



Smart Widget: Rotary Menu

DOCUMENT DATE: **9th MAY 2020**
DOCUMENT REVISION: **1.0**

WWW.4DSYSTEMS.COM.AU



Description

This application note shows how to create a rotary menu using a smart gauge in combination with Genie-Magic on a Picaso, Diablo16, and Pixxi touch screen display modules.

Before getting started, the following are required:

Hardware

- Any [4D Systems display module](#) powered by any of the following processors:
 - o Picaso
 - o Diablo16
 - o Pixxi24/44
- [Programming Adaptor for target display module](#)
- [uSD Card](#)
- [USB Card Reader](#)

Software

- [Workshop4](#)
- This requires the **PRO** version of Workshop4

This application note comes with one (1) ViSi-Genie project.

Note: Using a non-4D programming interface could damage the processor and void the warranty.

Content

Description	2
Content	2
Application Overview.....	3
Setup Procedure	3
Create a New Project	3
Design the ViSi-Genie Projects.....	4
<i>Add a Smart Gauge Object</i>	<i>5</i>
<i>Add a User Images Object</i>	<i>8</i>
<i>Add Form1, Form2, and Form3.....</i>	<i>9</i>
<i>Add Magic Code Objects.....</i>	<i>12</i>
<i>Add a Magic Touch Object.....</i>	<i>14</i>
<i>Add a Magic Move Object</i>	<i>15</i>
<i>Add a Magic Release Object.....</i>	<i>16</i>
Run the Program.....	16
Proprietary Information	17
Disclaimer of Warranties & Limitation of Liability	17

Application Overview

The rotary menu discussed in this application note is a set of icons arranged on quarter circle. Each icon is enlarged when in focus. The rotary menu widget is implemented using a smart gauge and custom 4DGL code. Custom 4DGL code is added to the ViSi-Genie project in the form of Genie-Magic objects. Also, a user images object is added as a selector.

It is assumed that the reader knows how to create smart widget objects and how to use magic objects. Recommended relevant application notes are:

- **Smart Widgets Circular Progress Bar**
- **ViSi-Genie How to Add Magic Objects**
- **ViSi-Genie Magic Code Insertion Points**

Setup Procedure

For instructions on how to launch Workshop4, how to open a **ViSi-Genie** project, and how to change the target display, kindly refer to the section “**Setup Procedure**” of the application note

- [ViSi-Genie Getting Started - First Project for Diablo16 Display Modules](#)
- [ViSi-Genie Getting Started - First Project for Picaso Displays](#)
- [ViSi-Genie Getting Started - First Project for Pixxi Display Modules](#)

Create a New Project

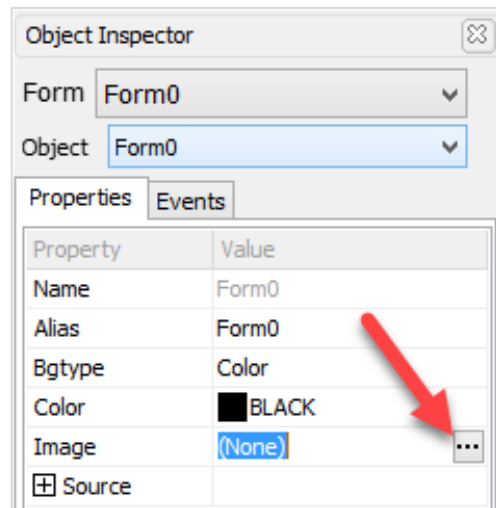
For instructions on how to create a new **ViSi-Genie** project, please refer to the section “**Create a New Project**” of the application note

- [ViSi-Genie Getting Started - First Project for Diablo16 Display Modules](#)
- [ViSi-Genie Getting Started - First Project for Picaso Displays](#)
- [ViSi-Genie Getting Started - First Project for Pixxi Display Modules](#)

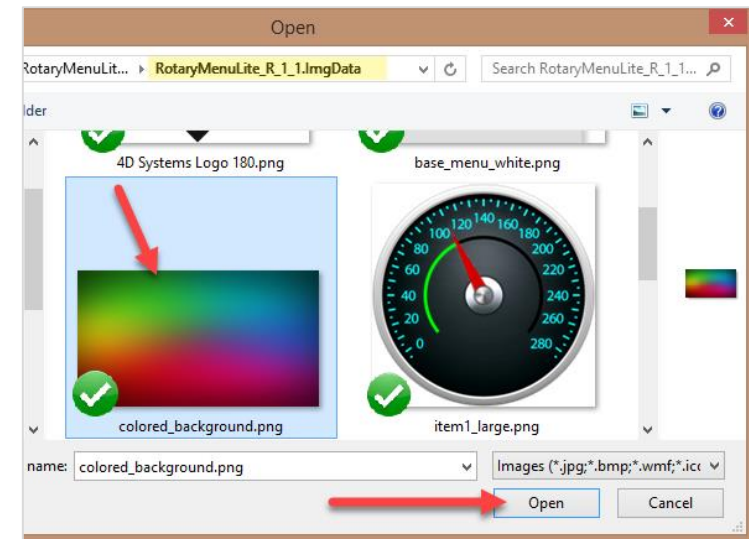
Design the ViSi-Genie Projects

In this application note, a gen4-uLCD-43DCT-CLB is used. A working project is also attached for reference (RotaryMenuLite_R_X_Y.zip). The same procedure is applicable to any Picaso and Diablo16 displays.

