The Turtle System

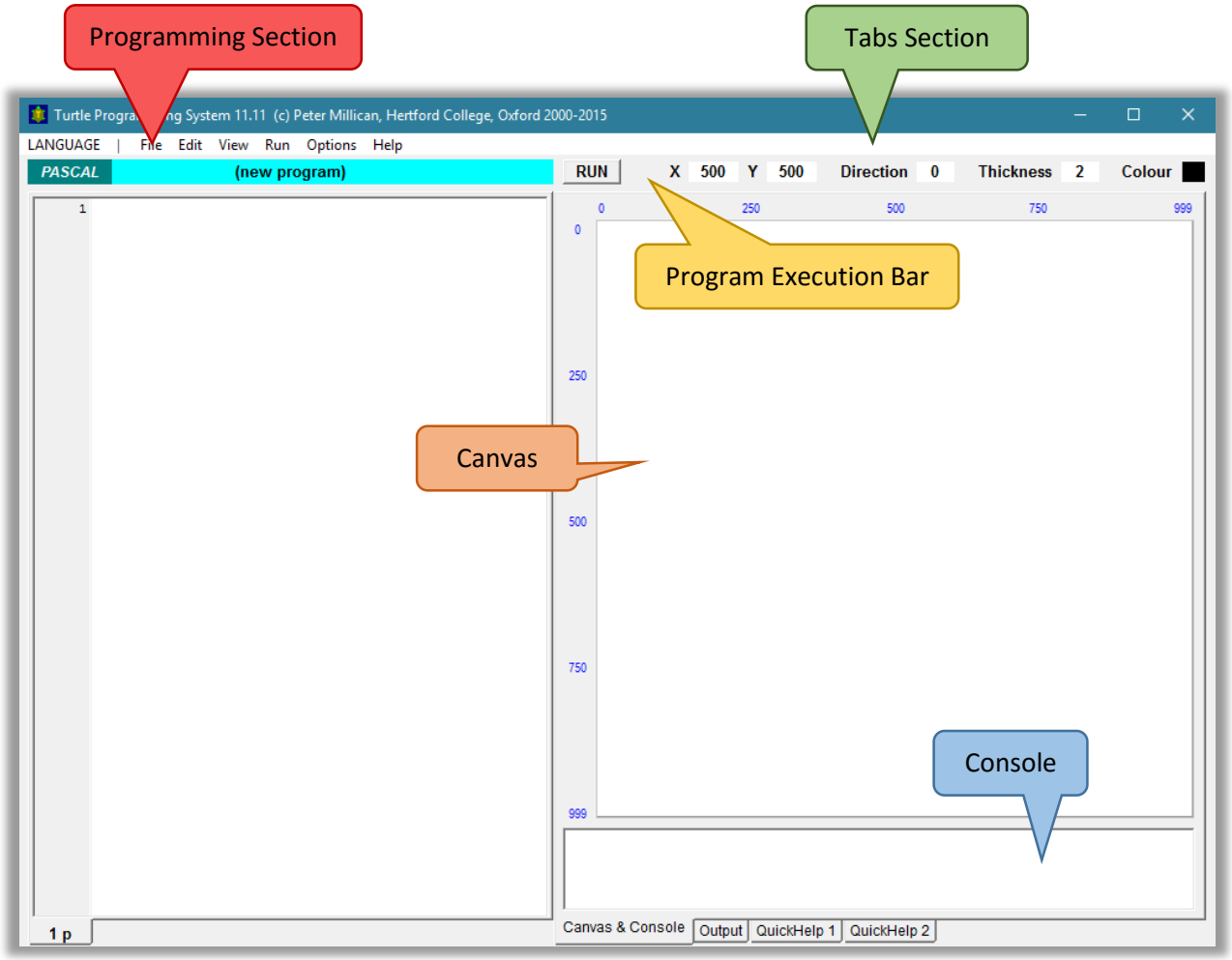
Guide for Normal Users

1. Interface Overview

The *Turtle System* is divided into two sections, the **programming section** on the left, and the **tabs section** on the right. The **programming section** is where you write your Turtle programs. You can have up to eight programs open at one time (if you attempt to open a ninth program, it will replace whatever is in the eighth slot, prompting you first if doing so will overwrite unsaved changes).

The **tabs section** on the right contains input and output displays for your programs. The first tab contains the **canvas** for graphical output, while the second tab contains a box for textual output. The smaller **console** underneath the **canvas** shows both textual output and textual input. The second two tabs—the "QuickHelp" tabs—contain information about how to program in the *Turtle System*. (This information is duplicated in the Help page of the *Online Turtle System*, at www.turtle.ox.ac.uk/online/help.)

