Scratch Microworlds: Introducing Novices to Scratch Using an Interest-Based, Open-Ended, Scaffolded Experience

by
Moran Tsur

Bachelor of Science in Cognitive Science and Computer Sciences
Hebrew University of Jerusalem, 2011

Submitted to the
Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements of the degree of
Master of Science
at the Massachusetts Institute of Technology

June 2017
© 2017 Massachusetts Institute of Technology. All rights reserved.

Signature redacted

Signature of Author

Program in Media Arts and Sciences
May 12, 2017

Signature redacted

Certified by
Mitchel Resnick
LEGO Papert Professor of Learning Research
Program in Media Arts and Sciences, MIT
Thesis Supervisor

Signature redacted

Accepted by
Pattie Maes
Academic Head
Program in Media Arts and Sciences
Scratch Microworlds: Introducing Novices to Scratch Using an Interest-Based, Open-Ended, Scaffolded Experience

by
Moran Tsur

Submitted to the
Program in Media Arts and Sciences,
School of Architecture and Planning,
on May 12, 2017
in partial fulfillment of the requirements of the degree of
Master of Science
at the Massachusetts Institute of Technology

ABSTRACT
Currently, many introductory coding activities for children focus on engaging them in solving puzzles. This thesis explores a different approach to introducing coding that engages children in creating projects based on their interests. I present the iterative design and testing of Scratch Microworlds, simplified versions of the Scratch coding environment that contain a small set of blocks for making projects based on a theme, such as dance, soccer, or music. I use a design-based research approach to iteratively design, implement and evaluate Scratch Microworlds. The design of Scratch Microworlds is guided by three questions: (1) how to simplify initial experiences while still supporting creativity, (2) how to provide scaffolding while maintaining learners' agency, and (3) how to provide starting points that spark rather than limit the imagination. This thesis describes the design process, and analyzes the results of user-testing with children and educators. It concludes with a set of guidelines for the design of newcomer experiences into coding that support children as creative thinkers, informed by constructionist learning theory.

Thesis supervisor: Mitchel Resnick
Title: LEGO Papert Professor of Learning Research

This thesis is based on research supported by the National Science Foundation under grant number 1348876. Any opinions, findings, and conclusions, or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the National Science Foundation.
Scratch Microworlds: Introducing Novices to Scratch Using an Interest-Based, Open-Ended, Scaffolded Experience

by

Moran Tsur

Advisor

Signature redacted

Mitchel Resnick
LEGO Papert Professor of Learning Research
Program in Media Arts and Sciences, MIT
Scratch Microworlds: Introducing Novices to Scratch Using an Interest-Based, Open-Ended, Scaffolded Experience

by

Moran Tsur

Karen Brennan
Associate Professor of Education
Harvard Graduate School of Education
Scratch Microworlds: Introducing Novices to Scratch Using an Interest-Based, Open-Ended, Scaffolded Experience

by
Moran Tsur

Signature redacted

Reader

Sepandar Kamvar
LG Career Development Professor of Media Arts and Sciences
Scratch Microworlds: Introducing Novices to Scratch Using an Interest-Based, Open-Ended, Scaffolded Experience

by

Moran Tsur

Signature redacted

Reader __________________________  Natalie Rusk
Research Scientist, Lifelong Kindergarten, MIT Media Lab
Acknowledgements

The work presented here would have never been possible without the extensive support and encouragement I got from my different communities.

To my advisor, Mitch Resnick, thank you for giving me the opportunity to spend two years in the most amazing creative learning environment I’ve ever encountered. Thank you for your guidance, and for always challenging my thinking while still allowing me to build my own path.

I’m grateful to my thesis readers - To Natalie Rusk, my reader, collaborator, and mentor - thank you for your ongoing support - in the lab, and outside of it. Thank you for countless hours of discussions, which helped me frame my learning and understanding. To Karen Brennan, thank you for helping me see the joy of learning by creating and for inspiring me with your work. To Sep Kamvar, thank you for making me think outside of the box and open up to new ideas.

The Scratch Microworlds project is a result of many people working together. I want to thank the creators of the original Microworlds - Benjamin Berg, Shane Clements, Sarah Otts, Ricarose Roque, Natalie Rusk, Eric Schilling, and Chris Willis-Ford. Paula Bonta who supported me with thoughtful feedback, design ideas and easy to use tools. My interns, Michelle Wantuch, Esther Finkelstein, and Vivien Chen who were a major part of this project - from helping playtest to reflecting and building together - thank you for willing to jump right in and explore!

To the many educators and children we met - thank you for helping us by playing with the microworlds and sharing your feedback and excitement.

To the LLK and Scratch team family, who contributed immensely to my well being as well as to the project. It’s great to know there is always someone you can count on. Special thanks to Andrew Sliwinski, for his constant feedback and support from the very early stages of the project. To Ray Schamp, Eric Rosenbaum, and Matt Taylor - for supporting my technical challenges when building the prototypes. To the many facilitators who helped run workshops - Eric Schilling, Stefania Druga, Kreg Hanning, Kasia Chmielinski and Chris Garrity - many thanks! I’m especially grateful for the LLK grad students, current and former, for always being there when I needed advice, support, or a laugh. I keep learning from you all every day.

