# Enriching Programming Instruction using Visualization

Marianne P. Ang
Computer Science Department
Ateneo de Naga University
Ateneo Avenue, 4400 Naga City
angmarianne@yahoo.com

Allan A. Sioson
Computer Science Department
Ateneo de Naga University
Ateneo Avenue, 4400 Naga City
allan@adnu.edu.ph

## ABSTRACT

Visualization is a powerful facility to teach abstract concepts in programming. A description of the design of iC++, a memory diagram-based visualization tool, and a discussion of the results of using it to help teach C++ programming are presented in this paper.

## Keywords

Visualization, Programming, Computer Science Education

## 1. INTRODUCTION

Computer programming is a difficult skill to teach. Limited training of students on problem solving and logical reasoning skill are often used as reasons why students find little success in learning how to program a computer. Soloway [12] observes that novice programmers find it difficult to "put the pieces together." A novice programmer builds his skills in algorithm design while he learns how to translate the steps of the algorithm to equivalent statements in a target programming language.

Inadequate understanding of program state as each statement of a program is executed contributes to confusion of a novice programmer. To help reinforce understanding of how program works, visualization techniques may be used. In this research work, we used memory diagrams as a visual representation to help novice programmers see memory variables as they are created and manipulated in the computer's main memory as each program statement is executed.

## 2. BACKGROUND OF STUDY

The process of learning how to program typically involves three activities: (a) Learning the features and syntax of the programming language; (b) Algorithm design and implementation; and (c) Algorithm comprehension. Learning difficulties often emanate from at least one of these activities.

7th National Conference on I.T. Education (NCITE 2009)
Capitol University, Cagayan de Oro City, 21-23 October 2009

**Features and syntax of programming language.**
Soloway and Spohrer [13] and Pane and Myers [9] reported common deficits in novices' understanding of specific programming language constructs. In their respective results, they noted that variable initialization seems to be more difficult to understand than updating or testing variables. They also observed that bugs, especially in repetition (loop contructs) and selection (branching constructs), are common. Understanding actions that take place "behind the scenes," such as updating loop variables in "for" loops, are difficult to comprehend for some students. Furthermore, expressions that are syntactically close to each other or expressions that mean different things in different contexts often cause confusion (For example, `"123"` and `123`).

**Algorithm design and implementation.**
In a different perspective, Davies [4] observed that the main source of difficulty is not programming language syntax nor understanding of concepts, but rather basic program planning. Holliday and Luginbuhi [6] emphasized that it is important to distinguish between programming knowledge and programming strategies. A student may understand how a programming construct works (e.g., an array construct or a data structure) but still fails to use it appropriately and correctly in a program.

**Algorithm comprehension.**
Another aspect of programming that novice programmers find difficult is understanding all the issues related to the execution of a program. Du Boulay [2] stated that "... it takes quite a long time to learn the relation between a program on the page and the mechanism it describes." He observed that students have difficulties in understanding that each instruction is executed in the state that has been created by the previous instructions. He stated that there should be a simple "notional machine," that simplifies the language and the machine so that all the program transformations can be visible.

**Visualization as an effective technique.**
Soloway [12] notes that the use of visualization is recognized as an effective teaching strategy to help novice programmers understand basic programming. In Ateneo de Naga, Sernande [11] developed ProgViz, a web application that allows visualization of primitive C++ data objects. While Progviz only supports primitive data types, simple single linked list, and simple single dimensional arrays, students can use it to visualize the *end state* of each data objects af-

ter a sequence of statements are executed. In this paper, we describe iC++ [1], a windows application that allows visualization and animation of program states of an input C++ code. We also report preliminary results in using iC++ in teaching fundamental programming constructs.

## 3. DESIGN OF IC++

iC++ is developed using the prototyping method typically used in database applications development [5]. The application heavily used memory diagrams to illustrate how memory variables are created and manipulated in the computer's main memory as each C++ statement is executed. A memory diagram represents the state of data objects in a computer's memory at a specific point in the execution of a program [6]. Figure 1 provides an example of how memory diagrams are drawn.

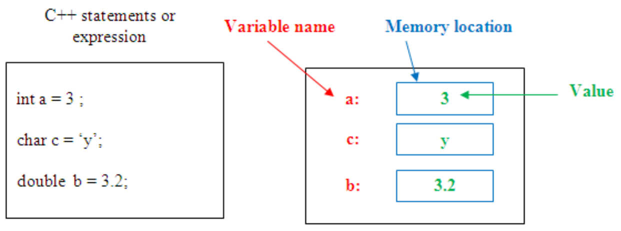


**Figure 1: C++ Code and Equivalent Memory Diagram**

iC++ supports the usual operators: (a) unary operators (unary-plus, unary-minus, and not) (b) binary operators (addition, subtraction, multiplication, division, modulo, and relational operators). Furthermore, iC++ supports the following statement types: data declaration, assignment, decrement and increment of integer variables, selection (**`if`**, **`if-else`**, **`switch-case`**), and repetition (**`while`**, **`do-while`**, **`for`**). The primitive data types that iC++ supports are **`int`**, **`char`**, **`double`**, and **`float`**. iC++ also supports **`string`**, single and two dimensional arrays, and simple singly linked list of integers. Other constructs such as C++ function and user defined classes are not supported.

