A system to help teaching and learning algorithms

Leônidas de Oliveira Brandão (Computer Science Department at IME)
Anarosa A. F. Brandão (Computer & Digital Systems Engineering Department at POLI)
Romenig da Silva Ribeiro (master student at CSD-IME)

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

University of São Paulo – USP
São Paulo, Brazil

A system [illegible] teaching and l[illegible]thms

University of São Paulo (USP)
biggest (research university in Latin America)

Agenda

- The Context
- The iVProg
- Experiments Using iVProg
- Some results
- Conclusion and Future Work

The Context

- Teaching and learning algorithms is a big challenges for students in Science, Technology, Engineering, and Mathematics **(STEM)**
- Pointed difficulties in introductory programming courses:
  - students are presented to new learning environments in order to
  - edit, debug, and compile their algorithms,
  - written in some tradicional programming language, besides
  - need to solve *algorithmic problem* (formal reasoning)
- Usually, it's an *overload* of new information

The Context

- Difficulties in introductory programming courses:
  - new learning environments (compiler, editor, debug...)
  - language programming (formal language)
  - rationale difficulties in solving an “algorithmic problem“

- First approach is to simplify the environment in two directions
  - providing **Visual Programming** sytem
  - integrated with **Learning Management System (LMS)**

The Context

- Attempt to reduce didifficulties: simplify the envorinment in 2 directions
  - **Visual Programming (VP)** system similar to *Alice* and *Scratch*
  - integration with LMS particullarly **Moodle**
- **I**n 2009, we started/adapt a VP system, to be integrated to *Moodle* or others LMS that use the **interactive Learning Module** (**iLM**) [iLM/iAssign was presented in FIE 2010]

The iVProg

- It can to be used as an *application* as well as an *applet*
- Is based on *Alice*, but with less than 10% of Alice source code
- It was removed the *Alice* animation features

- **iVProg** - is based on *Alice* code (from Carnegie Mellow University) it is free (for now the JAR - the new one will be free software)
  http://www.matematica.br/ivprog

- **iAssign** - *iVprog* could be immersed on *Moodle*
  http://www.matematica.br/ia - in FIE 2011

- Such as *Alice* and *Scratch*, *iVProg* has *visual blocks* representing commands, such as *if-then-else* or *loops*;

