Entity Component Systems

Or: Being Able to Make Anything in the Game Explode

Who am I?

Stephen Whitmore

Indie Game Developer, Open Source Enthusiast, Photographer.

Enthusiasm makes up for the rest.

What Have I Made?

Probably nothing you’ve heard of.

***TRUSTY SWORD vs ROCKET LAUNCHER*?**

***Robot Story*?**

***Alone in the Rain*?**

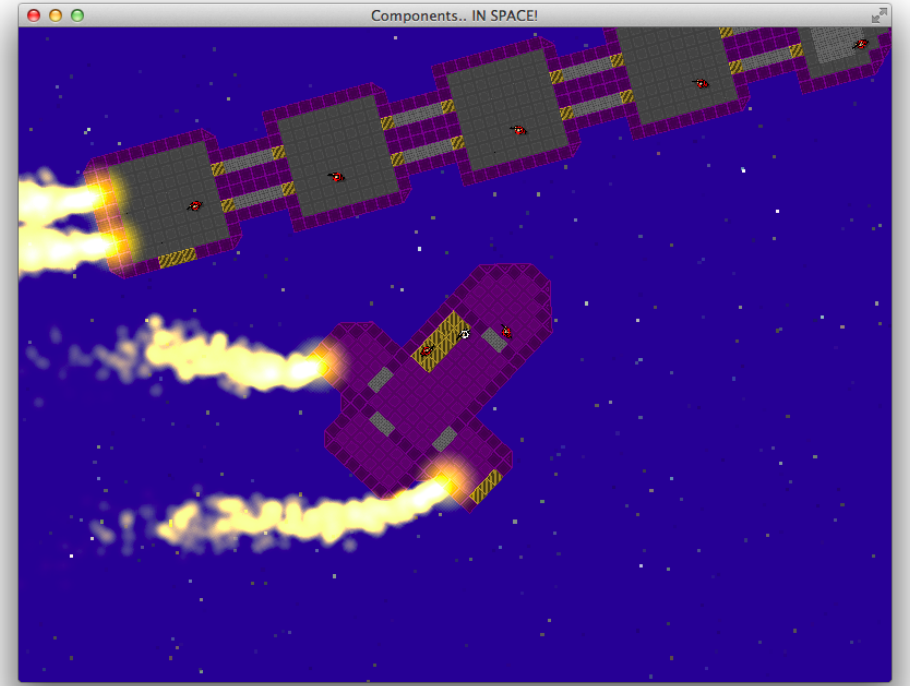
Membrane Massacre?

(That’s cool. Don’t feel bad. I don’t.)

What am I Making? </plug>

Boarding Action

A game about fighting your way aboard big spaceships and exploring the galaxy.



Entity-Component-Systems

Definition: *Entity-component-system is a software architecture pattern that separates the functionality into individual components that are mostly independent of one another. (Thanks, Wikipedia.)*

Yeah, It’s vague. There is no single, undisputed implementation of E-C-S.

Entity-Component-Systems

One of the first patently “E-C-S games” was *Dungeon Siege*, of 2002 fame.

Adam Martin’s work on *Operation Flashpoint: Dragon Rising* formalized some of the original ideas.

Phrases like, *"Entities as ID's"*, *"Components as raw data"*, and *"code stored in Systems, not in Components or Entities"* should all be semantically meaningful by the end of this talk.

Entity-Component-Systems

Origins of my hearing about E-C-S?