Above all of the tabs on the right is the **program execution bar**, containing buttons to RUN, PAUSE, and HALT your program, and real-time displays of the Turtle's current properties (X and Y coordinates, direction, pen thickness, and pen colour).



2. Technical Overview: what happens when I click 'RUN'

Computer programs are typically written in some high-level language (like *BASIC*, *Java*, *Pascal*, or *Python*), which is relatively easy for human beings to read and understand. They are then "compiled" (translated) into machine code. Machine code is what the computer actually executes, and it is generally much harder for human beings to read and understand.

The *Turtle System* implements a virtual *Turtle Machine*, which is a software representation of an actual computer, designed to execute its own special machine code. When you write a program and click 'RUN', the first thing that happens is that the *System* attempts to compile your program into this machine code. If it succeeds, it will then send this code to the *Turtle Machine* to be executed, and you will see the results of your program on the **canvas** (and/or in the textual output). If it fails, because of errors in your code, it will throw up an appropriate error message, showing you the line number in which the error was encountered, and the most likely cause of the problem.

For users who are interested in learning more about machine code, and in how computers work "under the bonnet", the *Turtle System* provides a number of advanced features. To learn about these, please see the accompanying *Guide for Power Users*. For those who simply want to learn how to program, the present guide (and the associated documentation on the Turtle programming languages) is all you will need. To better understand the *System*, however, and to make best sense of this *Guide for Normal Users*, it is helpful to be aware of these two processes beneath the surface: first the program is compiled into machine code, and then this machine code is executed by the *Turtle Machine*.

3. The LANGUAGE Menu

The *Turtle System* supports programming in four languages, *Turtle BASIC*, *Turtle Java*, *Turtle Pascal*, and *Turtle Python*. (Note however that *Turtle Java* is currently being redeveloped, and has been disabled in the latest version of the software.) By default, the *System* opens in *Turtle Pascal*, but you can change between these languages at any time using the "LANGUAGE" menu.

Changing the language will change the language-specific help under the "QuickHelp 1" tab, but also—more importantly—will change the compiler to expect programs in the new language. Note that this can lead to some surprising error messages. If you try to compile a correct *Turtle Pascal* program in *Turtle Python*, for example, the compiler will very quickly encounter an error. No attempt is made to automatically guess which language your program has been written in, and this is by design: we do not wish to hide from users the fact that different languages have to be handled by different compilers.

4. The File Menu

The "File" menu contains the obvious entries you would expect, giving you the facilities to open and close programs, create new programs, save your programs to your computer, or print them. Here you can also save the graphical output of your programs, either by copying the **canvas** image to your clipboard, or saving it as a bitmap image file. In addition to creating a new (blank) program, you can also create a skeleton program, which saves you time by preloading the program editor with the minimum framework you need to get your program functioning.

When saving a program in the ordinary way, what is saved to your computer is simply a text file containing your program code. In addition to this, however, the "File" menu contains an option to "Save Export/Upload file". This larger file includes, alongside your code, additional information about your program (which you enter into the dialogue box that appears), and also—more importantly—the compiled machine code that the *Turtle Machine* runs when your program is executed. These precompiled files can be run immediately in the *Online Turtle System*, without having to be compiled.

5. The Edit Menu

The "Edit" menu contains the obvious facilities you would expect: undo, redo, cut, copy, paste. You also have the option here to store a copy of the program you are currently working on, and subsequently restore that copy—a quick way to temporarily save a version of your program before experimenting with some changes, allowing you to revert to the earlier version if those changes don't work out. Note that this will not save the program to your computer, and any copies stored in this way will be lost when you close the program (unless you have saved them in the usual way). You can store at most one copy for each open program in this way, and so clicking on this option a second time will overwrite the existing copy.

The "Edit" menu also contains an Auto-Format feature. This very useful feature will automatically arrange the code of your program into a standard and consistent format, making it much easier to read, and optionally removing any program comments. Clicking on "Auto-Format program" when in *Turtle Pascal* brings up a dialogue box with a number of options. Having changed the options, you can click "Apply" to save your preferences without formatting the current program, or click "Format" to save your preferences and format the current program according to those preferences. Clicking on "Auto-Format program" in *Turtle BASIC* or *Turtle Python* simply formats the program with the default options, as the options are not currently editable in these languages.

