Information Technologies and Sciences Department

Botbeans

Pedro Miguel Aparício Dias

Master in Open Source Software

Supervisor:

Prof. Dr. Sancho Oliveira, Assistant Professor,
ISCTE-IUL

June 2011

Abstract

Programming can be a daunting task from a beginner's perspective. Since earlier times of computer programming, tools have been designed and developed in order to make programming friendlier to beginners. However the majority of these tools target beginners that are already motivated and have an idea of what computer programming is. This allows these tools to skip the initial requirements for learning how to program since these beginners will compensate with their motivation and effort. This thesis describes a learning tool called Botbeans. By using a new hybrid visual programming language with a tangible interface, Botbeans creates a highly motivating and collaboration friendly environment to present what is programming to a user that never had previously contact with it. The design and implementation of Botbeans are described and the results of some initial experiments with students are analysis.

Resumo

Aprender a programar pode ser uma tarefa difícil e assustadora do ponto de vista de um iniciante. Desde dos tempos iniciais da programação diversas ferramentas foram desenvolvidas com o intuito de tornar a aprendizagem da programação mais amigável a iniciantes. Algumas destas ferramentas têm como publico alvo iniciantes já altamente motivados para a programação e já com uma ideia do que esta é e para que serve. Isto permite a estas ferramentas saltar alguns dos pré-requisitos necessários para começar a aprender a programar, visto que este tipo de iniciante irá compensar com a sua motivação e empenho. Esta tese descreve uma ferramenta de aprendizagem chamada Botbeans que utilizando uma linguagem gráfica híbrida e uma interface tangível cria um ambiente altamente motivante para demonstrar o que é a programação e para que serve a um utilizador que nunca teve contacto com esta. O design e desenvolvimento do Botbeans são descritos ao longo da tese assim como os testes inicias já efectuados.

Table of Contents

Table of figures

Table of tables

1 Introduction

Learning how to program can be a difficult task for beginners, it requires using multiple skills. In addition to learn the basic topics in programming and acquire the needed abstract and algorithmic thinking a beginner needs to learn the syntax of the programming language he's using.

Each of one of these topics separated are easy to tackle, the problem is that typically a beginner needs to tackle all of these topics simultaneously which can overwhelm and discourage him, as said by (Kelleher & Pausch, 2005).

In order to overcome some of the previous obstacles in learning how to program, researchers started developing graphical languages to remove some of the obstacles, like language syntax, keeping the student focused in the algorithm and their solutions. "*Many graphical programming systems exist to break down the barriers to computer programming, helping to reduce the threshold for beginners to learn how to program and reap the benefits of programming*" (Roque, 2007)

There are mainly two types of graphical representations for these visual programming languages: node-based and block-based.

In block-based programming, users connect a group of blocks together like in a puzzle, building their program this way. "*In graphical block programming, users manipulate and connect puzzle-piece objects to build their programs.*" (Roque, 2007). The algorithm is validated by the structure of the language itself because only complementary blocks (same shape) may connect to each other, not allowing an invalid program to be built. (Roque, 2007). Besides shape, color is also used to identify blocks, but usually it is used to represent the data or logic flow in the algorithm.

Node-based visual languages allow the logic and data flow to be easily identified, simultaneously allowing its users to easily associate each node to the corresponded programming structure in a textual programming language (Smith, 2009). In node-based languages the number of outputs and inputs each node can have

are usually used to validate the algorithm during its implementation. Like block-based languages this is used to reduce and/or eliminate implementation errors.

Graphical languages have a scalability problem, if the problem requires a complex solution it will take screen real estate and the solution will become harder to understand. But since the main objective of these graphical languages is to teach/learn programming this problem will not be a important factor in the real world. (Roque, 2007)

It has been proved that novice users learn more efficiently when they get involved by designing and/or creating something. "*Years of research have shown that children learn best when actively involved in designing and creating their own inventions.*" (Stern, 2007).

Tangible interfaces are known to involve more its user in the task he is trying to accomplish (Horn, Solovey, Crouser, & Jacob, 2009), this is why it is not surprising that much of the research conducted with tangible interfaces is in education (O'Malley & Stanton Fraser, 2004) allowing the user to feel the knowledge source.

This type of interfaces allow students to touch/feel in the real world what they are creating, letting them touch and experience the results of the application of their existing knowledge (Jacob et al., 2008). Another big advantage of TUIs in education is the collaboration characteristic in them. Very often the usage of TUIs ends up as a catalytic for collaboration during the creation process, this is very important in education because every participant ends up with a richer experience, since each one will bring its own contribution to the table. (Horn et al., 2009).

The main objective of this thesis was to develop a learning tool that was able to motivate and show to beginner that never had contact with programming what it is while giving them the pre-requisites needed to learn programming, requisites like reasoning and algorithmic thinking.

The learning tool developed is called Botbeans (Pedro Dias & Oliveira, 2010). By using a new hybrid visual programming language and a tangible interface it gives the pre-requisites needed to learn more advanced topics in programming in a motivating environment from the user perspective.

An initial research was conducted about the human learning process. This research allowed to understand how different learning methodologies, active and

passive learning, could affect the process of learning programming and which could be the best to help solving the problem at hands. This research is described more in detail in Chapter 2 of this thesis.

After the initial research in learning theory, in Chapter 3 is described the next step which was the identification of other learning tools that try to aid in learning programming each one had its own different approach, all these approaches were studied while identifying their best characteristics.

The next obvious step was to develop a tool that implemented the strategy defined by the previous research and studies, how this tool was designed and implemented is described in Chapter 4. Each design decision was justified by all the previously research, allowing to keep focus in the main problem.

Finally after a fully functional version was implemented (Pedro Dias & Sancho Oliveira, 2011) initial tests conducted with Botbeans are described and analyzed, identifying where they should be improved in order to allow a good identification of possible improvements in Botbeans.

2 Motivation

Younger students are very often scared from programming before they even understand what it is or which purpose it serves, because of its complexity.

Learning how to program is complex, besides learning the syntax of the language, the apprentice needs to gain additional skills, like abstract and algorithmic thinking. Students that are naturally inclined to computers will make an effort and try to overcome this extra complexity to learn how to program, but students that don't have this predisposition will probably not try, scared by the complexity of programming.

There are multiple learning tools to help a student in the learning process of programming. The majority of these tools try to simplify the process of learning of how to program by removing complexity from the programming languages; each tool has its own approach to accomplish this goal.

A lot of these tools focus in the implementation of the algorithm, the objective of Botbeans is not to focus in the implementation of the algorithm but instead how to develop the algorithm using abstract and algorithmic thinking. This is the first pre-requisite for learning programming.

2.1 Learning process

The human learning process was always involved by some mystery, but there is no doubt that there is a correlation between learning, knowledge and intelligence, these three are interconnected in some way, their connection will be used in order to find out how can the learning process be improved by a more active type of learning.

While studying the human learning process, educator Edgar Dale, developed the "Cone of Learning" (Figure 1). It is a cone shaped graphic that shows that there is a direct relation between active activities during the learning process and how easily the knowledge gets absorbed and understood by its student.

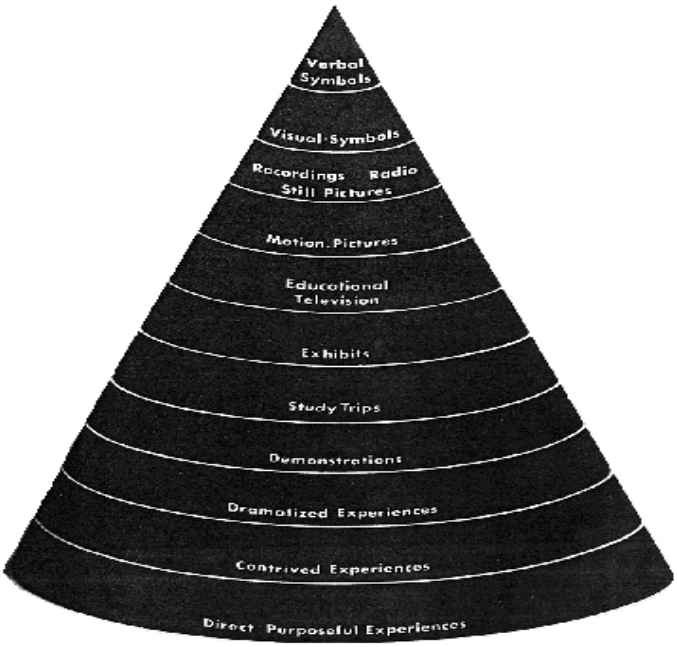


Figure 1 Original Cone of learning

His initial model showed which indicators better represented the knowledge retention. It can be seen from the cone that Edgar Dale believed that the more active/hands on experiences have the power to behave as catalyst in knowledge retention, while comparing them against more passive experiences. This was one of the earlier studies focusing the importance of a more active type of learning, a type that allows the student to interact with the knowledge's source and even with the knowledge itself.

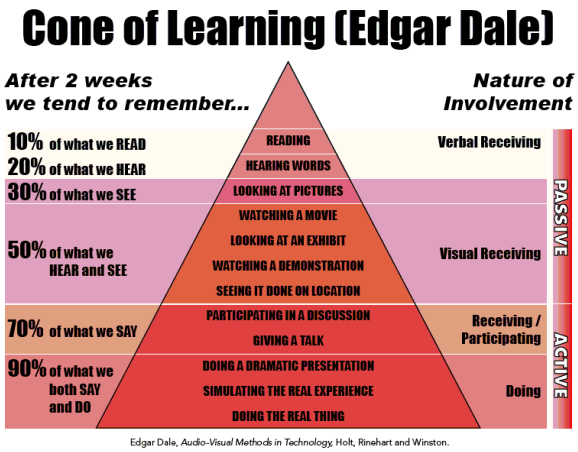


Figure 2 Cone of Experience, incorrectly named Cone of Learning

Some years after Dale published his cone, a new version of the cone appeared, there is no track of who developed this new cone called “Cone of Experience” (Figure 2). This new version is very controversial, since it features percentages spread along the cone. This percentages were never justified and don’t have any scientific background, there is even multiple versions with different percentages, passing by has the original Dale’s cone (Figure 3).

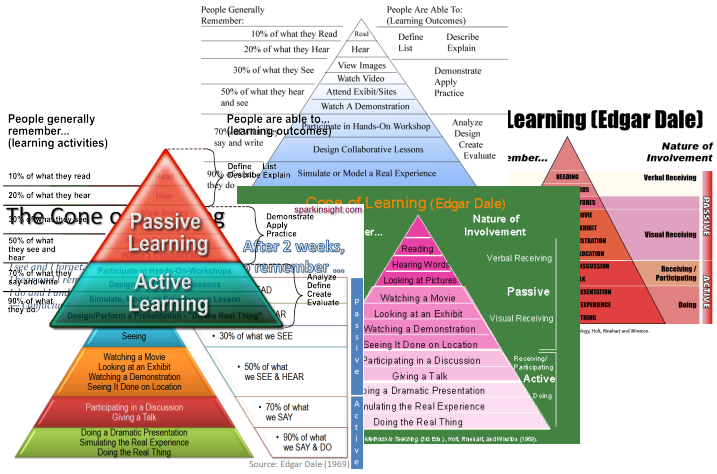


Figure 3 Cone versions

An important factor while absorbing any type of knowledge is its retention power, meaning how long will it stays active in the student mind. This of course is much related to how often the student will use the new skill set he acquired from the absorbed knowledge. This is also related with a more active approach in learning, as Benjamin Franklin said, “Tell me, and I forget; teach me, and I may remember; involve me, and I learn.” Or “I hear and I forget. I see and I remember. I do and I understand” from Confucius.

One person can stay months studying something, but it’s only when he gets actively involved in that something, that he truly understands how it works; some researchers call it a type of reverse engineering.

Reading and writing are, nowadays, communication skills naturally acquired, but they are an important type of skills, since these skills are used in the most typical learning processes. But there are more communication skills that can and should be used, like listening to a lecture or watching a movie, or even brainstorming an idea

with a group of people, these are all examples of a more active group of activities that can improve the learning process.

There have been studies about how the human brain behaves during the learning process. It has been proved that this process involves multiple mental operations, which interconnect with senses as input. This process takes time and this time changes from person to person and it directly depends on what is being learned.

2.1.1 Active and passive learning

In an educational sense, two types of learning process can be defined: passive and active. Passive is when a student studies a subject without directly interacting with the subject he is studying, it's the most common type of learning. He will try to retain knowledge about the subject without interacting with the knowledge source using his senses. Active learning allows the student to interact with the knowledge and its source and by using his senses.

2.1.1.1 Passive learning

Passive learning allows in a short time to retain a lot of information, but although the information was retained does not mean it was understood or even processed. From the teacher point of view it allows to have a more controlled learning environment since all the factors can be previously predicted, this way unexpected situations can be reduced to a minimum to the student and teacher.

There are also disadvantages in passive learning, the most important one is the fact that there is no way to the teacher to know if the student understood or assimilated the information he just absorbed, the opposite may also occur where the student may not feel comfortable enough to report his doubts or problems about the studied subject.

This problem may result in a complete demotivation from the student side, leading the student to give up. Once this happened it's hard to repair this situation since the student already acquired "antibodies" to the subject he was studying and from which he just gave up.

2.1.1.2 Active learning

In active learning the student gets actively involved in the learning process, he is not a simple player, he is also the process and the process is he. This is positive since the student will look at the subject he needs to study not like an obligation but from a constructive perspective.

In order to absorb the knowledge he will need to create and it will be he's creation, something he will intellectually own, and this will boost student motivation. Consequently students loose the fear of exposing their doubts and questions about the subject, reducing the differences between students, since the ones with more difficulties will reach out to the teacher to try to compensate their difficulties.

Like in passive learning not everything is good, active learning is more time consuming while also asking more from the teacher. This isn't always a bad thing, but can be problem when a big number of students are involved, traditionally this type of learning works better with a smaller class than passive learning.

Usually in passive learning the points of interaction are very limited in number, like reading something or listening to a teacher explanation, this isn't enough in active learning. Active learning needs new methods of interaction between the student and the target he's studying, and all these methods need to actively involve the student, things like public discussion of case studies for example.

Was the search for new types of interaction that brought the tangible interfaces to the learning process, they are a tool that the teacher can use to motivate his students by appealing to the student to use his senses in the process, involving him in the studied subject.

