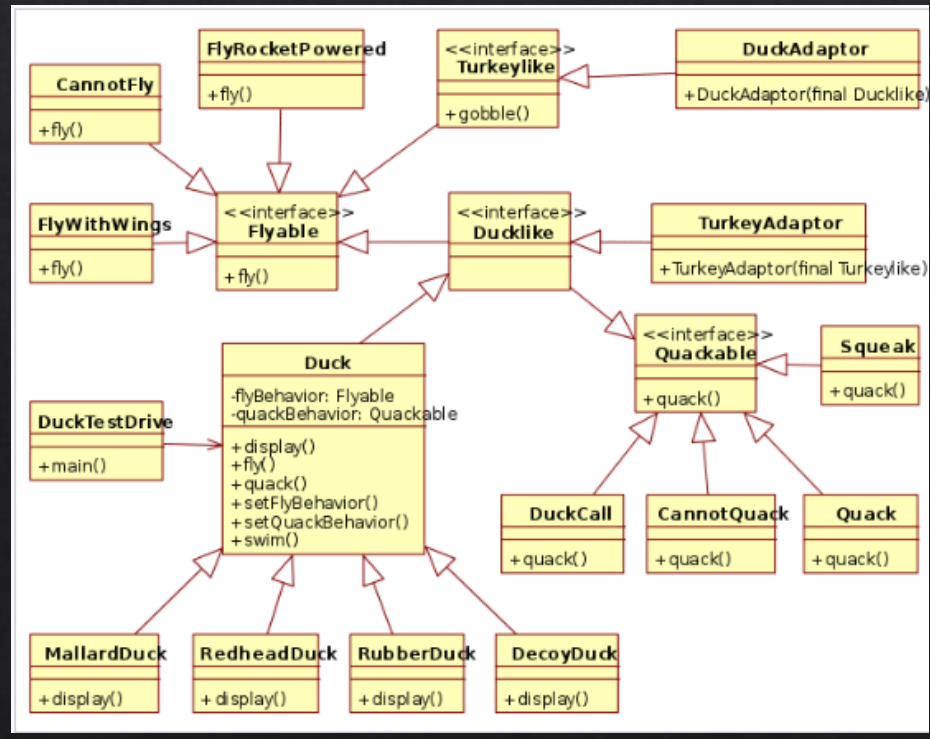
ETGG3802

Lecture4: Entity-Component System (ECS)

Software Design: Composition > Inheritance (?)

- Problem setup:
  - Class GameObject. Virtual functions (empty body) for Update, Draw, Collide
  - Class Mover with Update function that has a body
  - Class Solid with Collide function that has a body
- User wants to create a few "concrete" classes
  - Class Trigger : public GameObject, public Solid {};
  - Class Player : public GameObject, public Solid, public Mover {};
- Problem: Multiple Inheritance + DDD (Deadly Diamond of Death)
- Another Example:
  - From https://en.wikipedia.org/wiki/Composition_over_inheritance



Composition

- Have a container (GameObject in our case / Unity's) class
- Have "components" that you snap-into that container to add functionality
  - Like BoxCollider, ParticleEmitter, SoundSource, etc. in Unity
  - In C++, these classes are derived from a common base class (Component)
- Inheritance still plays a role, but this is an alternative.
  - Discuss in the Duck problem

Polymorphism in C++

- Virtual
- Pure Virtual
- Inheritance
- Base class pointing to derived class
  - Containers of Component objects (that really point to derived classes)

C++ circular includes

- #pragma once
  - (and the old-fashioned version)
- Situation where this doesn’t help
- The fix

C++ std::map review

- Creating
- Adding pairs
- Getting matching value
- Finding a key
- Iterating through
- Removing from

Our ECS

- Component (base) class
  - A pointer to the containing game object
  - Pure virtual methods:
    - `Component::ComponentType getType();  // ComponentType is an enum of all components`
  - Virtual methods (with empty body)
    - `Void setVisible(bool);`
    - `Void update(float dt);`
- Derive multiple component classes from this base class
- GameObject has…
  - a map (ComponentType => Component*)
  - Methods to create / add a component (e.g. createMeshComponent(fname))
- GOM update function
  - Update all GameObjects
  - Each GameObject updates its components.

Optional Improvement

- Cache coherency?
- Allocate all XYZComponents from a pool (contiguous in memory)
- Update all

[Prep for Lab5]

- Get an XML library and build from source (dll / shared project) – tinyxml2 recommended
- Incorporate into our project
- Recursively visit each node in the document
- TinyXML2 tools to do that (we'll look at more in the next lab):

```
tinyxml2::XMLError tinyxml2::XMLDocument::LoadFile(char * fname);
tinyxml2::XMLElement * tinyxml2::XMLDocument::RootElement();
const char* tinyxml2::XMLElement::Name();          // The "tag" name
tinyxml2::XMLElement* tinyxml2::XMLElement::FirstChild();
tinyxml2::XMLElement* tinxml2::XMLElement::NextSiblingElement();
```