Funny coincidence: ***Cogswell***. *Hold your heads up high, folks.*

Entity-Component-Systems

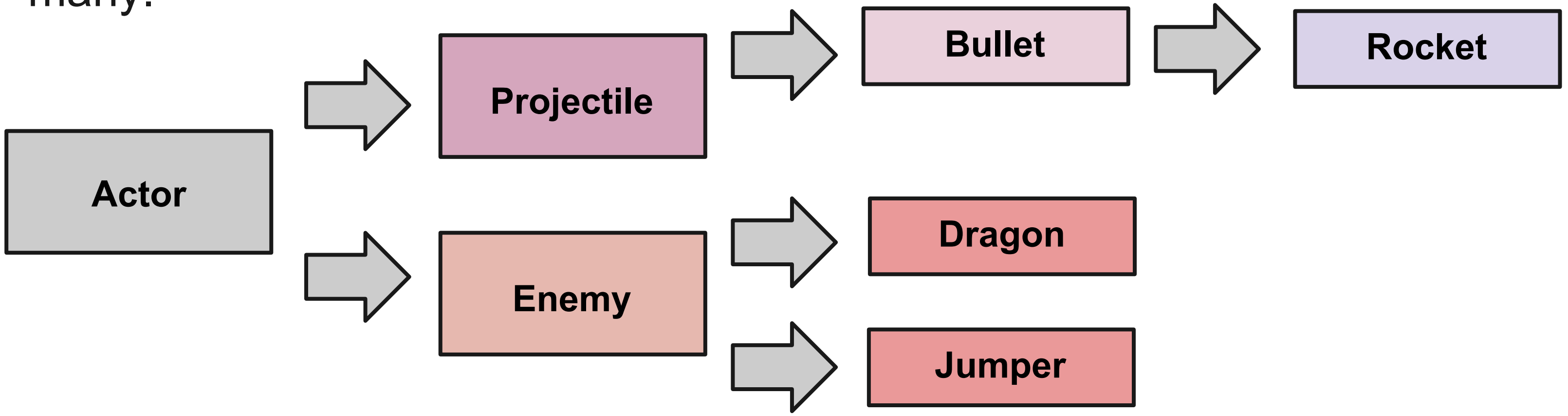
Oh, yeah. A disclaimer:

You’re at the mercy of a man who is not an expert.

OOP: The Holy Grail?

..maybe?

Object Oriented Programming gives a nice, hierarchical model of reasoning about how games can be logically structured. It's been the golden standard for many.



OOP: The Holy Grail?

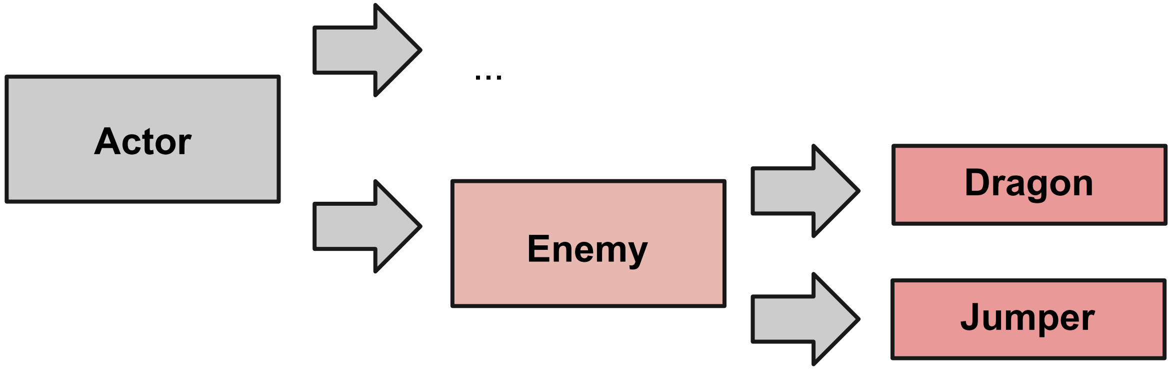
But this is fundamentally challenging: how can the developer be expected to have a complete understanding of all types of game objects to know what hierarchical relationship to arrange them in?

What if s/he did, but requirements changed?

