FoxUtils

100 Essential Blueprint Utility Nodes for Unreal Engine

User Guide

Version 1.1.0

Supported Engine: UE 5.4, 5.6–5.7

Alchemy Fox Studio

Table of Contents

■ Installation

■ Quick Start Tutorial

■ Category Reference

1. String (10 Nodes)
2. Array (10 Nodes)
3. Math (8 Nodes)
4. Actor / World (14 Nodes)
5. Debug (6 Nodes)
6. Platform (10 Nodes)
7. Collision / Trace (8 Nodes)
8. Delayed Loops (6 Nodes)
9. Instanced Mesh (HISM/FISM) (6 Nodes)
10. Camera (6 Nodes)
11. Audio (5 Nodes)
12. Save / Load (6 Nodes)
13. Inventory Helpers (5 Nodes)

■ FoxToolkit Ecosystem

■ Support

Installation

Option A: Via Unreal Engine Tab

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

Option C: Manual Installation

1. Copy the **FoxUtils/** 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 **FoxUtils**, then restart the editor

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

Supported Engine Version: UE 5.4, 5.6–5.7

Quick Start Tutorial

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

Example 1: Highlight the Closest Enemy

Goal: When the player presses a key, find the closest enemy within 2000 units and print its name on screen.

Step 1 — Open Your Character Blueprint

Open your player character Blueprint (e.g. BP_PlayerCharacter). In the Event Graph, add an Input Action event (e.g. "IA_Highlight") so the logic fires on key press.

Step 2 — Get Closest Actor of Class

Right-click and search for Get Closest Actor of Class (under FoxUtils | Actor). Connect your input event to this node. Set Actor Class to your enemy class, Location to Self → GetActorLocation, Max Radius to 2000.0.

Step 3 — Null-Check and Print

Use an **IsValid** node on the Return Value. If valid, call GetDisplayName, then connect to a FoxPrint node with a highlight color and 3.0 second duration.

Step 4 — Play and Test

Press Play, then press your bound key. The closest enemy's name appears on screen. Total setup: 4 nodes, no C++ required.

Example 2: Delayed Loop for Wave Spawning

Goal: Spawn 10 enemies one at a time, with a 0.5-second delay between each spawn.

Step 1 — Add a For Loop with Delay

In your spawner Blueprint, search for Fox For Loop with Delay. Set First Index = 0, Last Index = 9, Delay Per Step = 0.5.

Step 2 — Connect Spawn Logic

From the "Loop Body" pin, call Spawn Actor Simple. Use the Index output to offset spawn location (e.g. multiply Index × 200 and add to a base position).

Step 3 — Handle Completion

Connect the "Completed" pin to a FoxPrint node with message "Wave Complete!". This creates a dramatic wave spawn with just 3 FoxUtils nodes.

Example 3: Quick Save/Load with INI Files

Goal: Save and load the player's high score without creating a SaveGame object.

To save: search Save Int to INI. Set Section = "Player", Key = "HighScore", Value = your score variable. To load: search Load Int from INI with the same Section and Key, Default = 0. The value is stored in Saved/Config/FoxUtils.ini and persists between sessions. Two nodes, no boilerplate.

Example 4: On-Screen Indicator for Actors

Goal: Only show a health bar widget above enemies that are currently visible on the player's screen.

Step 1 — Get All Enemies

In your HUD or Widget Blueprint, use Get All Actors of Class to get all BP_Enemy actors.

Step 2 — Check Visibility

For each enemy, call Is Actor On Screen. Pass the enemy actor and the Player Controller.

Step 3 — Toggle the Widget

Branch on the result: if true, set the health bar to Visible. If false, set to Hidden. Much simpler than manual dot-product checks.

Example 5: Stealth Detection with Cone Check

Goal: An AI guard detects the player only if they are within a 60-degree forward cone and 1500 units range.

Step 1 — Set Up Detection

In the guard's Blueprint, call Cone Check for Actors. Set Origin to the guard's location, Direction to forward vector, Range to 1500, Half Angle to 30 (= 60° total), Filter Class to your player class.

Step 2 — React to Detection

If the returned array is not empty, the player is detected — trigger alert state. Combine with Visibility Check to also require line of sight.

Example 6: Performance Profiling with Perf Timer

Goal: Measure how long an expensive operation takes in your Blueprint.

Call Perf Timer Start with Label = "MyOperation" before the expensive code. After it completes, call Perf Timer Stop with the same label. The elapsed time prints on screen (e.g. "MyOperation: 2.34 ms"). Use to identify bottlenecks without leaving the editor.

Example 7: Inventory Grid with Find Best Fit Slot

Goal: Implement Diablo-style grid inventory where items have different sizes (e.g. 2×3 sword, 1×1 potion).

Maintain a boolean array for occupied cells (e.g. 10×6 grid = 60 booleans). When picking up an item, call Find Best Fit Slot with the grid array, dimensions, and item size. It returns the top-left cell (X, Y) where the item fits, or (-1, -1) if no space. Mark cells as occupied and place the item widget at that position.

Category Reference

Complete reference for all 100 FoxUtils nodes organized by category. Each node includes a description, inputs, outputs, and a practical tip.

1. String

10 Nodes

Truncate String

Cuts a string to a maximum length and appends a suffix when truncated. Useful for UI labels, chat messages, or any text that must fit a fixed space.

Input	Type	Description
Source	FString	The source string to truncate
Max Length	int32	Maximum character count before cutting
Suffix	FString	Appended when truncated (default: "...")

Output	Type	Description
Return Value	FString	Truncated string with suffix if needed

🟩 *Ideal for HUD name labels, item tooltips, or chat bubbles. Combine with Format Number for stat displays.*

Contains (Ignore Case)

Checks if a string contains a substring, ignoring upper/lower case differences. No more manual ToLower chains.

