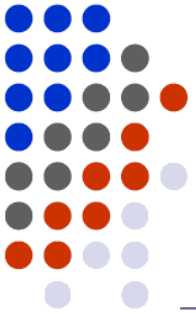


IX ITALIAN STATA USERS GROUP MEETING
Bologna, 20-21 September 2012

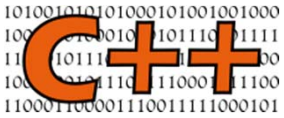
Advanced Stata Dialog Programming with VISUA

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

Introducing Visua

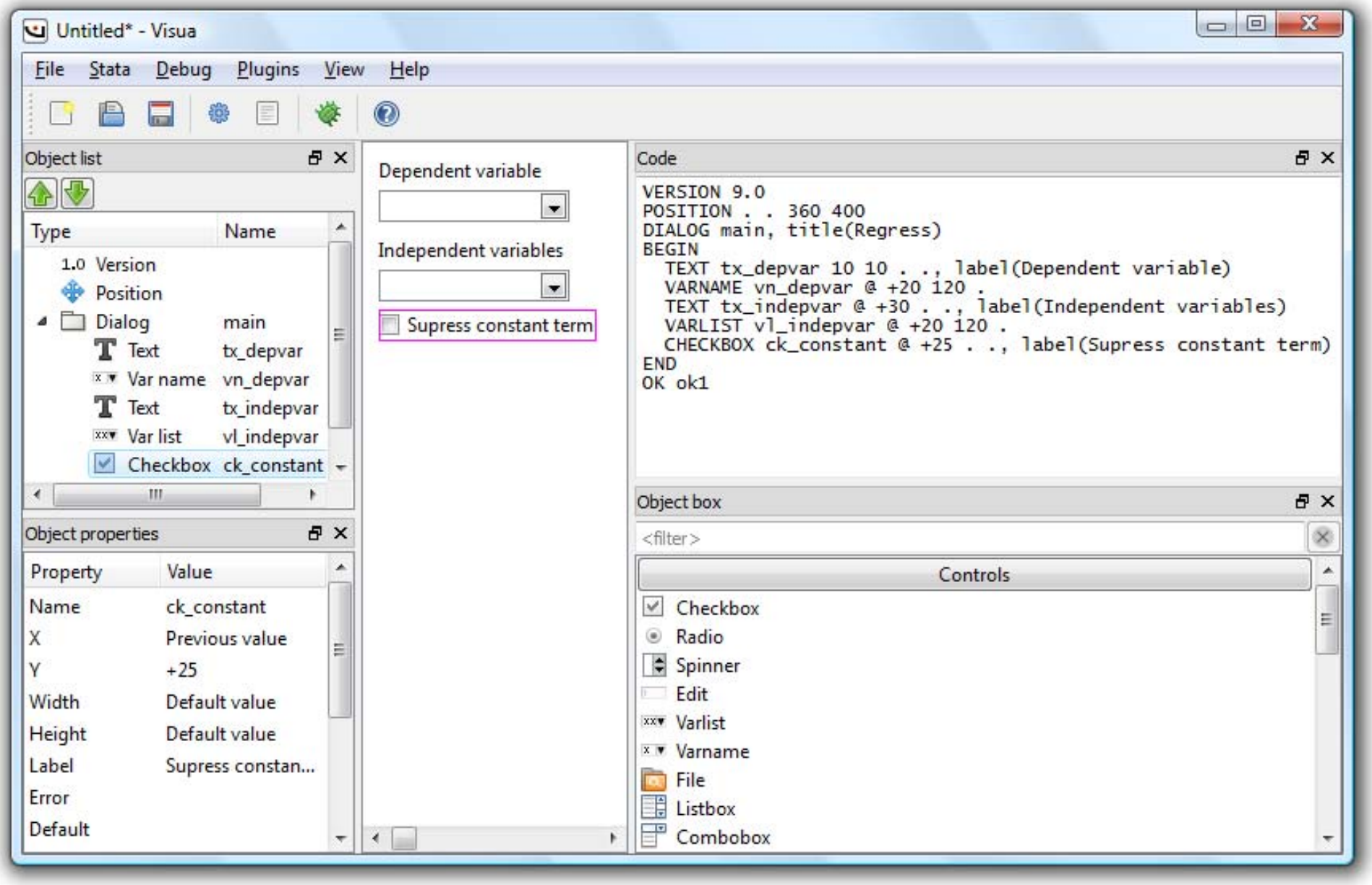
RAD (rapid application development) software for Stata dialog programming

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

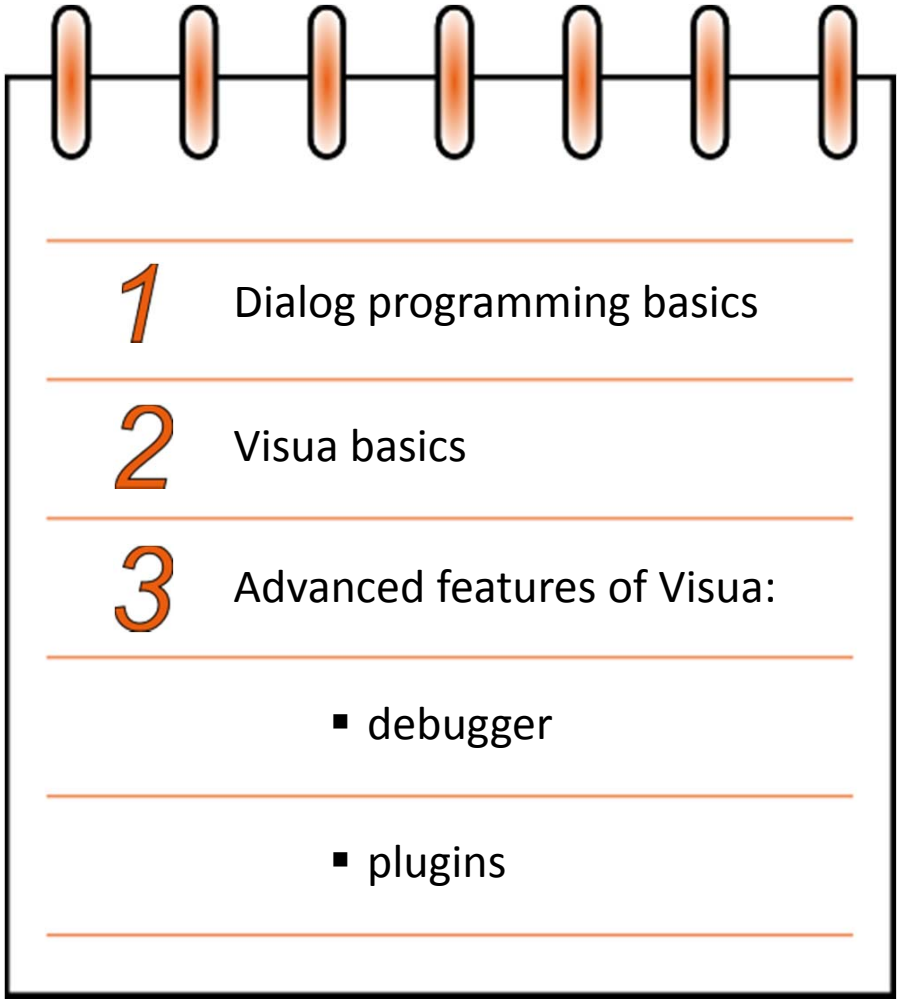


Stand-alone software written in C++

A first look at Visua

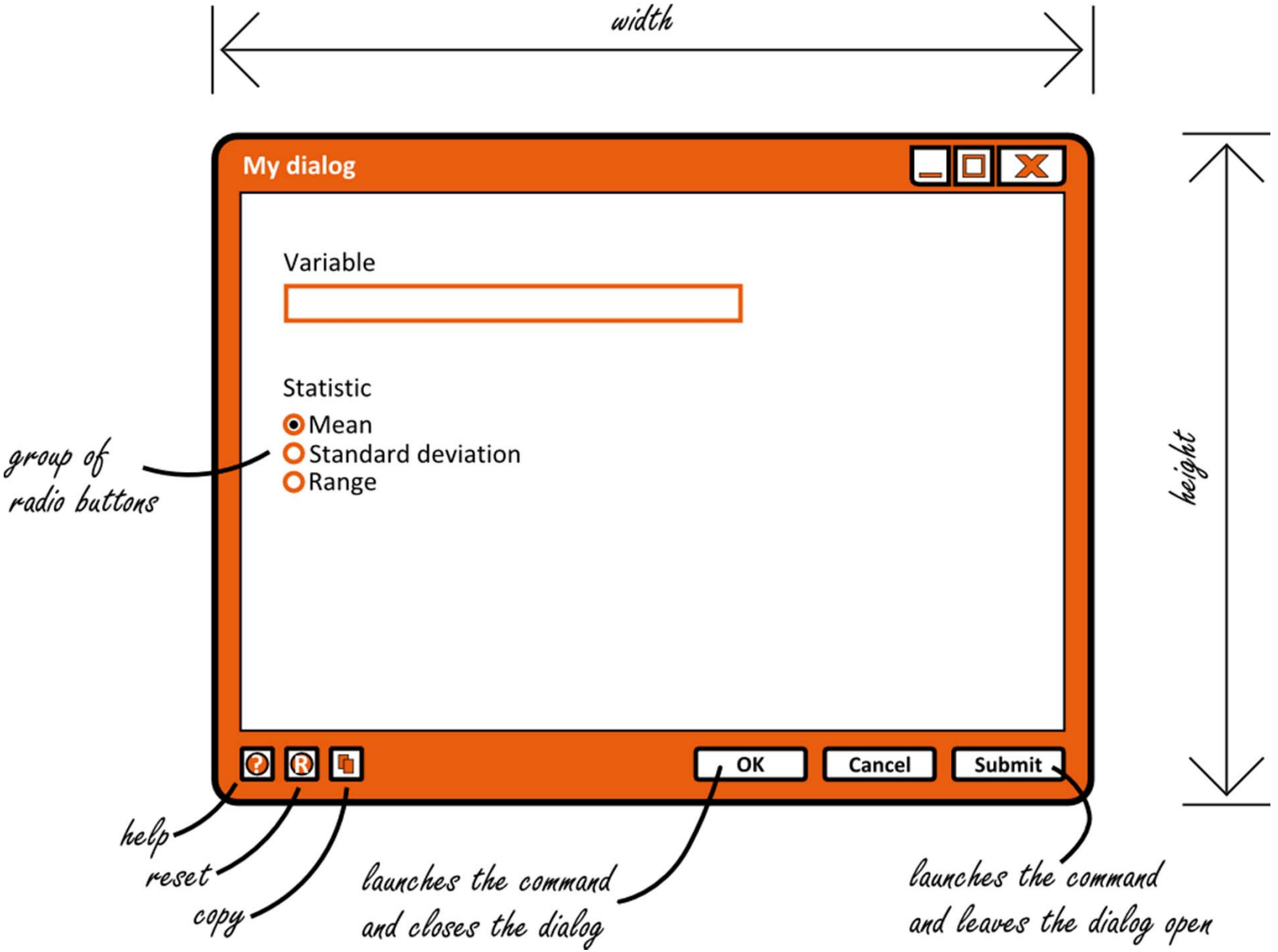


Presentation outline



1. Dialog programming basics
2. Visua basics
3. Advanced features of Visua:
   - debugger
   - plugins

Stata dialog programming basics

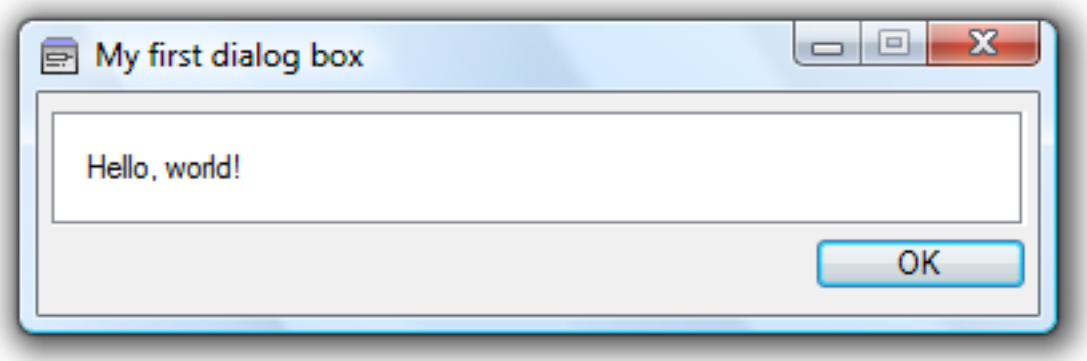


Hello, world!

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



How to execute the “Hello, world” dialog box

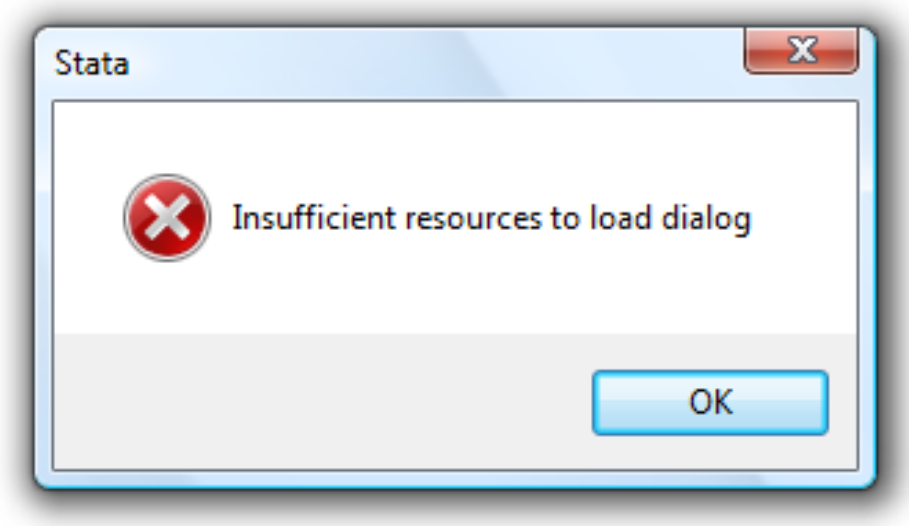
1) Save “helloworld.dlg” in a valid ado directory

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

```
discard
db helloworld
```

What `discard` does

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

A useful tip

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

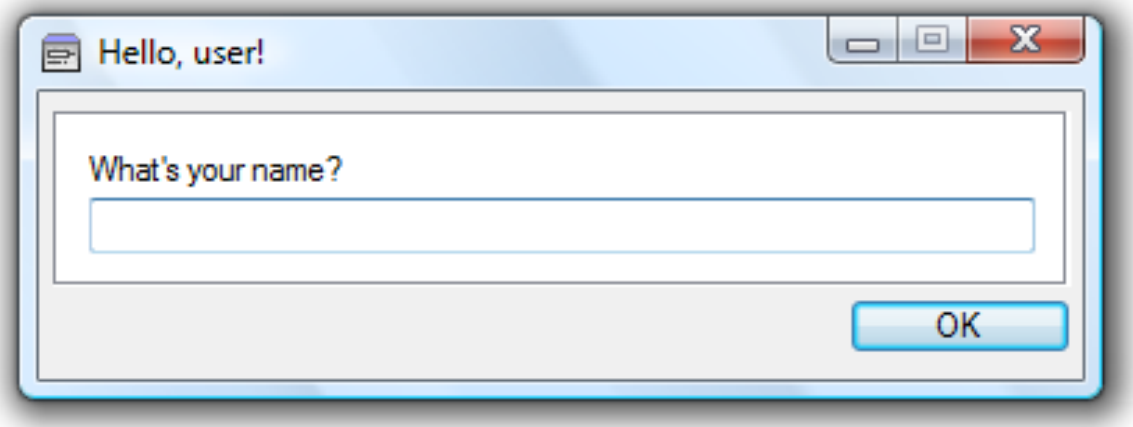
Use

```
db_ helloworld
```

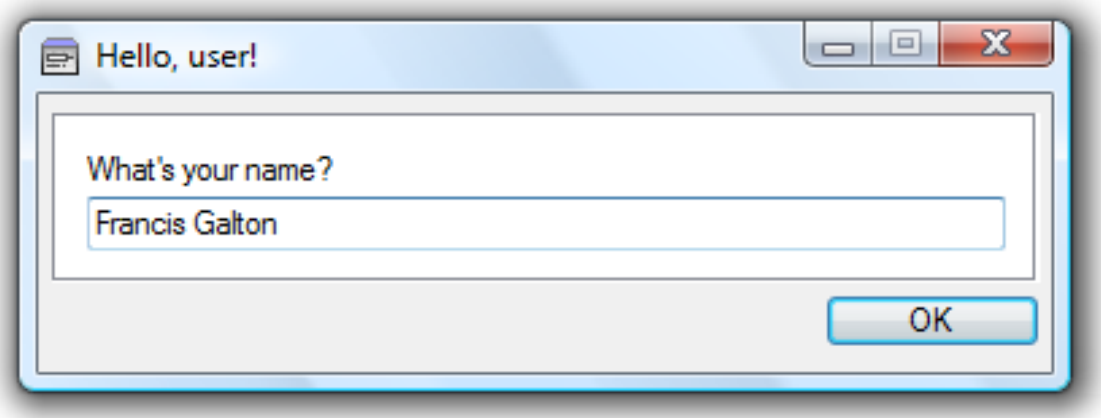
Hello, user!

