ISSN 2277-3061

Memory Transfer Language: an absolute learner-driven visualizer

Leonard J. Mselle
School of Informatics, University of Dodoma,
P.O. Box 490 Dodoma, Tanzania UR

ABSTRACT

This article discusses visualization as a technique to enhance programming comprehension. It points out that current animation approach, being machine-based technique, is inadequate due to the fact that machine-based animation tools are difficult to integrate in the current teaching materials. In addition, machine-centered animators do not guarantee the learner with absolute engagement. In this paper, MTL, as a visualization technique which is absolutely learner-driven is demonstrated and discussed. It is shown that MTL can be integrated with current materials for teaching and learning programming and it can guarantee absolute authority to the learner.

Keywords

Visualization; Memory Transfer Language (MTL); Programming.

Council for Innovative Research
Peer Review Research Publishing System
Journal: INTERNATION JOURNAL OF COMPUTERS AND TECHNOLOGY
Vol. 13, No. 9
editorijctonline@gmail.com
www.ijctonline.com, www.cirworld.com

ISSN 2277-3061

1. INTRODUCTION

From the beginning of the computer age, people have sought easier ways to learn, express, and understand computational ideas. To-date, the quest to make code easier to express, manipulate, and understand by a broader group of people is an ongoing challenge. Various researchers have reported about the positive impact of visualization in teaching programming to novices [4, 7, and 16]. However, Hundhausen et al [5] report that visualization is not widely popular among programming instructors. They confirm that one of the main reasons is because the teachers responsible for the courses refuse to use new methods in teaching. They also confirm that the sole use of visualization systems doesn't necessarily improve the learning results. They argue that it is more important to engage the learners in the subject using visualization system as an aid.

All popular visualization tools are machine-based and or language specific [1, 3, 5, 6, 7, 17, 18, 19] This property, apart from being divorced from the conventional modes of presenting programming materials it denies the learner of two important cognitive advantages: firstly, the opportunity to visualize the program outside the machine environment hence, perpetuating the machine authority at the expense of the learners' authority and secondly, the opportunity for the learner to enjoy absolute engagement during the entire programming process [3].

2. A BRIEF REVISION OF GENERAL ANIMATION FRAMEWORKS

The animation framework exploits machine capabilities to reveal what is actually taking place in the computer memory. In their elementary nature, codes in animators are prewritten by experts. Learners run the animator to see the changes that take place inside variables. This framework of animation is depicted in Figure 1.

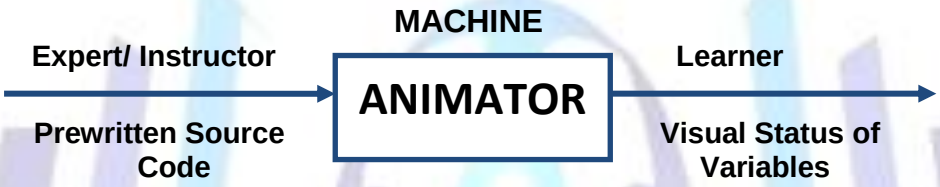

Figure 1: Machine-based animation framework (alternative one)

Machine-based animation highlights code-lines while providing the visual status of what is taking place in the computer memory as the lines are executed. The permanent involvement of the expert and the machine in the entire process denies the learner of the opportunity to initiate the code and scan its meaning visually in his/her mind outside the machine.

The second machine-based animation framework provides the learners with opportunity to modify and even write their own source codes [18]. This framework is more engaging since it gives the learners the opportunity to scan their codes in their minds during writing. This framework is presented in Figure 2.

Figure 2: Machine-based animation process (alternative two)

Although this framework provides the learner with the opportunity to originate the code in his/her mind it denies him/her the opportunity of scanning its meaning visually in his/her mind outside of the machine. Again, learners' authority and engagement is partly taken away by the machine and the expert.

3. MTL AS AN ABSOLUTE LEARNER-DRIVEN VISUALIZATION TOOL

Algorithm visualizations (AV) are used in computer science education since the early eighties. In spite of their educational potential, they have not been incorporated into the mainstream of computer science education. This lack of use has two main reasons; from the instructors' point of view, animations are not usually easy to use, deploy and adapt. From the students' point of view, more interaction, than just viewing animations, is needed to obtain learning improvements [5].

MTL-based animation is conceived on the assumptions that; all high-level language statements can be sketched by the learner to reflect the dynamic characteristics of variables as lines of code are executed. Such sketches can be used by the learner to debug, analyze, close-track and understand a code outside of the machine, hence centering the entire process on the learner. The same sketches can be integrated in the current materials used to present programming materials. Figure 2 is the general schematic representation of MTL-based animation.

Figure 3: General framework for MTL-based animation

As shown in Figure 3 the entire process is driven by the learner. That is from writing the source code, scanning its visual reality and producing the visual outputs. During the entire process there is neither machine nor expert intervention.

