Iconic vs. Text-Based Programming in the Introductory Programming Sequence†

Donald J. Bagert, Ben A. Calloni, H. Paul Haiduk
Texas Tech University

Abstract

Research has been undertaken to evaluate the effectiveness of using iconic (as opposed to visual) programming environments in teaching the first two computer programming courses. The authors have developed a Windows-based iconic programming language named BACCII, which allows the user to program with icons representing all the major programming constructs and data structures within a syntax-directed environment. The user can then generate syntactically correct code for any one of several text-based languages such as C++ and Pascal. More recently, work on adding object-oriented extensions to BACCII for use in the data structures/object-oriented programming (CS2) course was undertaken, resulting in BACCII++.

Recent research involving BACCII had included the development of a complete set of course materials for the use of BACCII++ in teaching both CS1 and CS2 using C++. Laboratory courseware, tutorials and other materials were developed. An experiment, addressing the question "Can icon-based programming languages be used to teach first-year programming concepts to undergraduate students more effectively than text-based languages?", is being run using the new teaching materials at Texas Tech during the 1995-96 school year. Future research hopes to extend this program to series of pilot programs at other institutions.

1. Introduction

Research has been undertaken to answer the following question: Can icon-based programming languages be used to teach first year programming concepts to undergraduate students more effectively than text-based languages?

Many noted researchers (e.g. Glinert[5] and Scanlan[6]) have empirically established the cognitive advantage which graphical methodologies provide over textual ones. Research undertaken by the co-authors resulted in development a Windows-based iconic programming language named BACCII‡. This environment allows the user to program with icons representing all the major programming constructs, such as loops, conditional branching, within a syntax-directed environment[2,3]. The user can then generate syntactically correct code for any one of five text-based languages.

---

† This work was partially supported by the National Science Foundation Division of Undergraduate Education under Grant DUE-9455614 to Texas Tech University.

‡ BACCII (pronounced ba-chee) is © and ™ 1992-1996 Ben A. Calloni. BACCII stands for **B**en **A**. **C**alloni **C**ode for **I**nformation **I**nterchange.

This system has previously been used as a supplement to teach a Pascal-based introductory computer science (CS1) course required of both computer science and electrical engineering majors; despite having limited teaching materials for BACCII at that time, the empirical results showed a 4-8% increase in learning and comprehension[4]. More recently, work on adding object-oriented extensions to BACCII for use in the data structures/object-oriented programming (CS2) course was undertaken, resulting in BACCII++[§].

Current research concerns the development of a complete set of course materials for the use of BACCII++ in teaching both CS1 and CS2 using C++. Lecture notes, text, and lab manuals are under ongoing development, and are being tested during the 1995-96 school year. Preliminary results from the fall 1995 semester are once again encouraging. This research has widespread potential significance: if it can be demonstrated that iconic languages perform significantly better than text-based languages in the learning of programming skills for a wide variety of schools and student groups, it could revolutionize how software development skills are taught in every computer-related program in higher education.

Section 2 of this paper provides some background on the procedure-oriented BACCII environment. Section 3 describes a previous study of the effectiveness of using BACCII in the laboratory to learn procedural programming skills. The enhancement of BACCII to include object-oriented constructs, which evolved into the language BACCII++, is discussed in Section 4. In Section 5, the courseware currently under development for using BACCII++ in C++-based CS1 and CS2 courses, along with a current experiment being conducted that uses these materials, is discussed. Finally, Section 6 provides a summary and outlines some future directions for the use of BACCII++.

2. Background

The current use of C++, Ada, or any other programming language whose syntax is text-based has the major drawback that regardless of the language chosen, beginners quickly lose sight of the problem solving aspects of using an algorithm once the "details" of the syntax begin to surface. It seems that the question should not be which language to use for a first programming course, but rather the question should be whether students can be educated in a learning environment which keeps the student focused on the problem solving aspects of the algorithm development, frees that student from the details of the syntax of any language, incorporates some software engineering concepts, and provides a gentle nudge in the area of theoretical computer topics.

