PROFESSIONAL UE5 PLUGIN

Paired Animation System

Cinematic executions, takedowns, finishers and synchronized two-character interactions

Version 1.3.0 · UE 5.5 / 5.6 / 5.7 / 5.8 · by Erak Dev / Adrenaline Games

What's New in v1.3.0

1.3.0 is a major correctness pass. Every feature this manual documents is now actually wired to runtime. Previously several data asset fields existed in the editor but the runtime never read them — they are now either implemented or cleanly removed.

Headline changes

- **3 real camera modes** (FOV Only, Focus on Target, Fixed Angle) replace 5 previously documented but never-implemented modes
- **Effects pipeline runs**: StartSound, ImpactEffect (Niagara), AmbientSound, CameraShake are spawned at start, AmbientSound auto-stops on cleanup
- **Invulnerability tag**: `State.PairedAnimation.Invulnerable` applied via ASC
- **Buffs / debuffs / resource effects** applied through ASC, server-authoritative
- **Slow motion** with configurable start time, dilation factor, duration
- **Cooldown per animation** tracked per initiator
- **Rewards delegate** broadcasting XP and currency on completion (server only)
- **Validation with reason string** for clean debugging
- **Performance**: BT decorator and task use OverlapMulti instead of full-world actor iteration
- **Networking**: server validates distance and cooldown (anti-cheat), state set before multicast
- **Packaging**: single, clean plugin folder (no more duplicate nested structure)

IMPORTANT: **Migration matters.** The QTE system, Chain Animations, Counter system, Environment requirements, Camera Cuts timeline, Post-Process settings, Dialogue/Effort Sounds timelines, MovieScene Sequencer track, and 3 GAS target modes (LowestHealth, HighestHealth, HighestThreat) were

previously documented features that never worked in 1.2.0 and earlier. They are removed in 1.3.0. See Migration from 1.2.0 for replacement strategies.

Installation

Plugin works on UE 5.5, 5.6, 5.7 and 5.8.

1 Pick your engine version ZIP

- `PairedAnimationSystem_UE5_5.zip`
- `PairedAnimationSystem_UE5_6.zip`
- `PairedAnimationSystem_UE5_7.zip`
- `PairedAnimationSystem_UE5_8.zip`

2 Extract

Extract into one of:

- `YourProject/Plugins/` (per-project)
- `Engine/Plugins/Marketplace/` (engine-wide)

Result: `Plugins/PairedAnimationSystem/PairedAnimationSystem.uplugin`

3 Enable in editor

Edit → Plugins → Animation → Paired Animation System → Enabled. Restart editor when prompted.

4 Required engine plugins

These are referenced as dependencies in `.uplugin` — UE enables them automatically:

- GameplayAbilities
- EnhancedInput
- Niagara

Quick Start — Your First Paired Animation

10-minute setup. Pick two existing montages (an attacker animation and a victim animation that play well synced).

1 Add component to your characters

Open your Character Blueprint. Add Component → Paired Animation Component. The component must live on an `ACharacter` (not just `APawn`) with a SkeletalMesh and AnimInstance.

- Set `bCanInitiatePairedAnimations = true` for actors that attack
- Set `bCanReceivePairedAnimations = true` for actors that can be victims

2 Create a data asset

Content Browser → right-click → Miscellaneous → Data Asset → pick `PairedAnimationData`.

Fill in:

- **AnimationTag**: `Execution.Sword.Front` (your tag)
- **DisplayName**: human readable
- **ActorAnimations**: 2 entries (Initiator + Receiver) each with a montage
- **PositioningMode**: `FaceToFace` for most executions
- **ActorDistance**: the distance the animations were authored for (e.g. 80 cm)
- **MaxActivationDistance**: starting range (e.g. 200 cm)
- **RequiredFacingAngle**: how much initiator must face target (e.g. 45 deg)

3 Trigger it

```
// C++
PairedAnimComp->StartPairedAnimation(
    FGameplayTag::RequestGameplayTag("Execution.Sword.Front"),
    TargetActor);

// Or via library
UPairedAnimationLibrary::StartPairedAnimationByTag(MyActor,
TargetActor, MyTag);
```

In Blueprint: node **Start Paired Animation By Tag**.

