Programmers Guide to DDGui

(Second Edition)

A programmers guide to using the DDGui GUI system that comes with GLBasic

By Nicholas Kingsley

DDGui was written by Gernot Frisch

ISBN #: 978-1-4475-8191-8

Designed By Nicholas Kingsley

Revision : 2.0.0.0

For errors and omissions, I can be contacted at njk@un-map.com

Index

Part 1

DDgui

Introduction

What DDGui is

DDGui is designed to allow each platform supported by GLBasic to have a GUI system that looks, acts and used in the same way.

DDGui allows the creation of buttons, sliders, edit boxes, check and radio buttons in a window that can be moved

What DDGui isn't

DDGui isn't a fully fledged GUI system – there are many things that haven't been implemented within DDGui – for example positioning is done relative to other other items, as opposed to positioning to X and Y coordinates

Features of DDGui

DDGui has the following features :

- Windows
- Static text
- Buttons
- Sliders
- Toolbars
- Check boxes
- Radio Buttons
- List Box
- Combo
- File Dialog
- Text Boxes
- Tabs
- Frames
- Spacing

There are also some helper functions :

- Dialog Window
- Colour Picker
- Inserting and Deleting text
- Retrieving text
- Hiding and Showing

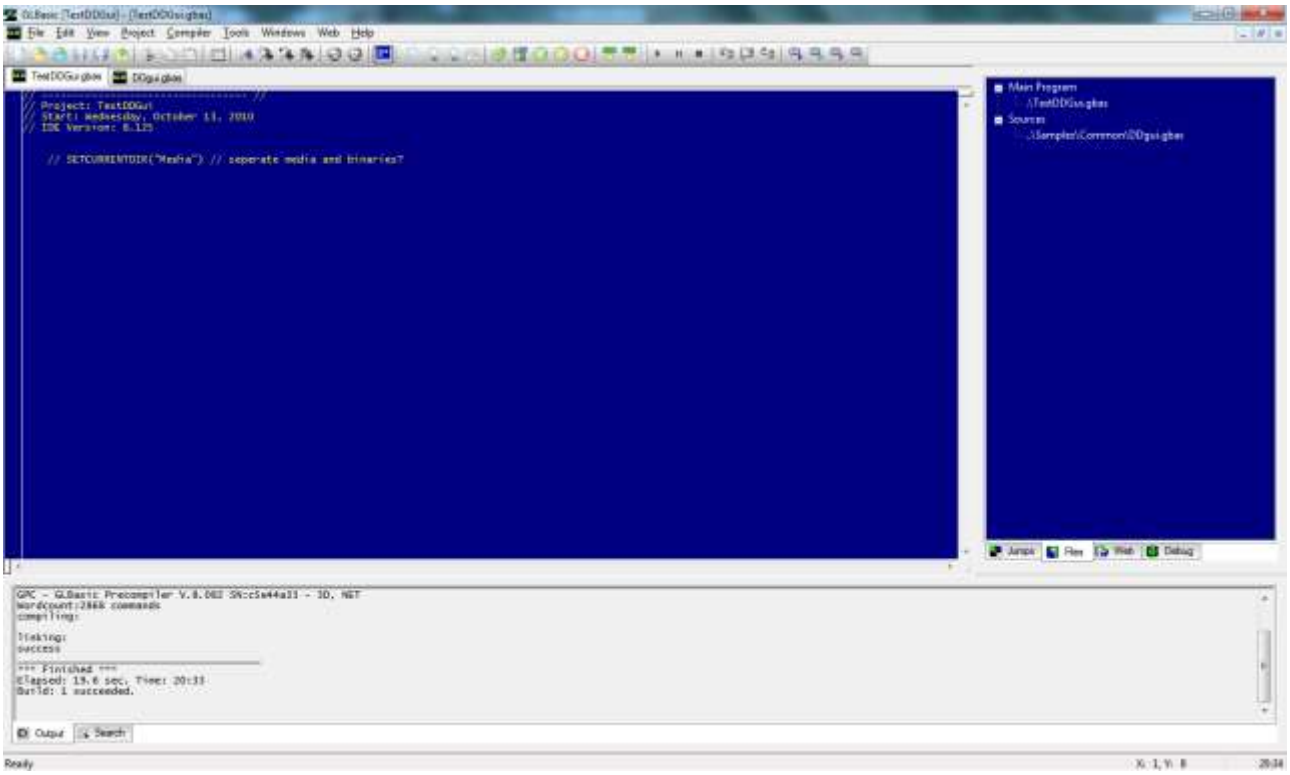
Other areas :

- User-defined routines
- Positioning
- Read-Only
- Tool-tips
- Input
- Progress Bar

Using DDGui

To start using DDGui in your programs, you need to add the DDGui.gbas file to your program. You can do this by adding the file from the Samples → Common to your project.

The file will then be added to the sources list. Your project layout could then look like :



You could compile the project like this, but it certainly won't do much, so the first thing we need to do is create a window, using the following :

```
DDgui_pushdialog(0,0,100,100,FALSE)
```

This creates a window 100 by 100 pixels in height, positioned at 0 pixels across and 0 pixels down from the top left corner of the screen. We make sure that the window is not centred to the middle of the screen

To keep showing the window, we need to call the show window command in a loop, which in this case is `DDgui_show`.

