FoxInput

Gesture Detection, Combos & Input Management for Blueprints
User Guide

Version 1.0
Alchemy Fox Studio

Supported Engine: UE 5.2·5.4, 5.6·5.7

Table of Contents

Installation

Option A: Via Unreal Engine Tab

1. Open the Epic Games Launcher and go to the Unreal Engine tab.
2. Search for FoxInput.
3. Click **Install to Engine**, select your Engine version, and click **Install**.
4. Open your project, go to **Edit > Plugins**, enable FoxInput, then restart the editor.

Option B: Via Fab Library

1. Open the Epic Games Launcher and go to **Fab**.
2. Open **My Library**, search for FoxInput and click on it.
3. Click **Install Plugin**, select your Engine version, and click **Install**.
4. Open your project, go to **Edit > Plugins**, enable FoxInput, then restart the editor.

Option C: Manual Installation

1. Copy the FoxInput/ folder into **YourProject/Plugins/**.
2. Right-click the .uproject file and select **Generate Visual Studio project files**.
3. Open the project, go to **Edit > Plugins**, enable FoxInput, then restart the editor.

All nodes appear under **FoxInput | <Category>** when you right-click in any Blueprint graph.

Supported Engine Version: **UE 5.2·5.4, 5.6·5.7**

Quick Start Tutorial

Every FoxInput node is a static Blueprint function — no setup, no initialization, no subsystem required (unless noted). Below are step-by-step examples showing how to use FoxInput in real project scenarios.

Example 1: Double-Tap to Dodge

Goal: Fire a dodge roll only when the player taps a direction key twice within 0.3 seconds.

Add the Detect Double Tap Node

From your IA_Move input event (or a dedicated IA_DodgeLeft), connect to Detect Double Tap with Key=W (or the relevant directional key), TimeWindow=0.3.

Handle Outputs

Wire Double Tap to your dodge roll logic. Wire Single Tap to normal movement. Wire Held to sprint. One node replaces all the timer boilerplate.

Prevent Spam

Combine with a FoxBranch Cooldown Gate (duration=0.6) on the Double Tap output so the player can't dodge faster than the animation allows.

Play and Test

Tap W twice quickly — dodge fires. Tap once — normal movement. Hold — sprint. All three behaviors from one Detect Double Tap node.

Example 2: Long Press for Charged Attack

Goal: Show a charge progress bar while the player holds a button, then fire a charged attack on release.

Wire Detect Long Press

From IA_Attack, connect to Detect Long Press with Duration=1.2 seconds.

Drive the Progress Bar

Each tick, the Progress output (0–1) updates. Feed it directly into your charge bar widget's SetPercent. No tick event or manual timer needed.

Fire on Completion

The Long Press Completed output fires once after 1.2 s. Connect it to your charged attack logic (bigger hit radius, more damage).

Handle Short Tap

For a quick tap (no long press), use a FoxBranch gate — if Progress < 0.1 when released, fire a normal attack instead. Or use Detect Hold or Tap for a single node that does both.

Example 3: Lock Input During Cutscene

Goal: Block all player input during a cutscene and automatically re-enable it when the scene ends.

Lock Input at Cutscene Start

When the cutscene begins, call Lock Input with your PlayerController reference and Tag="Cutscene". The tag lets multiple systems lock independently.

Disable UI Interactions Too

Also call Lock Input with Tag="CutsceneUI" if you want to block widget interactions. Multiple tags stack — all must unlock before input is restored.

Unlock When Done

When the cutscene ends (or the player skips), call Unlock Input with Tag="Cutscene". If "CutsceneUI" is still locked, input remains blocked until that is unlocked too.

Play and Test

Trigger the cutscene — movement and actions are blocked. When it ends, control returns seamlessly. Check Is Input Locked in your HUD to hide/show the action prompts.

Node Reference

Complete reference for all 28 FoxInput nodes organized by category. Each node includes a description, inputs, outputs, and a use case hint.

1. Gesture (6 Nodes)

6 Nodes

1.1 Detect Double Tap

Provides 1.1 Detect Double Tap functionality.

Input	Type	Description
WorldContext	Object Reference	
PlayerController	APlayerController*	
Key	FKey	Unique key name
TimeWindow	Float	Time in seconds

Output	Type	Description
Return Value	UFoxDetectDoubleTapAction*	

Use Case: Gesture detector — detects an input pattern as a latent async action.

