Simlogical and the Dead End Nights

The Sims 3 => TS3 Tutorials and Reference => Gameplay Tuts and Refs => Topic started by: Nona Mena on 15 December 2012, 20:46:35

Title: **How to inject an interaction with a pure script mod**
Post by: **Nona Mena** on **15 December 2012, 20:46:35**

*** Read more of this thread in http://www.simlogical.com/ContentUploadsRemote/uploads/1596/How_inject_interactio_with_pure_script_mod.pd ***

Injecting an interaction with a pure script mod is a lot easier than you might think! There are only three parts you absolutely must have to make an interaction injection mod:

1. The interaction itself. This looks the same as any other interaction.
2. The Instantiator class. You'll need this to load the interaction into the game and apply it to game objects/sims/terrain.
3. The Instantiator XML. This is just an XML file that will be part of your package file. If your interaction requires no tuning, your package file could have only two resources: the S3SA with the script and the instantiator XML.

NOTE: This post assumes you are already familiar with scripting modding in the Sims 3. If you have never made a script mod for The Sims 3, please read the pure scripting modding tutorial at MTS (http://modthesims.info/wiki.php?title=Tutorial:Sims_3_Pure_Scripting_Modding).

Part One - The Interaction
Make an interaction. I'm going to assume you know how to make an interaction already. If you don't, go read the Object Modding (aka Adding Interactions) Tutorial (http://modthesims.info/wiki.php?title=Tutorial:Sims_3_Object_Modding). I also highly recommend Cmar's Tutorial: Adding pie menu options to sims (http://www.modthesims.info/showthread.php?t=491875).

This post is primarily about the instantiator class.

Part Two - The Instantiator
Now it's time to make your Instantiator class. Most of this is covered in the Pure Scripting Modding Tutorial at MTS (http://modthesims.info/wiki.php?title=Tutorial:Sims_3_Pure_Scripting_Modding), but I'll go over it again.

Download: Example Instantiator Class (http://www.den.simlogical.com/ContentUploadsLocal/ijDownloadsServer.php?topic=1596&file=WhatsMyTempInstantiator.zip)

Step One: Give your assembly the [assembly: Tunable] attribute.
Before you go further, go to the AssemblyInfo.cs and make sure you add the following (if you haven't already):

Code: [Select]

```
using Sims3.SimIFace;

[assembly: Tunable]
```

It is extremely important that you add these lines to your AssemblyInfo.cs, as without them, you cannot load your script into the game. Actually, you could put the [assembly: Tunable] attribute in your Instantiator cs but you might as well just get into the habit of adding it to your AssemblyInfo. It should look something like this:

Code: [Select]

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using Sims3.SimIFace;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: Tunable]
[assembly: AssemblyTitle("Nona_WhatsMyTemp")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("Nona_WhatsMyTemp")]
```

```
[assembly: AssemblyCopyright("Copyright ©  2012")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

etc.
```

Step Two: Make an Instantiator.cs (Download example instantiator (http://www.den.simlogical.com/denforum/index.php?topic=1596.0))
Since I'm writing this, we're doing doing it my way. This is how the skeleton of your Instantiator class will look. Feel free to copy & paste this or just use my example instantiator as a sort of template/guideline. Just don't forget to change the namespace!

Code: [Select]

```
using Sims3.Gameplay.Actors;
using Sims3.Gameplay.Autonomy;
using Sims3.Gameplay.EventSystem;
using Sims3.SimIFace;
using System;

namespace NonaMena.WhatsMyTemp.Common
{
    public static class Instantiator
    {
        [Tunable]
        internal static bool kInstantiator = false;

        static Instantiator()
        {
           World.OnWorldLoadFinishedEventHandler += new EventHandler(OnWorldLoadFinished);
        }
        private static void OnWorldLoadFinished(object sender, EventArgs e)
        {
        }
    }
}
```

The tunable kInstantiator is what will get our foot in the door. Then we will use the OnWorldLoadFinished() callback method to get our interaction injected.

Step Three: Put together the OnWorldLoadFinished() callback method.
Interaction injection is easy. All we need to do is tell the game to add our interaction to whatever objects need it. In the case of my What's My Temp? mod, I am adding the interaction to all sims. Put this in your OnWorldLoadFinished() callback method :

Code: [Select]