Figure 1. iVProg commands toolbar

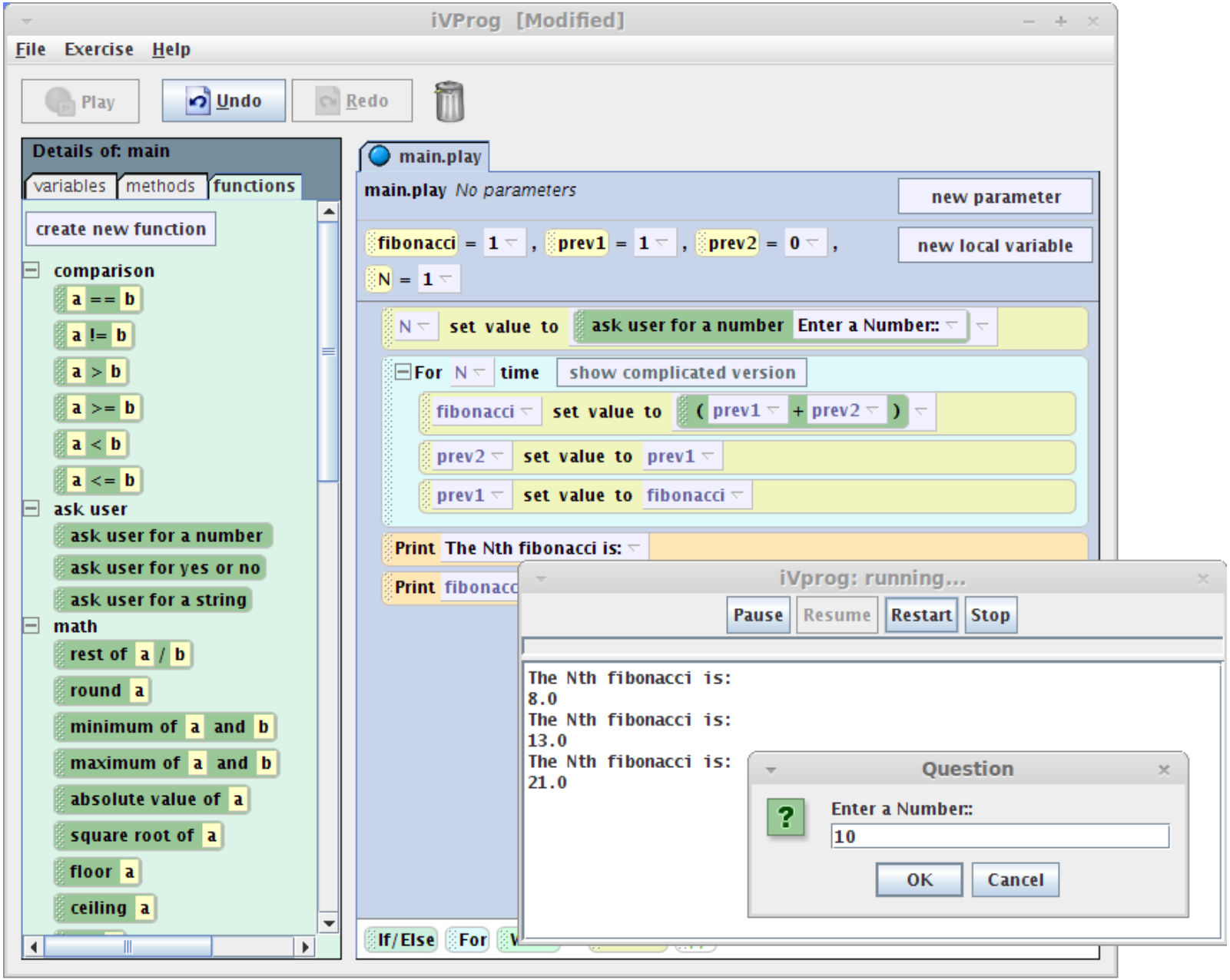


Figure 2. iVProg with a program to compute Fibonacci sequence

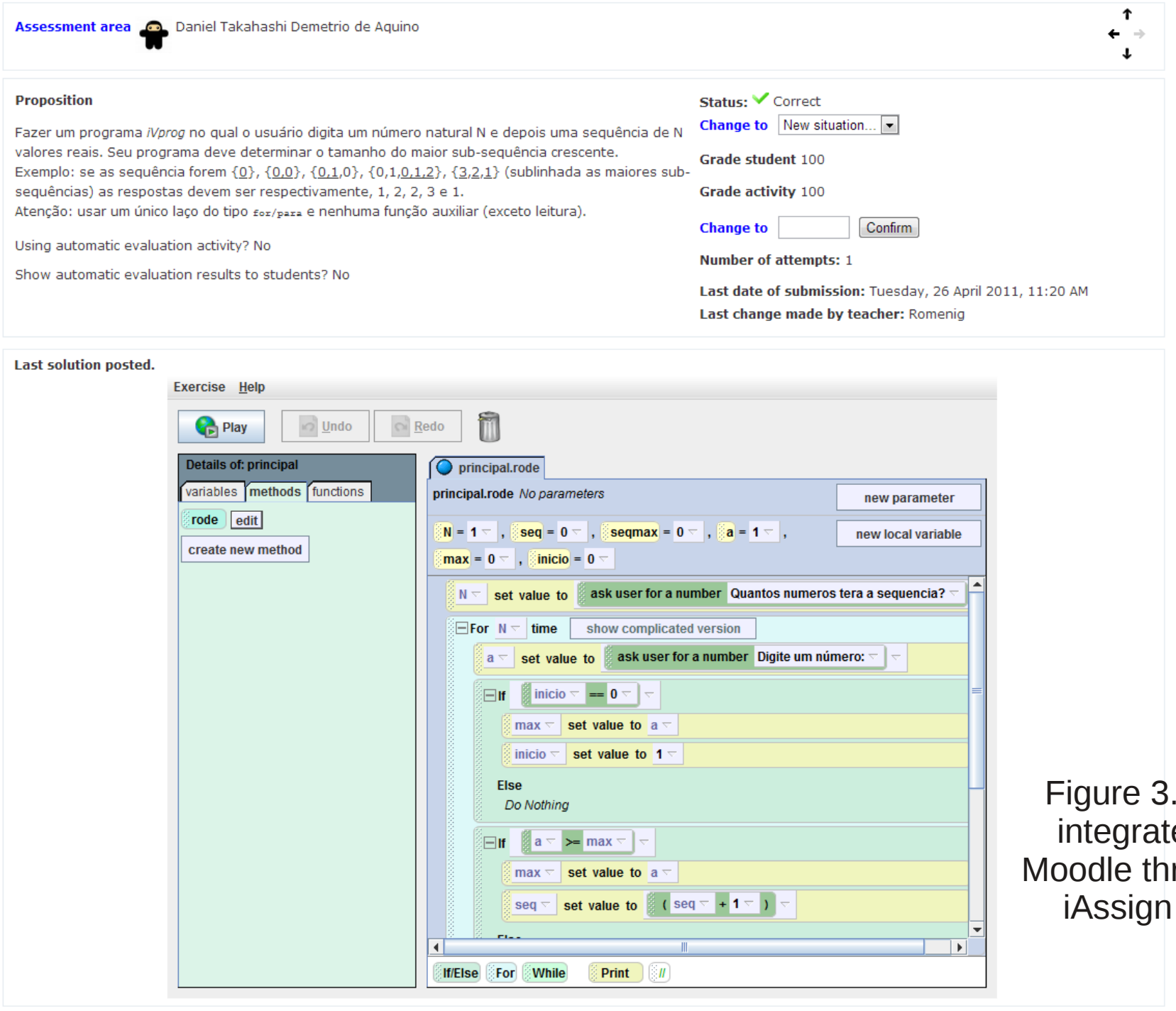


Figure 3. iVProg integrated with Moodle through the iAssign plugin

Experiments

- The experiments during mandatory courses - introductory programming
- Different semesters with classes of undergraduate Math students

- A blended learning aproach was adopted
- All classes in laboratory sessions (hands on)

> 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
> a tradicional programming environment

- There were two kinds of experiments:
  1) One short session of observation - exercises (one class)
  2) Semester course using *iVProg* in formal courses (with all classes)

Experiment 1: 2011

- Conducted with 6 students volunteered, one session of 1H
  Randomly divided in two groups (**G1**,**G2**)
  Activities observed in a laboratory with one student per computer
  Each observer registered:
  compilation attempts; sintax mistakes; time used

- The participants answered a questionnaire with two questions related to:

  Q1) Previous experience with C language
  Q2) Previous experience in the discipline (MAC110)

- Each group was asked to solve two problems in one hour:

  G1: should attack the problems first with iVProg, then with C language
  G2: should attack the problems first with C language, then with iVProg

- **Exercise 1 (E1):**
  Build a program that receive two numbers as input and print their average.

- **Exercise 2 (E2):**
  Build a program that ask for: a natural number N, and a sequence of N numbers. It must print the largest and the smallest numbers of the sequence.

- **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.*

- **E1:** *average of two numbers*
- **E2:** *largest and smallest numbers in a sequence*

- **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.*

- **G1**: tryed to solve first in iVprog, then in C
- **G2**: tryed to solve first in C, then in iVprog

Table I