Input	Type	Description
Source	FString	The string to search in
Sub String	FString	What to search for

Output	Type	Description
Return Value	bool	True if substring is found (case-insensitive)

🟩 *Great for search/filter fields in inventory UIs or console command parsing.*

To Title Case

Capitalizes the first letter of every word. Turns "hello world" into "Hello World".

Input	Type	Description
Source	FString	Input string to convert

Output	Type	Description
Return Value	FString	Title-cased string

🟩 *Perfect for displaying player names, item names, or dynamically generated quest titles.*

Is Empty or Whitespace

Returns true if the string is empty or contains only spaces, tabs, or newlines. Saves you from chaining IsEmpty + TrimStart + IsEmpty.

Input	Type	Description
Source	FString	String to check

Output	Type	Description
Return Value	bool	True if empty or whitespace-only

🟩 *Use before saving player input to avoid blank names, empty chat messages, or whitespace-only search queries.*

Format Number

Inserts thousands separators into a number for readable display. 1234567 becomes "1,234,567". Supports large numbers via int64.

Input	Type	Description
Number	int64	Number to format
Separator	FString	Character between groups (default ",")

Output	Type	Description
Return Value	FString	Formatted number string

🟩 *Essential for score displays, currency, damage numbers. Combine with Truncate String for compact stat HUDs.*

Format Time Seconds

Converts a float of total seconds into a human-readable time string. Supports both clock format (2:05) and letter format (02m 05s), with optional milliseconds.

Input	Type	Description
Total Seconds	float	Total seconds to format
Use Letters	bool	True = letter format, False = clock format
Show Milliseconds	bool	Append milliseconds to output

Output	Type	Description
Return Value	FString	Formatted time string

🟩 *Use for speedrun timers, cooldown displays, play-time counters, or session length tracking.*

Remove Whitespace

Strips all whitespace characters (leading, trailing, and inner spaces, tabs, newlines) from a string.

Input	Type	Description
Source	FString	Input string

Output	Type	Description
Return Value	FString	String with no whitespace

🟩 *Useful for sanitizing codes, IDs, or serial numbers entered by players.*

Pad String

Pads a string to a minimum length by prepending or appending a fill character. "7" with MinLen=3, PadChar='0', PadLeft=true becomes "007".

Input	Type	Description
Source	FString	Input string
Min Length	int32	Desired minimum length
Pad Char	FString	Character to fill with (default '0')
Pad Left	bool	True = prepend, False = append

Output	Type	Description
Return Value	FString	Padded string

🟩 *Great for level numbering (Level 01), leaderboard ranks, or file naming conventions.*

Split First

Splits a string at the first occurrence of a delimiter, returning the left and right parts via output pins.

Input	Type	Description
Source	FString	Input string to split
Delimiter	FString	Character or substring to split on

Output	Type	Description
Left	FString	Part before the delimiter
Right	FString	Part after the delimiter
Return Value	bool	True if delimiter was found

🟩 *Handy for parsing chat commands like "/give SwordOfFire" into command + argument.*

Join String Array

Joins an array of strings into a single string with a separator between each element.

Input	Type	Description
Array	TArray	Strings to join
Separator	FString	Inserted between elements (default ", ")

Output	Type	Description
Return Value	FString	Combined string

🟩 *Use for building comma-separated lists, log output, or combining tag arrays for display.*

2. Array

10 Nodes

Shuffle Int Array

Randomly shuffles an integer array using the Fisher-Yates algorithm. Returns a new array; the original is not modified.

Input	Type	Description
Array	TArray	Integer array to shuffle

Output	Type	Description
Return Value	TArray	New shuffled array

🟩 *Perfect for randomizing loot tables, quiz questions, enemy spawn orders, or card decks.*

Shuffle String Array

Randomly shuffles a string array using the Fisher-Yates algorithm. Returns a new array.

Input	Type	Description
Array	TArray	String array to shuffle

Output	Type	Description
Return Value	TArray	New shuffled array

🟩 *Use for randomizing dialogue options, name lists, or playlist order.*

Remove Duplicates (Int)

Removes duplicate integers from an array while preserving the original order of first occurrences.

Input	Type	Description
Array	TArray	Array with potential duplicates

Output	Type	Description
Return Value	TArray	Array with unique elements only

🟩 *Useful after merging ID lists or collecting indices from multiple overlap queries.*

Remove Duplicates (String)

Removes duplicate strings from an array while preserving order.

Input	Type	Description
Array	TArray	Array with potential duplicates

Output	Type	Description
Return Value	TArray	Array with unique elements only

🟩 *Use after merging tag collections, inventory lists, or player name arrays from multiple sources.*

Random Element (Int)

Picks a random element from an integer array. Optionally removes it from the source array (pop behavior).

Input	Type	Description
Array	TArray (by ref)	Source array (modified if Remove is true)
Remove	bool	If true, removes the picked element from array

Output	Type	Description
Return Value	int32	The randomly selected element
Success	bool	False if array was empty

🟩 *Ideal for drawing cards, picking random rewards, or selecting an enemy from a pool without repetition.*

Random Elements (Int)

Picks N unique random elements from an integer array without replacement.

Input	Type	Description
Array	TArray	Source array
Count	int32	How many elements to pick

Output	Type	Description
Return Value	TArray	Array of N unique random elements

🟩 *Great for selecting multiple loot drops, random shop items, or a subset of quiz questions.*

Sort Int Array

Sorts an integer array in ascending or descending order.

