CS354P

DR SARAH ABRAHAM

COMMUNICATION IN UNREAL

COMMUNICATION IN A GAME ENGINE

- Fundamental to a game engine's design
  - How should systems communicate?
  - How should objects communicate?
- Choices in communication will effect every other system in the game

QUERYING THE WORLD

- All Actors (in fact all UObjects) have a GetWorld() method
  - Accesses the current world (or level) the actor exists within
  - Note: will return `null` if actor is not currently spawned
- Useful for working with the world space or other objects that exist in that space
- Accessing the world:
  - `AActor->GetWorld()`
  - `GEngine->GetWorldFromContextObject(const UObject * WorldContextObject)`★

★ Observation: `GEngine` is static so it uses the `WorldContextObject` to determine which World that object is in

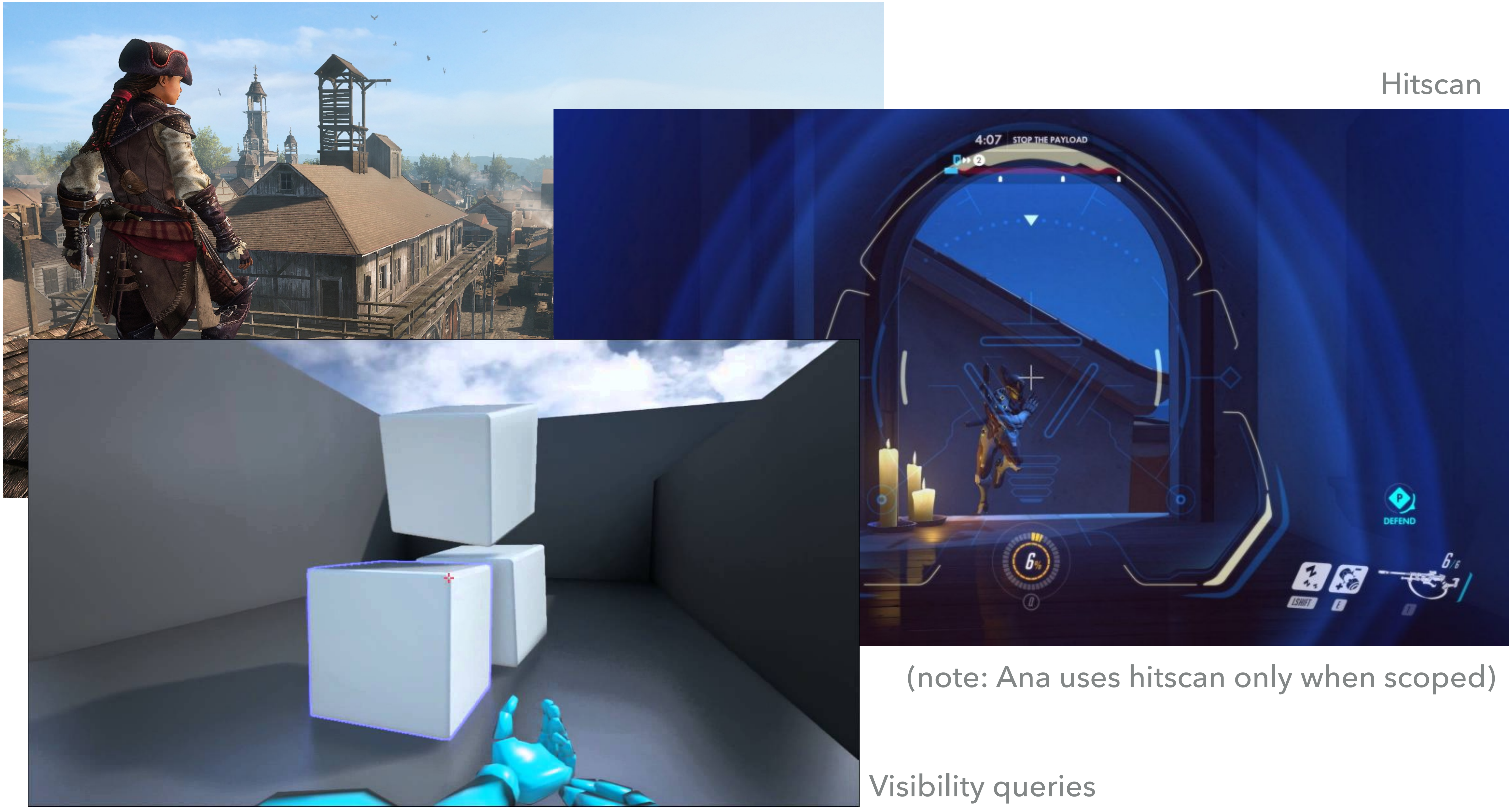
RAY-CASTS AND SWEEPS

- Ray-casting, or sweeping, is a common way to check for intersects along a ray or line segment
  - Can trace by **channel** or by **object type** for efficient results
  - Can choose whether to return single or multiple hits (i.e. get the first object intersected or every object intersected)
- Sweeps track **blocking** intersections encountered by an object
  - Can sweep by channel or by object type
  - Can choose whether to return a single intersect or multiple intersects