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



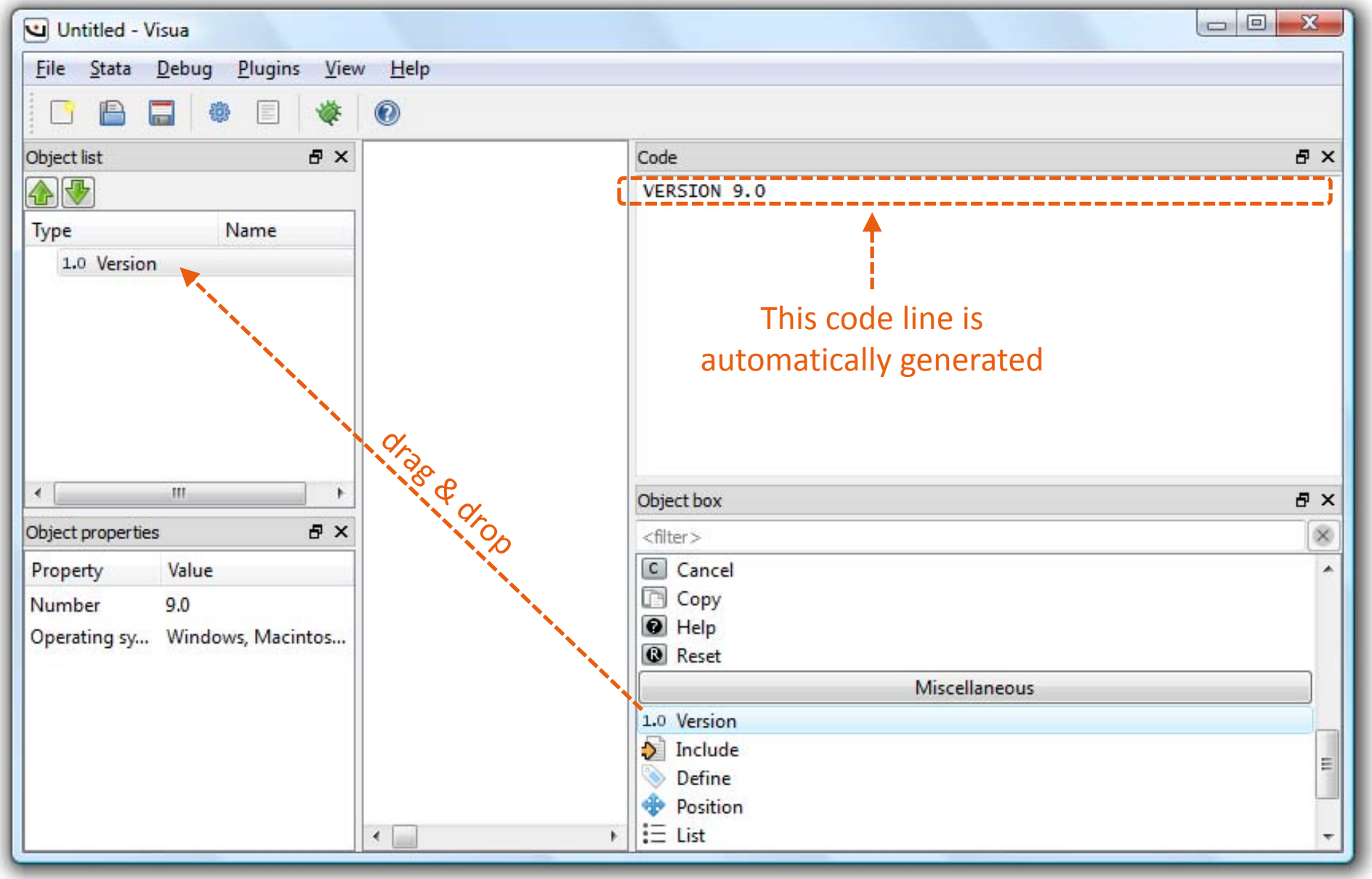
1) The user clicks on the OK button

2) The return string is constructed

3) The return string is executed:

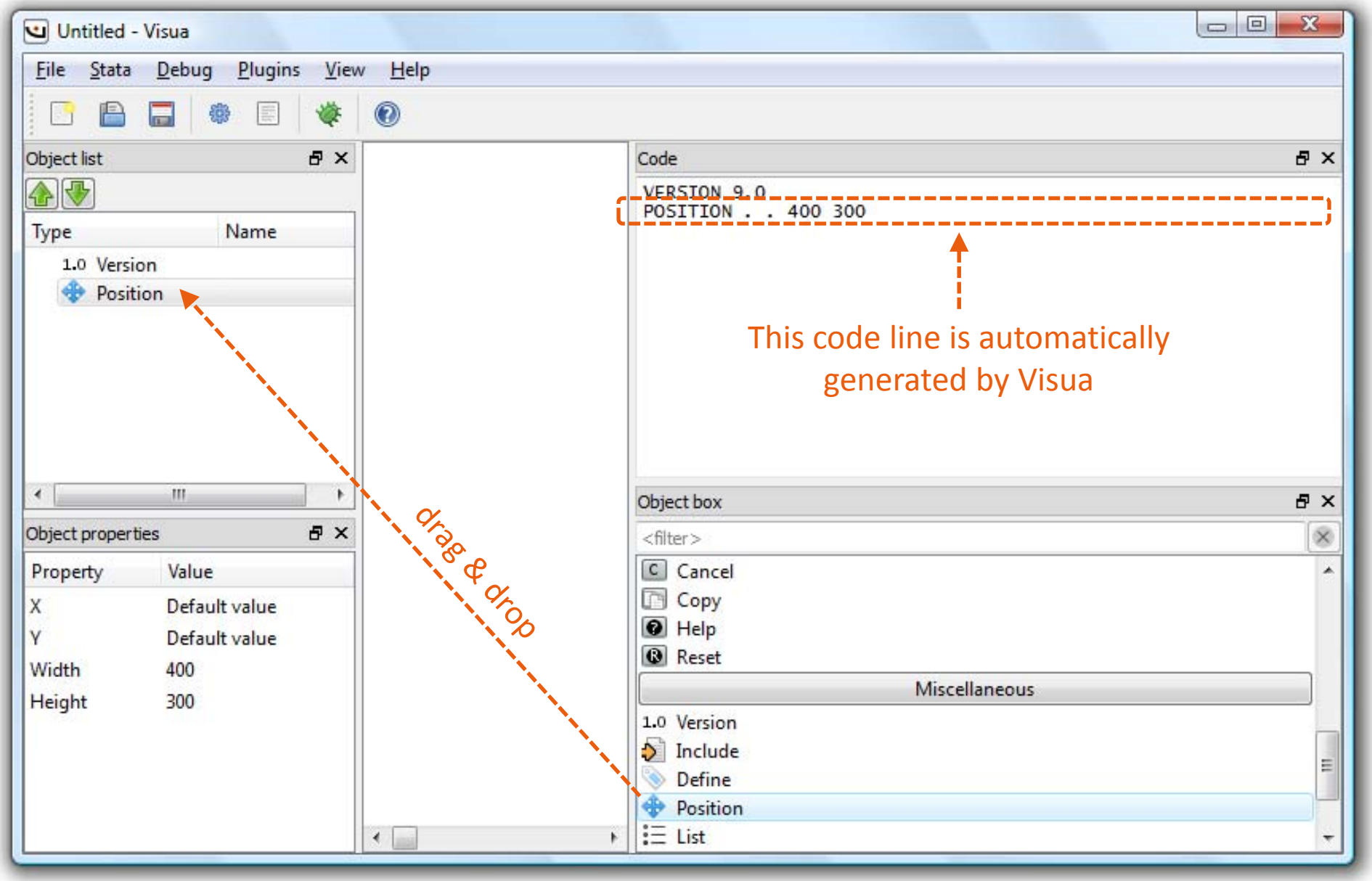
```
. display "Hello, Francis Galton!"
Hello, Francis Galton!
```

The “Hello, world!” dialog box in Visua



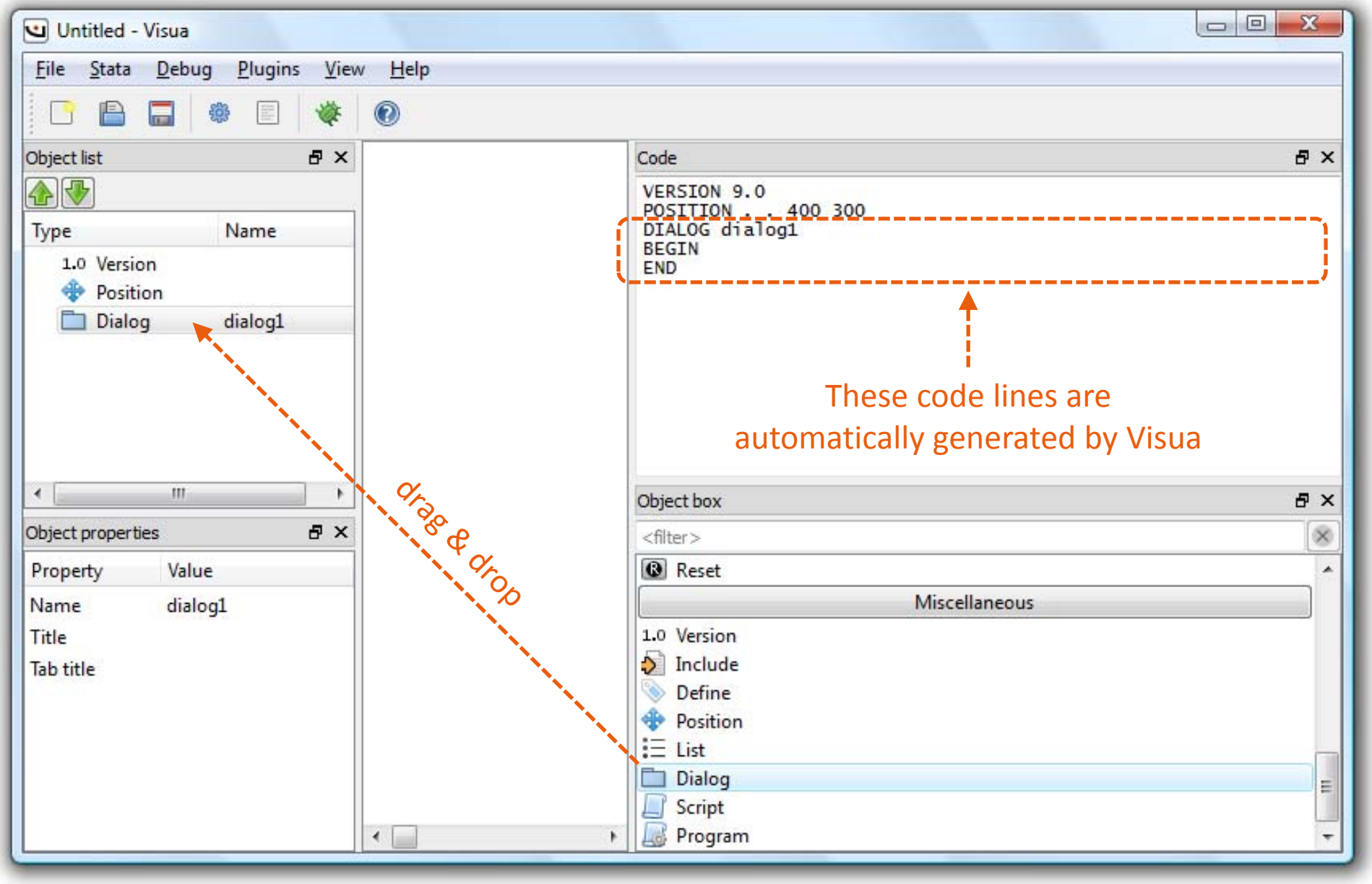
Drag the “Version” object from the object box and drop it into the object list

The “Hello, world!” dialog box in Visua



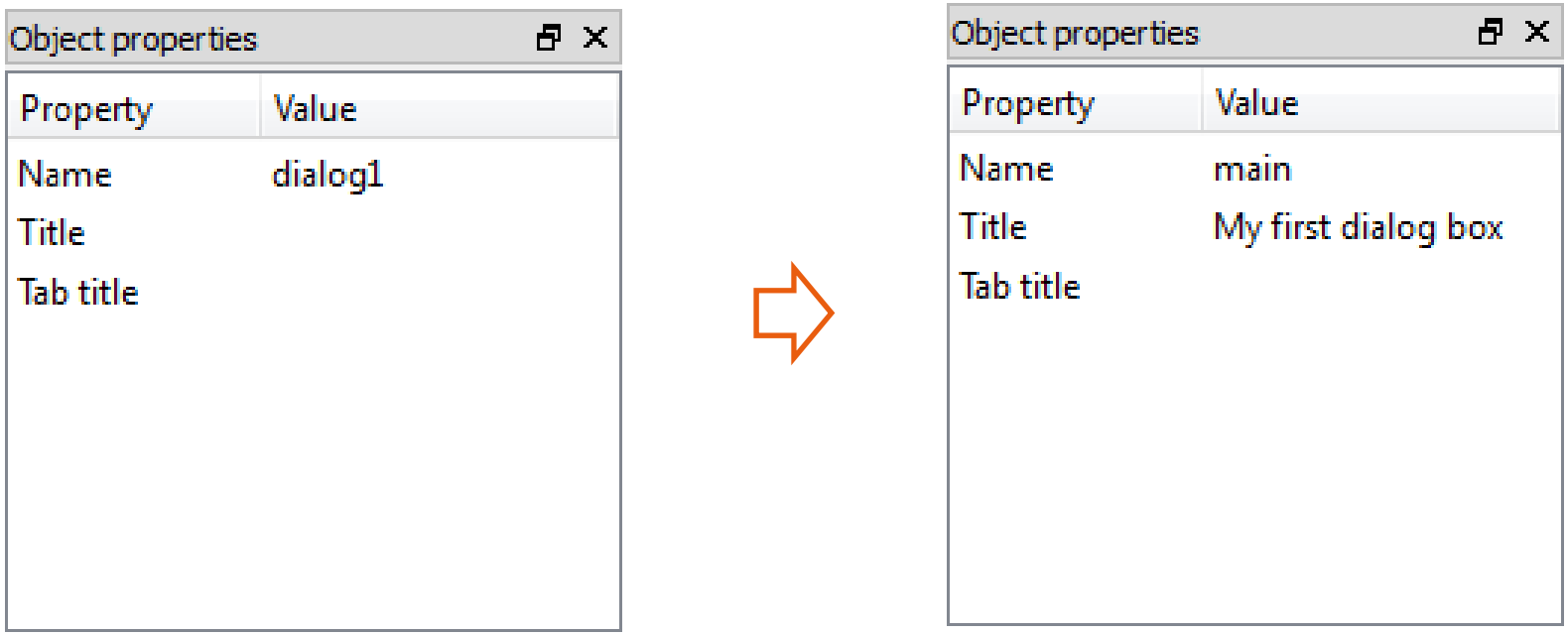
Drag the “Position” object from the object box and drop it into the object list