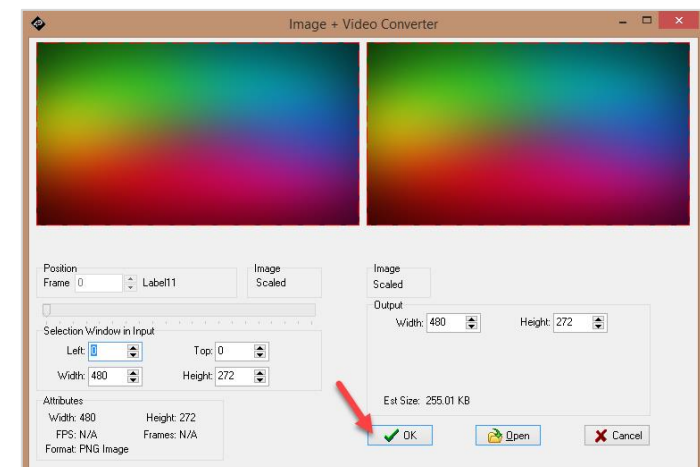
First, add a background image to **Form0** by clicking on the ellipsis dots in the object inspector as shown below.



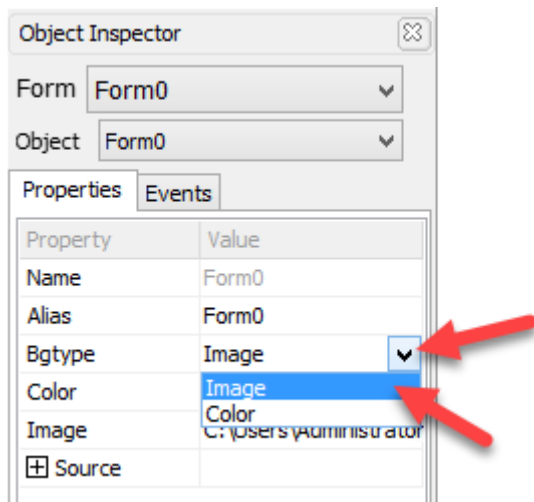
The background image used in the attached project can be found inside the folder indicated below.



The image + video converter window appears. Click **OK**.



Set the value of **Bgtype** to “Image”.

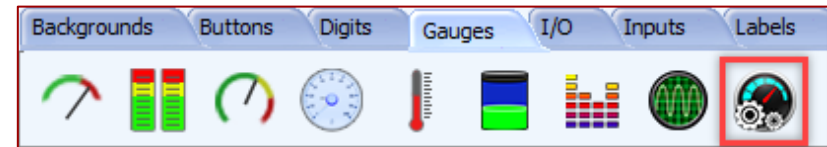



The WYSIWYG area should now be updated.

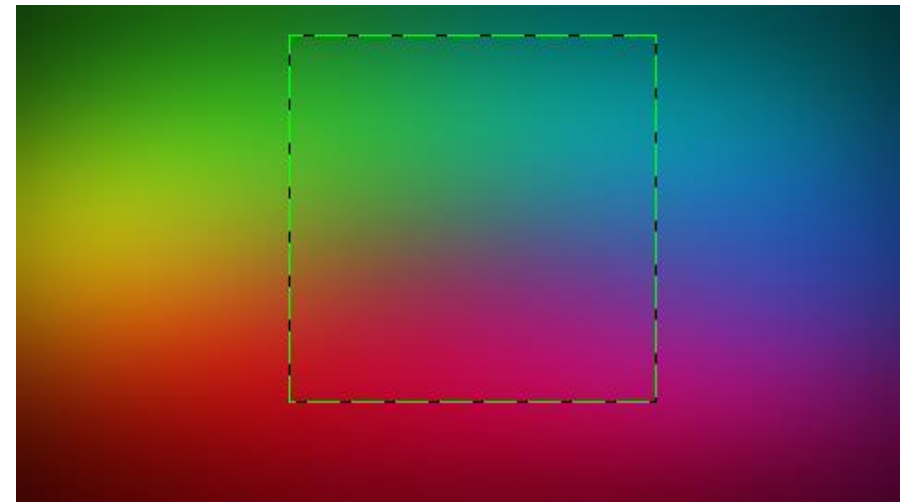


Add a Smart Gauge Object


Add a smart gauge object to the ViSi-Genie project. Workshop4 will name this as “SmartGauge0”. The smart gauge icon can be found under the Gauges tab of the widgets pane.

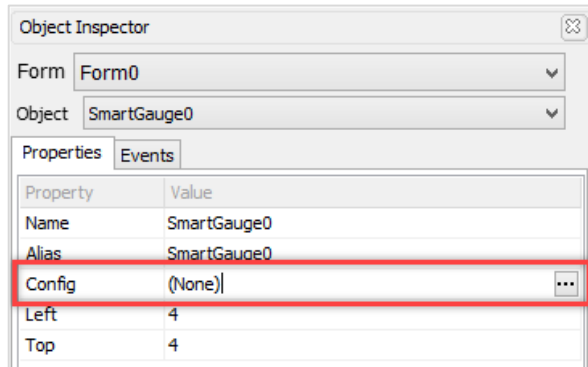


Simply click on the icon  to select it. Then place it on the WYSIWYG area.

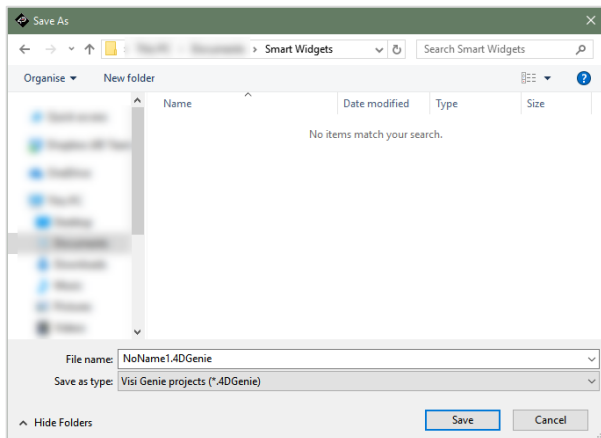


As displayed on the above image, the widget appears empty when placed in the WYSIWYG area.

