# TOONTALK REBORN
# Re-implementing and re-conceptualising ToonTalk for the Web

Ken Kahn[1]

## *Abstract*

*The design and implementation of ToonTalk began in 1992, the same year that Microsoft released Windows 3.1. ToonTalk, being a programming system with the look and feel of a video game, needed to be tied to a software platform capable of displaying game animation on typical computers of its day. Initially MSDOS was chosen to be replaced by Microsoft Windows.*

*Today, thanks to implementations of HTML5 and JavaScript in web browsers, it is relatively easy to implement systems like ToonTalk in a cross-platform manner. Furthermore, there is no need to install any software, a practise that is difficult in schools with locked-down computers and tablets.*

*After twenty years of experience with ToonTalk and more recent programming systems we can do much better than just re-implementing ToonTalk. Due to a new software architecture* ***ToonTalk Reborn*** *is very modular, open, and radically customisable. Interface styles ranging from the original video game style to a conventional graphical interface with buttons, menus, and windows are possible. New ToonTalk widgets can be independently authored and easily incorporated into ToonTalk Reborn.*

***Keywords*** ToonTalk, ToonTalk Reborn, programming languages, constructionism, web apps

## 1. Background

### 1.1. History of ToonTalk

ToonTalk [1, 2] was conceived in 1992 with the goal of using video game technology to bring the most advanced computer science research in programming languages to a wide audience. The core idea is that each abstraction in the computation model has a corresponding *concretization* that captures the fundamental properties of the computational abstractions. These concretizations can be playful and understood in their own terms, even by very young children. The computation model chosen was *concurrent constraint programming* [3] which provides a safe well-grounded way of introducing concurrency, communication, and synchronisation to programs. The other core idea underlying ToonTalk is the construction of programs by example followed by the removal of details for abstraction [4].

[1] ToonTalk@gmail.com, University of Oxford, Oxford, UK.

ToonTalk was first published in 1998 by publishers in Sweden, Brazil, Portugal, Japan, and the UK. It was self-published in North America. Two large European research projects, Playground [5] and WebLabs [6], used ToonTalk to explore game making and sharing by children. Playground focussed on young children while WebLabs focussed on children making serious games in science and mathematics. As part of his doctoral research Leonel Morgado explored the use of ToonTalk with pre-school children and teachers [7]. In 2009, after three major releases, development stopped and ToonTalk was made free and open source.

Two fundamental enhancements to ToonTalk arose in response to the needs of the Playground and WebLabs projects. The Playground Project explored *anima-gadgets*, composable high-level components that enabled a kind of 'middle-out' programming. Children can add behaviours for making sounds, movement, reacting to events, and much more to pictures and other objects. The implementation of an anima-gadget is available on its back side for inspection or editing. Anima-gadgets can be authored by anyone and easily shared.

For the WebLabs Project exact arithmetic was introduced to ToonTalk [8]. Integers with literally millions of digits are supported, as well as rational numbers made out of pairs of these big integers. Decimal expansions of fractions are also supported. Infinite expansions are displayed with shrinking digits to convey the fact that there is no limit to the number of digits. When numbers are expanded more digits are computed as they become large enough to see. Confusions caused by round-off errors never arise due to the exact internal representation of numbers. The WebLabs Project explored the use of ToonTalk's communication and synchronisation mechanism to implement infinite sequences of numbers [9].

### 1.2. The Modelling4All Project and NetLogo

In 2006 the predecessor of the Modelling4All Project was started at the University of Oxford. This was inspired by the earlier work on the *Space Game Maker* [10]. The Modelling4All Project aims to make agent-based modelling more accessible, open, and collaborative [11]. The project developed the *Behaviour Composer*, a web-based tool for assembling, customising, and authoring *micro-behaviours* written in NetLogo [12]. Using a web browser one can find micro-behaviours hosted anywhere on the web, edit their parameters, and enhance them to build agent-based models. Micro-behaviours, models, and supporting resources are all shared as web pages. It is designed to support a *Web 2.0* community around building, sharing, discussing, and rating agent-based models and resources.

## 2. ToonTalk Reborn's Design Challenge

The challenge is to design a new version of ToonTalk incorporating the ideas from the Modelling4All Project and NetLogo. Also the support for anima-gadgets needs redesigning to be better grounded and more flexible. Furthermore, the design should be very modular to support experimentation, different looks-and-feels, and open source community contributions.