The design of the iC++ system is self-explanatory and illustrated in Figure 2. Each inputted code fragment is translated to a complete C++ driver program (`driver.cpp` in Figure 2). The driver program is compiled using minimalist GNU C++ compiler for windows [8] included in the wxDevCpp package [7, 14]. The translator's lexical analyzer and parser are developed using Flex [10] and Bison [3] respectively. The graphical user interface (GUI) of iC++, developed using wxWidgets C++ Library [15], consists of two windows:

1. **The main window.** The iC++ main window consists of two menus (see Figure 3), two tabs and the compile button. The two menus are File and About. The `File` menu consists of three submenus namely: `Open`, `Save as`, and `Exit`. The `About` menu have one submenu which is the `About  IC++` that contains the basic information of the software. There are two tabs in the main window namely: *Code Fragment* tab and *Error Message* tab. The Code fragment tab houses the edit area where users can type the input C++ program. The Error message tab contains the display area where the error messages are shown whenever errors in the input C++ program is detected. The compile button start the cascade of processes that create create the data files needed in program state visualization and animation.

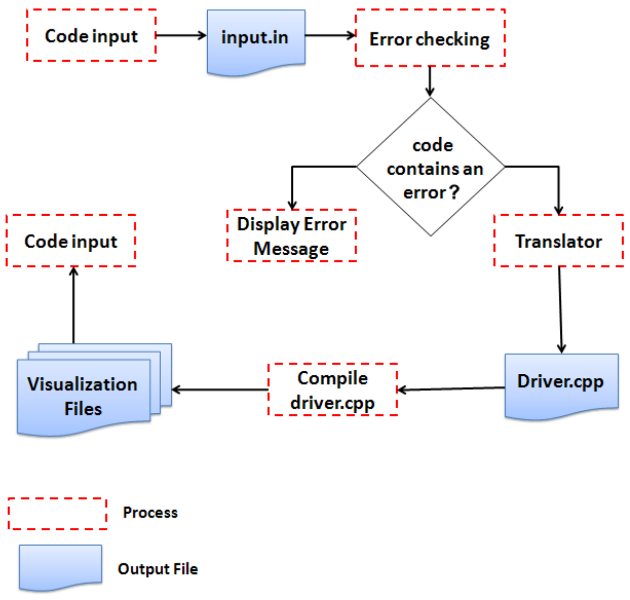
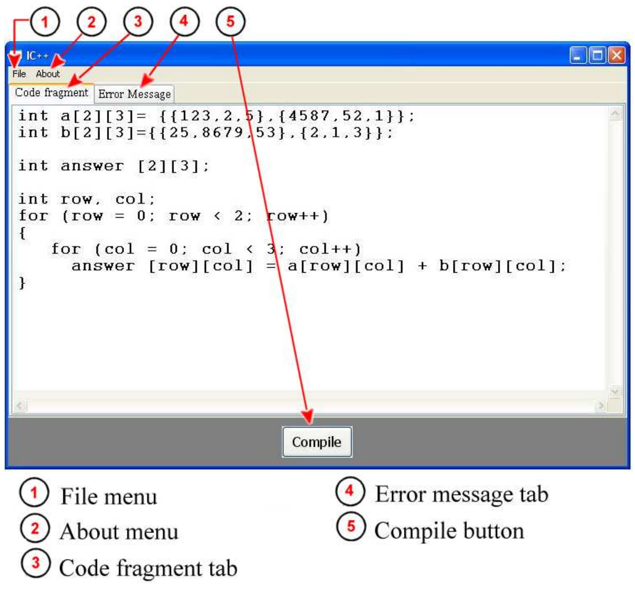