Input	Type	Description
Array	TArray	Array to sort
Descending	bool	True = highest first

Output	Type	Description
Return Value	TArray	Sorted array

🟩 *Combine with a score array for leaderboards, or sort indices by priority.*

Sort Float Array

Sorts a float array in ascending or descending order.

Input	Type	Description
Array	TArray	Array to sort
Descending	bool	True = highest first

Output	Type	Description
Return Value	TArray	Sorted array

🟩 *Use for sorting distances, weights, or damage values for priority-based logic.*

Sort String Array

Sorts a string array alphabetically, ascending or descending.

Input	Type	Description
Array	TArray	Array to sort
Descending	bool	True = Z→A

Output	Type	Description
Return Value	TArray	Sorted array

🟩 *Use for alphabetical inventory lists, player name sorting, or file listings.*

Weighted Random Index

Returns a random index biased by an array of float weights. Higher weight = higher chance. Weights [10, 5, 1] give roughly 62%, 31%, 6%.

Input	Type	Description
Weights	TArray	Array of weight values (>0)

Output	Type	Description
Return Value	int32	Selected index based on weights

🟩 *Essential for loot rarity systems, weighted spawn tables, or AI decision-making with priorities.*

3. Math

8 Nodes

Remap Range

Maps a value from one range to another. For example, 50 in [0, 100] maps to 0.5 in [0, 1]. Optional clamping keeps the result within the output range.

Input	Type	Description
Value	float	Input value to remap
In Min	float	Source range minimum
In Max	float	Source range maximum
Out Min	float	Target range minimum
Out Max	float	Target range maximum
Clamp	bool	Clamp result to output range (default: true)

Output	Type	Description
Return Value	float	Remapped value

🟩 *Use for health bar fill, audio volume curves, or mapping joystick input to movement speed.*

Round to Decimals

Rounds a float to a specified number of decimal places. 3.14159 with Decimals=2 becomes 3.14.

Input	Type	Description
Value	float	Number to round
Decimals	int32	Number of decimal places

Output	Type	Description
Return Value	float	Rounded value

🟩 *Great for displaying clean stats, prices, or percentages in UI without long decimal tails.*

Normalize Angle

Wraps an angle into a standard range: 0–360 (unsigned) or -180 to 180 (signed).

Input	Type	Description
Angle	float	Input angle in degrees
Signed	bool	True = -180..180, False = 0..360 (default)

Output	Type	Description
Return Value	float	Normalized angle

🟩 *Essential for compass HUDs, rotation comparisons, or turret aiming logic.*

Distance 2D

Calculates the distance between two vectors using only X and Y, ignoring the Z axis.

Input	Type	Description
A	FVector	First position
B	FVector	Second position

Output	Type	Description
Return Value	float	2D distance (XY plane)

🟩 *Perfect for top-down games, minimap distances, or any gameplay where height should not affect range checks.*

Is Nearly Equal (Float)

Compares two floats with a configurable tolerance (epsilon). Avoids the classic floating-point comparison pitfall.

Input	Type	Description
A	float	First value
B	float	Second value
Tolerance	float	Maximum allowed difference (default: 0.0001)

Output	Type	Description
Return Value	bool	True if difference <= tolerance

🟩 *Use for checking if a timer has reached a target, or if two positions are "close enough" to snap.*

Sign (Float)

Returns -1 if negative, 0 if zero, or 1 if positive.

Input	Type	Description
Value	float	Input number

Output	Type	Description
Return Value	int32	-1, 0, or 1

🟩 *Handy for flip-direction logic, wall-normal reactions, or quick positive/negative branching.*

Wrap (Float)

Correct modulo operation that handles negative values properly. Unlike the % operator, Wrap(-10, 0, 360) returns 350 instead of -10.

Input	Type	Description
Value	float	Input value
Min	float	Range minimum
Max	float	Range maximum

Output	Type	Description
Return Value	float	Wrapped value within [Min, Max)

🟩 *Use for looping indices, wrapping coordinates, or cyclic animations.*

Percentage

Calculates "A is X% of B" with safe division — returns 0 if B is zero instead of crashing.

Input	Type	Description
A	float	Part value
B	float	Whole value

Output	Type	Description
Return Value	float	Percentage (0–100+)

🟩 *Ideal for health percentage, loading progress, completion tracking, or stat comparisons.*

4. Actor / World

14 Nodes

Get Closest Actor of Class

Finds the nearest actor of a given class to a specified location. Optional max radius limits the search range.

Input	Type	Description
Actor Class	TSubclassOf	Class to search for
Location	FVector	Search origin
Max Radius	float	Maximum distance (0 = unlimited, default)

Output	Type	Description
Return Value	AActor*	Closest actor found (null if none)

🟩 *Use for target-lock systems, nearest pickup detection, or AI finding the closest patrol point.*

Get Actors in Radius

Returns all actors of a class within a radius, sorted by distance from closest to farthest.

Input	Type	Description
Actor Class	TSubclassOf	Class to search for
Location	FVector	Search origin
Radius	float	Search radius in units

Output	Type	Description
Return Value	TArray	Sorted array of actors in range

🟩 *Great for AOE damage, proximity alerts, or populating a "nearby enemies" UI list.*

Is Actor On Screen

Checks if an actor is currently within the player camera's view frustum.

Input	Type	Description
Actor	AActor*	Actor to check
PC	APlayerController*	Player whose screen to test

Output	Type	Description
Return Value	bool	True if actor is visible on screen

🟩 *Toggle health bars, name tags, or interaction prompts only for actors the player can actually see.*

Set Actor Active

Toggles an actor's visibility, collision, and tick in a single call. The all-in-one switch for object pooling.

