# Cover Letter

From: Michael Bruce-Lockhart and Theodore S. Norvell  
Electrical and Computer Engineering  
Faculty of Engineering and Applied Science  
Memorial University of Newfoundland  
St. John’s, NL, A1B 3X5  
[mpbl@mun.ca](mailto:mpbl@mun.ca) , [theo@mun.ca](mailto:theo@mun.ca)  
ph. (709) 737-8962, fax (709) 737-4042

To: The awards committee, Premier Awards.

Dear Sirs and Madams,

Please find enclosed our submission for The Teaching Machine and WebWriter++. As the authors of this software, we are entitled to submit it for competition.

WebWriter++ is copyright of Michael Bruce-Lockhart. The Teaching Machine is copyright of Michael Bruce-Lockhart and Theodore S. Norvell, except the editor component of the Teaching Machine, which is copyright of Hao Sun (a former student). Derek Reilly authored much of the C++ compiler and much of the compiler framework. The example websites were authored by (and are copyright of) Michael Bruce-Lockhart and Theodore S. Norvell. All authors have been contacted about our submission. Hao is currently a software engineer in Calgary and Derek is a post-doc at Georgia Tech.

We hereby permit Engineering Pathways/NEEDS to distribute the Teaching Machine, WebWriter++ and all accompanying materials submitted to the Premier awards.

Here follows a brief description of the Teaching Machine and WebWriter++:

The Teaching Machine is a program animation system. It reveals to the student exactly what happens inside a computer when a computer program executes; that is it animates the relationship between the static text of a computer program and its dynamic computations.

WebWriter++ is a JavaScript library to aid instructors in creating interactive websites for courses, especially courses on programming. WebWriter++ makes it easy to embed code examples on web pages —handling the conversion from programming language to HTML, including excerpting and syntax highlighting; such examples may be sent to the Teaching Machine by a click of a button.

Sincerely,

Michael Bruce-Lockhart and Theodore S. Norvell

# Dean and Departmental Chair

Our Dean is

John Quaicoe  
Dean, Faculty of Engineering and Applied Science  
Memorial University of Newfoundland  
Arctic Avenue  
St. John’s, NL  
A1B 3X5  
Canada  
[jquaicoe@mun.ca](mailto:jquaicoe@mun.ca)  
phone (709) 737-8810, fax (709) 737-4042

Our Chair is

Dennis Peters  
Chair, Electrical and Computer Engineering  
Memorial University of Newfoundland  
Arctic Avenue  
St. John’s, NL  
A1B 3X5  
Canada  
[dpeters@mun.ca](mailto:dpeters@mun.ca)  
phone (709) 737-8929, fax (709) 737-4042

# Video and Distribution

We have prepared two demonstration videos to explain the Teaching Machine and WebWriter++. These may be found at

http://www.teachingmachine.org/premier/tm.swf

and

http://www.teachingmachine.org/premier/ww.swf

A distribution of the Teaching Machine and WebWriter++ can be found at

http://www.teachingmachine.org/downloads.htm

# Body

## Description of the Teaching Machine

[Before reading on, we suggest viewing our videos on the Teaching Machine and WebWriter++]

The Teaching Machine is a program animation system. It reads in programs written in either C++ or Java, compiles them to an intermediate form and then executes the programs step by step, on a virtual machine, under user control. As execution proceeds, the state of the virtual machine is displayed back to the user in visual form in a number of views. These views include

* Code View. The original text of the program is presented with syntax highlighting. The currently executing line is highlighted with a yellow background. The code view auto-scrolls as the current line changes.
* Stack, Static and Heap Views. These views present the state of the memory in a tabular form. Locations are identified by variable name (if any) as well as address. Values are shown symbolically by default (e.g. in decimal for numbers), but can be shown in binary, if the user wishes. Linked View. An alternative view of memory. Objects and arrays are represented by boxes within boxes. Pointers are represented by arrows between boxes.
* Symbol-table View. Shows the type and address of each variable currently in scope.
* The Expression Engine. Shows the current state of evaluation of an expression. As an expression is evaluated, its representation in the Expression Engine is reduced, operation-by-operation, to a single, final value.

