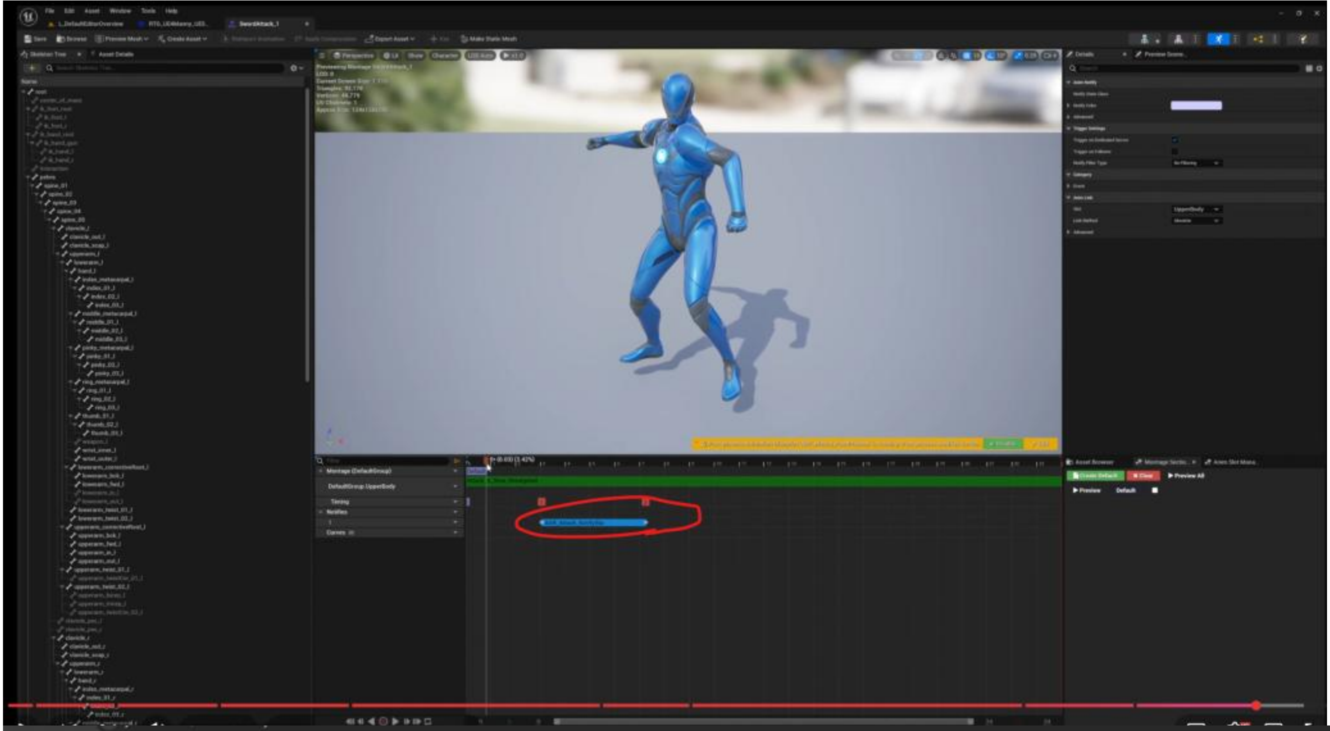
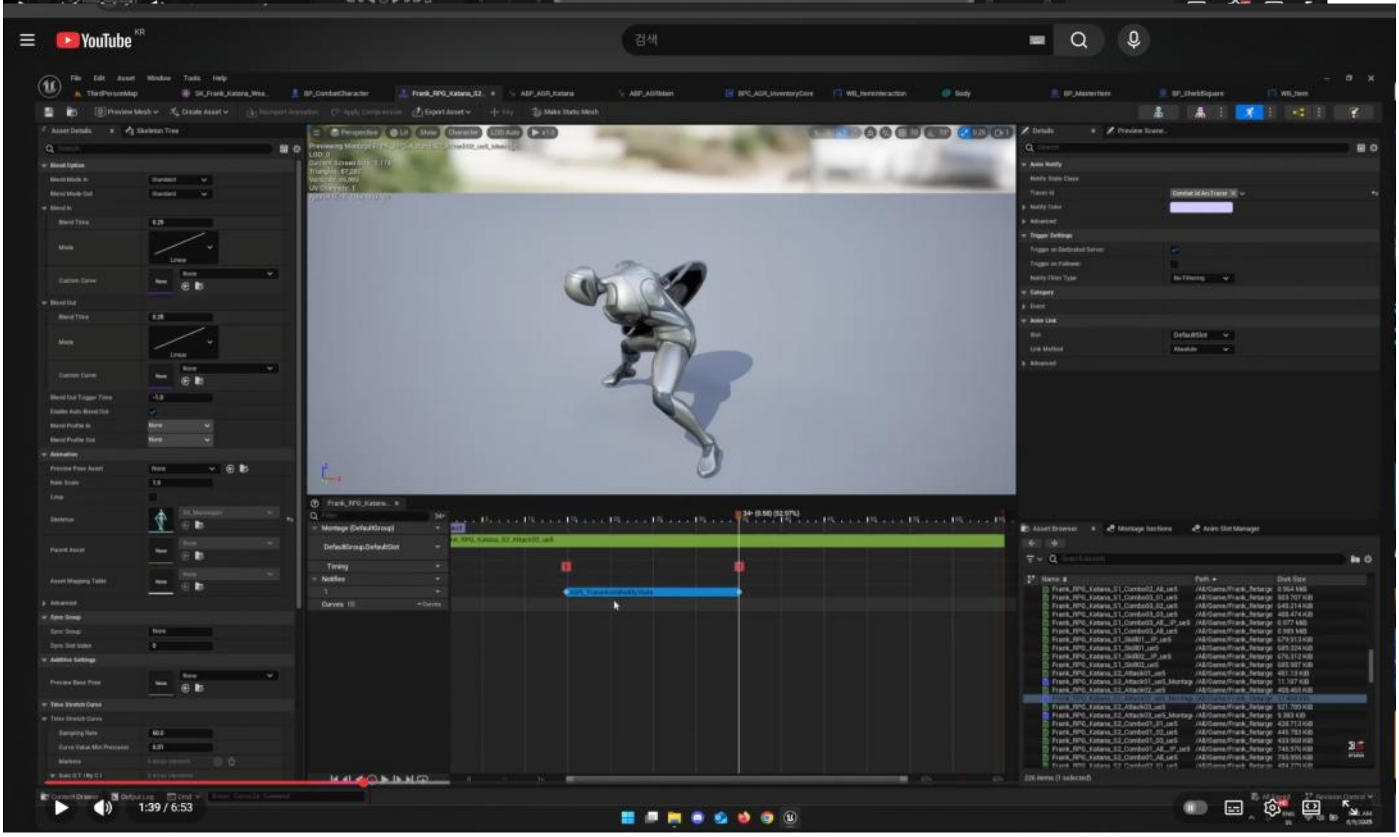
# Role
You are an Expert Unreal Engine 5 C++ Plugin Developer.
Your specialty is Action RPG combat systems, Network Replication, and Physics Collision queries.

