HelloScratchJr.org: Curricular Design and Assessment Tools to Foster the Integration of ScratchJr and Computational Thinking into K-2 Classrooms

Juan Carlos Olabe, *jolabe@cbu.edu*
Electrical and Computer Engineering Department, CBU

Xabier Basogain, *xabier.basogain@ehu.es*
Department of Engineering Systems and Automatics, University of the Basque Country, EHU

Mikel Olabe, *miguelangel.olabe@ehu.es*
Department of Engineering Communications, University of the Basque Country, EHU

Abstract

ScratchJr was developed to incorporate programming into the classrooms of K-2 grades. A successful achievement of this goal requires the creation of age appropriate curriculum and sets of corresponding assessment tools. This paper presents a set of design criteria and developmental guidelines for curricular materials and assessment tools and methods. The website HelloScratchJr.org has been created to disseminate these findings. The site is scheduled to go online in the summer 2015.

Keywords

ScratchJr, curricular development, programming, assessment tools

1. Introduction

The formal study of computational skills in schools (K-12) has been recognized by many institutions and administrations. As an example, England, starting in the academic year 2014-15, formally instituted the study of computational thinking and computer programming as part of the core curriculum of primary and secondary education as described in the 'National curriculum in England: computing programmes of study' (Department for Education England, 2013).

ScratchJr plays an essential role in this process since it has been designed specially as the introductory computer programming environment for children ages 5 to 7. ScratchJr is a graphical programming environment specifically designed for the developmental and learning needs of children in kindergarten, first, and second grades (Flannery et al., 2013).

MIT and Tufts University released the iPad application ScratchJr to the general public in the summer of 2014, coinciding with the biannual Scratch@MIT conference (Strawhacker, Lee and Bonta, 2014).

It is very likely that many K-2 classroom teachers may have had little or no programming training during their studies or professional development, and therefore the need for curricular materials and appropriate assessment tools is critical in these early years of education.

The following sections describe a set of guiding strategies and criteria for the creation of curricular and assessment tools. These guidelines are intended to help teachers evaluate the content of current curriculum as well as to develop new, age appropriate materials. The guidelines will be illustrated with concrete examples to provide conceptual anchors that teachers will use in their learning and teaching endeavors.

2. Curricular Materials

2.1. The Two Branches of the Tree

ScratchJr, as many other programming environments, including Scratch, allows the creation of programs that could be described as games, or stories, or combination of interactive stories and games (Resnick et al., 2009).

In these programs we could classify the building blocks that make up a project into two groups: 1) the blocks that implement the actions of the protagonists, the sprites, (for example: to move right, to say hello, to disappear); and 2) the blocks that control when those actions occur (for example: when the sprite is clicked, when the sprite is touched by another character, forever, etc.)

If we think of these sets of blocks as belonging to two branches of a tree, we can start to assess the cognitive skills required to implement a particular project by looking at the location of the project blocks in the tree.

Action blocks, such as move to the right or say hello, are cognitively simpler since they are already part of the daily vocabulary of a 5 year old child.

Control blocks, especially when acting in coordination, such as ‘when A occurs then move to the right and at the same time move up,’ require higher cognitive skills to process and understand their operation. If these blocks are used indiscriminately or in poorly designed environments, they will lead to inscrutable behaviors that not even adults could decipher. (Using ‘send and receive’ messages it is quite easy to create in ScratchJr projects with behaviors that will puzzle the vast majority of adults. This is a territory to be avoided in the classroom.)

2.2. The First Three Years: K-1-2

Using the metaphor of the Two Branches of the Tree, the curricular content in Kindergarten will

heavily draw from the branch of Action Blocks, and the content of first and second grades will gradually add components from the branch of Control Blocks.

The blocks of the Branch of Actions can be grouped in three thematic areas: Motion, Physical Appearance, and Story Telling (Fig. 1).

- Motion: The group of motion includes the blocks that allow the sprites to move and turn. There are eight blocks, and they all are included in the blue motion menu.
- Physical Appearance: This group includes the blocks that can alter the appearance of the sprites. There are five blocks: three of them allow altering the size of the sprite, and two allow the control of whether the sprites are visible or not.
- Story Telling: This group allows the expression of ideas through written text or audio, or special effects through sounds and music. There are three blocks that implement these actions: Say, and Play Recorded Sound and Play a pop sound.