4. DEMONSTRATION OF MTL AS EMPLOYED IN ALGORITHM VISUALIZATION

Consider the code as represented by Program 1 in Figure 4. MTL is used to visualize variable declaration, data feeding (assignment) and data operation (addition).

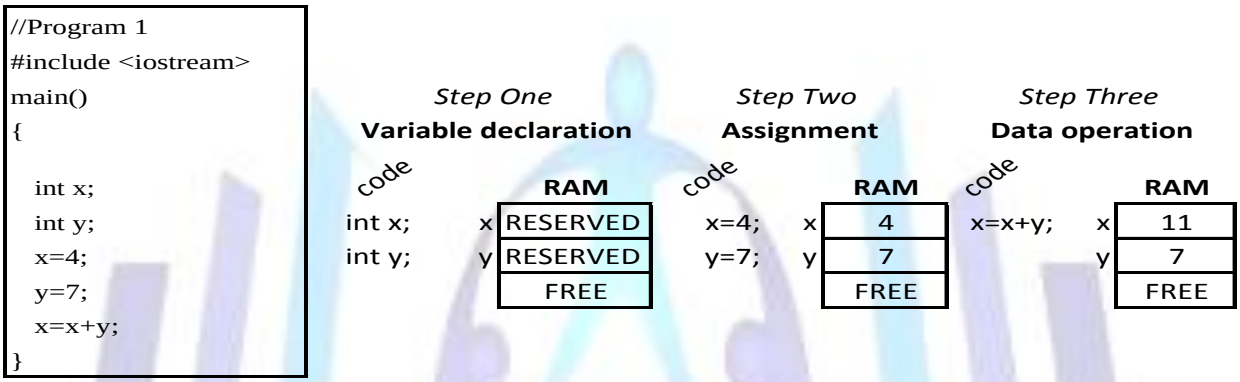

Figure 4: MTL Visualization of variable declaration, assignment and data operation

Visualization of flow of control (*while loop*) by MTL is demonstrated in Figure 5 which is an interpretation of Program 3 (insert).

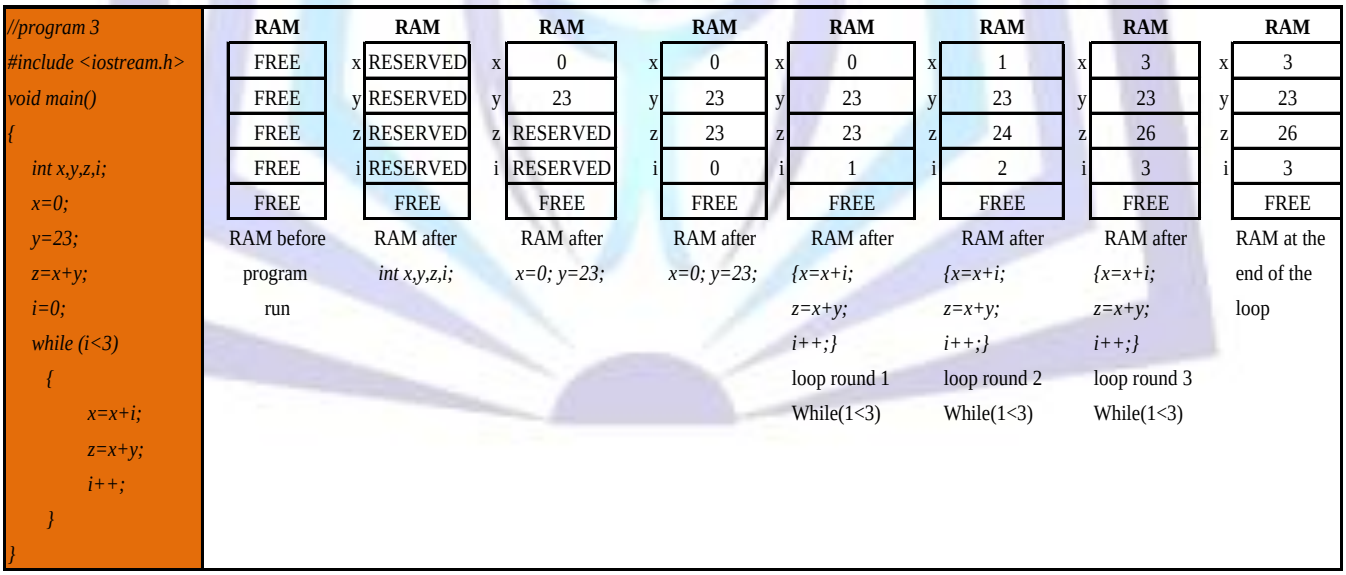

Figure 5: Code interpretation (while loop) using MTL

C Version

```
1  //Program 1
2  #include <io.h>
3
4  void main()
5  {
6     int x;
7     int y;
8     double z;
9     scanf("%d",&x);
10    scanf("%d",&y);
11    if (x>y)
12       z=x/y;
13    else
14       z=y/z;
15    println(z);
16 }
```

