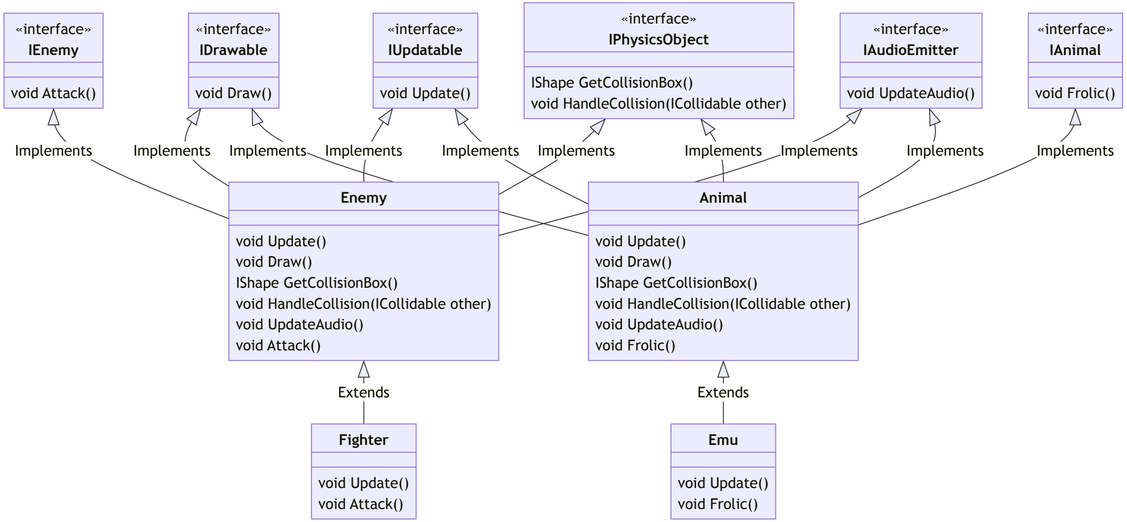
CSE 3902: Entity Component Systems

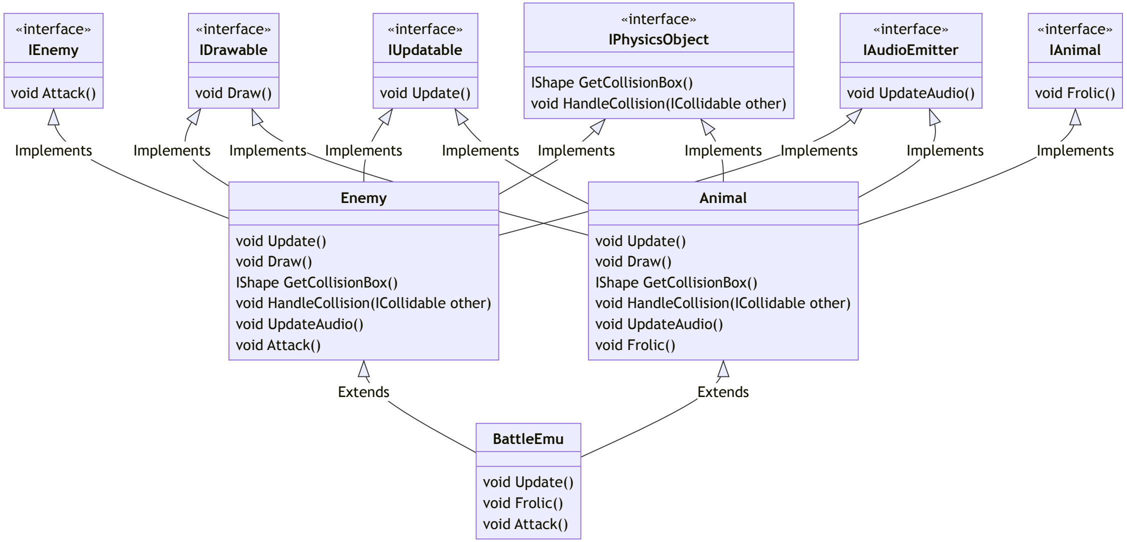
Justin Holewinski

The Ohio State University

Revisiting Game Object Design - Typical OO Design



Revisiting Game Object Design - The BattleEmu



Entity Component Systems

Entity: A game object composed of components

Component: Property attached to an entity

System: Code that acts on components

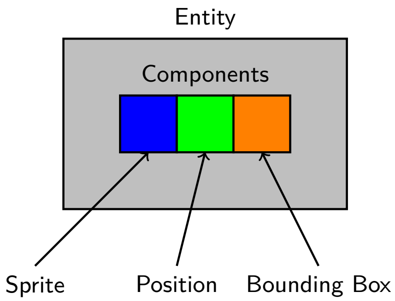
Key Idea: Separate data from code

- An entity contains components, but not code to act on the components
- A component contains data, but not code to act on the data
- A system contains code to act on component data, but not (game object) data itself

Composition over inheritance

- Entites are *composed* of components, instead of *inheriting* classes

Entities and Components



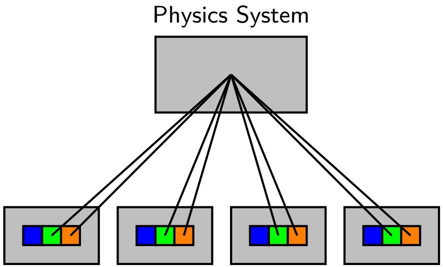
An entity is essentially a container for components

A component is plain data

- Sprite
- Position/Velocity
- Collision Geometry
- Health
- Audio Emitter
- Trigger

```
public struct PhysicsComponent
{
    public Vector2 Position;
    public Vector2 Velocity;
    public float Mass;
}
```

Systems



A *system* is a piece of code that operates on components

- Read and manipulate component data
- Can involve multiple components per entity
- Examples
    - Draw sprites
    - Detect and resolve collisions
    - Play sounds

Advantages

Solves multiple-inheritance issue (mostly)

- OOP favors moving code to common base classes using *extension*
- But *extension* can easily lead to diamond inheritance
- Remember the `BattleEmu`!

Separation of code from data

- Entities are just chunks of data
- Can reduce code duplication

Components can be stored in cache-friendly structures

- List of sprites
- List of positions
- List of bounding boxes
- Better for cache locality when iterating over components of the same type