Open the Smart Widgets Editor tool by clicking on  of **Config** in the Object Inspector Properties tab.

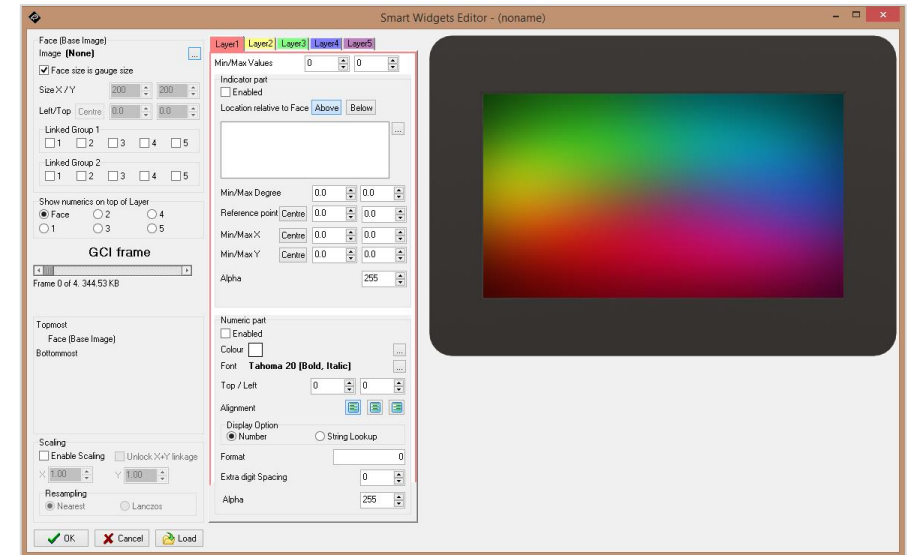


The tool requires that the project is already saved before the tool opens. Therefore, since in this case, it hasn't been saved yet, Workshop4 will automatically prompt the user to save

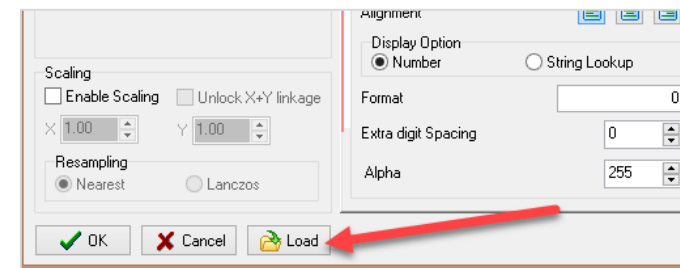


Save the project to a desired location.

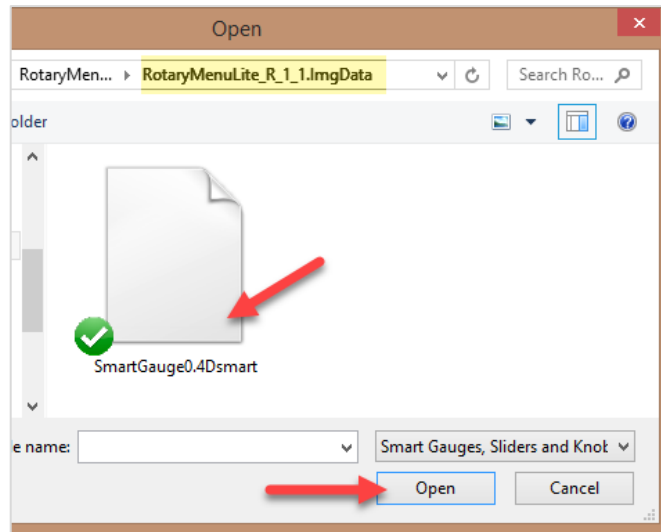
The tool will open after the project has been saved.



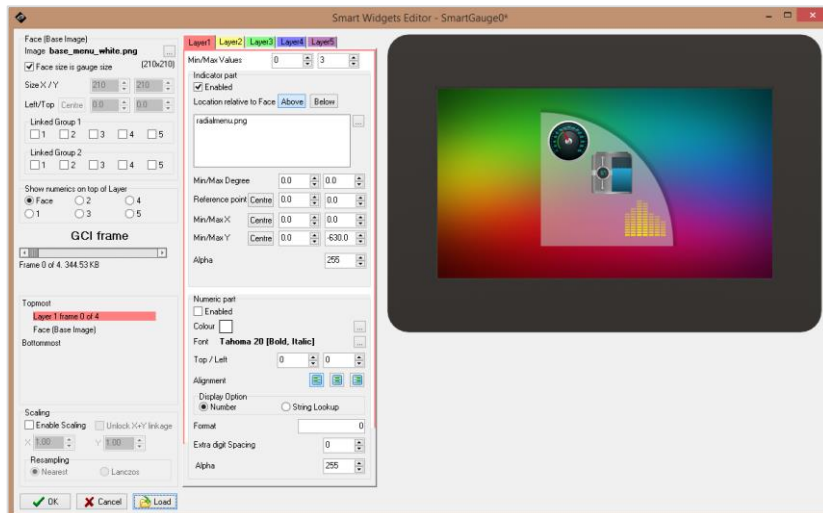
Click the **Load** button to load an existing smart gauge object.