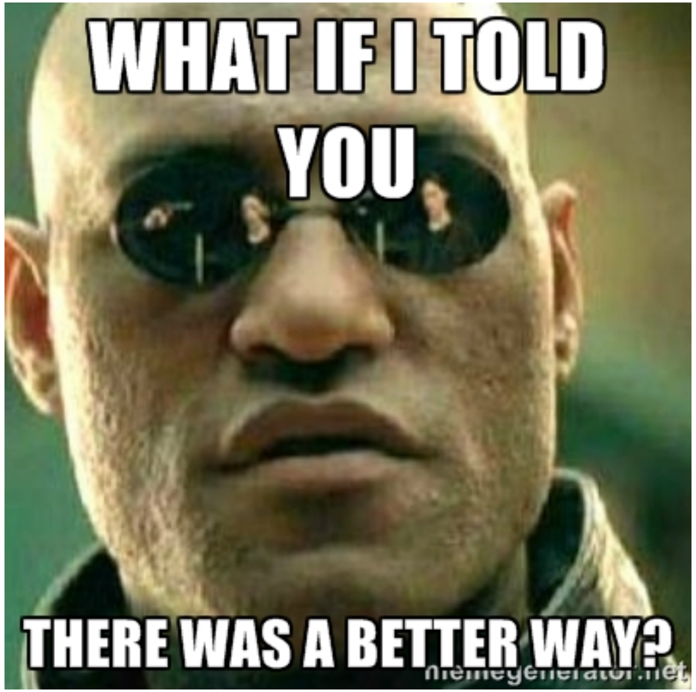
OOP Problems: New Type?

Imagine you have to add a new type of object to the game: *where do you put it?*

As your hierarchy grows, the task of tactically placing new types into the tree becomes harder as you decide which functionality you want to derive from existing types, and which functionality you have to sacrifice.

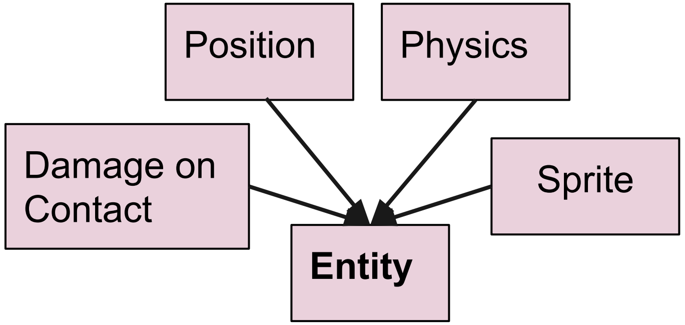


Does This Coupling Need to Exist?

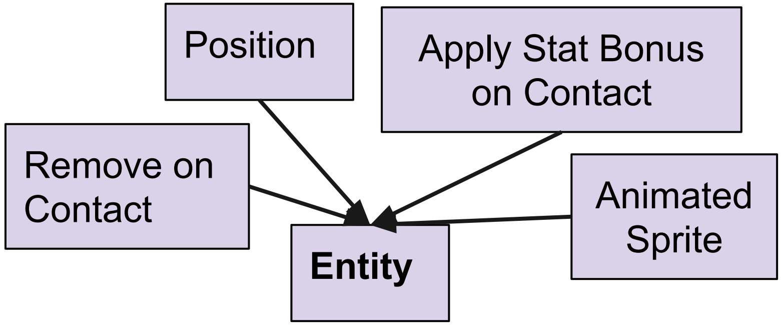


Favour Composition over Inheritance

What if every game object was a collection of data that defined behaviour?



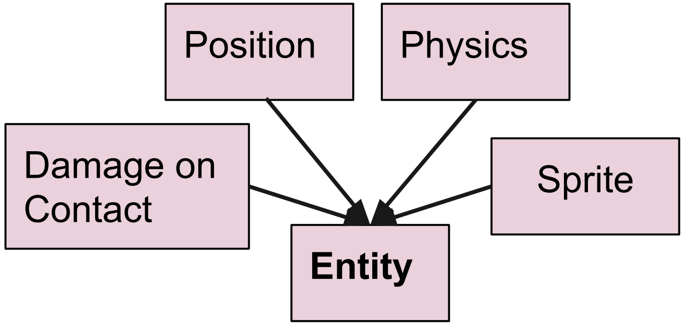
Projectile



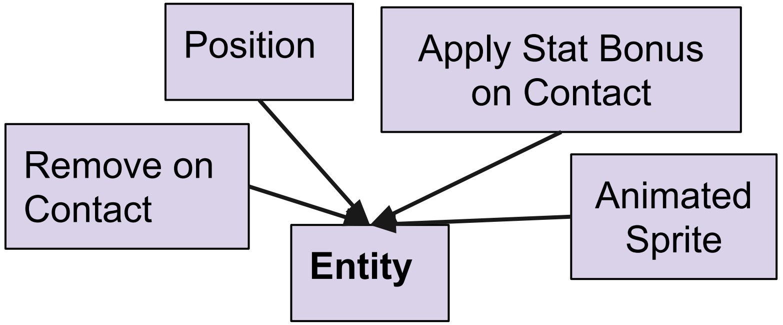
Animated Power-Up

Favour Composition over Inheritance

More specifically: what if every game object was just a collection of data-only components?