The “Hello, world!” dialog box in Visua



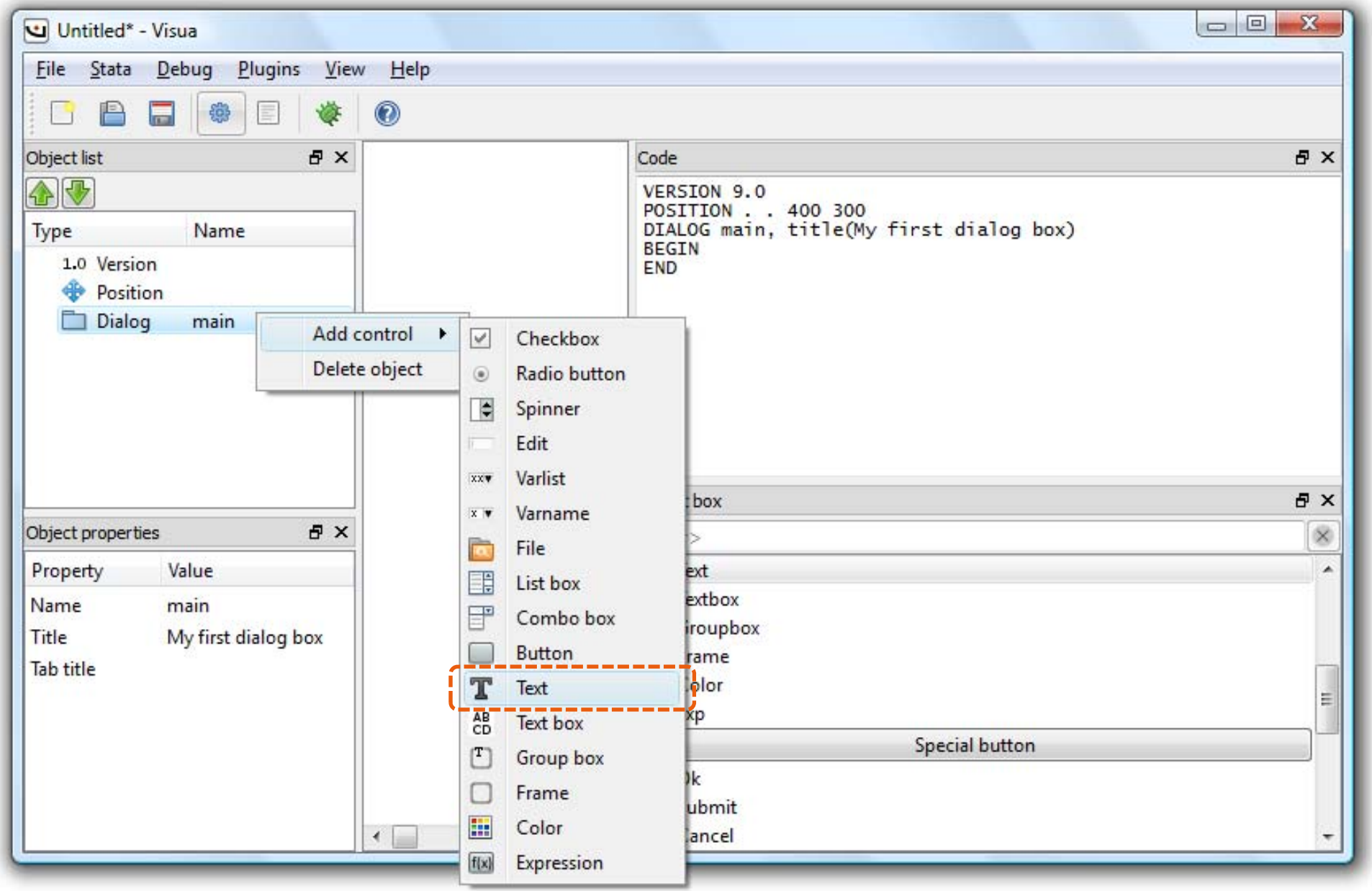
Drag the “Dialog” object from the object box and drop it into the object list

The “Hello, world!” dialog box in Visua



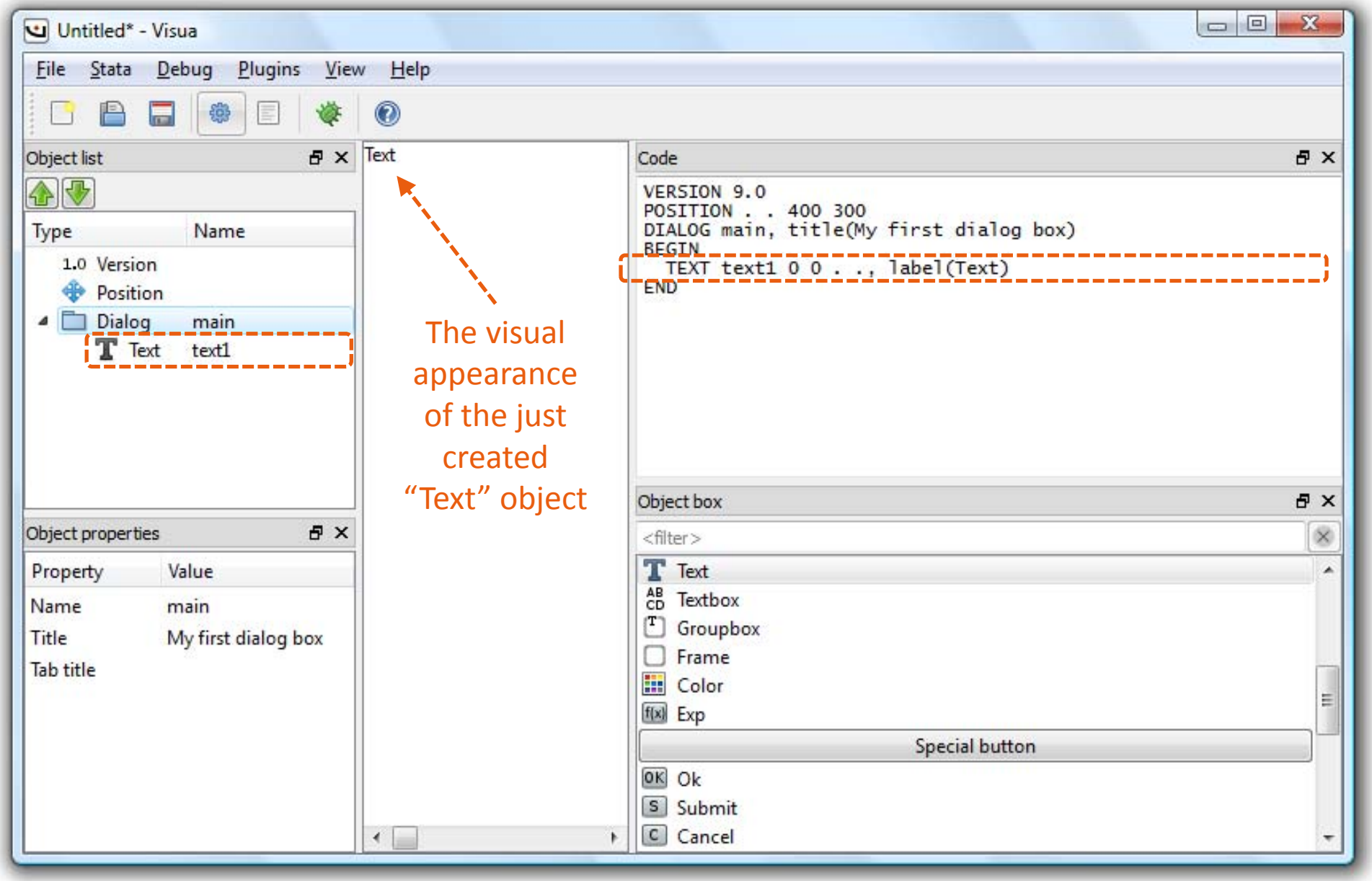
Select the “Dialog” object and modify its properties (double click on each value to modify it)

The “Hello, world!” dialog box in Visua

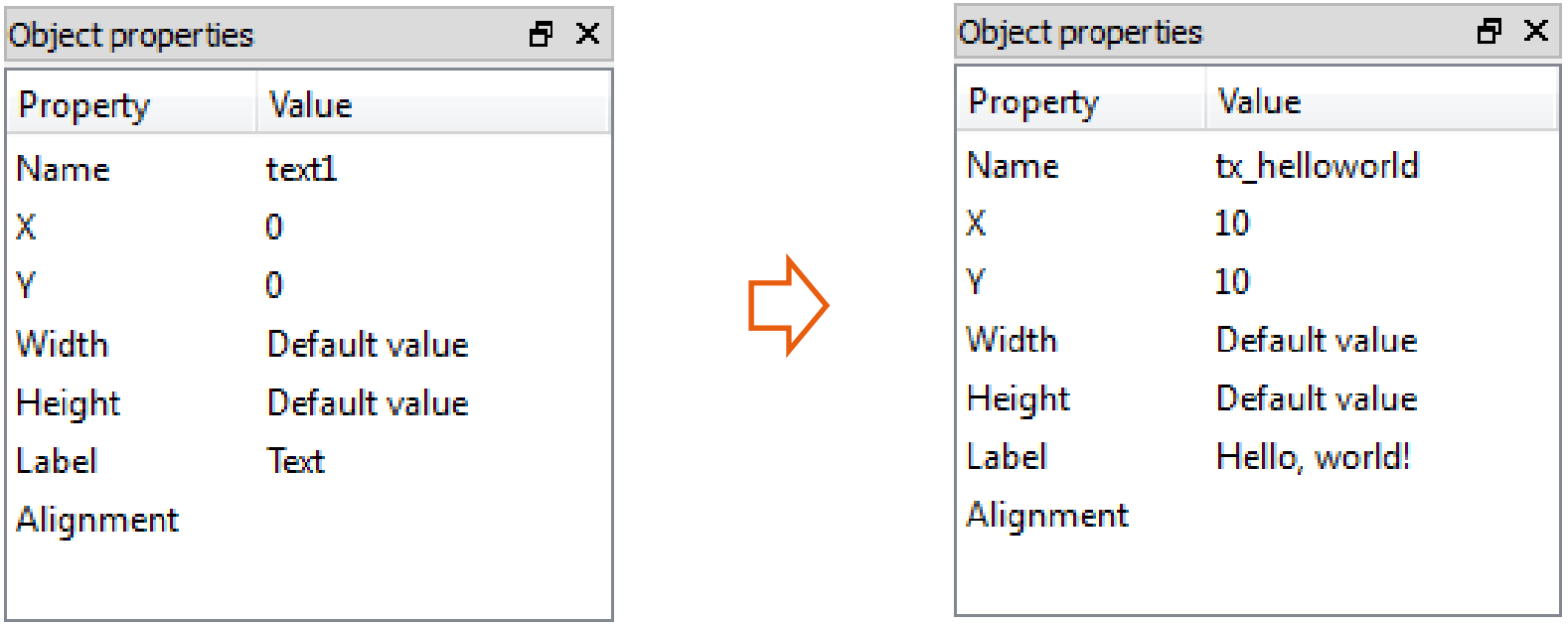


Right click on the “Dialog” object, “Add control” ⇨ “Text”