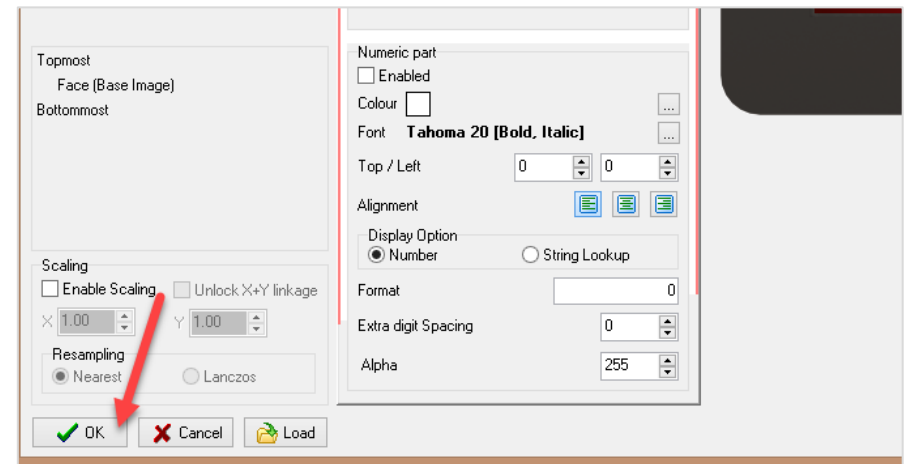
Select the file inside the folder indicated below then click **Open**.



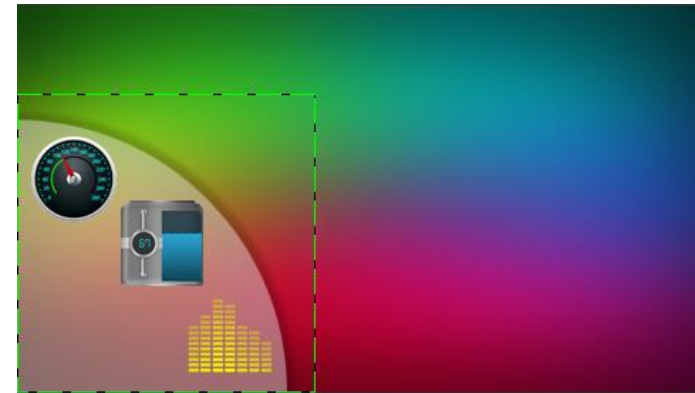
The selected smart gauge object shall now appear in the WYSIWYG area of the Smart Widgets Editor tool.



Click **OK**.

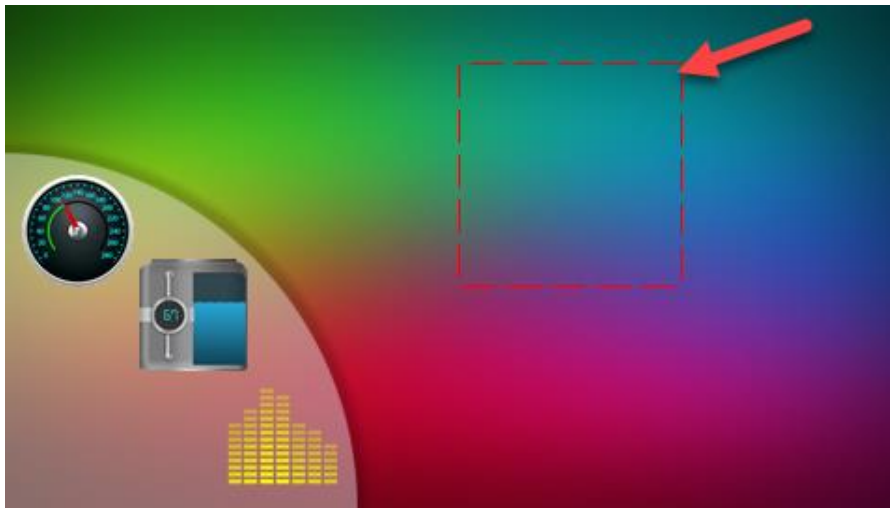


The smart gauge object should now appear on the WYSIWYG area. It can be moved to another location as shown below.