1.2 Detect Hold Or Tap

Provides 1.2 Detect Hold Or Tap functionality.

Input	Type	Description
WorldContext	Object Reference	
PlayerController	APlayerController*	
Key	FKey	Unique key name
HoldThreshold	Float	

Output	Type	Description
Return Value	UFoxDetectHoldOrTapAction*	

Use Case: Gesture detector — detects an input pattern as a latent async action.

1.3 Detect Long Press

Provides 1.3 Detect Long Press functionality.

Input	Type	Description
WorldContext	Object Reference	
PlayerController	APlayerController*	
Key	FKey	Unique key name
Duration	Float	Duration in seconds

Output	Type	Description
Return Value	UFoxDetectLongPressAction*	

Use Case: Gesture detector — detects an input pattern as a latent async action.

1.4 Detect Multi Tap

Provides 1.4 Detect Multi Tap functionality.

Input	Type	Description
WorldContext	Object Reference	
PlayerController	APlayerController*	
Key	FKey	Unique key name
RequiredTaps	Int	
TimeWindow	Float	Time in seconds

Output	Type	Description
Return Value	UFoxDetectMultiTapAction*	

Use Case: Gesture detector — detects an input pattern as a latent async action.

1.5 Detect Tap Pattern

Provides 1.5 Detect Tap Pattern functionality.

Input	Type	Description
WorldContext	Object Reference	
PlayerController	APlayerController*	
Key	FKey	Unique key name
Pattern	Array of Float	

Output	Type	Description
Return Value	UFoxDetectTapPatternAction*	

Use Case: Gesture detector — detects an input pattern as a latent async action.

1.6 Detect Directional Input

Provides 1.6 Detect Directional Input functionality.

Input	Type	Description
AxisX	Float	
AxisY	Float	
bEightDirections	Bool	Boolean flag

Output	Type	Description
Return Value	EFoxInputDirection	

Use Case: Gesture detector — detects an input pattern as a latent async action.

2. Combo (1 Nodes)

1 Nodes

2.1 Input Sequence Detector

Provides 2.1 Input Sequence Detector functionality.

Input	Type	Description
WorldContext	Object Reference	
PlayerController	APlayerController*	
Sequence	Array of FKey	
TimeWindow	Float	Time in seconds
bOrderMatters	Bool	Boolean flag

Output	Type	Description
Return Value	UFoxInputSequenceDetectorAction*	

Use Case: Combo detector — detects a full input sequence with timing window.

3. Buffer (7 Nodes)

7 Nodes

3.1 Flush Input Buffer

Provides 3.1 Flush Input Buffer functionality.

Input	Type	Description
Buffer	UFoxInputBufferComponent*	

(none)

Use Case: Input buffer — record/query the last-N inputs for combo recognition.

3.2 Get Buffered Input

Provides 3.2 Get Buffered Input functionality.

Input	Type	Description
Buffer	UFoxInputBufferComponent*	

Output	Type	Description
Return Value	Bool	
Action	Name	

Use Case: Input buffer — record/query the last-N inputs for combo recognition.

3.3 Input Buffer Size
Provides 3.3 Input Buffer Size functionality.

Input	Type	Description
Buffer	UFoxInputBufferComponent*	

Output	Type	Description
Return Value	Int	

Use Case: Input buffer — record/query the last-N inputs for combo recognition.

3.4 Add To Buffer
Provides 3.4 Add To Buffer functionality.

Input	Type	Description
Action	Name	

(none)

Use Case: Input buffer — record/query the last-N inputs for combo recognition.

3.5 Flush Buffer
Provides 3.5 Flush Buffer functionality.

(none)

(none)

Use Case: Input buffer — record/query the last-N inputs for combo recognition.

3.6 Get Next Buffered Input
Provides 3.6 Get Next Buffered Input functionality.

(none)

Output	Type	Description
Return Value	Bool	
OutAction	Name	

Use Case: Input buffer — record/query the last-N inputs for combo recognition.

3.7 Get Buffer Size
Provides 3.7 Get Buffer Size functionality.

(none)

Output	Type	Description
Return Value	Int	

Use Case: Input buffer — record/query the last-N inputs for combo recognition.