The “Hello, world!” dialog box in Visua

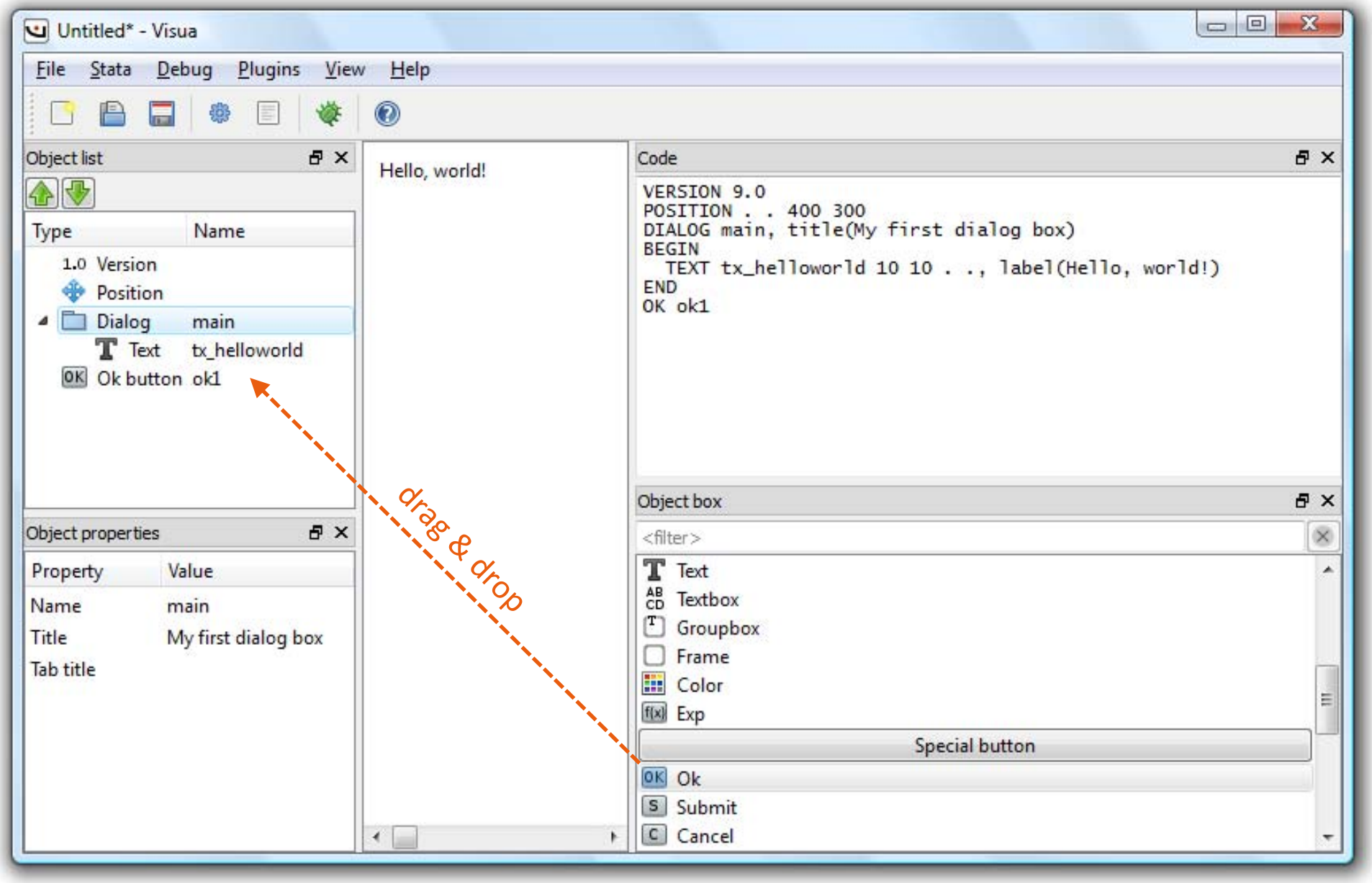


The “Hello, world!” dialog box in Visua



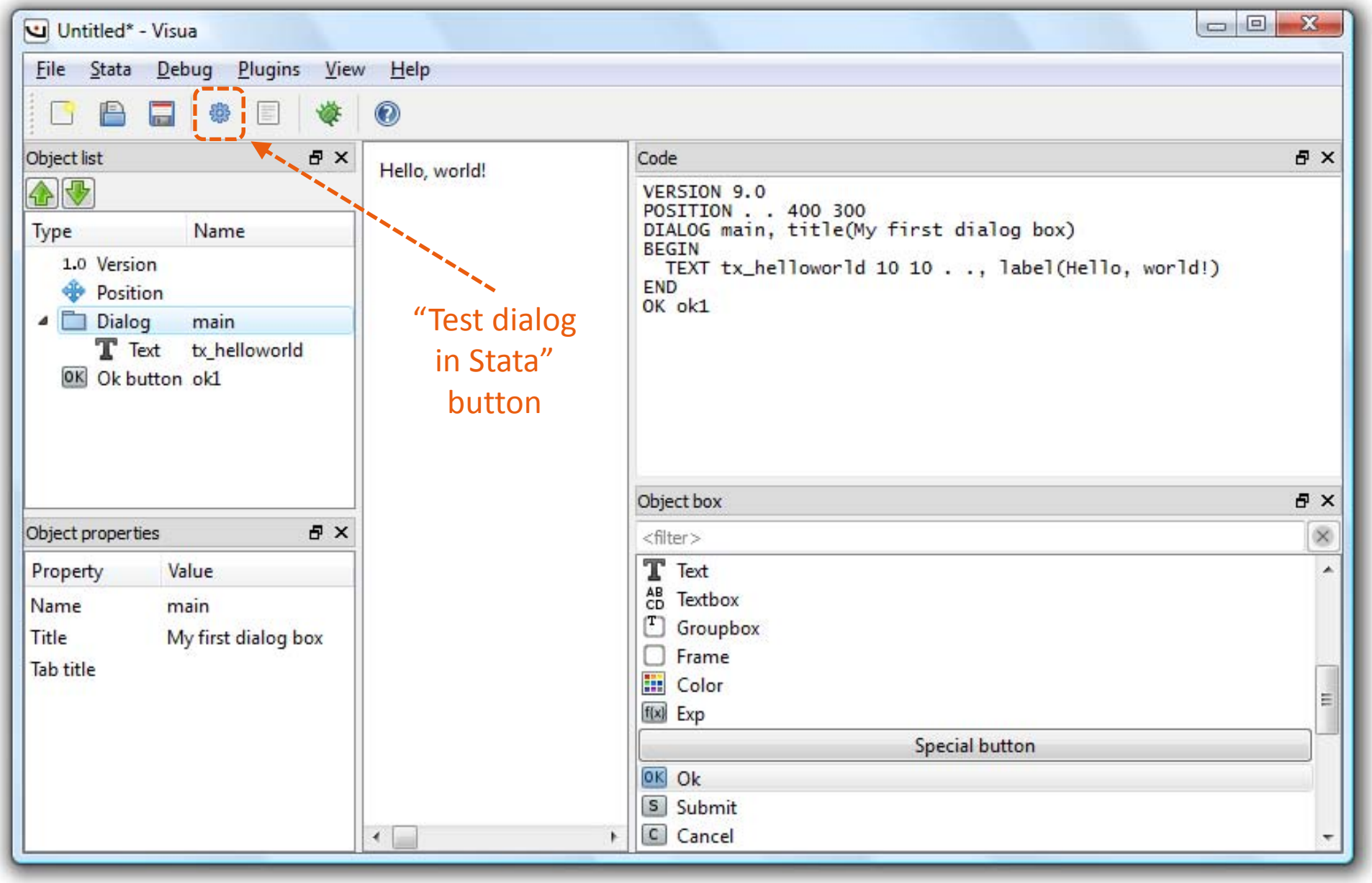
Select the “Text” object and modify its properties

The “Hello, world!” dialog box in Visua



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

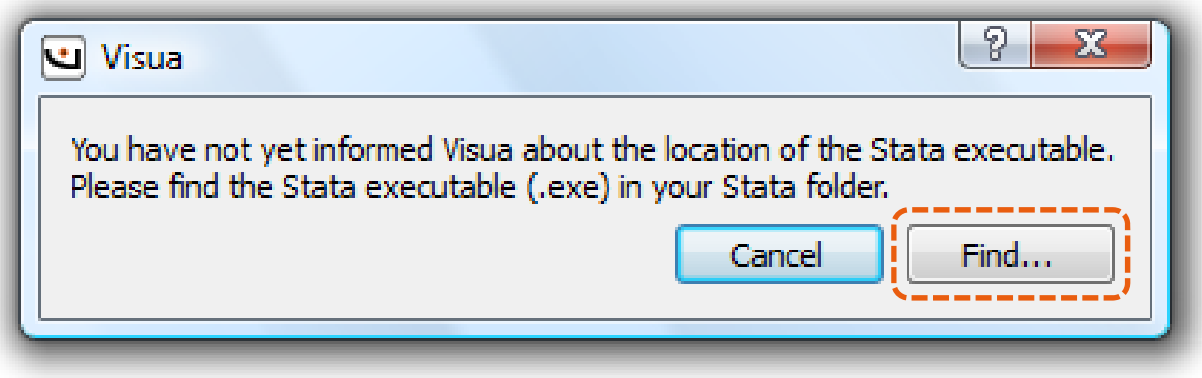
The “Hello, world!” dialog box in Visua



Click on the “Test dialog in Stata” button to test the dialog

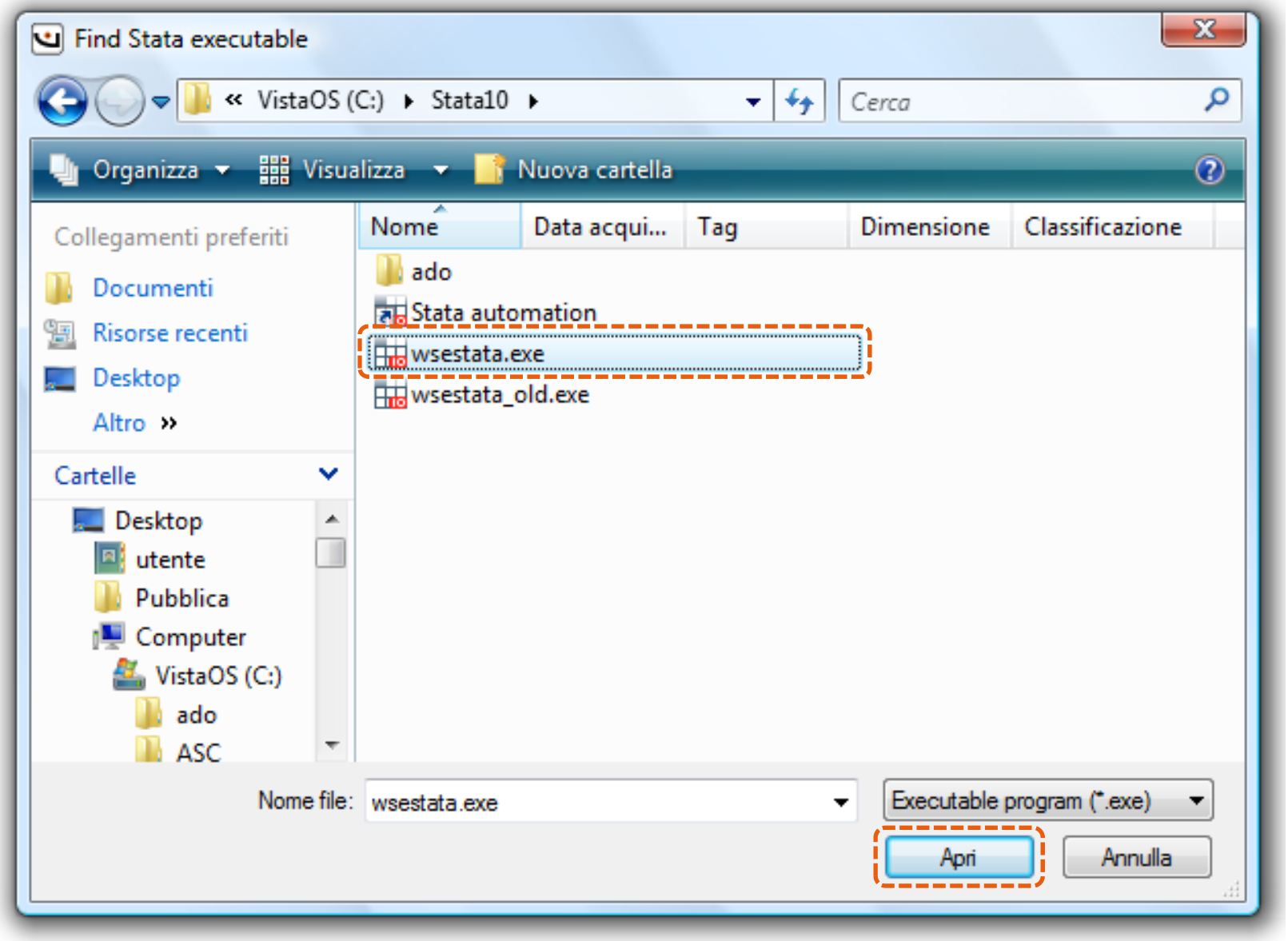
The “Hello, world!” dialog box in Visua

This message will only appear the first time you try testing the dialog in Stata



Click on the “Find…” button

The “Hello, world!” dialog box in Visua



Choose the Stata executable (“wsestata.exe” on my pc)

How the test process works

1) VISUA

1.1) Generates *visuatest.dlg*

1.2) Calls Stata to execute *visuatest.dlg*

2) STATA

2.1) Executes *visuatest.dlg*

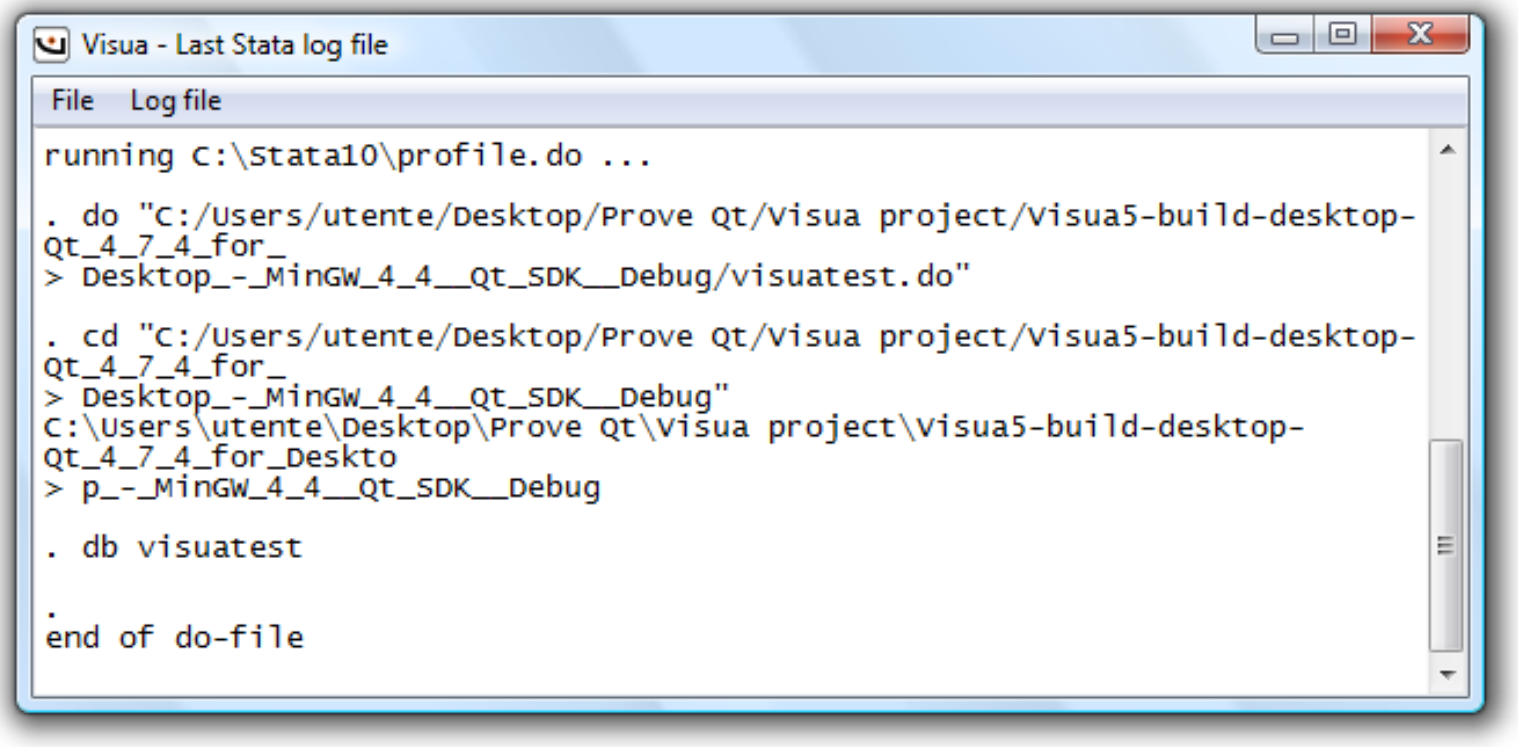
2.2) Generates the log file *visuatest.log*

Note:

visuatest.dlg and *visuatest.log* are temporary files stored in the “Visua 0.1 beta\Visuatest” folder

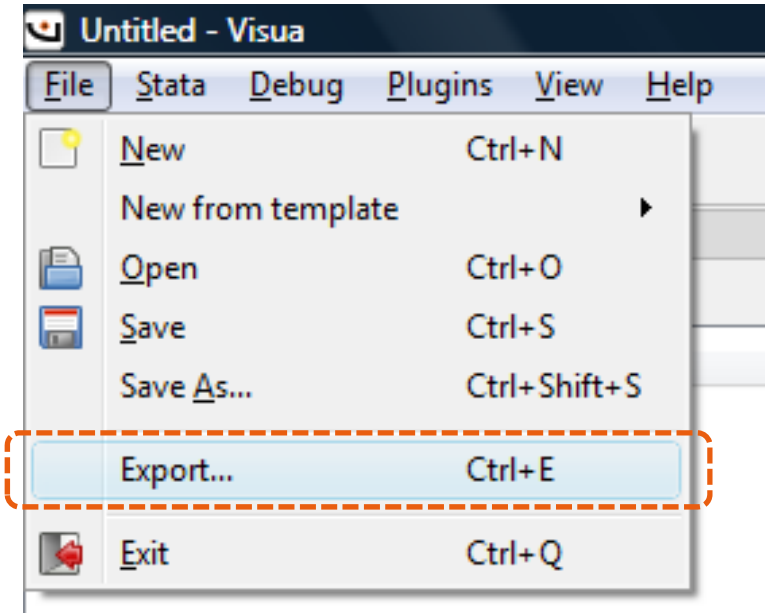
Viewing the last Stata log file

“View last Stata log file” button



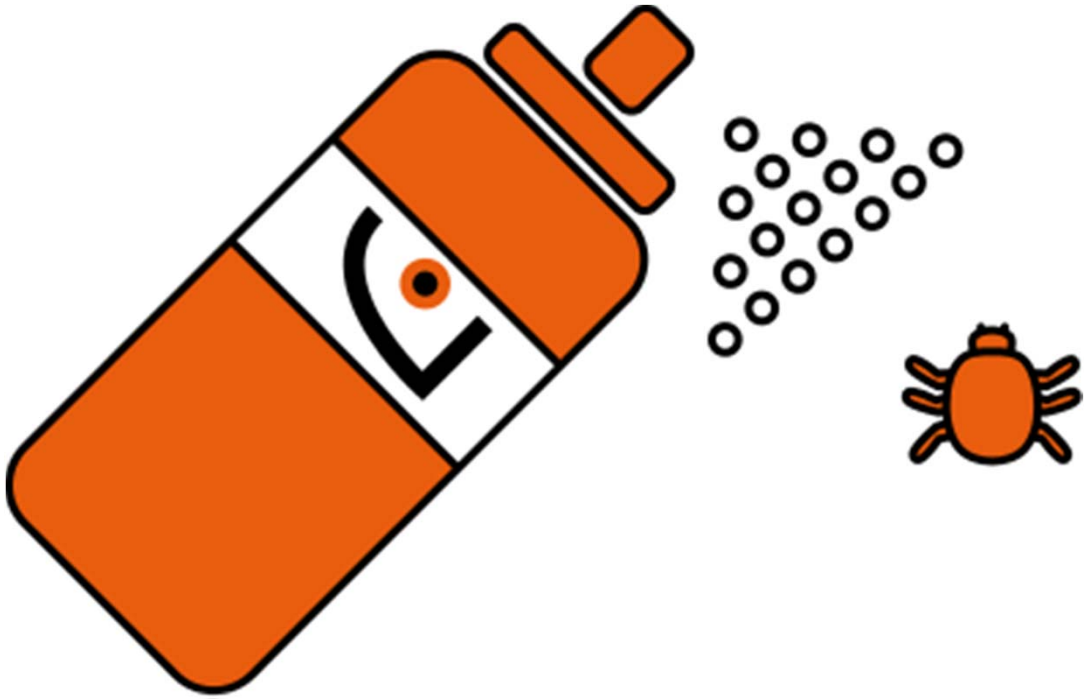
Click on the “View last Stata log file” button

Exporting to a .dlg file



“File” ⇨ “Export…”

Debugging



Debugging with Visua

“Debug report” button

Strenghts

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

Weaknesses

Currently, only some errors are detected

Where is the error?