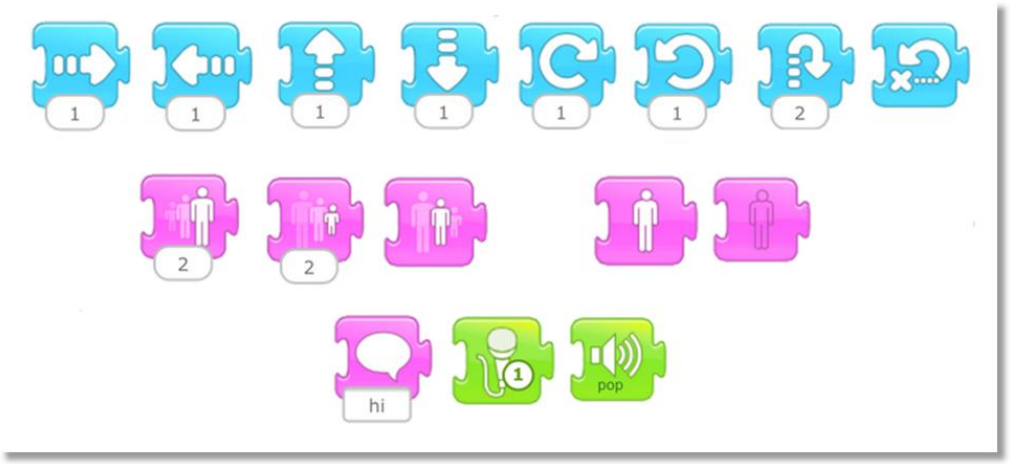


Figure 1. Blocks of the Branch of Actions

The blocks of the Branch of Control can be grouped in five thematic areas: Start/Stop, Communication via Messages, Repetition, Change Pages/Projects, and Set Speed (Fig. 2).

- Start/Stop: Three blocks allow to start execution of a script: Green Flag, Tap and Bump. The Wait block temporarily halts a script. The Stop block stops all the character's scripts.
- Communication via Messages: Two blocks, Send Message and Start on Message, implement this communication. A total of six colors allows up to six sets of pairs in a project. If the Send and Start components of the pair reside in different sprites, and therefore reside in different scripting areas, there will be a great demand on the short-term working memory of the child, especially because the color code of the message is void of intrinsic meaning, unlike Green Flag, Tap or Bump where the meaning is explicit. For these reasons, communication with messages will require the greatest level of cognitive skills.
- Repetition: Two blocks implement repetition, Forever, and Repeat a number of times.

- Change Pages/Projects: The Change of Page block is in fact a change of project block since each page is a separate project that does not necessarily share sprites with the other pages and where no communication via messages is possible. At the same time, each page, or project, can issue multiple Change Page commands, within one or many sprites, opening the door to unwanted complexity and uncontrolled flow of the program.
- Set Speed: A Set Speed block allows the change of the rate of certain actions into slow, average or fast settings.

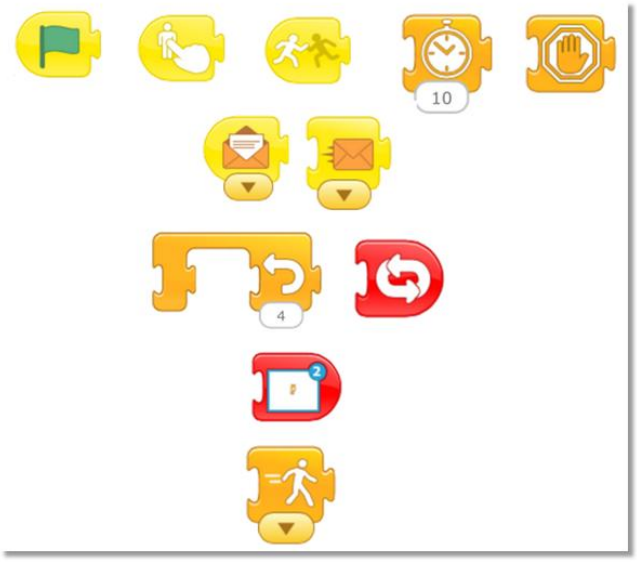


Figure 2. Blocks of the Branch of Control

In the curriculum of Kindergarten it is appropriate to use any of the Branch of Actions blocks in the creation of a project since they form scripts that are equivalent to actions in the daily vocabulary of a 5 year old.