It is the investigators' opinion that the most effective methodology for learning programming skills lies within the bounds of graphical representations of algorithms. Much work has been accomplished in iconic programming but almost all of it has been developed for high level workstations. Two factors contribute to the selectivity: speed and memory requirements. The rapid progress in high speed, high resolution, large RAM, low cost computers in the last three or four years has made possible more visually based programming environments to a wider range of users.

Such a programming environment can use a syntax-directed approach, which would guarantee syntactically correct code before the compiler even sees it. Also, by providing a graphical representation for the flow of control, one could increase semantic accuracy by accurately conveying to the user which variables of the proper

§ BACCII++ is © and ™ 1996 Ben A. Calloni.

type could be used at a particular point in the algorithm. The BACCII programming environment was developed with these factors in mind.

BACCII provides a standard access to code regardless of the subroutine. Click in the body section of the bit map display and the system moves to a new screen (Figure 1).

In order to insure syntactic correctness, BACCII allows the student to input new statements only if they would be correct for the program's parse tree. The student selects a statement from the "paint menu" and clicks on the <STMT> node in the display. Any selection other than a statement non-terminal is not accepted. BACCII automatically calculates screen locations for each new selection and scrolls the screen display down.

Single statements, such as read, write, assignment, and so forth, simply replace the <STMT> icon. The selection and iteration statements always are created in groupings. For instance, one cannot create just the "if" part of a selection. The student always gets both <STMT> options for true-false. The same process is applied to all "multiple statement" icons.

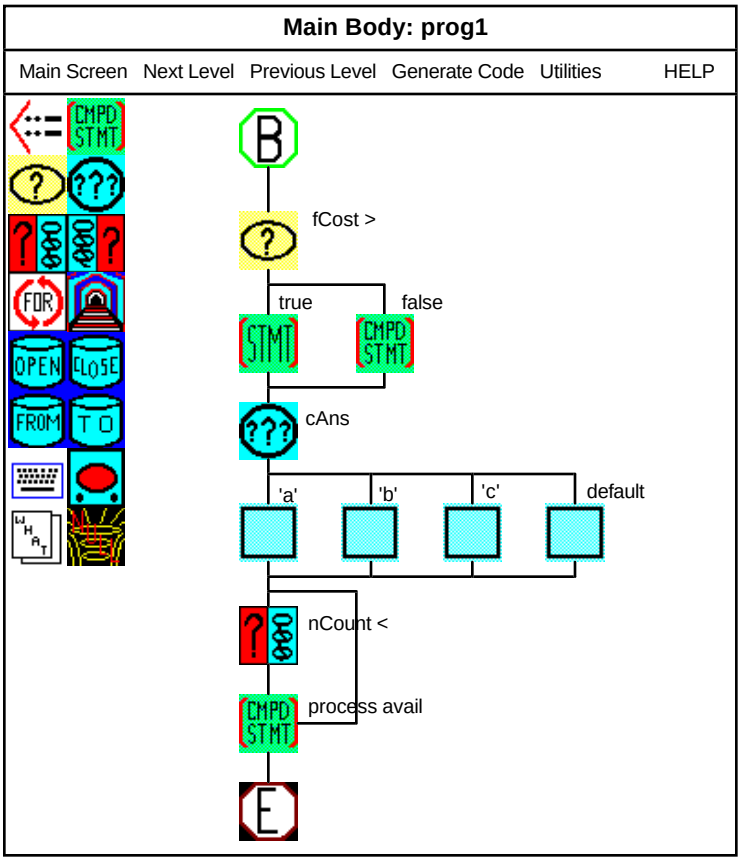

Figure 1. BACCII Coding Screen.

3. Previous Study

In the spring of 1993, BACCII was used in the introductory programming ("CS1") course at Texas Tech University, Computer Science 1462 (Fundamentals of Computer Science I). This course has its roots in the updated ACM Curriculum 78 CS1 course, but is actually closer to the course CD 101 described on pages 103-4

of Computing Curricula 1991[7]. C S 1462 is a four-semester hour course with a "partially closed" laboratory which currently uses BACCII, and has recently switched to C++ as the programming language taught.

