TIMELINE INTEGRATION

View online: https://www.construct.net/en/make-games/manuals/addon-sdk/guide/timeline-integration

Adding timeline support to a 3rd party addon, be it a plugin, behavior or effect is quite easy. A little bit of extra work is needed though, here is how to do it.

Plugins

1. Set the interpolatable plugin property option to true in all the plugin properties which should be supported by timelines.
2. Implement the GetPropertyValueByIndex(index) method in the plugin instance class.

   index argument

   Refers to the index of each property in the plugin as they are given to the constructor of the plugin instance class.

   return value

   The current value associated with the passed in index. Depending on the current implementation this could be as easy as returning an existing variable.

   ***Colors**: color properties should be returned as an array of 3 values. If the internal representation used by the plugin is different, the conversion needs to be made before returning the color.*

   ***Angles**: if a plugin uses a value as an angle, but it is not specifically defined as an angle in the plugin definition, this method needs to make sure the value is in the same format as it is shown in the editor before returning it. **Ex.** A plugin internally converting a property from degrees to radians, should make sure the value returned by **GetPropertyValueByIndex** is in degrees.*

3. Implement the SetPropertyValueByIndex(index, value) method in the plugin instance class.

   index argument

   Refers to the index of each property in the plugin as they are given to the constructor of the plugin instance class.

   value argument

The new value that needs to be applied to the specified property. The passed in value is absolute so it should be applied directly with the = operator.

return value

No return value is required.

Depending on the implementation, additional work might be needed when a property changes. ***Ex.*** *If the plugin relies on any sort of caching relating to a property, changing that property*
would require an update to the cache in order for everything to continue working properly as the plugin is modified by a timeline.

Colors*: these values are received as arrays of 3 values. The values should be applied according to the plugin's internal representation.*

Angles*: if a plugin uses a value as an angle, but it is not specifically defined as angle in the plugin definition. This method needs to make sure the incoming value is using the same format as the internal representation before making the assignment.* ***Ex.*** *A plugin internally converting a property from degrees to radians, will need to convert the incoming value of* ***SetPropertyValueByIndex*** *from degrees into radians.*

Plugin that need layout view preview updates

Some plugins might need to update the layout view to give a preview of the changes they are making. In this case a few more methods needs to be implemented so the plugin can update it's internal state when a timeline starts preview and when it stops preview.

1. Implement the OnTimelinePropertyChanged (id, value, detail) method in the plugin instance class.

id argument

The id of the property that is changing.

value argument

The value that is being applied by the timeline.

detail argument

An object with details about the value. It has a **"resultMode"** property with a value of either **"absolute"** or **"relative"**.

> *__"value"__ and __"detail"__ are not needed in the most common use case of just updating the internal state of the plugin.*

> *This method is similar to __OnPropertyChanged(id, value)__ and in most cases can be implemented in similar fashion.*

> *In this function the plugin needs to update the corresponding internal state, namely using the new __GetTimelinePropertyValue(id)__ method from __IObjectInstance__ which gets the value of a property with any changes a timeline might be applying to it. After the corresponding internal state is updated, refresh the layout view to view the changes.*

2. implement OnExitTimelineEditMode (). This method is called when timeline edit mode is turned off. In this method the plugin's internal state should be updated again so any timeline changes from the preview are reset. Using **GetPropertyValue(id)** to get values without timeline changes, applying those to the relevant internal variables of the plugin and refreshing the layout view should be enough.

Behaviors

See plugin integration above, all steps apply.

Effects

Edit the effect's .json file and add the interpolatable property with a value of true to each paramenter definition which should be supported by timelines. No additional modifications needed.

> ***NOTE**: Remember that not all properties need to be supported, if it looks like it doesn't make sense for a property to receive dynamic updates, it is ok to not support it.*