A super-awesome special thanks to my partner in crime - Carmelo Presicce, you know I couldn’t have done it without you.

To my friends, near and far, who have been a source of constant encouragement and patience. And last, to my family, and my husband Roy - your love and support are what keeps me going. Thank you all for always believing in me, taking care of me, and pushing me to find what I love. I think I found it.
Contents

Acknowledgements .................................................................................................................. 13
Contents .................................................................................................................................. 14
List of figures ............................................................................................................................ 16

Part I Motivation and Context ................................................................................................. 17
  Chapter 1 - Introduction ....................................................................................................... 19
  Chapter 2 - Background and Related Work ......................................................................... 22
    Constructionism and Instructionism.................................................................................... 22
    Scaffolding and Agency ...................................................................................................... 24
      Scaffolding ...................................................................................................................... 24
      Agency ............................................................................................................................... 25
    Interest-Based Learning ...................................................................................................... 26
    Tinkering ............................................................................................................................. 26
    Microworlds ....................................................................................................................... 27
    Puzzles vs. Projects ............................................................................................................ 28
      Puzzle-Based Environments ............................................................................................ 30
      Project-Based, Open-Ended Environments .................................................................... 32
      Combining Puzzles and Projects .................................................................................... 35
  Chapter 3 - Research Design ............................................................................................... 37
    Research Question ............................................................................................................ 37
    Research Design and Methodology .................................................................................. 37
      Evaluation ........................................................................................................................ 38
  Chapter 4 - What are Scratch Microworlds? ....................................................................... 41
    Main Design Features ........................................................................................................ 41
    Environment and Content .................................................................................................. 42
    Design Process and Iterations ............................................................................................ 42

Part II Design and Observations ............................................................................................. 47
  Chapter 5 - Simplifying while Supporting Creativity .......................................................... 49
    Reducing the Number of Coding Blocks .......................................................................... 50
      General-Purpose vs. High-Level Abstraction Blocks ...................................................... 50
      Selecting Expressive Subsets of Blocks ......................................................................... 51
      Number of Blocks ............................................................................................................ 52
    Simplifying Features and Tools ........................................................................................ 53
    Simplifying the Sprites ..................................................................................................... 54
    Tinkering as a Pathway to Creativity ............................................................................... 56
  Chapter 6 - Scaffolding While Keeping Learners' Agency .................................................. 57
    Scaffolding ........................................................................................................................ 57
The Blocks Palette ........................................................................................................ 58
Open in Scratch ........................................................................................................ 61
Similarity to Scratch ............................................................................................... 62
Starter Code .............................................................................................................. 62
Contextual Help ........................................................................................................ 64
Learner Agency ......................................................................................................... 67
Learners Decide What to Create ............................................................................... 67
Learners Decide When to Move Forward .................................................................. 68
Learners Decide What to Explore Next ..................................................................... 69

Chapter 7 ................................................................................................................ 71
Providing Starting Points that Spark the Imagination .............................................. 71
Interest-Based Starting Points .................................................................................. 72
"Deep" Interest ......................................................................................................... 73
Tinkerable Starting Points ......................................................................................... 74
Support Diversity of Outcomes .................................................................................. 75
Playful Explorations ...................................................................................................... 77

Part III Reflections and Suggestions ................................................................ 79
Chapter 8 - Learning and Teaching with Scratch Microworlds ......................... 81
Learning with Scratch Microworlds ........................................................................ 81
Newcomers to Scratch ............................................................................................. 81
Intermediate Coders ................................................................................................. 85
Experienced Scratchers ............................................................................................. 86
Teaching with Scratch Microworlds ......................................................................... 87
Supporting Educators ................................................................................................. 89
Facilitation Tips for Educators ................................................................................. 90

Chapter 9 - Looking Back, Looking Forward ....................................................... 91
Guidelines for Designing Microworlds .................................................................... 91
Creating from the Start ............................................................................................. 91
Simplicity and Creativity ......................................................................................... 92
Scaffolding and Choice ............................................................................................. 92
Interest-Based Experiences ....................................................................................... 93
Limitations .................................................................................................................. 94
Importance of the Context ...................................................................................... 94
Single-Meeting Evaluation ....................................................................................... 94
Fast Iterations ............................................................................................................ 94
Facilitation .................................................................................................................. 94
Future Directions ..................................................................................................... 95
Design and Implementation ....................................................................................... 95
Research .................................................................................................................... 96

Bibliography ............................................................................................................... 99
List of figures