```
foreach (Sim sim in Sims3.Gameplay.Queries.GetObjects<Sim>())
            {
                AddInteractions(sim);
            }
```

AddInteractions is the method I will use to apply the interactions. You can change the name to whatever you like. However, it should look like this:

Code: [Select]

```
private static void AddInteractions(Sim sim)
        {
            foreach (InteractionObjectPair pair in sim.Interactions)
            {
                if (pair.InteractionDefinition.GetType() == CheckMyTemp.Singleton.GetType())
                {
                    return;
                }
            }
            sim.AddInteraction(CheckMyTemp.Singleton);
        }
```

Like this, your instantiator will apply your interaction to all sims in the game when the game is loaded. But what about sims that get added to the game during gameplay, such as newborns or immigrants?

Next: Step Three: Using the EventTracker to apply interactions to newly instantiated sims.

Title: **Re: How to inject in an interaction with a pure script mod**
Post by: **Nona Mena** on **15 December 2012, 21:56:03**

So far we have:

- Created the static kInstantiator tunable which will allow the game to load our script.
- Written our OnWorldLoadFinished() callback method.
- Applied our new interaction to all sims in the game when the game is loaded.

We're almost done!

Step Four: Using the EventTracker to apply interactions to newly instantiated sims.
In order to apply interactions to new sims, we need an EventListener. Add this line to your OnWorldLoadFinished() callback method.

Code: [Select]

```
EventTracker.AddListener(EventTypeId.kSimInstantiated, new
ProcessEventDelegate(Instantiator.OnSimInstantiated));
```

Your delegate method should look something like this:

Code: [Select]

```
private static ListenerAction OnSimInstantiated(Event e)
        {
            try
            {
                Sim sim = e.TargetObject as Sim;
                if (sim != null)
                {
                    Instantiator.AddInteractions(sim);
                }
            }
            catch (Exception)
            {
            }
            return ListenerAction.Keep;
        }
```

Now our interaction will be applied whenever a new sim is added to the game. This means users won't have to save their game and re-load if they want to use our interaction on new sims.

And guess what? We're done with our Instantiator class! Just like that we have created an Instantiator that will apply our new interaction to all sims. Walk in the park, am I right?

Part Three - The Instantiator XML
Your Instantiator XML is almost as important as assigning your assembly the [assembly: Tunable] attribute. Luckily, it's easy to create. If you haven't already, open S3PE and click on the Resource menu. Choose Add.

The Type should be _XML
The Group: 0
IMPORTANT: Then go to the Name field and enter the Namespace+Class of your Instantiator class. In the example Instantiator class I provided, this is: NonaMena.WhatsMyTemp.Common.Instantiator
Press the FNV64 button and then press OK.

Now, select the new XML file, right-click and choose Text Editor/Notepad. Paste the following in:
Code: [Select]