4. State (5 Nodes)

5 Nodes

4.1 Lock Input For Duration

Provides 4.1 Lock Input For Duration functionality.

Input	Type	Description
WorldContext	Object Reference	
PlayerController	APlayerController*	
Duration	Float	Duration in seconds
Tag	Name	

Output	Type	Description
Return Value	UFoxLockInputForDuration Action*	

Use Case: State control — read/write current lock or runtime state.

4.2 Lock Input

Provides 4.2 Lock Input functionality.

Input	Type	Description
PlayerController	APlayerController*	
Tag	Name	

(none)

Use Case: State control — read/write current lock or runtime state.

4.3 Unlock Input

Provides 4.3 Unlock Input functionality.

Input	Type	Description
PlayerController	APlayerController*	
Tag	Name	

(none)

Use Case: State control — read/write current lock or runtime state.

4.4 Is Input Locked
Provides 4.4 Is Input Locked functionality.

Input	Type	Description
PlayerController	APlayerController*	

Output	Type	Description
Return Value	Bool	

Use Case: State control — read/write current lock or runtime state.

4.5 Get Key Hold Duration
Provides 4.5 Get Key Hold Duration functionality.

Input	Type	Description
PlayerController	APlayerController*	
Key	FKey	Unique key name

Output	Type	Description
Return Value	Float	

Use Case: State control — read/write current lock or runtime state.

5. Context (5 Nodes)

5 Nodes

5.1 Push Input Context
Provides 5.1 Push Input Context functionality.

Input	Type	Description
PlayerController	APlayerController*	
MappingContext	UInputMappingContext*	
Priority	Int	

(none)

Use Case: Input context — push/pop input contexts (gameplay / UI / cutscene).

5.2 Pop Input Context

Provides 5.2 Pop Input Context functionality.

Input	Type	Description
PlayerController	APlayerController*	

Output	Type	Description
RemovedContext	UInputMappingContext*	

Use Case: Input context — push/pop input contexts (gameplay / UI / cutscene).

5.3 Switch Input Context

Provides 5.3 Switch Input Context functionality.

Input	Type	Description
PlayerController	APlayerController*	
OldContext	UInputMappingContext*	
NewContext	UInputMappingContext*	
Priority	Int	

(none)

Use Case: Input context — push/pop input contexts (gameplay / UI / cutscene).

5.4 Get Active Input Contexts

Provides 5.4 Get Active Input Contexts functionality.

Input	Type	Description
PlayerController	APlayerController*	

Output	Type	Description
Return Value	Array of UInputMappingContext*	

Use Case: Input context — push/pop input contexts (gameplay / UI / cutscene).

5.5 Has Input Context

Provides 5.5 Has Input Context functionality.

Input	Type	Description
PlayerController	APlayerController*	
Context	UInputMappingContext*	

Output	Type	Description
Return Value	Bool	

Use Case: Input context — push/pop input contexts (gameplay / UI / cutscene).

6. Helpers (4 Nodes)

4 Nodes

6.1 Wait For Any Key

Provides 6.1 Wait For Any Key functionality.

Input	Type	Description
WorldContext	Object Reference	
PlayerController	APlayerController*	

Output	Type	Description
Return Value	UFoxWaitForAnyKeyAction *	

Use Case: Helper — input/device query or shared utility.

6.2 Get Last Pressed Key

Provides 6.2 Get Last Pressed Key functionality.

(none)

Output	Type	Description
Return Value	FKey	
TimeSincePress	Float	Time in seconds

Use Case: Helper — input/device query or shared utility.

6.3 Is Gamepad Connected

Provides 6.3 Is Gamepad Connected functionality.

(none)

Output	Type	Description
Return Value	Bool	

Use Case: Helper — input/device query or shared utility.

6.4 Get Input Device Type

Provides 6.4 Get Input Device Type functionality.

Input	Type	Description
PlayerController	APlayerController*	

Output	Type	Description
Return Value	EFoxInputDeviceType	

Use Case: Helper — input/device query or shared utility.

Implementation Notes

- Gesture detection nodes use FTimerHandle internally for timeouts.
- Input Buffer is a UActorComponent for per-actor state (max buffer size configurable, default 8).
- Input Lock uses static TMap> for tag-based multi-lock system.
- Input Context stack stored in static TMap>.
- Enhanced Input context functions gracefully fall back to no-op on Legacy Input.