This command takes one parameter which determines whether all created windows are displayed, or just the last one. Like all graphics commands in GLBasic, `SHOWSCREEN` has to be called to swap screen buffers and update the display.

So, the routine will be :

```
WHILE TRUE
        DDgui_show(FALSE)
        SHOWSCREEN
WEND
```

The full program looks like :

```
DDgui_pushdialog(0,0,100,100)
WHILE TRUE
        DDgui_show(FALSE)
        SHOWSCREEN
WEND
```

The full code is :

```
DDgui_pushdialog(0,0,100,100)
WHILE TRUE
        DDgui_show(FALSE)
        SHOWSCREEN
WEND
```

And the results look like :

As mentioned in the *GLBasic Programmers Reference Guide*, the fault font makes text hard to read, so its always a good idea to create a readable font at this point, and place it in the with the executable (non-Mac), or for the Mac, place in the Media directory and manually load the font.

Text Display

From V9.054 all text display by default uses proportional text printing. This can be disabled (or re-enabled) by using `DDgui_UpdateFont`.

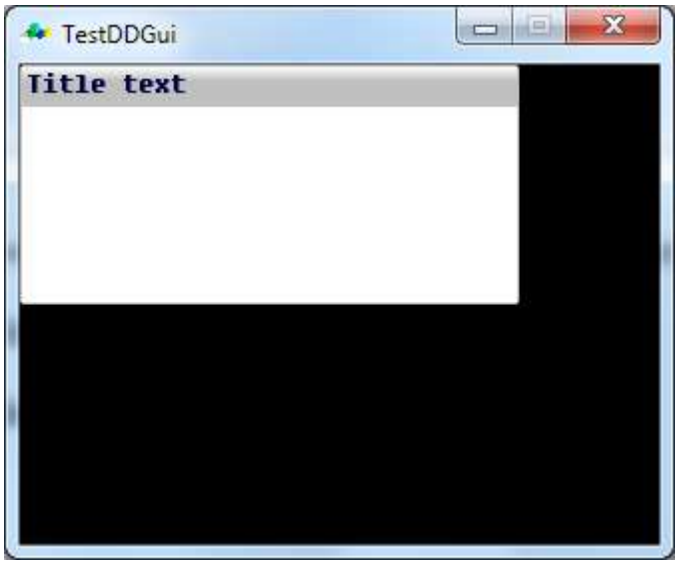
Windows Abilities

Titles

Windows can have titles, which can be set using DDgui_set, with an empty ID value :

```
DDgui_set("","TEXT","Title text")
```

Produces the following title :

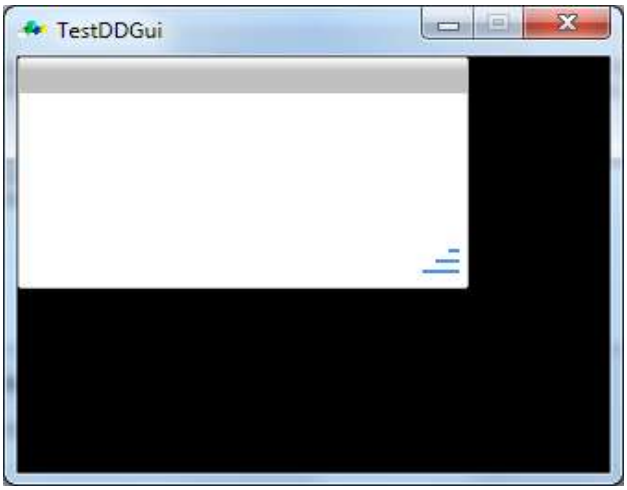


Allowing re-sizing

By default, a DDgui window can't be re-sized by the user. This ability can be activated by calling

```
DDgui_set("","SCALEABLE",TRUE)
```

This produces a window with a “gripper” (blue chevrons) on the bottom right of the window, which allow the window to be changed horizontally and/or vertically :



Allowing movement

You can allow the window to be moved by using :

```
DDgui_set("","MOVEABLE",TRUE)
```

Feedback

You can get feedback about the re-sizing and movement status by using `DDgui_get`, which is discussed in the reference section.

Multiple Windows

Multiple windows can be created with widgets. However, only the last created one will receive user input. In addition, there is currently no way of selecting another window.

ID's

All widgets (except for windows) require a unique string ID – this enables widgets to be uniquely identified and manipulated.

