FoxBranch

Flow Control, Gates & Branching Nodes for Blueprints

User Guide

Version 1.0
Alchemy Fox Studio

Supported Engine: UE 5.4, 5.6, 5.7, 5.8

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 FoxBranch.
3. Click **Install to Engine**, select your Engine version, and click **Install**.
4. Open your project, go to **Edit > Plugins**, enable FoxBranch, 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 FoxBranch and click on it.
3. Click **Install Plugin**, select your Engine version, and click **Install**.
4. Open your project, go to **Edit > Plugins**, enable FoxBranch, then restart the editor.

Option C: Manual Installation

1. Copy the FoxBranch/ 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 FoxBranch, then restart the editor.

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

Supported Engine Version: **UE 5.4, 5.6, 5.7, 5.8**

Quick Start Tutorial

Every FoxBranch 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 FoxBranch in real project scenarios.

Example 1: Cooldown Gate for an Ability System

Goal: Prevent players from spamming a dash ability — enforce a 1.5-second cooldown per press.

Place the Cooldown Gate

From your IA_Dash input event, connect Exec to Cooldown Gate. Set CooldownDuration to 1.5.

Handle Both Outputs

Wire Passed to your dash logic (launch character forward). Wire On Cooldown to a "play reject sound" node. Optionally display RemainingTime in your HUD.

No Cleanup Needed

State is per-node-instance — each Cooldown Gate in each Blueprint tracks its own timer independently. No timers to manage, no variables to declare.

Play and Test

Press dash rapidly — the first press always goes through. Subsequent presses within 1.5 s fire the "On Cooldown" branch with the remaining time available on the pin.

Example 2: Round Robin Patrol Waypoints

Goal: Cycle an NPC through 4 waypoints in sequence, wrapping back to the start.

Set Up Waypoints

Create a variable Waypoints (TArray) in your AI Controller with 4 locations. Store waypoint count as 4.

Add Round Robin

When MoveCompleted fires, call Round Robin with OutputCount=4. Each output (Exec1–Exec4) corresponds to one waypoint.

Move to Waypoint

From each output exec, call MoveToLocation with the matching Waypoints[0]..Waypoints[3]. CurrentIndex tells you which output just fired if you prefer a single path with a Get node.

Play and Test

The NPC walks 1→2→3→4→1 indefinitely. Connect the Reset pin to a "patrol interrupted" event to restart from waypoint 1 on alert.

Example 3: Debounce for Search Input

Goal: Only trigger an expensive search query after the player has stopped typing for 0.4 seconds.

Wire the Text Input

In your search widget's OnTextChanged event, connect Exec to Debounce Gate with DebounceTime=0.4.

Trigger Search on Fired

Wire the Fired output to your actual search function. The gate will discard all rapid keystrokes and only fire once typing has paused.

No Timer Management

The Debounce Gate resets its own internal timer on each new call. You never interact with FTimerHandle directly.

Play and Test

Type quickly in the search box — only one query fires after you stop. This prevents the server from being hit on every keystroke.

Node Reference

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

1. Gates (7 Nodes)

1.1 Cooldown Gate

Provides 1.1 Cooldown Gate functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
CooldownDuration	Float	Time in seconds

Output	Type	Description
Result	EFoxCooldownResult	
RemainingTime	Float	Time in seconds

Use Case: Gate node — controls execution flow based on time, count or chance.

1.2 Debounce Gate

Provides 1.2 Debounce Gate functionality.

Input	Type	Description
WorldContextObject	Object Reference	
DebounceTime	Float	Time in seconds
LatentInfo	FLatentActionInfo	

(none)

Use Case: Gate node — controls execution flow based on time, count or chance.

1.3 Throttle Gate

Provides 1.3 Throttle Gate functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
ThrottleInterval	Float	

Output	Type	Description
Result	EFoxThrottleResult	

Use Case: Gate node — controls execution flow based on time, count or chance.

1.4 Count Gate

Provides 1.4 Count Gate functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
Input	EFoxGateInput	
RequiredCount	Int	Item count

Output	Type	Description
Result	EFoxCountGateResult	
CurrentCount	Int	Item count

Use Case: Gate node — controls execution flow based on time, count or chance.