```
<?xml version="1.0" encoding="utf-8"?>
<base>
  <Current_Tuning>
    <kInstantiator value="True" />
  </Current_Tuning>
</base>
```

And save the changes. Ta-da! Your Instantiator _XML is done!

DOWNLOAD: Instantiator.xml
(http://www.den.simlogical.com/ContentUploadsLocal/ijDownloadsServer.php?topic=1596&file=Instantiator.xml.zip)

Part Three - Putting the Package together.

Your package must have at least two resources: the S3SA resource with the compiled .dll from your mod, and the Instantiator XML. If you don't know how to add an S3SA resource, see the Pure Scripting Modding Tutorial at MTS (http://modthesims.info/wiki.php?title=Tutorial:Sims_3_Pure_Scripting_Modding).

Title: **Re: How to inject in an interaction with a pure script mod**
Post by: **Nona Mena** on **15 December 2012, 22:05:05**

Applying Interactions to Game Objects
Sometimes you might want to add an interaction to an object and not a sim. You can use the same concepts in this post to apply interactions to other objects in the game as well. Let's use my Meditate by Candlelight (http://www.den.simlogical.com/denforum/index.php?topic=1582) mod as an example.

In my Meditate by Candlelight mod, my OnWorldLoadFinished() callback method looks like this:

Code: [Select]
```
private static void OnWorldLoadFinished(object sender, EventArgs e)
        {
            foreach (Candle candle in Sims3.Gameplay.Queries.GetObjects<Candle>())
            {
                Instantiator.AddInteractions(candle);
            }
            EventTracker.AddListener(EventTypeId.kBoughtObject, new
ProcessEventDelegate(Instantiator.OnObjectChanged));
            EventTracker.AddListener(EventTypeId.kInventoryObjectAdded, new
ProcessEventDelegate(Instantiator.OnObjectChanged));
            EventTracker.AddListener(EventTypeId.kObjectStateChanged, new
ProcessEventDelegate(Instantiator.OnObjectChanged));
        }
```

Notice that the foreach statement looks pretty much the same, just with different object names. However, I did need to use different EventTrackers to make sure my interactions were always applied. The foreach statement will apply interactions to all objects in the game at loading time, but you're going to need to apply interactions whenever a new object is purchased. For objects that don't go in your sim's personal inventory, it should be enough to use only the kBoughtObject Event. The kInventoryObjectAdded and kObjectStateChanged events are necessary for inventory objects that might travel with sims (Thanks for Twallan & MDM for this knowledge (http://www.den.simlogical.com/denforum/index.php?topic=1166.msg8092#msg8092)).

Here are my AddInteractions() and OnObjectChanged() methods:

Code: [Select]
```
private static void AddInteractions(Candle candle)
        {
            foreach (InteractionObjectPair pair in candle.Interactions)
            {
                if (pair.InteractionDefinition.GetType() == MeditativeCandle.Meditate.Singleton.GetType())
                {
                    return;
                }
            }
            candle.AddInteraction(MeditativeCandle.Meditate.Singleton);
        }
        private static ListenerAction OnObjectChanged(Event e)
        {
            try
            {
                Candle candle = e.TargetObject as Candle;
                if (candle != null)
                {
                    Instantiator.AddInteractions(candle);
                }
            }
            catch (Exception)
            {
            }
            return ListenerAction.Keep;
        }
```

DOWNLOAD: You can take a look at my full instantiator class for my Meditate by Candlelight mod. (http://www.den.simlogical.com/ContentUploadsLocal/ijDownloadsServer.php?topic=1596&file=MeditativeCandleInstantiator.zip)

Applying Interactions to Terrain
Applying terrain interactions (see Different Kinds of Interactions, below) is pretty much exactly the same concept as applying interactions to other objects. I haven't made a terrain interaction yet, but Consort was kind enough to allow me to share his own code for it. Thanks, Consort!. For terrain interactions, your OnWorldLoadFinished() method would like this

Code: [Select]
```
private static void OnWorldLoadFinished()
```

```
		{
			Terrain[] objects = Sims3.Gameplay.Queries.GetObjects<Terrain>();
			for (int i = 0; i < objects.Length; i++)
			{
				Terrain terrain = objects[i];
				terrain.AddInteraction(ConsortModSimFixer.TerrainClick.Singleton);
			}
		}
```

Notice that in this case, Consort has added the interaction with `terrain.AddInteraction(ConsortModSimFixer.TerrainClick.Singleton);` without using another method.

Loading Interactions With Delays
If you have trouble with your interactions while loading, you can delay applying them until after the world has already loaded. To do this, instead of adding your interactions in OnWorldLoadFinished(), place only an alarm (examples provided by Consort):

Code: [Select]

