App Review: ScratchJr (Scratch Junior)

Steve Goschnick, Swinburne University of Technology, Melbourne, Australia

ScratchJr (Scratch Junior) is a visual coding environment in an app, designed to “introduce 5 to 7 year olds” to many of the concepts and joys of digital creation and programming. Technically, it draws from the wider-known Scratch blocks-based programming environment. Educationally, a considerable amount of pedagogical design has gone into it (Flannery, et. al., 2013).

Having seen what Scratch proper can do, I was hesitant to use the lesser-featured Scratch Junior app, as I was a little skeptical as to whether it could lower the target minimum age for getting kids into coding that far, all without sacrificing a lot of the coding concepts evident in the original Scratch. However, watching my 8 year old (with very minimal guidance from me) get into ScratchJr very quickly, I soon dropped the scepticism. Note: While a long-time user of an iPad, she has neither been introduced to programming at school or at home. Probably the closest to coding she had done prior to ScratchJr, was using some redstone blocks in MineCraft to open doors via pressure-pads and the like.

Previously, I had baulked at introducing her to full-featured Scratch, because I know very well that in introducing a child to coding, it is so very easy to go just a bit too far, and put them off it entirely for some time afterwards - been there, done that. I figured I needed a coding environment where I could say and do very little, and let the learner just go ahead and do their own thing, while I could hover about or pass by occasionally to give some help with the next ‘feature’ that she wanted to add to her own evolving creation. Getting kids into coding is prime-territory for user-centred learning, and the mentor really has to stay out of the way, as much as possible. There are three very useful stand-out features of ScratchJr I will mention right up top:

1. It is an app that runs standalone on tablets, without the need to have a web browser open and connected to the Internet (as is the case with the more sophisticated Scratch environment). Parents and/or teachers of very young children - the target group of ScratchJr (5 to 7 year old novices to coding) - often do not want them connected to the Internet, and certainly not for long periods of time.

DOI: 10.4018/IJPOP.2015010104
2. There is a ScratchJr for both iPad and Android tablets, giving it wide-ranging access to the young coding novices out there. A vast number of this cohort of learners have access to either an iPad or an Android tablet at home. Often young siblings will differ in which of the two platforms they have - like my two youngest: the youngest chose an iPad mini, while the other, an Android tablet. It gives them some independence from each other, but they also like to get together from time-to-time to share and compare their time and knowledge on the same app or game. The versions of ScratchJr across both the iPad and Android, are indistinguishable in features and interface. As such they can share their knowledge and also share the content that each of them creates via a built-in email feature (i.e. there is a project file format, that can be used to export and import Projects from within ScratchJr).

3. There is ample and highly usable Help built into the app itself, specifically: an Interface Guide (see Figure 1) which describes the single screen interface using numbered call-outs; a Blocks Guide detailing all the available coding blocks which are grouped together by colour according to like-or-related functionality (Block Categories); and a Paint Editor Guide for the built-in editor used to create either new image assets, or modify the nicely supplied repertoire of images. Also, a little animated introduction to the interface and how it is used.

Soon enough she had four scenes (Pages) set up and inter-linked in a few ways, creating an interactive story. The four images down the right-hand side are thumbnails of these four scenes, each edited in the built-in Paint Editor, after choosing a starting image from about 20 supplied.

*Figure 1. The Interface Guide built into the App is accessible via tabs along the bottom of screen*
backdrops. The current scene/Page being worked on (Page 1), is enlarged in the centre of the screenshot (the Stage).

Down the left-hand side are the objects (the Characters) which have been placed on the current scene/page, for the planned interactivity when the program is set in motion. The current object selected is a simple ‘stick’ (like a ‘j’ but inverted). She has placed it on the current Page, over near the driveway to the garage, as a trip-wire. The two interlinked blocks of code at the bottom of Figure 2, enact the simple functionality needed to make the trip-wire work. I.e. When the story Character (the Alien on the road) travels to the right and hits the trip-wire, they get transported to another scene/page in the game/story (Page 3). Those 2 blocks are all that is needed here: the first one causes an action when two objects touch - i.e. the trip-wire and the Alien in this case; and the second one defines what action then takes place, i.e. Go to Page 3.