Input	Type	Description
Actor	AActor*	Target actor
Active	bool	True = visible + collision + tick on

🟩 *Essential for object pooling patterns. Deactivate projectiles/enemies instead of destroying and respawning them.*

Get Player Character (Safe)

Returns the player character for a given player index. Returns null instead of crashing if the index is invalid.

Input	Type	Description
Player Index	int32	Player index (default 0)

Output	Type	Description
Return Value	ACharacter*	Player character or null

■ *Safer alternative to the engine's GetPlayerCharacter in contexts where the player may not exist yet (loading screens, menus).*

Get All Components of Type (Recursive)

Finds all components of a given type on an actor, including components on child actors.

Input	Type	Description
Actor	AActor*	Root actor to search
Component Class	TSubclassOf	Type to find

Output	Type	Description
Return Value	TArray	All matching components

■ *Use when working with modular actors (vehicles with child seats, weapons with attachments) to find all mesh or audio components.*

Spawn Actor Simple

Simplified SpawnActor that only needs a class and transform. No collision handling method or owner parameters required.

Input	Type	Description
Actor Class	TSubclassOf	Class to spawn
Spawn Transform	FTransform	Where and how to place it

Output	Type	Description
Return Value	AActor*	Newly spawned actor

■ *Perfect for quick prototyping, wave spawners, or any situation where default spawn settings are fine.*

Get Ground Location

Traces straight down from a position to find the ground. Returns the hit location and surface normal via output pins.

Input	Type	Description
Start Location	FVector	Starting point for downward trace
Trace Distance	float	How far down to check (default 10000)

Output	Type	Description
Ground Location	FVector	World position of ground hit
Ground Normal	FVector	Surface normal at hit point
Return Value	bool	True if ground was found

■ *Use for placing objects on terrain, footstep effects aligned to slopes, or gravity-based spawn systems.*

Is Facing Actor

Checks if the Source actor is looking toward the Target actor within a configurable angle tolerance.

Input	Type	Description
Source	AActor*	The looking actor
Target	AActor*	The target actor
Angle Tolerance Deg	float	Half-angle in degrees (default 45)

Output	Type	Description
Return Value	bool	True if Source faces Target within tolerance

🟩 *Combine with Visibility Check for stealth/detection. Also useful for dialogue triggers ("face the NPC to talk").*

Look At Rotation (Smooth)

Calculates an interpolated rotation for an actor toward a target location. Call per-tick for smooth turret or camera tracking.

Input	Type	Description
Source	AActor*	The actor that is rotating
Target Location	FVector	World point to look at
Interp Speed	float	Rotation speed
Delta Time	float	Frame delta time

Output	Type	Description
Return Value	FRotator	New interpolated rotation

🟩 *Ideal for turrets, security cameras, AI head-tracking, or smooth third-person camera pivots.*

Get Forward Direction 2D

Returns the actor's forward vector with Z zeroed out and normalized. For movement on the XY plane.

Input	Type	Description
Actor	AActor*	Source actor

Output	Type	Description
Return Value	FVector	Forward direction (Z=0, normalized)

🟩 *Use in top-down or 2.5D games where vertical component of facing direction would cause issues.*

Predict Actor Location

Estimates where an actor will be in N seconds based on its current velocity.

Input	Type	Description
Actor	AActor*	Actor to predict
Time Ahead	float	Seconds into the future

Output	Type	Description
Return Value	FVector	Estimated future position

🟩 *Great for AI lead-aiming, projectile intercept calculations, or predictive pathfinding.*

Get Distance to Ground

Returns how far above the ground an actor currently is.

Input	Type	Description
Actor	AActor*	Target actor
Max Trace	float	Maximum trace distance (default 10000)

Output	Type	Description
Return Value	float	Height above ground in units

🟩 *Use for fall damage thresholds, landing detection, or "ground slam" ability range checks.*

Find Look At Rotation 2D

Calculates a look-at rotation from one point to another, ignoring the Z axis entirely.

Input	Type	Description
Source	FVector	Observer position
Target	FVector	Target position

Output	Type	Description
Return Value	FRotator	Yaw-only rotation toward target

🟩 *Perfect for top-down character facing, billboard effects, or 2D minimap indicators.*

5. Debug

6 Nodes

Fox Print

Enhanced on-screen debug message with custom color, duration, and category prefix. Optionally also writes to the output log. Development only — stripped in shipping builds.

Input	Type	Description
Message	FString	Text to display
Color	FLinearColor	Text color on screen (default: Green)
Duration	float	How long to show in seconds (default: 5)
Category	FString	Optional prefix category label
Also Log	bool	Also write to output log (default: true)

🟩 *The go-to debug node. Use colors to differentiate systems (red=damage, blue=AI, green=pickups).*

Fox Log

Writes a message to the Unreal output log with a custom category string. Supports Normal and Warning levels. Development only.

Input	Type	Description
Message	FString	Log message text
Category	FString	Log category label (default: "FoxUtils")
Warning	bool	True = log as Warning, False = Normal

🟩 *Use for persistent logging that survives screen-clear. Filter by category in the Output Log window.*

Print Vector

Prints a labelled vector to the screen in a clean, readable format. No more manually breaking vectors into 3 floats. Development only.

Input	Type	Description
Label	FString	Prefix label
Value	FVector	Vector to display
Color	FLinearColor	Text color (default: Yellow)
Duration	float	Display duration (default: 5)

🟩 *Essential for debugging movement, physics, or spatial calculations. Shows as "Label: (X, Y, Z)".*

Perf Timer Start

Starts a named performance timer. Call before the code section you want to measure. Development only.

Input	Type	Description
Label	FString	Unique name for this timer

🟩 *Combine with Perf Timer Stop to measure any Blueprint operation. Great for finding bottlenecks in loops or heavy calculations.*

Perf Timer Stop

Stops a named performance timer and returns the elapsed time in milliseconds. Optionally prints the result on screen. Development only.