```
private static void OnWorldLoadFinished(object sender, EventArgs e)
		{
			AlarmManager.Global.AddAlarm(2f, TimeUnit.Minutes, new
AlarmTimerCallback(ConsortModSimFixer.OnWorldLoadDelay), "Consort was here", AlarmType.NeverPersisted, null);
		}
```

Remember, you can name your Alarm and your AlarmTimerCallback whatever you want. Then, add the interactions in OnWorldLoadDelay() -- or whatever you name your method -- the same way you would in OnWorldLoadFinished().

Title: **Re: How to inject in an interaction with a pure script mod**
Post by: **Nona Mena** on **15 December 2012, 22:32:12**

Hiding and Replacing EA Interactions

Hiding EA Interactions
In general, if you want to replace an EA interaction, your best bet is to simply hide the EA Interaction and allow the user to see only your version. You can do this with the RemoveInteractionByType() overload method:

`RemoveInteractionByType(Type interactionDefinitionType)`

For example:
Code: [Select]

```
base.RemoveInteractionByType(StaircaseSpiral.Use.Singleton);
```

Don't forget, you may need to replace the `base` with something that would fit in the context of your code.

Replacing EA Interactions
Sometimes it is simply not feasible, or it is entirely inconvenient, to hide an EA interaction. In that case, your next option is to actually replace EA's interaction definition with your own. **Important:** When you do this, you create the possibility that your mod may conflict with another script mod that also replace's EA interaction definition.

Replacing an interaction definition is really easy. In fact, it's even easier than applying interactions to sim or objects. If you're only replacing one interaction definition, your instantiator will be really short! Here's an example:

Code: [Select]

```
public static class Instantiator
    {
        [Tunable]
        internal static bool kInstantiator = false;