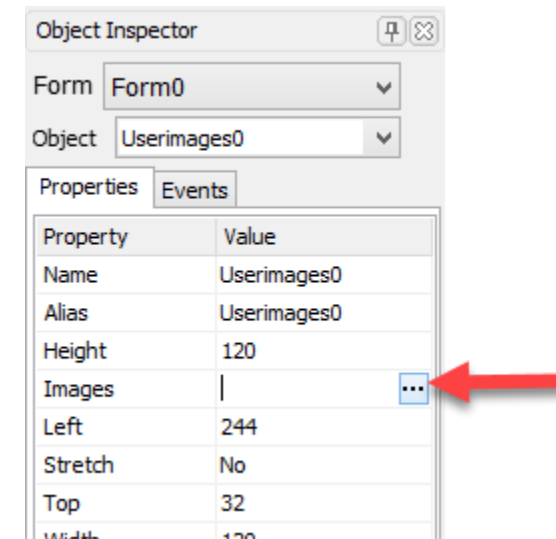


Add a User Images Object

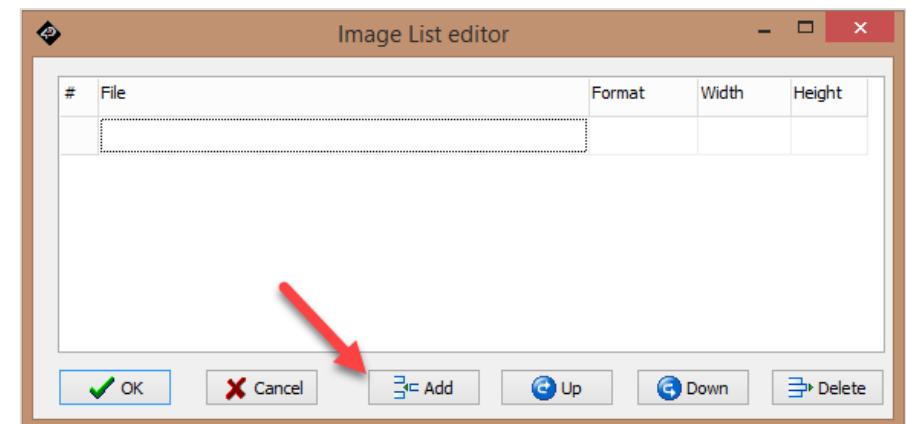
Add a user images object to the project. Workshop4 will name this as "Userimages0". This will be used as a selector button to enable the user to navigate to the corresponding form for the highlighted icon. The user images icon is found under the System/Media tab.



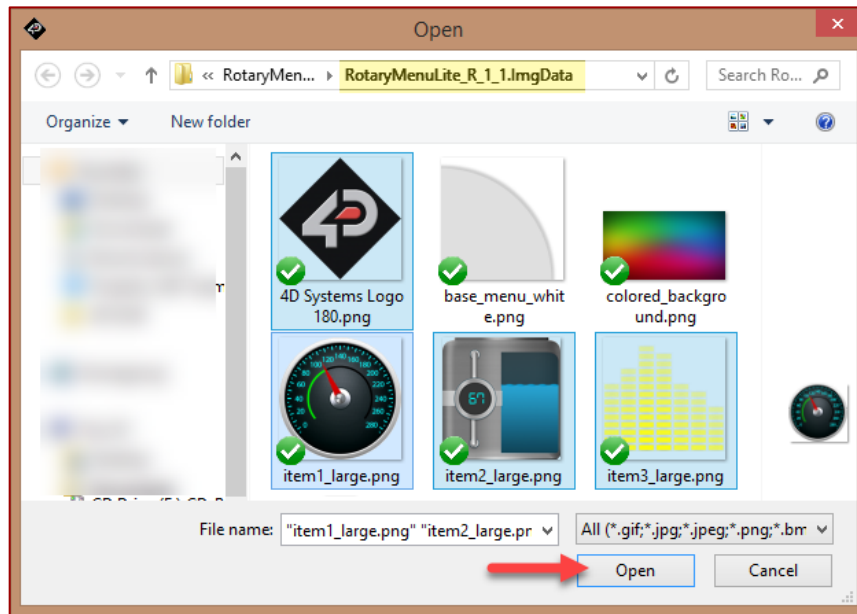
To add images to the user images object, click on the ellipsis dots of the object inspector as indicated below.



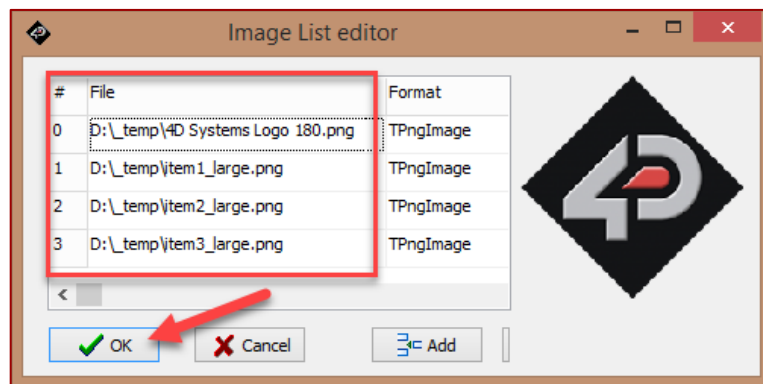
The image list editor appears. Click on the add button.



Select the four image files as shown below then click open.



The image list editor appears again. Make sure that the images are indexed in the order shown below, then click OK.

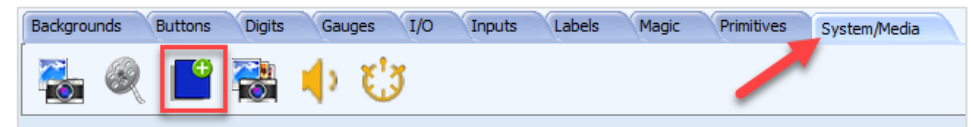


The WYSIWYG area should now be updated.

