A Virtual Laboratory For The Introductory Engineering Course

Professor Richard J. Reid
Computer Science Department
Michigan State University

Abstract - The new version of our introductory course for engineering students gives the students experience with virtual (physical) devices and has them also learn about computer-based tools for working with the underlying mathematics and physics, and for reporting their accomplishments. Students still learn about programming, but without much of the syntactic and semantic rigors of a computer language *per se*, as was previously emphasized.

The course includes sections on document preparation, worksheets, matrix mathematics and Visual Basic, but the emphasis is on the mathematics and science as they support engineering design.

Introduction

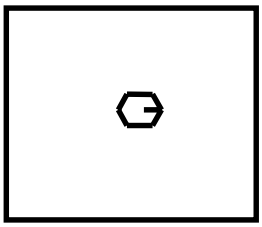
Students come to this technical computing course usually as first or second semester freshmen. The course is required for or elected by most Engineering College students, and is completed by about 400 students each semester.

The prerequisite mathematics background is only college algebra. Other mathematics, including matrix algebra and linear recurrence relations, is introduced as needed for laboratory exercises. Numerical evaluations of distance, speed and acceleration are encountered and the relation to derivatives of polynomials is illustrated in passing (most students in the course are in their calculus courses). Statistics are encountered in exercises, but only simple probability, frequency counts and histograms are developed and used. Our intention is that, if a problem area is interesting, we will develop whatever mathematics is necessary for support using numerical examples and illustrations.

The newest feature of the course is the inclusion of laboratory exercises that require the students to use virtual devices in the laboratory, or on their own computers, and to control them and/or gather data from them. This paper reports on two of these virtual devices: a hexagonal robot, the Hexobot, and an adjustable-length pendulum.

The Hexobot

At the time course materials were being developed for this past year, Pathfinder and Sojourner were just landing on Mars. This event, and the knowledge of the growing importance of mechatronics[1] that most of our engineering students will be facing, gave rise to the Hexobot. The Hexobot is a hexagonal robot patterned after Karel the Robot[2] and RoBOTL[3]. As an introduction to the Hexobot the students will be using, an actual computer-controlled robotic vehicle is demonstrated in the lecture.



The Hexobot

The Hexobot incorporates features of mathematics and computer science in a simple setting for controlling its actions. The Hexobot accepts primitive commands such as "f" for "move forward", "d" for "put the pen down", etc., and can be directed to draw pictures on the screen. The Hexobot accepts command groups and repetition factors, and has a function facility that allows for the definition and invocation of functions, and supports sequential, conditional, iterative and recursive statements. Students create individualized drawings and ultimately cut and paste so they have a hand-in document reporting their creation and including a listing of the file of commands developed, and the drawing made by their Hexobot.

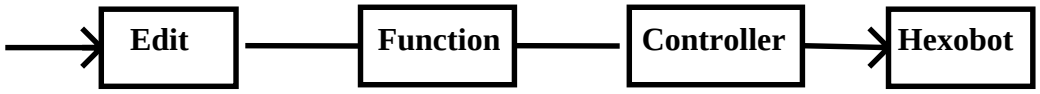
The following primitive commands can be issued:

Primitive Commands	
Command	
Character	**Action**
R	Reset and clear the display window.
b	Move backward.
c	Next drawing color. Cyclic sequence: Black, Red
	Green,Blue,Custom,White
d	Put the pen down to leave a trail.
f	Move forward.
l	Left turn, CCW, 10 degrees.
n	Narrow the trailing line.
r	Right turn, CW, 10 degrees.
u	Move the pen up, so there is no trail.
w	Widen the trail line.
+	Push the Hexobot State onto the State Stack.
-	Pop the previous State from the State Stack.
!	Beep

These commands are repeated to give the desired result. (For example, to put the pen down and move eight steps forward: d f f f f f f f f.)

The Hexobot State, which may be saved and retrieved using the "+" and "-" commands, consists of the Hexobot position, orientation and pen: up/down, width and color.