In Kindergarten we consider that it is advisable to create projects that are sequential in nature (avoiding parallel threads), and that do not include physical interaction between sprites (because it is difficult to visualize interacting scripts that reside in separate scripting areas: a child's short-term working memory is still very limited in scope and time.) The communication via messages and the coordination of multiple projects via the Go to Page block, will be studied in later years.

The three blocks of the Branch of Control that are age appropriate for Kindergarten students are the Star on Green Flag and Start on Tap because they form scripts linguistically similar to their language, and the block Repeat Forever, which only repeats the sentence again and again.

Students in First Grade, after having consolidated the concepts and strategies gained during Kindergarten, are ready to increase their lexicon with some blocks from the Control Branch of the Tree.

From the Start/Stop group it is appropriate to include any of the blocks. In particular the block Bump requires imagining a future event that in general will have to be triggered by some other scripts. For example, in order for the cat to bump into the rabbit, either or both will have to move, and the scripts controlling this motion will be the cause of triggering the bump script.

The Repetition and Set Speed blocks are modifiers of concepts already seen in Kindergarten and therefore will be easily assimilated by first graders.

Finally, the Change of Page block allows the creation of multiphase projects, and as in a multi-chapter book, the space for creativity is enlarged. But as in a book, the Change of Page should be used to create a linear narrative.

Students of Second Grade will now be ready to complete their lexicon with the use of communication via messages, which allows for a great deal of interactivity among scripts and among sprites. The new ideas for the second graders will need to be introduced systematically with incremental projects, as well as a set criteria for programming style that will prevent some of the undesired effects described earlier.

3. Curriculum of Computational Ideas

The design of a computational thinking curriculum is not necessarily the creation of projects where the emphasis is in the list of the different blocks of the language that are utilized. Rather, it is the development of activities and projects where the student is exposed to, and allowed to practice to become familiar with fundamental and powerful computational ideas. The familiarity with, and knowledge of these ideas will allow the student to combine them into more powerful ones, and to explore other new ideas following some of the patterns discovered in the first ideas.

To illustrate the need for a comprehensive portfolio of ideas in the design of a curriculum, this section analyzes some concrete examples where just creating a functioning program may not necessarily lead to new and more complex projects.

For this task we selected projects that are available to all ScratchJr users. The ScratchJr iPad app includes a quick intro to ScratchJr and a set of eight Sample Projects: Under the Sea, Farm, Cat on Bat, Friends, Jack Be Nimble, Animal Race, Bump and Quick Intro (ScratchJr Itunes, 2015).

Under the Sea: This project is an animation which includes Fish1, Fish2, Seahorse and Starfish. We will concentrate on the scripts of the two fish. When the project is executed, the two fish move synchronously, left-right, left-right, left-right, as a well-trained dancing team. Since both fish move in the same way, the same distance, at the same time, the observer should deduce that similar scripts control the two fish. That should be the correct assumption, and the scripts should validate it. Looking at the scripts we find that Fish1 repeats the following pattern: Move-Right(2), Move-Right(-2); and Fish2 does: Move-Left(-2), Move-Left(2). Notice the double switch: Right-Right for Left-Left, and (2)(-2) for (-2)(2). These two different scripts are designed to implement

the very same Left-Right constant dance. A good script should approximate the mental description that a child would use to describe the action (Fig. 3).

Figure 3. Project Example: Under the Sea

(The concept of negative numbers is introduced in the schools of the United States in Sixth grade (Common Core State Standards, 2015) (6.NS.C.5). ScratchJr has been designed for K-2 grades, however the vast majority of the blocks of Motion and Physical Appearance have been designed to work also with negative numbers. For K-2 students these features should be avoided.)

Tap the Farm: This project includes a Pig, a Horse, and a Chicken. When the Pig is tapped one script triggers a sound and another makes him move. This simple set of two scripts is repeated for the Horse and the Chicken. In each sprite the sound is different, and the movement is different. One good idea (sound and movement when tapped) is reinforced by applying it to all sprites; and one good idea (sound and movement when tapped) is made even better by showing how flexible it is when the sounds and the movements are changed (Fig 4).