RAY-CAST EXAMPLES

Climbing/Parkour

Hitscan

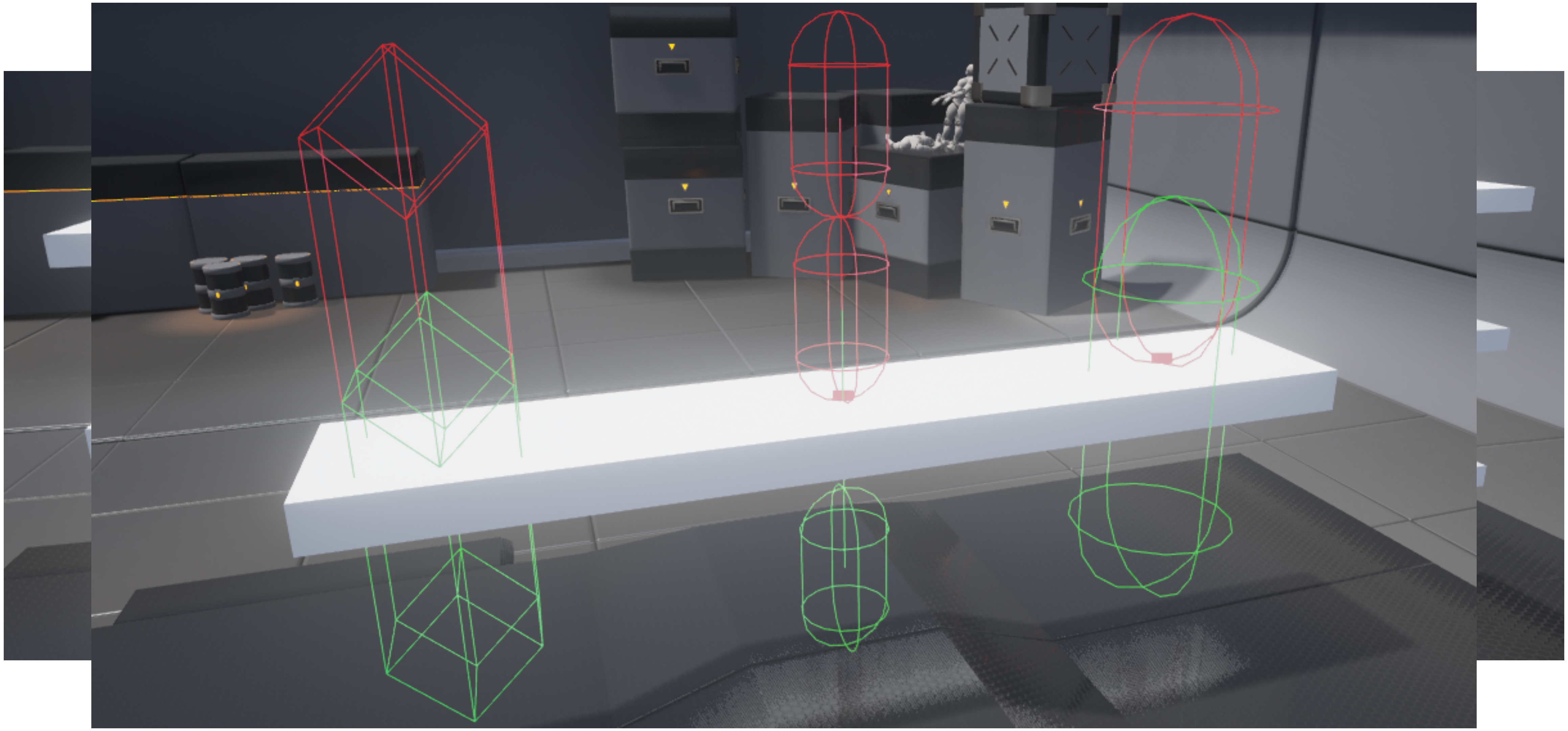


(note: Ana uses hitscan only when scoped)

