March 2009

GUIAnchor is a module of code for 4D that gives the developer more control over the moving and resizing of window objects during a window resize or splitter movement. It is given as is to the 4D community. I would, however, appreciate hearing about bugs and enhancements that anyone finds or makes.

There is a 4D 2004 demo database included. Conversion to 4D v11 SQL has also been tested. The demo includes one window which show a variety of ways the code can be used. The demo also includes a the GUIAnchor folder with the six methods that make up the module.

The Array_Append method could be replaced by your own if you have one. If you do replace this method, you'll also want to delete the relevant lines in Compiler_GUIAnchor. GUIAnchor_ReadMe just explains how the module works.

The main methods are GUIAnchor_WindowInit which should be called once for a new window (a new process, really), GUIAnchor_SetAnchor which is generally called 1..n times to set up the anchors, and the GUIAnchor_HandleWindowResize which is called from the On Resize form event.

To use the code in your own databases, just move/copy/paste the methods in the GUIAnchor folder.

I've included the contents of the GUIAnchor_ReadMe method below so you can learn more about what it does without running the demo.

Feel free to contact me if you have questions or comments.

Cannon Smith, cannon@synergyfarmsolutions.com

```
`================================
`   Credits
`================================

`This module was created by Cannon Smith (cannon@synergyfarmsolutions.com) in January 2009.


`================================
`   Introduction
`================================

`4D has object properties which allow control over how an object on a form is moved and/or resized during a window resize.
`In many situations, this amount of control is sufficient. However, sometimes it is desirable to have a greater degree of control
`over how objects are moved and resized during window resizes. This is especially true when following interface guidelines that
`expect objects and groups of objects to appear centered in a window. This module is a generic way to get this type of control
`in an easy way.


`================================
`   Usage
`================================

`First, call GUIAnchor_WindowInit (usually in the On Load form event)

`Second, make a series of calls to GUI_SetAnchor to anchor object sides to other sides. This is also done in the On Load form event.
`Between 1 and 4 calls to GUI_SetAnchor will be made for each object you want the module to control during a resize. How many
`calls per object will depend on how many sides you want anchored. There are also times, usually due to manually moving a splitter,
`where you may need to reset a certain anchors. GUI_SetAnchor can be called again later on as necessary.

`Third, call GUIAnchor_HandleWindowResize in the form's On Resize event. In some interfaces, this may also need to be called
`during splitter events.


`================================
`   How it Works
`================================

`An object "side" refers to the left/top/right/bottom of an object as well as the horizontal/vertical center of an object.

`A form object's side is anchored to the side of some other form object (called the anchor object). If the name of the anchored
`object is not specified, then the module will assume the anchor object is to be the window itself.
```

`For example, the left side an object can be anchored to the horizontal center of another object. Or the vertical center of an object `can be anchored to the bottom of another object.

`A single object can be anchored to multiple objects, one anchor object per set anchor. For example, an object's left side could be `anchored to the left of the window while the object's right side could be anchored to the horizontal center of another object.

`If only one of the horizontal sides of an object is anchored, the object will be moved but not resize (unless 4D is growing the `object). If two of the horizontal sides of an object are anchored, the object has the potential to be resized as well as move. If all `three horizontal sides are anchored, the horizontal center anchor is ignored. These rules also apply to the vertical anchors.

`It is important to understand that the anchor is a relative anchor and each anchor can be one of two types: fixed or variable.
` When the anchors are first set during the On Load event, an offset between the object side and the anchor object side is calculated `and stored for each anchor that is set. This means the offsets are based on the relative positions of form objects as they are when `developing the form. During later resizing, the relative positions are maintained.

`For example, if we anchored the horizontal center of an object to the horizontal center of the window, and if the object was actually `20 pixels off center, then during the resizing, the object would always stay 20 pixels off center, no matter how the window was `resized. This is if the anchor is a "fixed" anchor. If the anchor is a "variable" anchor, rather than an actual pixel count for the offset, `the offset is actually a percentage of the anchor object's width (or height for vertical anchors). So if the object was 20% off center, it `would remain 20% off center during the resize. This would mean that the actual pixel offset would vary.

`You might think that the "variable" mode would only be useful if the object's anchor side falls within the width (or height) constraints `of the anchor object. This is normally the case, but the code will work if it is outside these bounds and can cause some interesting `effects.