        static Instantiator()
        {
            World.OnWorldLoadFinishedEventHandler += new EventHandler(Instantiator.OnWorldLoaded);
        }
        public static void OnWorldLoaded()
        {
            CommonDoor.LockDoor.Singleton = new NonaLockDoor.LockDoor.Definition();
        }
    }
```

Keep in mind: When you replace an interaction definition, you **must** create a new ITUN resource for the interaction. If your interaction does not have an ITUN resource, sims will not be able use the interaction autonomously, and they will not gain motives when using the interaction. Some interactions simply do not work properly without an ITUN, even if you don't want it to be used autonomously, such as computer interactions. These interactions may have posture preconditions, or other requirements that the ITUN resource covers. Some interactions do not require tuning. These interactions usually have the [DoesntRequireTuning] attribute.

You can also force the game to use EA's tuning by injecting EA's tuning into your own interaction. Both Twallan and Buzzler have tuning injection mods. My Cow Plant Eat Cake interaction mod uses Twallan's method of tuning injections, so feel free to take a look at that. I recommend NRaas Shooless (http://nraas.wikispaces.com/Shooless) if you want to see twallan's code.

Different Kinds of Interactions
Although this post is less about creating interactions and more about loading them into the game, Consort suggested we mention some of the differen types of interactions available in the game. There are many kinds of interactions in the game. These include, but may not be limited to:

- SocialInteraction
- MountedInteraction
- MountedTerrainInteraction
- ImmediateInteraction
- TerrainInteraction

Similarly, useful InteractionDefitions include (but may not be limited to):

- InteractionDefinition
- ActorlessInteractionDefinition
- ImmediateInteractionDefinition
- ImmediateInteractionGameObjectHit
- SoloSimInteractionDefinition

Title: **Re: How to inject in an interaction with a pure script mod**
Post by: **Nona Mena** on **15 December 2012, 23:04:58**

Additional References

Tutorial:Sims 3 Pure Scripting Modding (http://modthesims.info/wiki.php?title=Tutorial:Sims_3_Pure_Scripting_Modding)
Tutorial:Sims 3 Object Modding (http://modthesims.info/wiki.php?title=Tutorial:Sims_3_Object_Modding)
Tutorial:Sims 3 In Depth Scripting (http://simswiki.info/wiki.php?title=Tutorial:Sims_3_In_Depth_Scripting)

Tutorial: Adding pie menu options to sims (http://www.modthesims.info/showthread.php?t=491875)

Game Runtime Events (http://www.den.simlogical.com/denforum/index.php?topic=1166.0)

How to Create and Load Custom Moodlets with a Pure Script Mod (http://www.den.simlogical.com/denforum/index.php?topic=1063.0)

Additional Credits
None of the information I have provided here is brand new. In fact, much of it is explained in the Pure Scripting Modding tutorial at MTS. I could not have shared this information without that tutorial.

The following people have been indispensable in the creation of this post. Without them, there would be no post.
Buzzler
Twallan
MDM
Consort
Cherry
Inge Jones
Cmo

I must also give credit to the various other script modders out there, who have provided their knowledge via forums, tutorials, PMs and sharing their own mods: velocitygrass, misukisu, cmarNYC,

_ani, treeag, SimsMX. I hope I didn't forgot anyone!

One more thing...
I wrote this up because of a couple of people in Simlogical chat requested it, and I had been putting it off. However, it wasn't until I was actually writing my "Additional Credits" and everything was pretty much finished that Consort pointed out Cmar's newest tutorial (Tutorial: Adding pie menu options to sims (http://www.modthesims.info/showthread.php?t=491875)). Oh well, there can never be enough tutorials! Plus, hers is way more in depth and takes a slightly different angle.

Title: **Re: How to inject in an interaction with a pure script mod**
Post by: **Nona Mena** on **15 December 2012, 23:33:02**

Reserved.

Title: **Re: How to inject in an interaction with a pure script mod**
Post by: **Nona Mena** on **15 December 2012, 23:47:50**

Please feel free to share comments, questions, suggestions, criticisms, and corrections :)

Title: **Re: How to inject in an interaction with a pure script mod**
Post by: **cmo** on **16 December 2012, 03:41:08**

Sweet, ty. :)

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **JT** on **06 February 2013, 11:47:05**

This has been immensely useful for me so far, and is quite possibly one of the most concise tutorials available.

One question: is there an EventTypeId that triggers when an object is created, rather than bought, transferred into the inventory, or "state changed"? Or would state changed cover those instances?

Specifically, I'm looking to hook interactions into objects that have been "created outside of the world", as the game likes to call it, then moved somewhere onto the ground.

One way or another I imagine such a hook would be quite expensive, but the only other means I can think of hooking into a spawned object is to continuously poll until the total count of the object changes, which is probably just as expensive if not more so, especially if time criticality of the interaction appearing is important!

I'd really, really prefer to avoid the OBJK method if I can otherwise help it, but I think that'll be the only practical solution. Fortunately, since the object is spawned at runtime by its medator anyway, this means that I'm guaranteed to get a new one of that type.

If only EA had given us an array of interactions that would be shared across all objects of the same type, instead of a dynamic array for each individual instance...

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Nona Mena** on **06 February 2013, 16:51:11**

I think kObjectStateChanged and kInventoryObjectAdded would work fine, so I would say try those and post back with the results. I went through something similar when I made my poisoned apple mod, and I decided that OBJK override was better in my case. But it really depends on you want to do.

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Consort** on **06 February 2013, 18:44:56**

If you work with your own custom object instead of any pre-existing object in the game you can simply override the typical GameObject methods for your object.

It would look something like this:
Code: [Select]