TIP: If the animation does not fire, jump straight to Validation & Debugging to get a specific reason.

Animation Montages

Authoring montages for paired animations.

Both montages must align

The two characters' montages play at the same start time. Frame 0 of both should be the moment the actors are positioned correctly. Author them assuming the initiator faces the receiver at `ActorDistance` centimeters apart (the value you set on the data asset).

Root motion

Each `FPairedActorAnimData` entry has `bUseRootMotion`. If your animations use root motion, leave it on. If not, the system positions the actors statically at the start and they stay put unless your montage moves the mesh.

End-of-montage sync

Two settings handle the "capsule snaps back to wrong place" problem:

- `bSyncCapsuleToMeshOnEnd` (default true) — moves the capsule to where the mesh actually is when the montage finishes
- `bSyncRotationFromBone` + `SyncBoneName` (default `pelvis`) — reads the final yaw from a bone instead of the root

> **TIP:** For most character skeletons authored in Maya/Blender the root bone has no animation; pelvis or hips drives the actual transform. The default settings handle that.

Duration time-warping

`Duration` default is `0.0` (use the montage's native length). Set non-zero to time-warp: the component computes `PlayRate = MontageLength / Duration`.

Triggering Animations

Wire input or AI logic to start paired animations.

From an input action (Enhanced Input)

In your Character Blueprint, bind an Input Action (e.g. `IA_Execute`) and call:

```
UPairedAnimationLibrary::StartPairedAnimationByTag(
    this,
    GetTargetInRange(),
    FGameplayTag::RequestGameplayTag("Execution.Front"));
```

From AI Behavior Tree

See AI Behavior Tree. Drop `BTTask_ExecutePairedAnimation` into your tree, gate it with `BTDecorator_CanExecutePairedAnimation`.

From GAS ability

See GAS Integration. Use `GA_PairedAnimation` as your base class.

From Sequencer (Cinematic)

Use an Event track that calls a Blueprint function which calls `StartPairedAnimationBetween`. The dedicated MovieScene track was removed in 1.3.0 because it was a placeholder; the Event-track route is the recommended approach.

Finding a Target

Three methods, all using `OverlapMultiByChannel(ECC_Pawn)` under the hood (bounded cost, not full-world iteration).

Method 1: nearest valid target

```
AActor* Target =
UPairedAnimationLibrary::FindBestPairedAnimationTarget(
    /*Initiator*/ this,
    /*SearchRadius*/ 300.0f,
    /*RequiredTags*/ FGameplayTagContainer(),
    /*IgnoredActors*/ TArray<AActor*>());
```

Method 2: all candidates

```
TArray<AActor*> All =
UPairedAnimationLibrary::FindAllPairedAnimationTargets(
    this, 300.0f, RequiredTags, IgnoredActors);
```

Method 3: scored selection

The component's `FindBestPairedAnimation(Target, AnimList)` scores each candidate animation by distance and facing angle. Useful when an actor has many animations and you want the best fit for the current geometry.

Data Asset Reference — PairedAnimationData

Complete field list. Everything documented here is wired to runtime in 1.3.0.

Identification

Field	Type	Purpose
`AnimationTag`	FGameplayTag	Tag used by Subsystem lookup
`DisplayName`	FText	For UI
`Description`	FText	For UI

Animation

Field	Type	Purpose
`ActorAnimations`	TArray<FPairedActorAnimData>	Per-role montages (Initiator + Receiver)
`Duration`	float	`0` = native montage length. Non-zero = time-warp

Positioning

Field	Type	Purpose
`PositioningMode`	enum	FaceToFace / Behind / Side / Custom
`ActorDistance`	float	cm between actors at start
`bInstantSnap` / `SnapSpeed`	bool / float	Snap or interpolate to position
`bSwapActorRoles`	bool	Swap initiator/receiver positioning anchor
`bSyncCapsuleToMeshOnEnd`	bool	Move capsule to mesh end-position
`bSyncRotationFromBone` / `SyncBoneName`	bool / FName	Read final yaw from named bone

