A system to help teaching and learning algorithms

Leônidas de Oliveira Brandão
Anarosa A. F. Brandão
Romenig da Silva Ribeiro

{leo, romenig}@ime.usp.br, anarosa.brandao@poli.usp.br

University of São Paulo – USP
São Paulo, Brazil

Agenda

- The Context
- The iVProg
- Experiments Using iVProg
- Results
- Conclusion and Future Work

The Context

- Teaching and learning algorithms have big challenges for teachers and students [1] [2];
- Moreover, students are presented to new learning environments in order to edit, debug and compile their algorithms written in some tradicional programming language;
- It's an overload of new information;
- To overcome this problem some attempts of using visual sytems to support the learning of introductory programming were proposed such as Alice[3] and Scratch[4];
- Finally, in 2009, we decided to invest in this same approach, then iVProg was created;

The iVProg

- Based on Alice, but with less than 10% of its initial source code volume. Moreover, it was adapted to teach procedural programming approach and the Alice's resources to create animations were removed;

- Works as a Desktop or Applet application. It's also possible to integrate it on Moodle through iAssign [ref] plugin;

- Such as Alice and Scratch, iVProg has visual blocks that represents structures such as *if-then-else* or *loops*;

Figure 1. iVProg toolbar.

Figure 2. iVProg running a program.