3 State of the art

3.1 Tangible interfaces

Tangible interfaces are a new approach in education, they allow the student to use his senses to interact with an object in the real world, and this object is usually where the inputs and outputs of the used learning platform reside. They can use different type of senses like vision, audition, or feel, this allow to grab the attention of different types of students, since each human being uses and feels their senses different from the other.

Tangible interfaces can be included has a type of interface of ubiquitous computing or ubicomp, this model was developed by Mark Weiser at Xerox Palo Alto Research Center in 1988, and initially it had only three defined type of devices in mind, in recent years ubicomp is usually associated with interfaces that allow the computer to enter the human physical world instead of the opposite, usually an human needed to enter the computer environment to use them (York & Pendharkar, 2004).

This type of model is perfect to use in programming learning, traditionally students needed to enter the computer environment, which uses the keyboard and mouse has input and the screen as output, to learn programming, but this requires that the student first needs to understand the inputs and outputs of the system, or else he will not understand if the output displayed on the screen is what he wanted or not. With a tangible interface this problem can be improved, by using a TUI (tangible user interface), new types of input are created and the old output is swapped with an output that is in the physical human environment.

When using this type of interfaces in education some factors need to be addressed, it's through this interface that the student will express his thinking and creativity, so it's very important that the interface allows the student to express himself and can never be an obstacle to his creativity.

In the end the student will feel useful and since he is using his creation to learn he will have a feel of ownership and accomplishment, decreasing that initial feeling of obligation and being forced to learn something.

3.1.1 Collaboration catalyst

How can an interface be a catalyst for collaboration? The best example nowadays is the Internet. It is a catalyst for communication, before the Internet people communicated a lot less, people they didn't had this information highway at the reach of their hands everywhere they go.

With a TUI, the tool gained multiple input points, but the important factor here it's not the number of input but the capability of using some of them in parallel by multiple users at the same time. There are two main points of interaction: the computer where the algorithm is going to be implemented and the TUI, which will be the eyes and ears of the tool in the real world. This allows accommodating multiple users in same experience.

3.2 Other educational tools to teach programming

3.2.1 BlueJ

The BlueJ tool was initially developed by a research project related to object-oriented programming teaching and it's currently maintained by La Trobe University and University of Ken. Its aim is to help students who are new to object-oriented programming and specifically new to Java (Xinogalos, Sartatzemi, Dagdilelis, & Evangelidis, 2006; Xinogalos, Satratzemi, & Dagdilelis, 2007), although BlueJ gives a good background for any object-oriented programming language; it was developed with Java in mind.

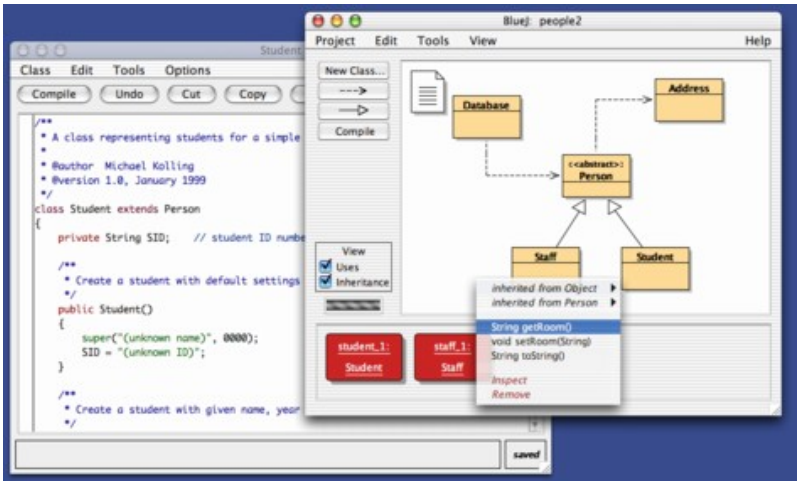


Figure 4 BlueJ

Like many other tools for programming teaching it also supports a graphical language, BlueJ's graphical representation is very oriented for the object-oriented paradigm, where each shape will represent a class and each will contain it's attributes and methods. Instead of a complex graphical representation, BlueJ developed a system of dialog windows, where each problem is divided in smaller problems, which is in favor of an object-oriented design.

BlueJ's graphical representation can have similarities to the UML representation, in particular to class diagrams. Each relation between classes is drawn but using different types of connectors, each connector represents the type of relation between the two classes it connects (extends, implements, etc.).

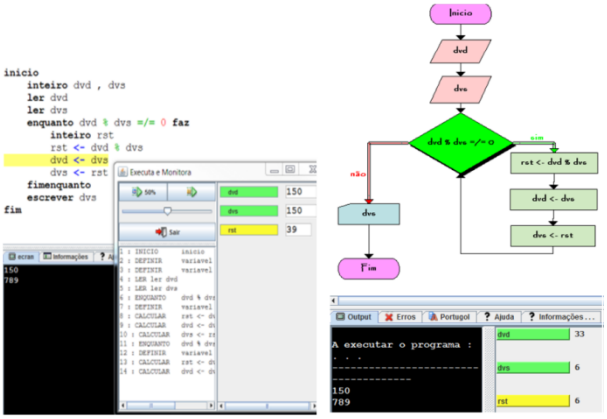
Another key feature of BlueJ is its capability of sharing projects in a central repository, using this repository students can checkout, update and commit changes to the different projects. This is very important because it allows the students to do collaborative work and learn about version control systems without it's usually complexity.

Since BlueJ focus in some programming advanced topics, its target audience is students that already have knowledge about the basic programming topics which now jumping to more advanced topics.

3.2.2 Portugol

Portugol IDE is an environment for algorithm exploration designed for the teaching of programming developed at Polytechnic of Tomar in 2005.

It uses a Portuguese lexicon-based language for encoding algorithms (Portugol language) and a graphic language (flowchart). These languages have been defined in a manner that allows their execution by a computer. Portugol's flowcharts are a graphic language consisting of parameterized geometric shapes and arrows representing the flow of execution inside the implemented algorithm.



a) portugol b) flowchart

Figure 5 Portugol IDE

Portugol has a small set of instructions and operators with a flexible syntax. Although its flowchart is executed through Portugol language, this process is transparent to the student.

Initially this tool supported only the Portuguese language, but in future versions it will support multiple languages, alloying its users to even add new languages to the system.

The key feature of Portugol is its ability to keep the textual representation and graphical representation synchronized (Manso, Marques, & P. Dias, 2010), allowing the student to write his algorithm in one representation and with one click switch to the other one.

All conversions needed to switch between representations are done transparently to the user, avoiding overloading the user with unnecessary back end information.

The student has the ability to choose between multiples modes of execution: debug, run and stepped run. In debugging mode student can, pause and resume the execution has they see fit; the node that is in execution is always highlighted. Stepped run is similar to debug but execution automatically continues in a speed that the student can choose, allowing him to easily reverse engineer an algorithm when needed.

3.2.3 Alice

Alice main focus like BlueJ is object oriented programming, although Alice does not feature a visual programming language it implemented a system that's does not requires its user to remember the language syntax, removing one of the problems from textual programming languages.

Alice's IDE features an interactive helping system, similar to the traditional code completing solutions, allowing the user to see all the available options in each instruction, allowing writing code without directly typing it or even knowing the language syntax.

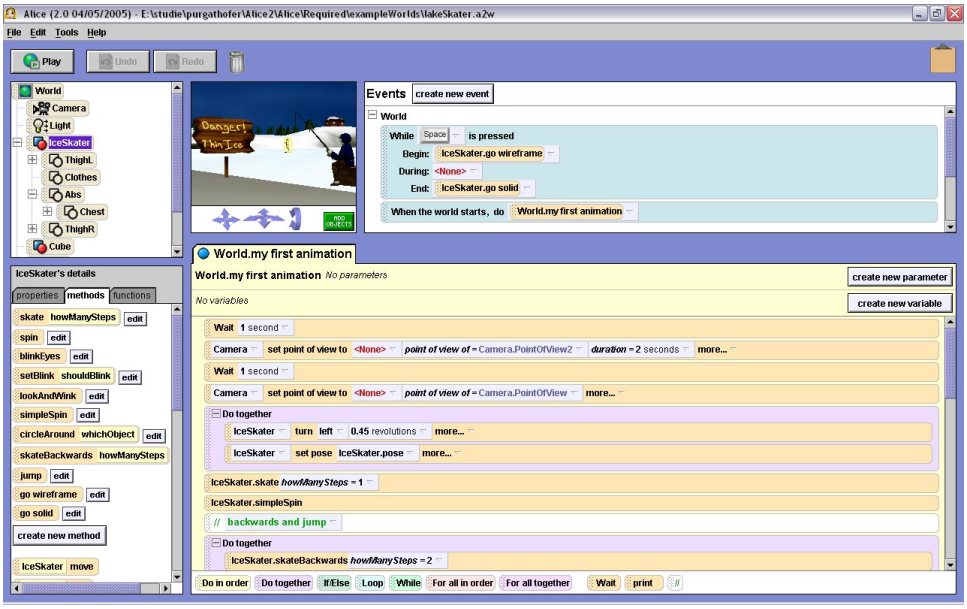


Figure 6 Alice

The main objective of Alice is computer animation, where the user has a group of 3d models available to interact with using the Alice's programming language.

These animations allow the users to easily understand the relationship between the programming statements and the behavior of objects in their animations.

3.2.4 Greenfoot

Greenfoot like Alice and BlueJ focus in object oriented programming more exactly in Java and features an IDE which fully supports the Java language. The key features of Greenfoot are its capability of object interaction and visualization, it supports a system for object interaction similar to BlueJ where the objects move and interact in a 2D grid map, and users can program the behavior of the objects in the grid.

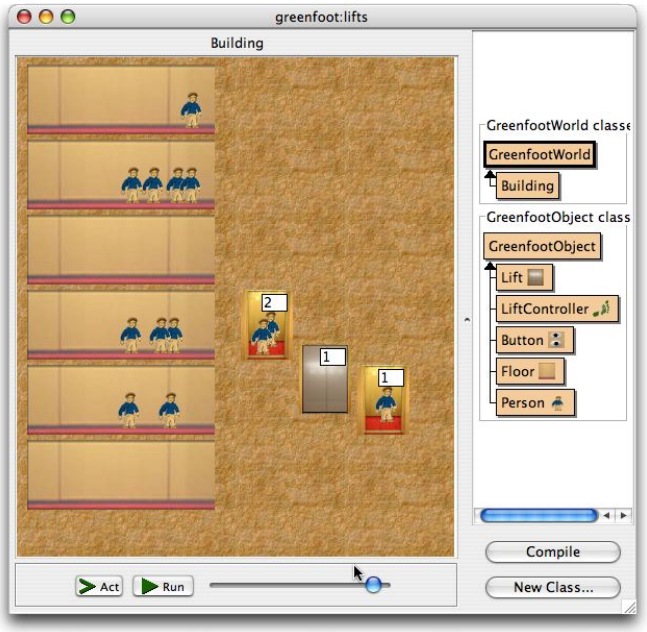


Figure 7 Greenfoot

Each object can have a graphical representation in the grid, this way the user sees his objects not in the usual abstract world but in a graphical world where the objects can interact with each other.

Like other learning programming tools that focus in object oriented programming, these require the student to have previous knowledge of the basic control structures in programming.

3.2.5 Scratch

Scratch is a programming language and developing environment for teaching and learning programming without any previous experience in the matter. The first version was developed by the Lifelong Kindergarten group at the MIT Media Lab and released to the public in 2007.

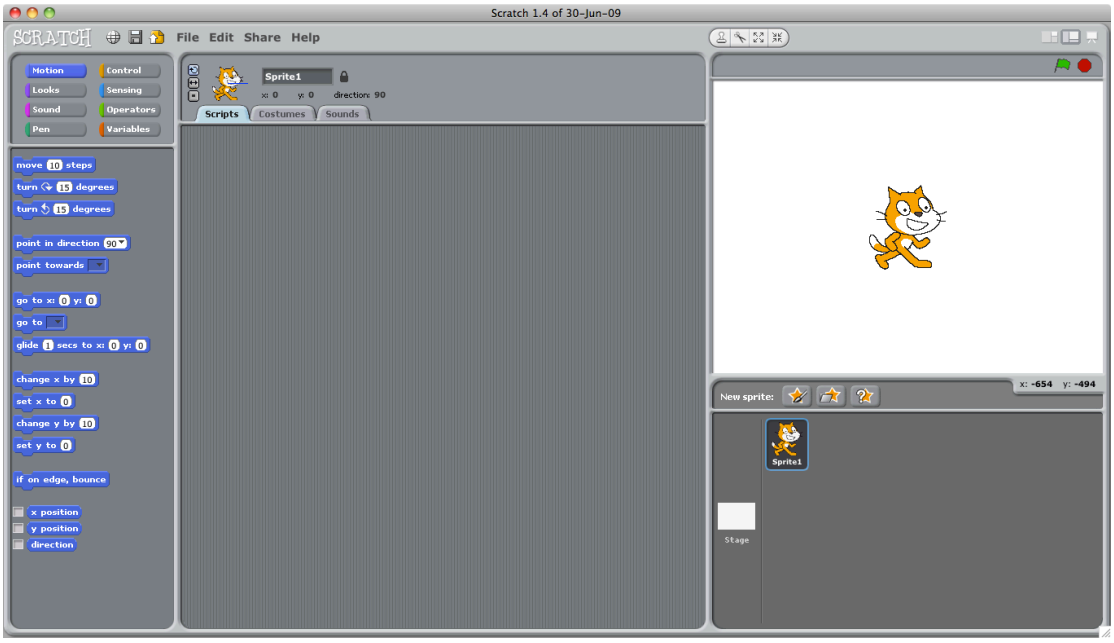


Figure 8 Scratch

Scratch's IDE is divided in three main panes: palette, scripting area and the stage. The palette and scripting area is where the algorithm implementation will occur, since it uses the classical approach of visual languages, which is drag from the palette, drop into the drawing area. In the stage pane is where the sprites will reside, these sprites are a key feature in Scratch, allowing users to program these sprites giving them a behavior.

The visual programming language used by Scratch is a pure block based language influenced by the StarLogo project.

3.3 Graphical Programming

In computing, a graphical or visual programming language is a programming language that allows its users to implement algorithms by manipulating basic

programming elements graphically rather than by specifying them textually. There are different types of graphical languages each one with its own characteristics and purposes.