Ideally, ToonTalk Reborn should support these communities:

1. **Learners of computational thinking and computer science**. The semantics of ToonTalk is built upon a solid computer science foundation (concurrent constraint programming). The

predominant computation model underlying most programming systems is imperative programming with some object-orientation. Rather than compete directly with such systems (e.g., Scratch, Logo, JavaScript, Python, Java), ToonTalk Reborn can offer learners a well-grounded *alternative* way of thinking about and visualising computation and computer programming. See [13] for a comparison of Logo and ToonTalk.

2. **Web page authors who want to create and incorporate interactive widgets on their pages**. This entails good integration with current web components and technologies. This should support both beginners and those authoring ToonTalk Reborn tutorial material.
3. **Makers of computer games**. While competing with existing game authoring systems is perhaps too ambitious, the ability to create games complements the other strengths of the system. Also ToonTalk's communication and synchronisation mechanisms are particularly well-suited for building multi-player games in a straight-forward manner.
4. **Beginning agent-based modellers**. Agent-based modelling is a great way to learn science as StarLogo and NetLogo have demonstrated.
5. **Illiterate users**. A unique feature of ToonTalk is that it can be used by children who have yet to learn to read (or have reading difficulties). This also facilitates the sharing of ToonTalk programs between different language speakers.
6. **Off-line users**. While additional features will be provided by servers, the basic functionality should be available to those without an Internet connection.
7. **Open source community**. The design needs to support independent developers interested in contributing kernel widgets, different 'skins', tutorials, sample programs, and other resources.

## 3. The Architecture

The architecture is built around the idea of containers of widgets. Widgets have three aspects:

1. **A semantic core**. This implements a protocol for implementing the basic ToonTalk actions such as pick up, drop, use, and modify.
2. **A front side**. This implements the appearance and interface of the widget as seen by its users.
3. **A back side**. This implements the interface for authoring and editing the widget.

Widgets can contain other widgets, for example, ToonTalk boxes. Backsides can be composed to define behaviours consisting of sub-behaviours.

ToonTalk Reborn robots are the primary way of defining behaviours. Their appearance and training can be as in ToonTalk but any widget that implements the robot protocol can be used. For example, instead of training them with examples, they could be created in a comic strip fashion [14]. Native behaviours can be implemented in JavaScript to introduce a new kind of functionality that can't be defined by composing existing behaviours. Robots working on the back side of a widget can alter the widget.

The container for ToonTalk Reborn widgets can be any HTML element. When all the widgets are provided by trusted sources, this is adequate. We are exploring the use of Caja [15] to provide a *secure* way of combining widgets from any source within the same HTML element.

## 4. Examples

When completed users of ToonTalk Reborn will be able to create and run all original ToonTalk programs including games such as Pong and Space Invaders, physics and ecosystem simulations, and explorations of infinite sequences [6, 7, 9]. A very simple example is illustrated in Figure 1 where a robot was trained to add a number and its copy repeatedly. The training involved only clicking the 'Train' button, clicking the 'Copy' button, picking up the resulting copy, dropping it on the original number, and clicking the 'Stop training' button. The conditions specifying when the robot will run then needed to be relaxed by selective removal of detail so the robot will work with any number [4].