The selection, positioning and labelling of the views may be controlled by the user or by a configuration file. For example in the accompanying Teaching Machine video, the Teaching Machine was set up for an introductory course with the stack view labelled “Local Memory” and the static view labelled “Global Memory”, while the heap view was suppressed completely.

The user, who may be either a learner or an instructor, can advance the state of the computation by a choice of actions at various temporal granularities:

* Go forward. One operation is executed.
* Step into. The next expression is executed. If a subroutine call is made, execution stops at the start of the subroutine.
* Step over. The next expression is executed. If a subroutine call is made, execution does not stop for the subroutine call.
* Step out. The program is executed until the return from the current subroutine.
* Step to line. The program is executed until a chosen line is reached.
* Go back. The computation is backed up to the previous stopping point.

The Teaching Machine can run either as a stand-alone application, as an Eclipse plug-in, or as a web applet. Written in Java, it runs on the Windows, Unix, Linux, and Mac operating systems.

The Teaching Machine contains compilers for both C++ and Java. The C++ compiler supports a large subset of the language including classes and objects. It does not support templates and exception handling. The Java compiler supports almost all of Java 1.4. It does not support concurrency or generics.

## Description of WebWriter++

WebWriter++ is a JavaScript library for authoring interactive course websites, especially for programming courses. It is designed to allow instructors, with a moderate investment of time, to focus on content while creating effective interactive websites that support multiple learning modes.

Some of its many features include.

* Automated table of contents generation.
* Automated navigation button generation.
* Multiple view modes: Lecture and self-study.
* Hidden notes. These are useful for providing extra information that will be skipped during the lecture.
* Glossary. Keywords may be defined in a glossary and uses of these words are shown in a special colour. Hovering the mouse over such a use causes the glossary entry to appear in a popup window.
* Popups. Besides glossary entries, other popup windows can be included.
* Code excerpts. Excerpts from program files (i.e. C++ or Java files) can be included on the web page. The portion of the file to be included is tagged by special comments. The excerpt is converted to HTML and is presented with syntax highlighting.
* Pop-ups in code. By means of special comments in the code, pop-ups can be included in code excerpts.
* Interface to videos. Each code excerpt may be optionally associated with a video. WebWriter++ adds a button to the excerpt’s banner to launch the video.
* Interface to the Teaching Machine. Each code excerpt may optionally be sent to the Teaching Machine. WebWriter++ adds a button to the excerpt’s banner that allows the reader to send the excerpt to the Teaching Machine. In fact the whole file is sent, but the designer has the option of asking the Teaching Machine to display only the excerpt. The Teaching Machine appears in its own window and that window is brought to the fore.

This last point is what most connects the Teaching Machine and WebWriter++ and is the reason that, while each is powerful on its own, the combination is particularly effective.

## Response to the Premier Award Evaluation Criteria

### Learning Objectives: Learning objectives and goals are clearly stated and supported by the software and learning experience.

The Teaching Machine and WebWriter++ are tools that instructors can use to create learning experiences. WebWriter++ allows excerpting of code examples in order to focus the student’s attention only on the most relevant section of code. The Teaching Machine allows configuration of its views and excerpting of code so that the student can focus only on the most relevant aspects of behaviour.

### Interactivity: The learner is actively involved in the learning process—the interaction enhances learning.

WebWriter++ sites may be used by the instructor in the classroom or by the student on their own. In the latter mode, the student is free to interact with the web-site. For each example, the student can also chose to watch a prepared video of an instructor using the Teaching Machine or they can choose to use the Teaching Machine themselves. The student can even alter the code of the examples before they are sent to the Teaching Machine. The Teaching Machine allows the student to control not only the pace of execution, but also its granularity, with the possibility of backing up to an earlier point in the computation and re-executing it, possibly at a different granularity.

### Cognition/Conceptual Change: Learning appears to be significant and long lasting—strong and useful cognitive models can be built.

