Making Visual Programming Accessible to the Blind

Robert M. Siegfried, Denis Diakoniarakis and Kenneth G. Franqueiro
Department of Mathematics and Computer Science
Adelphi University
Garden City, NY 11530 U. S. A.

Abstract - The proliferation of graphical user interfaces has had a dramatic impact on the ability of the blind to work as programmers. It is particularly difficult for the blind to design forms for programs written in Visual Basic. A scripting language is introduced that enables the blind to create Visual Basic forms without the need to specify all the detailed information that Visual Basic requires and without the "point and click" approach that the blind cannot use. The syntax of the language is described and future plans, including the expansion of the language and testing are discussed.

Keywords: Visual Programming, Graphical User Interfaces, Visual Basic, Blind Programmers

1.0 Introduction

Computer programming has provided an accessible means of employment for the blind and visually impaired. This has been unusual, given that the blind have a very high unemployment rate [1]. This has been facilitated by a collection of specialized tools including screen readers such as JAWS, speech recognition programs such as Dragon, and Braille terminals and screen enlargers for the visually impaired. These tools and the relevant training have allowed the blind to compete in the text-based world of computers.

Since Microsoft Windows version 3 debuted in 1990, graphical user interfaces have become the standard for modern computing. The change has also had a dramatic impact on the job market for blind programmers. Most Windows-based applications are developed using software tools that the blind cannot use. Additionally, it has been difficult to update screen readers frequently enough to remain current with new developments in GUIs. As a result, most blind programmers continue to concentrate on text-based applications. Less than 10% of blind programmers work in Windows application development[2].

While the original goal of the project was to develop a text-based, a platform-independent programming language that was suitable for use by blind programmers for rapid application development (RAD), feedback from members of the *Blind Programming* mailserv list indicated that they preferred tools that would help them work in languages such as Visual Basic. A grammar for the scripting language was originally proposed in 2002[3] and a prototype compiler was developed in 2003[4]. In 2004, the language was expanded to include all the classes of objects that can appear within a Visual Basic form.

```
Form MyForm
Location = Top Right
Caption = "Sample 4"
Organization = Columns
Section
  Textbox txtMyBox
    Height = 2
    Width = Medium
    Label = "Sample TextBox"
  End
  ScrollBar scrMine
    Orientation = Horizontal
      Length = Small
      Min = 0
      Max = 100
      Value = 50
      SmallChange = 1
      LargeChange = 5
  End
  Frame fraYours
    Caption = "Your Frame"
    Optionbutton optDenis
      Caption = "Denis"
      Visible = True
    End ' option button
    Optionbutton optKenny
      Caption = "Kenny"
      Visible = True
    End ' option button
  End 'frame
End ' Section

Section
  DirListBox dirName
    Height = 3
    Width = Large
  End
  DriveListBox drvName
  End
  FileListBox filName
    Height = 1
    Width = Medium
  End
  CommandButton cmdTryThis
    Caption = "Press here"
  End' command button
End ' Section
End ' Form
```

Figure 1. Script for Form in Figure 2.

