care
- Separates code
 - No headerparsing
 - Limited includes

- Allocated to manage an in-progress explosion
- Hosted by instigator, e.g. Damageable
- (Doug?)

- Updated on SPU now as gp shader
- Doug...
 - Build scattered memory DMA by iterating over list of allocated components

- Abstract
- Accepts a sequence of jobs, including pre & post delays & callbacks
- Handles running the jobs in sequence

- Manages a conversation among participants
- Implements JobSerializer
 - Lines implement SerializerJob
- Allocated on-demand, freed upon completion
- Scriptable
- Hosted “anonymously”

- More from Doug...
- Questions

