UObjects basics

- What is a UObject?
- Constructing a Uobject with NewObject<>
- Class default objects (and detecting it in constructors)
- IsValid() to check if a Uobject is still valid
- WeakObjectPtr<> to reference Uobjects outside the Uobject system
- Uobject::AddReferencedObjects if you need to keep references outside Tarray or Tmap
- FGCObject if you need AddReferencedObjects outside the Uobject system

UObjects and garbage collection (see further slides)

- How does garbage collection work, inc AddToRoot and RemoveFromRoot for root references
- How does GC know about properties? UPROPERTY() and GENERATED_BODY() magic
- How does GENERATED_BODY() work?
- UnrealHeaderTool (UHT) scanning source code and generating the .generated.h files
- GENERATED_BODY expanding to "GENERATED_BODY_<Filename>_<Line>" which maps to define inside .generated.h; aka how GENERATED_BODY() expands to different things for each class & file
- UPROPERTY tells garbage collector what to track off each UObject; also why you need it on every field that is a UObject pointer or has UObject pointers, even if it's not exposed to blueprints or replicated

GENERATED_BODY magic

- GENERATED_BODY() expands to include CURRENT_FILE_ID and __LINE__
- .generated.h contains all the extra fields for that class that are required for Uobject system including StaticClass(). Also contains Super and ThisClass type definitions, which is what makes Super::BeginDestroy() etc. work
- .gen.cpp contains registration code and that file is part of the build due to UBT
- How does registration code work? Constructor behaviour of global variables in C++, see https://godbolt.org/z/811Yhh1Y7

How does the engine know about properties?

- .gen.cpp file contains all property definitions, their types, metadata etc
- NewProp_<propertyname> inside statics
- Initializer list to set up all the metadata
- PropPointers contains list of properties
- Passed into ClassParams for registration

Garbage collector: how does scanning work

- Root set of objects (includes the “Level” object)
- GarbageCollection.cpp for internals
- ReferenceTokenStream on Uobjects track references from an object (i.e. from properties)
- UClass::AssembleReferenceTokenStream iterates over the properties in the class to discover possible references using TFieldIterator
- GC then determines unreferenced objects by iterating through root set and reference token streams to find out what is referenced
- GC is parallel in UE so actually untangling code from here on out is fairly complex, but high level design principals apply:
    - Root set objects are kept
    - Properties are recursively scanned to find out what is referenced
    - GC does BeginDestroy/FinishDestroy and deletes anything not referenced

Garbage collector: actor destruction

- If actors can be referenced in properties by other actors, how does Destroy / DestroyActor work?
- DestroyActor does some cleanup:
    - Checks: can you delete this actor? You might not be able to in networked games
    - Runs Destroyed event, detaches components, etc.
- Removes actor from "Actors" array in Ulevel that owns it
- Marks it as pending kill which causes refs to it to be set to null by GC and cleaned up