

Smart Widgets: DropDown Menu

DOCUMENT DATE: DOCUMENT REVISION: 9th MAY 2020 1.0



WWW.4DSYSTEMS.COM.AU

Description

This application note shows how to create a dropdown menu that overlaps a Gauge for Picaso, Diablo16 and Pixxi touch screen display modules.

Before getting started, the following are required:

Hardware

- Any <u>4D Systems display module</u> powered by any of the following processors:
 - \circ Diablo16
 - \circ Picaso
 - Pixxi28/44
- Programming Adaptor for target display module
- <u>uSD Card</u>
- USB Card Reader

Software

- Workshop4
- This requires the PRO version of Workshop4

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

This application note is applicable to all touch screen 4D displays. However, Smart Gauges can also be used on non-touch displays.

Content

Description	2
Content	2
Application Overview	
Setup Procedure	
Create a New Project	
Design the Project	
Add a Smart Gauge Object	4
Open the Smart Widgets Editor Tool	5
Design New Gauge or Load Saved Gauge	6
Preparing a Layer for DropDown Menu	6
Designing a DropDown Menu	7
Checking the Order of Frames	9
Use SmartGauge as Input Object	10
Add Multiple Input Objects	11
Run the Program	12
Proprietary Information	13
Disclaimer of Warranties & Limitation of Liability	13

Application Overview

The Smart Widgets Editor tool enables PRO version users to easily create custom widgets of their own design. It allows the user to create Sliders, Knobs and Gauges.

The purpose of this application note is to introduce the PRO version exclusive tool and to discuss how to create a tank using a Smart Gauge widget. This application note uses the ViSi-Genie environment.

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

- <u>ViSi-Genie Getting Started First Project for Diablo16 Display Modules</u>
- <u>ViSi-Genie Getting Started First Project for Picaso Displays</u>
- <u>ViSi-Genie Getting Started First Project for Pixxi Display Modules</u>

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

- <u>ViSi-Genie Getting Started First Project for Diablo16 Display Modules</u>
- <u>ViSi-Genie Getting Started First Project for Picaso Displays</u>
- <u>ViSi-Genie Getting Started First Project for Pixxi Display Modules</u>

Design the Project

For this application, gen4-uLCD-43DT will be used for the project. Same procedure is applicable for any Picaso, Diablo16 and Pixxi display module. First, set the background image of Form0.

Object Inspector	8
Form Form0	~
Object Form0	
Properties Even	ts
Property	Value
Name	Form0
Alias	Form0
Bgtype	Image 🚽 🔤 🛃
Color	BLACK
Image	blue-textures.jpg
Source	

Add a Smart Gauge Object

Add a Smart Gauge widget to your ViSi-Genie project. It can be found on the Gauges tab on the Widgets Pane.

Backgrounds	Buttons	Digits	Gauges	I/0 I	Inputs	Labels	Magic	Primitives	System/Media	
∽ 🚺	\bigcirc									

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



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

Open the Smart Widgets Editor Tool

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

Properties Eve	nts
Property	Value
Name	SmartGauge0
Alias	SmartGauge0
Config	SmartGauge0
Left	140
Тор	0

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

😔 Save As						×
← → * ↑		> Smart Widgets	~ ē	Search Smart V	Vidgets	Q
Organise 🔻 Ne	w folder				· = = •	?
	^ Name	^	Date modified	Туре	Size	
-		No ite	ms match your s	earch.		
a taste						
- Sector						
	v					
File name:	NoName1.4DGenie					~
Save as type:	Visi Genie projects (*.4D	(Genie)				~
∧ Hide Folders				Save	Cance	4

Save the project to desired location. The tool will open after the project has been saved.

Face (Base Image)	Bauel Bauel Street Court brent	
Image (None)	Layers Layers Layers	
Face size is gauge size	Min/Max Values U 🖶 U 🖶	
Size X / Y 200 🔅 200 🔅	Enabled	
Left/Top Centre 0.0 1 0.0 1	Location relative to Face Above Below	
Linked Group 1		
Linked Group 2		and the second se
Show numerics on top of Laver	Min/Max Degree 0.0 🜩 0.0 🜩	
● Face ○ 2 ○ 4	Reference point Centre 0.0 0.0	
01 03 05	Min/MaxX Centre 0.0 💠 0.0 🜩	
GCI frame	Min/Max Y Centre 0.0 \$ 0.0 \$	
C	Alpha 255	
Frame 1 of 1. 0 Bytes	Hand I I	
Topmost	Numeric part	
Face (Base Image)	Enabled	
Bottommost	Colour	
	Font Tahoma 20 [Bold, Italic]	
	Top / Left 0 💠	
	Alignment 🔳 🔳	
	Display Option O Stripp Lookup	
Scaing	Format	
	Extra disk Spacing	
Resampling		
Nearest Clanczos	Alpha 255 🜩	

As shown in the image, this tool has a lot of parts. The next steps will focus only on the minimum tool functionalities required to make a basic circular progress bar.