Requirements

Field	Type	Purpose
`RequiredInitiatorTags` / `RequiredReceiverTags`	FGameplayTagContainer	Hard gate for activation
`MaxActivationDistance`	float	cm
`RequiredFacingAngle`	float	degrees, 0-180
`CooldownSeconds` 1.3.0	float	Per-initiator cooldown for this animation

Effects

Field	Type	Purpose
`CameraShake` / `CameraShakeScale`	TSubclassOf / float	Played on initiator's PC at start
`ImpactEffect`	TSoftObjectPtr<UNiagaraSystem>	Niagara at midpoint between actors
`StartSound`	TSoftObjectPtr<USoundBase>	One-shot at midpoint
`AmbientSound`	TSoftObjectPtr<USoundBase>	Looping, attached to initiator mesh, auto-stopped
`AudioAttenuation`	USoundAttenuation*	For all spawned sounds

Gameplay

Field	Type	Purpose
`bMakeInvulnerable` 1.3.0	bool	Grants `State.PairedAnimation.Invulnerable` via ASC
`bDisableInput` 1.3.0	bool	Honored now (was ignored before)
`bCanBeInterrupted` 1.3.0	bool	If false, external StopPairedAnimation calls are rejected

Gameplay Effects (require GAS)

Field	Type	Purpose
`InitiatorBuffs` 1.3.0	TArray<TSubclassOf<UGameplayEffect>>	Applied to initiator via ASC on start
`ReceiverDebuffs` 1.3.0	TArray<TSubclassOf<UGameplayEffect>>	Applied to receiver via ASC on start
`ResourceDrainEffect` 1.3.0	TSubclassOf<UGameplayEffect>	Stamina cost etc. on start
`ResourceGainEffect` 1.3.0	TSubclassOf<UGameplayEffect>	Applied to initiator on natural completion

Rewards

Field	Type	Purpose
`XPReward` 1.3.0	int32	Broadcast via OnPairedAnimationRewards (server only)
`CurrencyReward` 1.3.0	int32	Broadcast via OnPairedAnimationRewards (server only)

Slow Motion

Field	Type	Purpose
`bUseSlowMotion` 1.3.0	bool	Enable slow motion scheduling
`SlowMotionSpeed` 1.3.0	float	0.05 - 1.0 (0.3 = 30% speed)
`SlowMotionStartTime` 1.3.0	float	Seconds after animation start
`SlowMotionDuration` 1.3.0	float	How long it lasts

Blueprint Library — UPairedAnimationLibrary

Static helper functions, all `BlueprintCallable` or `BlueprintPure`.

Function	Purpose
`StartPairedAnimationBetween`	Start using a specific DataAsset
`StartPairedAnimationByTag`	Start using a GameplayTag (subsystem resolves)
`StopPairedAnimationOn`	Stop on an actor
`IsInPairedAnimation` / `IsPlayingPairedAnimation`	Query state
`GetPairedAnimationPartner`	Get the other actor
`CanStartPairedAnimationBetween`	Bool check
`ValidateCanStartWithReason` 1.3.0	Bool + reason string
`FindBestPairedAnimationTarget`	OverlapMulti-based search
`FindAllPairedAnimationTargets`	All valid candidates
`AreActorsInRangeAndFacing`	Distance + angle check
`GetAngleToTarget`	Degrees, clamped Acos (NaN-safe)
`AddTagToActor` / `RemoveTagFromActor` / `ActorHasTag`	Component OwnedTags helpers

Validation & Debugging 1.3.0

"Why didn't my animation start" gets a concrete answer.

```
FString Reason;
bool bCanStart = UPairedAnimationLibrary::ValidateCanStartWithReason(
    Initiator, Target, MyAnimData, Reason);

if (!bCanStart)
{
    UE_LOG(LogTemp, Warning, TEXT("Cannot start: %s"), *Reason);
}
```

Possible reasons:

- `"AnimData is null"` / `"Target is null"` / `"Owner is not a Character"`
- `"Too far: 412 > MaxActivationDistance 200"`
- `"Bad angle: 92.5 deg > RequiredFacingAngle 45.0"`
- `"Initiator missing required tags: Combat.Sword"`
- `"Receiver missing required tags: Character.Enemy"`
- `"On cooldown for 2.4s more"`
- `"OK"` (passes)

> TIP: The same logic is exposed on the component as `ValidateCanStart(AnimData, Target, OutReason)`. Bind a Blueprint button to call this on selected pairs in PIE for instant diagnosis.