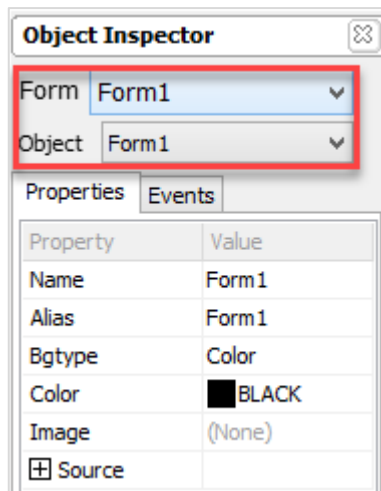


Add Form1, Form2, and Form3

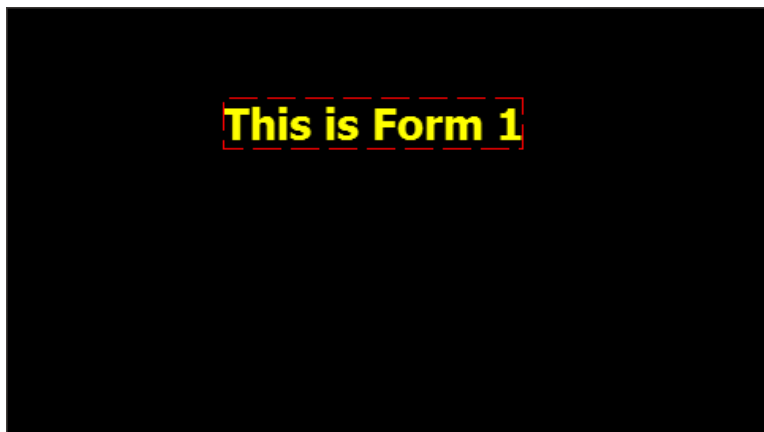
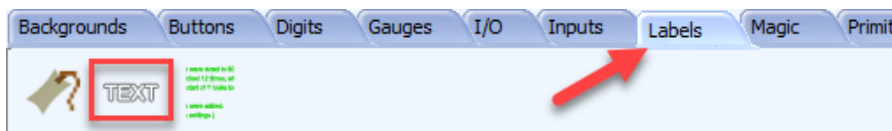
When an icon is selected, the program navigates to another form which corresponds to the selected icon. Thus, additional forms need to be added to the project. To add a form, go to the System/Media tab and click on the icon indicated in the image below.



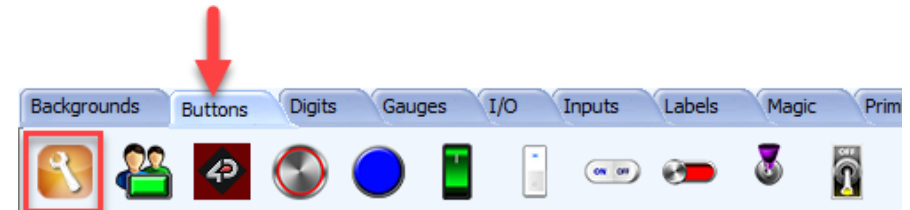
A new, empty form is now added to the project.



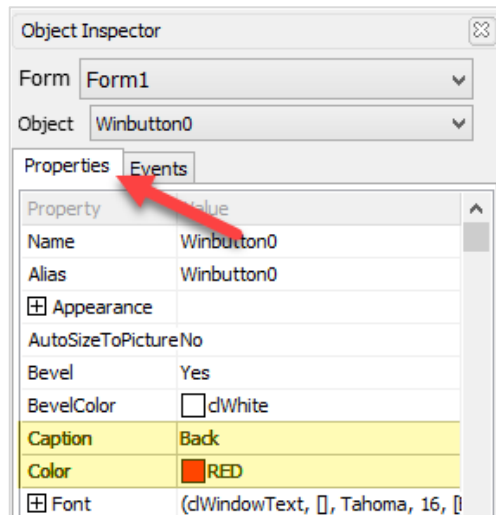
Add a static text object to the form. Static text objects can be used as labels.



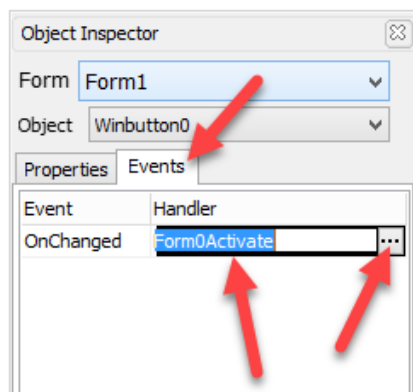
Furthermore, for the user to be able to navigate back to **Form0**, a winbutton object is added to the form.



Properties of the winbutton object such as **Color** and **Caption** can be changed thru the properties tab of the object inspector.



Finally, the **OnChanged** event handler of the button should be set to "Form0Activate" so that it navigates to **Form0** when pressed and then released.



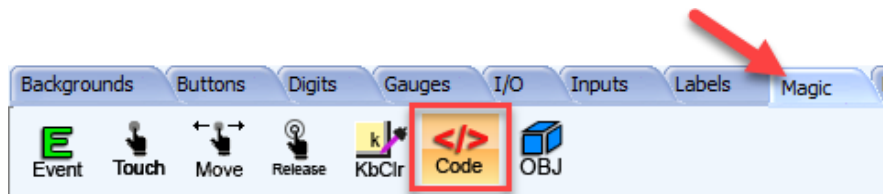
Repeat the above procedure to create **Form2** and **Form3**. When done, the project should now have a total of 4 forms – **Form0**, **Form1**, **Form2**, and **Form3**.



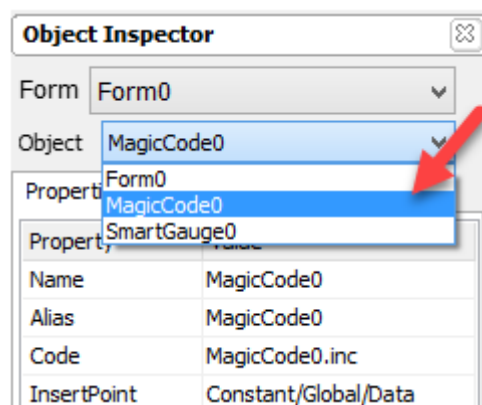
Add Magic Code Objects

Add two magic code objects to the project – these will be named as “MagicCode0” and “MagicCode1” by Workshop4. The 4DGL functions needed to implement the rotary menu functionality will reside in MagicCode0. MagicCode1, on the other hand, stores the routines to enable touch detection for the smart gauge and user images objects.