The main motivation behind the Teaching Machine is to help students build effective mental models for understanding computations and programs. It does this by illustrating the state of computations using an intuitive visual vocabulary. That is, it helps to build conceptual mental models of computations and their relationship to the program text. These mental models then form a structure on which to build new knowledge. The learner can connect the actions shown in the “expression engine” with the data in and the changes to the various views of memory (stack, static, heap, and linked).

To monitor their learning, students can anticipate the behaviour of the Teaching Machine and, when that behaviour differs from expectation, they can go back (using the Go Back action) and re-execute, perhaps using a finer granularity of time, to see exactly why the actual behaviour differs from the expected.

### Content: The content is well chosen and structured.

WebWrtiter++ and the Teaching Machine are tools for developing and presenting content; they are not content themselves. These tools encourage content producers to *show* the student material rather than simply *talk about* material.

### Multimedia use: Multimedia is used effectively and promotes the learning objectives and goals.

The Teaching Machine is in a sense a new medium. WebWriter++ enhances an existing medium —websites. The same code example can be shown on the web page, surrounded by written explanation; in a video demonstration, using the Teaching Machine together with verbal explanation; and in the Teaching Machine, so that the student can interact directly with the example. These various views are consistent because they all start with the same C++ or Java file.

### Instructional Use/Adaptability: The software can be used in a variety of settings.

WebWriter++ sites are styled via cascading style sheets (CSS) and can be switched between classroom style (large fonts) and self-study mode (small fonts). A third set of CSS is used for printing. Because all scripting is client-side, WebWriter++ sites, if downloaded, can be used without any connection to the Internet.

The Teaching Machine can be used as a stand-alone application, as an applet embedded in a webpage, or as an applet running in a dedicated window —as it is when invoked from a WebWriter++ site. The Teaching Machine can also be invoked from inside the Eclipse integrated development environment.

The Teaching Machine presents the state of the computer at various levels of abstraction. In early courses in programming, we use a simple model of memory in which the contents of the stack are presented as a table. For more advanced courses, such as courses in Data Structures or object-oriented programming, objects are depicted as boxes containing fields with arrows between them indicating pointers. Both views can be presented together to show that pointers are simply numerical addresses. The instructor can create configurations of the Teaching Machine that present only certain views and can control their size, position, and labelling.

The Teaching Machine is designed as a system of plug-ins and the ambitious instructor can create their own visualizations by creating a new plug-in. We have done this to support the presentation of arrays as images. A student team has created a view plug-in that shows a simulated robot moving around a course.

Both the Teaching Machine and WebWriter++ improve the way the instructor spends their time. The Teaching Machine allows instructors to spend classroom time more effectively. Rather than the instructor spending time creating pictures representing machine state on the board or spending time before class preparing such images, the Teaching Machine not only creates the needed images, but builds them incrementally and animates them. WebWriter++ allows the instructor to create interactive instructional websites in a fraction of the time that it would take otherwise.

### Engagement: The software holds the interest of a diversity of learners

Student engagement is one of the prime motivations for the Teaching Machine. “Seeing” is more engaging than “reading about”. In self-study, the student controls the actions of the Teaching Machine. They can use it to provide various inputs to programs and they can explore the computation at various temporal granularities and using the views of their choice. The ability to back computations up allows the student to review tricky parts multiple times, with different granularity, if they wish.

### Learner Interface and Navigation: The software is easy to use.

The interface to the Teaching Machine is fairly simple. Computations can be driven forward by one operation, one expression, or one subroutine call. The various view windows can be easily moved, stretched, minimized, and restored by obvious gestures.

WebWriter++ sites have table of contents and navigation buttons automatically generated.

### Technical Reliability: The software is free from technical problems.

We have tried to make the software reliable and resilient. The Teaching Machine uses the model/view pattern consistently so that all views are updated according to the current state of the model at the end of each user interaction.