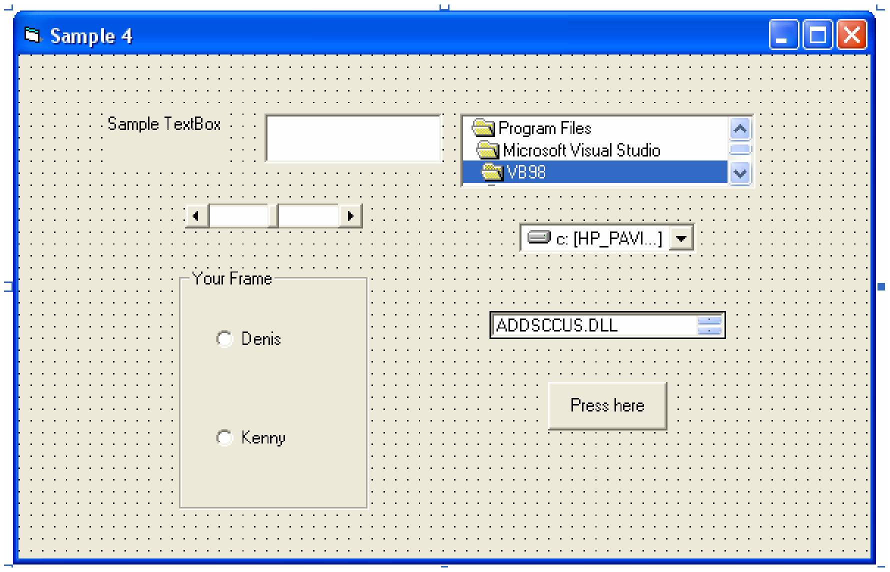

Figure 2: The form created from the above script

Visual Basic became a popular application because it allows users to create forms by "pointing and clicking." Visual Basic forms are stored in text format with the form's properties and its member objects' properties listed together with their values. While this allows one to change these values fairly easily, it is difficult to design a form by creating such a text file. For example, the size of a form is set by specifying the form's height and width in *twips* (twentieths of a point). Similarly, setting the position of a form on the screen is done by specifying the position of the left and top edges in twips.

This presents the blind with the task of providing a large amount of detailed information about the form's layout when creating Visual Basic forms. An example of a simple Visual Basic form appears in Figure 1. While it is easy to modify form properties using this file, it is difficult even for a sighted person to create a file like this without using trial and error to determine if the form is correctly constructed. This challenge is even greater for the blind because they cannot see the form. The main goal in a scripting language is to simplify the process of designing a form while allowing users to specify property values that differ from the default.

2.0 The Scripting Language and Compiler

2.1 The Compiler

The compiler is a console application, run separately from Visual Basic and then included in the Visual Basic project. The executable file for the compiler is `molly.exe`. The files containing the form scripts use the extension *.fms* (for form script). After creating the form script using any text editor (such as Notepad), the form script can be compiled a command prompt window using the command

```
molly FileName.fms
```

where *FileName*`.fms` is the name of the form script file. If the file's name is *test.fms* the command would be:

```
molly test.fms
```

This will produce a standard form file *test.frm*, which can then be included in a Visual Basic project. The script for the form shown is Figure 1 is:

The language uses the Basic style of comments where a comment begins after the apostrophe and continues until the end of the line.

2.2 The Scripting Language Syntax

The basic layout of a form script is:

```
Form FormName ↵
Location = VerticalAttrib HorizontalAttrib↵
Caption = "Caption on Title Bar of Form" ↵
Organization = { Rows or Columns }↵
SECTION
        SectionAttributes
        END ' Comments appear after an apostrophe until the end of the line
…
END
```

Where ↵ indicates a carriage return.

The screen is divided into three rows and three columns: the three rows are top, middle and bottom and the three columns are left, center and right. This allows the user to place the form in different areas of the screen without having to measures twips or use trial and error.

Most forms are organized in rows or columns, but usually not both within the same form. For this reason, the user specifies whether the form's organization is in rows or columns. After the organization is specified, each section (either or a row or column, depending on which one the programmer has chosen), is declared with one or more object in the section, which are automatically laid out sequentially within the section. Their exact placement depends on the particular object and its own space requirements. The programmer can specify as few or as many objects as desired in any given section, as long as they all fit within a window. Similarly, the only limit on the number of sections is their ability to fit within the window.

The initial prototype allowed the programmer to use any combination of five different object types: command buttons, text boxes, combo boxes, frames, and check boxes. The language now includes most standard object types that appear in Visual Basic.

2.3 Specifying Objects

Each of the eleven objects that can be specified has its own syntax because the key properties differ from one object to another. The general syntax for an object is:

```
ObjectType      ObjectName
 Properties
End
```

where *ObjectType* is `CommandButton`, `TextBox`, `ComboBox`, `Frame`, `CheckBox`, `ListBox`, `Timer`, `DriveListBox`, `FileListBox`, `DirListBox`, `ScrollBar`. *ObjectName* is the name that the object will have within the Visual Basic form. It must be unique within the form and should follow the standard name conventions for Visual Basic objects.