Table 1 – Main differences in design features between puzzle-based and project-based approaches to coding introduction ................................................................. 29
Figure 1 – CodeMonkey (right) and codeSpark Academy (left) ......................................................................................................................... 30
Figure 2 – Code.org's "Coding with Anna and Elsa" ................................................................................................................................. 31
Figure 3 – The Scratch editor, with the Tips Window open (on the right) ................................................................................................. 33
Figure 4 – The Scratch Coding Cards ......................................................................................................................................................... 33
Figure 5 – A castle built in Minecraft and a crafting table .......................................................................................................................... 34
Figure 6 – An example of a Khan Academy coding challenge .................................................................................................................. 35
Figure 7 – Snap! Hour of Code tutorial .................................................................................................................................................... 36
Table 2 – An overview of testing sessions ......................................................................................................................................................... 40
Table 3 – Scratch Microworlds features, inspired by puzzle-based and project-based approaches ................................................................................................................................. 41
Figure 8 – The full Scratch editor and its parts ............................................................................................................................................. 43
Figure 9 – The first microworlds prototype, hip-hop and soccer microworlds ........................................................................................... 44
Figure 10 – Scratch-like editor with scaffolding. Gymnastics microworld with category-based scaffolding (top), and music microworld with a "show more" button (bottom) ......................................... 45
Figure 11 – The last prototype, based on Scratch 3.0 – art (top) and music (bottom) .................................................................................. 46
Figure 12 – Example projects created by young people .......................................................................................................................... 52
Figure 13 – Evolution of the "show more" button, from the early prototype on the left to last one on the right ................................................................................................................................. 59
Figure 14 – Scaffolding with Categories ....................................................................................................................................................... 59
Figure 15 – Examples of starter code from different microworlds ................................................................................................................. 64
Figure 16 – Examples of contextual help from Code.org (top), CodeMonkey (bottom right) and Khan Academy (bottom right) ................................................................................................................................. 65
Figure 17 – Instructional prompt in the microworlds prototype (animated GIF) ........................................................................................ 66
Figure 18 – The microworlds homepage, featuring 6 starting points ........................................................................................................ 68
Figure 19 – codeAcedmy’s example of a level structure ............................................................................................................................ 68
Figure 20 – Different art projects created by children and educators ...................................................................................................... 75
Figure 21 – Hip-hop and music examples created by children and educators ............................................................................................ 76
Figure 22 – Example projects created by youth at the last playtest ........................................................................................................ 83
Part I Motivation and Context

In this part, I introduce the context for the Scratch Microworlds project. I discuss the motivation for it, the theories and examples that influenced me and describe the research design. I finish with an overview of what Scratch Microworlds are, and a description of the research process.
Chapter 1

Introduction

Many of the activities and websites designed to teach children to code today are designed as a series of puzzles for children to solve. Children are provided feedback on whether they have solved the puzzle correctly, and then can proceed to the next one. In the progression through the levels, the puzzles become more complex but usually do not let the learner make decisions about what or how they will code. This can be seen as what Seymour Papert [27] referred to as an "instructionist" approach to technology, which puts the computer in the role of teaching children by introducing a concept, giving them problems to solve, and giving feedback to tell them whether their solutions are right or wrong.

Scratch offers an alternative approach to learning how to code. It takes what Papert called a "constructionist" approach to technology, in which children build knowledge by creating personally meaningful projects, in open-ended environments, with more control over their learning process [32]. Scratch is a graphical programming environment that children use to create interactive animations, stories, and games – and share them with others around the world in an online community [40]. Scratch is being used by young people in schools, homes, and other learning environments around the world, with 120 million unique visitors on the website per year and approximately 25,000 projects shared each day [49]. Scratch attracts a diverse range of young people, and is used by many organizations that work to broaden the participation of underrepresented groups in computing [2, 15].

While Scratch is easier to learn than other programming languages, getting started can still be overwhelming for many beginners. The interface contains 120 coding blocks and more than 20 buttons and other interface elements. Each new project offers a blank canvas and one character (a cat), and allows you to add and create whatever you want. Like facing a blank sheet of paper, seeing the initial screen upon opening the Scratch editor can be seen as either inviting or intimidating, depending on the person.

Many educators also find the number of options available for students difficult to manage within the classroom environment. As a result, many educators prefer using one of the puzzle-based tools instead of offering students open-ended exploration in Scratch, and thus miss the rich learning opportunities available to students when they become engaged in creating their own projects. Others do use Scratch, but not for open-ended design activities. They use an "instructionist" mode of teaching, which replaces open-ended exploration with step-by-step tutorials and specific pre-defined goals. For example, asking all students to create a game where a dragon collects apples. They all use the same sprites, and they build the same game.
Chapter 1 - Introduction

In this work, I addressed these challenges by creating a new introductory experience to Scratch, called *Scratch Microworlds*. Scratch Microworlds build on Seymour Papert’s theory of constructionism [32], and motivated by the value of making it easier for beginners. It is inspired by Papert’s notion of educational “microworlds” [30, 31] and combines the creative, open-ended, and interest-based approach of Scratch with a more scaffolded approach that makes the first steps less overwhelming. These experiences are designed to provide a simplified introduction to coding using a project-based – rather than puzzle-based – approach.

By creating a simple, more supported, and self-guided experience, I also hope to provide an alternative to the puzzle-based tools for educators who are interested in letting their students explore computation. The microworlds encapsulates some aspects of a constructionist experience, making it easier for educators to allow learners to work on meaningful, creative projects, without being overwhelmed by the wealth of options. By grounding these experiences in themes that are relevant to youth lives, I hope to create pathways for youth that may have not seen themselves as part of the computing world before. I hope that Scratch Microworlds will provide a starting point for both young people and educators that will help them to build their confidence, motivation, and curiosity, and will provide a pathway into Scratch, where they could keep learning and creating.