`By combining different types of anchors and anchoring different sides of an object to different objects, you can achieve many `different possiblities with very little setup code.

`When the module calculates the new positions/sizes of anchored objects, it does so in the order you set the anchors up in the On `Load event. If objects rely on other objects for position and size, make sure they are set up in the correct order. This allows groups `of objects to be moved and resized together in complex ways while maintaining correct proportions.

```
`================================
`    The Math
`================================
```

`When the initial offsets are calculated for each anchor during the On Load event, the following formulas are used:

`If it is a fixed anchor: offset = anchor edge - object edge.
`If it is a variable anchor: offset = (object edge - anchor edge) / anchor width (and then scaled so we get the integer percentage)

`Later, when calculating the new location for an object, the formula's are just reversed:

`If it is a fixed anchor: object edge = anchor edge - offset
`If it is a variable anchor: object edge = anchor edge + (anchor width * offset) (the offset is scaled back to a real first)

```
`================================
`    Grouping Objects
`================================
```

`Objects can optionally be grouped together by passing a group name to the GUIAnchor_SetAnchor method. If a group name is `passed to GUIAnchor_HandleWindowResize, only the objects with that group name will be handled.

`This can be useful when using splitters. You might save processing time by only handling objects affected by a splitter's `movement.

`Note that if you try to set different groups to an object with multiple anchors, the first group set to the object will be the one that `is associated with the object.

```
`================================
`    Mixing This Module With 4D Object Settings
`================================
```

`This module works well with 4D move/grow settings for objects. If there is a conflict, this module always wins. It is usually best to `allow 4D to handle as much moving/resizing as possible and only set anchors with this module as needed. You can, for example, `have 4D grow an object while also setting an anchor to keep the object's horizontal center anchored to the right side of another `object. The other object could be set to grow with 4D settings. The net result is that the second object would expand in size, the `first object would also expand in size, but it would also stay centered around the second object's right side as it moved. Thus, it `would appear that the second object was growing in both directions while moving with the second object.

```
`================================
`    What It Won't Do
`================================
```

`This module necessarily used process variables and was not designed for use with multiple windows stacked in the same process.
`However, as long as there are no object name conflicts, it would be possible to group all the objects of the first window and group
`all the objects of the second window using the optional group parameter and then only process the respective group during the
`window resize as a work around.

```
`================================
`    Examples
`================================
```

`Example 1: You have an object that needs to stay centered (horizontally) on the window.

```
`GUIAnchor_SetAnchor ("Object";"HortCenter";"";"HortCenter";"Fixed")
```

`Example 2: You have a checkbox that is centered on the form. There is another checkbox underneath the first that is indented 8
`pixels. You want this "group" of objects to stay centered on the form.

```
`GUIAnchor_SetAnchor ("Checkbox1";"Left";"";"HortCenter";"Fixed")`Keep the first one centered
`GUIAnchor_SetAnchor ("Checkbox2";"Left";"Checkbox1";"Left";"Fixed")
```

`Note that this is an example where the first anchor could have been set using any of Checkbox1's side with the same result.
`This is often the case for objects that only have one anchor and do not grow.

`Example 3: You have a series of buttons (let's say three) at the top of the window. They are different widths and you want them
`to expand in width as the window grows while maintaining their relative widths. You want the left most button to stay the same
`distance from the left edge of the window and the right most button to stay the same distance from the right edge of the window.

```
`GUIAnchor_SetAnchor ("Button1";"Left";"";"Left";"Fixed") `Anchor the left edge of the button to the left edge of the window
`GUIAnchor_SetAnchor ("Button1";"Right";"";"Left";"Variable") `Let the right edge of the button vary with the window width

`GUIAnchor_SetAnchor ("Button2";"Left";"Fixed";"Button1";"Right")`Fix the left edge of this button to the right of the first button
`GUIAnchor_SetAnchor ("Button2";"Right";"";"Left";"Variable")`This edge also varies with the window width

`GUIAnchor_SetAnchor ("Button3";"Left";"Button2";"Right";"Fixed")`This edge is fixed against the previous button
`GUIAnchor_SetAnchor ("Button3";"Right";"";"Right";"Fixed")`The edge is fixed relative to the right edge of the window
```

`Note that if there was, say, 3 pixels of space between each button in design mode, that would continue to be the case during a resize.