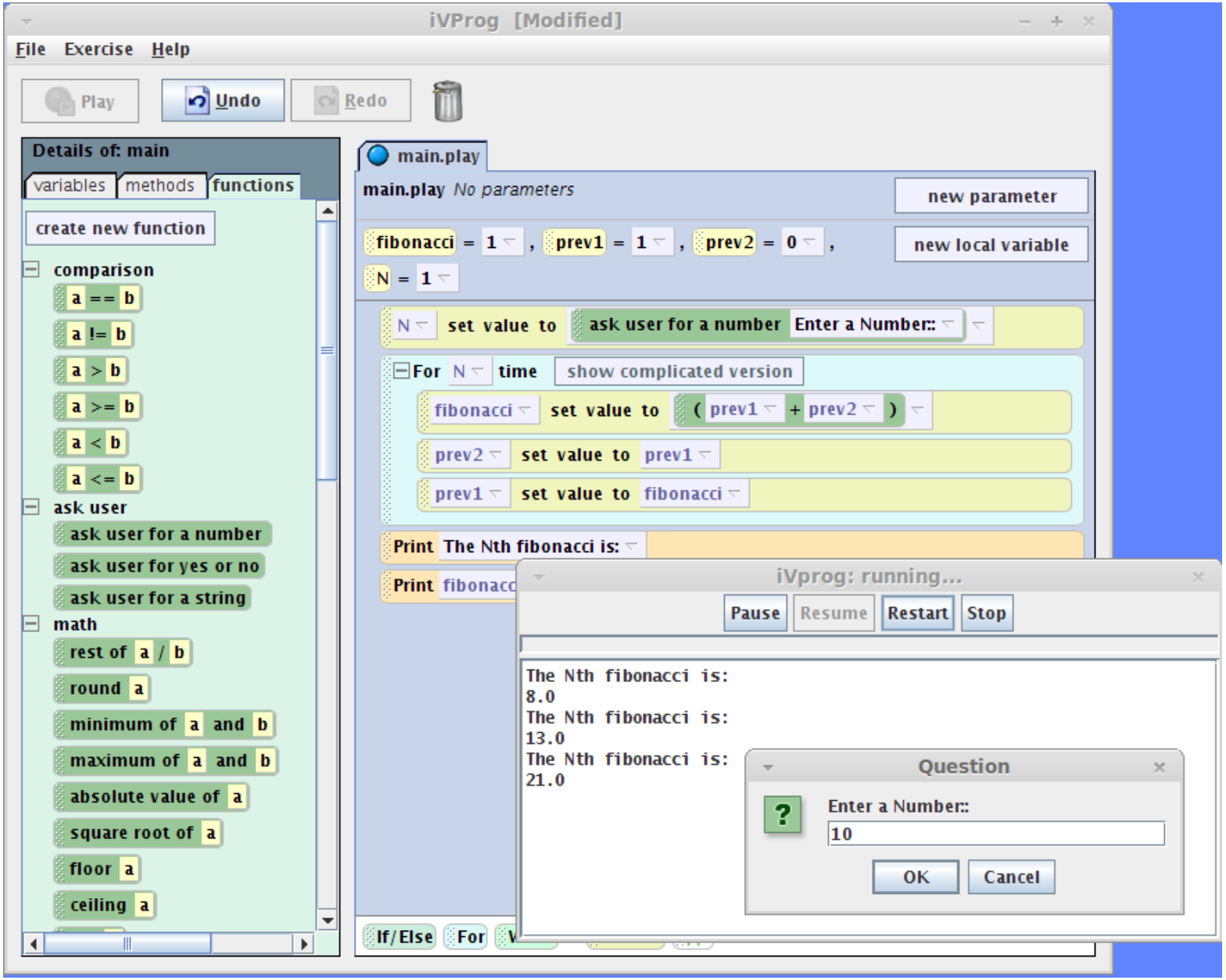
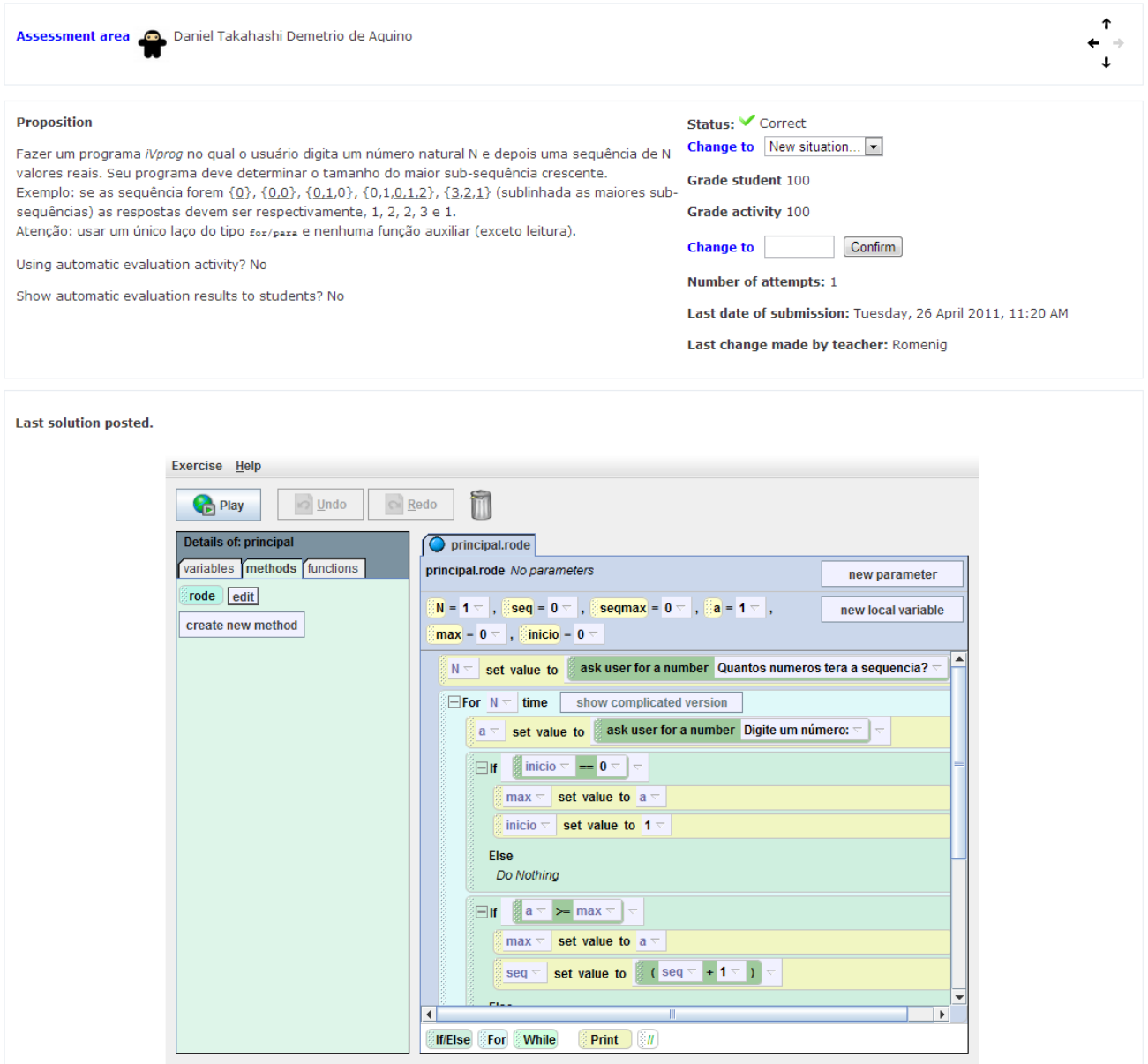


Figure 3. iVProg integrated with Moodle through the iAssign plugin.