A Hexobot Controller accepts more complicated control patterns of inputs and renders them as primitive commands sent to the Hexobot.



Sequence of Command Processing

Controller Commands	
Command	**Action**
%	Begin a comment that extends to the end of the current line.
[...]	Group multiple commands.
<Integer>	Repetition factor for primitive or grouped commands,
	maximum value: 999.
^	Cancel the repetition factor just entered.
.	Repeat the last command group.
<blank>	Space for readability.

Repetition factors and grouping can be combined and used recursively. For example, the command to draw a square 10 steps on a side is given as:

```
d 4 [ 10f  9r ]
```

Or, to draw a circular array of squares with color variations as:

```
d  18 [ 4 [ 20f 9r ] 2r c ]
```

Functions -- A simple function facility is provided for use in Hexobot control. The function facility is derived from work by Strachey[4], and uses a notation that differs slightly from common mathematical notation. Instead of function references being specified as f(x), they are given as (f,x). This notational variation allows the simple recognition of a left parenthesis, "(", in the input stream, to divert the input away from immediate presentation to the Hexobot Controller as a command. The diversion continues until the matching right parenthesis, ")", is encountered.

Built-In Functions	
Characters	**Action**
( )	Function reference using prefix notation.
,	Separator for function arguments.
=	Function for defining and/or redefining functions.
	Example: (=,A,25) (A) yields 25
+,-,*,/,%	Functions for: Add,subtract,multiply,divide,remainder.
	Only <+,-,0,...,9> in the arguments.
	Examples: (+,12,-13) yields -1; (*,3,(+,4,5)) yields 27

>,>=,==,~=,<=,<	Relational functions for comparisons.
And, Or, Not	Logical functions. Numeric 1 = True, 0 = False.
	Example: (And,(>,5,4),(<=,2,2)) yields 1
(<Integer>,...)	Function for forming repetitions of the argument.
	Example: (3,abc) yields abc abc abc .
{ }	Shielding quotes to prevent or delay evaluation.
	One pair of quotes is removed *as* an evaluation.
$	Marker for arguments in function definitions.
	Example: (=,F,{$2$1}) (F,There,Hello) -> Hello There
(ifelse,a,b,c,d)	Comparison function, if a = b give c otherwise give d.
	Examples: (ifelse,125,(*,5,25),yes,no) yields yes
	(ifelse,abc,abx,one,two) yields two
(for, , , ,)	Iteration function. Example: (for,x,1,4,{(x) }) yields 1 2 3 4
(RGB,rrr,ggg,bbb)	Set and change to a custom drawing color, having
	primary-color components red, green and blue. Each
	parameter is in the range: 0 <= rrr <= 255
(?,<status>)	<status>, Gives the Hexobot's:
	A -- Angular direction (in 10 degree steps from North).
	X -- X-coordinate position.
	Y -- Y-coordinate position.
	Example: (?,A) yields 9 (East)
(Rand)	Move the Hexobot to a random position.
(Here)	Give the Hexobot's position and direction: x, y, a.
(MoveTo,x,y,a)	Move the Hexobot to a given position and direction.

Sometimes the processing during the diversion results only in "side effects", sequences being named, parameterized, remembered, etc. Other times, the processing during the diversion produces output characters that are sent onwards to the Hexobot Controller.
Functions are evaluated by first evaluating each of the arguments in a left-to-right scan. Any of the argument evaluations, including the zeroth (the function name), can invoke other functions in a recursive manner. After all the argument evaluations are completed, signaled by the closing right parenthesis, ")", the designated function is called. The called function's definition is scanned, possibly calling other functions as may be required by the definition. Whenever an argument reference ($) is encountered in the body of the definition, the corresponding actual argument of the call is copied into that position without further evaluation. Extra calling arguments are harmless, and null is used for missing arguments.
Examples of naming and parameterizing:

```
(=,Square,{ (4, $1f  9r )})                 % Define parameterized form of "Square".

d (Square,25) 18r (Square,50)               % Use "Square" in two sizes.

(=,Pair,{d (Square,$1) 18r (Square,$2)})    % Define a Pair of Squares.
```