**Figure 2: System Design of IC++**



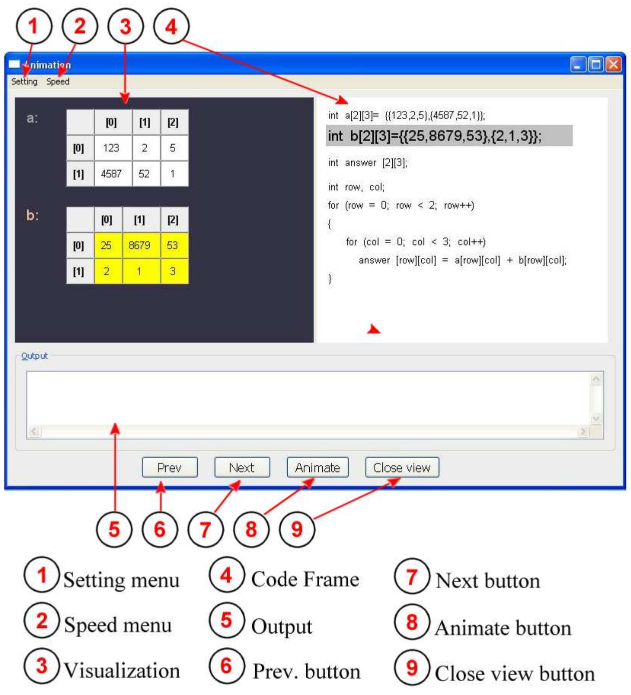
```
int a[2][3]= {{123,2,5},{4587,52,1}};
int b[2][3]={{25,8679,53},{2,1,3}};

int answer [2][3];

int row, col;
for (row = 0; row < 2; row++)
{
   for (col = 0; col < 3; col++)
     answer [row][col] = a[row][col] + b[row][col];
}
```

**Figure 3: iC++ Main Window**

2. **The animation window.** The iC++ animation window (see Figure 4) consists of the *Object visualization frame*, *Code fragment frame*, *Output*, and four control buttons namely: **`Prev`**, **`Next`**, **`Animate`**, and



a:

| | [0] | [1] | [2] |
|---|---|---|---|
| [0] | 123 | 2 | 5 |
| [1] | 4587 | 52 | 1 |

b:

| | [0] | [1] | [2] |
|---|---|---|---|
| [0] | 25 | 8679 | 53 |
| [1] | 2 | 1 | 3 |

```
int a[2][3]= {{123,2,5},{4587,52,1}};
int b[2][3]={{25,8679,53},{2,1,3}};
int answer [2][3];
int row, col;
for (row = 0; row < 2; row++)
{
    for (col = 0; col < 3; col++)
        answer [row][col] = a[row][col] + b[row][col];
}
```

**Figure 4: iC++ Animation Window**

`Close View`. This window has two menus: (a) The `Setting` menu has the `Reset` option, used to reset again the animation. (b) The `Speed` menu has three options: *Slow*, *Moderate*, and *Fast*. These options control the speed of the animation once the `Animate` button is clicked. Object visualization frame is where the visualization of the memory variables are displayed. Whenever the value of the variable is updated, the variable name and the box shown in object visualization frame changes its color to provide a visual cue to direct the user to see the effect of a statement execution. The code fragment frame is where the code typed by the user is displayed. The highlighted statement in the code fragment frame shows the statement currently being executed. The output section in the animation window displays the output of the given program. The `Prev` button is used to display the previous diagram of program state. The `Next` button is used to display the next diagram. The `Animate` button is used to display the visualization automatically until the last statement. The `Close` view button is used to close the Animation window.

## 4. EVALUATION

The iC++ program was used to review key concepts in introductory programming subject in C++. Selected students from the ZC11 section of ICST102 (Computer Programming 1) class participated in the activity This activity was conducted during the first semester of School Year 2008-2009. A total of 32 students participated. During the evaluation activity, students were asked to answer a prepared questionnaire meant to check how they understood specific program constructs.

The questionnaire includes questions about variable declaration, assignment statements, calculations using various operators, and the three basic program structures namely: sequence, selection, and repetition.

Seven aspects of programming were assessed during the activity:

1. **Default value of uninitialized variable.** The default value of uninitialized integer variable is oftentimes assumed to be zero. Compilers however may store a different default value to an uninitialized variable and that default value is not necessarily zero. We are interested to know if students have this misconception.
2. **Sequence.** As reported by Du Boulay [2], some students have difficulties in understanding sequential execution of program statements. We are interested to know if students understand the effect of sequential execution of program statements to the program's state.
3. **Decision.** Here, we are interested to know if students know how to evaluate boolean expressions.
4. **Branching.** Here, we are interested to know if students understand the operational meaning of branching structures.
5. **Assignment.** Here, we are interested to know if students understand the operational meaning assignment statements.
6. **Operations.** Here, we are interested to know if students mastered the use of arithmetic and relational operations.
7. **Expected Results.** Here, we are interested to know if students comprehend the computations being done by the input program and can deduce the program output correctly.

Three input programs were used in the evaluation activity. The main purpose of the evaluation is to compare the resulting program state (in terms of memory diagram) expected by a student with the correct program state.

