




Unreal Engine 3 Series Introduction to UnrealKismet

written by David Lally | davidmlally@gmail.com

The following tutorial will cover an in-depth overview of the benefits, features, and functionality within Unreal's node based scripting editor, Kismet. This document will cover an interface overview; a review of some of the more commonly used actions, various events and conditions that are available, as well as some functional examples. The tutorial assumes some basic knowledge of the Unreal 3 Editor, as well as some familiarity with the node-based workflow.

Kismet Interface Overview

The Kismet interface can be accessed by simply clicking the Kismet  icon in the Unreal 3 Editor's toolbar. The window for UnrealKismet is fairly simple and broken down into four major areas: the toolbar, the node workspace (large, central area), the properties window (bottom left), and the sequence window (bottom right). See Fig 1.0.

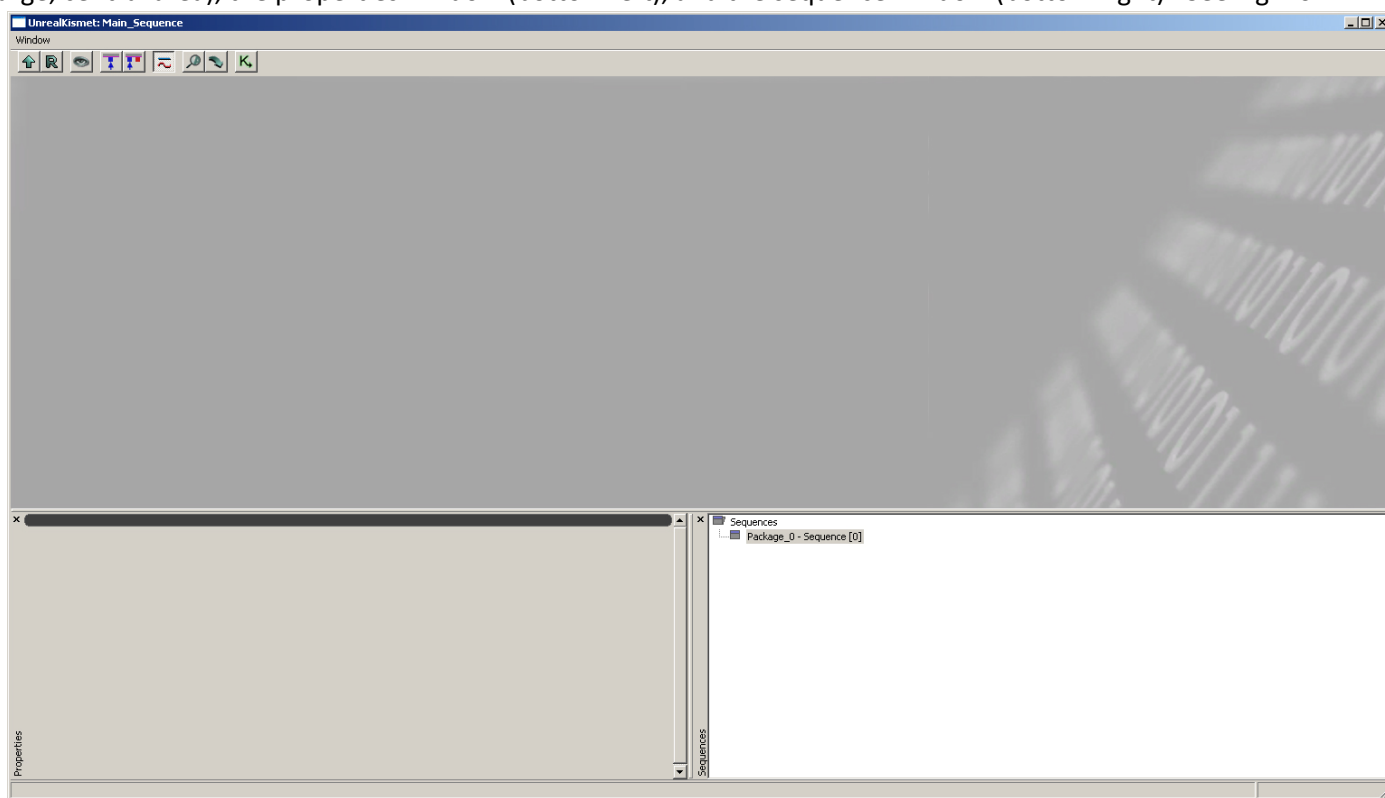


Fig 1.0 – The work space for Unreal Kismet. Note the options in the toolbar, the open node work space, the properties window in the bottom left, and the sequence window in the bottom right.

Interface: The Toolbar

With the general interface in mind, let's take a look at the specific areas. First, the toolbar features a number of useful

hot buttons for navigating and organizing the node workspace (Fig 1.1).

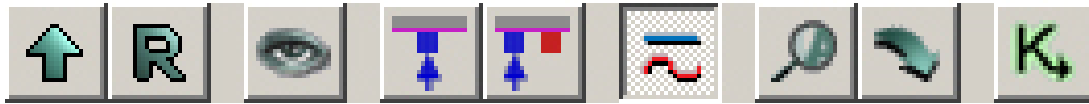


Fig 1.1 – The Toolbar’s Options

Here’s a basic break down of what these hot buttons can do:

Open Parent Sequence – Acts like an “Up One Directory” button in any Window’s Explorer style window but specifically for nodes within nodes.

Rename Sequence – Straight forward –renames a sequence.