Both computer science and electrical engineering students are required to take this course. The experiment was designed to divide the students into two groups: one which would use only Pascal (C++ became the CS1 language in 1994) and the other which used both Pascal and BACCII for development. The BACCII students were required to use BACCII for main programming assignments and submit BACCII files for evaluation, in addition to submitting correct Pascal code for grading. This step was necessary to insure that all Pascal students would know Pascal syntax upon completion of the course. (BACCII is robust enough that it could be used for program development without the need to learn Pascal syntax.) Use of BACCII for weekly laboratory assignments was optional. There were almost no supplemental teaching materials for BACCII, although a tutorial was completed about two-thirds of the way into the semester.

There were several areas which were evaluated. H0, the null hypothesis, is used to indicate that no difference exists between the population means. H1 is used to indicate that the BACCII group has a higher mean than the Pascal-only group.

1. H0: The BACCII environment will result in no difference in CS programming assignments. H0:$\mu 1=\mu 2$.
2. H1: The use of BACCII will result in higher scores on programming assignments. H1: $\mu 1 > \mu 2$
3. H1: The use of BACCII will result in EE students having higher programming grades than CS majors. H1: $\mu 1 > \mu 2$
4. H1: The use of BACCII will result in EE students having higher programming scores than non-BACCII engineers. H1: $\mu 1 > \mu 2$
5. H1: The use of BACCII will result in higher scores on Lab Assignments. H1: $\mu 1 > \mu 2$
6. H0: The use of BACCII will result in equivalent scores on (Pascal-only) exam scores. H0: $\mu 1 = \mu 2$
7. H0: The use of BACCII will result in equivalent course scores. H0: $\mu 1 = \mu 2$

To summarize the results: for all students, BACCII resulted in higher scores on the programming assignments, labs, exams and overall course grade; the EE majors using BACCII did the same as the CS majors using BACCII; however, there was no significant difference in how EE students performed using BACCII vs. just Pascal.

For more information concerning this experiment, please refer to Calloni and Bagert[4].

4. Enhancements for a One-Year Curriculum

Computer Science 2463 (Fundamentals of Computer Science II), is the second programming course; its prerequisite is C S 1462. The evolution of C S 2463 has paralleled that of 1462; it originally resembled the updated ACM Curriculum 78 CS2 data structures course, but is actually closer to the course CD 102 described on pages 103-4 of Computing Curricula 1991[7]. The course is taught using object-oriented programming from the very beginning. C S 2463 is a four-semester hour course with a completely open laboratory; it has also recently switched to C++ as the programming language taught.

This course has not previously used BACCII, since it did not contain the necessary object-oriented extensions. However the "visual" nature of the object-oriented paradigm (since most people view the world in terms of objects interacting with each other) makes it ideal for implementation using an iconic language. Therefore, work was undertaken to extend BACCII so that it can be used in the CS2 course, resulting in the

creation of BACCII++.

First, a new class must be created; the resulting class icon (for a stack class) is shown in Figure 2. Attributes and methods can then be declared in a similar manner to how variables and subroutines, respectively, were declared in BACCII, except for the fact that each attribute and method can be declared as public, private, or protected.

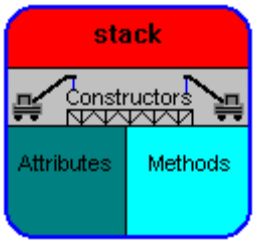

Figure 2. A class icon in BACCII++.

After the class is completely defined, the C++ class header and class implementation files can be generated and used.

5. Course Materials for the First Year

As stated in Section 3, the initial experiment was performed with almost no supplemental teaching materials for BACCII. In the summer of 1995, work began to develop course materials for the entire first year sequence, under a National Science Foundation Division of Undergraduate Education grant. The C S 1462 and 2463 materials were being implemented starting in the Fall 1995 and Spring 1996 semesters, respectively. Each course has closed laboratories throughout the semester, with each lab section having no more than 25 students. The courses are evaluated in a manner similar to one previously used (described in Section 3); i.e. half of the students would use BACCII++, while the other half would be a control group using only C++. (C++ was chosen as the programming language since several universities, including Texas Tech, are now using C++ as the introductory language, and because the AP Computer Science exam will be switching to C++ in 1999.)