While I focus on Scratch in this work, the questions and tensions I explored are relevant to the design of learning experiences in general. I hope it could help designers who are interested in exploring these tensions, and incorporating support for creativity and learner-led experiences in their work. Scratch Microworlds use coding as a way to empower learners, but I believe this approach could be beneficial not just in introduction to coding, but also for many others subjects, aimed at different audiences, and in different contexts.

This thesis is organized in three parts: motivation and context; design and observations; reflections and suggestions.

In part I, I set the context for the design of Scratch Microworlds. In chapter 2, I survey the main theories that influence my work as well as examples of different ways to introduce coding to children. Chapter 3 presents the research questions that grew out of these influences, and describes the research design and methodology. Then, chapter 4 gives an overview of Scratch Microworlds, and the iterative process in which they were created and refined.

Part II dives deeper into the design of Scratch Microworlds, and discusses both the design process and the observations we collected. It looks at the design process through three perspectives that guided our design work. Chapter 5 describes our efforts to simplify the Scratch coding environments without losing its creative spirit. Chapter 6 present the struggle to add more scaffolding to the newcomer experience without giving up the learner’s control over the learning process. Chapter 7 explores the topic of theme-based starting points, discussing the benefits and challenges of designing interest-based pathways for newcomer experiences.
Chapter 1 - Introduction

In part III, I look back and reflect on Scratch Microworlds, and recaps the findings and next steps. Chapter 8 takes a step back and reviews the microworlds as a whole, through the eyes of the children and educators that tested them. Chapter 9 discusses the relevance of these findings to the design of other learning experiences, and describes the limitations and the future directions for the research.
Chapter 2

Background and Related Work

My work has been influenced by a few core educational theories, and by a survey of existing tools for introducing children to coding. In this chapter I introduce the core theories and use them to look at examples of available tools.

Constructionism and Instructionism

"The kind of knowledge children most need is the knowledge that will help them get more knowledge"


The term constructionism, which was defined by Seymour Papert in his National Science Foundation proposal [27], has many layers and definitions. In the introduction to Constructionism in Practice, Kafai and Resnick provide a definition of constructionism:

Constructionism is both a theory of learning and a strategy for education. It builds on the “constructivist” theories of Jean Piaget, asserting that knowledge is not simply transmitted from teacher to student, but actively constructed by the mind of the learner. Children don’t get ideas; they make ideas. Moreover, constructionism suggests that learners are particularly likely to make new ideas when they are actively engaged in making some type of external artifact – be it a robot, a poem, a sand castle, or a computer game – which they can reflect upon and share with others. Thus, constructionism involves two intertwined types of construction: the construction of knowledge in the context of building personally meaningful artifacts.


Several aspects of constructionism are most relevant to this work. First, the idea that people construct their understanding of the world by constructing projects. Second is the idea that young people are especially motivated while working on projects that are personally meaningful for them [20]. Kafai & Resnick explain that “[Constructionist theory] asserts that activities involving making, building or programming – in short, designing – provide a rich context for learning” [20]. They mention another important aspect, which is that “constructionist learning environments encourage multiple learning styles”. Constructionism also emphasizes the importance of letting learners control their learning process [29, 31] and the need to provide
second paragraph

children with rich experiences, that will allow them to actively construct, design and make, and learn by doing so. And especially allow construction in the world:

... construction that takes place “in the head” often happens especially felicitously when it is supported by construction of a more public sort of “in the world”. ... Part of what I mean by “in the world” is that the product can be shown, discussed, examined, probed, and admired. It is out there.


Papert also described another, opposing approach, which is a more traditional educational approach. He called it instructionism [32]. According to his description, instructionism is the belief that improving learning requires improving instruction – it’s the idea that “the unique way to improve a student’s knowledge about topic X is to teach about X” [32].

Papert suggests that constructionism and instructionism perceive the nature of learning differently. While the instructionist approach sees learning as the result of transmitting or conveying information, constructionism sees learning as the product of a learner constructing meaning themselves [32, 33].

These differences lead to a different approach to the design of learning activities or environments. An instructionist learning environment will present knowledge to the learner, while a constructionist learning environment will focus on creating better conditions for the learners to discover and construct their own understanding.

While technology plays a big part in Papert’s writings, constructionism is not limited to digital environments. Papert’s focus on the role of “the computer” in education comes from the belief that computers are very rich with opportunities, and open up new ways to interact with knowledge that is otherwise very abstract. While instructionist approaches usually let the computer do the instruction (like Computer Aided Instruction), constructionist learning environments use it to open up new, rich experiences for the learners [28].

