V-Toys: an experiment in adding visual tiles to EToys

Pierre-André Dreyfuss
SEM Département Instruction Publique - Genève
pierre-andre.dreyfuss@edu.ge.ch

Serge Stinckwich
GREYC - Université de Caen
Serge.Stinckwich@info.unicaen.fr

Abstract

Usability issues in programming languages and associated environments are the topic of multiple studies, which led to the creation of many innovating programming systems where the user is the center of the design process, allowing non-programmers to create programs without understanding the underlying concepts. In this paper, we describe the V-Toys system which add visual programming to the EToys scripting system of Squeak. This framework is intended to children that have problems with text and reading. As a side effect, the fact that visual stuff eliminate the barrier of different language, enable V-Toys to be used as an international programming language since there is no need for translations to be able to understand it and use it. Built as an extension of EToys it is compatible with all the Squeak environment and can be mixed with stuff either programmed using Smalltalk or EToys.

1. Introduction

Since the early 1960’s, researchers have built a number of programming languages and environments with the intention of making programming accessible to a larger number of people.\[8\]

Among different environments accessible to children, Squeak[6] is one of the richest for educational use of computer. In this environment, kids are not just users but actors too. They can act on this environment by programming it with the EToys scripting system, but the text tiles used are not suitable for young children who are not yet able to read.

Now retired, one of the author\(^1\) was working in 2000 as a teacher in computer science for students 12-15 years old and as a researcher about introducing computers in schools. With some students, the Logo programming language\(^2\) was already too hard to learn. They have difficulties with syntax and remembering command names. Some experiments was done with ToonTalk[7], a visual programming system, more suitable for them and some children who seemed to be good at computer science lessons were in fact very weak on other matters. The reason is that most evaluation at school is done using written texts. With a visual language which does not use text for programming, they were able to show there skills in logic.

Students changed easily from ToonTalk to EToys when the change of software occurs during the year 2001. Many concepts were similar and since with EToys the use of text is limited, just reading is needed to choose a tile and then move it with the mouse into the script area. For young children that can’t yet read this is not satisfactory, a full visual language is needed. A visual language will also help to break language barrier at the time of projects exchange between children speaking different languages. We are reporting here some experiments we have done with a visual version of EToys, we have called V-Toys (Visual EToys).

The remainder of this paper is organized as follows: Section 2 recalls what are the main requirements for a programming tool for novice people like children and describes more precisely our proposal: V-Toys, a visual adaption of Squeak EToys. Section 3 describes some examples of V-Toys projects. Finally, section 4 is about related work and in section 5, we discuss some conclusion and planned future works.

2. V-Toys description

V-Toys is born from the conjunction of several ideas and experiments:

1. Novice programmers like children with language difficulties and older people with cognitive disabilities need a visual programming system (i.e. without text) so they should be able to express themselves and use different intelligence that the lexical one (coming up quickly with the appropriate word so that a problem can be worded and thereby made more clear). Another

\(^1\)Pierre-André Dreyfuss
\(^2\)http://en.wikipedia.org/wiki/Logo_programming_language
The benefit of using a completely visual system is to enable exchange of projects between children speaking different languages.

2. We would like to integrate V-Toys deeply with the already existing EToys Squeak environment and also keep the compatibility chain from the programming language level (Smalltalk) to the visual script level (V-Toys). For example, the EToys framework is developed in Smalltalk and EToys script can also be viewed as Smalltalk code. As you could bridge the gap between Smalltalk code and a visual programming system like EToys[1], we envision a system where the teachers can freely mix textual and iconic tiles if needed. Because of its openness and simple architecture, Squeak can help novice programmers switch from one level to another one progressively.

3. The design of V-Toys is based on the authors' expertise about visual programming tools for novice learners (EToys, ToonTalk, etc...). One solution doesn't fit all needs: we should allow the integration and communication between several styles of visual programming (for example behavior-based and tiles-based) in the same platform.

2.1. V-Toys's tiles

V-Toys programming is not basically different from using traditional EToys scripts. To identify the statements which are presented in the form of tiles, icons replace the words usually used for EToys tiles in the viewer of an object (figure 1).