Input	Type	Description
Label	FString	Must match the Start label
Print Result	bool	Print elapsed time on screen (default: true)

Output	Type	Description
Return Value	float	Elapsed time in milliseconds

🟩 *Results appear on screen as "Label: 2.34 ms". Use to profile before optimizing — measure, don't guess.*

Conditional Print

Only prints a debug message if a boolean condition is true. Eliminates the Branch + Print node pair. Development only.

Input	Type	Description
Condition	bool	Print only if true
Message	FString	Text to display
Color	FLinearColor	Text color (default: White)
Duration	float	Display time in seconds (default: 5)

🟩 *Reduces graph clutter. Use for conditional state logging like "only print when health < 20%".*

6. Platform

10 Nodes

Is Packaged Build

Returns true when running in a packaged/shipping build, false in the editor or PIE.

Output	Type	Description
Return Value	bool	True = packaged build

🟩 *Use to disable debug UI, cheats, or developer tools in your shipped game.*

Copy to Clipboard

Copies a string to the system clipboard.

Input	Type	Description
Text	FString	Text to copy

🟩 *Useful for "Copy Server IP", "Copy Error Log", or sharing save codes.*

Open URL in Browser

Opens the system's default web browser and navigates to the given URL.

Input	Type	Description
URL	FString	Web address to open

🟩 *Link to your Discord, documentation, patch notes, or feedback form from in-game settings.*

Get Engine Version

Returns the current Unreal Engine version string (e.g. "5.7.0").

Output	Type	Description
Return Value	FString	Engine version string

🟩 *Useful for plugin compatibility checks or displaying engine info in a debug overlay.*

Get Platform Name

Returns the platform the game is running on: "Windows", "Linux", "Mac", etc.

Output	Type	Description
Return Value	FString	Platform name string

🟩 *Use for platform-specific input hints ("Press A" vs "Press Space") or conditional feature toggling.*

Get GPU Name

Returns the name of the graphics card (e.g. "NVIDIA GeForce RTX 4090").

Output	Type	Description
Return Value	FString	GPU model name

🟩 *Display in a system info screen or use to auto-detect graphics quality presets.*

Get Total RAM (MB)

Returns the total system RAM in megabytes.

Output	Type	Description
Return Value	int32	Total RAM in MB

🟩 *Combine with Get Total VRAM for an auto-quality settings system on first launch.*

Get Total VRAM (MB)

Returns the GPU's video memory in megabytes.

Output	Type	Description
Return Value	int32	Total VRAM in MB

🟩 *Use to decide texture quality presets: <4GB = Low, 4–8GB = Medium, >8GB = High/Ultra.*

Get CPU Brand

Returns the CPU brand string (e.g. "Intel Core i7-12700K").

Output	Type	Description
Return Value	FString	CPU model string

🟩 *Show in a hardware info screen alongside GPU and RAM details.*

Get Desktop Resolution

Returns the native monitor resolution as an IntPoint (width, height).

Output	Type	Description
Return Value	FIntPoint	Monitor resolution (X=width, Y=height)

🟩 *Use to populate resolution dropdowns in settings menus or determine default window size.*

7. Collision / Trace

8 Nodes

Sphere Trace for Actors

Finds all actors of a class within a sphere radius, sorted by distance. Combines overlap + sort in one node.

Input	Type	Description
Origin	FVector	Center of the sphere
Radius	float	Sphere radius
Filter Class	TSubclassOf	Only return this class

Output	Type	Description
Return Value	TArray	Actors found, sorted by distance

🟩 *Use for explosion damage falloff, AOE heals, or proximity detection where you need distance sorting.*

Cone Check for Actors

Returns all actors within a cone defined by origin, direction, range, and half-angle. Perfect for field-of-view mechanics.

Input	Type	Description
Origin	FVector	Cone tip position
Direction	FVector	Cone direction
Range	float	Cone length
Half Angle Deg	float	Half of the cone angle in degrees
Filter Class	TSubclassOf	Only return this class

Output	Type	Description
Return Value	TArray	Actors inside the cone

🟩 *Ideal for stealth detection, flashlight mechanics, shotgun spread, or AI peripheral vision.*

Ground Check

Vertical trace downward returning a full FHitResult including physical material. More detailed than Get Ground Location.

Input	Type	Description
Location	FVector	Start position
Trace Depth	float	Distance downward (default 10000)
Complex	bool	Use complex collision (default: false)

Output	Type	Description
Hit Result	FHitResult	Full hit data including phys material
Return Value	bool	True if something was hit

🟩 *Use the physical material for footstep sounds — detect grass, stone, metal, wood surfaces automatically.*

Wall Check

Horizontal trace in the actor's forward direction. Detects walls for parkour, cover, or collision avoidance.

Input	Type	Description
Actor	AActor*	Source actor
Trace Length	float	How far forward to check (default 200)
Height Offset	float	Vertical offset from actor origin (default 0)

Output	Type	Description
Hit Result	FHitResult	Wall hit details
Return Value	bool	True if wall detected

🟩 *Combine with Ground Check to build a full parkour detection system (wall run + ledge grab).*

Line Trace (Ignore Self)

Standard line trace that automatically ignores the calling actor and any additional specified actors.

Input	Type	Description
Source Actor	AActor*	Actor to auto-ignore
Start	FVector	Trace start
End	FVector	Trace end
Additional Ignore	TArray	Extra actors to skip

Output	Type	Description
Hit Result	FHitResult	Hit data
Return Value	bool	True if something was hit

🟩 *Saves the manual "add self to ignore" step. Use for gunfire, grapple hooks, or interaction detection.*

Visibility Check

Line-of-sight test between two actors. Automatically ignores both actors in the trace, plus optional extra ignores.

