Advanced Stata Dialog Programming with

Giovanni Luca Lo Magno
lomagno.gl@virgilio.it
Department of Agro-Environmental Systems
University of Palermo
Introducing Visua

RAD (rapid application development) software for Stata dialog programming

Provides you with a friendly GUI (graphical user interface) you can use to rapidly create Stata dialog boxes

Stand-alone software written in C++
A first look at Visua
1. Dialog programming basics
2. Visua basics
3. Advanced features of Visua:
   - debugger
   - plugins
Stata dialog programming basics

My dialog

Variable

Statistic
- Mean
- Standard deviation
- Range

width
height

group of radio buttons

help
reset
copy

launches the command and closes the dialog

launches the command and leaves the dialog open

OK
Cancel
Submit
Hello, world!

helloworld.dlg

VERSION 9.0
POSITION . . 360 40

DIALOG main, title(My first dialog box)
BEGIN
  TEXT tx_helloworld 10 10 . ., label(Hello, world!)
END

OK ok

Please, always add a blank line at the end of the script file
1) Save “helloworld.dlg” in a valid ado directory

2) Launch the dialog:
   db helloworld
Two common problems when programming dialogs

1) Unexpected behaviour of the dialog

2) The following error message:
The solution

Use **discard** before launching the dialog

```plaintext
discard
db hello world
```

**What **discard** does**

discard clears the class system memory and prevents the dialog engine from entering into an unstable state

**A useful tip**

```plaintext
program db_
syntax name(name=dialog name id="dialog name")
discard
db `dialog name'
end
```

**Use**

```plaintext
db_ helloworld
```
Hello, user!

hellouser.dlg

VERSION 9.0
POSITION . . 360 60

DIALOG main, title(Hello, user!)
BEGIN
  TEXT tx_question 10 10 . ., label(What's your name?)
  EDIT ed_name @ +20 340 .
END

OK ok

PROGRAM command
BEGIN
  put `"display "Hello, "' main.ed_name `"!"'``
END
The “Hello, user!” dialog box in action

1) The user clicks on the OK button

2) The return string is constructed

3) The return string is executed:

   display "Hello, Francis Galton!"
   Hello, Francis Galton!
The “Hello, world!” dialog box in Visua

Drag the “Version” object from the object box and drop it into the object list

This code line is automatically generated
The “Hello, world!” dialog box in Visua

Drag the “Position” object from the object box and drop it into the object list

This code line is automatically generated by Visua
The “Hello, world!” dialog box in Visua

Drag the “Dialog” object from the object box and drop it into the object list.

These code lines are automatically generated by Visua.
The “Hello, world!” dialog box in Visua

Select the “Dialog” object and modify its properties (double click on each value to modify it)
The “Hello, world!” dialog box in Visua

Right click on the “Dialog” object, “Add control” ⇒ “Text”
The "Hello, world!" dialog box in Visua

The visual appearance of the just created "Text" object

Code:

```
VERSION 9.0
POSITION . . 400 300
DIALOG main, title(My first dialog box)
BEGIN
  TEXT text1 0 0 .., label(Text)
END
```
Select the “Text” object and modify its properties
Drag the “Ok” object from the object box and drop it into the object list.
Click on the “Test dialog in Stata” button to test the dialog
This message will only appear the first time you try testing the dialog in Stata.

Click on the “Find...” button
Choose the Stata executable ("wsestata.exe" on my pc)
How the test process works

1) **Visua**

1.1) Generates *visatest.dlg*
1.2) Calls Stata to execute *visatest.dlg*

2) **Stata**

2.1) Executes *visatest.dlg*
2.2) Generates the log file *visatest.log*

**Note:**
*visatest.dlg* and *visatest.log* are temporary files stored in the “Visua 0.1 beta\Visuatest” folder
Click on the “View last Stata log file” button
Exporting to a .dlg file

“File” ⇒ “Export...”
Debugging
**Debugging with Visua**

![Debug report button]

**Strengths**
- User-friendly error messages
- Detection of multiple issues
- Warnings about non-critical issues

**Weaknesses**
Currently, only some errors are detected
WHERE IS THE ERROR?

DEBUGEXAMPLE.DLG

VERSION 9.0
POSITION . . 400 100

LIST mylist
BEGIN
  mean
  range
END

DIALOG main, title(My dialog box)
BEGIN
  TEXT tx_choice 10 10 . ., label(Make your choice:)
  COMBOBOX tx_choice @ +20 200 100, dropdownlist contents(mylist) ///
    onselchangelist(action_list)
END