The course materials that are being developed and disseminated as a result of this grant include a supplementary textbook for using BACCII++ with C++; a set of closed laboratories and other tutorial material for both courses, using both standard and multimedia delivery systems; and sample on-line and off-line examination problems. Many of the course materials have been developed using Asymetrix ToolBook[1]. To reduce experimental bias, similar lab courseware was developed both for using C++ with BACCII++ and for using C++ without BACCII++. Examples of courseware screens for the CS1 and CS2 courses can be found in Figure 3 and 4, respectively.

In using the courseware in the fall of 1995 in the CS1 course, there were again several areas which were evaluated in a similar manner to the experiment described in Section 3.

1. H_0: The use of BACCII++ will result in no difference in lab assignments. H_0: $\mu_1 = \mu_2$
2. H_1: The use of BACCII++ will result in higher scores on lab assignments. H_1: $\mu_1 > \mu_2$
3. H_0: The use of BACCII++ will result in no difference in programming assignments. H_0: $\mu_1 = \mu_2$
4. H_1: The use of BACCII++ will result in higher scores on programming assignments. H_1: $\mu_1 > \mu_2$
5. H_0: The use of BACCII++ will result in no difference in scores on (C++ only) exams. H_0: $\mu_1 = \mu_2$

6. H_1: The use of BACCII++ will result in higher scores on (C++ only) exams. H_1: $\mu_1 > \mu_2$
7. H_0: The use of BACCII++ will result in equivalent overall course average. H_0: $\mu_1 = \mu_2$
8. H_1: The use of BACCII++ will result in higher overall course average. H_1: $\mu_1 > \mu_2$

BACCII++ resulted in higher scores on the labs, with a 99.5% confidence interval; and higher exam average and overall course average, with a 95% confidence interval in these cases. However, for the programming assignments, no significant difference between the BACCII++ and the C++ only groups were found; this is possibly because different (but similar) programming assignments were used, while in the case of the laboratory assignments, very similar assignments were used and for the exams, identical (C++ only) exams were administered. (For the spring of 1996, identical programming assignments are being used.)