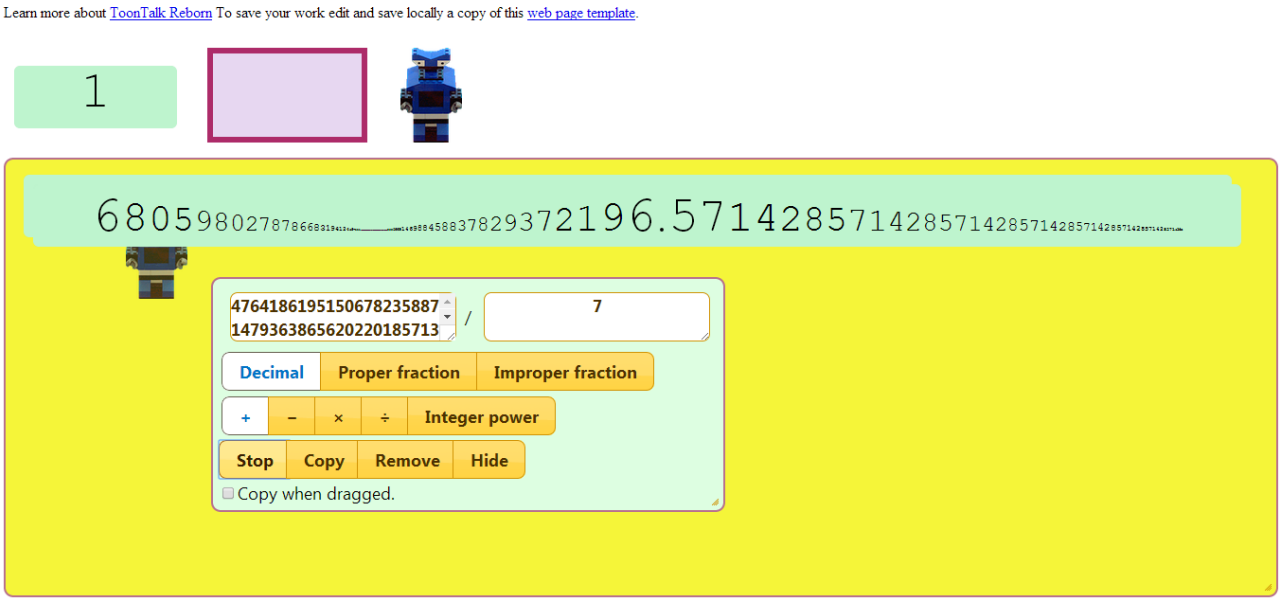



**Figure 1 – Screen shot of a robot trained to repeatedly double 1/7**

Another example is illustrated in Figures 2 and 3. This robot was trained to compute successive Fibonacii numbers by copying the first number, adding the second one to the first, and placing the copy in the second hole. It then places the ratio of the two numbers in the third hole.

