IGN Events and Functions Library
Igneous01

Contents

Installation

General

Extract the IGN_EH file somewhere in your Arma 3 editing folder. This will create a new folder called 'IGN Events' with 3 different folders inside.

Each folder is a specific way of installing/using this library in your game. These are:

- **Scripted (Header Only)**
  - folder can be copy/pasted into your mission – only need to call the initialization from init.sqf
  - Currently does not support debugging

- **cfgFunctions**
  - folder can be copy/pasted into your mission – need to include IGN_EH_Definition.hpp into your description.ext
  - All Functions will be available through the in-game functions viewer
  - Currently does not support debugging

- **Addon (WIP)**

Install for Scripted (Header) version

- copy the 'IGN Events\Scripted\IGN_LIB' folder into your mission
- put this into your init.sqf file:
  call compile preprocessfilelinenumbers "IGN_EH\IGN_EH_INIT.sqf";

Install for cfgFunctions version

- copy the 'IGN Events and Functions\CfgFunctions\IGN_LIB' folder into your mission
- put this into your description.ext file:
  #include "IGN_EH\IGN_EH_Definition.hpp"
  Note that you may need to re-save your mission in-game for the functions to be compiled.

Install for Addon version

- Currently in the works

Events

Events and Event Driven Programming

"*An event is a message sent by an object to signal the occurrence of an action. The action could be caused by user interaction, such as a mouse click, or it could be triggered by some other program logic. The object that raises the event is called the event sender. The object that captures the event and responds to it is called the event receiver.*" - MSDN - Events and Delegates for .NET

An Event is basically a kind of sender. It takes a message, and when its time, it will 'send' this message to anyone who is listening. That's all there is to it – ofcourse events can come in a variety of flavors that make them easier or more robust to work with. But that's all just overhead.

Any listeners that 'receive' the message will do something with it. This could be anything - ranging from complex tasks such as interpreting key presses and changing a windows state (GUI programming) to simple logging for debugging purposes. In short, the listener can do whatever he wants when the message has been received (including ignoring it).

I'm sure some of you are wondering "what does all this have to do with scripting?"

Well, we can use this idea (or paradigm) of messages and signals to help simplify our code.

Problem with handling mission flow

For example, lets say we have a mission with a lot of tasks. These tasks have to be completed in sequence before we can move on to the next one. We could create all our tasks in one file, but then all the tasks would show up on the players screen. "No problem, I'll just make some triggers and synch everything together", then you end up spending a few hours confused because your 'wiring' becomes a mess with more triggers to make sure that *these* triggers dont activate before *these other* ones.

You know, like having to write in the condition of a trigger:

```
myCond && (!taskCompleted thisTask) && (taskCompleted thisOtherTask) && etc...
```

"OK, manually script and manage each task by hand" you say, but that is pretty cumbersome too. Then you have to spend more time setting up the mission because everything has to be done by hand. (create each trigger, set the code, declare some bools, check for conditions, etc..) And even still, you have to resort to referencing local variables outside the script scope like in triggers and killed eventHandlers...

Truth be told, handling these kinds of things is not what plain sqf/arma is good at. You as a scripter aren't interested in the 'when' or 'how' something happens, just as long as you get what happened. This is not that different to the needs of GUI programmers – they don't care when or how the mouse was clicked, just as long as it was clicked on their button.

You don't know when that chopper will land, you dont care when it lands or how it lands – so long as you get the message... oh... derp...

Events in Arma 3

Currently Arma 3 only supports a very limited number of events (Although CBA does try to make up for this). It covers some of the essentials like when a unit is killed, but it doesn't cover everyones needs. It's unfortunate that BI hasn't allowed us to define our own custom events, but thats why you're here reading this.

Events in Arma are pretty important for a number of reasons:

- You don't have to manage when something will happen
- You're instantly notified when something happens
- Events check by frame with decent performance
- It abstracts the workload – you don't need to care about that stuff, the engine does it for you

But also equally important – events make your life as a mission maker EASIER

Common Event Uses in Arma 3

You as a mission designer, want to know...

- when a helicopter lands
- when a player has connected
- when a task has been created
- when a unit/object has been destroyed
- when a group is dead
- if all surviving members are inside a vehicle
- when enemies have detected the player
- when a unit is out of ammo
- when the state of a task has changed
- when something is in range
- when the server does something that effects all clients

These are signals – you want the game to signal to you when one of these things happens, so you can respond to it. Hopefully with this guide you're life should be much easier, so you can focus on what you want from a mission, rather than managing when you want it.