3.3.1 Why graphical

Textual representation is a lot more complex to human than a graphical representation. This can be observed in the earlier development of a child, where if a phrase is shown to a child, in order to that child be able to comprehend the meaning of that phrase and the information within it, will need to have a few prerequisites like knowing how to read, this by its own has requisites, like know the language in which the phrase was written and the charset used by that language. But if a drawing with the same meaning of the phrase was shown to the child, that will have a much higher probability of being understood, since that child will not need to know how to read and will not need to know a language or its charset to understand the message.

The same also happens with a programming language, programming languages also have their own charset, syntax and dictionary. Without knowing these first it's very hard for a student to understand the algorithm implemented in a textual programming language.

Using a graphical language to define an algorithm increases the probability of success of a student understanding the algorithm behind it, although this is very subjective since it highly depends on language's definition. If the graphical language has a complex definition the problems of a textual representation will also occur and prerequisites will appear for the student. This must be avoided at all costs.

3.3.2 Graphical languages

There are different types of graphical languages, some focus their definition around the different data types others in algorithm's data flow or logic flow. The most common representation is a diagrammatic representation, where a flowchart is usually built by connecting different types of nodes, using arrows as connectors.

Each language will have different types of information within the flowchart. For example in NXT-G two diagrams form the flowchart, one diagram represent the logic flow and other the data flow, which interconnect at the nodes.

Diagrammatic representation in the format of a flowchart was one of the first graphical representations of an algorithm, in nowadays is well known for its usage in modeling languages like UML.

Recently a different type of representation is gaining points and users, this new type of representation was developed in MIT for StarLogo TNG and after was transformed in a framework by the name of OpenBlocks.

3.3.2.1 Representation types

3.3.2.1.1 Node / diagrammatic based

In node based visual languages, users have a set of nodes available, which they can connect to achieve the desired result. Each node has properties and a specific functionality that normally represents an operation, after applying the operation to the input it will produce an output. The connections between the different nodes represent the logic flow between them. The graph produced by all the nodes interconnected represents the algorithm and provides the student with an overview of all the logic flow inside their implementation.

Figure 9 Diagrammatic Hello World

This overview is one of the best advantages of node based visual languages since it allows the user to quickly understand the logic and/or data flow inside his implementation (Smith, 2009).

Using the memory display during execution, users can comprehend the impact of each node by watching the data before and after each node affects it, allowing them to divide one big problem into smaller problems that they can easily comprehend and solve.

There is although a problem with this type of representation, usually logical and mathematical expressions are displayed in a textual format as a property of a node (Kelleher & Pausch, 2005), if the user has no problem comprehending the expressions this will not be an obstacle although it creates a pre-requisite for its users.

3.3.2.1.2 Block based

Block-based languages are represented by a construction made by grouping different blocks together. Similar to node- based each block has its own functionality and properties. In education this representation is very often associated with the creative process of children playing Legos.

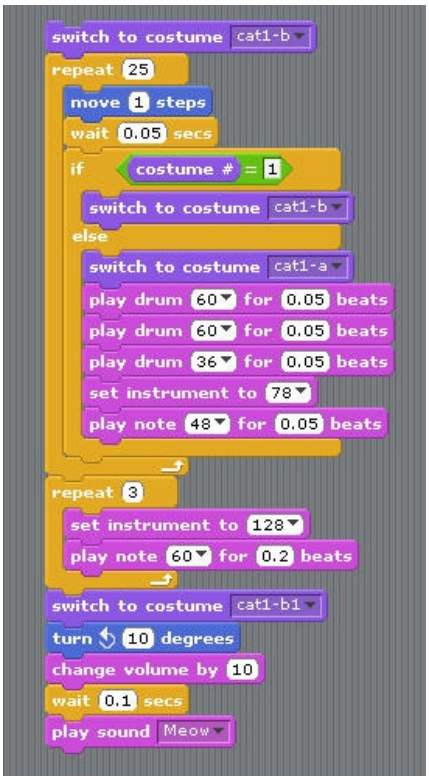


Figure 10 Block based visual language

This type of representation is very good for expressions and data types, because each block have a color and a shape and each shape corresponds to a different data type, allowing a natural differentiation of each data type by just looking at the blocks (Roque, 2007).

Like a puzzle each block can only connect to blocks with compatible shapes, validating student's construction at building time. Data type identification and validation by block's color and shape is a good improvement from the textual representation in node-based, in terms of expressions.

3.3.3 Other Graphical languages

3.3.3.1 LabVIEW

LabVIEW is usually associated with data acquisition, instrument control and industrial automation; it is developed by National Instruments and had its first version released in 1986 with support only for the Apple Macintosh.

Each LabVIEW program has three important components: Block diagram, front panel and connector panel. Connector panel will represent the entry parameters of the program, is used when this program is used as component in another program like a subroutine or module. In the block diagram is where all the logic is implemented by using a visual language called "G". Finally the front panel is the program's frontend to the user, since LabVIEW is usually used in industrial environment and instrument control, front panel resembles to a control panel from which the user can control physical control points in the real world.

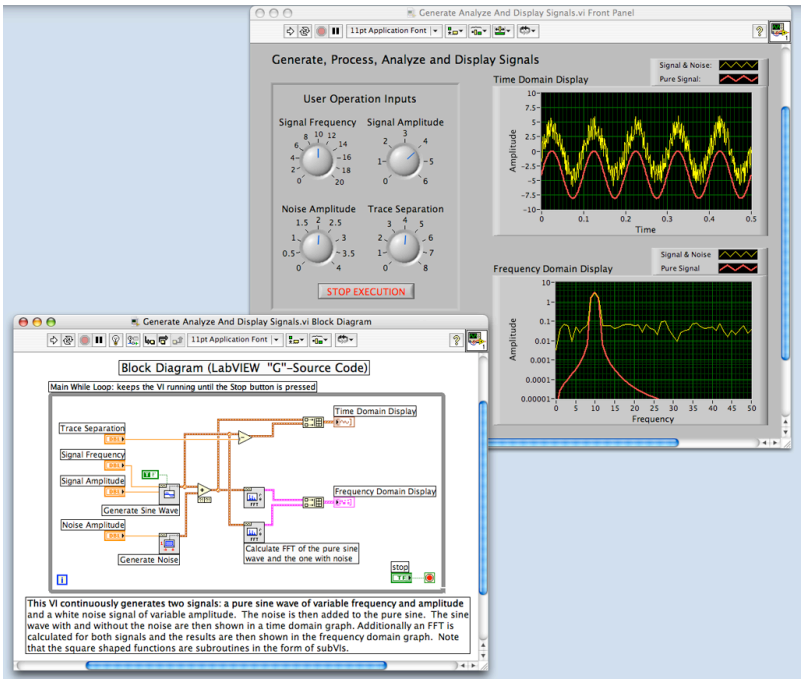


Figure 11 LabVIEW

G is a dataflow programming, where each node can execute as soon as all the needed input is available; this is characteristic that defines the execution as data

driven. Since each node can be executed when his input data is available, this might be the case for multiple nodes simultaneously, giving the language parallel execution inherently by design.

LabVIEW is also used has a seed for other visual programming environments and languages like Lego's Mindstorms NXT NXT-G, which is heavily based in LabVIEW. In NXT-G, since the target changed completely from industrial environment to an educational and hobbyist environment, the GUI was re-designed to be more users friendly and less powerful in order to reduce its complexity.

3.3.3.2 Microsoft Visual Programming Language

Like LabVIEW, Microsoft's VPL is also a dataflow visual programming language by definition and both share some similarities. The big difference between these two dataflow languages are their target usage, VPL was specially designed for the Microsoft Robotics Studio, although in the future it may have potential for other usages, its current version comes bundled in the Robotics Studio and features functionalities specific for robotic simulation and development.

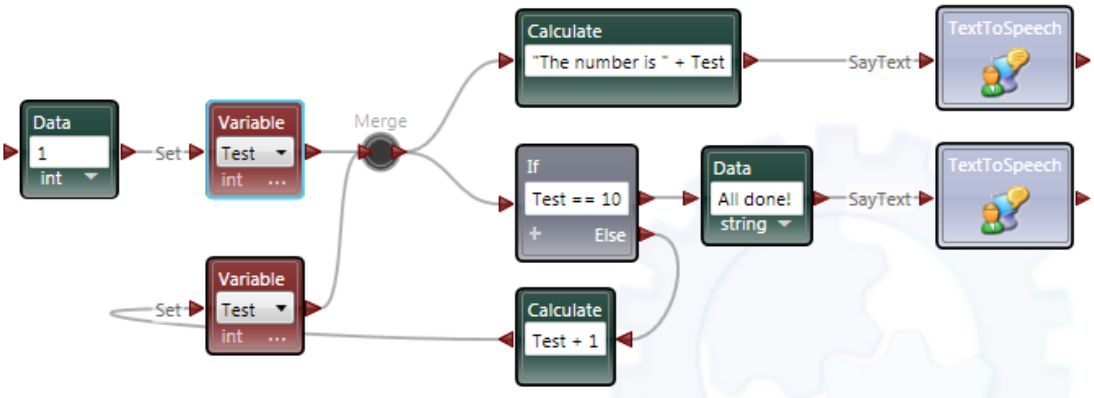


Figure 12 Microsoft VPL

Since VPL is a dataflow language a node is executed when all the needed input data is available to him. Nodes are connected to each other where each connection represents the data flow between the two connected nodes. Each node may have multiple input pins and multiple output pins. There are two types of output pins: result output or notification output.

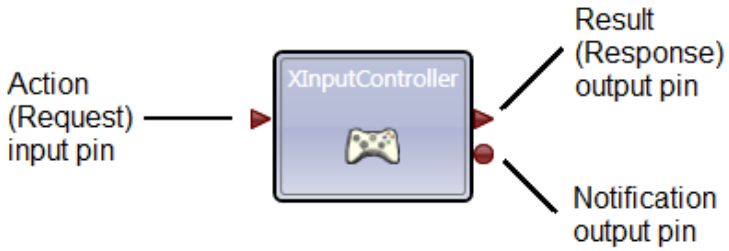


Figure 13 Microsoft VPL's Node

Result output pin is from where the processed is sent to the next connected node after a specific incoming action occurred or received, notification output can send data without the need of first received a specific event or data.

3.3.3.3 Quartz Composer

Quartz Composer is a node-based visual programming language provided bundled with the Xcode development environment for processing and rendering graphical data. This language has a very precise purpose which is manipulation of graphical data.

All the available nodes and data types (Table 1) were designed for image manipulation.

Table 1 Quartz Composer data types

Data type	Description
Boolean	Boolean value
Index	Positive integer between 0 and 2147483647
Number	Double precision floating point number
String	Unicode string
Color	RGBA or CMYK quartet, or a Gray scale value
Image	2D image of arbitrary (possibly infinite) dimensions
Structure	named or ordered collection of objects, including

	nested structures
Virtual	Polymorphic, any
Mesh	collection of vertices, and per-vertex normals, texture coordinates, and colors in 3-space
Interaction	valueless type used to associate user input with user-interactive elements of the composition

Quartz Composer is a language with a very well defined purpose. Its characteristics need to be analyzed in its environment, which is image manipulation. Taking this in account it's a powerful language but lacking functionalities from a complete generic visual programming language.

4 Botbeans

Botbeans was developed using the Netbeans Platform, this platform is a rich client platform developed by Sun now Oracle which is known as the core code-base of the Netbeans IDE.

Using a modular architecture inherited from the platform allowed to achieve a highly modular design where each one of the main functionalities was separated in different modules allowing future growth. Scalability is important in a learning tool since it allows involving its users into the development process of the application.

Group of users like schools can develop themselves new modules for the tool allowing the tool to support new functionalities without affecting the core of the application. This is why all the source-code is going to be made available to its users licensed by the GNU General Public License making sure that it and all the derivative work will always stays open.

4.1 Rich Client Platform development

A Rich Client Platform is a software development platform already with pre-developed generic components. These components can and should be used during the development of an application in order to cut development time and enrich the application by using already powerful modules for specific functions.

The final application will inherit the platform's characteristics, like portability, installer, update manager and more. Each one of these characteristics is usually a module or component that is available for the final developer.

4.1.1 RCP's

There are three main open-source RCP's for Java: Spring, Netbeans and Eclipse. Spring was never an option for Botbeans since Spring's main focus is J2EE development, leaving Eclipse and Netbeans as the two available options for Botbeans.

Eclipse and Netbeans RCPs are very similar in many ways, but there are two main differences related to the module system and user interface framework/toolkit (Table 2).

Table 2 RCP comparison

	NetBeans RCP	Eclipse RCP
UI Toolkit	Swing	SWT
UI Design	Matisse GUI Builder	Not supported
Module System	OSGi or Netbeans module system	OSGi

Eclipse uses SWT as the UI toolkit, SWT uses native bindings in order to look and feel like a native application, but this causes a big problem. Using native bindings requires SWT to have a native library for each target platform, the problem is that some functions aren't available on some platforms or may behave differently, this means that a GUI that behave correctly in Windows may fail in Mac or Linux, trashing Java's philosophy of interoperability.

Netbeans's own module system is a lot simpler when comparing it to OSGi based system of Eclipse, although using Netbeans keeps the door open for future OSGi usage since it also supports it.

Due to the SWT problems and not being a standard toolkit and the availability of two different module systems in Netbeans RCP to choose from, the choice was in favor of the last.

4.1.2 Advantages

The classroom environment is hazardous to any software because there is, more than often, an intensive unpredicted use of it. Using an RCP solved some of the potential stability problems related to this harsh environment, by using a RCP some of the core code base has already been heavily tested by the platform developer, giving a good base start for the final application.

A learning/teaching tool has to be good looking but simultaneously have a simple GUI. Since the RCP already has a lot of the GUI related functionality

implemented, functionalities like: drag and drop, window management, action engine and others. This allowed during the development of Botbeans to focus the development effort in key specific features of Botbeans since all the IDE standard generic functionalities, like save and load system, were already semi-available in the platform, allowing to end up with more functionality with less code.

As explained in the next chapter, modular architecture is a key feature in Botbeans, modular design in inherent to the RCP concept so it was an obviously choice to use an RCP as base to Botbeans.

4.2 Architecture

The main strategy while defining the architecture for Botbeans was a modular design. From the beginning, since a RCP was used in the development and a modular architecture is inherent to all RCP's, modules handling was supported and essential for the platform to work correctly.