Inspired by Papert’s constructionist approach, at the Lifelong Kindergarten group at the MIT Media Lab, we design creative learning experiences based on four core elements, sometimes referred to as “the 4 P’s of creative learning”. Its four core elements, as described by Resnick [38] are:

- Projects. People learn best when they are actively working on meaningful projects – generating new ideas, designing prototypes, refining iteratively.
- Peers. Learning flourishes as a social activity, with people sharing ideas, collaborating on projects, and building on one another’s work.
- Passion. When people work on projects they care about, they work longer and harder, persist in the face of challenges, and learn more in the process.
Chapter 2 - Background and Related Work

- Play. Learning involves playful experimentation – trying new things, tinkering with materials, testing boundaries, taking risks, iterating again and again.

To make sure we support learners of all levels of experience, we design learning activities with the metaphor of “low floor, high ceiling and wide walls” in mind. While designing experiences that have a low barrier to entry (low floors), we try to support advanced learners as well (high ceiling). To design for “wide walls” we create learning environments which support many types of projects, interests, and learners [40].

Scaffolding and Agency

When looking for ways to support learners’ first experience, I found a few key ideas in designing learning environments very relevant. Providing scaffolding for learning, and providing support for learner agency are two of these core ideas. In this section I present them and the tension between them.

Scaffolding

Scaffolding focuses on ways to enable a learner to succeed in something that would otherwise be beyond their reach [56]. Wood describes it as controlling – the adult controls elements of the task that are beyond the reach of the child, allowing them to focus on the ones that are within their reach. As the child becomes able to handle it on their own, it’s the adult’s task to slowly retreat from this control [34]. Vygotsky’s concept of Zone of Personal Development, a conceptual set of activities children cannot perform on their own, but can perform with the support of adults [53], offers another perspective on scaffolding. “Channeling and focusing” and “modeling” are two aspects of scaffolding [34]. Channeling and focusing refers to reducing levels of freedom from the task and drawing the learner’s attention to specific features, and modeling refers to the adult’s ability to model more advanced solutions to a task. As the use of technology for supporting learning increased throughout the years, the term “scaffolding” started being used to describe the role of features, tools, and processes in the learning process [34]. Software-realized scaffolding is also described by Guzdial [16] as a way to embed scaffolding in computer-based environments. Software-realized scaffolding can provide a more authentic context to the learners, can make more aspects of the learning visible and explicit, and can support students’ engagements with others [35].

Facilitators play different roles to in scaffolding learning in instructionist and constructionist environments. In a traditional instructionist approach, the role of the educator is to help students follow a predetermined learning path, helping them understand what to do next and giving them guiding feedback. With constructionism, the role of the educator is to support students as they are searching for inspiration and ideas, suggesting prompts that generate possibilities and offering materials or tools [23].
Chapter 2 - Background and Related Work

Learning tools often use similar approaches, trying to mimic the role of the facilitator. Scaffolding in an instructionist learning tool could help the learner follow the pre-designed path. But for constructionist tools, designing scaffolding without an in-person facilitator adds another complication. Constructionist scaffolding requires providing support as the learners progress in different directions and ways, while working on open-ended projects. Supporting this through a tool requires planning for all possible scenarios ahead of time. While educators can be flexible in responding to learners' needs, in a digital tool every response needs to be built into the system.

Agency

“...In most contemporary educational situations where children come into contact with computers the computer is used to put children through their paces, to provide exercises of an appropriate level of difficulty, to provide feedback, and to dispense information. The computer programming the child. In the LOGO environment the relationship is reversed: The child, even at preschool ages, is in control: The child programs the computer.”

- Papert, Mindstorm, p. 19 [31]

Papert chose to start the first chapter in Mindstorms, a book that inspired many to think about the role of technology in education and learning, with the above quote. Written 35 years ago, but just as relevant today as it was then, the question of who controls the learning process is one that concerns me a lot. In many coding experiences the “computer” still controls the experience (backed by its designers). Other tools, like Scratch and LOGO, support the learner in making their own decisions about the learning process.

Several theorists have emphasized the importance of learner agency in designing learning environments. Brennan, following Bandura, defines learner agency as the ability for a learner to “define and pursue learning goals” [3, 6]. Defining and pursuing learning goals enables the learner “to play a part in their self-development, adaptation, and self-renewal with changing times” [3, 6].

Support for learner agency is a main value in constructionism, as can be seen from Papert’s quote. But maintaining learners agency while trying to provide more support for their learning is a challenge. Without giving special attention to this balance, adding support and scaffoldings for the learner usually limits their agency. The definition of scaffolding as an adult (or technology) “controlling” some features of the experience instead of the learner, in turn means that the learner has less control over these features. This is true both in interpersonal and technological interactions, where supporting becomes synonymous to telling the learner what to do.
The tension between providing structure and agency is in an ongoing discussion, but they should not be seen as mutually exclusive. The structure of a learning experience can be both constraining and enabling [14] and by carefully considering the affordances and limitations, one can reconcile them for the benefit of the learner [6].

Interest-Based Learning

“Ben Franklin once wrote: “An investment in knowledge always pays the best interest.”

I’d suggest a twist on this aphorism:

“An investment in interest always pays off with the best knowledge.”