```
public override void OnHandToolPlacementOnTerrain()
{
```

```
    //your code
    base.OnHandToolPlacementOnTerrain();
}
```

Of course you can't inject this code into any object, it only works for your home cooked custom objects.

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **JT** on **07 February 2013, 03:48:13**

OnStartup() is what I'm using, with an OBJK override to target my own script class that inherits from the base class.

This is because I'm trying to inject additional interactions into the Newspaper, but EA didn't design this to be readily accessible outside of their designers (of course, why would they?):

1) Hooking into kBoughtObject and kInventoryObjectAdded works only if it is bought from a rabbithole. The paperboy/papergirl don't actually carry the object -- they simulate carrying the object and then spawn a new copy on the ground that, as far as the game logic is concerned, has never actually touched an inventory. Technically, the interactions should appear after you pick it up for the first time, but that doesn't give you the ability to use the interactions directly from the newspaper on the ground like you can with any other default newspaper interaction.

2) Hooking into kObjectStateChanged sadly doesn't do a thing. Apparently the state only changes after an object exists, and the scenarios where a kObjectStateChanged event is triggered are not evident (I think I might go ahead and decompile the entire Sims3GameplayObjects just so I can document all of the less-clear event listeners by their actual behaviour in searchable plain text inside the function bodies, since ILSpy only searches for members, classes, and types, not "full text").

3) Creating an OBJK override will work well, but will make it incompatible with any other mod that wants to inject interactions into newspapers.

What I'm leaning towards, based on these discoveries, is actually chopping out my newspaper derived class as a separate library. It would simply contain a class with a publicly-accessible static array of interaction singletons that will be automatically added to every newspaper of the derived class when they are created. Then my mod would use it as a library reference, and if anyone wants to design inter-compatible newspaper interactions they need only reference the same library and add their interaction singletons to the same array. This also gives me a few other ideas to make it more appealing even as a standalone mod in general.

Of course, I don't foresee a lot of people releasing newspaper interaction mods anyway, especially now that the product lifecycle is drawing to a close, so I'm wondering if I should just go ahead and get more sleep by not worrying about it. Pair that with some people's slants (twallan's and Pescado's for instance) that they will never use other people's mods in their own and the appeal factor drops a bit more. ;-)

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **JT** on **07 February 2013, 04:40:03**

...And just now, I just realise -- why don't I just replace the DeliverNewspaper singleton? =P

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Kuree** on **09 February 2013, 01:38:42**

If you just want to *add some interactions to newspaper, I think you don't need to use a objk override. It's better to use a pure-script to solve this problem. In your class name, you can add something like this to solve this problem. XXXX:Sims3.Gameplay.Objects.Miscellaneous.Newspaper (XXX is your class name)

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **JT** on **09 February 2013, 02:38:18**

I'm a bit confused by your post: you say at first that I don't need an OBJK override, but then suggest creating a new class. If I'm creating a new class, the only way to get it into the game is either to make an OBJK override (the easy option) or to replace all interactions that could ever spawn the old class with ones that spawn my own class (the hard option), and either method would not be cross-compatible with other modders' efforts.

Anyway, further discussion should be sent over to the new thread (http://www.den.simlogical.com/denforum/index.php/topic,1772.msg11335.html#msg11335) so we can keep this thread relevant to the tutorial itself. =)

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Inge** on **09 August 2013, 17:21:58**

Trying to replace an EA interaction with my version of it, I get this reply my my instantiator:
Quote

> 'Sims3.SimIFace.World' does not contain a definition for 'OnWorldLoadFinishedEventHandler'

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Sims MX** on **09 August 2013, 18:23:54**

You have to use "sOnWorldLoadFinishedEventHandler" ;)

Or:

Quote from: Twallan

> Peter's program works in most cases, however, the delegates in UI do not like being unprotected in that manner.
>
> You will need to alter them manually, and set them back to "public" access. :)

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Inge** on **09 August 2013, 18:41:31**

Thanks. Also I was using example for replacing EA interaction, and in that example Nona had not included the object sender, EventArgs e bit so I got complaint about overloads not matching, but I sorted that out by referring to one of her other examples.

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Inge** on **10 August 2013, 10:46:01**

Can you tell me anything about how to "alter them manually" like Twallan says - or else how come I am the only modder who seems unable to use OnWorldLoadFinishedEventHandler in my code? What is everyone else doing different?

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Nona Mena** on **10 August 2013, 15:08:13**

I don't unprotect SimIFace because I haven't had a need to, which is why I don't use sOnWorldLoadFinishedEventHandler. If you don't unprotect SimIFace, you shouldn't need sOnWorldLoadFinishedEventHandler.

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Inge** on **10 August 2013, 15:17:28**

Just for completeness I will add what I already told you in chat. Peter has fixed unprotect so it should deal the the event handlers correctly, and will presumably publish the new version shortly.

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Peter** on **10 August 2013, 17:55:22**

Quote from: Inge on 10 August 2013, 15:17:28

> Just for completeness I will add what I already told you in chat. Peter has fixed unprotect so it should deal the the event handlers correctly, and will presumably publish the new version shortly.

Ooh, it worked?

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Inge** on **10 August 2013, 19:37:06**

Yes but my mod is still not working. Or at least it works *erratically*. It's like sometimes it uses my

version and sometimes EA's and it's apparently random.

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Nona Mena** on **19 August 2013, 21:36:12**

Did you get this fixed? Sorry, I've just been rather busy and stressed lately.

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Inge** on **19 August 2013, 21:42:45**

Reassignment of the singleton/definition didn't work well so in the end I decided to hide and add the interaction per sim.

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Zerbu** on **14 November 2013, 23:00:35**

Great tutorial ;), I only have one question: what do you do with your OnObjectChanged method if you have multiple interactions for different object types? The code used in the example looks like it's only made if you're only adding interactions to one type of object.

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Nona Mena** on **15 November 2013, 07:12:30**

There are certainly more, elegant/fancier ways to handle such a situation, but you can also just do it like this:

Code: [Select]