The obvious next step was to define how to create added value with the modular framework contained in the platform. Modularity is a key feature in a learning tool, its important because the process of learning how to program is also a modular process. It's modular because it can be teach or learned in individual and contained modules where each module has its own information and added value to the final result which is learn how to program or acquire abstract/algorithmic thinking.

Each one of these modules can have requirements that usually are other modules, this also happens in the learning process. For example it is hard for a student to learn the concept of a structure without comprehending the concept of basic and native data types, since these are the ones that together form a structure. This idea was used in the development of the entire prototype where only the functionalities and information that the student needs is shown, nothing more nothing less, avoiding information overload that usually happens when a student uses an industrial programming tool.

Another two important characteristics of using a modular design are error/bug containment and mass selective updating of tool components.

Error containment is important, a learning tool has one of the worst types of users an application can have. Students give bad users because they are users without any education in the application and more important in the applications operation area, which is programming in this case.

Users of a programming learning tool will do things that their developers never thought a user could or wanted to do. Besides all this characteristics about their users, when a bug is found and an error occurs this error must be contained or else the student will be distracted by it and more important the tool can lose credibility resulting in a loss of confidence by the student and consequently in the tools results.

For example when the student do an error in his programming he will have always the hypotheses of a “tool bug”, when in reality it was an error in his implementation, demoting him into digging and debugging his implementation and from learning from his error without the teacher interaction, which had the potential to be a lot more rewarding.

4.2.1 Modules

Each key feature in Botbeans was implemented into a separated module: Bot Blocks for expression builder, Bot Shapes main visual diagrammatic programming language, Bot Control robot communication abstraction layer, Bot Common utilities module and finally external libraries modules as libraries wrappers.

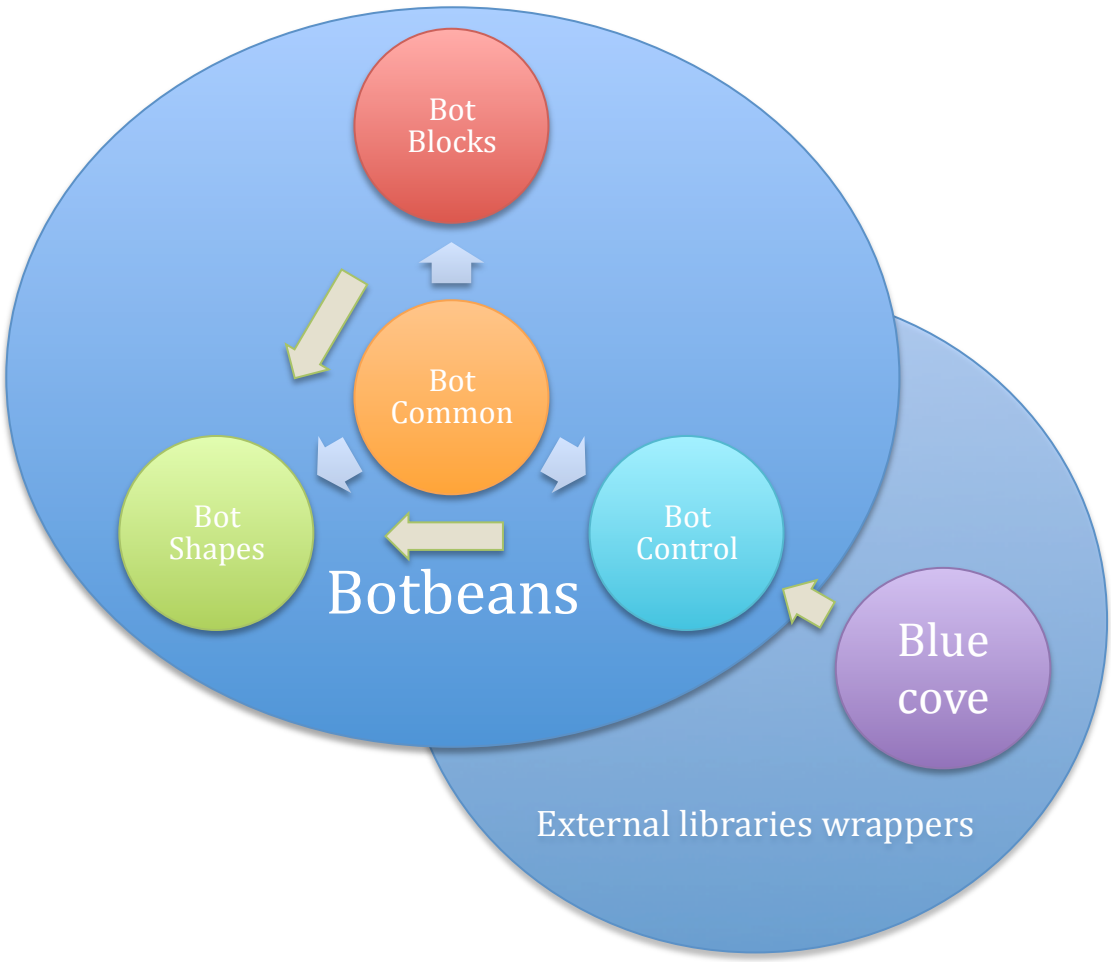


Figure 14 Modules architecture

Each module has his distinct function and they are only interconnected in the form of library/module dependency between each other.

4.2.1.1 Bot Blocks

In Open Blocks, the language definition is defined in a static XML file, this was problem since the available blocks for the expression builder depended on the information the student already had in his diagrammatic implementation.

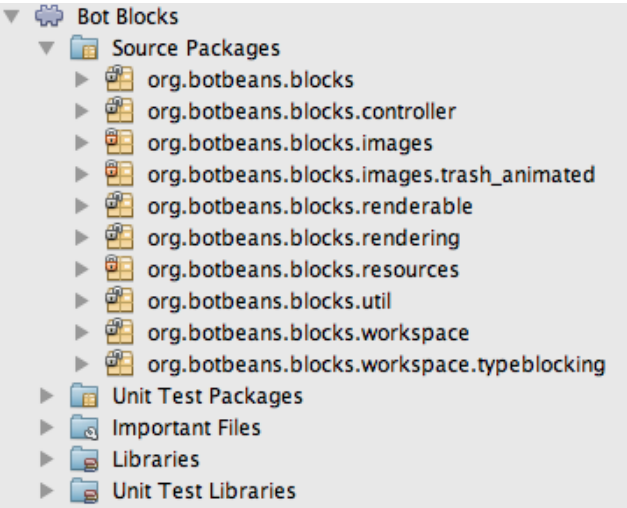


Figure 15 Bot Blocks module

For example, if a student used a variable named "a" in the diagrammatic language, a block representing that variable needs to be made available inside the expression builder in order for the student to be able to use it in his expressions. In order to accomplish this the language definition loading system was modified in order to allow to inject new definition, in addition to the definition are in the base XML file, at running time.

Open Blocks relied on on-screen rendering for some parts of their frontend, this was a problem for Botbeans since off-screen rendering is needed for some functionalities, like open and save projects with both visual representations. A rudimentary off-screen system was in order to be able to draw blocks representations on Botbeans without the need to load the entire Open Blocks framework and GUI.

Unfortunately the Open Block's source code was not refactored, all the source code needed to be refactored into a organized hierarchy of packages to comply with the platform, although all the existing author information was kept.

4.2.1.2 Bot Common

Since there cannot be circular dependency between modules in order to avoid dependecy looping, the solution is to incorporate all the common parts that multiple modules need in one common module. All the needed modules will then use the comon module as dependecy, improving the design and avoiding depency loops as can be seen in Figure 14.

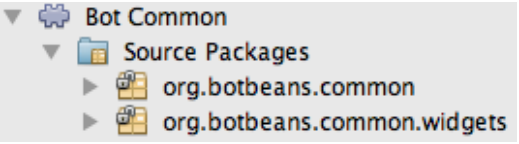


Figure 16 Bot Common module

In Botbeans this module contains utilities related to file operations that can be used in other modules and more important data models for the visual language. These models needed to circulate across the Bot Shapes and Bot Blocks modules since the expressions built inside the expression builder need to be displayed in main visual language inside the matching node.

4.2.1.3 Bot Control

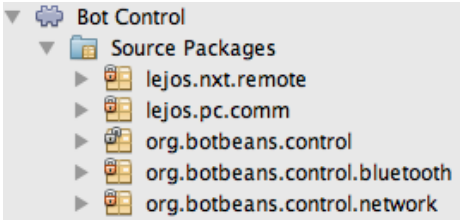


Figure 17 Bot Control module

All the communications architecture described in the communications chapter was implemented in this module. In current version of Botbeans, only one robotic platform is implemented, although it can support infinite number of platforms. Support for more platforms should be implemented in separate modules using this module as dependency.

4.2.1.4 Bot Shapes

This block was the first to be developed. It's in it that all the diagrammatic graphical language and its palette was implemented using the Netbeans's visual library, which is one of the dependencies for this module.

Since the diagrammatic graphical language is the main language for Botbeans, some of the core functions of the IDE were also implemented in this module by necessity. Functions like dialogs and actions for the entire graphical user interface

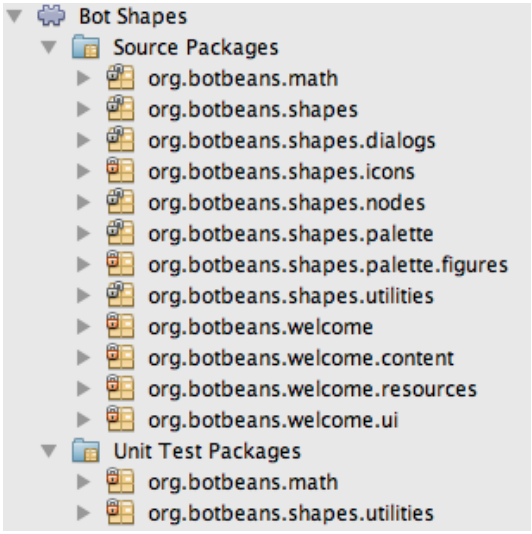


Figure 18 Bot shapes module

Execution is also included in this important module since it is also done on top of the diagrammatic graphical language. This module is the core of Botbeans it will need other modules as dependency, like it can be seen in Figure 14.

4.2.1.4.1 Components

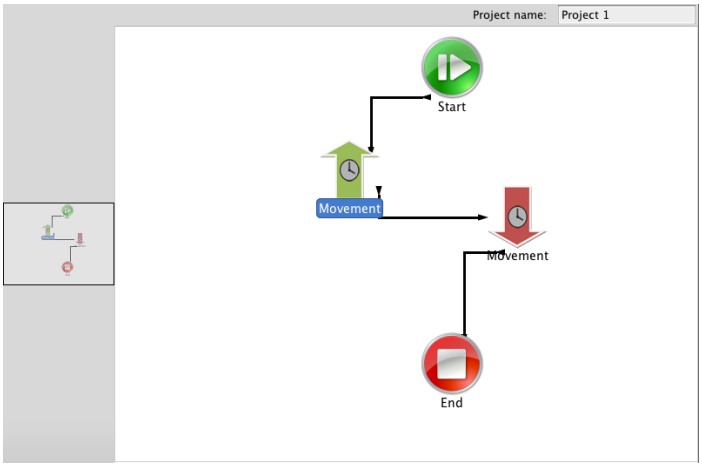


Figure 19 Botbeans canvas

ShapeTopComponent – This component is key for the diagrammatic visual language, this is where all the language's frontend is implemented. Frontend for a visual language is not a completely brainless software component has in other applications, it must accept events from the drag and drop system and render all the visual nodes.

There can be more than one ShapeTopComponent instantiated at the same time allowing opening multiple projects using a multi-tabbed interface. This functionality is very important since it allows for the student to work in multiple exercises inside the classroom.

The logic for the visual language is centralized in GraphSceneImpl, it is in here that every action in the language (adding a connection, removing a node, etc.) is validated and inserted into the according data models.

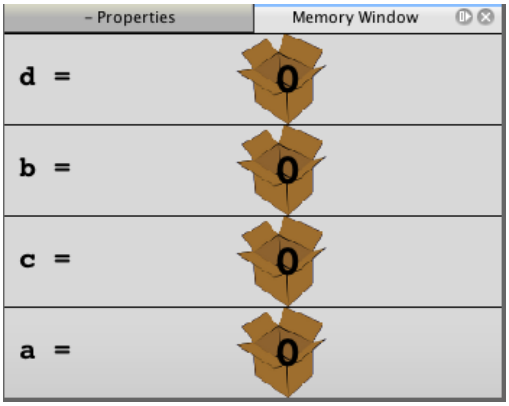


Figure 20 Botbeans memory display

MemoryTopComponent –In earlier tests with Botbeans it was obvious there was a problem with memory display, student couldn't monitor the content of all the different variables he was using in his algorithm.

Since the student is able to pause and resume the execution, by using this component, he can monitor the contents of each variable. This component was developed to support an infinite number of variables, when needed each variable's screen real state will be adjusted in order to accommodate an increased number of variables.

4.2.1.5 External libraries wrappers

Netbeans rich client platform like almost all RCPs uses strict modular development architecture; because of this they do not support a classic jar dependency, supporting only modular dependency.

In order to work around this and at the same time comply with the architecture, the official solution is to wrap the jar library in a wrapper module, which will contain the library. This wrapper module makes the classes from the library available at runtime to the module and modules that use this module.

The most important wrapper in Botbeans is the module wrapper for the Bluecove library. Bluecove is a java Bluetooth implementation that interfaces with the operating system's Bluetooth stack. It is through this wrapper that Botbeans communicates with robots that use Bluetooth.

4.3 Graphical language

4.3.1 Hybrid solution

Previously we saw that in node based languages users have a good overview of the flow of logic although they are usually forced to use textual mathematical and logical expressions, in the other hand block based languages have a good data type system which is based on block shapes avoiding expressions with a textual representation.

A hybrid solution was developed in order to get the best of both worlds. In this hybrid language all textual expressions in the diagrammatic language were swapped by expressions built using the expression builder, which uses a block-based language.