- Resnick, Lifelong Kindergarten [39]

People learn better, work longer and harder, and enjoy more when working on things they care about [38, 40]. They have higher levels of self-efficacy, longer attention span, and they do better with setting goals and using different strategies in areas they are individually interested in [36]. Learners are also more likely to return to an experience and engage with it again if they are interested in its subject, and are more likely to create new connections and develop new ways of thinking [39]. Ito [19] emphasizes two other important aspects of interest-driven learning – the social support from peers and adults, and the ability to connect this learning and interest to the bigger context of the world surrounding them.

To support learners in following their interest, in the Lifelong Kindergarten group we use the metaphor of “wide walls” as a reminder of the importance of supporting many different types of projects, interests, and learners [40]. Scratch was specifically designed to support different types of projects – interactive stories, animations, games, and more. Because learners can use their own images and sounds, they can use Scratch to make a project about anything they care about. Looking at the diversity of projects created with the tool is one way to assess how wide its walls really are [39]. In the Scratch online community, young people share projects about almost any topic. Scratch’s approach to interest-based pathways into coding is part of what make it attract a diverse range of young people [37].

Tinkering

Turkle and Papert [52] discuss the importance of supporting multiple “ways of knowing” in learning environments. They describe the difference between “planners”, who use the traditional, logical way to writing code in a top-down model, and “bricoleurs” that use concrete and personal approaches, building their projects from the bottom up. They argue that while the computer as a tool can support both ways of thinking and working, the computer culture often only values the planners’ way of thinking. Designing tinkerable learning environments can help shift this perspective, making room for bricoleurs as well, and expanding the culture to include groups that otherwise feel excluded. Many times these groups are the people we most want to engage.
Chapter 2 - Background and Related Work

Tinkering builds on this definition of “bricoleurs” and their way of creating. Resnick & Rosenbaum [41] describe it as an approach to making “characterized by a playful, experimental, iterative style of engagement, in which makers are continually reassessing their goals, exploring new paths, and imagining new possibilities”. Karen Wilkinson and Mike Petrich add to it:

Because when you tinker, you’re not following a step-by-step set of directions that leads to a tidy end result. Instead, you’re questioning your assumptions about the way something works, and you’re investigating it on your own terms. You’re giving yourself permission to fiddle with this and dabble with that. And chances are, you’re also blowing your own mind.


Both definitions refer to tinkering as an exploratory, iterative process, which is in opposition to the traditional approach of top-down planning. Tinkerers tend to use a bottom-up approach, which starts with “messing around” with materials, and leads to goals that are emergent and flexible [41]. Supporting tinkering supports creativity, as different types of making and creating are being valued and accepted. However, tinkering is also important for the learning itself, as it shifts it from a mode where everything needs to be explained to a mode where learners discover them on their own.

Two key aspects to supporting tinkering in learning environments are those of immediate feedback and the ability to shift goals easily. Because the process of tinkering is usually composed of many small, quick iterations of experimentation, allowing the learner to quickly see the effect of their change is crucial. As one long-time Scratcher described in Roque & Rusk [46], being able to explore and experiment in Scratch led her to develop a “creative confidence”:

When I'm in other languages, where you are almost too scared to get something wrong and type the wrong thing and be judged, but Scratch it's like playing, it's like chucking things together, if they don't work, that's fine. And being able to make mistakes is part of the thing that develops creative confidence, well for me anyway.

- Erin, 15 years old, in Roque & Rusk, 2017 [46]

Microworlds

“It's by looking at little slices of reality at a time that you learn to understand the greater complexities of the whole world, the macroworld”

- Papert, Microworlds: Transforming Education [30]
Chapter 2 - Background and Related Work

Papert suggested an approach that combines some of these educational theories. He coined the term "Microworld" in the educational context, and refers to it as "a little slice of reality", limited by nature, but rich in opportunities [30]. He used the term "microworld" in *Mindstorms* [31] to describe environments in which children could freely explore, play, and create with a small set of concepts to gain familiarity with them. Like with building blocks or making mud pies, there is no right or wrong answer; you can do what you want within the constraints of the construction kit [30]. Papert believed that acting in those "small reality slices" allows children to control their own learning better, and that over time they would form connections and understand the whole world, "the macroworld".

Papert gave examples of microworlds to enable children to tinker with scientific concepts (such as Newtonian laws of motion) [30], yet he also described a simple microworld that a girl decided to create for herself within Logo. Papert explained that a child named Deborah was feeling overwhelmed by the number of Logo programming commands. Deborah decided to limit her own exploration to use just a small number of commands to make the graphical turtle draw on the screen (using the command to turn right, but not the command to turn left). She experimented with the commands and numbers and figured out ways to use these commands to create projects. At her own pace, she decided to expand her use of commands for further explorations and creations [31]. This shows an example of how a simplified world can allow building confidence.

Over the years, a few other researchers have been building on the ideas of a microworld, and interpreting them in different ways. Hogle defines a microworld as a cognitive tool – an interactive learning environment that is "a conceptual model of some aspect of the real world", usually a simplified version, that allows learners to explore and manipulate the modeled concept [17]. Some researchers have interpreted a microworld as a simulation where children can explore a scientific phenomenon, some interpreted it as a tool to promote problem solving and critical thinking, and some as a vehicle for reflection on design issues [12, 43]. One way to define the difference between a simulation and a microworld is the relation to the underlying model – if novices can begin to understand the underlying conceptual model, it's a microworld [17]. Resnick & Silverman refer to a successful microworld as one where "different kids engage in different design activities ... , but all encounter and use the same underlying ideas as a natural and integral part of the design process" [42]. Rieber [44] suggests a few characteristics of a microworld, among them: a small, but complete subset of the domain; provides an immediate "doorway" for novices to gain immediate access to a domain through experiential learning; promotes problem solving through "debugging."