# Task
I need to implement a "Melee Combat Trace System" as a standalone Unreal Engine Plugin.
This system must be highly accurate, performant, and support Multiplayer (Client -Side Detection, Server Validation).

Please analyze the attached design document and screenshots, then provide the C++ implementation strategy and core code.

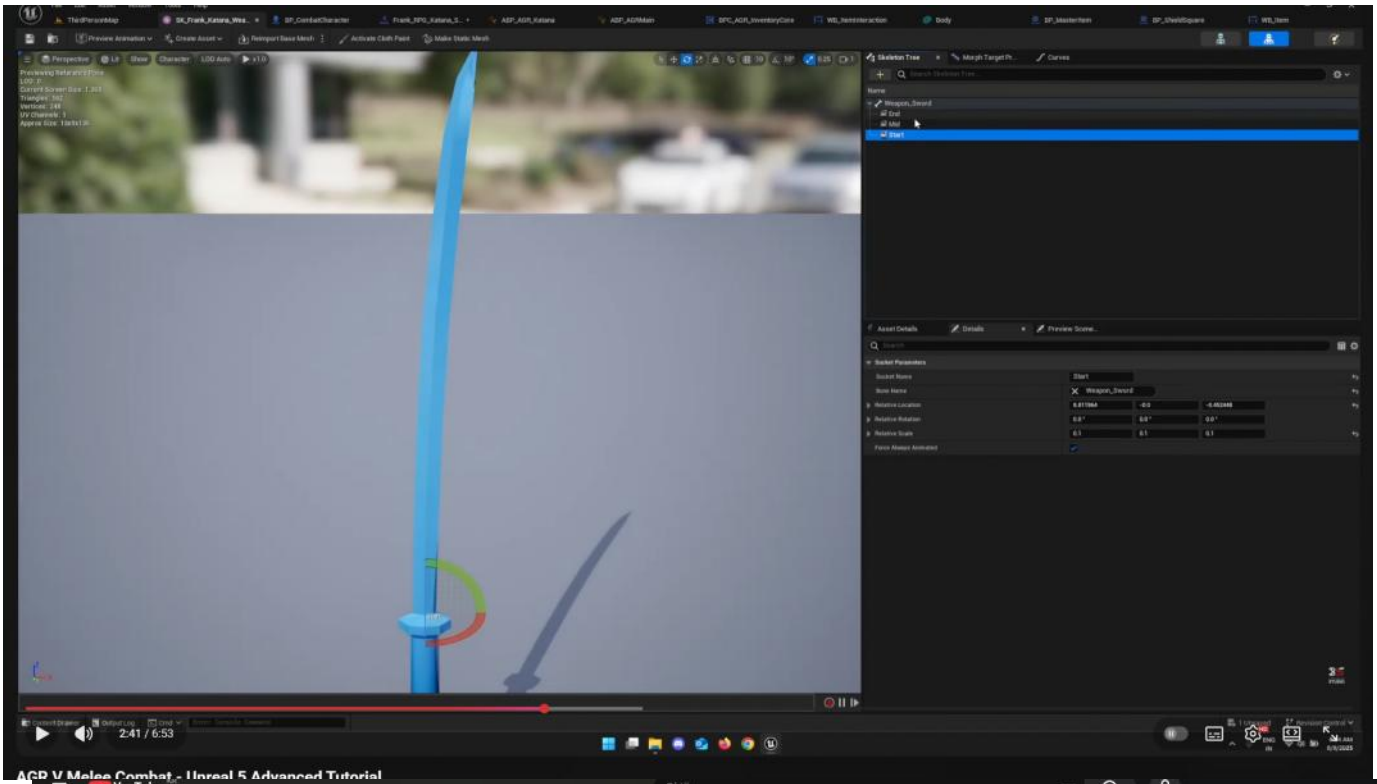
# Context & Visual References
Here is the design concept and the expected behavior in the editor.
1. **Overall Concept & Hit Detection Visualization:**





- Key Point: The trace must follow the weapon's trail accurately.