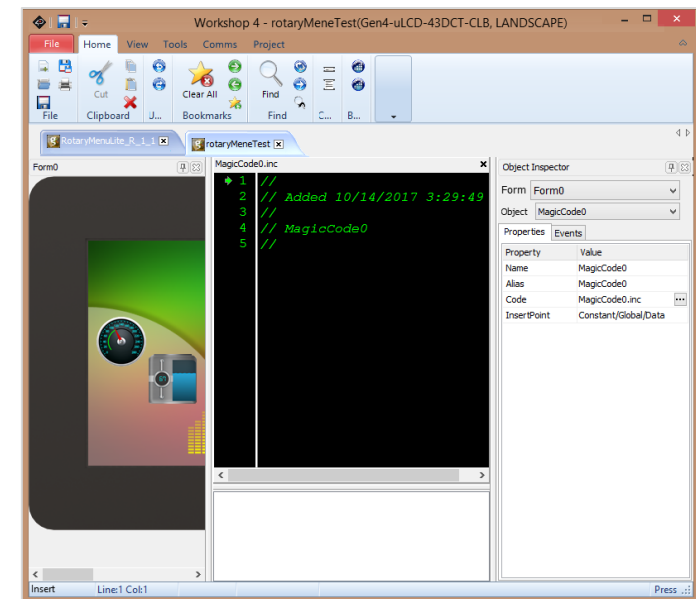
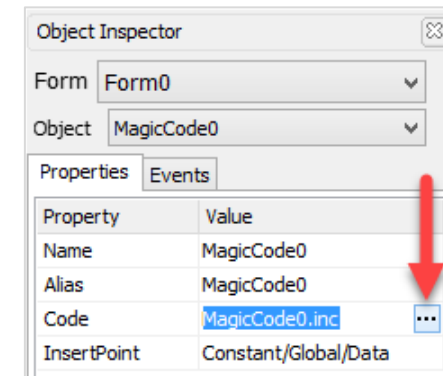
To add a magic code object, click on the icon indicated in the image below.



The magic code object should now appear in the Objects Inspector.

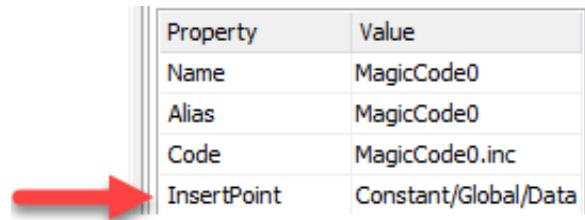


To open the source code for **MagicCode0**, click on the ellipsis dots as indicated below.



As can be seen above, **MagicCode0** has no code yet. Open the attached project “RotaryMenuLite_R_X_Y” and copy the contents of its **MagicCode0**, then paste it to **MagicCode0** of the current project.

Note that the insertion point for **MagicCode0** is “Constant/Global/Data”.



Property	Value
Name	MagicCode0
Alias	MagicCode0
Code	MagicCode0.inc
InsertPoint	Constant/Global/Data

Note further that **MagicCode0** has one global variable and 4 functions, which are described below.

Variable or Function Name	Description
selectedMenuItem	This variable holds the value of the currently selected icon. This variable is updated depending on the coordinates of the touch point.
updateMenu()	This function is used to display a frame of SmartGauge0 and Userimages0. selectedMenuItem is used as the index of the frame.
evaluateMenuTouch()	This function checks the x and y touch coordinates when the touch status is “pressed” and/or “moving”, updates the value of selectedMenuItem and runs updateMenu() .
evaluateMenuRelease()	This function checks the x and y touch coordinates when the touch status is “released”, updates the value of selectedMenuItem and runs updateMenu() .

resetMenuForm()

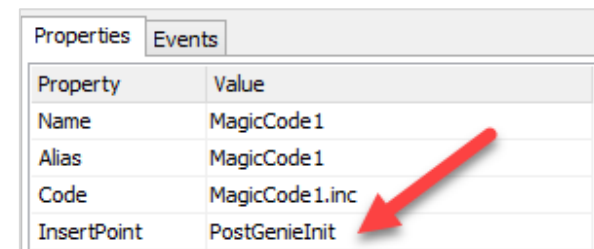
This functions resets the value of selectedMenuItem to “0” (which means nothing is currently selected) and runs **updateMenu()**.

Add another magic code object to the project - **MagicCode1**. Similar to what was done for **MagicCode0**, copy the 4DGL code of **MagicCode1** in the attached project “RotaryMenuLite_R_X_Y”. **MagicCode1** contains two lines of code:

```
img_ClearAttributes(hndl, iSmartGauge0, I_TOUCH_DISABLE);
img_ClearAttributes(hndl, iUserimages0, I_TOUCH_DISABLE);
```

These lines simply enable touch detection for **SmartGauge0** and **Userimages0**. Note that these objects are normally non-touchable in ViSi-Genie. However, with Genie-Magic, finer control of behavior of objects is possible.

Note that the insertion point for **MagicCode1** is “PostGenieInit”.



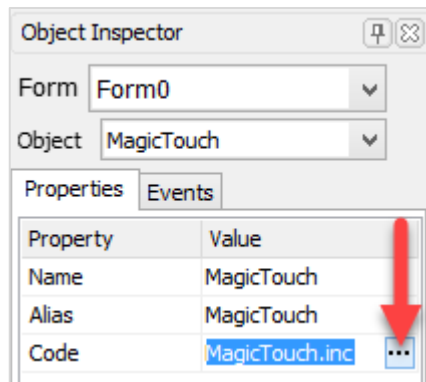
Property	Value
Name	MagicCode1
Alias	MagicCode1
Code	MagicCode1.inc
InsertPoint	PostGenieInit

Add a Magic Touch Object

To add a magic touch object, click on the magic touch icon under the Magic tab, as indicated in the image below.