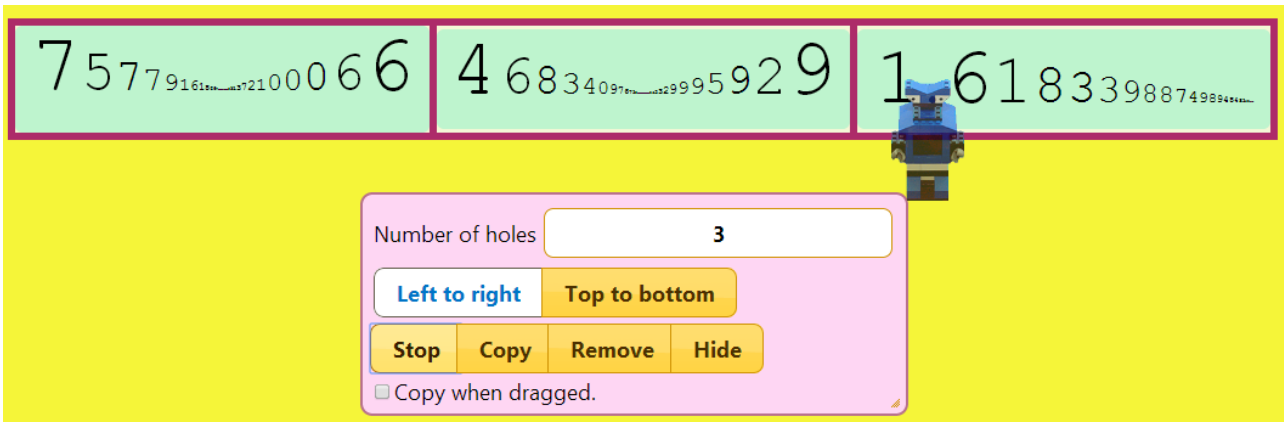



**Figure 2 – A robot computing Fibonacci numbers and their ratio**

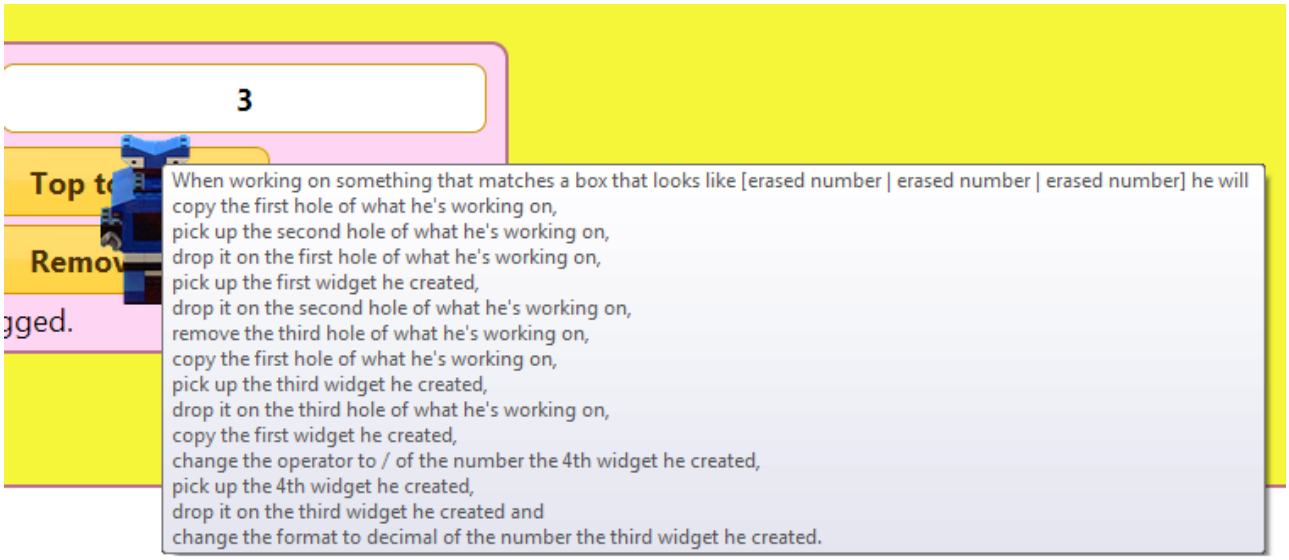
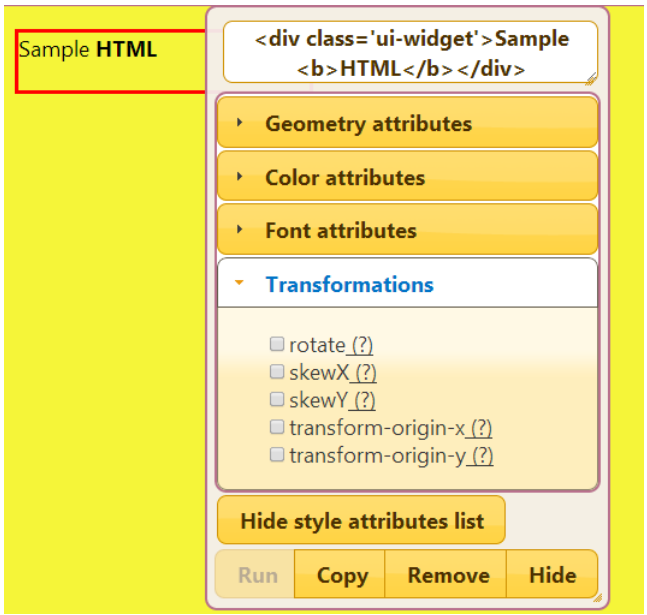



**Figure 3 - The automatically generated title that appears when hovering over the robot**

## 5. Programming HTML elements

ToonTalk Reborn currently supports the drag and drop of text, images, and HTML rich text. These become "element widgets". The backside of these widgets can contain viewer/controllers for any CSS style attribute. Because there are hundreds of such attributes the implementation currently supports about a dozen including position, size, font-size, color, and transformations.




5. **Figure 4 – The backside of an Element Widget**

Figure 6 illustrates a robot that spins an image at a constant speed and grows the image geometrically. Note that an alternative, more modular, implementation would be to use two "anima-gadgets". One that spins any element and another that changes the width and height (perhaps by in turn using two anima-gadgets that change each attribute separately). These anima-gadgets can be presented on web pages where they can run to demonstrate their behaviour. They can be dragged

into a ToonTalk Reborn work area and their backsides can be added to other elements to combine behaviours.