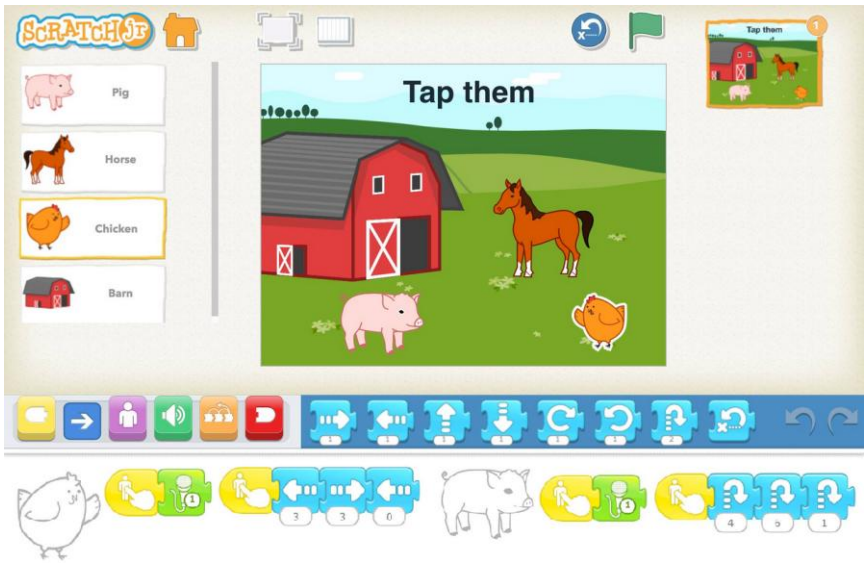


Figure 4. Project Example: Tap the Farm

Friends: In this project there are three pages, and the third one includes the background of a soccer field where a soccer ball moves into the goal, and when it does, the six children in the field scream 'Goal!' This is an example of synchronization with broadcasting: 'when the ball enters the goal, it should command the children in the field to trigger their scripts.' Instead, the project is implemented with seven independent scripts, all triggered at the same time. The process of cause and effect is simulated by adding individual Wait blocks in the children's scripts. If for some reason the timing of the ball is changed, all the other scripts will have to be individually changed as well. The idea of one event triggering many other simultaneous events, synchronization with broadcasting, is a fundamental computational idea, and it needs to be implemented with the appropriate structures (Fig. 5).

Figure 5. Project Example: Friends

The design of curricular computational content should have as a goal the gradual introduction of computational thinking ideas and practices. In describing curricular content, an important criterion should be to list the specific ideas that the child would have acquired permanently, that is, a description of his/her computational thinking portfolio. This portfolio should be explicit and comprehensive, and the child should be familiar with it. It should be similar to when a child is asked 'what songs, stories and games do you know,' and the child lists the songs, stories and games, and also the child is able to sing the songs, narrate the stories and describe the rules of the games. In the same way, a goal in curricular design is to enable the student to tell others what computational ideas he or she has acquired, and to explain to others what these ideas are, and how they are implemented, in this case with ScratchJr.

4. Assessment Tools

Mastery Learning describes the idea that profound knowledge of a subject is attained when at each step of the learning process a formal evaluation of the concepts and skills learned is implemented, and the feedback provided to the learner either requires additional work on the current topic, or readiness for the next material is acknowledged (Kulik, Kulik & Bangert-

Drowns,1990).

The mind operates in two distinct and very different modes: consciously applying a set of rules (System-2), for example multiplying 27x14; or automatically, effortlessly, and unavoidably (System-1), for example recognizing a friendly face. Learning ScratchJr, like learning any natural language, is a process of storing in long-term working memory a set of ideas and procedures. Long-term working memory resides in what is called System-1 (Kahneman, 2003; Stanovich & West, 2000) and therefore requires that the assessment tools be designed to address long-term working memory properties. This structure requires two sets of assessments: recognition and synthesis.

Using the example of the project *Friends* described earlier, the task of recognition involves seeing a soccer field with a ball and six children, where the ball moves, and when it reaches the goal, all children scream 'Goal!' and recognizing that the process is an instance of a synchronization with broadcast, where the conclusion of one action, the ball entering the goal, triggers all the other scripts simultaneously.