1.5 Every Nth Gate

Provides 1.5 Every Nth Gate functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
N	Int	

Output	Type	Description
Result	EFoxEveryNthResult	
CallCount	Int	Item count

Use Case: Gate node — controls execution flow based on time, count or chance.

1.6 Chance Gate

Provides 1.6 Chance Gate functionality.

Input	Type	Description
Chance	Float	

Output	Type	Description
Result	EFoxChanceResult	

Use Case: Gate node — controls execution flow based on time, count or chance.

1.7 Threshold Gate

Provides 1.7 Threshold Gate functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
Value	Float	The computed value
Threshold	Float	
bUpward	Bool	Boolean flag

Output	Type	Description
Result	EFoxThresholdResult	

Use Case: Gate node — controls execution flow based on time, count or chance.

2. Branch (6 Nodes)

2.1 Multi Branch

Provides 2.1 Multi Branch functionality.

Input	Type	Description
Condition1	Bool	
Condition2	Bool	
Condition3	Bool	
Condition4	Bool	
Condition5	Bool	
Condition6	Bool	
Condition7	Bool	
Condition8	Bool	

Output	Type	Description
Result	EFoxMultiBranchResult	

Use Case: Branch node — pick one output based on input value or class.

2.2 Multi Branch All

Provides 2.2 Multi Branch All functionality.

Input	Type	Description
Condition1	Bool	
Condition2	Bool	
Condition3	Bool	
Condition4	Bool	
Condition5	Bool	
Condition6	Bool	
Condition7	Bool	
Condition8	Bool	

Output	Type	Description
Return Value	Int	
bAnyTrue	Bool	Boolean flag

Use Case: Branch node — pick one output based on input value or class.

2.3 Priority Select Int

Provides 2.3 Priority Select Int functionality.

Input	Type	Description
C1	Bool	
V1	Int	
C2	Bool	
V2	Int	
C3	Bool	
V3	Int	
C4	Bool	
V4	Int	
DefaultValue	Int	

Output	Type	Description
Return Value	Int	
SelectedIndex	Int	Zero-based index

Use Case: Branch node — pick one output based on input value or class.

2.4 Branch On Range Int

Provides 2.4 Branch On Range Int functionality.

Input	Type	Description
Value	Int	The computed value
R1Min	Int	
R1Max	Int	
R2Min	Int	
R2Max	Int	
R3Min	Int	
R3Max	Int	
R4Min	Int	
R4Max	Int	

Output	Type	Description
Result	EFoxRange4Result	

Use Case: Branch node — pick one output based on input value or class.

2.5 Branch On Range Float

Provides 2.5 Branch On Range Float functionality.

Input	Type	Description
Value	Float	The computed value
R1Min	Float	
R1Max	Float	
R2Min	Float	
R2Max	Float	
R3Min	Float	
R3Max	Float	
R4Min	Float	
R4Max	Float	

Output	Type	Description
Result	EFoxRange4Result	

Use Case: Branch node — pick one output based on input value or class.

2.6 Branch On Class

Provides 2.6 Branch On Class functionality.

Input	Type	Description
Object	Object Reference	
Class1	Class Reference	
Class2	Class Reference	
Class3	Class Reference	
Class4	Class Reference	

Output	Type	Description
Result	EFoxClass4Result	
CastedObject	Object Reference	

Use Case: Branch node — pick one output based on input value or class.

3. Flow (5 Nodes)

3.1 Do Every N

Provides 3.1 Do Every N functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
Input	EFoxGateInput	
N	Int	

Output	Type	Description
Result	EFoxFlowResult	
CallCount	Int	Item count

Use Case: Flow node — manage execution patterns (per-key once, periodic, latent pulse).

3.2 Do Once Per Key

Provides 3.2 Do Once Per Key functionality.

Input	Type	Description
WorldContextObject	Object Reference	
Input	EFoxDoOnceKeyInput	
Key	Name	Unique key name

Output	Type	Description
Result	EFoxDoOnceResult	

Use Case: Flow node — manage execution patterns (per-key once, periodic, latent pulse).

