**Comparison between GEAR Plugin systems**

|  |  |  |  |
| --- | --- | --- | --- |
| **Old plugin system (v0.0)** | | **New plugin system (v1.0)** | **Changes** |
| **using** System;  **using** System.Collections.Generic;  **using** System.Text;  **using** System.ComponentModel;  **using** System.Drawing;  **using** System.Data;  **using** System.Windows.Forms;  **using** Gear.EmulationCore; | | **using** System;  **using** System.Windows.Forms;  **using** Gear.EmulationCore; | Clean up extra libraries not used, of the default for all plugins.  You should include in your own definition if you wish to use them in your plugins. |
| **namespace** Gear.PluginSupport {  **public** class PluginBase : UserControl { | | **namespace** Gear.PluginSupport {  **public** class PluginBase : UserControl { | - |
|  | | **protected** PropellerCPU Chip; | New - reference added by default for all plugins. The user doesn’t need to add manually anymore, because it is inherited to your plugin.  Note: in v0.0, the class PropellerCPU was name Propeller, so in your plugins you have to consider this. |
| **protected** PluginBase() { } | New - default constructor only for system plugins. Don’t use it. |
| **public** PluginBase(PropellerCPU chip)  { Chip = chip; } | New - constructor for user plugins. In your plugins you must declare as (changing the class name to yours):  class YourClassName : PluginBase  {  public YourClassName(PropellerCPU chip)  : base(chip)  { /\* your initialization code here \*/ } |
| **public** **virtual** **string** Title  { **get** { **return** "Bus Module"; } } | | **public** **virtual** **string** Title  { **get** { **return** "Plugin Base"; } } | Modify – more neutral title. |
| **public** **virtual** Boolean AllowHotKeys  { **get** { **return** **true**; } } | | | - |
| **public** **virtual** Boolean IsClosable  { **get** { **return** **true**; } } | | | - |
|  | **public** **virtual** Boolean IsUserPlugin  { **get** { **return** **true**; } } | | New - to distinguish system from user plugins. |
| **public virtual void** PresentChip(  Propeller host)  { } | **public virtual void** PresentChip()  { } | | Modify - removed parameter because Chip is declared now in all the plugins. |
| **public** **virtual** **void** OnReset() { } | | | - |
|  | **public** **virtual** **void** OnClose() { } | | New - to clean up before closing a plugin. |
| **public** **virtual** **void** OnClock(  double time) { } | **public** **virtual** **void** OnClock(  **double** time, **uint** sysCounter) { } | | Modify - added parameter to use system clock into a plugin. |
| **public** **virtual** **void** OnPinChange(  **double** time, PinState[] pins)  { } | | | - |
| **public** **virtual** **void** Repaint(**bool** force)  { } | | | - |
|  | **public** **void** NotifyOnPins()  { Chip.NotifyOnPins(**this**); } | | New – to simplify user plugins with declaring the most used methods. |
| **public** **void** NotifyOnClock()  { Chip.NotifyOnClock(**this**); } | | New - to simplify user plugins with declaring the most used methods. |
| **public** **void** DrivePin(  **int** pin, **bool** Floating, **bool** Hi)  { Chip.DrivePin(pin, Floating, Hi); } | | New - to simplify user plugins with declaring the most used methods. |
| **public** **void** BreakPoint()  { Chip.BreakPoint(); } | | New - to simplify user plugins with declaring the most used methods. |
| }  } | }  } | | - |

**Example of porting a plugin to Version 1.0:**

|  |  |  |
| --- | --- | --- |
| **Old plugin system (v0.0)** | **New plugin system (v1.0)** | **Explanation of Changes** |
| **using** Gear.EmulationCore;  **using** Gear.PluginSupport; | **using** Gear.EmulationCore;  **using** Gear.PluginSupport; | - |
| **class** PinNoise : PluginBase  { | **class** PinNoise : PluginBase  { | - |
| **private** Propeller **Chip;**  **private int Drive;**  **private bool Clock;**  **private double last;** | **private** **int** Drive;  **private** **bool** Clock;  **private** **double** last; | As the **Chip** reference is added by default for all plugins, you don’t need to add it anymore; it is inherited to your plugin.  Note as the former class Propeller is renamed PropellerCPU in V1.0, you need to change manually if you used it as parameter to a user method. |
|  | **public** PinNoise(PropellerCPU chip)  : **base**(chip)  {  OnReset();  } | In v0.0 you don’t need to invoke the constructor, but in v1.0 it provides you a way to initialize variables or a customized interface.  In this case we call the OnReset() method to reset the variables to consistently initial behavior. |
| **public** **override** **string** Title  {  **get** { **return** "Pin Noise"; }  } | **public** **override** **string** Title  {  **get** { **return** "Pin Noise"; }  } | - |
| **public** **override** **void** PresentChip(  Propeller host)  {  last = 0;  Chip = host;  Chip.NotifyOnPins(**this**);  Chip.NotifyOnClock(**this**);  } | **public** **override** **void** PresentChip()  {  NotifyOnPins();  NotifyOnClock();  } | 1) The parameter Propeller host is not needed anymore (the host reference is set in the constructor of base class), so if you need to call Chip, just do it in your code.  2) Note the initialization of variable last, now is better to do inside the OnReset() method.  3) As the methods NotifyOnPins() and NotifyOnClock() are inherited from PluginBase class, you just invoke them without to call from Chip, to make the code simpler to read. |
| **public** **override** **void** OnPinChange(  **double** time, PinState[] pins)  {  Drive++;  **if** (Drive % 3 == 0)  {  //Chip.DrivePin(int pin\_number,  // bool Floating,  // bool Hi)  Chip.DrivePin(3,  false,  (Drive & 1) == 1);  }  } | **public** **override** **void** OnPinChange(  **double** time, PinState[] pins)  {  Drive++;  **if** (Drive % 3 == 0)  {  //DrivePin(int pin\_number,  // bool Floating,  // bool Hi)  DrivePin(3,  false,  (Drive & 1) == 1);  }  } | As the method DrivePin() is inherited from PluginBase class, you just invoke it without to call from Chip, to make the code simpler to read. |
| **public** **override** **void** OnClock(  **double** time)  {  **if** (time - last >= 0.00001)  {  last += 0.00001;  Clock = !Clock;  Chip.DrivePin(2, false, Clock);  }  } | **public** **override** **void** OnClock(  **double** time, **uint** sysCounter)  {  **if** (time - last >= 0.00001)  {  last += 0.00001;  Clock = !Clock;  DrivePin(2, false, Clock);  }  } | 1) In v1.0 now it is possible to use the counter value inside this method, and not only the time as double in seconds.  2) As the method DrivePin() is inherited from PluginBase class, you just invoke it without to call from Chip, to make the code simpler to read. |
|  | **public** **override** **void** OnReset()  {  Drive = 0;  Clock = **false**;  last = 0.0;  } | Note that maybe in this case declaring that initial values are not necessary (because the built-in data types **int**, **bool** and **double** are implicitly initialized to zero), it is cleaner to do it this way, and in this method is a good place if you want to reset the initial behavior every time the same. |
| } | } | - |