```
private static ListenerAction OnObjectChanged(Event e)
        {
            try
            {
                Candle candle = e.TargetObject as Candle;
                IncenseHolder incenseHolder = e.TargetObject as IncenseHolder;
                if (candle != null)
                {
                    Instantiator.AddInteractions(candle);
                }
                if (incenseHolder != null)
                {
                    Instantiator.AddInteractions(incenseHolder);
                }
            }
            catch (Exception)
            {
            }
            return ListenerAction.Keep;
        }
```

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Zerbu** on **16 November 2013, 04:16:06**

Quote from: Nona Mena on 15 November 2013, 07:12:30

> There are certainly more, elegant/fancier ways to handle such a situation, but you can also just do it like this:
>
> Code: [Select]
>
> ```
> private static ListenerAction OnObjectChanged(Event e)
>         {
>             try
>             {
>                 Candle candle = e.TargetObject as Candle;
>                 IncenseHolder incenseHolder = e.TargetObject as IncenseHolder;
>                 if (candle != null)
>                 {
>                     Instantiator.AddInteractions(candle);
>                 }
>                 if (incenseHolder != null)
>                 {
>                     Instantiator.AddInteractions(incenseHolder);
>                 }
>             }
>             catch (Exception)
>             {
>             }
>             return ListenerAction.Keep;
> ```

```
        }
```

Thanks, that seems to work as intended (except in my case I used different "AddInteractions" functions for each object, but it still worked). :D

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Nona Mena** on **16 November 2013, 15:39:41**

Yep, my AddInteractions() is an overload method, so there is AddInteractions(Candle candle) {} and AddInteractions (IncenseHolder incensholder){}. But it doesn't matter how you make the methods :)

Glad it's working!

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **icarus_allsorts** on **10 January 2014, 17:30:28**

Hi Nona, thank you so much for this tutorial, with it I've managed to make an actual script mod that works! (for now anyways...)

I had a small problem when using your instructions on Applying Interactions to Game Objects though: the interaction would successfully be added to the object on loadup and when bought in regular Build/Buy mode but NOT when the object is placed on a lot during Edit Town. I tried it with you Candle Meditation mod as well and found that the meditate interaction also wouldn't appear on candles placed on community lots in Edit Town. No biggie really since the interaction would still be applied once I save and loaded the game again, but I thought I should point it out.

In the end I tried using the method suggested by Sims MX in this thread:
http://www.den.simlogical.com/denforum/index.php/topic,1772.msg11335.html#msg11335 (Reply #6)
to apply my interaction regardless of whether the object was bought during regular gameplay or Edit Town

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Nona Mena** on **11 January 2014, 04:50:02**

Thanks for sharing this, icarus. I'll have to add this information to the tutorial when I have some time (and give you and SimsMX some credit). I'll probably update my candle mod too, eventually. First I need to get caught up after my long trip though :)

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **JunJayMdM** on **29 March 2014, 11:39:06**

I made some new interesting discoveries as I was updating my mods with a new injection system. The reason I added kObjectStateChanged and kInventoryObjectAdded events was originally intended for adding interactions to inventory objects when traveling. At the time I didn't know any better so I took twallan's advice and didn't ask myself why interactions were not being added normally.
So, the reason is that objects in the world aren't instantiated as soon as the world has finished loading, especially when going on vacation or loading really big worlds. That means if the code tries to add interactions before everything is instantiated, it won't find any object available and no surprise interactions won't be there.

How to solve this?

The key is to start adding interactions after the instantiation and I found that the kEventSimSelected event is the perfect time. So one could do this :

Code: [Select]