Static Text

To display static text, you would use :

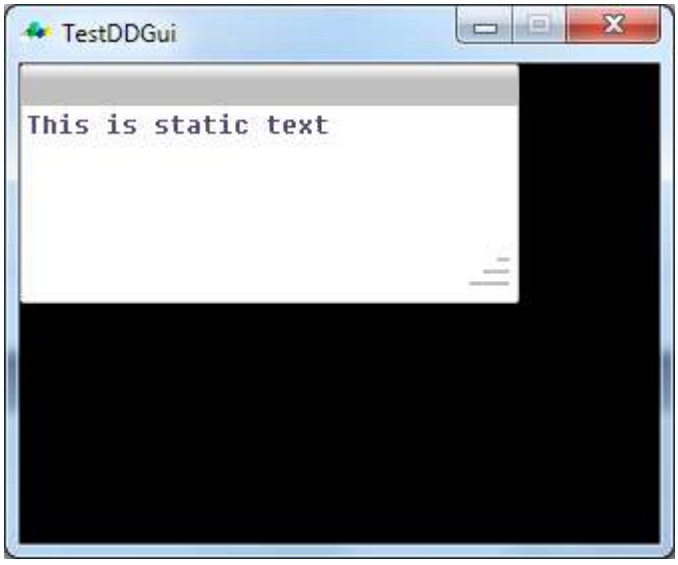
`DDgui_widget(id$, text$ [, width%, height%])`

with **id$** being a unique identifier, and **text$** is the text to be displayed.

width% and **height%** are optional. If these are given, the the widget will be created with the horizontal size defined by **width%** and the vertical size by **height%**.

If these values aren't given, the the widget with take the horizontal size of the text length and the font height.

Static text looks like :



Buttons

Buttons allow the user to select an option, and are created with :

`DDgui_button(id$, text$, [, width%, height%])`

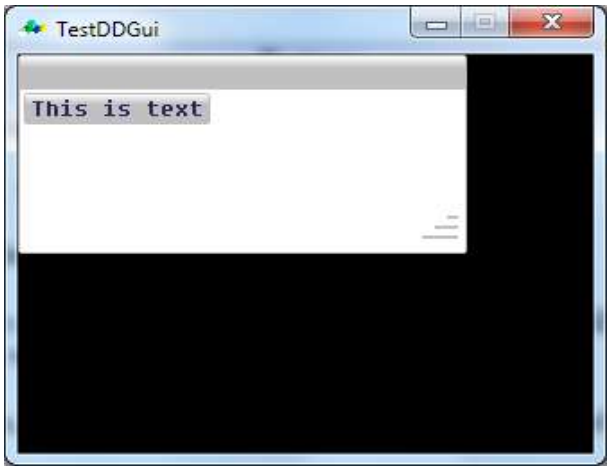
with **id$** being a unique identifier, and **text$** is the text to be displayed.

width% and **height%** are optional. If these are given, the the widget will be created with the horizontal size defined by **width%** and the vertical size by **height%**.

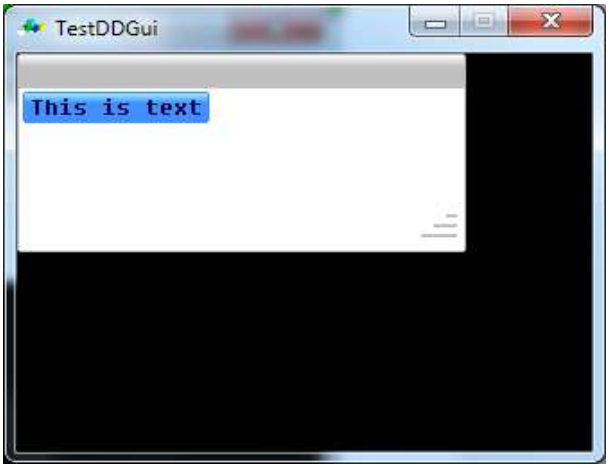
If these values aren't given, the the widget with take the horizontal size of the text length and the font height.

A button has two graphic modes. One when the mouse pointer is over it, and one when it isn't :

Mouse NOT over button



Mouse OVER button



Buttons can also have sprites in them. This is done by setting **text$** to "SPR_B" and appending the sprite index to it

With SPR_B, the sprite size (if **width%** and **height%** are 0) will be used to calculate the size of the button.

In addition, a button can have a filled colour. This is done by setting **text$** to "SPR_C" and appending an integer colour to it. The size of the button created is based on the
width% and **height%** values – if either of these are 0, then the default width (or height) is 32

SPR_C buttons, when pressed will open the Colour Dialog window.

This graphic shows a red graphic being displayed (using SPR_B), whilst the green button was created using SPR_C :

```
DDgui_button("test1","SPR_B0",0,0)
DDgui_button("test2","SPR_C121712",0,0)
```

Sliders

No, this isn't to do with the American Sci-Fi program – sliders are a horizontal bar that the user can can to specify a value.

Sliders can be created with :

`DDgui_slider(id$, value` *`[,width%, height%]`*`)`

with **id$** being a unique identifier, and **value** is the default value to be used.

width% and **height%** are optional. If these are given, the the widget will be created with the horizontal size defined by **width%** and the vertical size by **height%**.

By default, the slider range is between 0.0 and 1.0, with a step value of 0.01

The minimum range can be changed using DDgui_set and a MINVAL parameter, whilst the maximum value can be changed using DDgui_set and a MAXVAL parameter

Note :

- You need to make sure that the pointer position is always between the minimum and maximum range

A slider looks like :