For detailed discussion on how each part works, please refer to the **Smart Widgets Editor User Guide**.

Design New Gauge or Load Saved Gauge

You may design a new Gauge or use a previously saved one before adding the dropdown menu.

Ensure that you have at least one unused layer for the Dropdown menu. For this appnote project, a previously saved gauge will be used.



The following buttons can be found on the bottom-left area of the editor window. Click on **Load.** It will automatically open to the directory of smart widgets from Workshop4.

Find SimpleGauge.4Dsmart and click Open.



At the time, this appnote was written the gauge is still available. In case it is not available or there are any issues with the copy found, you may also find it the appnote project folder under the same filename.

The editor should display the gauge after it loads the 4Dsmart file as shown in the previous image.

Preparing a Layer for DropDown Menu

Enable another layer. This will be used for the dropdown menu. Ensure that it is the topmost layer. If you're following the same procedure as the appnote, you should be able to enable Layer 2.

Layer1 Layer2 Layer3	Layer4	Layer5
Min/Max Values	0	0
Indicator part		_
Location relative to Fac	e Above	Below

Place it above the base/face image. And swap it with Layer 1.

Topmost	
Layer 1 frame 0 of 31	-
Layer 2 frame 0 of 1	 •
Face (Base Image)	
Bottommost	

After swapping the 2 Layers, you'll notice that the 31 frames of Layer 1 is transferred to Layer 2:

Topmost	
Layer 1 frame 0 of 1	•
Layer 2 frame 0 of 31	•
Face (Base Image)	
Bottommost	

The transparent part of the base image will be filled up with liquid. This liquid part can be another image which moves vertically or even diagonally to fill the tank. To implement this, we need a layer containing a manipulated image. We will use Layer 1 for this purpose.

Designing a DropDown Menu

A dropdown menu should have both close:



And open state:



Additionally, for both states, it should show the currently selected item.

For that purpose, the open dropdown menu state needs to have a frame for each item. Additionally, there needs to be multiple duplicates of close state so that it equals the number of open states.

In this appnote, the frames used for the layer are:

- Closed_Dropdown.png
- Closed_Dropdown.png
- Closed_Dropdown.png
- Closed_Dropdown.png
- Open_Dropdown_RED.png
- Open_Dropdown_Blue.png
- Open_Dropdown_Green.png
- Open_Dropdown_Yellow.png

Enable the Numerical Part and set it to String Lookup. The string should contain the following:

- RED
- BLUE
- GREEN
- YELLOW
- RED
- BLUE
- GREEN
- YELLOW

Position the strings correctly and choose your desired settings.

This appnote project uses the configuration below:

Numeric part Enabled	
Colour	
Font Segoe UI	14 [Bold]
Top / Left	1 🛊 13 🛊
Alignment	EE
Display Option Number	String Lookup
RED BLUE GREEN YELLOW	^
Alpha	255 🚔

Finalize the gauge by bringing the gauge down so that when the dropdown menu is open, the gauge is not covered by it.

	Face (Base Image Image WS4_Ga) uge03_`	1 smal	l.png
	🔲 Face size is ga	uge size		(200x200)
	Size X / Y	200	÷	250 🌻
<i>•</i>	Left/Top Centre	0.0	÷	50.0 🌻
	Linked Group 1	3	4	5
	Linked Group 2	3	4	5

Adjust the face size and top position accordingly.



You'll notice that the needle was left behind by the base image.

arrow3.png					
Min/Max Degree		0.0	•	300.0	•
Reference point	Centre	100.0	•	100.0	▲ ▼
Min/Max X	Centre	100.0	•	100.0	•
Min/Max Y	Centre	150.0	•	150.0	•

Adjust the Y position accordingly

Checking the Order of Frames

The order of frames can easily be observed by utilizing the GCI frame slider.



You can slowly observe the frame by utilizing the buttons at both ends of the slider.

The first few press on the button on the right end of the slider will show that the dropdown menu changes state from frame 0 to 7; moving a total of 8 frames. Afterwards, it will return to the layer's frame 0 (1st frame) which is when the gauge will move 1 unit.