Cooldown System 1.3.0

Per-animation cooldown tracking on the initiator's component.

Set `CooldownSeconds` on the data asset. When the initiator triggers the animation, the component records `EndTime = WorldTime + CooldownSeconds` in `CooldownEndTimes[AnimationTag]`. Subsequent `CanStartPairedAnimation` calls check this map and reject early, writing the remaining seconds into the validation reason.

How this differs from GAS Cooldown

	Per-animation Cooldown (this plugin)	**GAS GameplayEffect Cooldown**
Granularity	One cooldown per animation tag	One cooldown per ability
Best for	Multiple animations in one ability with different timing	Whole ability gate (block all executions for X seconds)
Configured on	PairedAnimationData asset	GA_PairedAnimation subclass + Cooldown GE

You can use both at the same time. The GAS Cooldown blocks the ability from activating at all; the per-animation cooldown blocks specific data assets within an ability that handles several.

Rewards Hook 1.3.0

Plug XP and currency into your progression layer.

`OnPairedAnimationRewards` is a server-only multicast delegate. Signature:

```
void OnPairedAnimationRewards(
    AActor* Initiator,
    const UPairedAnimationData* AnimData,
    int32 XPReward,
    int32 CurrencyReward);
```

It fires:

- Only on the server
- Only on natural completion (not interruption)
- Only for the initiator (one broadcast per animation)
- Only when `XPReward > 0` or `CurrencyReward > 0`

C++ binding example

```
void AMyCharacter::BeginPlay()
{
    Super::BeginPlay();
    if (HasAuthority() && PairedAnimComp)
    {
        PairedAnimComp->OnPairedAnimationRewards.AddDynamic(
            this, &AMyCharacter::HandlePairedRewards);
    }
}

void AMyCharacter::HandlePairedRewards(AActor* Init,
    const UPairedAnimationData* Data, int32 XP, int32 Currency)
{
    // your game's XP/coin/unlock logic
}
```

> **NOTE:** For unlock-on-first-execution behavior (the removed 1.2.0 `bUnlockSpecialMoveOnFirst` field), keep a server-side set of "moves the player has done at least once" and check it inside this handler.

Slow Motion 1.3.0

Cinematic time dilation scheduled and reverted automatically.

Enable on the data asset:

- `bUseSlowMotion = true`
- `SlowMotionSpeed = 0.3` (30% speed)
- `SlowMotionStartTime = 0.5` (start 0.5s after animation begins)
- `SlowMotionDuration = 0.5` (lasts 0.5s of real time)

The component captures the current `GlobalTimeDilation` at start, schedules two timers to apply and revert the dilation, and falls back to the pre-animation value in `CleanupPairedAnimation` if the animation is interrupted — so an early stop never leaves the world stuck in slow motion.

> **IMPORTANT:** Slow motion is global per peer. In multiplayer this means the initiator's machine slows down; other clients keep normal time. This is intentional — cinematic slow motion in a shared world breaks the feel. If you need true synchronized slow motion across all clients, you'll need to multicast the dilation change yourself.

Invulnerability Tag 1.3.0

Built-in damage immunity during paired animations.

When `bMakeInvulnerable = true` on the data asset:

1. On start, the component grants `State.PairedAnimation.Invulnerable` to the actor via ASC (`AddLooseGameplayTag`)
2. If the actor has no ASC, the tag is written into the component's own `OwnedTags` as a fallback
3. On cleanup (natural or interrupted), the tag is removed

Using it in your damage pipeline

In your damage execution / GE application path, check for the tag and skip damage:

```
const FGameplayTag InvTag = FGameplayTag::RequestGameplayTag(
    "State.PairedAnimation.Invulnerable");

if (TargetASC && TargetASC->HasMatchingGameplayTag(InvTag))
{
    return; // skip damage
}
```

TIP: The same tag is granted to BOTH actors in a paired animation. If you want only one of them invulnerable (e.g. only the initiator), set `bMakeInvulnerable = false` and apply your own ASC tag from a Blueprint that listens to `OnPairedAnimationStarted`.