Results observed in Experiment 1 - **G1**

		Student 1	Student 2	Student 3	Avarage
Experi-	Q1	no	no	no	
ence	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

Table II
Results observed in Experiment 1 - **G2**

		Student 4	Student 5	Student 6	Avarage
Experi-	Q1	yes	no	no	
ence	Q2	yes	yes	no	
E1 C	A)	23	50	35	36,0
	B)	4	10	11	8,3
	C)	2	1	11	4,7
	D)	3	6	10	6,3
E1 iVprog	a)	2	5	2	3,0
	b)	1	1	1	1,0
	c)	0	0	0	0,0
E2 C	A)	30	did not finish	did not finish	-
	B)	5	-	-	-
	C)	4	-	-	-
	D)	4	-	-	-
E2 iVprog	a)	13	did not finish	did not finish	-
	b)	5	-	-	-

Results of Experiment 1

G1 (First iVProg then C):

- The time spent to solve E1 using *C* was 5 times longer than using *iVProg*, even student knowing the solution under *iVProg*
- In exercise E2 the discrepancy was smaller, solution under *C* took twice the time with *iVprog*.
- All students solved both exercises with *iVProg* and *C* *(student* 2: 13 minutes - *student* 3: 53 minutes)

G2 (First C then iVProg):

- Only *student 4* solved both exercises (with *C* and *iVProg*)
- But spent 68 minutes to finish them, even with previous experience with *C* and had attended MAC110
- *Students 5* and *6 did not* finish E2 (neither with C, nor with iVProg)
- They focused on solving debugging problems and got stuck...

Experiment 2: 2010 & 2011

- First discipline (MAC110) of programming
- To undergraduate Math students
- The main goal was to analyze *iVProg* influence in the learning process
- It was conducted during 2 years (1 semester each, 18 weeks)
- It was used a control class of a previous discipline with the same course

- The experiment involved 3 classes:
- T1: control, not using visual programming environment (2005)
- T2 and T3: using iVProg with different didactic approach (2010, 2011)

- All classes in laboratory sessions (4H per week), of the same course and discipline, and with the same teacher

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)	10 problems	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 IV
Results of the the main assessments