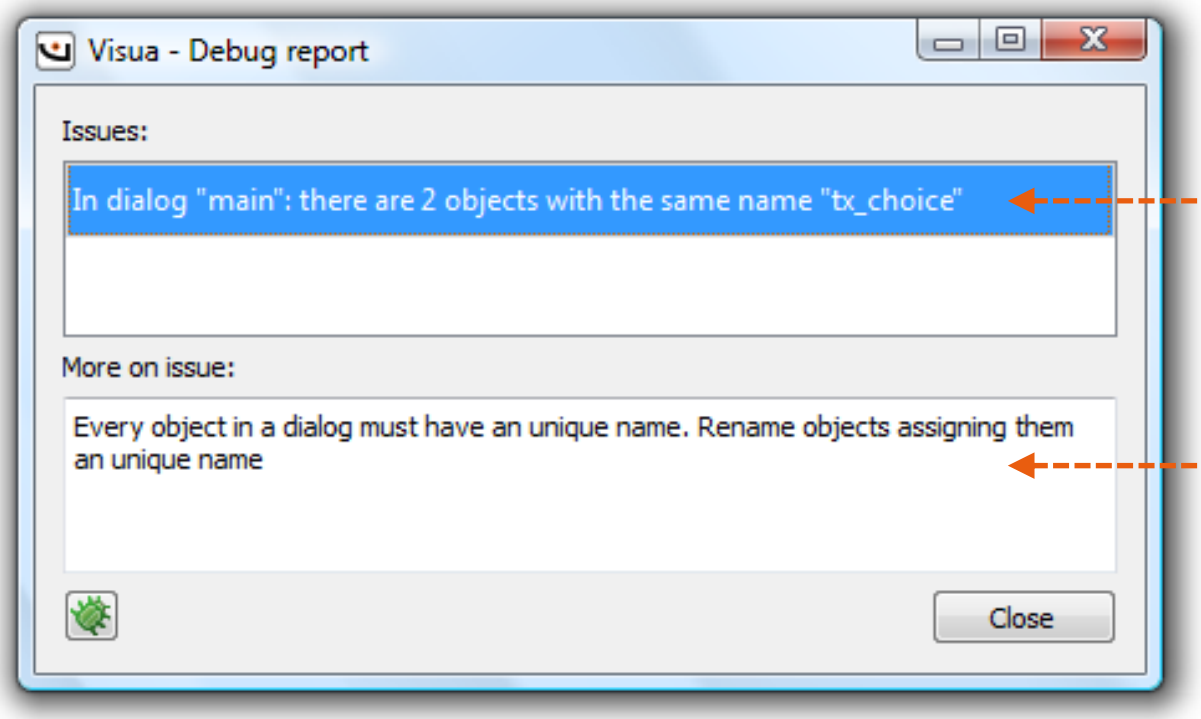
debugexample.dlg

```
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

```
. discard

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

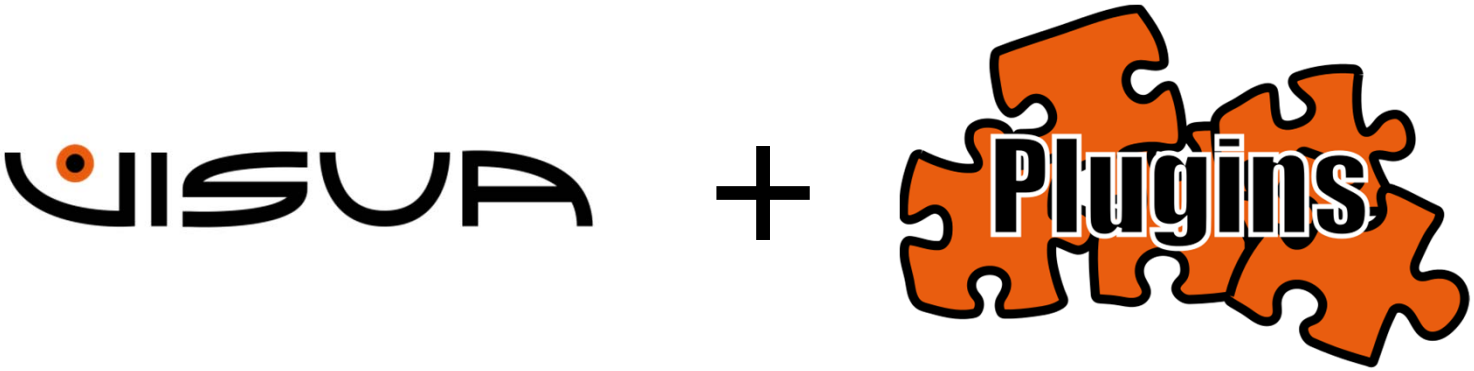
Obscure and unhelpful error message



More informative error message

Clear and user-friendly explanation about the issue

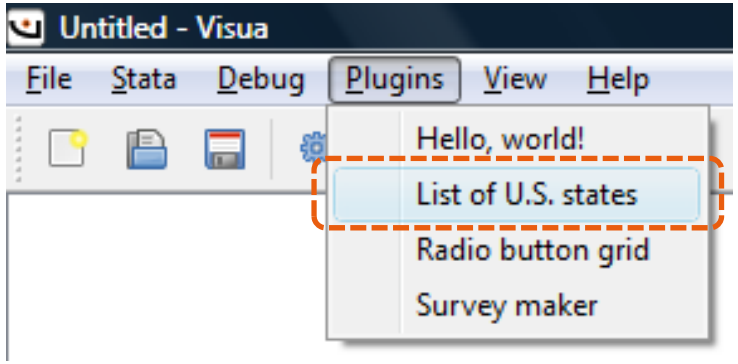
Plugins



The Visua plugin framework

- 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

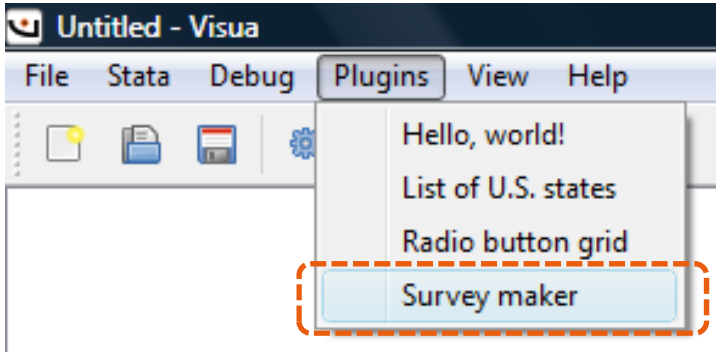


The generated code

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

53 lines of code were generated

The “Survey maker” plugin

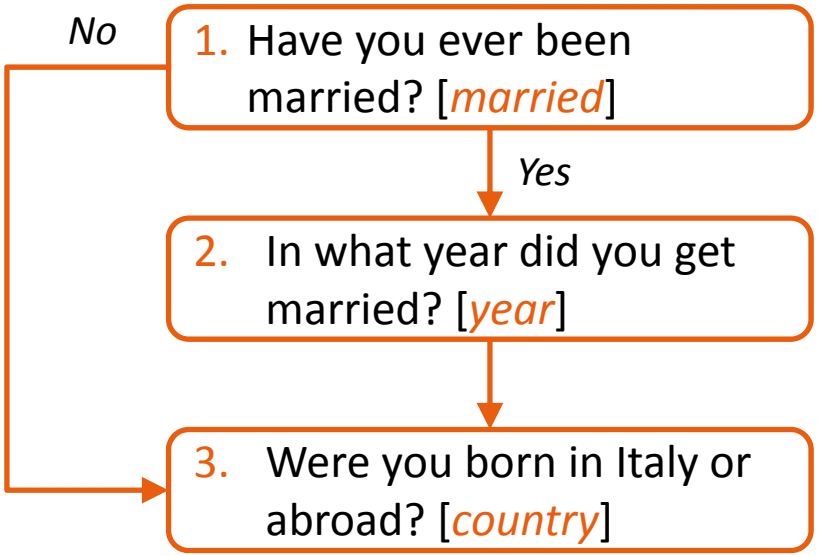


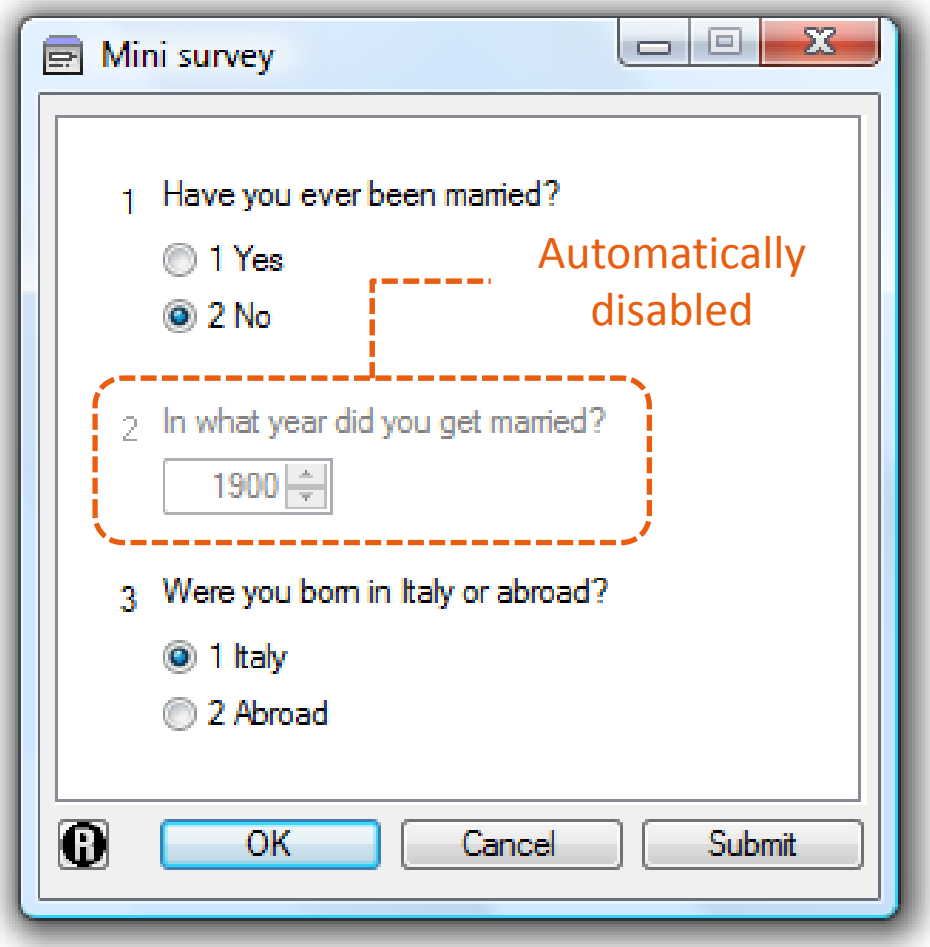
- The “Survey maker” plugin creates a dialog box which allows the user to insert complex survey data directly into Stata

