A Programmer's Tour to Pause Controller

Intro:

Pause Controller API gives developers access to pausing and unpausing individual GameObject. There's a few key methods and class to note while developing code with the Pause Controller API.

Legend:

O*range text represents methods and verbs.*
Blue text represents classes, instances or variables.

Key Classes in the PauseController Suite:

Class	***Description***
`PauseController`	This is a MonoBehaviour that manages all objects that are to be *paused*.
`PlayMakerFsmPause`	This MonoBehaviour requires a PauseController be attached to the GameObject
`IPauseProtocolInterpreter`	This interface manages *pausing* of a particular component.
`PausingListing`	This object contains a list of IPauseProtocolInterpreter and organizes *pausing*, *pause-blocking*, or *resuming* the interpreted components. This object is depended on the PauseController that *returned* it as well. So if that PauseController is deleted. PausingListing will not work.
`DummyPauseInterpreter`	This object contains does nothing when it's methods is called unless there's a simple value expected in the return.

Key Methods for Pause Controller Suite:

Method	***Description***
`PauseController.ActivatePauseProtocol`	This invokes *pausing* to occur with the current filter settings.
`PauseController.DeactivatePauseProtocol`	This invokes *unpausing* to occur with the current filter settings.

PauseController.GetFilteredPauseProtocols	*Returns* a PausingListing that allows *pausing* on a individual bases.
PauseController.GetPauseIntepreters	*Returns* List< IPauseProtocolInterpreter > associated with a collection of components.
PauseController.GetPauseIntepreter	This *returns* a IPauseProtocolInterpreter for the component.
PauseController. InsertPauseInterpreterSuccessful	This method allows a class deriving from IPauseProtocolInterpreter to be assigned to "interpreter" a target type that is passed along with this method. An useful example is that you can insert custom pause interpreters for a class that needs special maintenance to be pause. (This is how the PlayMakerFsmPause works).
PauseController. InsertTypePauseBlockerSuccessful	This method assigns a DummyPauseInterpreter for this particular type. Thus it will not be paused.
PausingListing.Pause	This will *pause* all components assigned to it.
PausingListing.Unpause	This will *unpause* all paused components .
PauseController.Reset	When overriding this method, but sure you call on the base method within the override.
Pausecontroller.Awake	When overriding this method, but sure you call on the base method within the override.
<YourMonoBebehaviour>.PauseControllerWillPause Declaring example:: void PauseControllerWillPause(PauseMessageParameter message)	This a message that is sent to a monobehaviour
<YourMonoBehaviour>.PauseControllerWillNotPause Declaring example:: void PauseControllerWillPause(PauseMessageParameter message)	
<YourMonoBehaviour>.PauseControllerResuming Declaring example: void PauseControllerResuming(	

`PauseMessageParameter message )`	

Pausing an Individual GameObject:

Pausing an individual GameObject can be achieve simply by calling "*GetFilteredPauseProtocols(GameObject)*" method and getting a PausingListing and calling the *Pause* method.

Pausing Objects under the current filters settings:

Filtering GameObjects in the scene to be pause can be easily down in 2 ways.

Setting the public variable PauseController.activatePause to true will activate a standard pausing. Setting this variable to false after it has been set to true will unpause the currently paused GameObjects.

Another approach is to call the method *`ActivatePauseProtocol()`*. This will also activate standard pausing and set the variable activatePause to true. To unpause the GameObjects, call the method *`DeactivatePauseProtocol()`* and this will also set activatePause to false.

Default PauseController Messages

The "Interface and Manual" file covers Advance Options that contains a feature that invokes methods on a MonoBehaviours during the pausing process.

However, there are default messages that are always sent to a MonoBehaviour. These messages are *PauseControllerWillPause*, *PauseControllerWillNotPause*, and *PauseControllerResuming*. They are called right before disabling or enabling a MonoBehaviour.

They also pass an argument of type `PauseMessageParameter` that contains a reference to the PauseController instance invoking it and the GeneralData that's set through the interface.