Note that there is only one magic touch object, called “MagicTouch” in the project. To open the source code for **MagicTouch**, click on the ellipsis dots as indicated below.



Workshop4 opens the code area and displays the code for “MagicTouch”. At this point, the code is empty. Copy the code from **MagicTouch** of the attached project “RotaryMenuLite_R_X_Y”.

As the name implies, the magic touch code is executed by the Genie program every time that the status of touch is "pressed". The index (**ImageTouched**) of the object on which a press has occurred is evaluated. If

the object being pressed is **SmartGauge0**, the function **evaluateMenuTouch()** is executed. On the other hand, if the object being pressed is **Userimages0**, the current value of **selectedMenuItem** is used to navigate to another form. Shown below is the code for **MagicTouch**.

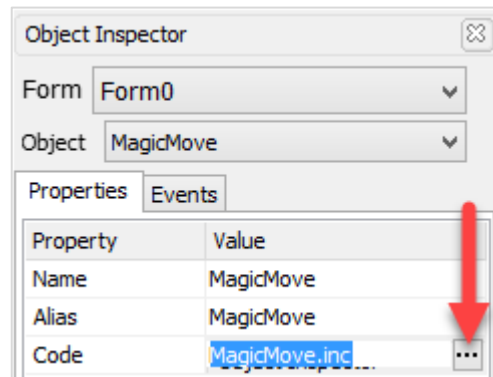
```
switch(ImageTouched)
  case iSmartGauge0:  // If Rotary menu is touched
    evaluateMenuTouch();  // Evaluate
    ImageTouched := -1;
    break;
  case iUserimages0:  // If Item Display is touched
    switch(selectedMenuItem)
      case -1:
      case 0:
        break;
      default:
        ActivateForm(selectedMenuItem);  //Navigate to
corresponding form
        resetMenuForm();
        break;
    endswitch
    ImageTouched := -1;
    break;
endswitch
```

Add a Magic Move Object

To add a magic move object, click on the magic move icon under the Magic tab, as indicated in the image below.



Note that there is only one magic move object, called “MagicMove” in the project. To open the source code for **MagicMove**, click on the ellipsis dots as indicated below.



Workshop4 opens the code area and displays the code for **MagicMove**. At this point, the code is empty. Copy the code from **MagicMove** of the attached project “RotaryMenuLite_R_X_Y”.

As the name suggests, the magic move code is executed by the Genie program every time that the status of touch is “moving”. The index (**ImageTouched**) of the object on which a movement is detected is evaluated. If the object on which a movement is occurring is **SmartGauge0**, the function **evaluateMenuTouch()** is executed. Shown below is the code for **MagicMove**.

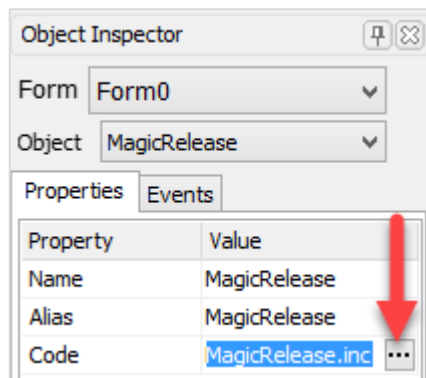
```
switch(ImageTouched)
  case iSmartGauge0: // If touch is moving in menu
    evaluateMenuTouch(); // Evaluate
    ImageTouched := -1;
    break;
  case iUserimages0:
    ImageTouched := -1;
    break;
endswitch
```

Add a Magic Release Object

To add a magic release object, click on the magic release icon under the Magic tab, as indicated in the image below.



Note that there is only one magic release object, called “MagicRelease” in the project. To open the source code for **MagicRelease**, click on the ellipsis dots as indicated below.



Workshop4 opens the code area and displays the code for **MagicRelease**. At this point, the code is empty. Copy the code from **MagicRelease** of the attached project “RotaryMenuLite_R_X_Y”.

The magic release code is executed by the Genie program every time that the status of touch is "released". The index (**ImageTouched**) of the object on which a touch release has occurred is evaluated. If the object is **SmartGauge0**, the function **evaluateMenuRelease ()** is executed. Shown below is the code for **MagicRelease**.

```
switch(ImageTouched)
  case iSmartGauge0: // If the menu is released,
    evaluateMenuRelease(); // Evaluate menu
    ImageTouched := -1;
    break;
  case iUserimages0:
    ImageTouched := -1;
    break;
endswitch
```

Run the Program

For instructions on how to save a **ViSi-Genie** project, how to connect the target display to the PC, how to select the program destination, and how to compile and download a program, please refer to the section “**Run the Program**” of the application note

- [ViSi-Genie Getting Started - First Project for Diablo16 Display Modules](#)
- [ViSi-Genie Getting Started - First Project for Picaso Displays](#)
- [ViSi-Genie Getting Started - First Project for Pixxi Display Modules](#)

Proprietary Information

The information contained in this document is the property of 4D Systems Pty. Ltd. and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

4D Systems endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of 4D Systems products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Systems.

All trademarks belong to their respective owners and are recognised and acknowledged.

Disclaimer of Warranties & Limitation of Liability

4D Systems makes no warranty, either expresses or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

In no event shall 4D Systems be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Systems, or the use or inability to use the same, even if 4D Systems has been advised of the possibility of such damages.

4D Systems products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage ('High Risk Activities'). 4D Systems and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Systems' products and devices in 'High Risk Activities' and in any other application is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless 4D Systems from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Systems intellectual property rights.