Early in the process she called on me as a mentor, to make an object or button, function as a hotspot to do something when touched - in particular, to move about. Interestingly, she then took the idea straight to do a set of 4 such buttons mimicking the X, Y, A and B buttons on a game controller. She then wanted the 4 buttons to enact the movement in the player’s avatar (the Alien), to go Left, Right, Up and Down. Figure 3 shows the code required to make the Alien do just that. ScratchJr has a neat set of two coding blocks to broadcast and receive messages between objects, visible in Figure 3 as the last two blocks in the toolbar (the Block Palette) just below the central Stage: a closed-envelope and an open-envelope - to send a message and to receive a message, respectively. ScratchJr uses colours to identify different messages, so in this case the Y button emits a Yellow message on being touched, the X button a Blue message, and so on.

The learner quickly conceived and developed an interactive story, virtually unaided. She clearly drew on her prior experience of other apps, games and devices, for inspiration in what

Figure 2. An Interactive Story with 4 scenes (4 Pages down the right-hand side)
she wanted to create herself and how. The game-controller-like buttons, being the first obvious example. As an avid writer of little ebooks using the well-regarded Book Creator app, her idea of creating a multi-scene interactive story, is a natural progression on an ebook.

As someone who has taught the Delegation Event Model used in the Java language to new undergraduate Information Systems students - with some difficulty in many cases - I find what ScratchJr has done to do much the same, breathtaking in its simplicity. The concepts one needs to grasp it in Java include: Event Listeners, Event Sources, Events themselves, registering EventListeners with Event Sources, ActionListener, and the need to instantiate the ActionListener (or other) Interfaces as methods; and optionally, Anonymous Methods - all needed to understand how a single click of a button gets passed to the programmed object for which it was intended!

ScratchJr has: turned the Event (message) into a colour; turns an object (interactive Character) into an Event Source simply by attaching a closed-envelope block to it by drag-and-dropping one into the Programming Area while that object is the current Character; then effectively registers an Event Listener to an object (an interactive Character) similarly by dropping an open-envelope block in the same way. All done employing a very natural metaphor, in a simple drag-and-drop action.

Other coding blocks include looping logic controls, timers and so on - see Figure 4 for some further examples as described in the Blocks Guide - which I believe require less cognitive load in most programming languages, than learning the aforementioned event model in most higher-level languages.
CONCLUSION

ScratchJr does indeed allow a very young learner to get into the creation of interactive stories, games and other digital delights, by themselves, with just a little mentoring along the way. In the process, they pick up some of the main concepts needed in higher level programming languages, all while having some fun and being visibly empowered. In addition, there are online and book-based resources focused on ScratchJr, that a teacher might use in a classroom learning context.

However, the technical limits of ScratchJr are soon reached by an app-savvy 7 or 8 year old. For example, there is no concept of variables which would allow one to carry values across scenes and so forth (e.g. to enable a Character to pick up a key in one scene/page, and to use it to unlock a door in another). Much more restrictive and seemingly arbitrary is the limit of just 4 scenes/pages in a single Project - given that young kids quickly write ebooks much bigger than that in apps such as Book Creator. Removing these two limits in a later version would make the app usable well beyond the 7 y.o. upper age limit currently targeted by the developers of ScratchJr.

ScratchJr also allows music and recorded sounds, text and speech-bubbles, and the inclusion of photographs of/by the young coder.

Figure 4. Control and end blocks in the blocks guide

![Figure 4. Control and end blocks in the blocks guide](image-url)
It is however, an excellent first-step before going into Scratch proper, or to a coding environment such as Greenfoot or App Inventor, similarly aimed at late primary or secondary aged students. A step that will not put them off the whole endeavour of coding before they get meaningful results in creating things that they themselves want to create. I certainly now have no hesitation in encouraging my youngest child to get into one or more of those other coding environments, whereas before this experience with ScratchJr, I did.

AUTHOR’S NOTE

The ScratchJr app is available as a free download from both the Apple App Store and Google Play Store via the iPad or an Android tablet, respectively.

REFERENCES


Steve Goschnick is an Adjunct Professor in the School of Design, at Swinburne University of Technology, Australia. His research interests include people-oriented programming, intelligent agents, user interaction design, programming environments, data modeling and meta-models. He developed and taught foundational masters level subjects at The University of Melbourne for 10 years, into the MIS, MIT and MBIT courses, up until 2013. He has consulted widely in industry and been called upon as an expert witness in both programming and interaction design. He is the founding co-editor-in-chief of the International Journal of People-Oriented Programming.