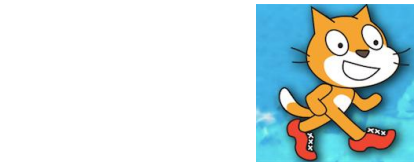
The task of synthesis is one in which the child is told, for example, the phrase 'Friends Project' and he or she is able to reconstruct the sequence of events, and the scripts that implement them in a way that replicates his/her memory of it. This implies that the project needs to be already in the child's long-term working memory.

Because ScratchJr is an iPad application, it offers the computer environment to implement both aspects of the assessment. An appropriate technique to implement the first assessment is to have simultaneous access to a video of the action to be recognized, and a set of alternative scripts that would implement it. The careful selection of alternative scripts can discriminate among several possible misconceptions on the problem, providing information on the success of the student, or what the misconception is. Because these assessments are automatic and implemented in the iPad, a teacher can have fine grain analysis of the progress of each individual student.

A set of paper-based assessments are available for the ScratchJr community (Circle the Box, and Reverse Engineering, Fig. 6) (ScratchJr.org, 2015). These assessments, however, may provide biased information and a skewed vision on the complexity of the programming project.

The first assessment set, Circle the Blocks, is based on a group of videos of recorded projects that the students view three times. Later they circle the blocks they consider were part of the script among a set of all blocks available in ScratchJr. This is a discrimination task to be implemented by System-2 (Kahneman, 2003; Stanovich & West, 2000), with interfering subtasks. Imagine being asked: 'select among these five pictures the one of the elephant with a pink bow' (System-1); versus being told: 'find and circle in this 10 by 10 box of random letters all the letters in the phrase "elephant with a pink bow (System-2)."'

In the Circle the Blocks assessment a total of seven tests are presented: two ranked easy, three ranked medium, and two ranked hard. Analyzing the complexity of the scripts it seems plausible



that the difficulty ranks are based on the percentage of students that successfully completed the assessment. However the complexity of an assessment should not be determined by the number of students able to successfully complete it, but by the complexity of the ideas and structures underlying it.

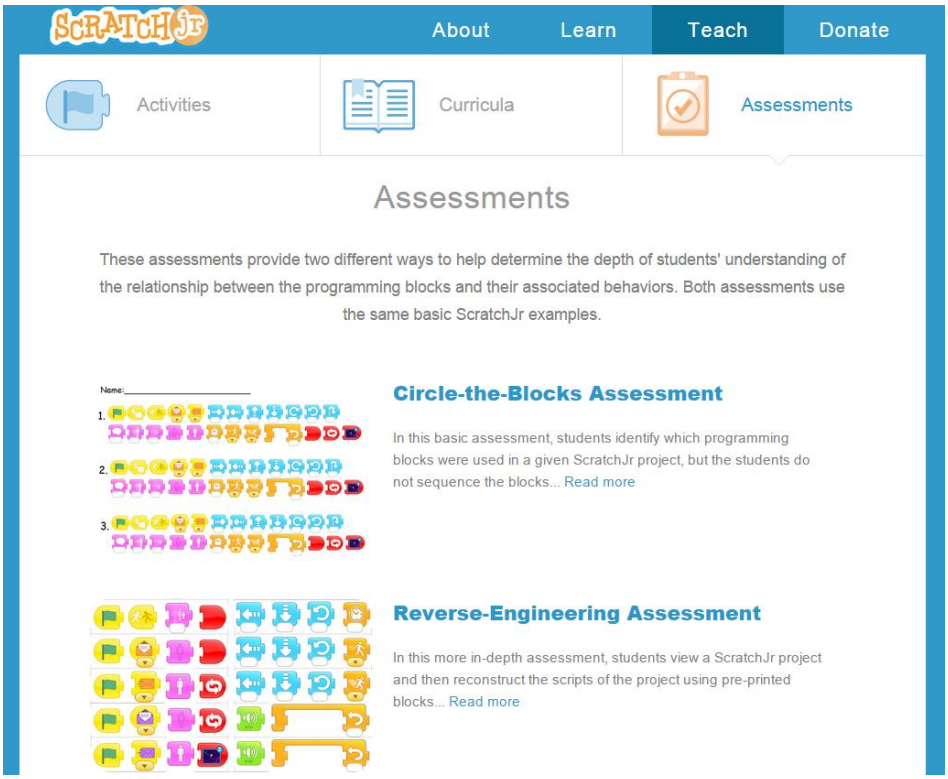


Figure 6. ScratchJr.org - Assessments

For example, test2, Hop Twice, Wait, Hop Again, which is ranked Easy, is of the same degree of difficulty as test3, Turn Right and Left, Go Up and Down, which is ranked Medium. Because how the test was implemented, it is plausible that the limited short-term working memory of the children played a major role in the overall performance of the assessment. Testing short-term working memory does not address the interest of the assessment: long-term working memory. It could however be a symptom that no long-term working memory is yet available in the child.