PROGRAM command
BEGIN
  put `"display "Your main choice is "' main.cb_choice `""'`
END

OK ok
SUBMIT submit
CANCEL cancel
Error messages from Stata and Visua

**Obscure and unhelpful error message**

```
. discard
db debugexample
class types are not the same
r(4015);
```

**More informative error message**

Visua - Debug report
Issues:

In dialog "main": there are 2 objects with the same name "tx_choice"

More on issue:
Every object in a dialog must have an unique name. Rename objects assigning them an unique name

**Clear and user-friendly explanation about the issue**
Plugins
Plugins extend the capabilities of Visua

They can save a lot of manually written code

They are .dll files placed in the “plugins” folder

The user can write plugins (in C++)
An example: “List of U.S. states” plugin

The generated code

```
LIST list_of_us_states
BEGIN
   Alabama
   Alaska
   Arizona
   ...  
   Wyoming
END
```

53 lines of code were generated
The “Survey maker” plugin creates a dialog box which allows the user to insert complex survey data directly into Stata.

- Items are disabled, depending on the response from the previous questions.
- Non-consistent data entries are avoided.
The “Survey maker” plugin: a filter question

1. Have you ever been married? [married]
   - Yes
   - No

2. In what year did you get married? [year]

3. Were you born in Italy or abroad? [country]

Consistent record:

<table>
<thead>
<tr>
<th>married</th>
<th>year</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1996</td>
<td>1</td>
</tr>
</tbody>
</table>

Non-consistent record: the dialog will avoid this
The Survey Maker plugin in action

Jump to item 3 if married = 2 (No)
A simplified labour force survey

1. Did you work for at least one hour in the previous week?
   - Yes
   - No

2. Did you in any case have a job you did not perform for any particular reason?
   - Yes
   - No

3. What is the main reason why you did not work in that week?
   - Till three months

4. Will this period of absence last more or less than three months from start to finish?
   - Yes, 50% or more
   - Three months or more

5. Is this period of absence paid at least in part?
   - Yes, less than 50%
   - Yes, 50% or more
   - No, it is unpaid

(continue)

End

Item 9
A simplified labour force survey

6. Would you tell me the name of your occupation?

7. What is your employment status?
   - Entrepreneur
   - Professional worker

8. Do you have employees?

9. Have you actively sought employment in the last 4 weeks?

10. Are you available for work in the next two weeks?

End
A simplified labour force survey

Did you work at least one hour in the previous week?

Options

<table>
<thead>
<tr>
<th>Label</th>
<th>Text</th>
<th>Value</th>
<th>Jump to</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>1</td>
<td>end</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
A simplified labour force survey

1. Did you work at least one hour in the previous week?
   - 1 Yes
   - 2 No

2. Did you in any case have a job you did not perform for any particular reason?
   - 1 Yes
   - 2 No

3. What is the main reason why you did not work in that week?
   - 1 Lay-off (ordinary or extraordinary)
   - 2 Other reasons
Generated code
412 lines of code were automatically generated!

Check before data insertion
Before inserting data into the dataset, the dialog box checks if:
- the variables exist
- the variables are of the expected type

Tip
Add a missing value option to each item if you want to allow missing value data
A simplified labour force survey (missing values are allowed)
Why should you use Visua?

- Speed-up the code writing process
- Avoid numerous syntax errors
- Easy development environment
- What you see is what you get
- Great power with plugins
Thank you for your attention
Appendix A – How to write Visua plugins
What do you need to write Visua plugins?

- Proficiency in the use of the C++ programming language and the Qt 4 programming framework

- The Qt 4.x SDK (software development kit) ([http://qt.nokia.com](http://qt.nokia.com))

- A reference to the Qt Plugin Framework:

- The *visuapuginerface.h* file (see next slide)
#ifndef VISUAPLUGININTERFACE_H
#define VISUAPLUGININTERFACE_H

#include <QtGui>

class VisuaPluginInterface
{
public:
enum Behaviour {
    CreateNewVisuaDocument,
    AppendAlways,
    AppendToDialogOnly,
    AppendOutsideDialogOnly};

virtual ~VisuaPluginInterface() {};
virtual QString visuaCode() = 0;
virtual QString pluginName() = 0;
virtual int dialogExec(QWidget *parent) = 0;
virtual bool isDialog() = 0;
virtual VisuaPluginInterface::Behaviour behaviour() = 0;
};