Experiments

- The experiments took place during two semesters of an introductory programming discipline (MAC0110), in an undergraduate course of Mathematics. A blended learning aproach was adopted;

> The idea was to investigate if the use of iVProg could improve the understanding of how to solve problems algorithmically and how Visual Programming would impact the ability of using an imperative programming language (such as C) in tradicional programming environment;

- There were two experiments:
  Experiment 1) Observed exercises;
  Experiment 2) Using iVProg in formal courses;

Experiment 1

- Experiment 1 was conducted with 6 students volunteered, randomly divided in two groups, **G1** and **G2.** The activities were performed inside a laboratory with one student by computer. An observer, that would annotate the number of mistakes related to syntax and the number of times the program was run until the student evaluate it as correct, was aside of each student;

- The participants answered a questionaire composed of two questions:
  Q1) Do you know the C language? If yes, how experienced are you?
  Q2) Did you already attend MAC0110? If yes, what was the programming language and environment used?

- Each group was asked to solve two problems in one hour;
  G1: They solved the problems with iVProg in the beginning and with C language after;
  G2: They solved the same problem with C first, then with iVProg;

Exercise 1 (E1): *Build a program that has two numbers as input parameters and returning their average.*

Exercise 2 (E2): *Build a program that ask for a natural number N and a sequence of N numbers as input parameters and print the large and the smaller numbers of the sequence in the screen.*

Using iVProg:

a) *time in minutes to solve the exercise;*

b) *number of times that use the RUN button;*

c) *number of times that syntax or logical mistakes were discovered.*

Using C:

A) *time in minutes to solve the exercise;*

B) *number of times the program was compiled;*

C) *number of times the program run;*

D) *number of times that syntax or logical mistakes were discovered.*

TABLE I

RESULTS OF OBSERVATION – EXPERIMENT 1 - G1

		GROUP 1			
		student 1	student 2	student 3	average
	Q1	no	no	no	-
	Q2	yes	yes	no	-
E1 iVProg	a)	2	1	3	2,0
	b)	1	2	3	2,0
	c)	0	0	0	0,0
E1 C	A)	9	5	15	9,7
	B)	2	1	5	2,7
	C)	2	1	2	1,7
	D)	1	0	1	0,7
E2 iVProg	a)	8	2	14	8,0
	b)	1	1	2	1,3
	c)	0	0	0	0,0
E2 C	A)	16	5	21	14,0
	B)	4	5	5	4,7
	C)	4	1	2	2,3
	D)	4	5	2	3,7

Exercise 1 (E1): *Build a program that has two numbers as input parameters and returning their average.*

Exercise 2 (E2): *Build a program that ask for a natural number N and a sequence of N numbers as input parameters and print the large and the smaller numbers of the sequence in the screen.*

Using iVProg:

a) *time in minutes to solve the exercise;*

b) *number of times that use the RUN button;*

c) *number of times that syntax or logical mistakes were discovered.*

Using C:

A) *time in minutes to solve the exercise;*

B) *number of times the program was compiled;*

C) *number of times the program run;*

D) *number of times that syntax or logical mistakes were discovered.*

TABLE II

RESULTS OF OBSERVATION – EXPERIMENT 1 – G2

		GROUP 2			
		student 4	student 5	student 6	average
	Q1	yes	no	no	-
	Q2	yes	yes	no	-
E1 C	A)	23	50	35	36
	B)	4	10	11	8,33333333
	C)	2	1	11	4,66666667
	D)	3	6	10	6,33333333
E1 iVProg	a)	2	5,0	2,0	3,0
	b)	1	1,0	1,0	1,0
	c)	0	0,0	0,0	0,0
E2 C	A)	30	didn't finish	didn't finish	30
	B)	5		-	5
	C)	4		-	4
	D)	4		-	4
E2 iVProg	a)	13	didn't finish	didn't finish	13,0
	b)	5			5,0
	c)	3			3,0

Results of Experiment 1

Experiment 1 G1 (First iVProg then C):

The time spent to solve E1 using C was 5 times longer than using iVProg, even had been solved it immediately before using iVProg. For exercise E2 such a discrepancy was smaller, just 2 times longer. All students did both exercises using iVProg and C, in an interval from 13 minutes (*student 2*) to 53 minutes (*student 3*).