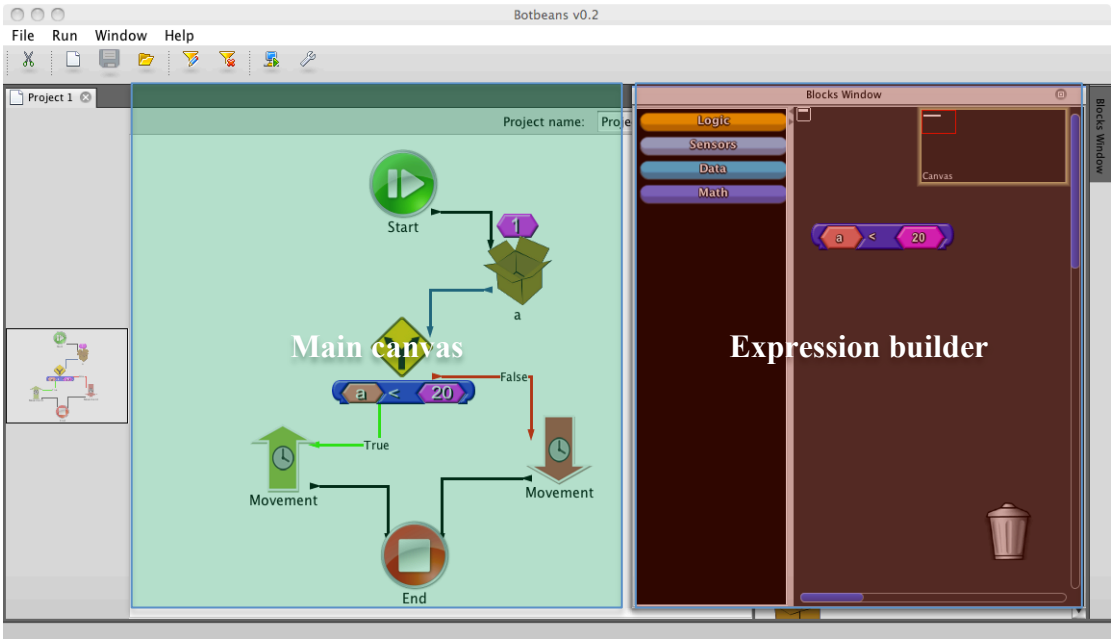


Figure 21 Expression builder GUI

When a user double-clicks an expression the expression builder slides from the left side of Botbeans's main window (Figure 21) allowing the user to see his algorithm in the diagrammatic form and at the same time build the wanted expression using the block based language. When the user clicks the main diagrammatic drawing area the expression builder transfer de expression to the diagram and closes by itself in order to avoid information overload to the user.

4.3.2 Definition

Nodes were divided into multiple categories in order to keep the palette clean and logically organized (Table 3). There are two main categories: “Control” for output and “Sensors” for input, it is using the nodes inside these categories that the user will be able to interact with the robotic platform.

Table 3 Node's Categories

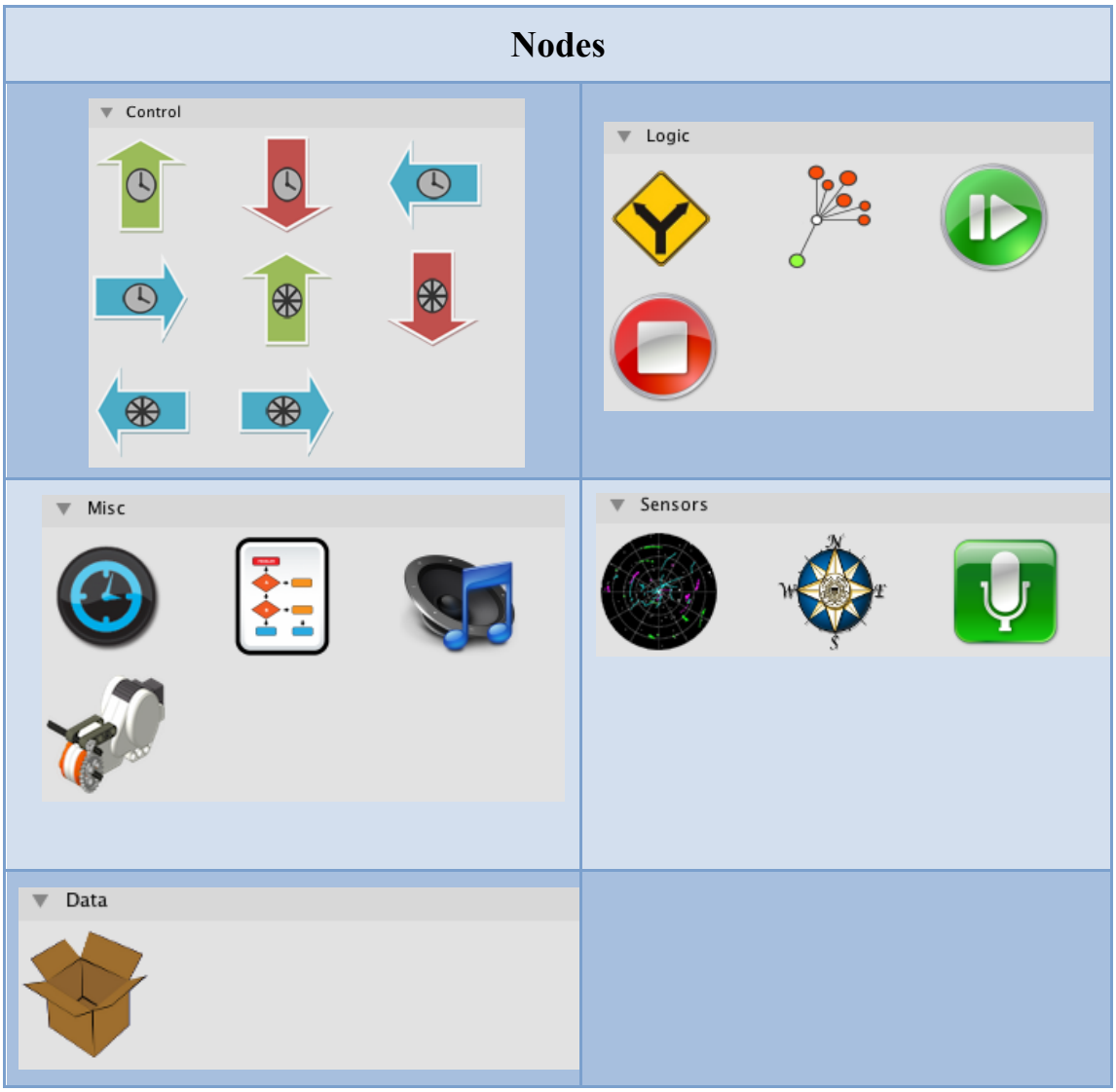
Category	Description
Control	Nodes to trigger robot’s actuators making it move.
Sensors	Sensor input, each sensor have his own node.
Logic	Nodes that affect the logic flow in the algorithm. Ex: decision and union nodes.
Data	Variable definition and attribution.
Misc	Timer, speaker and other miscellaneous nodes.
Custom blocks	Custom blocks, implemented by the user.

These two main categories used to interact with the robot are not enough to define a programming language, so all the basic control structures that affect the logic flow are in the “Logic” category. Ex. decision, union, start and end nodes.

Data category is where the nodes related with data manipulation will reside, in current version of Botbeans only one node exists since the same node is used for variable definition and attribution.

Finally the miscellaneous nodes like: timer or sleep, speaker, auxiliary actuator and other, are in the “Misc” category.

Table 4 Botbeans's nodes



4.3.2.1 Control

Figure 22 Control time based nodes

Figure 23 Control distance based nodes

There are two types of control nodes: distance based and time based. The control nodes with the clock icon are time based, what this means is that the robot will move for a specified time. Distance based nodes will move for a specified distance.

4.3.2.2 Logic

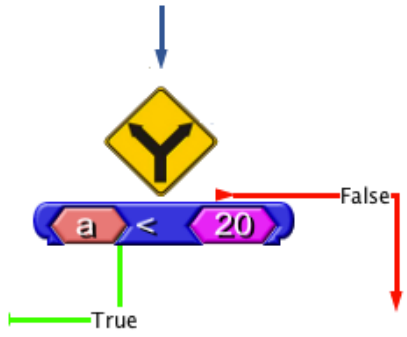


Figure 24 Decision node

Decision node is the decision control structure also known as "if" in almost all the textual programming languages. This node will obligatory need a logic expression to work, this logic expression when double clicked will open the expression builder in order for the user to be able to edit it.

There are two types of outgoing connections from this node, one if the expression returns true and another if the expression return false, only one connection will be executed since the expression can't be true and false at the same time.

Union node is one of the simplest nodes in the language, its objective is to keep the logic flow organized and without any hack or shortcuts. During the specification of the language it was defined that to aggregate multiple connections a special node was needed in order to avoid connecting multiple connections in every node, which could increase the complexity if the problem was big enough.

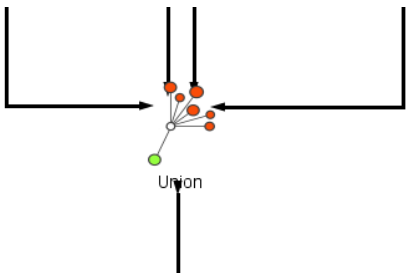


Figure 25 Union node

Figure 26 Start and end nodes

Start node define where the execution will start if the user don't specify a different starting point. User has two types of execution normal and debug, in normal mode the execution will always start in the "start node" this is why this node is obligatory. In debug mode the execution will start in the selected node allowing the user to debug a specific part of his implementation. The finish node is just for organization since the execution will always end when it doesn't have where to go.

4.3.2.3 Data

Variable node is a node that represents a variable in the language; this node will have a name, which will be the variable name, and a value, which will be attributed to the variable with the node's name. This node is used in variable definition but also in attribution, since if the variable was never used before Botbeans will implicitly define it in memory.

Figure 27 Variable node

4.3.2.4 Miscellaneous

Figure 28 Sleep node

Timer node is used when the user wants to provoke a delay in the execution, when this node is executed the robot will halt and wait the specified time, in milliseconds.

It is similar to the sleep function from other textual programming languages.

In order to give to users the possibility of using another of their senses, audition, a node was implemented that allows the robot to emit a sound.

The speaker node has two parameters: duration and frequency allowing the user to specify for how long the robot will emit the sound in the specified frequency. The Robot will wait until the sound finished being emitted and only then will send the response to the control module, which will send the next instruction.

Figure 29 Speaker node

4.3.2.5 Sensors

This category is where all the nodes responsible for the input from the robot are. All the sensor nodes work the same way in terms of functionality, the only thing that changes between them is what each one returns.

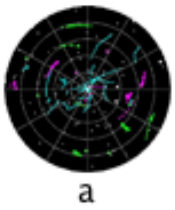


Figure 30 Distance sensor

Saves in the specified variable an integer with the distance read in centimeters.

Returns to the specified variable an integer with the direction in degrees.

Figure 31 Direction sensor

Figure 32 Sound sensor

Saves in the specified variable an integer with the read sound level in decibels.

4.3.2.6 Node's inputs and outputs

Table 5 Nodes characteristics

Node	Nº Inputs	Nº Outputs
Control nodes	1	1
Sensor nodes	1	1
Decision	1	2
Union	1+	1
Start	0	1
End	1+	0
Variable	1	1
Sleep	1	1
Speaker	1	1
Aux actuator	1	1
Custom blocks	1	1

Each node has a limited number of inputs and outputs (Table 5), when a user tries to make a connection between two nodes, the current number of inputs and outputs of the two nodes is verified if one of the limits has been reached it means the connection is invalid and the user is doing a logic error. This is how the logic flow inside the algorithm is validated at building time.

4.3.2.7 Custom Blocks

Custom blocks are nodes that contain multiple nodes inside them. Using custom blocks the user can divide his problem into smaller problems and implement the solution for each one of these problems separately. In the end to create the final solution he will then just need to use the previous implemented modules of the solution and integrate them all. This way the concept of subroutine or function is given naturally to the user without extra complexity.

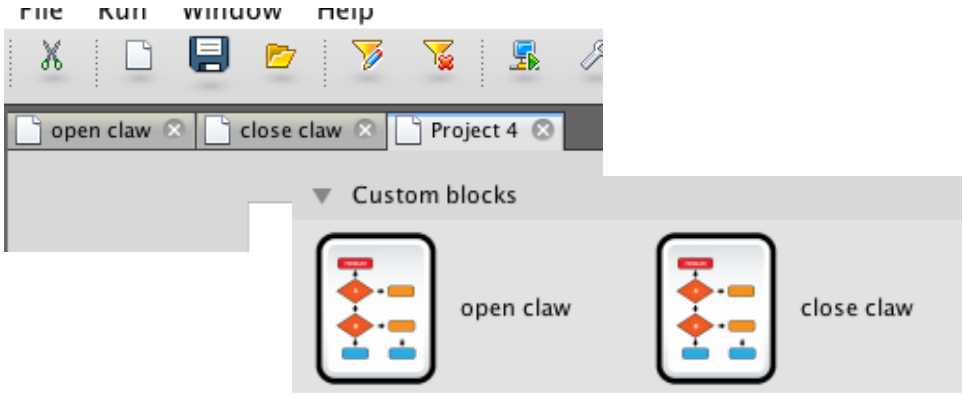


Figure 33 Botbeans custom blocks

To create a custom block the user just need to give a name to the current project he is working on, then this project will show in the palette of all the other projects has a custom block in the "Custom blocks" category. User will then be able to drag this block into his already existing implementation and use it like a function in the typical programming languages, with the difference that these custom blocks don't have parameters nor return data.

4.4 Tangible interface

Tangible interfaces are very often associated with learning, since they can boost the student interest, creativity and catalyze collaboration between multiple agents of the learning process (Schneider, Jermann, Zufferey, & Dillenbourg, 2010).

The computer running Botbeans GUI will only have inputs, but while a user is at the computer making adjustments, another user will naturally be at the robot monitoring his behavior in the real environment, although the user at the computer is able to see to robot of even hold it in his hands, he don't need to lose focus by multitasking between two points of interaction. The user that is monitoring and relocating the robot will give his feedback and opinion on how to debug or fix the problems that occur, and a brainstorm between these two users will naturally occur.

This collaboration advantage manifested clearly in the initial tests, where users with different profiles or skills chose where they did want to focus, computer or robot, this way they can exchange their skill set and knowledge between them, naturally leveling them to the same level.