With EToys, the receiver's name is the object's external name and is written in the beginning of every tile in the object's viewer. With V-Toys, the receiver is indicated by using the mouse and he is called a designator. A click on the designator followed by a click on the receiver will do the job. After that, a reduced picture of the receiver can be seen in the designator (figure 1). Since the thumb picture is often too small, a dedicated '?' tile helps to see which object is in a designator. The action of this tile is the same that the 'reveal me' in a viewer's menu: the object is blinking a while, then it is surrounded by the halo.

2.2. V-Toys's script

A script tile has a holder which will hold the tiles to form a script. V-Toys tiles are in fact just specific EToys tile and V-Toys script, just holders of V-Toys tiles. Clicking on the exclamation mark of this tile will execute all the tiles in the holder (figure 3). With EToys a script is only executed as a whole, with V-Toys each tile inside a script can be executed using its exclamation mark.

2.3. V-Toys's parameter

Blue designators are getting a value from a 'value tile' or a 'number' tile. A 'value' tile is like a getter tile in EToys, i.e. an EToys tile grabbed from the left part. These 'value' tiles are blue like the blue designator. A blue designator is getting the content of the variable 'valeur' (meaning value in French).

The classic "Drive a car" demonstration can easily be reproduce with VToys scripts (figure 4). To drive the

![Figure 1. V-Toys tile versus EToys tile](image1)

![Figure 2. Link from a designator to a receiver](image2)

![Figure 3. V-Toys script versus EToys script](image3)

car the script should be activated, this is done by the ‘acti-
vate’ tile (the tile with the clock). The tile is executed by
clicking on its exclamation mark.

2.4. Adaptations for young children

Several adaptations have been made in order to easier
the task of younger children. For example, a group of spe-
cial tiles have been set in order to recreate a Logo-like turtle.
Three tiles have a fixed value: forward 20, turn 90, turn -90.
An other example of adaptation is the status of the ‘pen up

2.5. Making tiles with EToys

In order to program VToys with EToys, some generic
names have been used (these names are currently French
names since V-Toys is written in French) across all the tiles.

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>action</td>
<td>any</td>
<td>The action done by the tile.</td>
</tr>
<tr>
<td>ajour</td>
<td>update</td>
<td>Player</td>
<td>Updates some display in the tile.</td>
</tr>
<tr>
<td>cible</td>
<td>target</td>
<td>Player</td>
<td>The receiver.</td>
</tr>
<tr>
<td>valeur</td>
<td>value</td>
<td>any</td>
<td>The value used by a parameter designator.</td>
</tr>
<tr>
<td>para</td>
<td>parameter</td>
<td>Player</td>
<td>The object from which a value is obtained.</td>
</tr>
</tbody>
</table>

As an example, the script for the "forward" tile is shown
in figure 7.

3. Examples and experimentations

No trials have been carried out with children but there
are many examples of projects done by Suzanne Guyader,
a retired teacher⁴. Suzanne and the authors have never ac-
tually met, but they have been collaborating by email for
many years. She was teaching Plastic Arts and starts using
Squeak EToys as a graphic tool. She has done a great deal of

⁴some projects have been made with collaboration of her granson
works to help teachers in the use of Squeak for developing creativity in children.

3.1. Didactic and teaching strategy

Creativity, exploration, constructivism is at work when she is creating new projects. Exactly the interactions we want between children and teachers. Children trying to do something of their own with the tools available and teachers giving the missing pieces to achieve the goal either showing were they are to be found, if existing, or creating them if needed.

As an example of evolution of a project:
1. An initial project with tiles for arithmetic operations was sent to Suzanne.
2. Suzanne wants to make a tutorial. She puts a rectangle in front of the display of the result to hide it (figure 8).
3. Visual tiles was made for showing or hiding objects and the ‘random number’ tile (figure 9).