- Items are disabled, depending on the response from the previous questions

- Non-consistent data entries are avoided

The “Survey maker” plugin: a filter question



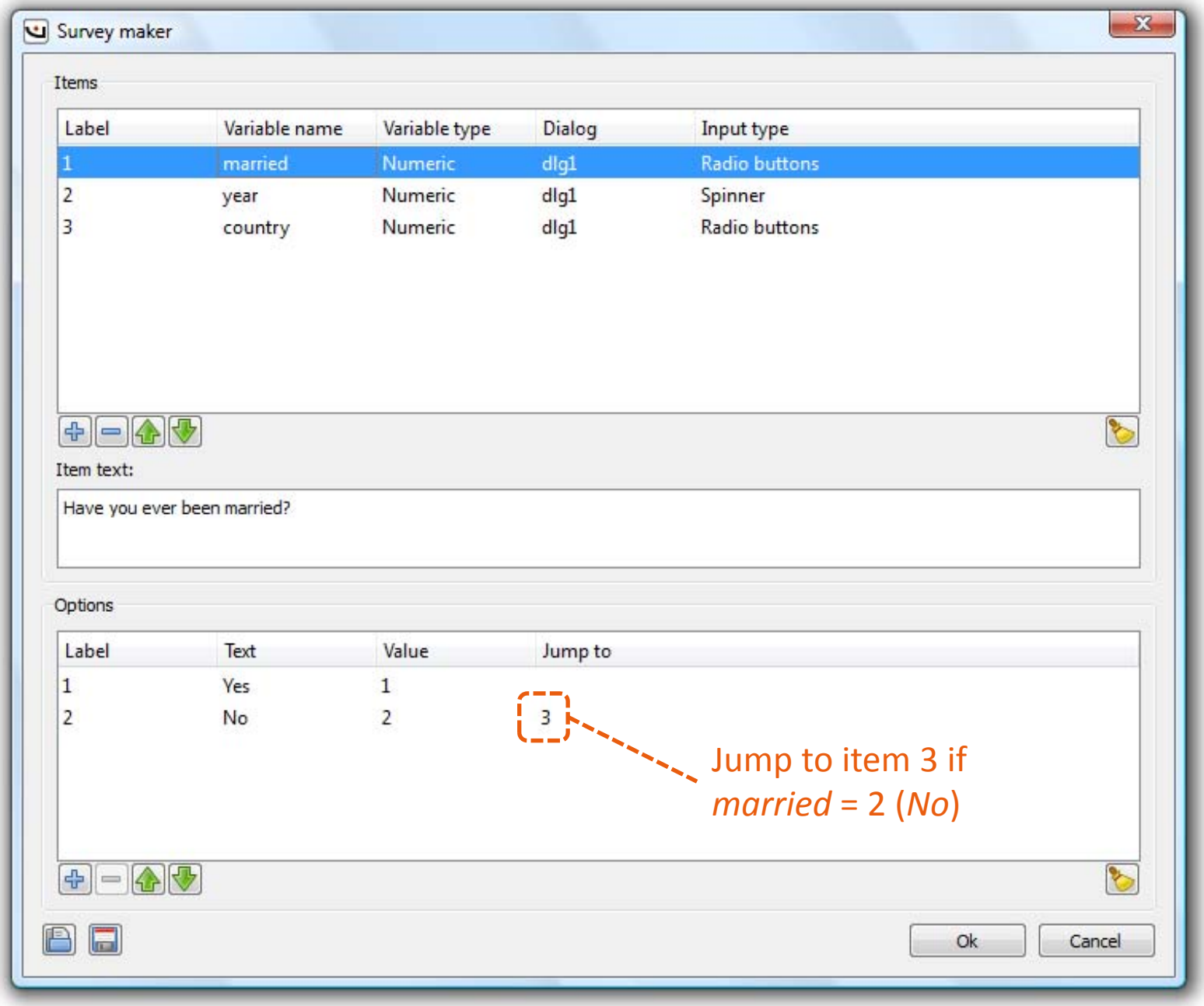


	married	year	country
1.	1	1996	1
2.	2	2005	1

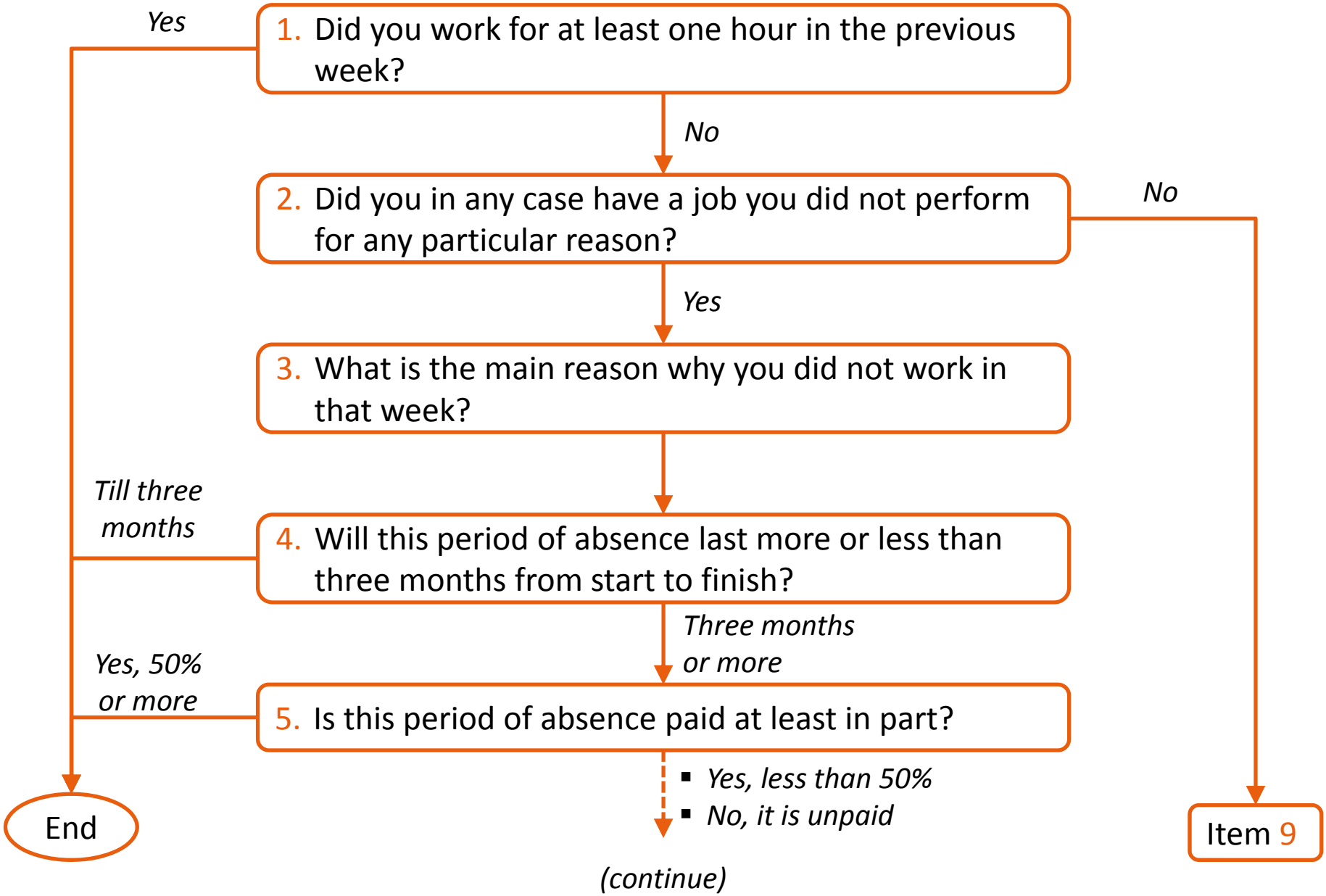
Consistent record

Non-consistent record: the dialog will avoid this

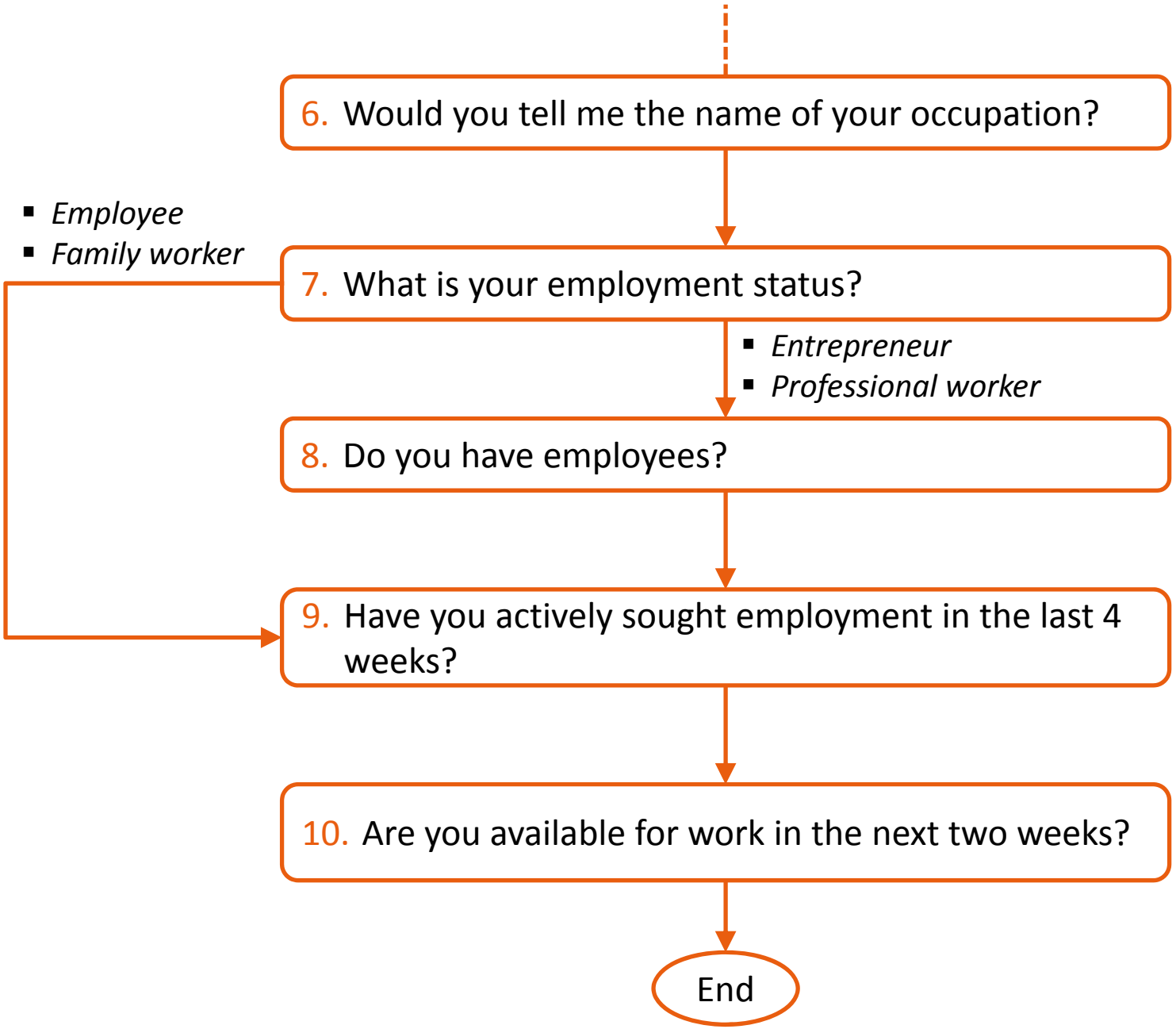
The Survey Maker plugin in action



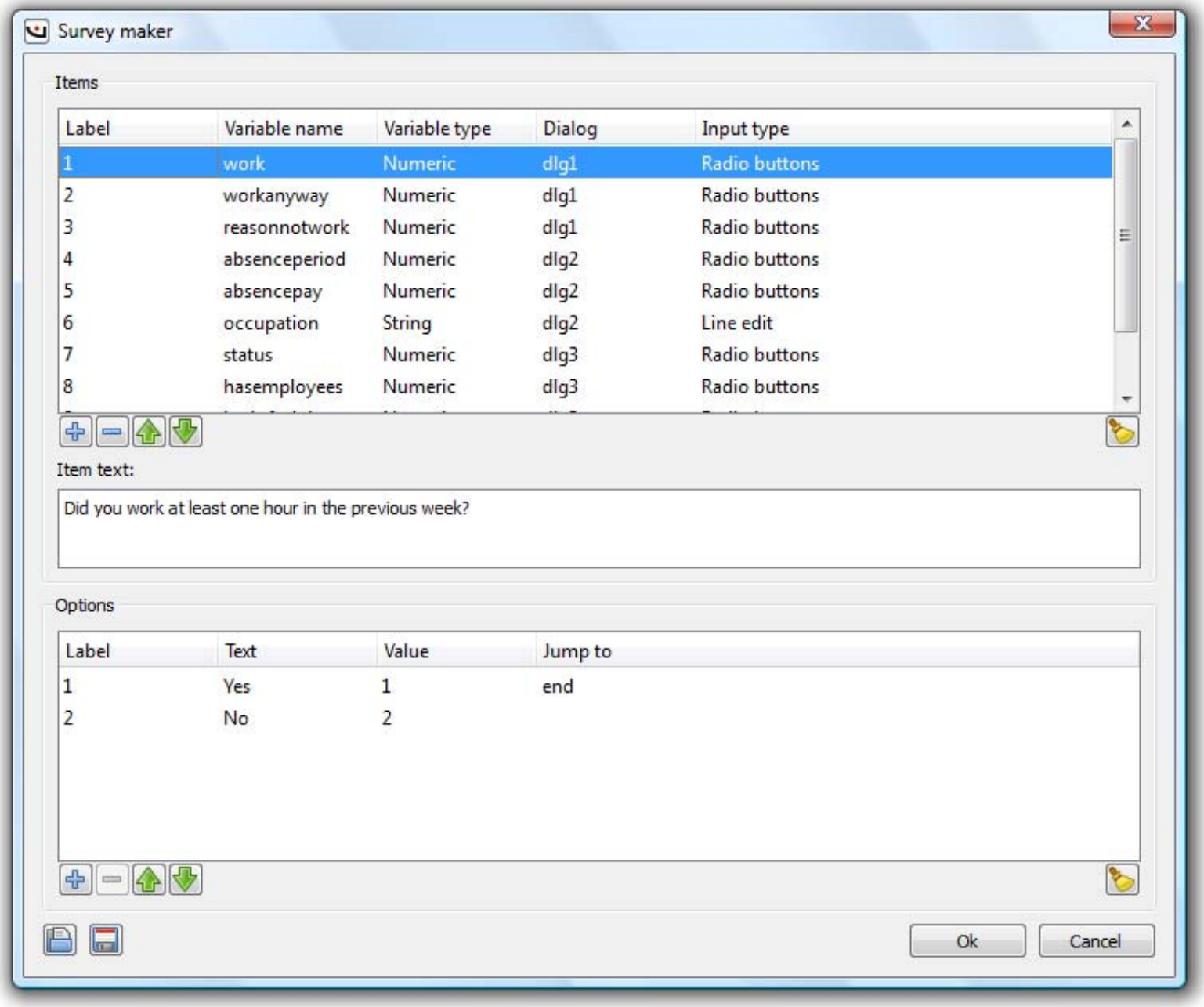
A simplified labour force survey



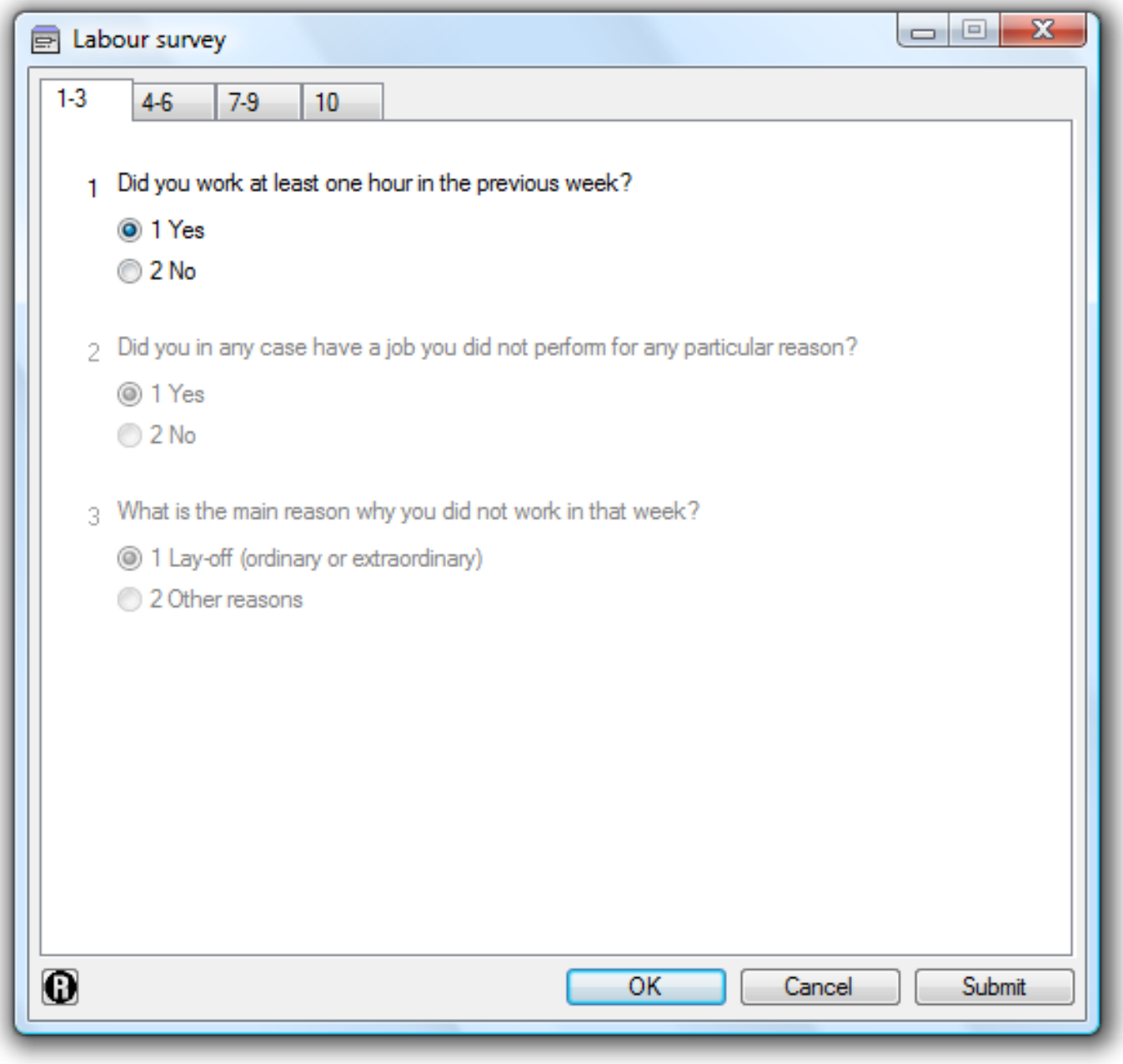
A simplified labour force survey



A simplified labour force survey



A simplified labour force survey



A simplified labour force survey

Generated code

412 lines of code were automatically generated!