The syntax for each type of object declaration appears in Table 1 along with an example. Some of the declarations are very simple with only one or two properties specified. Command buttons are examples of this; other than name, the property that is specified is the caption. Other objects have a larger number of properties specified. Scroll bars, for example, require that the programmer specify whether the scroll bar is horizontal or vertical, its length (as small, medium or large), as well as its minimum, maximum, small change and large change values. It is noteworthy that in all instances, a programmer will specify fewer properties that is he or she were creating a Visual Basic form file directly.

Table 1 - the objects that can be appear in a form script

Object Type	Syntax	Example

CommandButton	`CommandButton` *cmdButtonName*↵
	`Caption =` "*CommandButtonCaption*" ↵
	`End`↵
		`CommandButton cmdConvert`
		`Caption = "Convert to Metric"`
		`End`

TextBox	`TextBox` *txtTextboxName*↵
	`Height =` *NumberOfLines* `(1..5)`↵
	`Width = {Small` *or* `Medium` *or* `Large}`↵
	`Label = "`*`Text Box Label`*`"` ↵
		`TextBox txtName`
		`Height = 1`
		`Width = Large '2415 twips 'wide`
		`Label = "Enter Last Name"`
		`End`

	End↵	

ComboBox	ComboBox *cboComboBoxName*↵
	Sorted = *{* True *or* False *}*↵
	Style = *Number* ↵
	Width = *{* Small *or* Medium *or*
	          Large *}*↵
	End↵
		ComboBox cboSample
		Sorted = true
		Style = 1
		Width = Small '1215 twips
		              ' wide
		END

Frame	Frame *fraFrameName*↵
	Caption = *"Frame Caption"* ↵
	OptionButtonDeclarations↵
	End↵
		Frame fraQuantity
		  Caption = "# of Tickets"
		  OptionButton optNone
		    Caption = ""
		    Visibility = False
		  End
		  OptionButton optOne
		    Caption = "1"
		    Visibility = True
		  End
		  OptionButton optTwo
		    Caption = "2"
		    Visibility = True
		  End
		End

OptionButton	OptionButton *optOptionButtonName*↵
	Caption = *"Opt. Button Cap"* ↵
	Visibility = *{* True *or* False *}*↵
	End↵	

CheckBox	CheckBox *chkCheckBoxName*↵
	Caption = *"CheckBox Caption"* ↵
	Height = *NumberOfLines(1..5)*↵
	Width = *{* Small *or* Medium *or*
	 Large *}*↵
	End↵
		CheckBox chkSample
		Caption = "CheckBox Sample"
		Height = 3
		Width = Medium '1815 twips
		                'wide
		End

ListBox	ListBox *lstListBoxName*↵
	Sorted = *{* True *or* False *}*↵
	Style = *1 or 2 …* ↵
	Height = *1 or 2 …* ↵
	Width = *{*Small *or* Medium *or* Large
	}↵
	End↵
		ListBox lstNames
		Sorted = True
		Style = 0 'Standard not
		          'checkbox
		Height = 3
		Width = Medium
		End

Timer	Timer *tmrName*↵ Interval = *NumberOfSeconds*↵ End↵	Timer tmr1 Interval = 2 End
DriveListBox	DriveListBox *drvDrivelistboxName*↵ End↵	DriveListBox drvName End

FileListBox	FileListBox *filFileListBoxName*↵
	Height = *NumberOfLines (1..5)*↵
	Width = *{* Small *or* Medium *or*
	         Large*}*↵
	End↵
		FileListBox flbName
		Height = 4
		Width = Medium
		End

DirListBox	DirListBox *dirListBoxName*↵
	Height = *NumberOfLines (1..5)*↵
	Width = *{* Small *or* Medium *or*
	          Large*}*↵
	End↵
		DirListBox dirName
		Height = 3
		Width = Large
		End