4.4.1 Points of interaction

The initial idea of using a tangible interface in Botbeans was not only related to motivation but also to increase the number of points of interaction with the learning tool. Usually a programming learning tool is a one man experience, where a student sits at a computer and starts experiencing it alone or with a colleague, but since the computer only has one keyboard and one mouse it will end up in an uneven experience in terms of acquired knowledge.

With the implementation of a tangible interface Botbeans gains another point of interaction and this new poi has the potential to accommodate additionally more users than a computer screen.

The tangible interface used Botbeans it's in the shape of a robot, it's through this robot that the user outputs and inputs into his programming. Traditionally when someone is programming it uses the screen has output and the keyboard or mouse has input, for this to work the user needs to be able to understand the results that are being displayed in the screen, which isn't always simple for a beginner. Ex: A program return 123, does 123 the correct answer or wrong one?

By using the robot has output the user will naturally know if his implementation failed or not. Ex: Robot crashed, algorithm failed.

4.4.2 Robotic platform

In current Botbeans's version only one robotic platform is supported, which is the Lego Mindstorms NXT, although the initial development and proof of concepts were developed using a homebrew robotic platform developed at Institute Polytechnic of Tomar with the name Gualdim (Figure 34). This platform used Wi-Fi to communicate with Botbeans and subsequently to its communication layer, after initial tests with students from high school level, it was obvious that a more all around and easy to acquire platform needed to be supported by default.

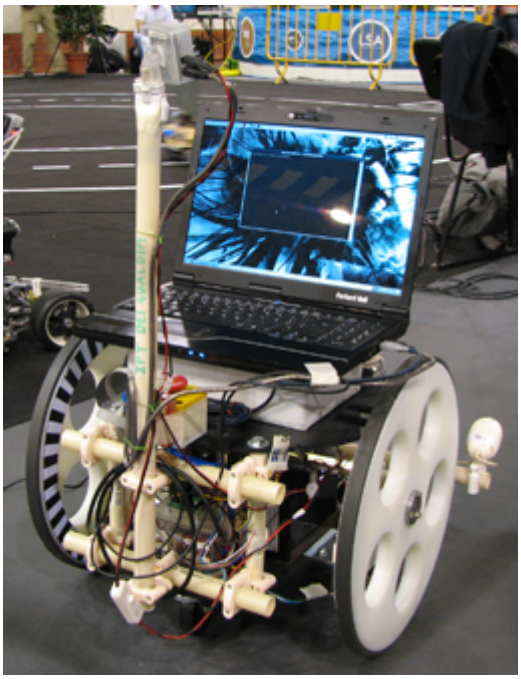


Figure 34 Gualdim

Lego Mindstorms NXT was the winner by many reasons, one of the most important was the fact that this was a platform already installed in many schools, students already lost the initial fear of interacting with it or even breaking something during their experience with it.

Another important fact is that by supporting a ready to run robotic platform, the software problems are sealed from the hardware allowing Botbeans users to focus where they want, hardware or software. Since Botbeans architecture was designed to support infinite number of robotic platforms, there can be schools or users that will develop a homebrew robotic platform using a simple microcontroller and some actuators allowing a complete experience from the low level hardware to the high level algorithm implementation in Botbeans.

4.4.2.1 Robotic platform requisites

- **Microcontroller** – where the control client will be implemented.
- **Connectivity** – connection to the computer running Botbeans.

- **2+ actuators** – Two actuators are needed for locomotion. Is advised to use encoded or stepped actuators, since is desirable that the control client is able to control the amplitude and/or speed of the actuator movement.
- **0+ Sensors** – There are no obligatory sensors.

4.5 Communications

Since a robot was used as part of the tangible interface, Botbeans needed to have a communication strategy and architecture. A few prerequisites were established since the beginning, the tool needed to be able to support more robotic platforms in the future without transmitting the complexity of this to the end user.

Since each robotic platform can have is own way of communicating, an abstract layer between the core of the prototype and communications modules was needed. This abstract layer allowed the system core to communicate in the same format all the times independently of the robotic platform used, allowing to use virtually any robotic platform without rewriting core code. This opens possibility to homebrew robotic platforms and this by itself opens doors to teaching in other areas like electronics and/or hardware, culminating in a multidisciplinary learning experience and possibly collaboration between student from different backgrounds and ages.

4.5.1 Communications architecture

The abstract communication layer was developed also as a module, which has an internal server running. This server is the responsible for receiving the data from the core of the application and relaying that data to the robot. This relaying process is not a simple pass-by since this data needs to be translated and formatted according to the robotic platform in use

Translation is needed in order to maintain a formatted data stream between the application core and the control server independently of the robotic platform in use, since the data stream is formatted according to the platform specification.

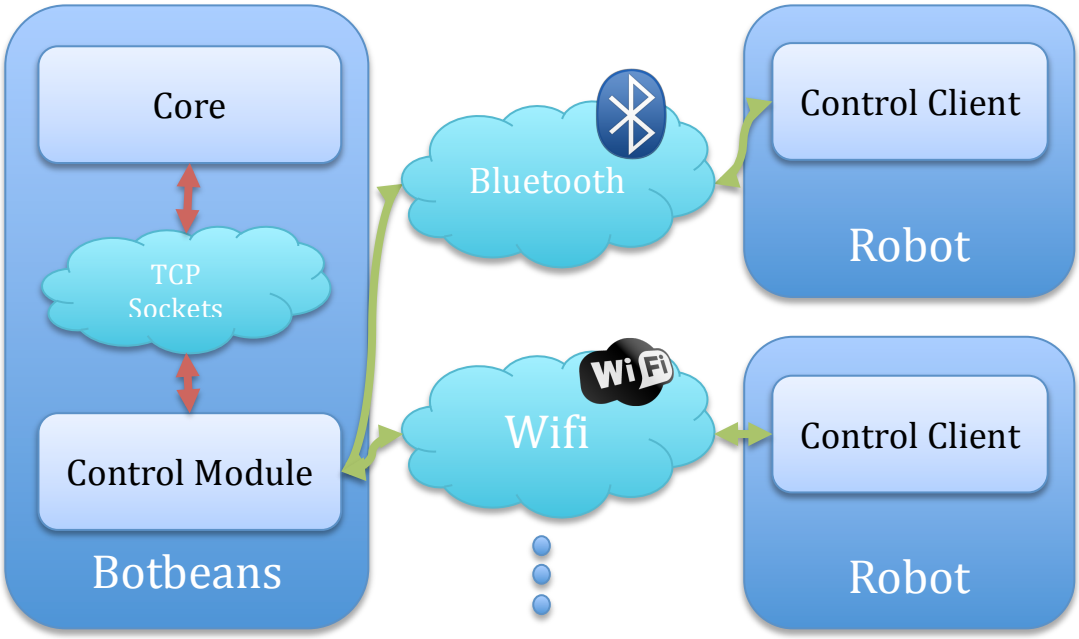


Figure 35 Communications architecture

Each robotic platform needs to run a client in order to be compatible with Botbeans, this client or "Control client" is the piece of software that implements the Botbeans protocol on the robot. It will listen for orders sent from Botbeans and answer accordingly to requests sent to it, ex. Sensor request.

This is why each robotic platform needs to have a different client implementation, since this client is implemented in the robot in its supported programming language in order to access all the actuators and sensors in the platform.

4.5.2 Protocol

Table 6 Packet structure

Packet				
Packet size	Operation	Data[0]	Data[1]	Data[n]
packet[0]	packet[1]	packet[2]	packet[3]	packet[n]

This packet structure (Table 6) is used for request messages, response messages use the same structure but without the operation field. Responses do not

need the operation field since the action that triggered the request will always wait for the robot response, working in a request/response model.

Table 7 Protocol operations

Operation	Description	Parameters	Return
1	Move distance.	Speed, Distance	0 – failure, 1- success
2	Move timed.	Speed, Time	0 – failure, 1- success
3	Rotate degrees.	Speed, Degrees	0 – failure, 1- success
4	Rotate timed.	Speed, Time	0 – failure, 1- success
15	Configuration	Ports layout	0 – failure, 1- success
16	Request distance value.		Distance in cms
17	Request compass value.		Value in degrees
18	Play tone.	Frequency, Duration	0 – failure, 1- success
19	Request microphone value.		Value in dbs
20	Move auxiliary motor.	Speed, Degrees	0 – failure, 1- success

Each operation represents a function call in the robot's control client where the operation id (Table 7) is used to determine which function is being requested from the control module. If new operations are implemented in the control module, they obligatory will need to be implemented in the client or else there's the risk of the control module requesting a function that's not supported by the control client.

4.6 Examples

During the earlier tests some examples were implemented in order to show the potential of Botbeans in different scenarios. Two examples are here described: a simple path finding algorithm and a square movement using custom blocks.

4.6.1 Simple path finding algorithm

This example was implemented in order to show a more advanced algorithm implemented in Botbeans. This example was considered advanced not because of its size, but instead because of the algorithm that is implemented which results in an interesting robot behavior.

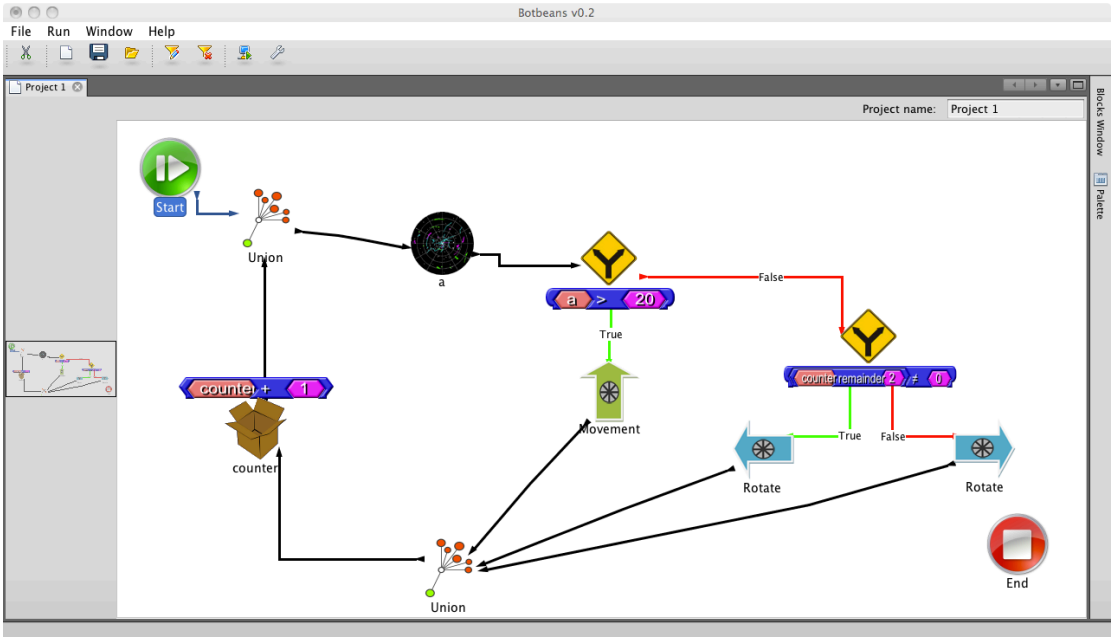


Figure 36 Simple path finding algorithm

In this example the distance from the distance sensor is read to the variable "a" if it is greater than 20 centimeters then the robot will move forward else if the counter variable has an even number the robot will turn right 90 degree else it turns left. Finally the counter variable is incremented and distance sensor read again.

This results in a behavior where the robot will try to find a new direction if an obstacle is detected closer than 20 centimeters.

4.6.2 Squares using custom blocks

The objective of this example was to demonstrate the usage of custom blocks. The initial problem was how to build an algorithm that allowed the robot to move in squares and then move two squares. A custom block for the square movement was created (Figure 37), which was then used in the main project in order to move the robot in two squares.

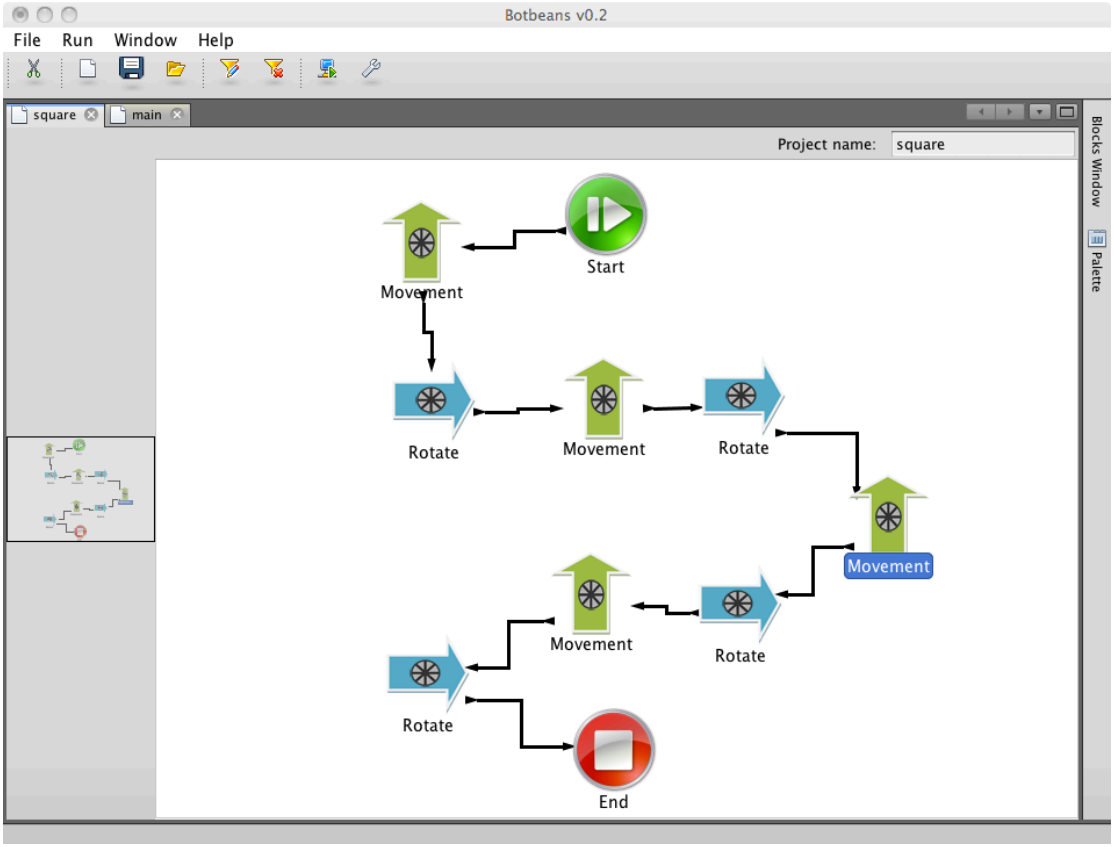


Figure 37 Square custom block

This algorithm (Figure 37) makes the robot move in a square. An custom block was created from this project since it have the project name field on the top right corner defined, creating a custom block called “square” which can be used in other projects.