6. The View menu

By default, the *Turtle System* contains everything integrated into one window. In the "View" menu, however, you have the option to separate the **tabs section** on the right from the **programming section** on the left, creating two windows. Similarly, you can also place the **canvas** in its own window. This latter option is particularly useful if you want to change the size of the **canvas** on your screen. By default, the **canvas** is 1000 by 1000 pixels, but scaled down to take up only 500 by 500 pixels on your screen. By detaching it you can view it full size, or indeed at any size by simply resizing this separate window.

Under the "View" menu you can also change the font for the program editor, place the tabs at the top rather than the bottom of your window, and clear the **canvas**, **console**, and textual output of any content that was generated by your previously executed program.

At the bottom of the "View" menu is the option to "Display Power User Menu". Clicking this will bring up a number of advanced options in the menus. Those interested in these higher level features should consult the companion *Guide for Power Users*.

7. The Run Menu

The "Run" menu contains entries to RUN, HALT, and PAUSE your program. In addition, there are two runtime options, to show the "Canvas & Console" tab when your program starts, and to show the "Output" tab when textual output is generated. These will automatically change the currently viewed tab on the right hand side. Note that you can switch between these tabs during the running of your program using the `output` command. See the "String functions" example program under "Examples 2 - further commands and structures" in the "Help" menu for an illustration of this.

8. The Options Menu

There are three sections in the "Options" menu. In the first section, you can change the starting size of the **canvas** for all of your programs. By default, the **canvas** is set to 1000 by 1000 pixels, but here you can select either a smaller **canvas** (500 by 500) or a larger one (2000 by 2000). This setting effects how your program is compiled, rather than how it is executed: every compiled program begins with some setup code, creating a default keyboard buffer, positioning the Turtle in the centre of the **canvas**, and fixing the **canvas** size and

resolution. (Note that this option only concerns the initial setup; the **canvas** size and resolution can also be changed during your program, using the `canvas` and `resolution` commands.)

The second section concerns the directories that the *Turtle System* will place you in to start with, when you open and save programs. These can be *independent*, in which case the *System* will remember the last directory you opened a file from, and the last directory you saved a file to, taking you back to those (possibly different) directories the next time you open or save a program respectively. Or they can be *linked*, in which case the *System* will remember only one last directory (the last directory you *either* opened from *or* saved to), taking you back there the next time you either open or save a file. Finally they can be *flexible* (the default option), in which case they will be linked when you open and save a file to the same directory, but made independent when you save a file to a different directory from the one you last opened a file from.

The third section lets you save your preferred settings to an options file on your computer, so that the next time you run the *Turtle System* it will be set up the way you like it. To do this, simply arrange the *System* to suit your preferences, and then click on "Save current settings to options file". You can save this file anywhere on your computer, and call it whatever you like, keeping multiple settings files if you wish. You can then load them by clicking on "Load and apply settings from options file". However, if you save an options file in the same directory as the *Turtle System* executable, and call it "Default.tgo", the settings from that file will automatically be applied each time you start up the *System*. To revert to the default settings, simply delete or rename this file.

An options file is simply a text file with the options written in a regimented format. If you wish, you can open this file in a text editor, and edit it directly yourself. It may be fairly obvious how to change some of the options in this way, but to be sure of getting the results you want, it is always safer to use the *Turtle System* itself to generate these options files.

9. The Help menu

The "Help" menu contains a wide selection of example programs, which together illustrate most of the programming features available in the Turtle languages. These examples range from very simple programs intended for complete beginners, to very sophisticated programs meant for more experienced programmers. We encourage students to explore the features of the Turtle languages by reading, running, and editing these programs.

The final entry in the "Help" menu, "Load corresponding example on language switch", toggles an option. With this option turned on, if you are currently viewing an example program, and then change language (using the "LANGUAGE" menu), the *Turtle System* will automatically load the corresponding example from the new language. This is a very nice way of seeing at a glance how the various languages differ (and the ways in which they are similar).

There are two things to note about this option. First, there is no automated translation happening, and you cannot automatically convert your own programs from one language to another in this way. It is simply that we have written our example programs in each of the different languages. Secondly, there are currently more example programs for *Turtle Pascal* than for the other languages (chiefly because *Turtle Pascal* has support for arrays, which the other languages do not yet have). When you change language with this option set, therefore, you *may not* get the corresponding example, since the corresponding example may not exist; in this case, you will simply get a new, blank program. In due course all of the languages will be developed to the same level (and will all include support for arrays), and so this will not be a problem for long.