Q_DECLARE_INTERFACE(VisuaPluginInterface, "Visua.VisuaPluginInterface/1.0")

#endif // VISUAPLUGININTERFACE_H
Example: let’s create the “Hello, world!” plugin

The files
- helloworld.h
- helloworld.cpp
- helloworld.pro

The Visua code output generated by the plugin

```
<visua version="0.1">
  <object type="text">
    <name>hello_world_text</name>
    <x>20</x>
    <y>20</y>
    <xsize>100</xsize>
    <ysize>.size
    <label>Hello, world!</label>
  </object>
</visua>
```

How it will be “translated” by Visua

```
TEXT hello_world 20 20 100 ., label(Hello, world!)
```
#ifndef HELLOWORLD_H
#define HELLOWORLD_H

#include <QtGui>
#include "visuapugininterface.h"

class HelloWorld : public QObject, public VisuaPluginInterface
{
    Q_OBJECT
    Q_INTERFACES(VisuaPluginInterface)

public:
    QString visuaCode();
    QString pluginName();
    int dialogExec(QWidget *parent);
    bool isDialog();
    VisuaPluginInterface::Behaviour behaviour();
};

#endif // HELLOWORLD_H
Example: the “Hello, world!” plugin – implementation file

helloworld.cpp (part 1 of 2)

```cpp
#include "helloworld.h"

QString HelloWorld::visuaCode()
{
    QString code;
    code += "<visua version="0.1">";
    code += "<object type="text">\n";
    code += "<name>hello_world_text</name>\n";
    code += "<x>20</x>\n";
    code += "<y>20</y>\n";
    code += "<xsize>100</xsize>\n";
    code += "<ysize>.</ysize>\n";
    code += "<label>Hello, world!</label>\n";
    code += "</object>\n";
    code += "</visua>";
    return code;
}

QString HelloWorld::pluginName()
{
    return "Hello, world!";
}

(continue)```
int HelloWorld::dialogExec(QWidget *parent)
{
    return 0;
}

bool HelloWorld::isDialog()
{
    return false;
}

VisuaPluginInterface::Behaviour HelloWorld::behaviour()
{
    return VisuaPluginInterface::AppendAlways;
}

Q_EXPORT_PLUGIN2(helloworld, HelloWorld)
Example: the “Hello, world!” plugin – Qt project file

helloworld.pro

```
TEMPLATE = lib
CONFIG += plugin

HEADERS += helloworld.h

SOURCES += helloworld.cpp
```
Example: the “Hello, world!” plugin – compilation and use

Compilation
The build process will end with a “helloworld.dll” file

Use
1) Put the “helloworld.dll” into the “plugins” folder under the Visua program folder
2) Restart Visua
3) The “Hello, world!” plugin will appear under the “Plugins” menu
virtual QString VisuaPluginInterface::visuaCode() = 0
  Returns the Visua code. The string must begin with the <visua version="0.1"> tag and end with the </visua> tag.

virtual QString VisuaPluginInterface::pluginName() = 0
  Returns the name of the plugin. It’s the name which will appear on the “Plugins” menu in Visua.

virtual QDialog::DialogCode VisuaPluginInterface::dialogExec(QWidget *parent) = 0
  This function is automatically called by Visua if the plugin has a dialog window. The dialog is always modal. The function returns a QDialog::DialogCode code, which can be QDialog::Accepted or QDialog::Rejected and it should be associated with the user final action with the dialog window. If the return value is QDialog::Rejected, the plugin has no effect on the Visua document.

virtual bool VisuaPluginInterface::isDialog() = 0
  The return value of this function must be set to “true” if the plugin has a dialog that will be executed when the plugin is called. Alternatively, the return value must be set to zero.

virtual VisuaPluginInterface::Behaviour behaviour() = 0
  The return value must be one of the following: Behaviour::CreateNewVisuaDocument, Behaviour::AppendAlways, Behaviour::AppendToDialogOnly, Behaviour::AppendOutsideDialogOnly. The return value controls the plugin behaviour i.e. where the Visua code can be appended and if the current document will be replaced after the execution of the plugin.
How to write the Visua code: an initial glance

Root opening tag

Object

Object

Object

Root closing tag