To see the resulting program state after executing a specific program statement, the user must click the `Next` button in the iC++ animation window. In the designed evaluation activity, each student is asked to answer first certain questions provided in the questionnaire before clicking the `Next` button. After the student sees the next program state, the student is asked to declare whether the resulting program state is the same as he or she expected.

Table 1 shows the results of the activity. Columns 2 to 4 shows the percentage of students who declared they answered correctly the questions related to the each of 7 aspects being assessed. We assumed that all the participants answered the questionnaire as truthful as possible.

The activity revealed that all the students simply assumed that the default value of uninitialized integer variables is 0.

| Category | Program 1 | Program 2 | Program 3 |
|---|---|---|---|
| Default var. value | 0 % | 0 % | 0 % |
| Sequence | - | 62.50 % | - |
| Decision | - | 84.38 % | 75.00 % |
| Branching | - | 68.75 % | 50.00 % |
| Assignment | 54.69 % | - | - |
| Operations | 93.75 % | - | - |
| Expected Result | 0 % | 40.63 % | 26.56 % |

**Table 1: Evaluation Results**

This misconception often leads to problems in C++ programs. Also, while most students can evaluate boolean expressions used in decision making (84.38% in program 2, 75% in program 3), fewer knew where the branching would go (68.75% in program 2, 50% in program 3). Furthermore, while most students know how operators work (93.75% in program 1), fewer knew how assignment works (54.69%). Most students failed to comprehend what is being computed in each of the input programs (0% in program 1, 40.63% in program 2, and 26.56% in program 3).

The necessary interventions were conducted after the results of the activity were revealed. Certain concepts were emphasized in the classroom during the lecture sessions. Some students who participated in the activity reported some difficulty in using iC++. This indicates that a usability study is needed.

## 5. CONCLUDING REMARKS

iC++ is a windows application for visualizing the state of memory variables in the computer's main memory as each C++ statement in a program fragment executes. It is meant to help novice programmers in understanding variable declarations, assignment statements and the three basic program structures namely: sequence, selection, and repetition.

To improve the usability of the iC++, a usability study is needed to identify the features that needs improvement. While the current version of iC++ has limited functionality, it can already be used in teaching early parts of an introductory course in C++ programming.

## 6. ACKNOWLEDGEMENTS

We thank all the students who participated in the software evaluation. They are members of the ICST102 (Computer Programming 1) class of first semester of School Year 2008-2009 at Ateneo de Naga University.

## 7. REFERENCES

[1] M. Ang. iC++. Senior project supervised by Dr. Allan A. Sioson, Department of Computer Science, College of Computer Studies, Ateneo de Naga University, March 2009.

[2] D. Boulay. Some difficulties of learning to program. In *Studying the Novice programmer (Soloway and Spohrer, Eds)*, pages 283–300. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1989.

[3] R. Corbett, R. Stallman, and W. Hansen. Bison - GNU parser generator. `http://www.gnu.org/software/bison/`.

[4] S. Davies. Models and theories of programming strategy. *International Journal of Man-Machine Studies*, 39:237–267, 1993.

[5] J. A. Hoffer, F. R. McFadden, and M. B. Prescott. Alternative IS Development Approaches. In *Modern Database Management*, pages 44–45. Prentice-Hall, Inc., 2002.

[6] M. A. Holliday and D. Luginbuh. CS1 Assessment Using Memory Diagrams. *ACM SIGCSE Bulletin*, 36(1):200–204, March 2004.

[7] C. Laplace, H. Lai, Y. Mandravellos, and M. Berg. The Dev-C++ Resource Site. `http://bloodshed.net/dev/`.

[8] MinGW Developers. MinGW – Minimalist GNU for Windows. `http://www.mingw.org/`.

[9] J. Pane and B. Myers. Usability Issues in the Design of Novice Programming Systems. Technical Report CMU-CS-96-132, School of Computer Science, Carnegie Mellon university, 1996.

[10] V. Paxson. flex: The Fast Lexical Analyzer. `http://flex.sourceforge.net/`.

[11] E. Sernande. Effects of Visualization of Data Objects for Novice Programmers. Senior project supervised by Dr. Allan A. Sioson, Department of Computer Science, College of Computer Studies, Ateneo de Naga University, March 2008.

[12] E. Soloway. Learning To Program = Learning To Construct Mechanisms And Explanations. *Communications of the ACM*, 29(9):850–858, September 1986.

[13] E. Soloway and J. Spohrer. *Studying the Novice Programmer*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1989.

[14] wxDev-C++ Developers. wxDev-C++ – a wxWidgets extended version of Dev-C++. `http://wxdsgn.sourceforge.net/`.

[15] wxWidgets Developers. wxWidgets – Cross-Platform GUI Library. `http://www.wxwidgets.org/`.