In the next project (Figure 38) it can be seen the “square” custom block being used in a simple example, this custom block was dragged from the custom blocks category.

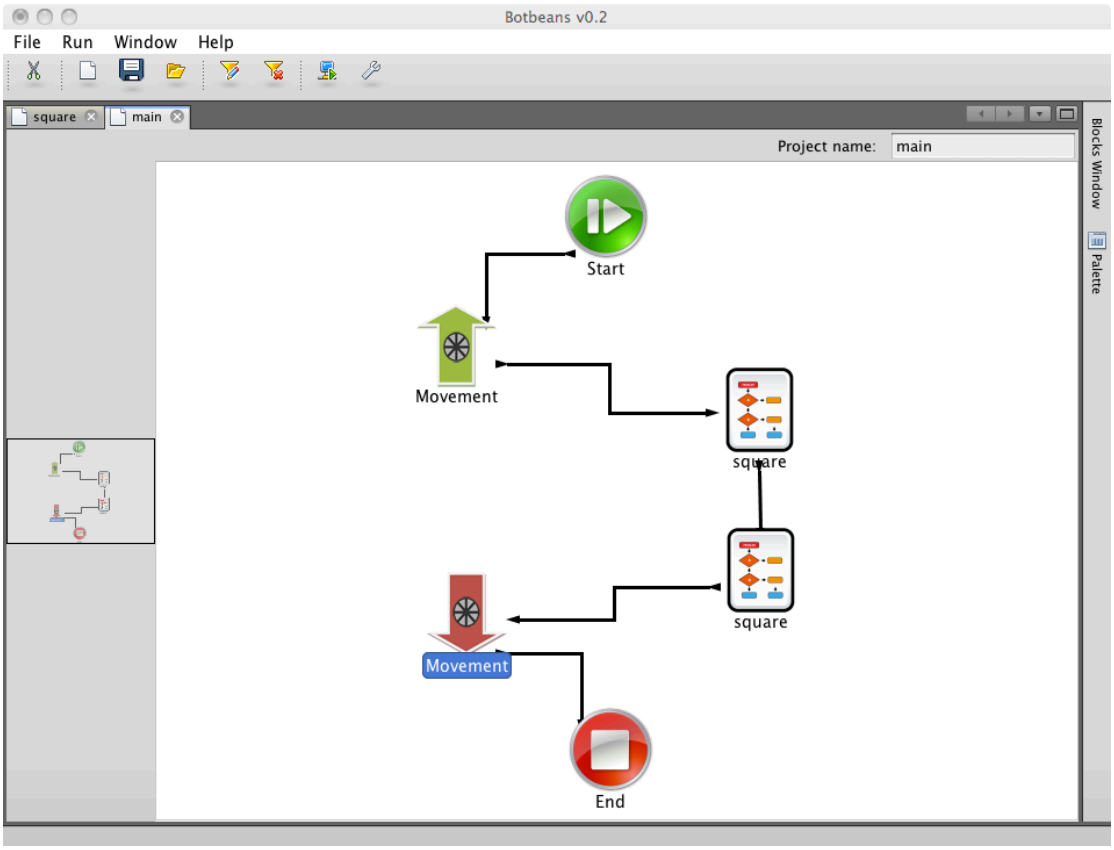


Figure 38 Usage of square custom block

This project (Figure 38) uses the, previously created, square custom block (Figure 37). When executing this project the robot will move forward 10 centimeters, draw two squares with his movement and finally move backward.

5 Results

5.1 Tool Comparison

This tool comparison was made in order to evaluate the key features of which one of the learning tools in Table 8. A learning tool is completely different from an industrial development tool, having more functionalities can be a good thing but also a bad thing if the user don't use all these functionalities and ends up with an interface overloaded.

Table 8 Tool Comparison

Tool Name	Graphical Language		Textual Language		Correlation between languages	Object Oriented
	Diagrammatic representation	Block representation	Static	Dynamic		
BlueJ	X		X		X	X
Portugol	X			X	X	
Alice	X		X		X	X
Scratch		X				X
Greenfoot	X		X		X	X
NXT-G	X					
Botbeans	X	X				

The key features analyzed in this comparison were: presence of graphical language, presence of textual language, correlation between languages and if it is object oriented.

If the tool featured a graphical language the type of visual representation used was verified, diagrammatic or block base presentation. Like in visual programming

languages, textual programming languages has two main types of syntax: static and dynamic.

Textual programming languages that feature a static syntax have a fixed syntax where the representation of each programming structure never changes, in languages with a dynamic syntax each programming structure textual representation may dynamically change.

The only tool that featured a dynamic textual language was Portugol since it allows its users to adjust the language syntax to their mother language. For example if a user with English has his mother language uses a decision control structure will type an “if”, if a Portuguese user uses the same structure will need to type “se” which is “if” translated to Portuguese.

Since the main objective of Botbeans is to help users that never experienced programming, this fact can be observed in the comparison table (Table 8), textual language was removed and the best features of both graphical languages types were merged into one hybrid graphical language. Botbeans doesn’t need have correlation between textual and graphical languages since it does not feature a textual language.

5.2 Earlier tests

In 2010 during the “Festa da Ciência, Cultura e Tecnologia”, an annual science fair at Institute Polytechnic of Tomar, an experience was built (Figure 39 FCCT 2011 Students collaborating) where some of the students visiting the science fair were invited in trying to play with Botbeans.

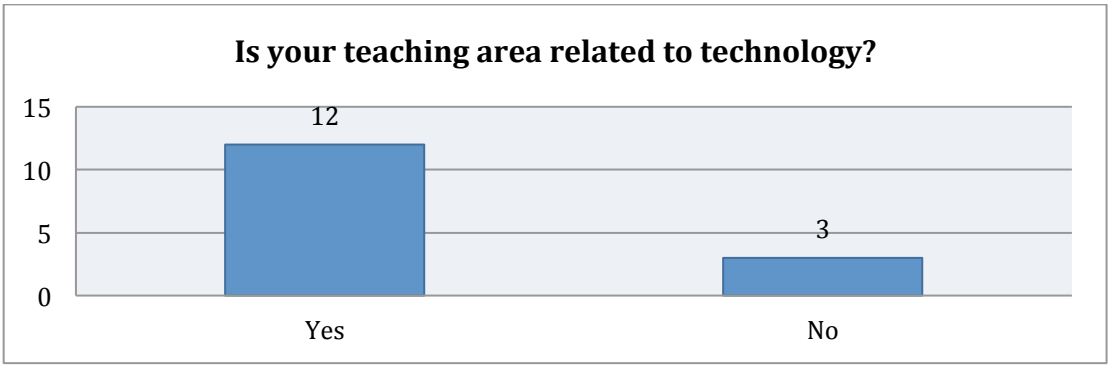
The experience consisted in a table where the robot moved and a laptop projecting the Botbeans GUI to the wall in front of the table, students could interact with the robot directly or by change the layout in table by moving the available obstacles around. A keyboard and mouse was on top of the table in order to interact with Botbeans GUI, creating to main points of interaction: robot and computer GUI.

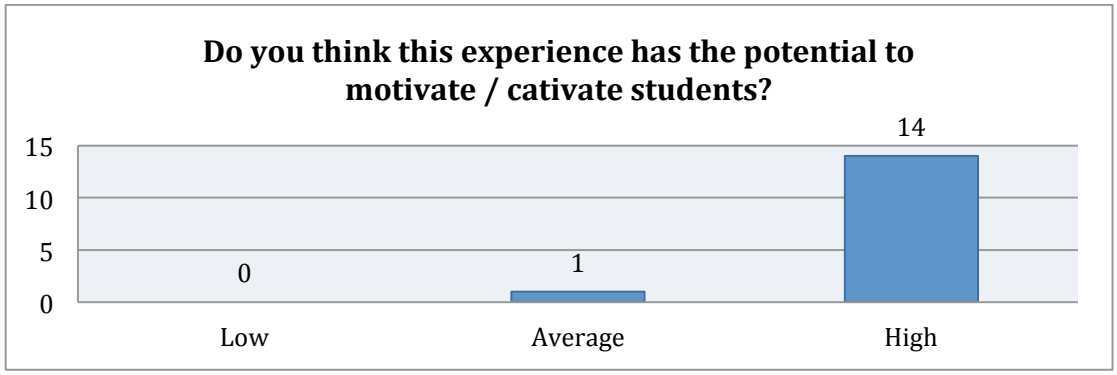
Figure 39 FCCT 2011 Students collaborating

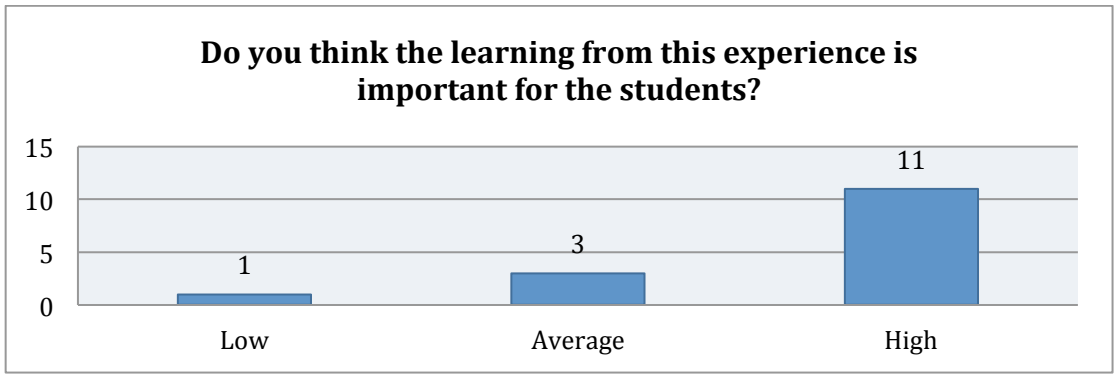
An immediate collaborative behavior was evident between the visitors. Very often when a student started executing the algorithm that he had just implemented, another student entered the scene giving suggestions to his colleague on how to improve the algorithm or try something different. The robot was the big motivator since during the execution one of the students was positioning/monitoring the robot and the other one were at the computer.

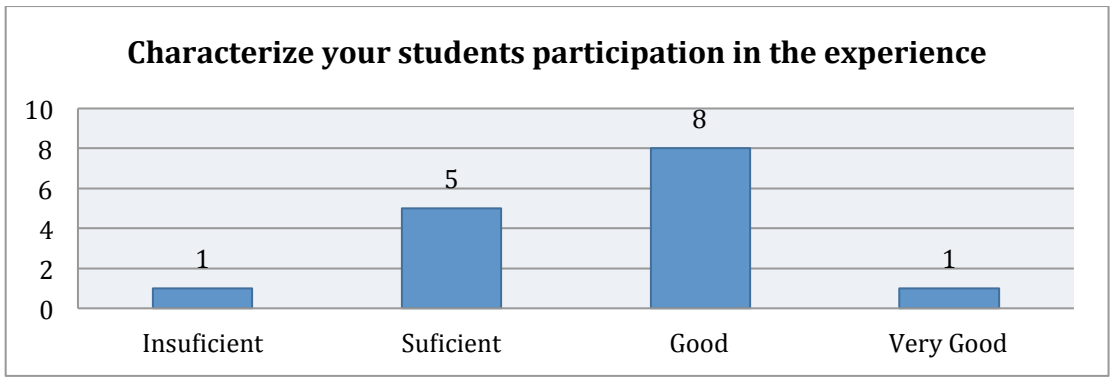
In 2011, the experience was again displayed at FCCT with a new version of Botbeans. This year an inquiry was made for the teachers that accompanied the students in the experience.

5.2.1 Inquiry









Each group of students was accompanied by one of their teacher, in the end of the experience the inquiry was given to the teacher while the students were freely playing with Botbeans.

A total of fifteen groups attended the experience during the three days, although it was a small and generic inquiry since not all the teachers taught topics related to computing, the results say that from the teacher's point of view the tool has a lot of potential in motivating students to this topic.

Participation was where the results were less positive. It some groups of students that attended the experience was hard to convince them to give the first step and embrace the experience hands on, some students were afraid that something would break in their hands other just afraid of doing something wrong and older students could see it since the experience was in an open space.

If the experience was in an environment that the student was already familiar with, like in their school with their own teachers. The no longer had strange subjects and/or objects in the experience allowing the students to relax and feel more confortable.

6 Conclusion

The main goal of Botbeans is to decrease the learning curve of computer programming. It helps people that never had any previously contact with programming to acquire the needed background to progress in more advanced topics.

Reaching to advanced programming topics like being object oriented was never the objective of Botbeans, these topics require previously knowledge in basic programing topics, like control structures and data types, in order to fully understand the concept of objects. Since Botbeans is designed to be an introduction to user's first contact with programming it will focus on algorithms and abstract thinking rather than implementation. This is the philosophy behind Botbeans.

If Botbeans featured a textual language it could help the jump from Botbeans to another industrial programming language, but it would drastically increase the complexity for users without any previously experience, instead Botbeans approach is to train the user's algorithmic and abstract thinking, leaving the syntax/implementation/programming training to the next level learning tools.

The development of a new hybrid visual language allowed removing almost all the textual representation from the programming language, removing one of the biggest problems for beginners. As the initial research in visual programming languages revealed, block representation is more intuitive for the user allowing him to easily understand the mathematical and logical expressions, which are key in computer programming.

On the other side keeping a diagrammatic, node-based visual programming language allows Botbeans to easily represent the logic and data flow inside an algorithm to the user, enabling him to easier comprehend the basic control structures in programming.

Developing Botbeans on top of a rich client platform allowed with less time to end up with more functionality and more important with a modular architecture inherited from the platform.