Experiment 1 G2 (First C then iVProg):

Only *student 4* did both exercises using C and iVProg, spending 68 minutes to finish them, even having previous knowledge about the C language (*answer Q1*) and had attended MAC110 before (*answer Q2*). *Students 5* and *6* couldn´t finish exercise E2 using both C and iVProg. They had focused on solving debugging problems and forgot the main goal that was to give an algorithm solution to the problem.

Experiment 2

- The main goal was to analyze iVProg influence in the learning process of algorithms and programming in an undergraduate course of Mathematics. It was conducted during 3 years during one academic semester a year (18 weeks) of an introductory programming discipline (MAC110);

- The experiment involved 3 classes, T1 (without using iVProg), T2 and T3 (using iVProg with different didactic approach), all of them from the same course and discipline, and having the same lecturer.

- Next table sumarizes and compare the courses;

TABLE III – Comparison between the courses

		T1 2005 (iCG)	T2 2010 (iVProg)	T3 2011 (iVProg)
Duration of the course		18 weeks	18 weeks	18 weeks
Usage of C language		from: 6th week to: 18th week	from: 7th week to: 18th week	from: 1st week to: 18th week
Usage of another aproach (iCG or iVProg)		from: 1th week to: 5th week	from: 1th week to: 6th week	from: 1th week to: 18th week
Exercises	non-C language (iCG or iVProg)	None	33 problems	25 problems
	C language	- 5 regular problems; - 3 modeling problems;	- 20 problems, with intersection with iVProg problems;	- 29 problems, with intersection with iVProg problems;
Number of exams		2 11th and 18th weeks	2 11th and 18th weeks	2 11th and 18th weeks

Results of Experiment 2

TABLE 4 – Results of Aplication

	T1 (2005)		T2 (2010)		T3 (2011)	
	Average	Variance	Average	Variance	Average	Variance
Frequency	56,48%	0,109	72,92%	0,072	71,02%	0,061
Test 1	3,48	8,3	5,59	8,7	5,85	5,64
Test 2	4,33	17,02	3,47	4,92	6,36	7,29
non-C	7	20,5	9,26	0,35	9,59	0,65
C	4,28	18,7	8,51	0,43	9,08	0,74
Final Average	4,06	10,44	4,59	6,31	5,24	6,19