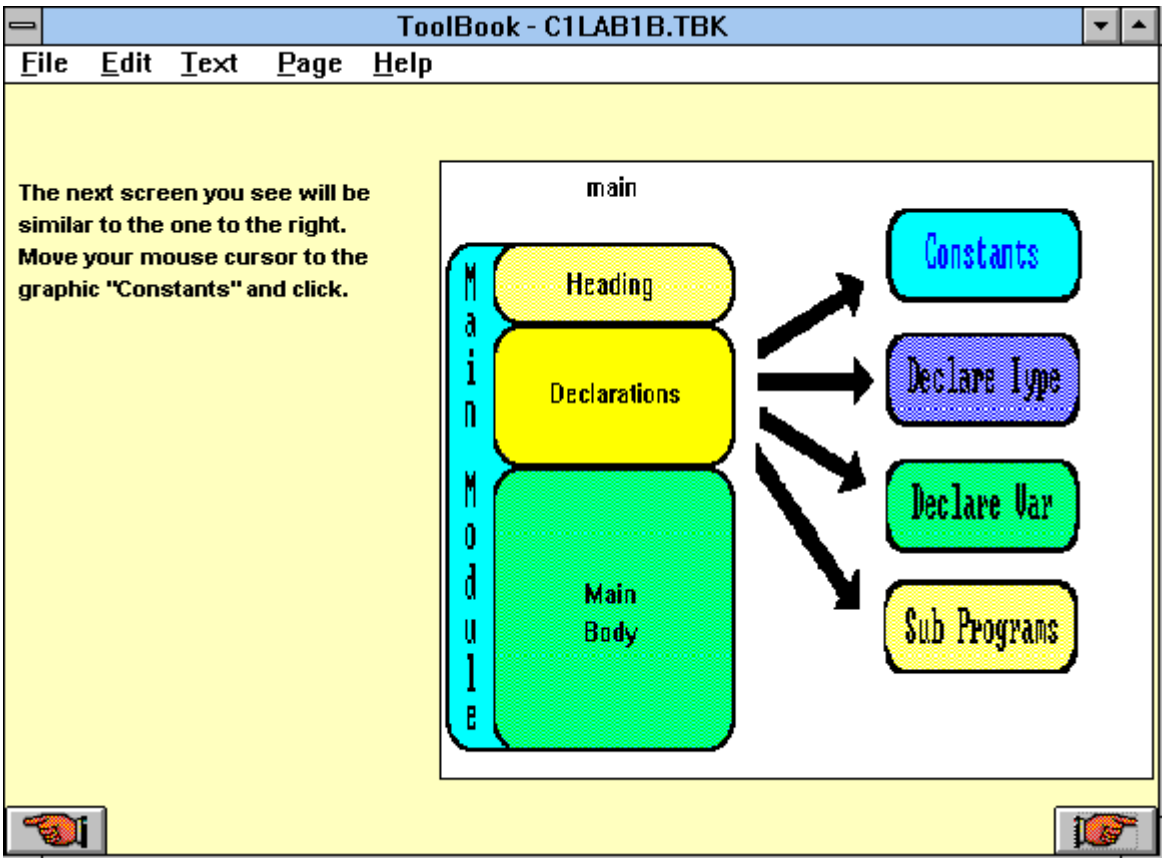

Figure 3. Example BACCII++ Courseware Screen for CS1.

6. Summary and Future Directions

The BACCII iconic programming environment can be used to build programs using icons instead of the traditional text-based statements. Programs written in BACCII can be translated into several different programming languages, including C++. BACCII has been used successfully in the introductory course at Texas Tech. Subsequent research was undertaken so that BACCII could also be used with the object-oriented paradigm, resulting in BACCII++. Finally, a comprehensive set of teaching materials for both courses is being developed under an NSF grant. An experiment is being run using these course materials during the 1995-96 school year; results from the fall 1995 semester are once again encouraging.