The GCI frame can be computed as shown below:

$$frame_{GCI} = frame_{L1} + 8 (frame_{L2})$$

Where

frame is from 0 to N.

The Layer1 frames can also be broken to 2 values.

Following the arrangement of the frames:

- Closed_Dropdown.png
- Closed_Dropdown.png
- Closed_Dropdown.png
- Closed_Dropdown.png
- Open_Dropdown_RED.png
- Open_Dropdown_Blue.png
- Open_Dropdown_Green.png
- Open_Dropdown_Yellow.png

It can be noticed that it will finish going through the 4 colors before moving to the Open state which will then go through the 4 colors then go back to the first frame.

That being the case, the formula can be expanded to:

$$frame_{GCI} = frame_{L1} + 8 (frame_{L2})$$

$$frame_{GCI} = 4 (state) + color + 8 (frame_{L2})$$

Where:

state is 0 for closed and 1 for opened dropdown menu

color is 0 to 3 for Red, Blue, Green and Yellow

This equation will be used in the code later.

Use SmartGauge as Input Object

SmartGauges are non-input objects by default. For this step, 4DGL code will be added to the project.



Touch functionality should be enabled for the SmartGauge object when currently at the form containing it and disabled if not. For that add a **Magic Code** to the project.

Backgrounds		Buttons	Digits	Gau	iqes I	0 Inp	uts Labels	Magic	Primitives	System/Media
Event To	uch	t Move	Release	k KbClr	Code	OB J				

Set the insert point to be **PostActivateForm**

Property	Value			
Name	MagicCode0			
Alias	MagicCode0			
Code	MagicCode0.inc ····			
InsertPoint	PostActivateForm			

Open the editor by clicking on 🛄 of the row **Code**

Then add the following lines of code.

```
if (CurrentForm == 0)
    img_ClearAttributes(hndl, iSmartGauge0, I_TOUCH_DISABLE);
else
    img_SetAttributes(hndl, iSmartGauge0, I_TOUCH_DISABLE);
endif
```

Add another Magic Code and set its insertion point to Global

Property	Value
Name	MagicCode 1
Alias	MagicCode 1
Code	MagicCode 1.inc
InsertPoint	Constant/Global/Data

var gaugeValue[4]; var selectedItem; var menuState;

Insert the above lines of code to declare the following variables and array. Notice that they are values relating to the frame's formula.

We need to program what it will do when a touch event occurs. Add **Magic Touch**, **Magic Release** and **Magic Move** objects to the project. Then add the following lines of code to each of the three.

```
if (ImageTouched == iSmartGauge0)
    ImageTouched := -1;
endif
```

Open Magic Touch object add these lines of code inside the *if* condition.

```
var item, topPos;
topPos := img_GetWord(hndl, iSmartGauge0, IMAGE_YPOS);
item := (TouchYpos - topPos) / 32 - 1;
if (item < 0)
    menuState := !menuState;
else if (menuState && item < 4)
    selectedItem := item;
endif
updateGauge();
```

As you may have noticed, there is a function used in the above code which we haven't defined yet. Go back to the **Global Magic Code** and add these lines.

```
func updateGauge()
    var frame, gaugeVal;
    gaugeVal := gaugeValue[selectedItem];
    frame := 4 * menuState + selectedItem + 8 * gaugeVal;
    WriteObject(tSmartGauge, 0, frame);
Endfunc
```

Add Multiple Input Objects

Add four slider or knob widgets to the project. The project included with this appnote uses four SmartSliders.



Add a Magic Event for each of the four input objects.

Form	Form0 ~					
Object	SmartSlider0	/				
Propert	Form0 MagicCode0					
Event	MagicCode1 MagicEvent0	1				
OnCha	MagicEvent1					
OnCha	MagicEvent2 MagicEvent3	J				
	MagicMove MagicRelease MagicTouch	T				

Set the **OnChanging** event of each input object to a Magic Event.

On event selection Report Message -Form0 MagicCode0Set MagicCode1Set			
MagicEvent0			
MagicEvent1 MagicEvent2 MagicEvent3 SmartGauge0Set SmartSlider1Set SmartSlider2Set SmartSlider3Set			
✔ ОК	S None	X Cancel	

Ensure that each of them has unique Magic Events

For each of the **Magic Event** objects. Insert these lines of code inside the function.

```
gaugeValue[n] := newval;
updateGauge();
```

You need to replace *n* with 0 to 3 so it is unique for each Magic Event

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
- <u>ViSi-Genie Getting Started First Project for Picaso Displays</u>
- <u>ViSi-Genie Getting Started First Project for Pixxi Display Modules</u>

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.