GAS Integration

Plug the system into Gameplay Ability System via `GA_PairedAnimation`.

Creating an ability

1. Content Browser → right-click → Blueprint Class → pick `GA_PairedAnimation`
2. Configure properties (table below)
3. Grant on character: `AbilitySystem→GiveAbility(FGameplayAbilitySpec(MyAbility, 1, 0));`
4. Activate by tag: `TryActivateAbilitiesByTag(AbilityTagsContainer)`

Target modes

Mode	Behavior
`Manual`	Reads `TriggerEventData→Target`. Use with `SendGameplayEvent`
`AutoNearest`	Closest valid target in radius
`AutoBestScore`	Scored by distance + facing angle
`LookingAt`	Best match within the look cone (`LookingAtConeAngle`)
`RandomValid`	Random pick from valid candidates
`Custom`	Calls `CustomTargetSelection(ValidTargets)` (BlueprintNativeEvent)

IMPORTANT: 1.3.0 removed three target modes that were stubs in 1.2.0: `LowestHealth`, `HighestHealth`, `HighestThreat`. They all fell through to `AutoNearest` with TODO comments. To replace them: use `Custom` mode and override `CustomTargetSelection` in Blueprint, reading your game's health/threat attributes from each candidate.

Manual target via GameplayEvent

```
FGameplayEventData EventData;
EventData.Target = ExplicitTargetActor;
EventData.EventTag =
FGameplayTag::RequestGameplayTag("Event.TriggerExecution");
AbilitySystem→HandleGameplayEvent(EventData.EventTag, &EventData);
```

Cost & cooldown

Standard GAS pattern. Set `CostGameplayEffectClass` and `CooldownGameplayEffectClass` on your ability subclass. The plugin honors these in `CommitAbility`.

Effects fields on the ability

The ability has its own effect fields separate from the data asset:

- `EffectsToApplyToInitiator` / `EffectsToApplyToReceiver`
- `EffectsOnCompletion`
- `TagsToRemoveFromInitiator` / `TagsToRemoveFromReceiver`
- `bEndAbilityOnComplete` / `bCancelAbilityOnInterrupt`
  Use ability effects for ability-wide concerns (e.g. ability tags, cost). Use data asset effects (`InitiatorBuffs` , `ReceiverDebuffs`) for animation-specific behavior.

Camera Control

3 working modes via `AnimNotifyState_PairedCamera`. Place the notify state on the initiator's montage, spanning the dramatic part of the animation.

Mode: FOV Only

Just smoothly changes FOV. Player keeps full camera control. Good for subtle dolly-zoom drama on big hits.

- `bOverrideFOV = true`
- `CustomFOV = 60.0` (default 60, narrower than typical 90)
- `FOVTransitionSpeed = 5.0`

Mode: Focus on Target

Smoothly aims the player camera at the paired animation partner (read from `PairedAnimationComponent::GetCurrentPartner()`). Tunable interp speed and pitch offset.

- `FocusInterpSpeed = 6.0` (higher = snappier)
- `FocusPitchOffset = -5.0` (slight downward look)

Mode: Fixed Angle

Snaps the view to a fixed orbit around the actor (yaw + pitch). Good for cinematic finishers where the camera needs to land on a specific shot.

- `FixedYawOffset = 90.0` (degrees from actor's forward; 0 = behind, 180 = front)
- `FixedPitchOffset = -10.0`
- `FixedAngleInterpSpeed = 6.0`

NOTE: All modes can stack the FOV override on top — set `bOverrideFOV = true` in any mode. Original FOV is captured in `NotifyBegin` and restored in `NotifyEnd`.

Audio & Effects

Sounds and particles spawned at animation start.

Three audio fields on the data asset

Field	When	Where
`StartSound`	Start (one-shot)	Midpoint between actors
`AmbientSound`	Looping for animation duration	Attached to initiator's mesh
Damage notify hit sound	At damage frame	Receiver location (set on the notify, not data asset)

Soft references

`StartSound`, `AmbientSound`, `ImpactEffect` are `TSoftObjectPtr`. The Subsystem calls `PreloadSoftReferences()` on registration (uses `StreamableManager::RequestAsyncLoad` — non-blocking), then the component calls `LoadSynchronous` at the moment of use as a safety net.