2. **Socket & Weapon Setup:**



Skeleton Tree

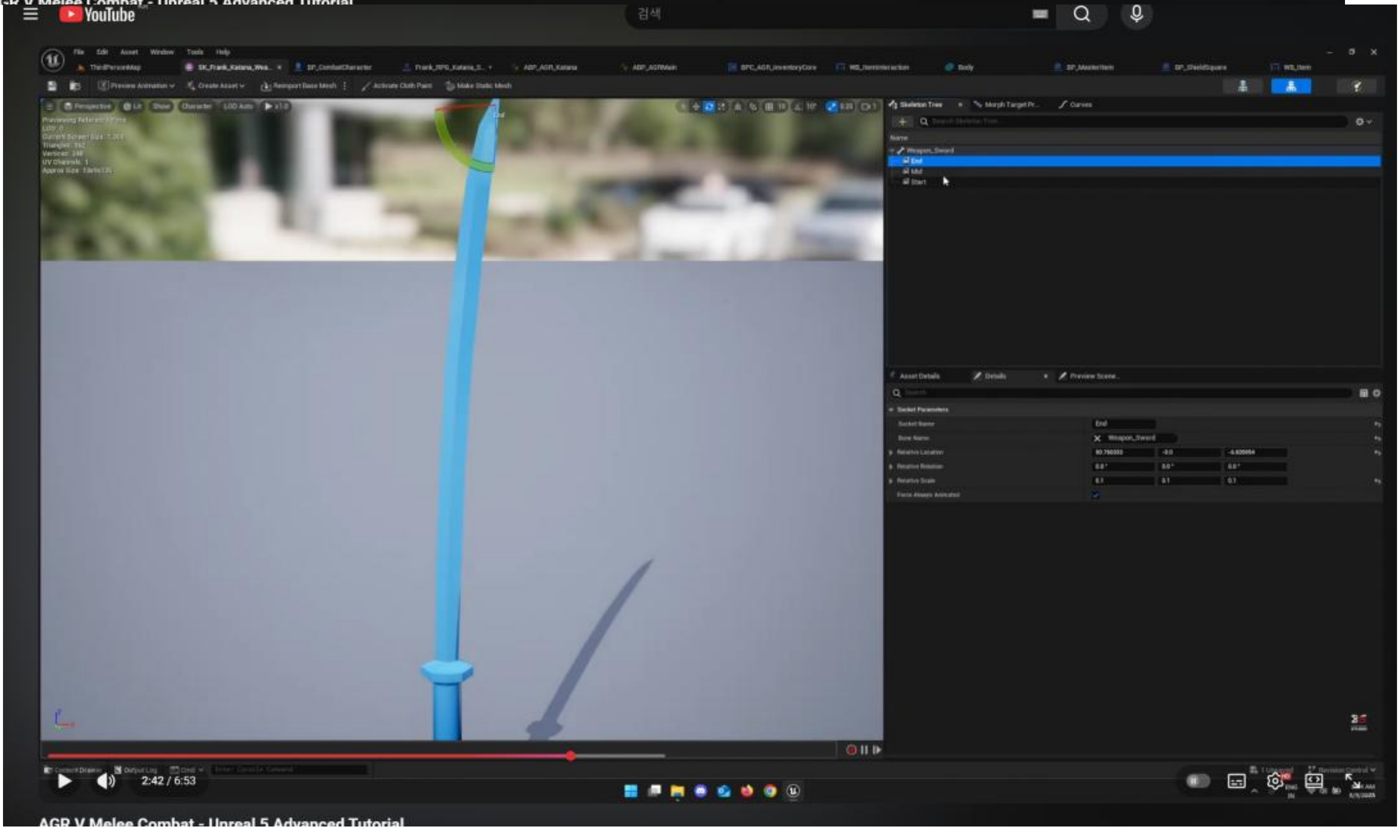
- Weapon_Sword
  - End
  - Mid
  - Start

Socket Parameters

| | | | |
|---|---|---|---|
| Socket Name | Start | | |
| Bone Name | Weapon_Sword | | |
| Relative Location | [illegible] | -0.0 | [illegible] |
| Relative Rotation | 0.0° | 0.0° | 0.0° |
| Relative Scale | 0.1 | 0.1 | 0.1 |
| Force Always Animated | ☑ | | |