![Figure 8. Arithmetic tutorial](image)

![Figure 9. Arithmetic tutorial, new version](image)

Now we can say that V-Toys is the fruit of close collaboration with her. Another interesting result of this collaboration is that documentation is mostly written by Suzanne, who is the user and don’t omit speaking about things which are evidence for the programmer but difficulties for a beginner. Most parts of the pages are written in French, but the projects can be downloaded and many have visual comments in them.

One of the main problem with the constructivist approach is that here are often too many possibilities for the learner. To correct this, the didactic steps are a progressive initiation using projects containing only the needed bricks. Children can build their own collection of tiles putting them in a flap of a personal project. For personal exploration and for teachers two projects contains all the tiles. One for the tiles for young children and one for the older. Tiles are grouped in a book with a visual index. It is very easy to browse the tiles clicking on the indexes. Tiles are documented by help balloon, this way balloons can be translated just by adding translations to the Squeak dictionary for a language.

3.2. The face projet

This setup (figure 10) is freely inspired by a similar project in Comikit[9], where a character could have several pictures, one for happy mood, another one for sad mood. Character pictures are painted using the painting tool of Squeak. If these tiles are used in a ‘collision tile’, the change of picture occurs when the face encounters another object the same way Comikit does.

In the figure 10 we see on the upper left side of the screen the pictures used as model in the scripts and on the right side the actors, the actors are the objects which will interact in the story. Moving these actors will produce the interactions described by the scripts in the lower left side of the figure.

Figures 11 and figure 12 show on the left the detail of an action and on the right its script.

- The face of the child will be sad if it is touched by the glass of wine. In the same time, the glass is emptied.
- The face of the child will be happy if it is touched by the chocolate and the chocolate is eaten.

Since we want to let someone else to play the story, the glass of wine will be refilled by touching the bottle and the chocolate is regenerated by the magic wand.

A lot of projects are available on the web site: [http://community.ofset.org/index.php/Squeak_Visuals-Toys](http://community.ofset.org/index.php/Squeak_Visuals-Toys)

4. Related Work

As the activity of programming is often too complex for children, the majority of the research work which related to
A first way to deal with these difficulties is to use a programming language close to a natural language. Smalltalk-80[11] for example was designed from the beginning as a programming language for children. This idea was later continued in the PhD thesis of Colette Girardot⁵[5] and more recently in the BotsInc⁶ project by Stéphane Ducasse[2].

There are many ways to categorize the existing visual tools for novice programming. A way of doing it is according to the two following approaches: "tile-based" vs "behavior-based". The first one consists in representing the expressions of the programming language as visual bricks (or tiles) easy to handle and compose. Tiles prevents novice of doing syntax errors. Moreover, it is not necessary to learn syntax since the kid could choose the appropriate expression inside a limited list of correct statements. Finally the tiles proposed at a given moment take into account the context of objects. Examples of systems that use a tile-based approach are Squeak EToys and Scratch.

Squeak EToys[6] are designed to allow children to learn ideas by “building and playing around with them” (Alan Kay) either through interacting with simulations others have built or creating their own simulations. The EToys environment provides kids with a variety of pre-made objects, from simple shapes (rectangles, star, Bezier curves, ...) to more elaborate ones (containers, button, sliders, ...). A simple drawing tool is provide to allow children to create their own objects. All objects have viewers with specific tiles that you drag out to build more elaborate programs that control the behavior of object. If-statements are the only control structures included in the EToys system. Users can trigger object behaviors based on a variety of mouse events, or the behaviors can be started, stepped and stopped with a set of pre-made buttons users can add to their simulations. The EToys programming style can be compared as a kind of situated event-driven programming[4] approach.

In V-Toys we can takes advantages of mixing textual and iconic tiles in the same program. The similitude will assure an easy transition from V-Toys to EToys.

Scratch[12] is an environment very similar to the one of EToys but in the context of a closed system. Like EToys, he is written in Squeak but the access to the underlying language is locked. This provides a simpler system but which prohibits all possibilities of transition between the visual programming level and the programming language (Smalltalk). The carefully design of Scratch allows his use by everybody, contrary to EToys, where bugs or positioning errors of objects are not rare (for example tiles not attached to script).