> TIP: If your sounds are large (e.g. high-quality cinematic dialogue), the async preload at registration time warms the cache so first play has no hitch.

Impact Niagara effect

`ImpactEffect` spawns at the midpoint between actors via `UNiagaraFunctionLibrary::SpawnSystemAtLocation`.

Camera shake

`CameraShake` + `CameraShakeScale` on the data asset is played on the initiator's player controller (if any) via `ClientStartCameraShake`. The

AnimNotify_PairedDamage notify has its own camera shake field (fires at the damage frame, not the animation start) — both can coexist.

Gameplay Effects (Buffs / Debuffs / Resource)

Server-authoritative ASC integration. Skipped if actor has no AbilitySystemComponent.

Field	Applied to	When
`InitiatorBuffs`	Initiator	On start (server only)
`ReceiverDebuffs`	Receiver	On start (server only)
`ResourceDrainEffect`	Initiator	On start (server only) — e.g. stamina cost
`ResourceGainEffect`	Initiator	On natural completion (server only) — e.g. adrenaline rush

The actual application uses:

```
UAbilitySystemComponent* ASC =
    UAbilitySystemBlueprintLibrary::GetAbilitySystemComponent(Actor);

FGameplayEffectContextHandle Context = ASC->MakeEffectContext();
FGameplayEffectSpecHandle Spec = ASC->MakeOutgoingSpec(Effect, 1.0f,
Context);
ASC->ApplyGameplayEffectSpecToSelf(*Spec.Data.Get());
```

IMPORTANT: All effect application happens only when `GetOwnerRole() == ROLE_Authority`. If your client tries to apply effects directly, nothing happens — everything routes through the server's authoritative state.

AI Behavior Tree

Drop-in BT nodes for AI-driven executions.

BTTask_ExecutePairedAnimation

Property	Purpose
`AnimationTag` or `AnimationData`	What to play
`TargetKey`	Blackboard key with target actor (optional)
`AutoTargetSearchDistance`	If no Blackboard target, OverlapMulti search radius
`RequiredTargetTags` / `ForbiddenTargetTags`	Filter
`bWaitForCompletion`	Block BT until animation ends
`bFailIfNoTarget`	Return Failed if no target found

BTDecorator_CanExecutePairedAnimation

Gate any BT branch on "can we do this paired animation right now". Checks component presence, not-already-in-animation, required tags, target in range.

> **TIP:** 1.3.0 swapped the underlying target search from `GetAllActorsOfClass` (full-world iteration on every BT tick — catastrophic with many AI) to `OverlapMultiByChannel(ECC_Pawn, SearchDistance)`. Cost is now bounded.

AnimNotifies

Frame-perfect damage and effects.

AnimNotify_PairedDamage

Place on the initiator's montage at the impact frame. Applies damage to the receiver. Reads the partner from `PairedAnimationComponent::GetCurrentPartner()` automatically.

Property	Purpose
`BaseDamage`	Damage to apply
`DamageType`	UDamageType subclass
`HitSound`	Sound at impact
`HitCameraShake`	Shake on initiator's PC
`HitEffectNiagara`	Particle at receiver location

AnimNotify_PairedEffect

Drop visual or audio cue at a specific animation frame. Independent of damage.

AnimNotifyState_PairedCamera

See Camera Control.

Events & Delegates

Bind to component events to react to animation lifecycle.

Delegate	Signature	When
`OnPairedAnimationStarted`	(Initiator, Receiver, Data)	On start (all peers)
`OnPairedAnimationFinished`	(Initiator, Receiver, Data)	On stop, natural or interrupted (all peers)
`OnStateChanged`	(OldState, NewState)	On every state transition
`OnPairedAnimationRewards` 1.3.0	(Initiator, Data, XP, Currency)	Natural completion, server only, initiator only

States (`EPairedAnimState`): `Idle` → `Positioning` → `Playing` → `Finishing` → `Idle` .

Multiplayer / Replication

Fully server-authoritative.