Zoom to Fit – If your nodes are unbearably a mess and you can’t locate something, zoom to fit will bring everything to 100% magnification and center it.

Hide Unused Connectors – This will simply hide any connectors that aren’t being used. They are still there, just hidden.

Show All Connectors – This will show all connectors, even ones that were previously hidden.

Toggle Curved Connections – An aesthetic flow chart option; Choose between linear and curved.

Search Tool – Search for nodes, etc. more easily here. Search off of names, comments, etc.

Update List – For any external information that has been updated and is out of date (indicated in red).

Open New Window – Have multiple Kismet windows open at once for different scripts, etc.

Interface: The Node Workspace

The node workspace is very similar to other node-based applications such as Houdini or even Maya’s Hypergraph. If you’ve already used Unreal 3’s Material Editor, you’ll feel comfortable in this workspace right away. The controls are the same (Fig 1.2).

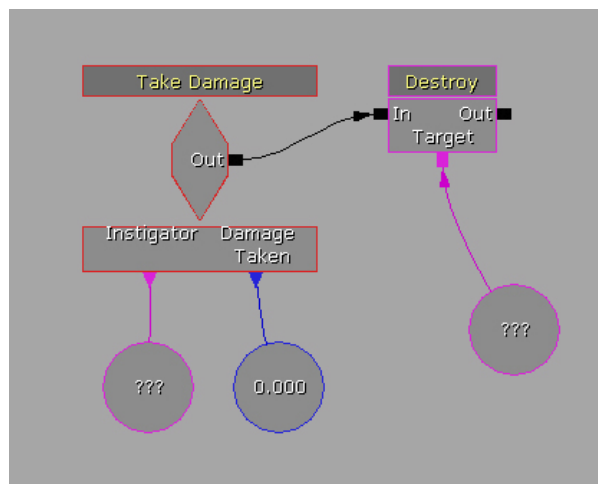


Fig 1.2 – Some sample nodes dropped into the open workspace.

Controls:

Left or Right click anywhere in the workspace to move around

Use the scroll wheel to zoom in and out of your nodes.

Left click to select any one node. Control click to select additional nodes.

Hold Control and left click to drag your nodes around.

Left click on an input or output and drag it to another input or output to create a connector between two nodes.

Right click in an empty area of the workspace to receive a quick menu of various nodes to utilize.

Side Note: You can also right click to add comments to existing nodes in order to keep track of your progress so you never lose your spot. This is great for situations where multiple programmers are involved with editing the same scripts.