**Puzzles vs. Projects**

With this set of educational inspirations in mind, we now turn to look at how different tools and learning environments are trying to teach coding to kids. With the growing understanding of the importance of coding education, many new tools are developed, and more young people are
Chapter 2 - Background and Related Work

exposed to coding through them. In this section, I discuss two of the main approaches to coding education for novices – using coding puzzles and using tools for building projects. Then, I use this framework to present a few examples of tools for coding education.

Many of those environments introduce coding through a series of puzzles. They offer a small set of building blocks for the learner to use, set a specific goal for each level, and expose learners to new concepts in a closely scaffolded way. By allowing learners to advance between levels only when they solved the last one, the designer of the experience guides and maintains control over the learning process. While children are definitely learning through these experiences and many find them engaging, it’s hard to achieve fluency in this way. Resnick uses the metaphor of crosswords puzzles – just like you can’t learn how to write by only solving crosswords puzzles, you cannot gain coding fluency by only solving puzzles [38].

An alternative approach, a project-based one, is motivated by the constructionist theory. It usually offers an open-ended environment where children work on personally meaningful projects. The experience usually doesn’t have one right answer, and learners are encouraged to explore within a wide range of possibilities. The learners are in control of their own learning pathway, and they have many building blocks to use in their project.

Table 1 summarizes the main features of both these approaches to teaching coding, and the differences between them.

<table>
<thead>
<tr>
<th></th>
<th>Puzzle-Based</th>
<th>Project-Based</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Focus</strong></td>
<td>solve puzzle</td>
<td>make project</td>
</tr>
<tr>
<td><strong>Outcome</strong></td>
<td>one right answer</td>
<td>no wrong answers</td>
</tr>
<tr>
<td><strong>Potential for Exploration</strong></td>
<td>few possibilities</td>
<td>many possibilities</td>
</tr>
<tr>
<td><strong>Potential for Creativity</strong></td>
<td>single solution</td>
<td>diverse projects supported</td>
</tr>
<tr>
<td><strong>Number of Coding Blocks</strong></td>
<td>a few blocks</td>
<td>many blocks</td>
</tr>
<tr>
<td><strong>Interface Design</strong></td>
<td>simple</td>
<td>complex</td>
</tr>
<tr>
<td><strong>Scaffolding</strong></td>
<td>scaffolded</td>
<td>not scaffolded</td>
</tr>
<tr>
<td><strong>Learning Pathway</strong></td>
<td>designer-led</td>
<td>learner-led</td>
</tr>
</tbody>
</table>

Table 1 – Main differences in design features between puzzle-based and project-based approaches to coding introduction.
Another useful metaphor is suggested by Marina Bers [5], which discusses the difference between “playgrounds” and “playpens”. Playgrounds are specifically designed to allow children to explore, interact with others, and be autonomous. Playpens, on the other hand, allow no room for autonomous exploration or creative options:

The playground promotes, while the playpen hinders, a sense of mastery, creativity, self-confidence, and open exploration

- Bers, Designing Digital Experiences for Positive Youth Development [5]

The puzzle-based experiences are “playpens”, because they leave learners with little autonomy and creative options. Project-based experiences, on the other hand, are aiming to be a playground, where kids can come up with their own ideas and construct their own learning.

In the next sections, I survey a few examples of tools and environments that aim to teach children how to code, using this framework of puzzles and projects. I focus on the Hour of Code [18] activities suggested by each tool. Hour of Code is an initiative started by the Code.org organization, which is "a worldwide effort to celebrate computer science, starting with 1-hour coding activities but expanding to all sorts of community efforts" [18]. Because Hour of Code activities are designed to be completed within an hour, looking at them gives a good overview of the tool's introduction, and can help infer the educational philosophy behind it.

Puzzle-Based Environments

CodeMonkey [9], codeSpark Academy with The Foos [10] and many of Code.org's tutorials [8] use a puzzle-based approach, which can be seen as a “playpen” [5]. In CodeMonkey, learners program a monkey to reach bananas. In codeSpark Academy they collect doughnuts, and in Code.org's Code Studio they choose between different themes featuring popular media like Star Wars, Minecraft or Angry Birds. While the theme is different, most of the tutorials offer a similar experience – a series of mazes, where the learner moves a character around the screen to collect objects. In a few Code Studio tutorials the goal is drawing geometric shapes, but even

![Figure 1 - CodeMonkey (right) and codeSpark Academy (left)]
then each level is a puzzle where the learner needs to figure out how to draw a specific shape. CodeMonkey is intended for children ages 9 and up, Code Studio is for ages 7 and up, and codeSpark Academy targets grades K-5.