Input	Type	Description
From	AActor*	First actor
To	AActor*	Second actor
Ignore Actors	TArray	Additional actors to ignore

Output	Type	Description
Return Value	bool	True if clear line of sight

🟩 *Combine with Cone Check or Is Facing Actor for full stealth detection (angle + range + line of sight).*

Box Trace Simple

Sweeps a box shape from start to end position. Useful for wider collision tests than a single line.

Input	Type	Description
Start	FVector	Sweep start
End	FVector	Sweep end
Half Extent	FVector	Box half-size
Orientation	FRotator	Box rotation

Output	Type	Description
Hit Result	FHitResult	Hit data
Return Value	bool	True if something was hit

🟩 *Use for wide melee attacks (sword slash), vehicle collision preview, or area scanning.*

Multi Line Trace

Returns ALL hits along a line, not just the first. Traces through multiple objects.

Input	Type	Description
Start	FVector	Trace start
End	FVector	Trace end
Trace Complex	bool	Use complex collision (default: false)

Output	Type	Description
Return Value	TArray	All hit results along the line

🟩 *Perfect for penetrating bullets, X-ray vision scanners, or counting how many walls a trace passes through.*

8. Delayed Loops

6 Nodes

Fox For Loop with Delay

Iterates from FirstIndex to LastIndex with a configurable delay between each step. Uses FPendingLatentAction — no Tick abuse, proper cleanup.

Input	Type	Description
First Index	int32	Loop start (inclusive)
Last Index	int32	Loop end (inclusive)
Delay Per Step	float	Seconds between iterations

Output	Type	Description
Loop Body	exec	Fires each iteration
Completed	exec	Fires when loop finishes
Index	int32	Current loop index

🟩 *Use for wave spawning, sequential UI animations, or staggered effects. Much cleaner than timer + counter setups.*

Fox For Each with Delay

Iterates over an array length with a delay between each element. Same latent action system as the For Loop variant.

Input	Type	Description
Array Length	int32	Number of elements to iterate
Delay Per Step	float	Seconds between elements

Output	Type	Description
Loop Body	exec	Fires for each element
Completed	exec	Fires when done
Index	int32	Current array index

🟩 *Perfect for populating UI lists with a stagger effect, or processing inventory items one at a time.*

Fox Batched Loop with Delay

Processes BatchSize elements per frame with a delay between batches. E.g. 1000 elements at BatchSize=50, Delay=0.01 runs 50 per frame across 20 frames.

Input	Type	Description
First Index	int32	Loop start (inclusive)
Last Index	int32	Loop end (inclusive)
Batch Size	int32	Elements per frame/batch
Delay Per Batch	float	Seconds between batches

Output	Type	Description
Loop Body	exec	Fires for each element
Completed	exec	All batches done
Index	int32	Current element index

🟩 *Essential for heavy operations (procedural generation, navmesh queries) that would freeze the game in one frame.*

Fox For Loop with Delay (Reversed)

Same as Fox For Loop with Delay but iterates from LastIndex down to FirstIndex (backwards).

Input	Type	Description
First Index	int32	Lower bound
Last Index	int32	Upper bound (starts here)
Delay Per Step	float	Seconds between steps

Output	Type	Description
Loop Body	exec	Fires each iteration
Completed	exec	Fires when done
Index	int32	Current index (descending)

🟩 *Use when removing items from arrays during iteration (always remove from end to avoid index shifting).*

Fox For Each with Delay (Reversed)

Iterates an array backwards with a delay between elements.

Input	Type	Description
Array Length	int32	Number of elements
Delay Per Step	float	Seconds between elements

Output	Type	Description
Loop Body	exec	Fires each iteration
Completed	exec	Fires when done
Index	int32	Current index (descending)

🟩 *Combine with array removal logic for safe "destroy all enemies" sequences with visual stagger.*

Cancel Delayed Loop

Stops a running delayed loop by its UUID. The loop stops immediately and does not fire the Completed pin.

Input	Type	Description
UUID	int32	Loop handle (from LatentInfo)

🟩 *Use to interrupt wave spawning when the player dies, or cancel a UI animation if the menu closes early.*

9. Instanced Mesh (HISM/FISM)

6 Nodes

HISM Refresh Custom Data

Calls BuildTreeIfOutdated on a Hierarchical Instanced Static Mesh component. Essential after modifying custom data values.

Input	Type	Description
HISM	UHierarchicalInstancedStaticMeshComponent*	Target HISM component
Async	bool	Rebuild asynchronously (default: true)
Force Update	bool	Force rebuild even if not outdated (default: false)

🟩 *Always call this after changing custom data (color, opacity, state). Without it, visual updates may not appear.*

FISM Refresh Custom Data

Same as HISM Refresh but for regular (Foliage) Instanced Static Mesh components.

Input	Type	Description
ISM	UInstancedStaticMeshComponent*	Target ISM component
Async	bool	Rebuild asynchronously (default: true)
Force Update	bool	Force rebuild even if not outdated (default: false)

🟩 *Use when working with foliage or non-hierarchical instanced mesh components that have custom data.*

HISM Get Instances in Radius

Returns instance indices within a radius of a given point. Much faster than iterating all instances manually.

Input	Type	Description
HISM	UHierarchicalInstancedStaticMeshComponent*	Target HISM
Location	FVector	Search center
Radius	float	Search radius

Output	Type	Description
Return Value	TArray	Instance indices within radius

🟩 *Use for harvesting systems (chop trees near player), area effects on foliage, or LOD management.*

HISM Get Nearest Instance

Finds the single closest instance and returns its index and world transform.