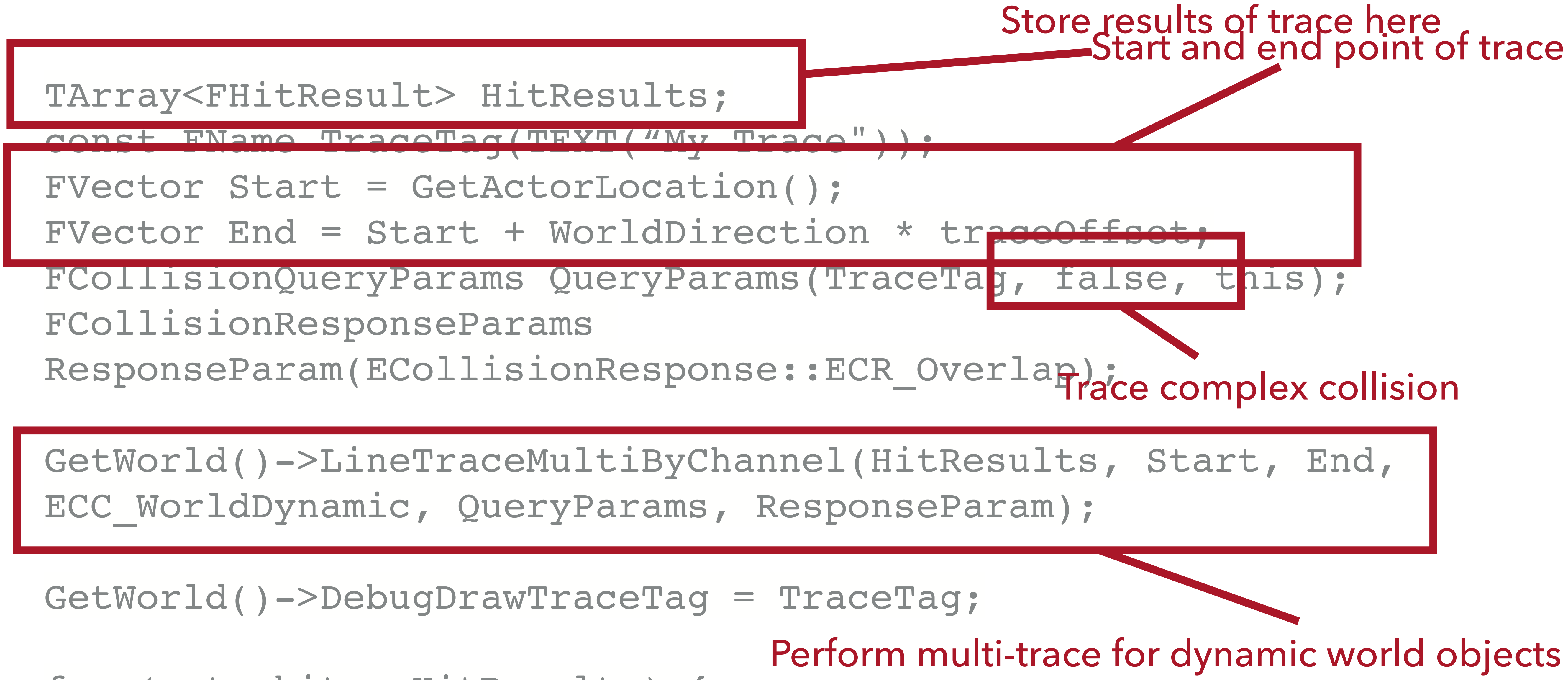
Visibility queries

UNREAL TRACE TYPES

- Line traces use the traditional ray-cast concept
- Also possible to trace with a box, capsule, or sphere



EXAMPLE: TRACING AND DEBUGGING CODE



```
TArray<FHitResult> HitResults;
const FName TraceTag(TEXT(“My Trace"));
FVector Start = GetActorLocation();
FVector End = Start + WorldDirection * traceOffset;
FCollisionQueryParams QueryParams(TraceTag, false, this);
FCollisionResponseParams
ResponseParam(ECollisionResponse::ECR_Overlap);

GetWorld()->LineTraceMultiByChannel(HitResults, Start, End,
ECC_WorldDynamic, QueryParams, ResponseParam);

GetWorld()->DebugDrawTraceTag = TraceTag;

for (auto hit : HitResults) {
    //Process hit results here
}
```

Store results of trace here

Start and end point of trace

Trace complex collision

Perform multi-trace for dynamic world objects

SWEEPS IN PRACTICE

- SweepByChannel methods used to determine if an actor has collided with a blocking object (`SweepSingleByChannel`) or multiple blocking objects (`SweepMultiByChannel`)
- `bSweep` is a flag used in to determine how an actor should move to a given location
  - If true, the actor can be blocked by geometry from reaching the given location
  - Used in methods such as `SetActorLocation`

EVENTS

- In event-driven programming, everything happens in response to events
    - Popular paradigm for GUI systems and other applications with lots of user interactions
- Events occur **asynchronously** with respect to the execution of the rest of the program
- When a particular type of event arrives, the **callback** code is executed automatically

BLUEPRINT EVENTS

- Unreal’s main event system is specifically for Blueprints in the EventGraph
  - Built in Blueprint events such as BeginPlay
  - Custom events created via Blueprint or the macro BlueprintImplementableEvent
- EventGraph manages the nodes to determine how and when Blueprint events are executed
- Events otherwise not supported directly for Unreal…