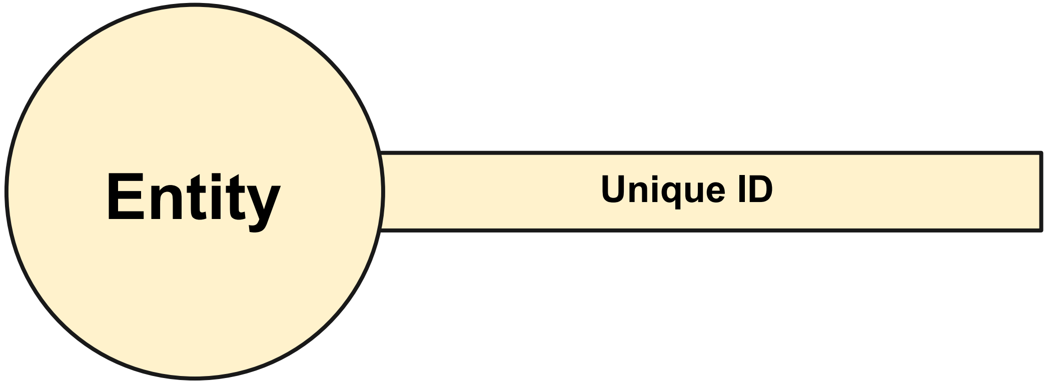


Projectile



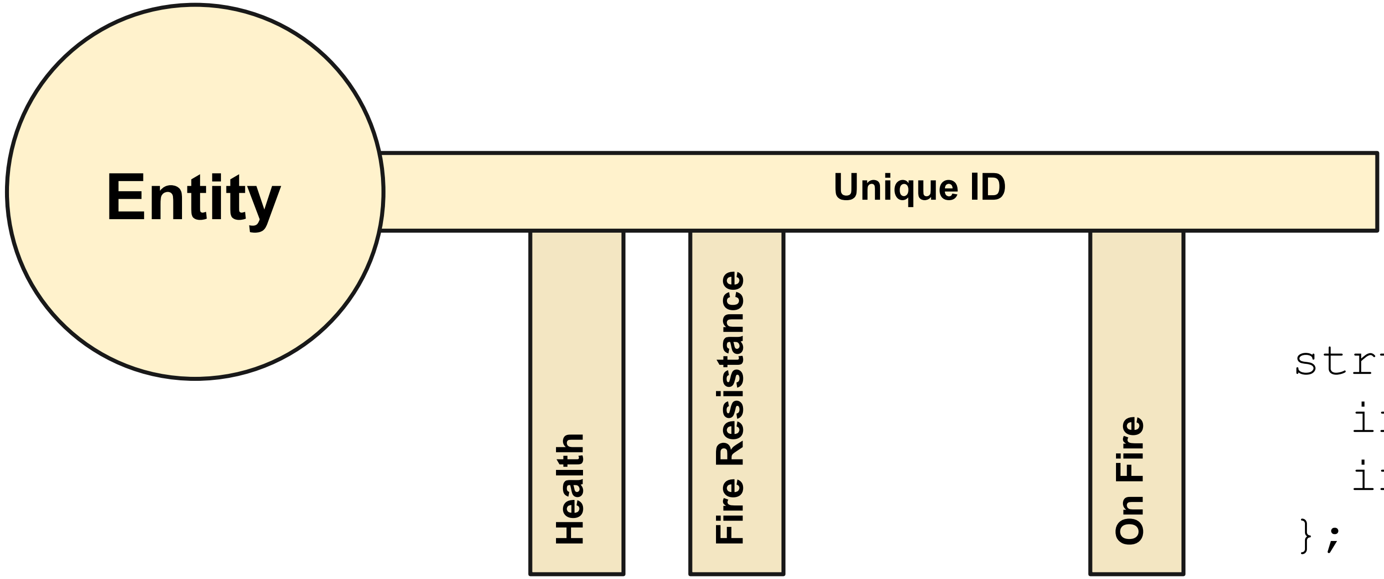
Animated Power-Up

E-C-S: The Entity



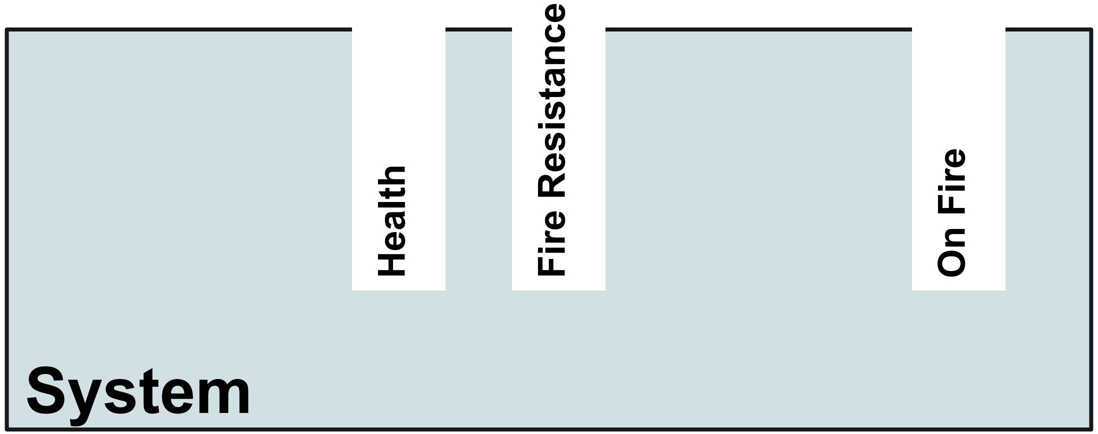
```
struct Entity {
  long id;
};
```

E-C-S: The Component



```
struct Health {
  int total;
  int current;
};
```

E-C-S: The System



```
void process(Entity
e) {
   ...
}
```

Example Configurations

HUD Icon (ScreenPosition, Sprite)

Scrolling Background (ScreenPosition, Sprite, ParallaxScrolling)

Bullet (Position, Physics, Sprite, SelfDestructCollisionResponse)

Player (Position, Physics, AnimatedSprite, InputController, KeyboardMap)

Enemy (Position, Physics, AnimatedSprite, InputController, WalkAroundAi)

Demo Time!

Listen, making slides is a lot of work.

So I’m just going to go ahead and expand this example to show you some cool things you can do with E-C-S.

E-C-S Libraries

Java, C#: **Artemis**

JS: **CraftyJS**

ActionScript, Haxe: **Ash Framework**

Dangers

- Performance can quickly become a problem
- System soup
- Marker components

Helpful Patterns

- Inter-system communication
- User input (keyboard, mouse, gamepad)
  - Controller, KeyboardBindings, *AI
- Entity hierarchicies
- Scripting
- Entity definition deserialization

Resources: Articles

- Entity Systems are the future of MMOG development
- Key/Lockhole explanation of E-C-S
- Dealing with hierarchies of entities
  - **Key idea**: E-C-S is meant is remove inheritence and hierarchies of classes. Hierarchies of other forms, such as composing hierarchies of entities, is by no means verboten
- Evolve Your Hierarchy (so-so)
- Understanding Component Entity Systems
  - Decent, but more like an overview that goes into zero implementation detail
- Richard Ash: What is an entity system framework for game development?
  - Pretty long; difficult to follow if you aren't already somewhat familiar with the ideas behind E-C-S
- Richard Ash: Why use an entity system framework for game development?
  - More accessible, though some knowledge of E-C-S assumed

Resources: Examples

I found studying others’ games critical: developing an intuition for how to re-approach my usual OOP-rooted ideas for modeling game aspects into entities, systems, and components was essential -- otherwise it’s very easy to fall back onto my old, traditional OOP mindset.

- Richard Lord’s *Asteroids*
- *Spaceship Warrior*
- Case Study: Bomberman-like Game
- Jario (Mario-like platformer)

Fin

Questions?