Input	Type	Description
HISM	UHierarchicalInstancedStaticMeshComponent*	Target HISM
Location	FVector	Reference point

Output	Type	Description
Return Value	int32	Nearest instance index (-1 if none)
Out Transform	FTransform	World transform of nearest instance

🟩 *Ideal for interaction prompts ("Press E to harvest nearest tree") or snapping placed objects to the closest instance.*

HISM Batch Add Instances

Adds many instances at once from an array of transforms. Returns an array of the new instance indices. Much faster than adding one at a time.

Input	Type	Description
HISM	UHierarchicalInstancedStaticMeshComponent*	Target HISM
Transforms	TArray	Transforms for new instances
World Space	bool	Transforms are in world space (default: false)

Output	Type	Description
Return Value	TArray	Indices of newly added instances

🟩 *Use for procedural placement (scatter rocks, spawn grass patches). Batch operations avoid per-instance tree rebuilds.*

HISM Batch Update Custom Data

Updates custom data float values for multiple instances in one call. Avoids repeated tree rebuilds.

Input	Type	Description
HISM	UHierarchicalInstancedStaticMeshComponent*	Target HISM
Indices	TArray	Instance indices to update
Custom Data Index	int32	Custom data channel (0, 1, 2...)
Value	float	New value for all specified instances

🟩 *Perfect for mass-updating foliage states (seasonal color, wind intensity) or highlighting all instances in a zone.*

10. Camera

6 Nodes

Camera Shake Simple

Triggers a quick camera shake with configurable amplitude, frequency, and duration. No CameraShake asset required.

Input	Type	Description
Amplitude	float	Shake intensity (default 5)
Frequency	float	Shake speed (default 20)
Duration	float	Shake length in seconds (default 0.5)
Player Index	int32	Which player to shake (default 0)

🟩 *Use for explosions, melee impacts, or landing effects. Small amplitude (0.5–2.0) feels impactful without being nauseating.*

Get Screen Center World Location

Projects the exact center of the screen into world space. Returns both the world position and direction.

Input	Type	Description
PC	APlayerController*	Player whose screen to use

Output	Type	Description
World Location	FVector	World position at screen center
World Direction	FVector	Forward direction from screen center
Return Value	bool	True if deprojection succeeded

🟩 *Use for crosshair-based aiming, interaction raycasts from screen center, or placing objects where the player looks.*

Project to Screen (Safe)

Projects a world position to screen coordinates. Unlike the engine version, returns false when the point is behind the camera.

Input	Type	Description
PC	APlayerController*	Player context
World Location	FVector	3D position to project

Output	Type	Description
Screen Position	FVector2D	2D screen coordinates
Return Value	bool	False if behind camera or off-screen

🟩 *Use for HUD markers, off-screen indicators, or name tags. The safe check prevents markers from flipping when behind you.*

Camera Distance to Actor

Returns the distance from the player's camera to any actor in the world.

Input	Type	Description
PC	APlayerController*	Player context
Actor	AActor*	Target actor

Output	Type	Description
Return Value	float	Distance in world units

🟩 *Use for LOD management, detail fading, or scaling UI elements based on distance from camera.*

Camera Look At Rotation

Smooth interpolated rotation from the current camera orientation toward a world target point.

Input	Type	Description
PC	APlayerController*	Player context
Target Location	FVector	World point to look at
Interp Speed	float	Rotation speed
Delta Time	float	Frame delta time

Output	Type	Description
Return Value	FRotator	New camera rotation

🟩 *Useful for cutscene cameras, lock-on targeting, or cinematic transitions between points of interest.*

Get Camera Transform

Returns the player camera's current world location and rotation in a single call.

Input	Type	Description
PC	APlayerController*	Player context

Output	Type	Description
Location	FVector	Camera world position
Rotation	FRotator	Camera world rotation
Return Value	bool	True if camera data is valid

🟩 *Saves two separate calls. Use for spawning projectiles from camera, audio listener position, or debug visualization origin.*

11. Audio

5 Nodes

Play Sound Extended

Plays a sound at a location with volume, pitch, start time, and attenuation all in one node. No AudioComponent setup needed.

Input	Type	Description
Sound	USoundBase*	Sound asset to play
Location	FVector	World position
Volume	float	Volume multiplier (default 1.0)
Pitch	float	Pitch multiplier (default 1.0)
Start Time	float	Offset in seconds to start playback (default 0)
Attenuation	USoundAttenuation*	Optional attenuation settings

Output	Type	Description
Return Value	UAudioComponent*	Spawned audio component

🟩 *Use for one-shot effects (explosions, pickups). For looping sounds, store the returned Audio Component to stop later.*

Play Random Sound

Takes an array of sounds and plays one at random. Ideal for natural variation in repetitive effects.

Input	Type	Description
Sounds	TArray	Array of sound options
Location	FVector	Play location
Volume	float	Volume multiplier (default 1.0)
Pitch	float	Pitch multiplier (default 1.0)

Output	Type	Description
Return Value	UAudioComponent*	The playing audio component

🟩 *Essential for footsteps, impact sounds, UI clicks. 3–5 variations prevent the "machine gun" repetition effect.*

Fade Sound Volume

Smoothly transitions a playing audio component's volume over a specified duration.

Input	Type	Description
Audio Component	UAudioComponent*	Sound to fade
Target Volume	float	End volume (0 = silence)
Fade Duration	float	Fade time in seconds

🟩 *Perfect for music crossfades, ambient zone transitions, or fading out combat music after the last enemy dies.*

Stop All Sounds on Actor

Immediately stops all AudioComponents attached to an actor.