Flow

1. Client (or server) calls `StartPairedAnimation*`
2. Routes through `ServerStartPairedAnimation` RPC
3. Server `_Validate` does distance sanity check (anti-cheat — rejects requests where actors are impossibly far apart)
4. Server `_Implementation` re-validates fully (distance, angle, tags, cooldown)
5. Server sets state on both components, applies cooldown bookkeeping
6. `MulticastStartPairedAnimation` runs on all peers
7. Each peer positions, plays montage, applies start-time effects, broadcasts `OnPairedAnimationStarted`

Replicated properties

- `CurrentState`
- `CurrentPartner`
- `CurrentAnimationData`
- `bIsInitiator`

> NOTE: Multicasts in 1.3.0 set `bIsInitiator`, `CurrentAnimationData`, `CurrentPartner` on both components **before** broadcasting events — so reads in the same multicast frame see consistent state. Previously these relied on repnotify race timing.

Troubleshooting

"My animation doesn't start. No error."

- Use `UPairedAnimationLibrary::ValidateCanStartWithReason` — you'll get a specific reason
- Common: distance, angle, missing tag, cooldown

"Owner must be a Character with valid AnimInstance"

- Component requires `ACharacter`, not `APawn`
- Character needs a SkeletalMesh with assigned AnimInstance class

"Animation not found for tag"

- Check tag spelling in GameplayTagEditor
- Verify data asset's `AnimationTag` field is set
- Check Subsystem's `SearchPaths` — default is `/Game`
- Set `bShowDebugLogs = true` on Subsystem and check `PrintRegisteredAnimations()` output

"Capsule snaps back after animation ends"

- Verify `bSyncCapsuleToMeshOnEnd = true`
- Verify `SyncBoneName` matches your skeleton (default `pelvis`)
- If your skeleton uses `hips` or `spine_01` as the anchor, update `SyncBoneName` accordingly

"Buffs / debuffs don't apply"

- Target needs an AbilitySystemComponent
- Application only happens on server (`ROLE_Authority`)
- Check that the GE class is set in the data asset, not just the ability

"BT decorator is slow with many AI"

- 1.3.0 already replaced `GetAllActorsOfClass` with `OverlapMulti`
- If still slow, reduce decorator tick frequency or use observer aborts
- Lower `SearchDistance` on the decorator

"Animation desynced in multiplayer"

- Verify owning actor has `bReplicates = true`
- Component is replicated by default; if you derived a subclass, don't accidentally clear that
- Network role: client triggers go through ServerStart, server triggers go straight to multicast

"Slow motion never reverts"

- Only one paired animation per peer should be controlling time dilation at a time
- If you also use slow motion outside the plugin, the component restores its captured pre-animation dilation — not necessarily 1.0

Migration from 1.2.0

Most migration is "the field you set never did anything anyway, just delete it."

Camera modes

The 5-mode enum (Fixed Angle, Orbit Around, Focus on Initiator, Focus on Receiver, Dynamic Action) is gone — only Focus on Target and Fixed Angle were partial designs and Orbit / Dynamic / Focus on Initiator/Receiver never worked. Replace with one of: `FOVOnly`, `FocusOnTarget`, `FixedAngle`. The new `FixedAngle` with `FixedYawOffset = 180` approximates the old "Orbit" intent.

QTE / Chain / Counter / Environment / CameraCuts / PostProcess / Dialogue / EffortSounds

All removed. Were stub fields in 1.2.0. Delete any values you had set — they were never read.

GAS target modes

`LowestHealth`, `HighestHealth`, `HighestThreat` removed (were stubs). Switch to `Custom` mode and override `CustomTargetSelection` (BlueprintNativeEvent) to query your game's health/threat attributes.

MovieScene Sequencer track

Removed. The track was a placeholder with the comment *"this serves as a placeholder for future implementation"*. Replace with a Sequencer Event Track that calls a Blueprint function which calls `StartPairedAnimationBetween`.

bUnlockSpecialMoveOnFirst / MoveToUnlock

Removed. Use the new `OnPairedAnimationRewards` server delegate — keep a server-side set of "moves played at least once" and check it there.