The second approach attempts to limit the low level algorithmic aspects by proposing catalogues of high level behaviors that can be combined to program. For example conditional or iterative structure control can be very difficult to understand and manage for children or beginners. This is a more ludic approach of programming, but if the goal is learning how to write program, some concepts will not be viewed. Examples of systems that use a behavior-based approach are ComiKit, MagicWords and ToonTalk.

ComiKit[9] has been the main inspiration work for the V-Toys project. Comikit is based a visual before/after...
mechanism, like Stagecast\(^7\) but without a grid. This mechanism is more intuitive than assembling tiles. However it runs in its own environment so the facilities of the Squeak environment are lost.

![Figure 13. MagicWords simulation with V-Toys scripts](image)

MagicWords[3][10] from the same author is intended for young children, but has textual labels. An original solution has been found to avoid the need of reading skills; when the mouse is over a label, the name of the label is pronounced. Like ComiKit, it runs in its own environment. Using V-Toys, it is easy to program labels having the same behaviour; this time running in Squeak GUI (figure 13).

ToonTalk\(^8\) use a physical metaphor for program execution and is one of the most intuitive visual software. It works by doing things with objects interacting when dropped on each other (programming by examples). For instance dropping a number on another number will result in a number containing the sum of the previous numbers. This corresponds to direct commands. To make a program, a robot is trained. This time the hand of the user is replaced by the robot’s hand. Children are no more describing what to do but just doing things. Activity is no more lexical but procedural. This change let children with lexical difficulties express their programming skills. This is making Toontalk hard for programmers but easy for kids.

ToonTalk uses visual metaphors for programming concepts among them: robots are methods, tuples or messages or vectors are boxes, transmission of data is done by birds flying to their nests. For example to copy a value from a variable to another variable, we have to get a copy of the value using the magic wand from a box (source variable) and put it in an other box (destination variable).

5. Conclusion and future work

In this communication, we describe a new visual framework, called V-Toys developed since 2006 by Pierre-André Dreyfuss. The main idea behind this work is to extend the Squeak EToys approach to new audiences like children not being in age of reading or people in difficulties, while preserving the integration with the Squeak environment.

V-Toys project still suffers from several limitations. The main one is the large amount of morphs needed in a project since tiles are morphs too. Especially in projects where morphs containing V-Toys scripts are copied: the size of projects becomes huge. One way to solve this problem is to replace V-Toys scripts by EToys scripts. We also need to make more experiences with children of various ages in order to validate the whole approach.

This prototype made it possible to bring concrete answers to a certain number of problems, in particular by a limited use of text to the profit of icons and visual feedback (like dices or moving pencils). An aspect which started to be tested and which it would be desirable to develop is that of a visual documentation language. The idea is to be able to give in a visual way the tasks to be realized by a child: representation of connections between objects, illustration of the movement, situations to be reached, ...

One future work who would like to conduct is the integration of visual language in order to control physical devices like robots in the Squeakbot[13] project. SqueakBot\(^9\) is a pedagogical platform where kids control robotic modules with the help of Squeak EToys. Scratchboard\(^10\) proposes a similar project where Scratch can be coupled with an external micro-controller-based card in order to control physical devices: light or temperature sensors, motors, etc ...

The authors of this paper would like to thank the many individuals who contributed to the development of V-Toys, notably Suzanne Guyader, a now retired Plastic Arts teacher, who was the first person to test them and to use them in her work. One will find at the following URL some french pages with more details about V-Toys: [http://community.ofset.org/index.php/Squeak_Visuals-Toys](http://community.ofset.org/index.php/Squeak_Visuals-Toys) with many examples of V-Toys tiles. Translation in English of these pages is under way.

References


\(^7\)http://www.stagecast.com/
\(^8\)http://www.toontalk.com/
\(^9\)developed in partnership with the University of Caen and the non-profit association under the french law: Planète Sciences [http://www.planetes-sciences/](http://www.planetes-sciences/).
\(^10\)http://scratch.mit.edu/pages/scratchboard