Input	Type	Description
Actor	AActor*	Target actor

🟩 *Call when destroying/deactivating actors. Prevents orphaned looping sounds (e.g. engine hum on a despawned vehicle).*

Is Any Sound Playing

Checks if any AudioComponent on an actor is currently playing.

Input	Type	Description
Actor	AActor*	Actor to check

Output	Type	Description
Return Value	bool	True if at least one sound is active

🟩 *Use to avoid stacking duplicate sounds on the same actor, or to wait until a sound finishes before playing the next.*

12. Save / Load

6 Nodes

Save String to INI

Saves a string value to an INI file under a section and key. File stored in Saved/Config/.

Input	Type	Description
Section	FString	INI section name (e.g. "Player")
Key	FString	Key within section (e.g. "Name")
Value	FString	String to save
File Name	FString	INI file name without extension (default: "FoxUtils")

🟩 *Use for lightweight settings: last selected level, player nickname, preferred language. No SaveGame boilerplate needed.*

Load String from INI

Loads a string value from an INI file. Returns a default if the key does not exist.

Input	Type	Description
Section	FString	INI section name
Key	FString	Key to look up
Default	FString	Fallback if not found (default: empty)
File Name	FString	INI file name (default: "FoxUtils")

Output	Type	Description
Return Value	FString	Loaded value or default

🟩 *Always provide a sensible default. The file is created automatically on first save.*

Save Int to INI

Saves an integer value to an INI file.

Input	Type	Description
Section	FString	INI section
Key	FString	Key name
Value	int32	Integer to save
File Name	FString	INI file name (default: "FoxUtils")

🟩 *Great for high scores, unlock flags, or simple progression counters.*

Load Int from INI

Loads an integer from INI with a default fallback.

Input	Type	Description
Section	FString	INI section
Key	FString	Key name
Default	int32	Fallback value if not found (default: 0)
File Name	FString	INI file name (default: "FoxUtils")

Output	Type	Description
Return Value	int32	Loaded integer or default

🟩 *Combine multiple keys under the same section for structured data: [Player] HighScore=100, Level=5, Coins=42.*

Save Float to INI

Saves a float value to an INI file.

Input	Type	Description
Section	FString	INI section
Key	FString	Key name
Value	float	Float to save
File Name	FString	INI file name (default: "FoxUtils")

🟩 *Use for volume levels, mouse sensitivity, brightness — any settings slider value.*

Load Float from INI

Loads a float from INI with a default fallback.

Input	Type	Description
Section	FString	INI section
Key	FString	Key name
Default	float	Fallback value (default: 0.0)
File Name	FString	INI file name (default: "FoxUtils")

Output	Type	Description
Return Value	float	Loaded float or default

🟩 *Load settings on game start, apply them to your systems, and let FoxUtils handle the file I/O.*

13. Inventory Helpers

5 Nodes

Calculate Stack Split

Splits a total count into N even stacks. 10 items into 3 stacks = [4, 3, 3]. Remainder distributed fairly.

Input	Type	Description
Total Count	int32	Total number of items
Num Stacks	int32	Number of stacks to split into

Output	Type	Description
Return Value	TArray	Array of stack sizes

🟩 *Use for inventory splitting, resource distribution among players, or dividing loot drops evenly.*

Can Stack Items

Checks if two item stacks can merge: same type (by FName match) and the target has room. Returns true if Current + Add <= MaxStackSize and names match.

Input	Type	Description
Item Type A	FName	Type name of first stack
Current Count A	int32	Items currently in the target stack
Item Type B	FName	Type name of items to add
Add Count	int32	Number of items to add
Max Stack Size	int32	Maximum stack size

Output	Type	Description
Return Value	bool	True if items can merge

🟩 *Call before attempting to merge stacks in your inventory UI. Prevents silent failures or exceeding limits.*

Calculate Total Weight

Sums up weight across an inventory from an array of FFoxWeightEntry structs (Weight + Count per entry).

Input	Type	Description
Items	TArray	Array of {Weight, Count} entries

Output	Type	Description
Return Value	float	Sum of all (Weight × Count)

🟩 *Use for encumbrance systems. Compare against max carry weight to slow the player or block picking up items.*

Fits in Container

Checks if an item fits in a container given its size and currently occupied slots.

Input	Type	Description
Item Size	int32	Slots required by the item
Container Slots	int32	Total slots in container
Occupied Slots	int32	Currently used slots

Output	Type	Description
Return Value	bool	True if there is room

🟩 *Quick pre-check before adding items. For grid-based inventories, use Find Best Fit Slot instead.*

Find Best Fit Slot (Grid)

Finds the first available position in a 2D grid where an item of given dimensions fits. Scans top-left to bottom-right. Returns true if a slot was found.

Input	Type	Description
Occupied Cells	TArray	Flat bool array, row-major (true = occupied)
Grid Width	int32	Number of columns
Grid Height	int32	Number of rows
Item Width	int32	Item width in cells
Item Height	int32	Item height in cells

Output	Type	Description
Out Cell X	int32	Top-left column (-1 if no fit)
Out Cell Y	int32	Top-left row (-1 if no fit)
Return Value	bool	True if a position was found

🟩 *The core node for Diablo/Resident Evil style grid inventories. Mark cells as occupied after placement.*

FoxToolkit Ecosystem

FoxUtils is part of the **FoxToolkit** family of Unreal Engine plugins by Alchemy Fox Studio. All plugins work standalone — no dependencies required.

Install **FoxCore** (free) to enable cross-plugin communication via a GameInstance Subsystem event bus with Gameplay Tags.

Browse all FoxToolkit plugins: fab.com/sellers/AlchemyFoxStudio

Support

Questions or bugs? Contact Alchemy Fox Studio at alchemyfoxstudio@gmail.com