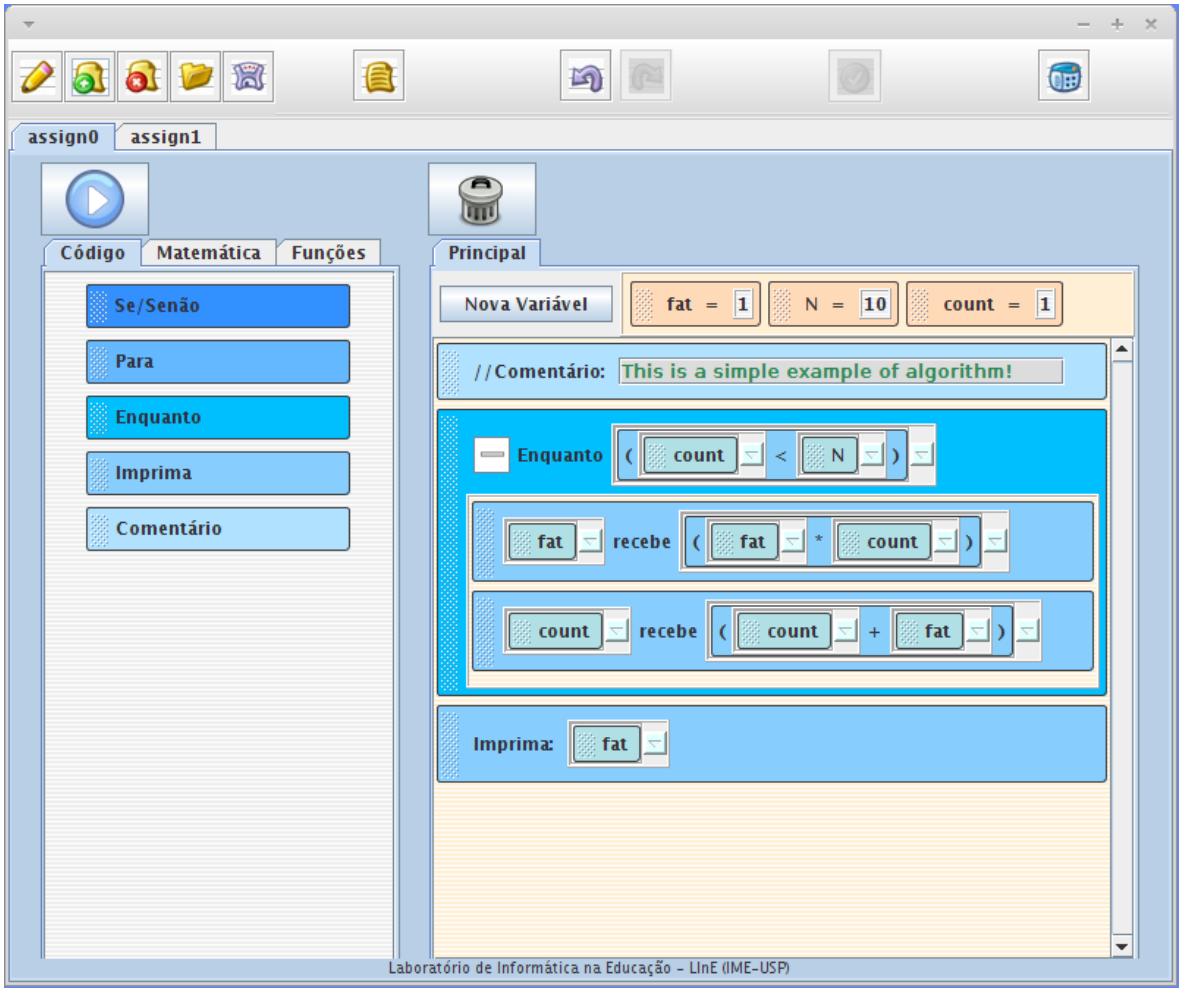
Conclusion and Future Work

- Experiments indicated that it does improve understanding about algorithms and may improve the understanding of a programming language such as C when used in combination of it from the very beginning;

- The system is being fully redeveloped using a Software Product Line approach [5];

- We also intend to improve:
  - feedback through an automatic assessment for students assignments;
  - educational range by implementing new resources that will provide different interfaces with various resources for an audience from kids to adults.

Currently


assign0
assign1
Código
Matemática
Funções
Se/Senão
Para
Enquanto
Imprima
Comentário
Principal
Nova Variável
fat = 1
N = 10
count = 1
//Comentário: This is a simple example of algorithm!
Enquanto (count < N)
fat recebe (fat * count)
count recebe (count + fat)
Imprima: fat
Laboratório de Informática na Educação - LInE (IME-USP)

Questions?

ACKNOWLEDGMENT:
Leonidas O. Brandão is partially supported by FAPESP grant 2011/10926-2 and CNPq grant 550449/2011-6.

Thank you for your attention!

References

[1] Lahtinen, E., Ala-Mutka, K., Järvinen, H., "A Study of the Difficulties of Novice Programmers", ITiCSE '05 *Proceedings of the 10th annual* SIGCSE conference on Innovation and technology in computer science *education,* Volume 37 Issue 3, September 2005, pp. 14-18.

[2] Bennedsen, J., Caspersen, M., E.,"Failure Rates in Introductory Programming", *ACM SIGCSE Bulletin*, Volume 39, Issue 2, June 2007, pp. 32-36.

[3] Conway, M., J., Pausch, R., "Alice: Easy to Learn Interactive 3D Graphics", *Computer & Graphics*, Vol. 31, Issue 3, 1997, pp. 58-59.

[4] Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E. "The Scratch Programming Language and Environment", *ACM Transactions on Computing Education,* Vol. 10 Issue 4, No. 16, November 2010

[5] Dalmon, D. L., Brandão, A. F. F., Isotani, S., and Brandão, L. O., Work in Progress – A Framework for Building Interactive Learning Modules, Proceedings of ASEE/IEEE Frontiers in Education Conference, 2011, S3E-1 - S3E-2.