3.3 Pulse

Provides 3.3 Pulse functionality.

(none)

Output	Type	Description
bPulse	Bool	Boolean flag

Use Case: Flow node — manage execution patterns (per-key once, periodic, latent pulse).

3.4 Accumulator Gate

Provides 3.4 Accumulator Gate functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
AddValue	Float	
Threshold	Float	
bAutoReset	Bool	Boolean flag

Output	Type	Description
Result	EFoxAccumResult	
CurrentSum	Float	

Use Case: Flow node — manage execution patterns (per-key once, periodic, latent pulse).

3.5 Batch Gate

Provides 3.5 Batch Gate functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
Input	EFoxGateInput	
RequiredCalls	Int	

Output	Type	Description
Result	EFoxBatchResult	

Use Case: Flow node — manage execution patterns (per-key once, periodic, latent pulse).

4. Edge (6 Nodes)

4.1 Rising Edge

Provides 4.1 Rising Edge functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
Value	Bool	The computed value

Output	Type	Description
Result	EFoxEdgeResult	

Use Case: Edge detector — fires only when state actually changes.

4.2 Falling Edge

Provides 4.2 Falling Edge functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
Value	Bool	The computed value

Output	Type	Description
Result	EFoxEdgeResult	

Use Case: Edge detector — fires only when state actually changes.

4.3 Value Changed Bool

Provides 4.3 Value Changed Bool functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
Value	Bool	The computed value

Output	Type	Description
Result	EFoxEdgeResult	
OldValue	Bool	
NewValue	Bool	

Use Case: Edge detector — fires only when state actually changes.

4.4 Value Changed Int

Provides 4.4 Value Changed Int functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
Value	Int	The computed value

Output	Type	Description
Result	EFoxEdgeResult	
OldValue	Int	
NewValue	Int	

Use Case: Edge detector — fires only when state actually changes.

4.5 Value Changed Float

Provides 4.5 Value Changed Float functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
Value	Float	The computed value

Output	Type	Description
Result	EFoxEdgeResult	
OldValue	Float	
NewValue	Float	

Use Case: Edge detector — fires only when state actually changes.

4.6 Value Changed String

Provides 4.6 Value Changed String functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
Value	String	The computed value

Output	Type	Description
Result	EFoxEdgeResult	
OldValue	String	
NewValue	String	

Use Case: Edge detector — fires only when state actually changes.

5. Routing (4 Nodes)

5.1 Round Robin

Provides 5.1 Round Robin functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
Input	EFoxGateInput	
OutputCount	Int	Item count

Output	Type	Description
CurrentIndex	Int	Zero-based index

Use Case: Router — pick one of N execs by random, weighted or rotation.

5.2 Weighted Random Exec

Provides 5.2 Weighted Random Exec functionality.

Input	Type	Description
Weight1	Float	
Weight2	Float	
Weight3	Float	
Weight4	Float	

Output	Type	Description
Result	EFoxRange4Result	
SelectedIndex	Int	Zero-based index

Use Case: Router — pick one of N execs by random, weighted or rotation.

5.3 Random Exec

Provides 5.3 Random Exec functionality.

Input	Type	Description
OutputCount	Int	Item count

Output	Type	Description
Return Value	Int	

Use Case: Router — pick one of N execs by random, weighted or rotation.

5.4 Ping Pong Exec

Provides 5.4 Ping Pong Exec functionality.

Input	Type	Description
WorldContextObject	Object Reference	
NodeKey	Name	Unique key name
Input	EFoxGateInput	
OutputCount	Int	Item count

Output	Type	Description
CurrentIndex	Int	Zero-based index

Use Case: Router — pick one of N execs by random, weighted or rotation.

Implementation Notes

- ☐ All stateful nodes use `FPendingLatentAction` or internal `TMap` keyed by node UUID for per-instance state.
- ☐ Wildcard pins implemented via `CustomThunk` UFUNCTION meta or templated helper functions.
- ☐ No dependencies on other FoxToolkit plugins.
- ☐ All nodes under `FoxBranch|` category prefix in Blueprint right-click menu.
- ☐ `DevelopmentOnly` flag: None (all nodes are runtime-safe).