### Engineering Content: Accuracy

The Teaching Machine supports large subsets of C++ and Java. Within those subsets we have tried to be faithful to the respective language standards. This allows the student to see a faithful depiction of the execution of programs written in these languages.

## Description of the impact of this courseware.

Websites generated with the aid of WebWriter++, and using the Teaching Machine, have been used at Memorial University in a sequence of three undergraduate courses: *Introduction to Programming*, *Advanced Programming*, and *Data Structures* This sequence of courses use C++. In these courses, depending on the instructor, WebWriter++/Teaching Machine sites are used either as the primary lecture notes for classroom lectures, or as an adjunct to non-interactive (e.g. Powerpoint) lecture notes. The sites are also intended to be used by the students for self-study and review. Although these courses are not now offered by distance, the predecessor to *Introduction to programming* was offered as a distance course with a WebWriter++/Teaching Machine site as the primary learning resource, effectively acting as an interactive e-textbook. Between 200 to 300 first year Engineering students take *Introduction to Programming* each year and WebWriter++ and Teaching Machine have been used in that course since 2004 so about 1500 students, at both campuses of Memorial University, have been taught at least one course using the software. All students taking Electrical Engineering and Computer Engineering (about 50 to 60 per year) over the past 8 years would have used the WebWriter++/Teaching Machine sites in one or two further courses.

In addition, WebWriter++/Teaching Machine sites have been used in a module introducing the Java programming language.

The Teaching Machine has also been used in courses at the University of Athens and the University of Florence.

## Description of how the courseware is used by a learner.

The learner uses the Teaching Machine to step through example programs and to see the effect of executing the program on given input. As the program is stepped, the state of the computer is shown to evolve under the control of the program. The primary pedagogical goal supported by the Teaching Machine is the building of effective mental models of program execution. The Teaching Machine can also be quite effective in explaining the subtleties of programming languages. (E.g., what is the difference between i/2 and i/2.0, or what is the difference, in Java, between p==q and p.equals(q), or what is the difference, in C++, between an array and a pointer.)

The Teaching Machine can be used on examples created by the learner. (Our Eclipse plug-in is particularly helpful for this.) Or it can be used on examples prepared by the course instructor or another courseware developer. In this case the best thing to do is to create a WebWriter++ site that presents the examples in context. Together with screen capture programs (such as Adobe’s Captivate), the Teaching Machine can also be used to create videos that illustrate various aspects of the course material. Such websites can be used out of class (or in a distance education setting), directly by the learner, or in the classroom (or virtual classroom), by the instructor.

## Description of the evaluation and assessment

To date we have not done controlled experiments to test the effectiveness of the Teaching Machine. We have done a number of questionnaires for students taking courses using the Teaching Machine and have found that students are quite positive about the software. Over the years a number of instructors have taught the *Introduction to Programming*  course and all have stuck with using the Teaching Machine and the WebWriter+ site for the course.

## Similar Programs

The Teaching Machine is a program animation system and there are a number of other program animation systems in use. Probably the most prominent is Jelliot 3. jGRASP also has some similarity. In addition debuggers can be used to achieve a similar effect. However the following features are unique to the Teaching Machine as far as we know:

* Multiple views of memory. We present both tabular and graphical (linked) views of memory. This makes the Teaching Machine suitable across a wide variety of courses.
* Back-up. Any number of computation steps can be undone.
* Fine-grained execution and Expression Engine. The computation can be stepped one operation at a time. An operation might be a memory fetch, a memory store, a type conversion, or an arithmetic operation. The Expression Engine shows the effect of the execution of each operation by reducing the expression in the Expression Engine and by changes to the memory.
* Multiple source languages. The Teaching Machine handles both C++ and Java. Plug-ins for other languages are certainly possible.
* Plug-in architecture. One can create specialized views through the use of plug-ins.
* Scripting. The program under execution can control the Teaching Machine and its plug-ins via special subroutine calls. These calls can be hidden from the student.

While there are many JavaScript libraries that allow source code to be included on Webpages, WebWriter++ does far more than that and it provides a conduit to the Teaching Machine.