C++ Version

```
1  //Program 1
2  #include <iostream.h>
3  using namespace std;
4  void main()
5  {
6     int x;
7     int y;
8     double z;
9     cin>>x;
10    cin>>y;
11    if (x>y)
12       z=x/y;
13    else
14       z=y/z;
15    cout<<z<<endl;
16 }
```

Java Version

```
1  //Program 1
2  public class sone {
3
4     public static void main(String[] args)
5  {
6        int x;
7        int y;
8        double z;
9        x=TextIO.getln();
10       y=TextIO.getln();
11       if (x>y)
12          z=x/y;
13       else
14          z=y/z;
15       System.out.println(z);
16    }
17 }
```

Figure 6: Code in parallel languages

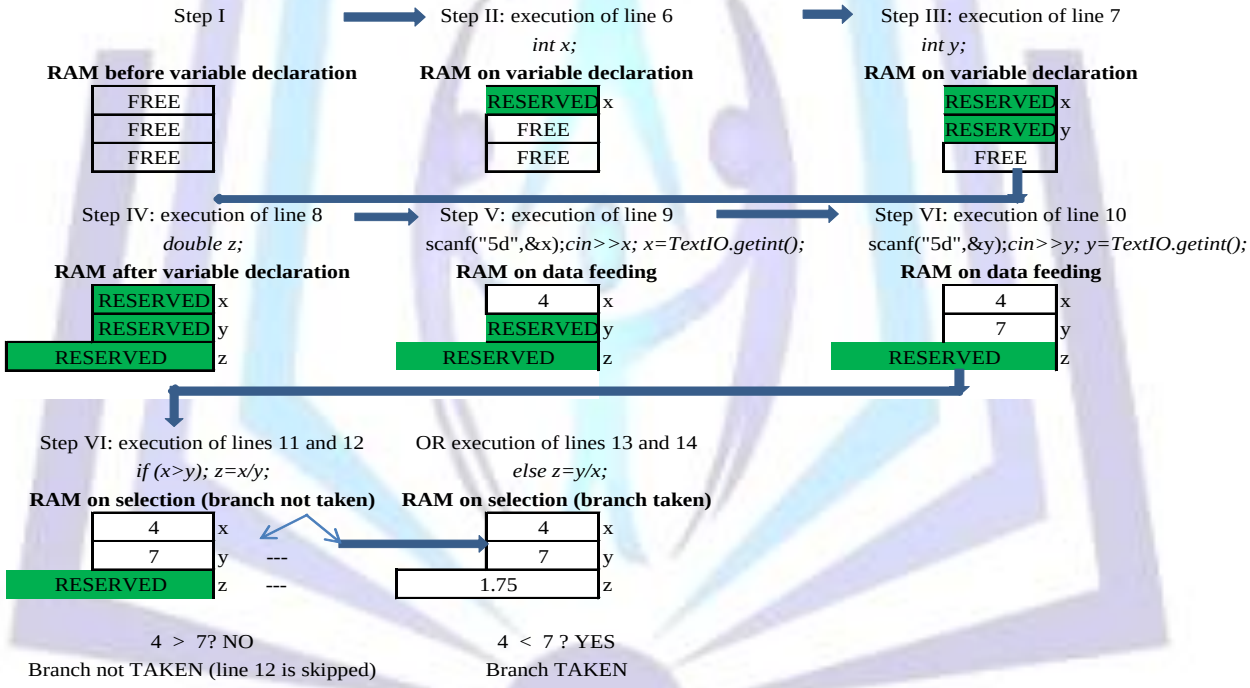

Figure 7: Single MTL interpretation of codes in parallel languages

As demonstrated in Figure 4 and 5 MTL is a learner-driven, sketch-based language used to visually interpret the code. MTL accepts the syntax of high level programming language as its inputs. The outputs are RAM status which reflect the visual semantics of the object code in a syntax that is understood by the programmer. MTL has been successfully integrated in current materials where all programming aspects covering function declaration, function calls, parameter passing, arrays, pointers and file handling can be visualized by MTL as demonstrated in works by Mselle [9-11]. MTL can be used to interpret codes written in multiple languages. This is useful in stressing that to the computer, the meaning of algorithm is one, no mater, which language has been used to write the code [9-13].

5. THE MTL GRAMMAR

The MTL grammar, GMTL, = (ΣM, NM, SM, PM), corresponding to the semantics;

$\alpha \Rightarrow G\beta$ iff $\exists\ u, v, p, q$ $(\Sigma M \cup NM)^*$: $\alpha = upv \wedge p \rightarrow q$ PM $\wedge\ \beta = uqv$, is a device to recognize correct use of MTL as a visual language based on high level languages, that enables a learner to visualize the code with absolute authority outside machine [13]. This grammar can be used by instructors, authors and learners to verify the correctness of using MTL to present programming materials as well as learning programming in any high level language.