After the initial tests with real subjects one factor was clearly obvious, which is the environment where Botbeans is used can alter the user experience. In order to have a complete experience users must feel comfortable with the environment where they are experiencing, this is more important in Botbeans than in other learning tool because of the tangible characteristic of Botbeans.

Incorporating a tangible component into the user experience allows Botbeans to bring the user experience to the real world, consequently bringing the characteristics of the real world into the user experience. This is why is advised that the user already is comfortable with his surroundings.

From the initial tests, Botbeans has a lot of potential in lowering the threshold from learning computer programming. It allows presenting what is programming while simultaneously keeping users motivation high.

A motivated user will more easily make an effort to pass the difficulties associated with gaining the pre-requisites or basic topics needed for programming.

6.1 Future work

Like everything related to computing nowadays Botbeans is not a finished product and there are still things to improve and research. Although the initial tests have given positive results, these tests revealed that the environment where the tool is experienced might influence the results. In order to validate this theory, future tests should be conducted in friendlier and known environments to the subjects and for a longer period.

Evaluating and quantifying the success of a learning process can be a daunting task, each group of subjects experience the task of learning a new topic in different ways and more importantly experience it at different speeds. In order to avoid this heterogeneous characteristic future tests should take in account the subject's knowledge about the topic previously and after tests.

Measuring the knowledge acquired during the experience will allow better evaluation of results. In order to accomplish this each subject should do a generic test that doesn't request any specific pre-requisite about the target topic and finally a test

after the experience is done. Comparing these two results should give a better answer about how did the experience affected the subject's knowledge about the target topic.

From the implementation point of view there are still possible improvements, memory display still needs some research on how to show in an interactive and casual format the data contained in memory. Current memory display although simple and without any interface complexity may have scalability problems if a big quantity of variables are used.

Botbeans in order to be friendly with e-learning methodologies needs to be able to integrate with existing systems like Moodle. Integrating Botbeans with other infrastructure learning tools opens possibilities to increase its collaborative characteristic by using a system of problems and solutions repositories, these repositories could allow its users to collaborate between each other or just share ideas and solutions and/or exercises.

Like in any other piece of software evolution never stops since software needs to follow its users and their needs, which will change from time to time. Botbeans will also need to be maintained and improved in order to be an updated and functional learning tool.

7 Bibliography

Dias, Pedro, & Oliveira, S. (2010). Botbeans: a new educational visual programming tool with tangible results. *ACM SIGDOC European Chapter/ Eurosigdoc Workshop on Open Source and Design of Communication*, OSDOC '10 (pp. 43–44). New York, NY, USA: ACM. doi:10.1145/1936755.1936768

Horn, M. S., Solovey, E. T., Crouser, R. J., & Jacob, R. J. K. (2009). Comparing the use of tangible and graphical programming languages for informal science education. *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09 (pp. 975–984). New York, NY, USA: ACM. doi:10.1145/1518701.1518851

Jacob, R. J. K., Girouard, A., Hirshfield, L. M., Horn, M. S., Shaer, O., Solovey, E. T., & Zigelbaum, J. (2008). Reality-based interaction: a framework for post-WIMP interfaces. *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08 (pp. 201–210). New York, NY, USA: ACM. doi:10.1145/1357054.1357089

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, *37*, 83–137. doi:10.1145/1089733.1089734

Manso, A., Marques, C. G., & Dias, P. (2010). Portugol IDE v3.x: A new environment to teach and learn computer programming. *Education Engineering (EDUCON), 2010 IEEE* (pp. 1007-1010). Presented at the

Education Engineering (EDUCON), 2010 IEEE. doi:10.1109/EDUCON.2010.5492469

O'Malley, C., & Stanton Fraser, D. (2004). Literature Review in Learning with Tangible Technologies. Retrieved February 2, 2011, from http://hal.archives-ouvertes.fr/hal-00190328/en/

Pedro Dias, & Sancho Oliveira. (2011). Meet and greet programming using graphical languages and tangible interfaces. *Information Systems and Technologies (CISTI), 2011*. Presented at the Information Systems and Technologies (CISTI), 2011.

Roque, R. V. (2007). OpenBlocks : an extendable framework for graphical block programming systems. Thesis, . Retrieved January 11, 2011, from http://mit.dspace.org/handle/1721.1/41550

Schneider, B., Jermann, P., Zufferey, G., & Dillenbourg, P. (2010). Benefits of a Tangible Interface for Collaborative Learning and Interaction. *Learning Technologies, IEEE Transactions on*, *PP*(99), 1. doi:10.1109/TLT.2010.36

Smith, B. J. (2009). Conceptual graphs as a visual programming language for teaching programming. *Visual Languages and Human-Centric Computing, 2009. VL/HCC 2009. IEEE Symposium on* (pp. 258-259). Presented at the Visual Languages and Human-Centric Computing, 2009. VL/HCC 2009. IEEE Symposium on. doi:10.1109/VLHCC.2009.5295242

Stern, T. I. (Tamara I. (2007). NetScratch : a networked programming environment for children. Thesis, . Retrieved February 2, 2011, from http://dspace.mit.edu/handle/1721.1/41677

Xinogalos, S., Sartatzemi, M., Dagdilelis, V., & Evangelidis, G. (2006). Teaching OOP with BlueJ: A Case Study. *Advanced Learning Technologies, 2006. Sixth International Conference on* (pp. 944-946). Presented at the Advanced Learning Technologies, 2006. Sixth International Conference on. doi:10.1109/ICALT.2006.1652599

Xinogalos, S., Satratzemi, M., & Dagdilelis, V. (2007). Re-designing an OOP course based on BlueJ. *Advanced Learning Technologies, 2007. ICALT 2007. Seventh IEEE International Conference on* (pp. 660-664). Presented at the Advanced Learning Technologies, 2007. ICALT 2007. Seventh IEEE International Conference on. doi:10.1109/ICALT.2007.214

York, J., & Pendharkar, P. C. (2004). Human-computer interaction issues for mobile computing in a variable work context. *International Journal of Human-Computer Studies*, *60*(5-6), 771-797. doi:16/j.ijhcs.2003.07.004

8 Appendixes

Installation process

Botbeans has two different options to install and run it: installation package or JNLP (Java Network launching Protocol). JNLP allows running the application automatically from a browser link pointing to a JNLP file, which will install the application as a web start application.

Installation packages are the more typical process of installing an application in a computer system, but in this case the user has to choose the right package for the operating system he is using.

Figure 40 Botbeans setup entrance

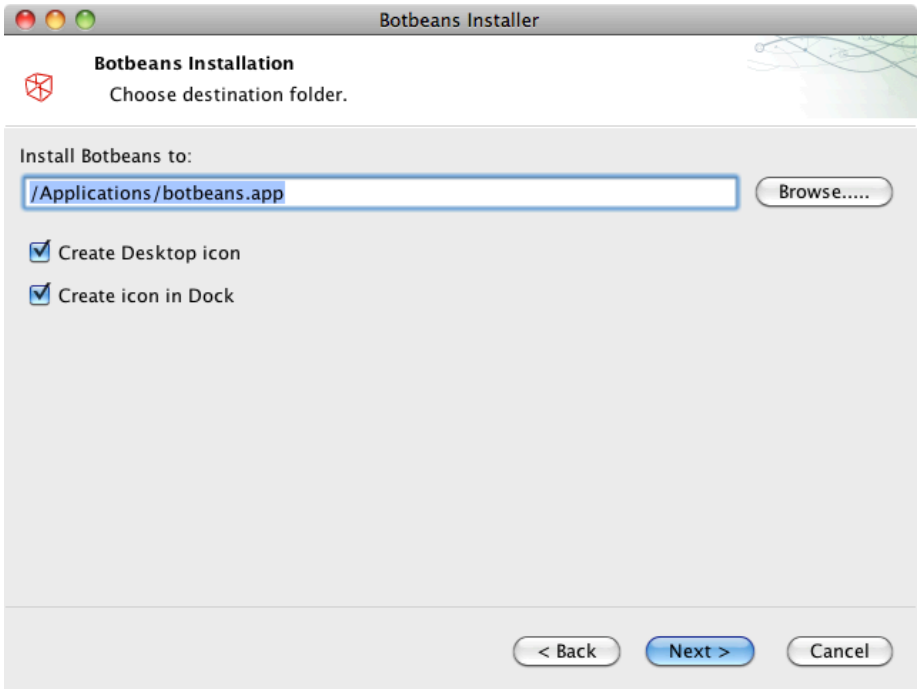


Figure 41 Botbeans setup

The installer (Figure 41) comes from the used rich client platform. This gives one big advantage since Netbeans Platform installer supports Windows, Linux and MacOSX allowing an easy installation in three main platforms.

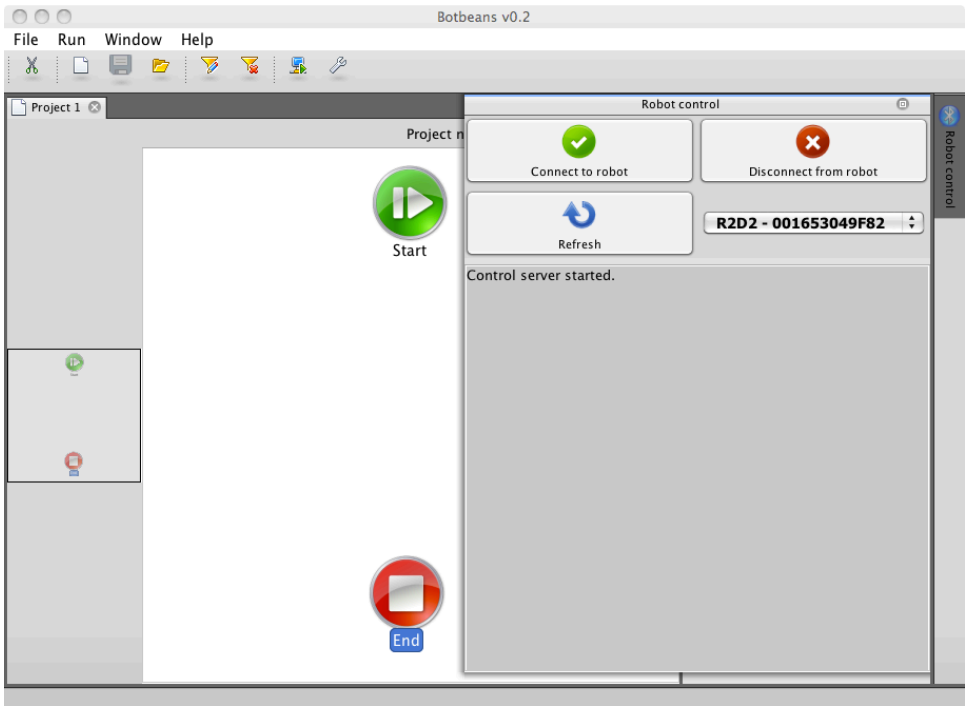


Figure 42 Robot control component

If a robot isn't configured the robot control component will open (Figure 42), this component will detect the available robots and display them in a combo box in order for the user to choose to which one he wants to connect.

If only one robot is detected, Botbeans will connect to it by default without showing the control component. After a connection is made, every thing is ready to execute an algorithm.

Curriculum vitae

Europass Curriculum Vitae

Personal information

First name(s) / Surname(s)	**Aparício Dias, Pedro MIguel**
Address(es)	Rua Nossa Sra. das Graças Nº98, 2200-156 Abrantes (Portugal)
Telephone(s)	912857824
E-mail	petermdias@gmail.com
Nationality	Portuguese
Date of birth	1985/05/16
Linkedin	http://pt.linkedin.com/in/pedromdias
Website	http://www.pedromdias.com
Gender	Male

Work experience

Dates	01/09/2008 →
Occupation or position held	Equiparado a Assistente do 1º Triénio (Teaching assistant)
Main activities and responsibilities	Teaching programming related classes (Introduction to Programming, OOP, Distributed Systems) in the computer science department; Administrator of department infrastructure (Xen Server, LAMP, SVN, Untangle,…) Maintainer and developer of Portugol IDE;
Name and address of employer	Instituto Politécnico de Tomar (Portugal)
Type of business or sector	Education
Dates	01/08/2010 - 01/01/2011
Occupation or position held	Senior Programmer
Main activities and responsibilities	.NET C#, PHP, Java, MySQL, MS SQL Linux systems administrator. Open-source software integrator.

Name and address of employer	RISA Consulting (Portugal)
Type of business or sector	Consulting
Dates	01/03/2011 →
Occupation or position held	Part-time Software Developer
Main activities and responsibilities	Software development;
Name and address of employer	LoveMachine, Inc San Francisco, USA
Type of business or sector	Software Development
Main activities and responsibilities	PHP, jQuery, MySQL, Titanium SDK, APIs Consumption

Education and training

Dates	20/09/2003 - 24/07/2008
Title of qualification awarded	Computer science bachelor degree.
Name and type of organisation providing education and training	Instituto Politécnico de Tomar, Escola Superior de Tecnologia de Tomar
Level in national or international classification	Bachelor degree.

Other language(s)

Self-assessment	Understanding				Speaking				Writing	
European level (*)	Listening		Reading		Spoken interaction		Spoken production			
English	C2	Advanced user	C2	Advanced user	C1	Advanced user	B2	Independent user	C1	Advanced user

(*)*Common European Framework of Reference for Languages*

Scientific publications

Pedro Dias, & Sancho Oliveira. (2011). Meet and greet programming using graphical languages and tangible interfaces. Information Systems and Technologies (CISTI), 2011. Presented at the Information Systems and Technologies (CISTI 2011), Braga, Portugal, June 2011.

P. Dias and S. Oliveira, “Botbeans: a new educational visual programming tool with tangible results,” ACM SIGDOC European Chapter/ Eurosigdoc Workshop on Open Source and Design of Communication, Lisbon, Portugal: ACM, 2010, pp. 43–44.

A. Manso, P. Dias, C. Marques, Ensino e aprendizagem de algoritmia com a ferramenta Portugol IDE - Published in XI International Conference on Engineering and Technology Education - Ilhéus bahia, Brazil, 7-10 March 2010.

	A. Manso, C. Marques, P. Dias - Portugol IDE v3.x- A new environment to teach and learn computer programming - IEEE Engineering Education 2010, Madrid, Spain , 14-16 April 2010.
Technical publications	http://netbeans.dzone.com/nb-simplified-ide-for-learning-to-program