2:41 / 6:53

AGR V Melee Combat - Unreal 5 Advanced Tutorial



Skeleton Tree

- Weapon_Sword
  - End
  - Mid
  - Start

Socket Parameters

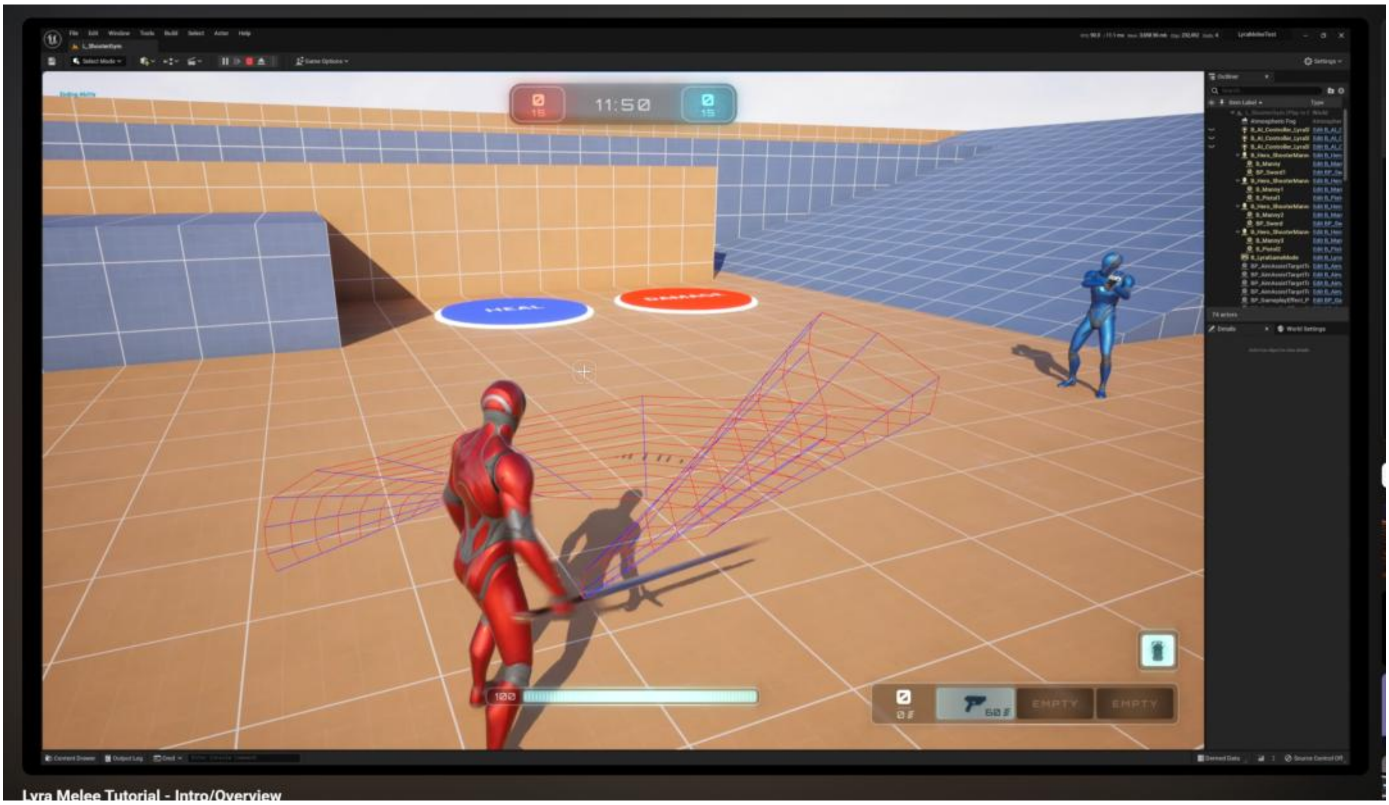
| | | | |
|---|---|---|---|
| Socket Name | End | | |
| Bone Name | Weapon_Sword | | |
| Relative Location | [illegible] | -0.0 | [illegible] |
| Relative Rotation | 0.0° | 0.0° | 0.0° |
| Relative Scale | 0.1 | 0.1 | 0.1 |
| Force Always Animated | ☑ | | |