Interface: The Properties Panel

Right click in your empty workspace. Go to New Event > Actor > Take Damage. You now see a “Take Damage” event node in your work space, similar to the one seen in Fig. 1.2. If you select this node, you’ll notice that the properties panel fills up with information. This is where all of the data for the “Take Damage” event is stored, including whether or not it is enabled. Be sure to always check “bEnabled” on your nodes that you want active from the start. You can leave them inactive if you want to activate them later through scripting. Don’t worry about selecting anything for the time being, but become familiar with some of the properties available (Fig. 1.3).

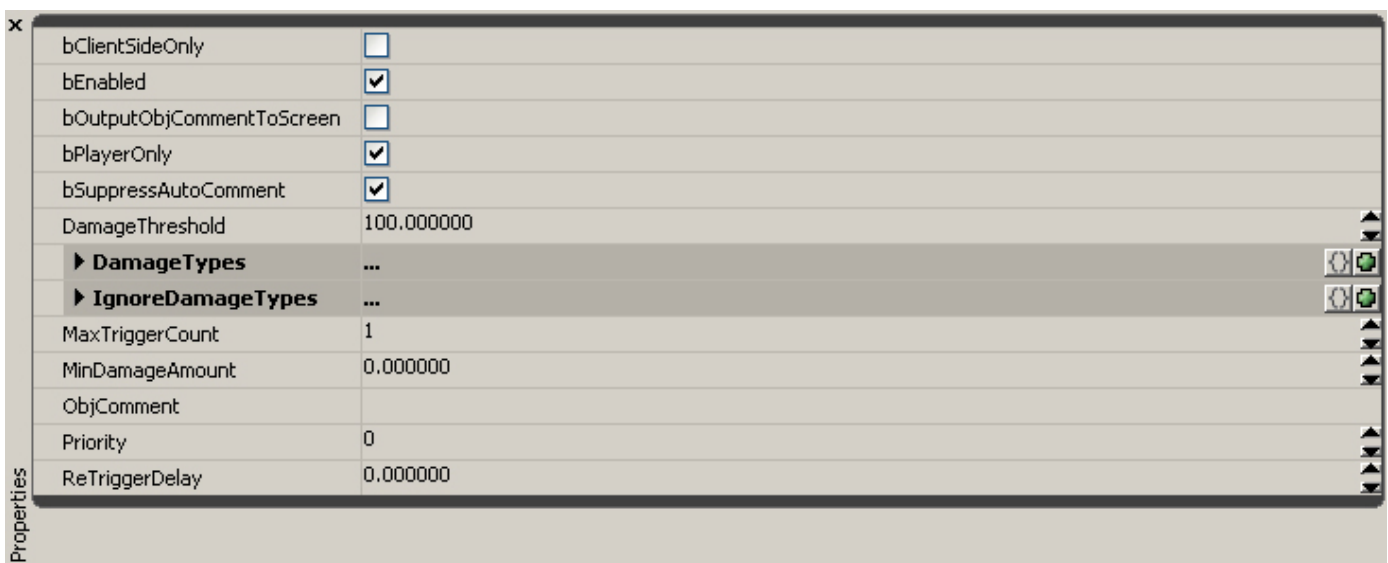


Fig 1.3 – Sample information in the Properties Panel

Interface: Sequences

This will allow you to manage your various sequences within the Kismet editor. It will become useful when we begin dealing with multiple packages with different scripts within each of them.

Common Action Nodes and Functions

There are many different action nodes available within UnrealKismet which will offer a wide range of functionality down the road, but in order to ease ourselves into this visual scripting method, let's simply take a look at some of the more common nodes we'll be using and what they do for us (Fig 1.4).