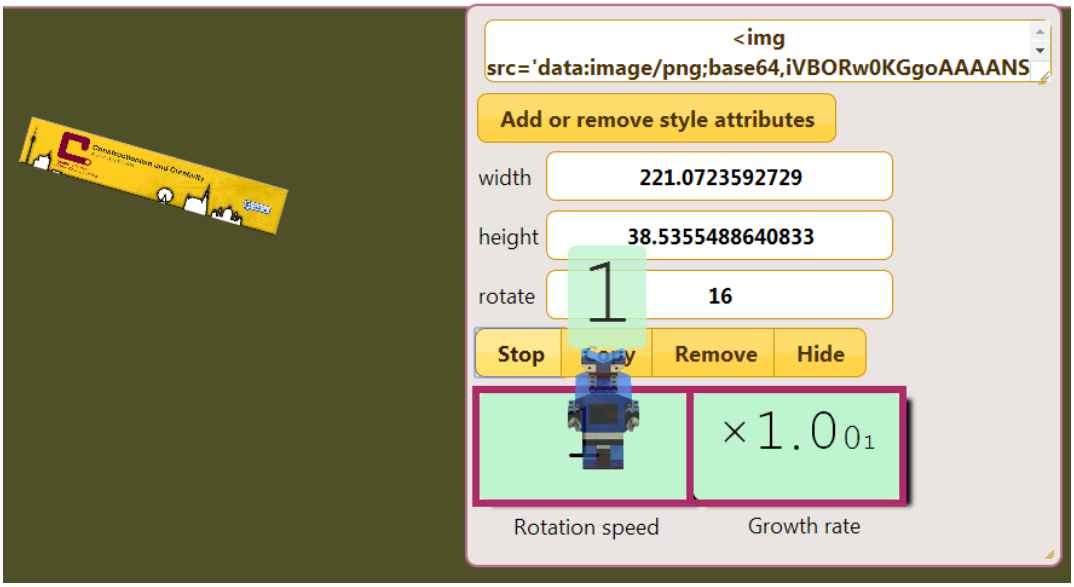



**Figure 5 – A screenshot of an robot spinning and growing an image**

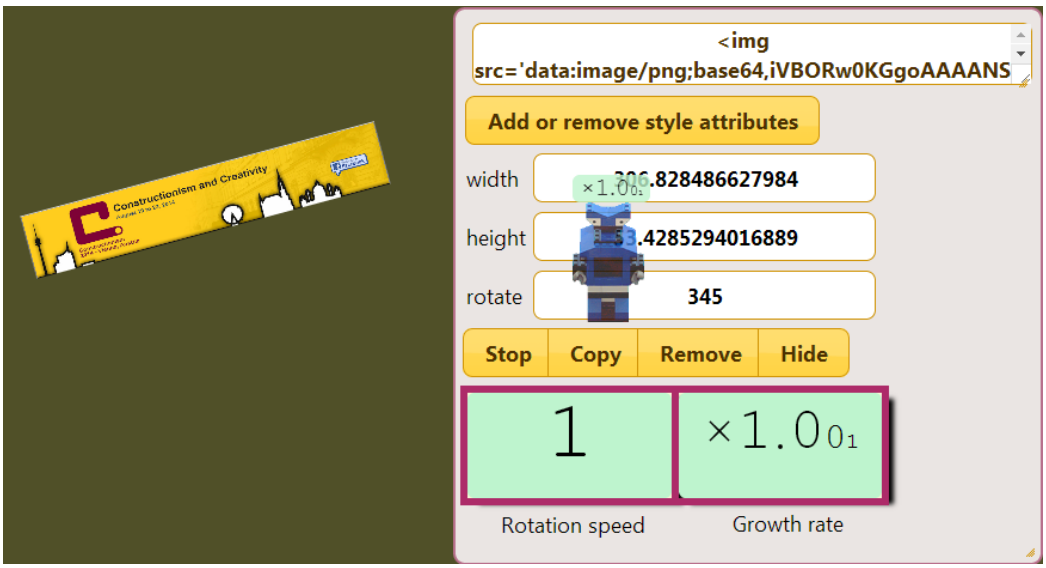



**Figure 6 - A later screenshot of the robot spinning and growing an image**

## 6. Implementation

At the time of this writing, the implementation is only of robots, numbers, boxes, and HTML elements. The interface (the backsides of these widgets) relies upon buttons, input boxes, and the like. Numbers have the full functionality of ToonTalk's exact arithmetic. The plan is to build a complete core implementation first and then explore game-like interfaces and new kinds of widgets.

We have learned from the Modelling4All Project the value of web pages that contain libraries of components, tutorial guides with live content, and documentation containing interactive widgets. This is acheived in ToonTalk Reborn very elegantly using the cross-browser drag and drop facility available in all modern web browsers. A user can read about a ToonTalk Reborn widget on a web

browser tab or window and then drag a live copy of it into their own workspace on a different tab or browser window.