ScrollBar	ScrollBar *scrScrollBarName* ↵
	Orientation =*{* Horizontal *or*
	               Vertical *}*↵
	Length = *{* Small *or* Medium *or*
	           Large*}*↵
		ScrollBar hsbMyScrollBarName
		Orientation =Horizontal
		Length = Large
		Min = 0
		Max = 60
		Smallchange = 1

	`Min =` *ConstantValue*↵ `Max =` *ConstantValue*↵ `Smallchange =` *ConstantValue*↵ `Largechange =` *ConstantValue*↵ `End` ↵	`Largechange =` 5 `End`

Several object types, including textboxes, comboboxes, checkboxes, and listboxes require that the width be specified. In all these cases, it is sufficient to specified them as small, medium or large. In each case, the width will be fixed as 1215, 1815 or 2415 twips respectively. The height, when required, is specified by the number of lines, not the number of twips; in all cases it is limited to 5 or fewer lines. The style is specified by the number used by Visual Basic to indicate one of several different styles; these codes are standard and do not require a programmer to use trial and error to determine the appearance of the object.

Although frames in Visual Basic forms can contain objects other than option buttons, the scripting language allows frames to contain only option buttons. This is the most common usage of frames; this was a design decision made to simplify the scripting language. Similarly, the scripting language treats horizontal and vertical scroll bars as different instances of the same object class differing only by the orientation property. This differs from the approach taken in Visual Basic where they are treated as different object classes.

3.0 Discussion

After posting the prototype compiler and its manual on Adelphi University's web site (http://www.adelphi.edu/~siegfrir/molly) and notifying members of the Blind Programming mail list of its availability, comments were received about the scripting languages; they indicate that the blind programmers who have read the specifications and the sample scripts believe that it has the potential to help them create Visual Basic forms without the aid of a sighted person. At the same time, there were questions about what it might contain in the future. Since this was only an initial prototype, there was never any doubt that additional features needed to be added to the language. The remaining object classes that were considered the most important in Visual Basic were added to the scripting language.

3.1 Form Files in Visual Basic .NET

The current version of Microsoft Visual Studio, Visual Studio .NET uses a standard text file format to represent forms in all of the language products that comprise Visual Studio .NET. While this facilitates the use of programs that are developed using more than one programming language, it makes the compiler out of date for those working with Visual Basic .NET. Fortunately, Visual Studio .NET provides a tool for converting Visual Basic forms to the new format; however this complicates the job of creating a single scripting language compiler that is capable of creating forms that are compatible with all formats. Since only a small portion of the compiler is involved in generating the form file, we expect to have a version of the compiler that generates .NET-compatible forms within a few months.

3.2 Further testing

Everyone who has worked on the project is sighted. As sensitive as the development team could be to the needs of the blind and visually impaired, it is impossible to anticipate entirely what the blind would consider more suitable for their needs. For this reason, we plan to begin testing within the next few months.

3.3 Conclusions

The prototype compiler simplifies the task of creating Visual Basic forms for blind programmers, for whom the “point and click” design method is not possible. Although testing has not been completed, preliminary comments suggest that additional features will make it a useful tool for the blind. These features allowing the scripts to specify object not currently included in the language and default values for certain properties. Additionally, a version of the compiler that can create form files compatible with Visual Studio .NET is needed as well as the current version, which creates form files compatible with version 6.

The compiler and manual are currently available online at http://www.adelphi.edu/~siegfrir/molly.

4.0 References

[1] Kirchner, C., & E. Schneidler, 1997, “Prevalence and Employment of People in the United States Who Are Blind or Visually Impaired”, Journal of Visual Impairment and Blindness 91(5), Sept/Oct, p.508-511.
[2] Siegfried, Robert M., Denis Diakoniarakis and Uchechukwu Obianyo-Agu, “Teaching the Blind To Program Visually”, Proceedings of ISECON 2004, v21 (Newport): §3265.
[3] Robert M. Siegfried, “A Scripting Language To Help The Blind To Program Visually”, ACM SIGPLAN Notices 32(2), February 2002, pp. 53-56.
[4] Siegfried, Diakoniarakis and Obianyo-Agu, op.cit.