6. CONCLUSIONS AND RECOMMENDATIONS

Unlike other visualization tools, MTL is a sketch-based language which is both machine and language independent. It can be integrated with current books, programming notes and laboratory examples as an effortless learner-driven visualizer which gives complete authority to the user.

It has been demonstrated [13] that MTL can be defined by a grammar using Chomsky's [2] definition. It can be pointed out that with the MTL grammar, the novice and programming teachers can verify the correctness of their interpretation of codes without the intervention of the machine or expert.

Like other visualization tools, MTL has yet to be infused in mainstream teaching of programming. Until this is achieved, it is not possible to fairly determine its effectiveness and impact in teaching programming. Further work to demonstrate MTL formalism by using graph grammars is recommended. Class experiments to measure the effectiveness of MTL in virtual learning environment, children schools and huge classes are recommended.

References

[1] Ben-Ari, M. (2001). Program visualization in theory and practice. Informatik/Informatique, 2, 8-11.

[2] Chomsky, N. (1957). Three models for the description of language. Department of Modern and Research Laboratory of Electronics. Massachusetts Institute of Technology, Cambridge, Massachusetts.

[3] Du Boulay, B. (1986). Some difficulties of learning to program. Journal of Educational Computing Research, 2(4), 459-472.

[4] Hundhausen, C. D., and Brown, J. L. (2007). What you see is what you code: A 'live' algorithm development and visualization environment for novice learners. Journal of Visual Languages and Computing, 18(1), 22-47.

[5] Hundhausen, C. D., Douglas, S. A. and Stasko, J. D. (2002). A meta-study of algorithm visualization effectiveness. Journal of Visual Languages and Computing, 13, 259-290.

[6] Kölling, M., Quig, B., Patterson, A., and Rosenberg, J. (2003). The BlueJ system and its pedagogy. Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, 13(4).

[7] Kuittinen, M., Tikansalo, T. and Sajaniemi, J. (2008). A Study of the development of students' visualizations of program state during an elementary Object-oriented Programming course. ACM Journal of Educational Resources in Computing, 7(4).

[8] Msele, L. (2010). Enhancing comprehension by using Random Access Memory (RAM) diagrams in teaching programming: class experiment. In *Proceedings of the 22nd Annual Psychology of Programming Interest Group* (Universidad Carlos III de Madrid, Leganés, Spain, September 19-22, 2010). Joseph Lawrence and Rachel Bellamy, editors.

[9] Mselle, L. *C++ for novice programmers*. LAP LAMBERT Academic Publishing, Berlin, 2010.

[10] Mselle, L. *Java for novice programmers*. LAP LAMBERT Academic Publishing, Berlin, 2011.

[11] Mselle, L. *C for novice programmers*. LAP LAMBERT Academic Publishing, Berlin, 2011.

[12] Mselle, L. (2011). The Impact of *Memory Transfer Language* (MTL) on reducing misconceptions in teaching programming. ITiCSE 2011, TU, Darmstadt, Germany, June, 27-29.

[13] Mselle, L (2011). Using Formal Logic to Define the Grammar for Memory Transfer Language (MTL) on the mould of Register Transfer Language (RTL) and High Level Languages**,** PPIG 2011 Symposium, The University of York, UK, September, 6-8, 2011.

[14] Moreno, A., Myller, N., Sutinen, E., and Ben-Ari, M. (2004) Visualizing programs with Jeliot 3. Proceedings of the Working Conference on Advanced Visual Interfaces, Gallipoli, Italy, 373-376.

[15] Naps, T., R¨oßling, G., Almstrum, V., Dann, W. Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S. and Vel´azquez-Iturbide, A. (2003). Exploring the role of visualization and engagement in computer science education. *ACM SIGCSE Bulletin*, 35(2), 131–152.

[16] Preparata, F. (1985) *Introduction to computer engineering*. Prentice Hall, New York.

[17] Rajala¸ T., Laakso, M., Kaila, E. and Salakoski, T. (2008). Effectiveness of program visualization: A case study with the ViLLe Tool. Journal of Information Technology Education: Innovations in Practice.

[18] R¨oßling G. and Ackermann, T. (2006). A Framework for generating AV content on-thefly.In Guido R¨oßling, editor, Proceedings of the Fourth Program Visualization Workshop,Florence, Italy, 106–111.

[19] R¨oßling, G. and Vellaramkalayil, T. (2008). First Steps Towards a visualization-based computer science hypertextbook as a Moodle module. CS Department, TU Darmstadt, Hochschulstr, 10 64289 Darmstadt, Germany.