AbilitiesToCooldown / ForcedCooldownDuration

Removed. Use a standard GAS Cooldown GameplayEffect on your `GA_PairedAnimation` subclass, or set `CooldownSeconds` on the data asset for per-animation cooldown.

EPairedAnimRole::Support

Removed (was unused). Only Initiator and Receiver remain.

Duration default

Was 2.0 in 1.2.0, now 0.0. `0` means "use montage's native length" which is what most users want. Set non-zero only when you want to time-warp.

Changelog

1.3.0 — 2026-06-29

Major correctness pass. Implementing what the manual previously claimed.

Fixed

- Plugin packaging: removed duplicate nested folder that caused UE to see two modules with the same name
- AnimNotifyState_PairedCamera: 3 real working camera modes replace 5 documented-but-unimplemented modes
- PairedAnimationLibrary::GetAngleToTarget: clamps DotProduct to [-1, 1] before Acos (NaN-safe)
- PairedAnimationSubsystem::Initialize: hooks `AssetRegistry::OnFilesLoaded` when registry still scanning (was returning empty list)
- BTDecorator and BTTask: replaced `GetAllActorsOfClass` with `OverlapMultiByChannel` (performance fix)
- ServerStartPairedAnimation_Validate: distance sanity check (anti-cheat)
- MulticastStartPairedAnimation: state set on both components before broadcasting events
- AnimNotifyState_PairedCamera::OriginalFOV: properly initialized to 90.0f
- `bDisableInput` / `bCanBeInterrupted` / `bMakeInvulnerable`: now actually honored at runtime
- TSoftObjectPtr assets (StartSound, AmbientSound, ImpactEffect): now preloaded via AssetManager and loaded at use time (were never loaded before)

Added

- Cooldown per animation (`UPairedAnimationData::CooldownSeconds`)
- Rewards delegate (`OnPairedAnimationRewards`) — server-only XP/currency hook
- Validation with reasons (`ValidateCanStart` / `ValidateCanStartWithReason`)
- Slow motion (`bUseSlowMotion` + speed / start / duration)
- AmbientSound playback attached to initiator, auto-stopped on cleanup
- StartSound played at midpoint on animation start
- ImpactEffect (Niagara) spawned at midpoint on start
- CameraShake from data asset played on initiator's PC at start

- InitiatorBuffs / ReceiverDebuffs / ResourceDrainEffect / ResourceGainEffect applied via ASC, server-authoritative
- `PreloadSoftReferences()` on data asset (called from Subsystem)
- GA Manual mode now reads `TriggerEventData→Target` (proper GAS pattern)

Removed

- **Ghost properties** from PairedAnimationData (fields without runtime support): QTEEvents, Chain props, Counter props, Environment requirements, CameraCuts, PostProcessSettings, Dialogue timelines, EffortSounds, StatusEffectsOnHit, AbilitiesToCooldown, bUnlockSpecialMoveOnFirst
- MovieScenePairedAnimationTrack + Section (placeholder, never functional)
- GA target modes: LowestHealth, HighestHealth, HighestThreat (all stubs returning AutoNearest)
- `EPairedAnimRole::Support` (was never read by runtime)
- `MovieScene` / `MovieSceneTracks` dependencies from Build.cs

Changed

- `Duration` default 2.0 → 0.0 (0 = native montage length)
- Build.cs: `IncludeOrderVersion.Latest`, added `PhysicsCore` private dep

1.2.0 — 2026-01-16

- GAS GameplayAbility (GA_PairedAnimation)
- Multiple target selection modes
- Tag-based requirements

1.1.0 — 2026-01-16

- Subsystem with auto-discovery and tag lookup
- AnimNotifyState_PairedCamera (FOV override only at this stage)
- Blueprint Function Library

1.0.0 — 2026-01-16

- Initial release
- PairedAnimationComponent, PairedAnimationData
- 4 positioning modes
- AnimNotify_PairedDamage, AnimNotify_PairedEffect

- Blueprint Library, multiplayer replication, GameplayTags, debug visualization

---

Paired Animation System v1.3.0

Copyright 2026 Erak Dev / Adrenaline Games. All Rights Reserved.

Support: support@adrenalinegames.pl · adrenalinegames.pl