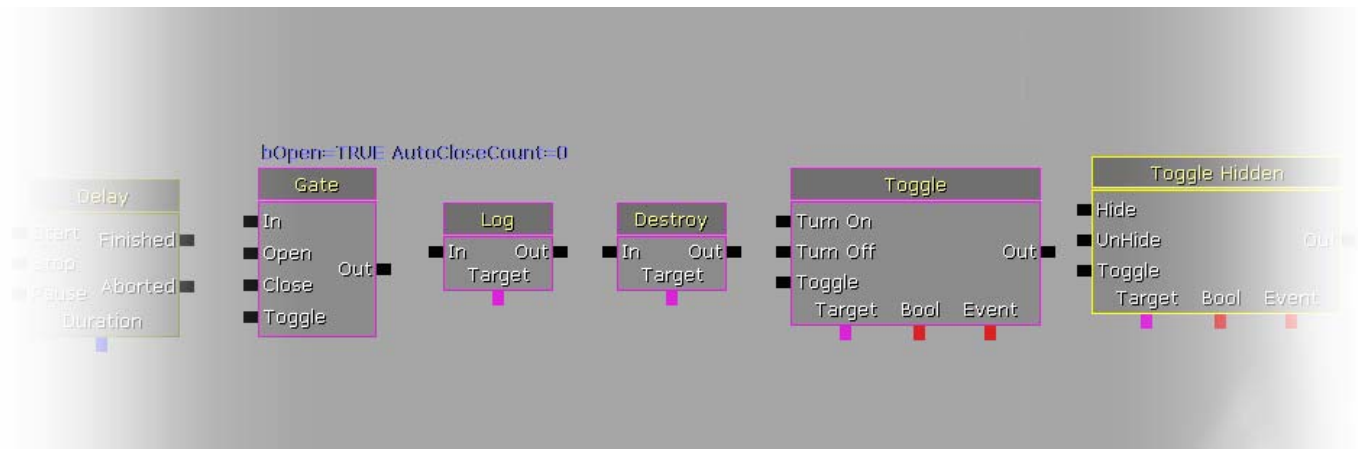


Fig 1.4 – Some of the more common nodes in Kismet.

To see the full list of available action nodes, right click in your workspace and click the “New Action” menu. Here, you will see a wide range of different action types. From those types, we’ll now go into detail of the most frequently used nodes.

Miscellaneous:

Delay – The ability to add a delay to any event. For example, if the player activates a trigger, we can set a delay before the result of that trigger is set off.

Gate – A gate is used as a method of filtering the resulting output of a node to check if the output is True or False and determine what output is able to flow the node into resulting nodes. It can also be used to toggle objects.

Log – Simply logs out info for testing and feedback purposes. Similar to a trace command in other scripting languages.

Actor:

Destroy – A simple and effective way to completely remove an actor from the level. For example, if a barrel from our game is shot and broken apart, we can then “destroy” it completely and remove it from memory with this script.

Toggle:

Toggle – An effective method of toggling actors “on” or “off.” For example we may have various lights in the scene that can be switched on or off in a room. It’s extremely useful since multiple objects can be toggled simultaneously through a single node.

Toggle Hidden – Very similar to our other toggle, however the Toggle Hidden will hide or unhide objects. This is useful if you don’t want to “Destroy” the actor completely and need the option to bring it back into the player’s vision later on.

Event:

Remote Event – This node acts as a method to jump around in the script to another area that would act as the receiving end of our node pipeline. Notice that it has no output because of this.

Switch:

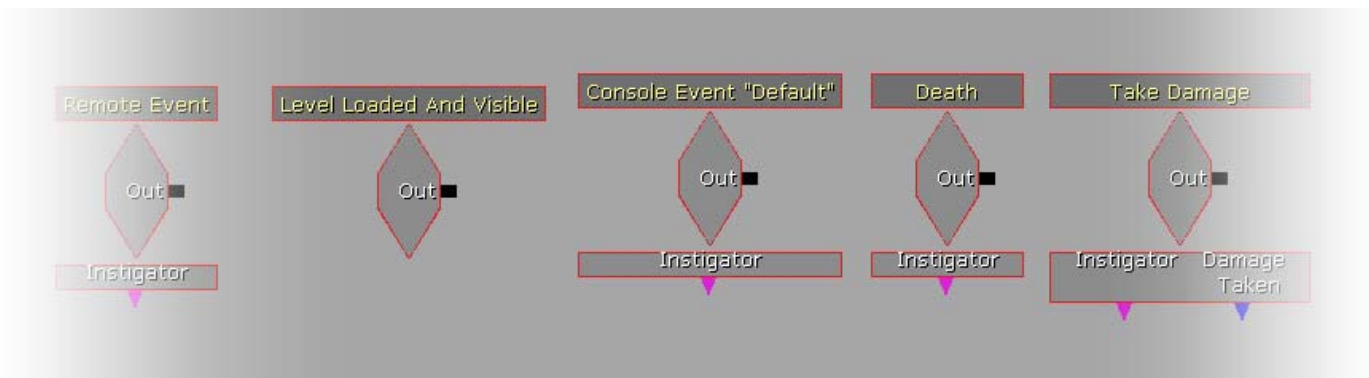
Switch – This node allows you to have multiple outputs over time by adding a set amount to the first “Link” within the node. For example, if a player receives a power up in the game we can have a certain output occur. The link then receives the “OK” to add the set amount to itself and now when the player receives another power up, we can activate a completely different output.

Delayed Switch – This is very similar to the basic Switch node, however we can set a delay between the various Links to time out outputs more effectively in order to drive a particular style of response that may be preferred.

As I mentioned previously, these are just some of the available nodes that you can work with that will become the basis for very extensive scripts that can allow a seemingly endless range of functionality to your game. With this node based workflow, even the team's designated artist can jump in and set up some scripts and test their work for implementation. I urge you to explore the functionality of other nodes and actively seek out creative methods of scripting online and elsewhere.

Common Events and Conditions

Just as there were many commonly used action nodes, there are a number of event and condition nodes that will see the same amount of use. The methodology of exposing you to these nodes is going to be the same. We'll take a look at some of the more commonly used event and condition nodes and what they do, and then I'll leave it up to you to do some of your own personal exploring. A combination of just the action, event, and condition nodes listed in this tutorial will be a great starting point for your scripting endeavors.



Events

Remote Event:

Remote Event – As listed in the action nodes section, the Remote Event has the ability to receive the output from another series of scripts.

Level Loaded and Visible:

Level Loaded and Visible – An event that indicates once everything in the virtual world is loaded and rendering. With this information confirmed, you can then fire an event to take place as a result.

Misc:

Console Event – Great for testing and debugging, you can tell something to occur within the console using this event and its output.

Pawn:

Death – Simply a great tool to indicate when an AI pawn has died within the level. From this, you can tell an additional event to occur using the event as a trigger.

Actor:

Take Damage – Indicates when a player, ai character, or some other actor within the world takes damage.

Comparisons

Comparisons can be found under “New Conditions” in the right-click menu. These are great to return simple math comparisons or even basic true/false comparisons. Below are some common Comparison nodes.

Comparison:

Compare Bool – Has a simple true or false output.

Compare Int – Slightly more complex than Compare Bool. The compare int allows you to check $A \leq B$, $A > B$, $A == B$, $A < B$, and $A \geq B$.

Int Counter – Very similar to the Compare Int, however, the Int Counter features an Increment Amount within its properties, similar to the Switch action node mentioned earlier.

Variables

Variables can be entered into your script by selecting “New Variable” under the right-click menu. Below is a list (and the associated colors) for some important variables.



Purple – Object Variables (can be assigned to objects within your Browser/Level)

Float (Blue) – Numbers featuring decimal places.

Bool (Red) – True / False Variable

Int (Light Blue) – Any integer values.

String (Green) – For string variables.

*Note that yellow simply means that you have the node selected.

In a later tutorial, we will cover the relationships between these nodes and how we can get them to communicate with each other and actors within our game world.