	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

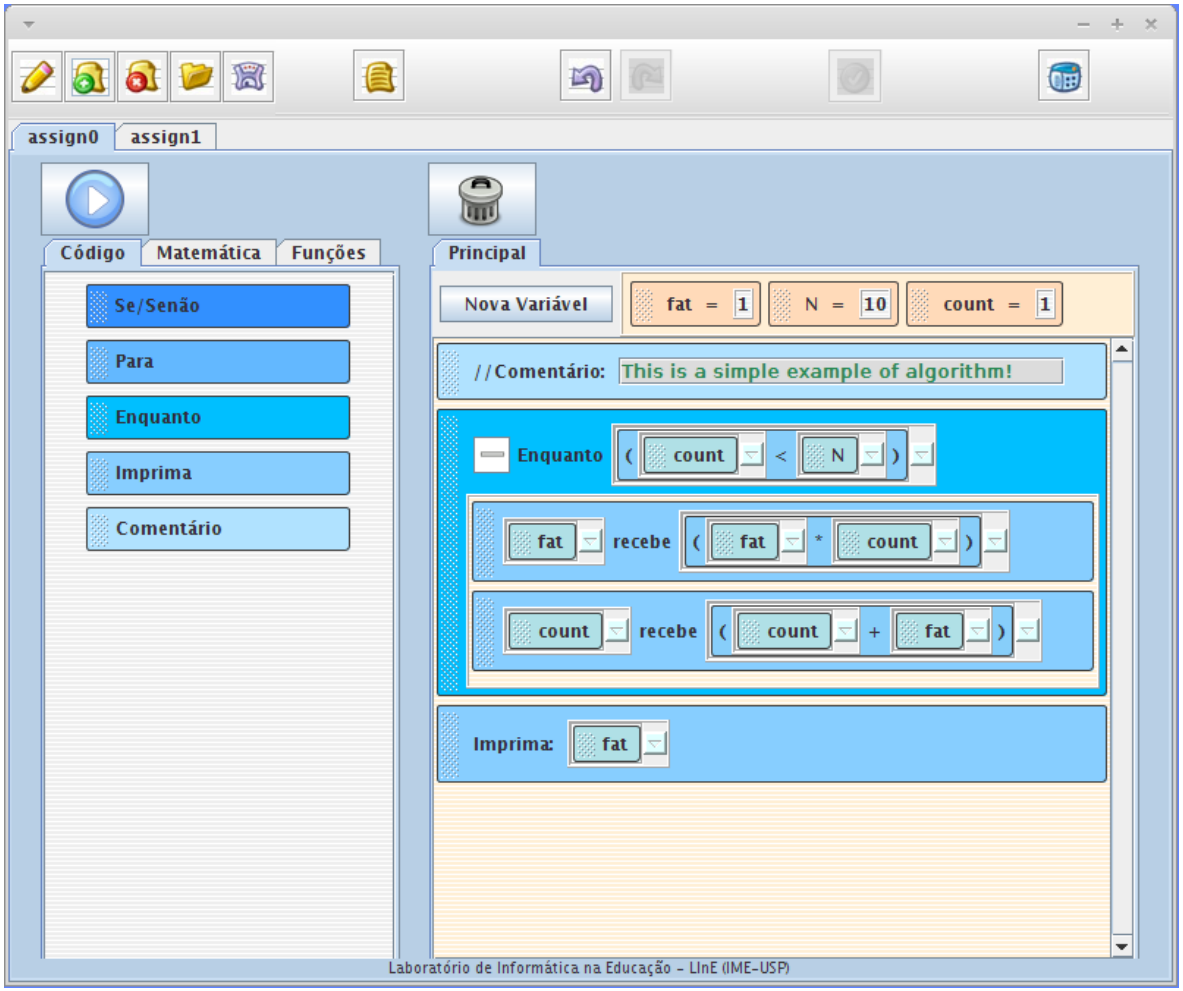
Conclusion and Future Work

- Experiments indicated that *Visual Programming* with *iVprog* could
  - improve understanding about algorithms
  - reduce difficuties with programming environment
  - improve the introduction to traditional programming language *C* (when used in combination of it from the very beginning)
- But the transition to C must be better studied
  e.g.,
  is it necessary to "prospective teacher"?
  a translator *iVprog*-*C*-*iVprog* could help?
- Now we are completelly rebuilding *iVprog*
  (software engineering problems: maintenance, evolution, ...)

Conclusion and Future Work

- The *iVprog* is being fully redeveloped using a *Software Product Line* approach for the iLM family of software (work in progress FIE 2012)
- Some of the next features in *iVprog* must be:
  - automatic assessment: quick feedback for students
    boolean feedback (heuristic)
    tutoring system (detecting typical student mistake)
  - different interface to different student (elementary, secondary and superior)
  - communication with Arduino robots

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?

Thank you for your attention!

Any Question?

- iVprog (version 0.2): http://www.matematica.br/ivprog
- iAssign (version 0.9): http://www.matematica.br/ia

- **ACKNOWLEDGMENT:**
  Project partially supported by
  FAPESP grant 2011/10926-2 and CNPq grant 550449/2011-6

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.