cT – Not Just Another Language

Russell C. Walker
Department of Mathematics
Carnegie Mellon University
Pittsburgh, PA 15213

Introduction. Modern computer applications emphasize powerful features such as high quality graphics, mouse interactions, a variety of text fonts, windows, and pop-up menus. While the end user finds these applications appealing and easy to use, only exceptionally skilled programmers can produce such applications with today´s programming tools. The cT[1] language is intended to make development of programs employing such features possible for those with far more limited programming experience.

The historical antecedents of cT are the TUTOR and MicroTutor languages developed by the Computer-based Education Research Laboratory for the PLATO System[2] at the University of Illinois at Urbana-Champaign. The TUTOR language was implemented on Control Data Corporation time sharing systems, and MicroTutor was designed to run on a variety of personal computers.

cT was first implemented on the Andrew environment of networked workstations at Carnegie Mellon University to provide students and faculty with a language to easily take advantage of the power of that system. Building on the efforts of the developers of MicroTutor to produce a language that would run on a variety of personal computers, cT has been implemented on several machines. cT programs can be run without change on the Macintosh, the IBM PC[3], and the IBM PS/2[4] series and the IBM RT PC, Sun 2, Sun 3, and VAXstation II models of advanced function workstations.

Further, because the main motivation of cT´s developers was the production of educational software, cT includes features to facilitate handling of student responses and the presentation of text and graphics. In order to be used by instructors with a minimum of assistance from professional programmers, cT includes a highly supportive development environment, including an on-line reference manual and informative error diagnostics. As the language has become better known, it has also begun to be used in the development of research tools.[5]

cT language features. cT includes the usual structures of loop, reloop, outloop including a unified treatment of until and while; case; and if, elseif, endif. In addition, it contains a number of commands specifically designed for educational software development.

A set of **graphing** commands generates axes and appropriate scaling for graphs. Tick marks and labels on the axes may be set by the programmer. Rotatable displays, pattern filled polygons and disks, and animated icons are also available.

Multi-font text including bold, italics, bigger, smaller, and centered are available. The display rectangle for text can be adjusted during execution.

Portability between computers is instant. Rescaling features of cT make it possible for the programmer to minimize the effort necessary to make displays adapt to differing screen sizes. Because of incompatibilities in fonts, one *does encounter difficulties in porting the program if special symbols are used.*

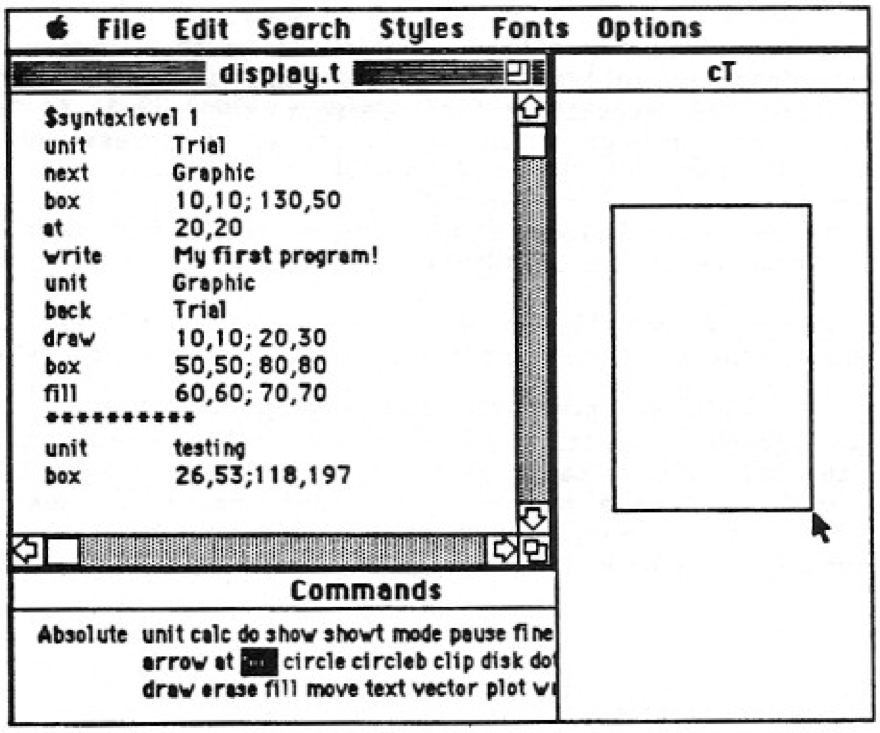
The **menu** command places a prompt associated with a routine in the program on a menu. *The type of the menu generated depends on the machine, i.e. pop-up* menus on a workstation, pull-down on a Mac.