DELEGATES

- Unreal uses **delegates** for executing functions on C++ objects
- A delegate contains a reference to another object's function and can execute that function
  - Allows objects to "act on behalf of" another object (i.e. delegation)
  - Events use delegates as the mechanism for callbacks
- Broad and fairly ambiguous term but here we will specifically assume delegates are function pointers

UNREAL DELEGATES

- Called in a generic, type-safe way
- Can be bound dynamically to an arbitrary object’s function
  - Caller does not need to know object’s type
- Passed by reference to avoid memory allocation on the heap
- Three types:
  - Single
  - Multicast
  - Dynamic

HOW DELEGATES WORK

- Since delegates are function pointers, they can be bound to valid functions
  - Functions must match delegate's expected signature
  - Functions bound to the delegate will be executed in the **reverse order** they were bound

TYPES OF DELEGATES

- Single Delegates: only one function can be bound
  - Called with Execute
- Multi-cast Delegates: multiple functions can be bound
  - No return values
  - Called with Broadcast
- Dynamic Delegates: dynamic binding of function
  - Can be serialized and functions found by name
  - Called with Execute (return values)/ExecuteIfBound (no return values)
- Note: executing a single delegate with no bindings can cause issues in memory, since they can return values (not an issue for Multi-cast Delegates)

DELEGATES IN ACTION

- Projectile Example:
- `ProjectileMesh->OnComponentHit.AddDynamic(this, &ALab1Projectile::OnHit);`
  - `AddDynamic` is a helper macro used with dynamic multi-cast delegates
  - Dynamically binds to the function name provided as the second parameter
- Delegates are intimately connected to events and the event system (user-generated events)
- Also useful for system-generated events (events created by the system itself)

UNREAL TIMERS

- Timers handled through the `TimeManager` associated with the World
    - `GetWorldTimerManager()`
- Use TimerHandles to distinguish timers with identical delegates
    - Can keep a reference to this handle to clear or pause the unique timer

USING TIMERS

- A common timer bound to a function without parameters:

  -
    ```
    SetTimer(FTimerHandle & InOutHandle, UserClass * InObj, FTimerDelegate::TUObjectMethodDelegate_Const< UserClass >::FMethodPtr InTimerMethod, float InRate, bool InbLoop, float InFirstDelay);
    ```

```
GetWorldTimerManager().SetTimer(myTimerHandle, this, &MyClass::Callback, 5.f, true, 0.f);
```

- A common timer bound to a function with parameters:

  -
    ```
    SetTimer(FTimerHandle & InOutHandle, FTimerDelegate const& InDelegate, float InRate, bool InbLoop, float InFirstDelay);
    ```

```
FTimerDelegate myTimerDelegate = FTimerDelegate::CreateUObject(this, &MyClass::Callback, parameter1, parameter2, parameter3);

GetWorldTimerManager().SetTimer(myTimerHandle, myTimerDelegate, 5.f, true, 0.f);
```

CREATING CUSTOM DELEGATES

1. Declare your delegate using a macro based on the function signature

```
DECLARE_DYNAMIC_MULTICAST_DELEGATE(FMyDelegate);
```

- This function does not have any parameters
- This declaration supports multiple entities (multi-cast) and delegates that can be saved/loaded within Blueprints (dynamic)
- By convention you should prefix with F

2. Declare the delegate in the .h

```
FMyDelegate OnEventMyDelegate;
```

3. Bind a function/functions to the delegate

```
ActorWithDelegate->OnEventMyDelegate.AddDynamic(this, &MyClass:Callback);
```

4. Broadcast when the event should occur

```
ActorWithDelegate->OnEventMyDelegate.Broadcast();
```

DIFFERENCE BETWEEN AN EVENT AND A MULTI-CAST DELEGATE?

- Not much in practice! Events are types of multi-cast delegates
- Any class can bind an event, but only the class that declares the event can invoke the `Broadcast`, `IsBound`, and `Clear` functions
    - Has better encapsulation as event objects are exposed publicly but do not reveal delegate class's internal workings

WHY USE CUSTOM DELEGATES?

- If you need to do something via C++ rather than Blueprint, you will need to
- Useful in situations where the non-delegate object should execute/broadcast a function related to another object
    - Example: Player class performs action that broadcasts to all interactable objects. Interactable objects handle delegation and response to simplify player package
    - Example: Information about player interactions within GUI are passed to objects in the world, which then handle implementing the expected behavior themselves

FURTHER READING

- A full code explanation of how to create delegates in UE5
  - <https://www.orfeasel.com/using-delegates/>
- An overview of delegate types and explanations about using them with Blueprint
  - <https://unreal.gg-labs.com/wiki-archives/macros-and-data-types/delegates-in-ue4-raw-c++-and-bp-exposed>