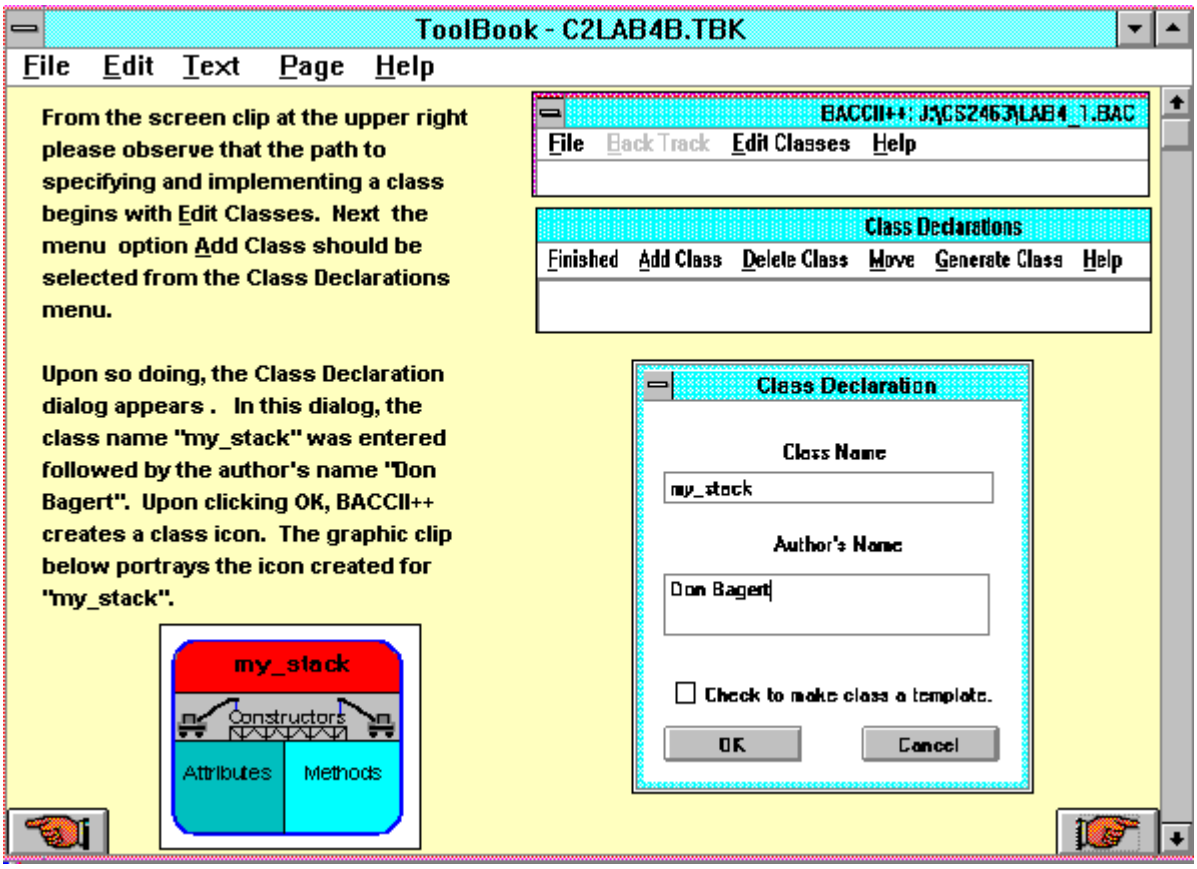

Figure 4. Example BACCII++ Courseware Screen for CS2.

Future work is being proposed to run experiments using the BACCII++ materials at five pilot schools, and to conduct additional workshops on using BACCII++ in the first year sequence. The authors also believe that BACCII++ can be beneficial in learning programming skills at the primary and secondary levels; in fact, it may be even more beneficial to those students.

References

[1] Asymetrix Corporation. *ToolBook User Manual.* Asymetrix Corporation, Bellevue WA, 1994.

[2] Calloni, Ben A. *An Iconic, Syntax Directed Windows Environment for Teaching Procedural Programming*. Master's Thesis, Department of Computer Science, Texas Tech University, May 1992.

[3] Calloni, Ben A. and Bagert, Donald J.. BACCII: An iconic syntax-directed system for teaching procedural programming, *Proceedings of the 31st ACM Southeast Conference*, Birmingham AL, April 15-16, 1993, pp. 177-183.

[4] Calloni, Ben A. and Bagert, Donald J.. Iconic programming in BACCII vs. textual programming: which is a better learning environment*? Proceedings of the 25th SIGCSE Technical Symposium on Computer Science Education*, Phoenix AZ, 10-11 March 1994, pp. 188-192.

[5] Glinert E. and Tanimoto S. Pict: An interactive graphical programming environment. *IEEE Computer*, 17, 11 (November 1984), pp. 7-25.

[6] Scanlan, David. Structured flowcharts outperform pseudocode: an experimental comparison, *IEEE Software*, Vol 6, No 5, Sept. 1989, pp. 28-36.

[7] Tucker, Allen B; Barnes, Bruce H et. al. *Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force*. Jointly published by ACM Press, New York NY and IEEE Computer Society Press, Los Alamitos CA, 17 December 1990.

Donald J. Bagert

Dr. Bagert is an Associate Professor of Computer Science at Texas Tech University. He has over thirty publications in the area of computer science education, and has also published numerous papers in the areas of object-oriented systems development and software engineering.

Ben A. Calloni

Mr. Calloni is a doctoral student and faculty lecturer in Computer Science at Texas Tech University. He developed the BACCII iconic environment as part of his Master's Thesis, under the direction of Dr. Bagert. His work on BACCII earned him first place in the graduate division of the *Second Annual ACM Student Poster Competition* held at the ACM Computer Science Conference in Phoenix on 8-9 March 1994.

H. Paul Haiduk

Mr. Haiduk is a doctoral student at Texas Tech University, and a Professor of Computer Science and Information Systems at Amarillo College. He is also currently the President-elect for the Consortium for Computing in Small Colleges.