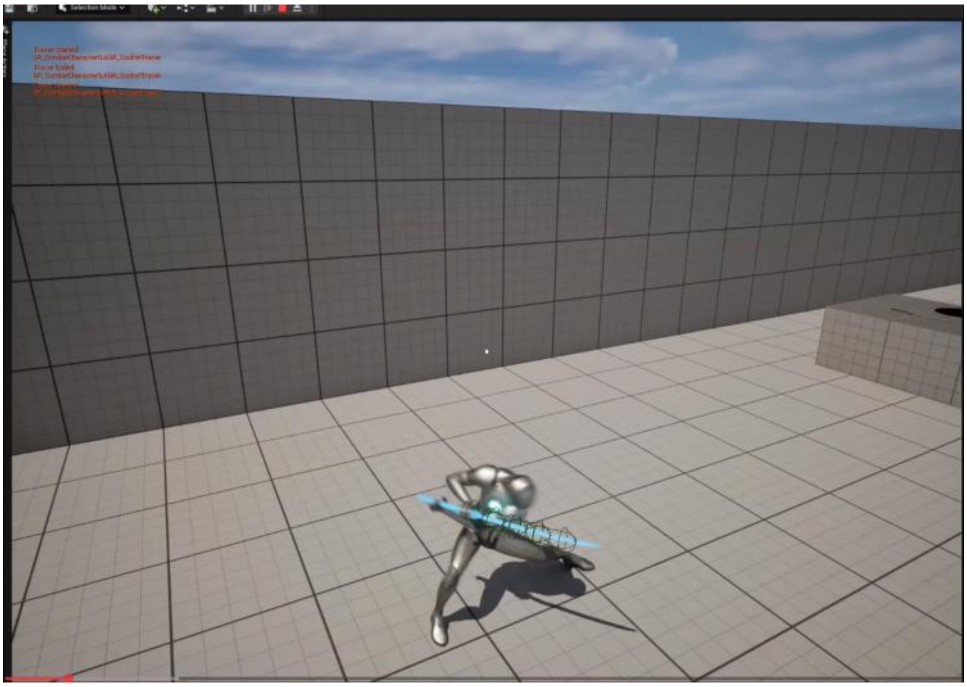
2:42 / 6:53

AGR V Melee Combat - Unreal 5 Advanced Tutorial

- Key Point: We define Start/End points on the weapon mesh.

3. **Multi-Trace & Debugging:**





- Key Point: Debug shapes should be visible, and hit points must be highlighted in red.

# Technical Requirements (Critical)

1. **Plugin Structure:**
   - Module Name: `AdvancedMeleeTrace`
   - Core Class: `UAnimNotifyState_MeleeTrace` (Inherits from `UAnimNotifyState`)

2. **Trace Logic (Most Important):**
   - Do NOT use simple `LineTraceSingleByChannel` at the current frame only. This causes tunneling issues with fast animations.
   - **Requirement:** You MUST implement a "Sweep" from the `Previous Frame Socket Location` to the `Current Frame Socket Location`.
   - Reference Logic:
     `SweepStart = SocketLocation_PrevFrame;`
     `SweepEnd = SocketLocation_CurrentFrame;`

3. **Trace Shapes:**
   - Support `Line`, `Box`, and `Sphere` shapes.
   - Parameters: Line Length, Box Extents (Width/Height), Sphere Radius.

4. **Hit Management:**
   - **Ignore Duplicates:** A single swing should damage a specific actor only once, even if the trace overlaps multiple times during the animation.
   - Maintain a `TArray<AActor*> HitActors` to track already hit targets during the NotifyState.

5. **Collision Settings:**
   - Support filtering by `GameplayTags` or `ECollisionChannel`.

6. **Event System:**
   - When a hit is detected, broadcast a Delegate or Interface call.
   - **Payload:** Must pass the full `FHitResult` struct (Impact Point, Normal, Bone Name, etc.).

7. **Network Replication:**
   - Strategy: Detect hits on the **Client (Autonomous Proxy)** to ensure responsiveness.
   - Then, send a Server RPC (`ServerValidateHit`) to verify and apply damage.

8. **Debugging:**

  - Use `DrawDebugLine`, `DrawDebugBox`, etc.
  - Color Code: Green for no hit, **Red** for hit detected.

# Output Request
1. Provide the **header file (.h)** and **cpp file (.cpp)** for the `UAnimNotifyState_MeleeTrace` class.
2. Specifically show the `NotifyTick` function where the `Prev -to-Curr` Sweep logic is implemented.
3. Show how to handle the `HitActors` array to prevent double hits on the same enemy.