The implementation is pure JavaScript together with the JQuery library. The server is hosted by the Google App Engine. The source code and beta versions of the system are available at https://github.com/ToonTalk/ToonTalk.

## 6. Plans

In addition to completing the set of widgets and tools found in ToonTalk, ToonTalk Reborn will incorporate new functionality. Event handling will be handled in the same way as other communications in ToonTalk, in other words with messages delivered to nests by birds. This will simplify the addition of new features and provide a better foundation for the sensors and controllers of ToonTalk.

ToonTalk required that communication and robot state be contained inside of boxes. This simplified the implementation and interface but was too restrictive and led to clumsy workarounds. Birds will be able to carry any widget. Robots can be trained to work on any widget and can reference other widgets for additional state. Among other things, this enables robots to perform meta-operations such as training other robots.

The original ToonTalk looks three-dimensional since its sprite imagery is based upon photos of clay sculptures and toys. But the implementation relies purely on two-dimensional graphics primitives. With WebGL it should be possible to build a truly three-dimension ToonTalk Reborn reusing the implementation of the semantic aspects of widgets and implementing only the new front and back sides of widgets.

The kernel of ToonTalk Reborn will be capable of building agent-based models but to properly support such activities we anticipate that a layer of software on top of the kernel will need to be constructed. We expect it will provide the functionality of many of the successful primitives of NetLogo.

## References


[1] K. Kahn, "ToonTalk - An Animated Programming Environment for Children", *Journal of Visual Languages and Computing,* June 1996.
[2] K. Kahn, *ToonTalk*, [online], Available: http://toontalk.com
[3] V. Saraswat, *Concurrent Constraint Programming*, MIT Press, March 1993
[4] K. Kahn, "Generalizing by Removing Detail: How Any Program Can Be Created by Working with Examples", *Communications of the ACM*, March 2000 and longer version in *Your Wish Is My Command: Programming By Example*, edited by H. Lieberman, Morgan Kaufmann, February 2001
[5] R. Noss, C. Hoyles, J-L. Gurtner, R. Adamson, S. Lowe, "Face-to-face and online collaboration: appreciating rules and adding complexity", *International. Journal of Continuing Engineering Education and Life-Long Learning*, 2002 Vol.12, No.5/6, pp.521-540
[6] C. Hoyles, K. Kahn, Y. Mor, R. Noss, E. Sendova, A. Sacristán, G. Simpson, "The WebLabs Project: Building New Formalisms for Mathematical and Scientific Ideas", In proceeding of: the *7th International Conference on Technology in Mathematics Teaching*, Bristol, January 2005
[7] L. Morgado, M. G. Cruz, and K. Kahn. "Taking Programming into Kindergartens", *EuroLogo Proceedings*, Portugal, August 2003.

[8] K. Kahn, “The Child-Engineering of Arithmetic in ToonTalk”, *Proceedings of the Interaction Design and Children Conference*, College Park, Maryland, June 2004.
[9] K. Kahn, E. Sendova, A. Sacristan, and R. Noss, “Young Students Exploring Cardinality by Constructing Infinite Processes”, *Technology, Knowledge and Learning Journal*, May 2011.
[10] K. Kahn, R. Noss, C. Hoyles, and D. Jones, “Designing digital technologies for layered learning”, *Informatics Education – The Bridge between Using and Understanding Computers*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2006.
[11] K. Kahn and H. Noble, “Modelling4All - Supporting a web-based community making, learning, sharing, and exploring computer models”, CAL Conference, Brighton, March 2009. Also [online] http://modelling4all.org
[12] Wilensky, U. *NetLogo*, [online], Availble: http://ccl.northwestern.edu/netlogo/ Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.
[13] Ken Kahn. “ToonTalk and Logo - Is ToonTalk a colleague, competitor, successor, sibling, or child of Logo?“, *Proceedings of the EuroLogo Conference*, August 2001.
[14] M. Kindborg, *Concurrent Comics: Programming of Social Agents by Children*, Issue 821 of Linköping studies in science and technology, Universitetet i Linköping, Sweden, 2003, ISSN 0345-7524
[15] Google, *Google Caja*, [online], Available: https://developers.google.com/caja/