On machines that permit several programs to run simultaneously, the **execute** command allows a cT program to initiate a Lisp, Fortran, or C program.

Response handling commands permit an entry from the keyboard to be examined for mathematical accuracy, for correct spelling, inclusion of keywords, or the inclusion of mathematical operators.

The usual **sequencing** commands are supplemented by menu entries for the next routine or the previous routine to facilitate moving through a sequence of lessons based upon a student´s progress.

The cT development environment. Because cT is designed to allow inexperienced programmers to develop sophisticated applications with speed and ease, a number of supportive features are built into the development environment. Several of the features are illustrated in the figure below from a Macintosh screen.



```
$syntexlevel 1
unit      Trial
next      Graphic
box       10,10; 130,50
at        20,20
write     My first program!
unit      Graphic
back      Trial
draw      10,10; 20,30
box       50,50; 80,80
fill      60,60; 70,70
**********
unit      testing
box       26,53;118,197
```

```
Absolute  unit calc do show showt mode pause fine
          arrow at     circle circleb clip disk do
          draw erase fill move text vector plot w
```

The **commands window** contains a list of cT commands. Clicking on a command causes that command to be entered in your program at the position of the insertion pointer. In the figure, the "box" command appears in reverse video in the commands window, indicating that that command was entered by a click in the commands window.

Graphics editing permits screen locations to be entered in the source code by means of mouse clicks. In the figure, the coordinates for the box command have been entered by clicks at the top left and the bottom right of the box. The requested box has then been drawn.

An on-line **reference manual** provides a tree-structured guide to the cT language. The manual is accessed by mouse clicks and includes executable examples that may be copied into a program.

cT programs are **incrementally compiled**, i.e. when a program is executed, the procedures needed first are compiled first. When changes are made, only the changed procedures are recompiled. This gives the speed of a compiled language, with a quick modify, test, re-modify cycle. A binary version of the program can be requested.

Useful **error diagnostics** are provided during both compilation and execution. Source code is scrolled to the location of the error, a message is displayed, and the insertion pointer placed at the position of the error.

Some sample cT applications.

Simplex Algorithm Mastery (Russell Walker, Mathematics Department, Carnegie Mellon University)

This program guides a student through the simplex algorithm to solve a linear program. Computations are done by the program, but the student is required to make all decisions regarding selection of pivots, formation of the dual, use of artificial variables, etc. Errors are detected, their consequences indicated, and the opportunity to correct the error is presented. For problems involving two or three variables, the progress toward solution is indicated on a graph of the set of feasible solutions.

Graphs & Tracks (David Trowbridge, Center for Development of Educational Computing, Carnegie Mellon University)

Graphs & Tracks I and II are aimed at the difficulties students have in making connections between observations of motion and graphs of that motion.

In part I, students are presented with a graph and must create a motion that matches the graph by setting up inclined tracks, positioning a ball, and starting the ball with a particular initial velocity. Part II goes the other way: the student views a motion on inclined tracks and must sketch a graph corresponding to the motion. There is abundant feedback to the student. These programs recently won EDUCOM/NCRIPTAL awards.

Handling of lab data (Robert Schumacher, Physics Department, Carnegie Mellon)

The ability of cT to present graphics, manipulate strings, and initiate processes in other languages makes it useful in the laboratory. In this application, a cT program accepts a binary data file, prepares it for processing by a FORTRAN program, and submits the FORTRAN program to a CRAY supercomputer. When the output is received, it strips off extraneous material and generates a graphic display of the result. This facility is used both to process research data and data from a modern physics experiment investigating chaos.

Fourier (Brad Keister and Harry Stumpf, Physics Department, Carnegie Mellon University)

Designed to help students visualize summation and convergence properties of Fourier series, this program begins with a tutorial which steps through the basic sequence of entering parameters and viewing results. The program calls for a functional form for the Fourier sine and cosine coefficients, along with the end points which define the period interval, and the function which the series is to represent. A histogram illustrates the relative weights of the coefficients. The user then enters the maximum number of terms to be summed, and the resulting partial sum is plotted alongside the anticipated function. Fourier is especially useful for examining convergence properties, Gibbs phenomena, etc.

----

1. cT is a service mark of Carnegie Mellon University.

2. The PLATO System is a development of the University of Illinois. PLATO is a registered trademark of Control Data Corporation.

3. IBM is a registered trademark. PC and PS/2 are trademarks of the International Business Machines Corporation. Macintosh is a registered trademark of Apple Computers, Inc.

4. Announcement of PC family implementation expected in early 1989.

5. For a more extensive discussion of cT, see "The cT Language and Its Uses: A Modern Programming Tool" by B. A. Sherwood and J. N. Sherwood, to appear in the Proceedings of the Conference on Computers in Physics Instruction, North Carolina State University, August 1-5, 1988.