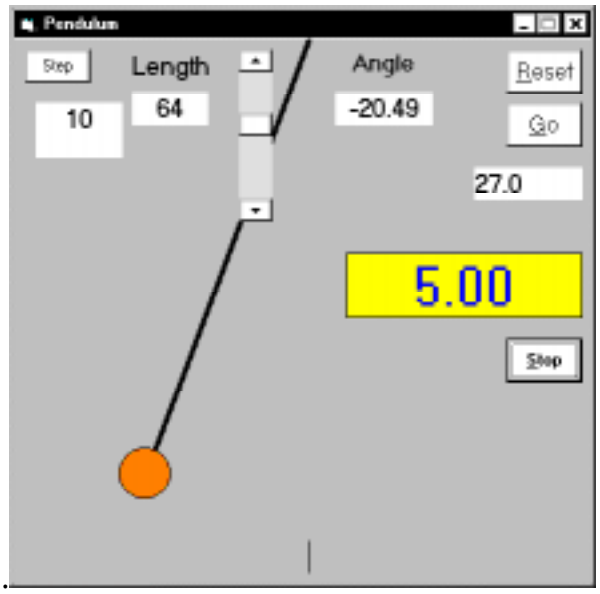
```
9r (Pair,35,65)                    % Draw a Pair.
```

Input is normally acted upon immediately allowing no chance for correcting typing mistakes. Clicking on the "Edit" box brings up an editor window that allows editing input before it is sent to the Function facility.

The Pendulum

Another device used in the laboratory is an adjustable suspension-length pendulum, with a built-in stopwatch. This pendulum is animated and uses a recurrence formulation for its mathematical model, so it is not restricted to small-angle oscillations. As an introduction to this exercise, an actual pendulum is demonstrated in the lecture, and its swing-period is compared to the virtual device for several suspension lengths.

In the laboratory the students adjust the length of the pendulum using the slider control and start it swinging. The stopwatch is started at a desirable point on the swing, and some number of swings are observed and counted. The stopwatch is stopped at the end of the last counted swing. Students use a worksheet to tabulate the suspension length, number of swings and the elapsed time. A worksheet calculation gives the times of the periods for each of the lengths chosen.



As a part of this exercise students experimentally select the constant of proportionality between the square-root of the pendulum length and the period, and compare their experimental results with the evaluation of the mathematical model.

Conclusions

By using simulation models of such things as the Hexobot and the pendulum, students encounter the math, science and engineering of real (virtual) devices and are able to work with them conveniently.

Having these virtual devices available gives more of a “real world”-like experience in performing the required experiments. Of course it is still virtual, but it is a practical laboratory encounter for courses that must enroll so many students.

Other virtual devices that have been developed and used include a ball falling in a gravitational field, a satellite orbiter and a Bungy Jump simulation.

Course materials can be found at the URL: http://www.cps.msu.edu/~cps131

References

1. S. B. Niku, "Teaching Mechatronics to First-Year Engineering Students," Computers in Education Journal, Vol. VII, No. 3, pp. 6-9, July-September 1997.
2. R. E. Pattis; Revised by J. Roberts and M. Stehlik, "Karel the Robot," 2nd ed. John Wiley & Sons, New York, 1995.
3. K. A. Lemone and W. Ching, "Easing into C++: Experiences with RoBOTL," ACM SIGCSE Bulletin, Vol. 28, No. 4, pp. 45-49, December 1996.
4. C. Strachey, "A General Purpose Macrogenerator," Computer Journal, Vol. 8, No. 3, pp. 225-241, August 1965.

RICHARD J. REID received B.S. and M.S. degrees in electrical engineering from Iowa State University, and the Ph.D. degree in electrical engineering from Michigan State University. Since 1965 he has been a Professor of Computer Science at Michigan State University. His current interests are in computer architecture and simulation.