Check before data insertion

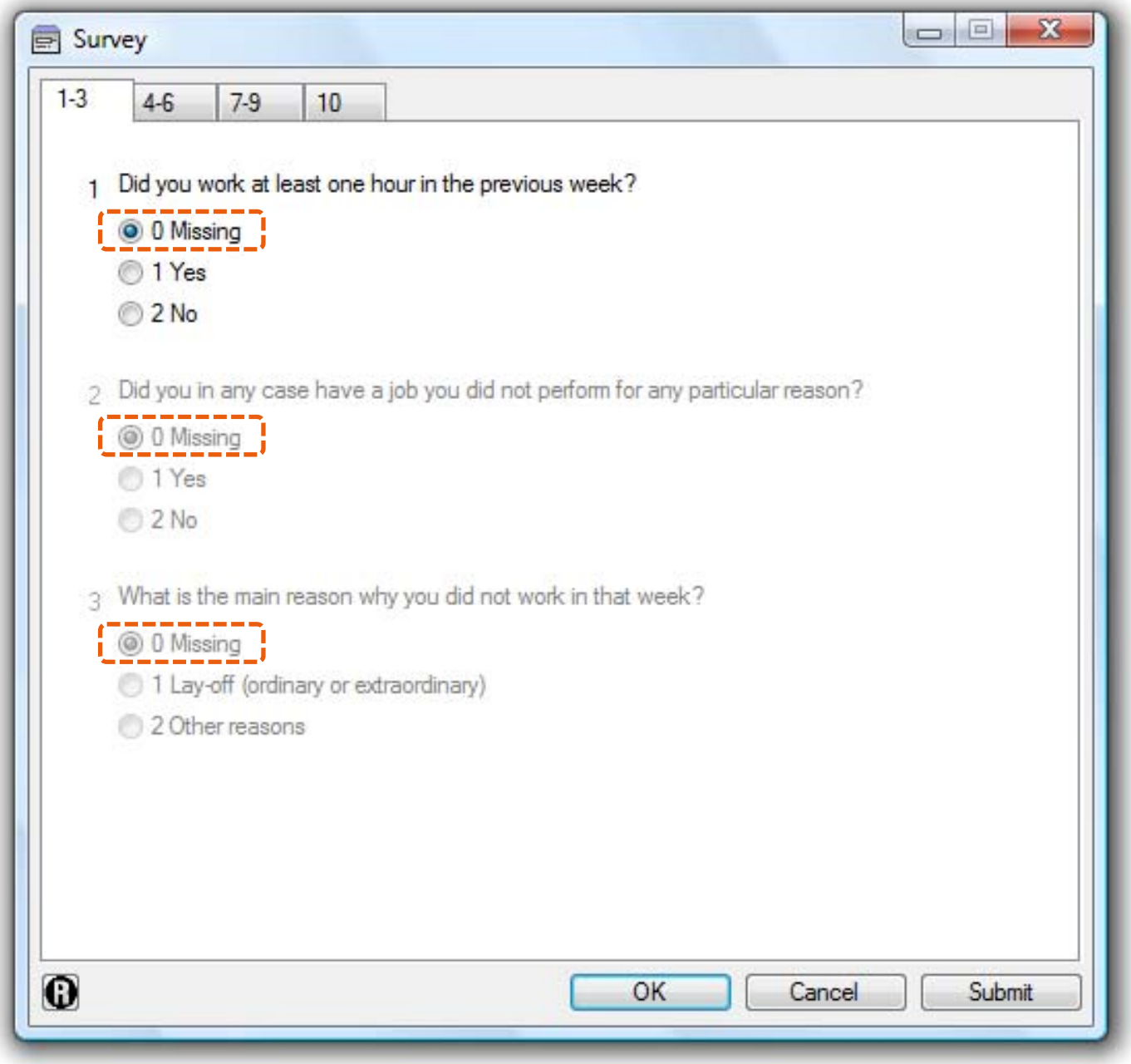
Before inserting data into the dataset, the dialog box checks if:

- 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)
- A reference to the Qt Plugin Framework:
  Blanchette J., Summerfield M. (2006), *C++ GUI Programming with Qt 4*, Prentice Hall.
- The *visuapluginterface.h* file (see next slide)

The Visua plugin interface file

visuapluginterface.h

```
#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>.</ysize>
        <label>Hello, world!</label>
    </object>
</visua>
```

How it will be "translated" by Visua

```
TEXT hello_world 20 20 100 ., label(Hello, world!)
```

Example: the “Hello, world!” plugin – header file

helloworld.h

```
#ifndef HELLOWORLD_H
#define HELLOWORLD_H

#include <QtGui>
#include "visuaplugininterface.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)*

```
#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)

Example: the “Hello, world!” plugin – implementation file

helloworld.cpp *(part 2 of 2)*

(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

The functions of the Visua plugin programming framework

`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.

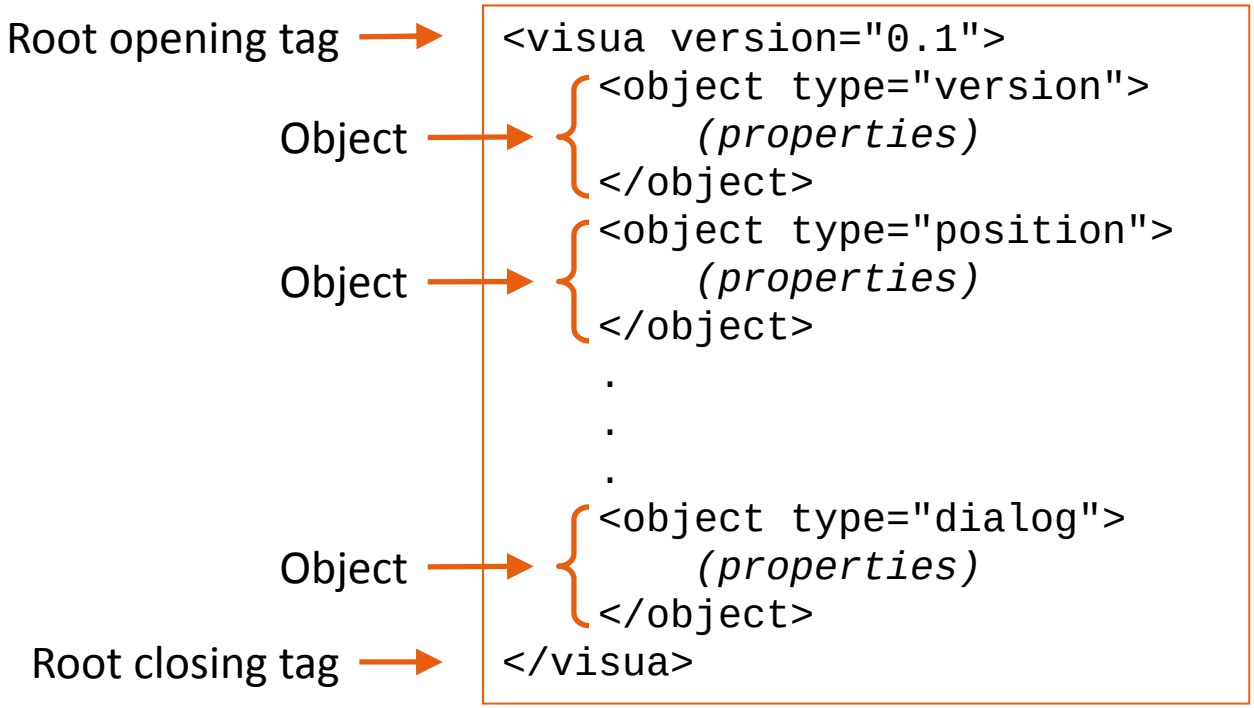
`virtual bool VisuaPluginInterface::isDialog() = 0`

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.

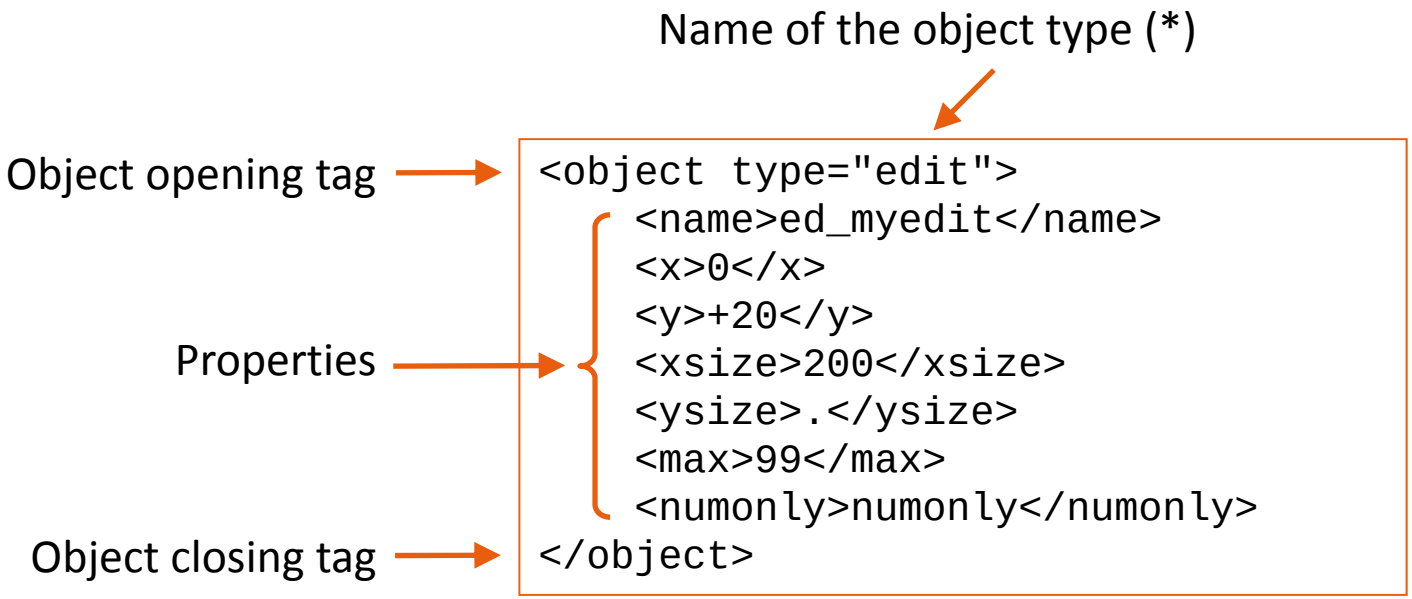
`virtual VisuaPluginInterface::Behaviour behaviour() = 0`

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

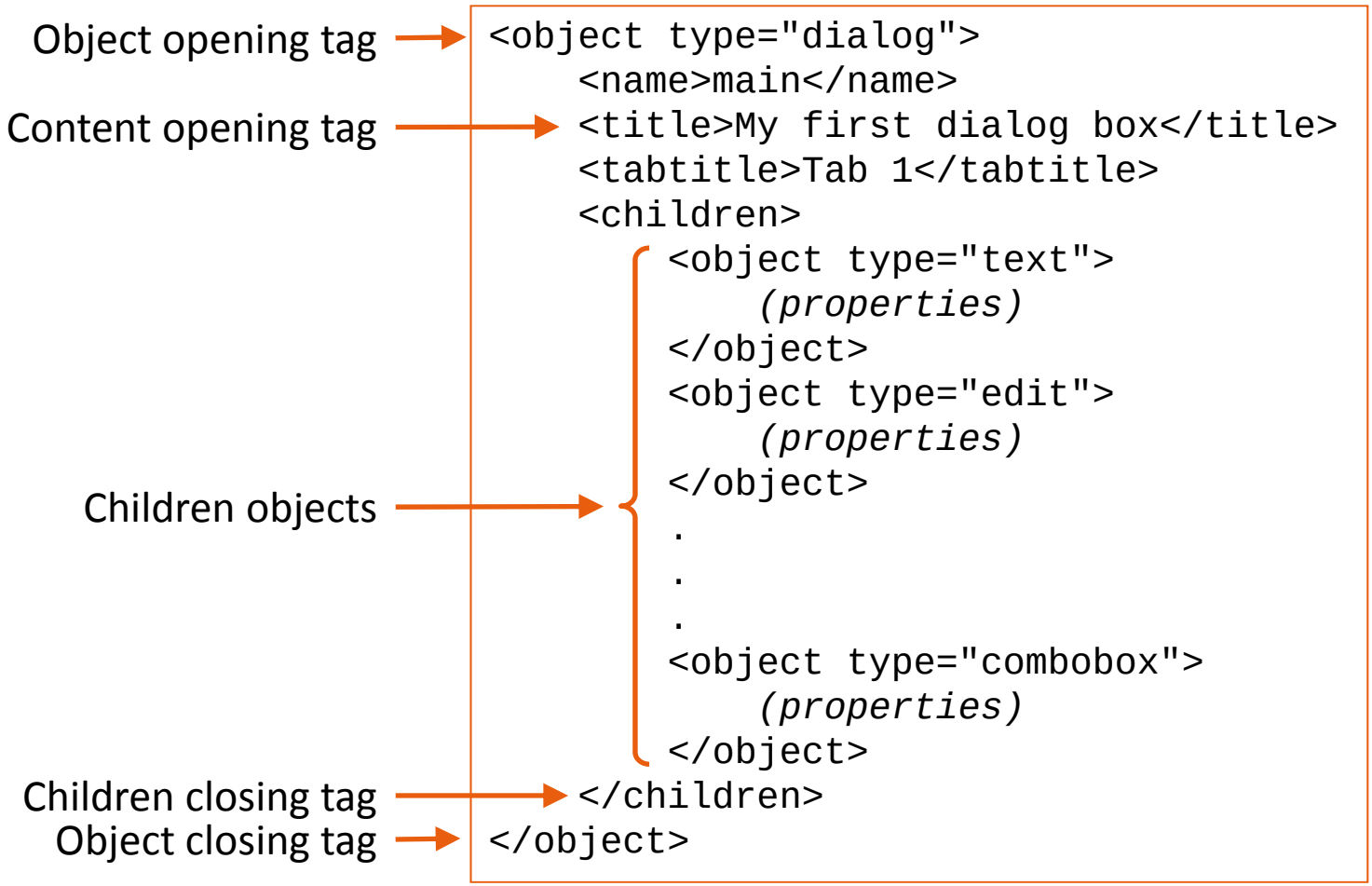


An example of an EDIT object

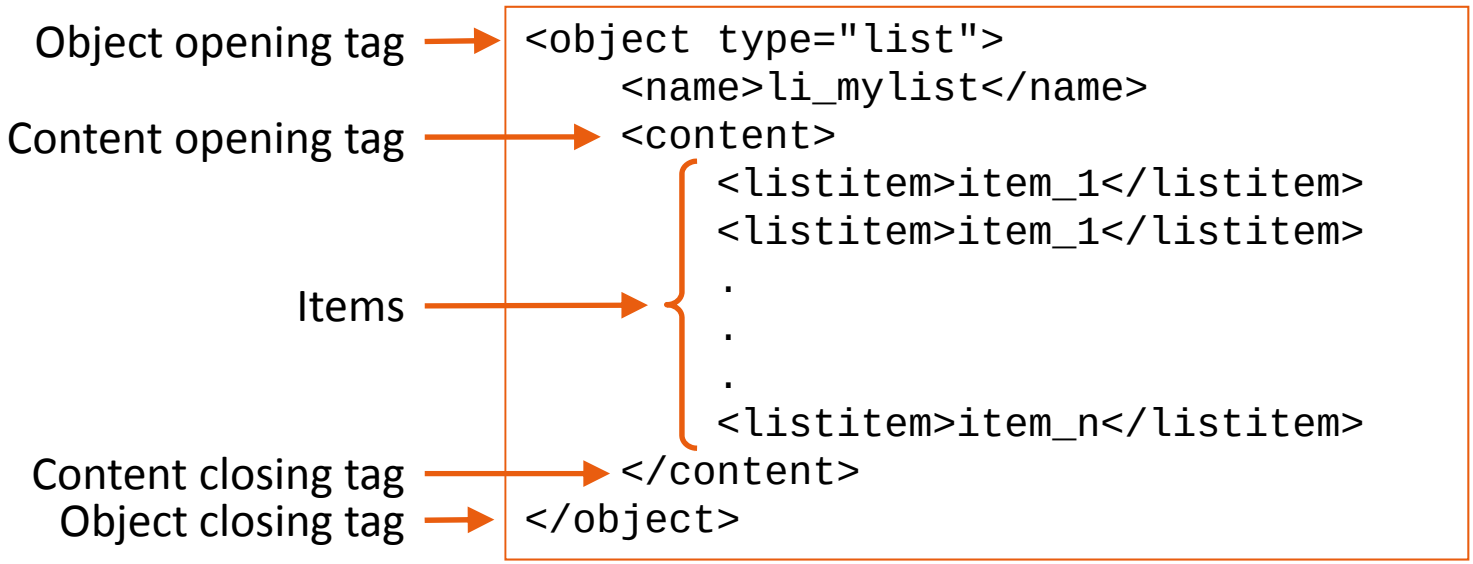


(*) It's the name of the control in all lower-case letters. For example, use "combobox" for the COMBOBOX object