```
<visua version="0.1">
  <object type="version">
    (properties)
  </object>
  <object type="position">
    (properties)
  </object>
  ...
  ...
  ...
  <object type="dialog">
    (properties)
  </object>
</visua>
```
An example of an EDIT object

Name of the object type (*)

Object opening tag

<object type="edit">
  <name>ed_myedit</name>
  <x>0</x>
  <y>+20</y>
  <xsize>200</xsize>
  <ysize>.</ysize>
  <max>99</max>
  <numonly>numonly</numonly>
</object>

Properties

Object closing tag

(*) It’s the name of the control in all lower-case letters. For example, use “combobox” for the COMBOBOX object.
Special object: DIALOG

Object opening tag  
<object type="dialog">
  <name>main</name>
  <title>My first dialog box</title>
  <tabtitle>Tab 1</tabtitle>
  <children>
    <object type="text">
      (properties)
    </object>
    <object type="edit">
      (properties)
    </object>
    ...
    <object type="combobox">
      (properties)
    </object>
  </children>
</object>

Children objects

Content opening tag  
<title>My first dialog box</title>
	<tabtitle>Tab 1</tabtitle>
<br>

Children closing tag  
</children>

Object closing tag  
</object>
Special object: LIST

Object opening tag
Content opening tag
Items
Content closing tag
Object closing tag

```
<object type="list">
  <name>li_mystlist</name>
  <content>
    <listitem>item_1</listitem>
    <listitem>item_1</listitem>
    ...
    <listitem>item_n</listitem>
  </content>
</object>
```
Special object: SCRIPT

Object opening tag → `<object type="script">
   <name>myscript</name>
   <iactionlist>
      <iactionlistitem>main.tx.hide</iactionlistitem>
      <iactionlistitem>main.ed.show</iactionlistitem>
      ...
      <iactionlistitem>main.cb.show</iactionlistitem>
   </iactionlist>
</object>`

I-action list opening tag → `<iactionlist>
   <iactionlistitem>main.tx.hide</iactionlistitem>
   <iactionlistitem>main.ed.show</iactionlistitem>
   ...
   <iactionlistitem>main.cb.show</iactionlistitem>
</iactionlist>`

Script lines → `...`

I-action list closing tag → `</iactionlist>`

Object closing tag → `</object>`
Special object: PROGRAM

<Object opening tag> <object type="program">
  <name>myprogram</name>
  <code>
    <codeline>put "mean " main.ed_1</codeline>
    <codeline>put "mean " main.ed_2</codeline>
    ...
    <codeline>put "mean " main.ed_n</codeline>
  </code>
</Object closing tag>
Example: the Visua code for the “Hello, world!” dialog (helloworld.vis)

```xml
<visua version="0.1">
    <object type="version">
        <versionnumber>9.0</versionnumber>
    </object>
    <object type="position">
        <x>.</x>
        <y>.</y>
        <xsize>360</xsize>
        <ysize>40</ysize>
    </object>
    <object type="dialog">
        <name>main</name>
        <title>My first dialog box</title>
        <children>
            <object type="text">
                <name>tx_helloworld</name>
                <x>0</x>
                <y>0</y>
                <xsize>.</xsize>
                <ysize>.
                <label>Hello, world!</label>
            </object>
        </children>
    </object>
    <object type="ok">
        <name>ok</name>
    </object>
</visua>
```
List of objects supported in the Visua plugin programming framework

- VERSION
- INCLUDE
- DEFINE
- POSITION
- LIST
- DIALOG
- CHECKBOX
- RADIO
- SPINNER
- EDIT
- VARLIST
- VARNAME
- FILE
- LISTBOX
- COMBOBOX
- TEXT
- TEXTBOX
- GROUPBOX
- FRAME
- COLOR
- EXP
- OK
- SUBMIT
- CANCEL
- COPY
- HELP
- RESET
- SCRIPT
- PROGRAM
Sorry, there is no complete official documentation

You can learn more by looking at the .vis files you can generate using Visua. You can open them with a plain text editor (for example Windows Notepad)

**Warning**
The tags may change in future version of Visua
Appendix B – Visua templates
What are templates?

- A template is a model you can use as a starting point to develop your own dialog

- Templates are accessible from “File” ⇒ “New from template”

- They are .vis files in the “templates” folder

- You can add new templates to the “templates” folder (restart Visua to load them)
The “Standard dialog with if-in dialog tab” template

Start with a dialog with a “if/in” tab

Save time