```
public void OnWorldLoadFinished()
{
    if (PlumbBob.SelectedActor != null) // I am checking if the Active Sim has been instantiated already, in which case the listener will be useless
    {
        this.InitInjection();
        return;
    }
    EventTracker.AddListener(EventTypeId.kEventSimSelected, new ProcessEventDelegate(this.OnSimSelected));
}

public ListenerAction OnSimSelected(Event e)
```

```
{
    this.InitInjection();
    return ListenerAction.Remove; // This will remove the listener, because we won't need it anymore
}

public void InitInjection()
{
    // Add other listeners, events and inject interactions (even delayed, if it's still a bit too early for
your needs)
}
```

Using this pattern I was able to get rid of the kObjectStateChanged and kInventoryObjectAdded events, and I'm having no problems with my mods, all interactions are being added properly everytime.
This method is also useful when you want to do something else with objects, aside from adding interactions.

I hope this was of help and can be added to your nice tutorial ;)

EDIT : Alternative Method

The Active Sim is a good trick to find out when the game has finished instantiating, however you might think that it also means there's an Active Household and it must be a saved game. WRONG!

I'm not going into details about why the game is forcing an Active Sim if it's a new game, because I didn't understand it clearly myself, however there's a trick you can use to stop the injection from happening on new games and only occur when a Household is made Active :

Code: [Select]

```
public void OnWorldLoadFinished()
{
    if (Household.ActiveHousehold != null) // I am checking if the Household is Active, this time, so we know
the Active Sim is the real deal
    {
        this.InitInjection();
        return;
    }
    EventTracker.AddListener(EventTypeId.kEventSimSelected, new ProcessEventDelegate(this.OnSimSelected));
}

public ListenerAction OnSimSelected(Event e)
{
    if (Household.ActiveHousehold != null) // Same as the method above
    {
        this.InitInjection();
        return ListenerAction.Remove;
    }
    return ListenerAction.Keep; // This will keep the listener active until the real Active Sim is selected
}
```

That's basically it, at this point the game has not yet transitioned to Live Mode, which means interactions will be available as soon as the UI appears with your Household.
This is some kind of "delayed" injection, but not as delayed as alarms, since alarms start triggering only when the GameFlow is active, because they use the SimClock, hence it provides a good checkpoint for your injection ;)

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **Nona Mena** on **29 March 2014, 12:08:41**

Thanks very much for your information, JunJayMdM. I shall add it to the tutorial, along with icarus' as soon as I can whip my lazy (but also busy) butt in gear. I hope it's soon. This thing could use a face lift.

Title: **Re: How to inject an interaction with a pure script mod**
Post by: **JunJayMdM** on **29 March 2014, 12:21:03**

Quote from: Nona Mena on 29 March 2014, 12:08:41

> Thanks very much for your information, JunJayMdM. I shall add it to the tutorial, along with icarus' as soon as I can whip my lazy (but also busy) butt in gear. I hope it's soon. This thing could use a face lift.

You're welcome :)