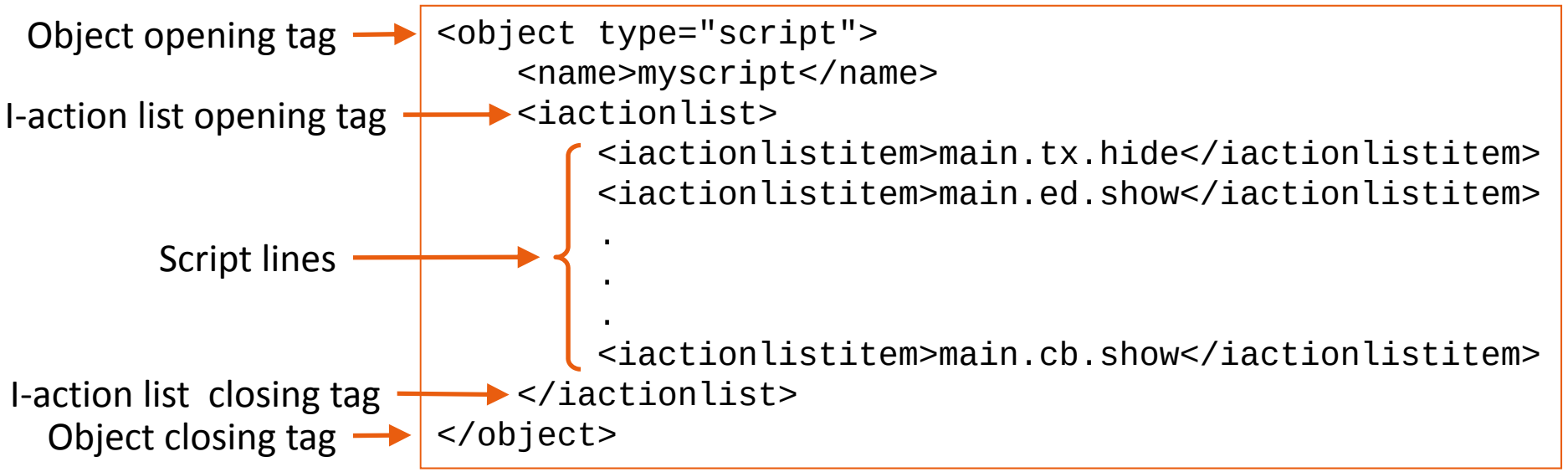
Special object: DIALOG



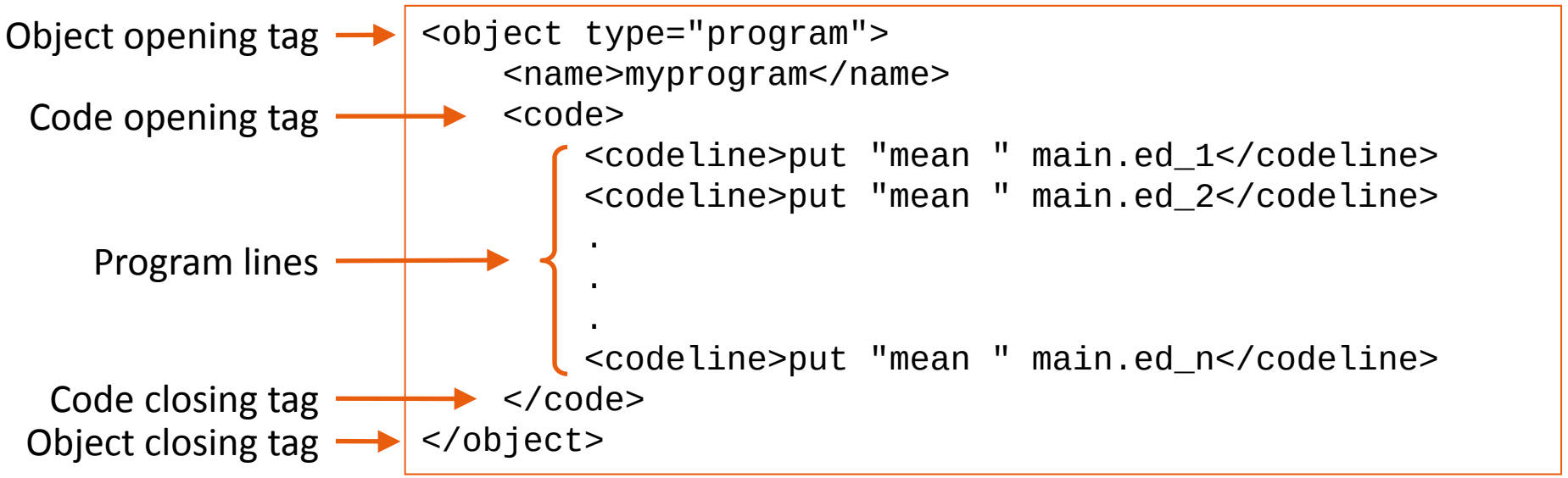
Special object: LIST



Special object: SCRIPT



Special object: PROGRAM



Example: the Visua code for the “Hello, world!” dialog (helloworld.vis)

```
<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>.</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

How to learn more on the Visua code tags?

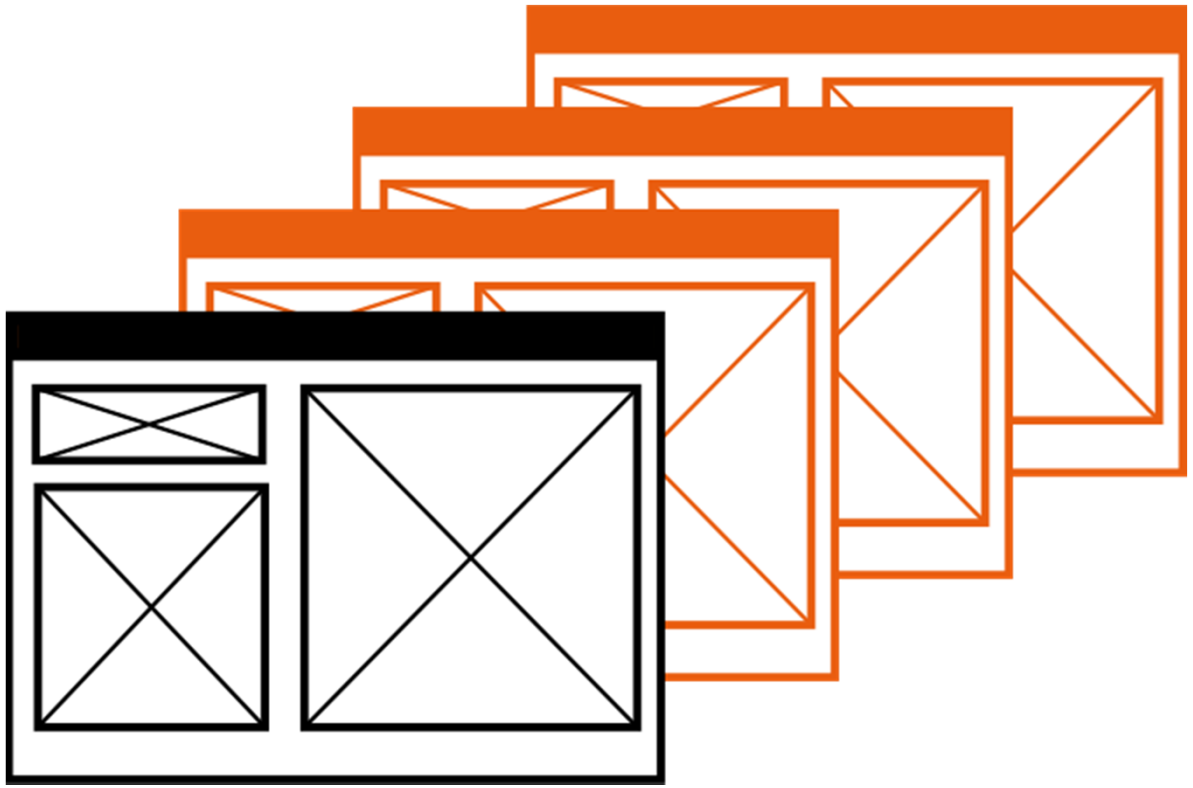
Sorry, there is no complete official documentation

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

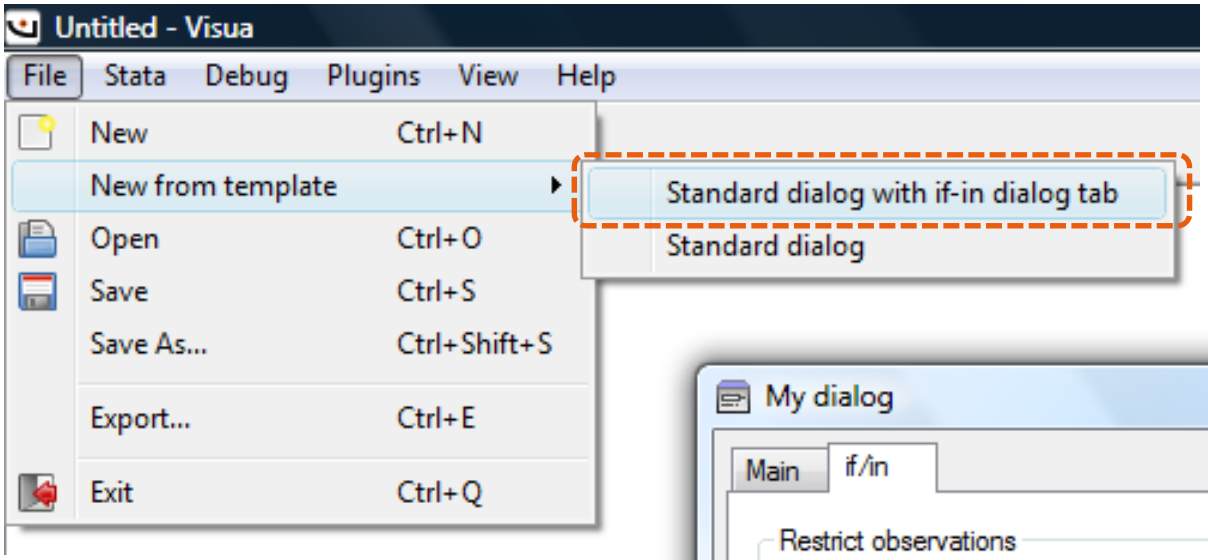
Appendix B – Visua templates

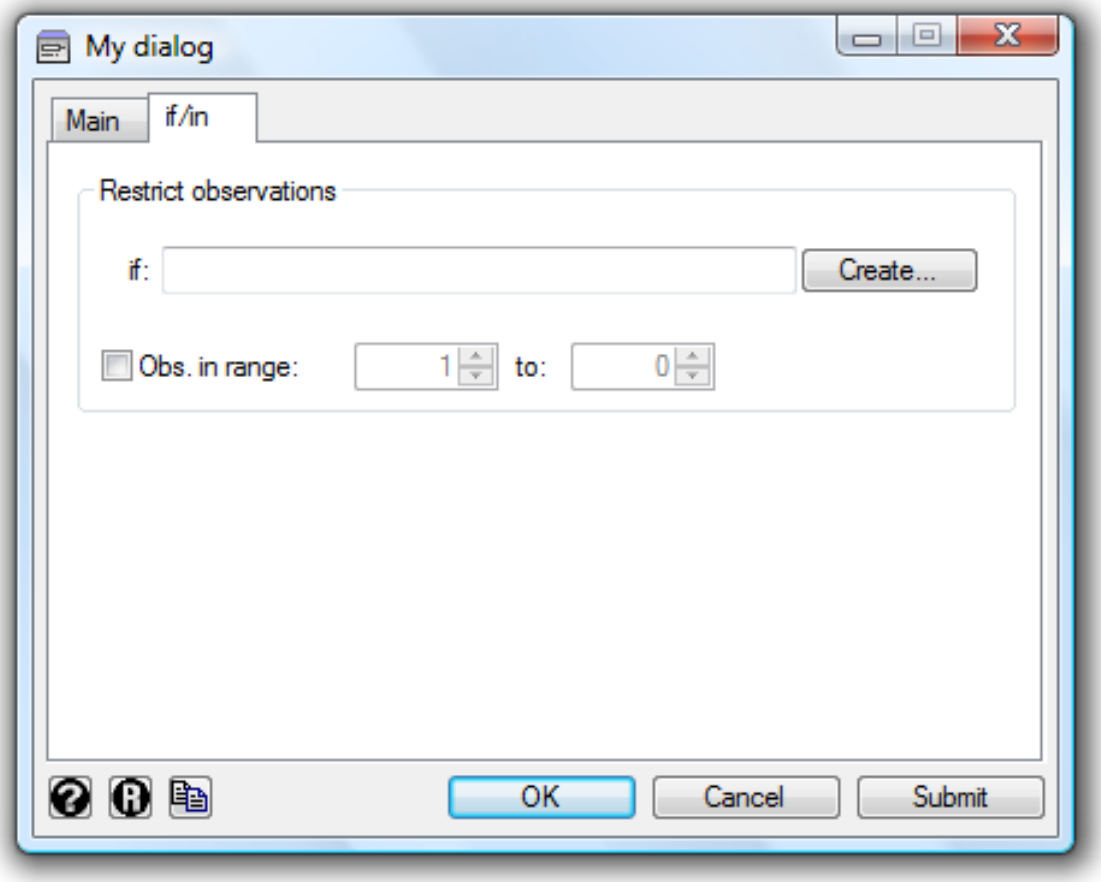


What are templates?

- 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