When analyzing test6, When Cat Touches Dog, Dog Disappears, which is ranked Difficult, (when in fact it should be ranked Easy,) a plausible explanation for the ranking is that the child has not yet acquired the idea of 'When A touches B, B does something.' Had this idea been available in longterm working memory, and had the assessment be addressed to System-1 and not System-2, the results would have been probably very different.

The second assessment set, Reverse Engineering, asks the students to paste together sticker images of the blocks in order to create a program that will generate the action seen in the video.

Reverse Engineering is a process of building a working prototype that will emulate the operation of the target system. For example: building a paper airplane that would fly like the paper airplane that the students saw in a video. In this example the children do not 'imagine' that they build a paper airplane, and later 'imagine' how the plane would fly. Similarly, in Reverse Engineering, the child needs to write the program in the computer; then execute the program, see the results, and adjust as necessary. Reverse Engineering is a synthesis process, a process of creating something. For this, the children need a sandbox where they can start building, explore their options, have feedback from their prototypes, and adjust as necessary. A sandbox is an important computational concept. It is the place where one can explore different alternatives, get feedback, see results, and adjust as necessary. Any programmer would attest that the feedback experienced when creating a project in a sandbox is essential in the learning process, in the self-evaluation process, and in the creative process. A system based on sticker images fails in these three areas.

The Reverse Engineering assessment should be implemented not with stickers but rather by writing a program in the iPad. This is the standard procedure, not only for younger students, but for college students as well (Coursera, edX) when they are assessed in programming synthesis tasks using a sandbox.

5. Summary

This paper has presented a set of guidelines in two complementary areas for the successful integration of ScratchJr into the classroom: 1) curriculum as comprehensive and specific set of computational ideas that gradually incorporates the functionalities of the language; and 2) assessment tools as a set of instruments that implement Mastery Learning and provide descriptive analysis of long-term working memory characteristics of the students. The website HelloScratchJr.org has been created to disseminate these findings. The site is scheduled to go online in the summer 2015.

Acknowledgments

The project HelloScratchJr.org is funded by the School of Engineering of Christian Brothers University and a Grant of Department of Education, Universities and Research – Basque Government (2010-15-IT863-13).

References

Common Core State Standards. Corestandards.org. [Online]. Available: http://www.corestandards.org/Math/Content/6/NS/C/5/. [Accessed 29 6 2015].

Department for Education England. (2013). Statutory guidance National curriculum in England: computing programmes of study. [Online]. Available:

https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study. [Accessed 29 6 2015].

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013). Designing scratchjr: Support for early childhood learning through computer programming. In *Proceedings of the 12th International Conference on Interaction Design and Children*, (pp.1-10). ACM.

Kahneman, D. (2003). Maps of bounded rationality: Psychology for behavioral economics. *American economic review*, 1449-1475.

Kulik, C. L. C., Kulik, J. A., & Bangert-Drowns, R. L. (1990). Effectiveness of mastery learning programs: A meta-analysis. *Review of educational research*, *60*(2), 265-299.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K. & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67.

ScratchJr. Itunes.com. [Online]. Available: https://itunes.apple.com/es/app/scratchjr/id895485086?mt=8. [Accessed 29 6 2015].

ScratchJr.org. [Online]. Available: http://www.scratchjr.org/teach.html#assessments. [Accessed 29 6 2015].

Stanovich, K. E., & West, R. F. (2000). Advancing the rationality debate. *Behavioral and brain sciences*, *23*(05), 701-717.

Strawhacker, A., Lee, M., & Bonta, P. (2014). ScratchJr. *Scratch@MIT Conference 2014*, 23. [Online]. Available: http://cdn.scratch.mit.edu/scratchr2/static/__34f16bc63e8ada7dfd7ec12c715d0c94__//pdfs/conference2014/Scratch%20at%20MIT%202014%20Conference%20Schedule.pdf. [Accessed 29 6 2015].