Each of these experiences has a level-based structure. CodeMonkey has over 200 puzzles, codeSpark Academy have a few dozens, and most Code Studio tutorials have 10-20 levels. They all start in a similar way – after a short introduction video or animation, the first few levels are used to introduce the learners to basic features and concepts. They use a simple interface, which only shows the few basic features needed. For example, both CodeMonkey and codeSpark Academy start with one coding command, “move” and the first step requires using it to collect an object. As the levels progress, learners have to avoid obstacles in their way to the goal (a banana, doughnut or sheep) by moving around the screen, using commands such as “move”, “turn” and “jump”. As they progress through the levels, they are introduced to new concepts such as loops and functions. In each level, the only coding commands available are the ones needed to solve it. In CodeMonkey and codeSpark Academy, moving to the next level requires completing the previous one, but Code Studio tutorials allow jumping back and forth between the levels without completing all tasks.

Many different methods are used to support the activity. Videos and animations are used to set the scene and provide inspiration and textual prompts are used when a learner fails to solve the puzzle. In many cases, some code is pre-assembled in the editor, either as a full or partial solution to the puzzle. CodeSpark Academy, targeting pre-reading ages, uses visual cues instead of textual ones. For example, a hand icon demonstrates how to drag coding blocks and where to put them, and how to start the program.

In all three tools, there’s a “creative component” at the end of the experience. While not being able to personalize and be creative most of the time, in the last level, learners can use everything they learned to create their own thing:

Congratulations! You've completed all our puzzles! Now mine, build, and create something unique by using everything you've learned so far.

- Code.org, Minecraft tutorial, level 14 (last one)
Chapter 2 - Background and Related Work

In Code Studio tutorials, learners usually get to this last level at the end of the hour, in codeSpark Academy they get there after completing 8 levels, but CodeMonkey suggests going to their creative “challenge builder” after you finished all 250 puzzles available.

All three tools position themselves to educators as activities that require no experience from the students, and from the educator. The tool paces the experience, poses the challenges and collects data about their achievements. For example, in CodeMonkey, assessment is integrated into the experience, with some levels that test learners’ abilities to solve a puzzle, and teachers have access to classroom management tools that show students’ progress.

Project-Based, Open-Ended Environments

Unlike the puzzle-based tools presented in the last section, some tools are using a more open-ended, project-based approach. As discussed in the introduction, Scratch was designed to support learning in an open-ended, project-based, personally-meaningful environment. In Scratch, learners use a graphical programming environment to create interactive animations, stories, and games – and share them with others around the world in an online community [40]. Learners can decide what they want to make and how they want to make it.

The design of the Scratch coding blocks is inspired by the design of LEGO bricks: children snap the blocks together to make their programs come to life. Like LEGO bricks, Scratch is designed to support children learning through the process of experimenting and tinkering. Children decide what they want to make and how they want to make it. They receive feedback from seeing the effects of their actions and through sharing their projects with friends and other community members. Scratch encourages learners to engage in creative learning experiences and express their ideas using code.

A new Scratch project starts with one character and no code, and learners can use any of the 120 available coding blocks right away. Scratch doesn’t offer contextual help, but the short feedback-loop between changing the code and seeing its effects help learners understand the code [40].

While Scratch emphasizes trying out things over careful step-by-step instruction, the Scratch Team is also constantly working on developing supporting learning resources for learners and educators. Like with the other examples, I focus here on the Hour of Code activities, but there are many other resources available as well (such as video tutorials and starter projects). The two main resources used for Hour of Code are the Tips Window tutorials and the Scratch coding cards [48, 50]. The Scratch resources emphasize the support of learners’ self-expression, creativity and agency. They make sure to allow much room for personalization, allows many choices, and leaves the learner in control of their learning process.

The Tips Window is an integrated help tool, which is available within the Scratch coding editor. It includes step-by-step tutorials for 12 different projects, which use textual prompts, images and
Chapter 2 - Background and Related Work

animations [50]. While they take learners step-by-step through the project creation, they also encourage learners to make changes, choose different sprites, and try different things. For example, the "Make It Fly" tutorial starts by saying "You can make anything fly!" and then encourages learners to choose any sprite they want. It also showcases a short video that introduces many different examples where anything from a taco to a Powerpuff Girl is flying.

Learners can browse through the different steps in a theme, or move between different themes while keeping their project. This allows them to pick-and-choose from the suggestions given, and decide what and when to add to their projects.

Figure 3 – The Scratch editor, with the Tips Window open (on the right)

The Scratch coding cards [1, 48], developed in the Scratch Team over the last year, extended and remix the "tips window" tutorials. The set contains 75 cards, which are organized into 10 themes like games, music, art and animation. As described in the overview of the cards:

Each card features step-by-step instructions for beginners to start coding with Scratch. The front of the card shows an activity kids can do with Scratch-like animating a character or keeping score in a game. The back shows how to put together code blocks to make the projects come to life!

- From the Scratch Cards description [48]

Educators, parents and children use these cards as a tutorial or as an inspiration. The careful design of the

Figure 4 - The Scratch Coding Cards
cards minimizes the complexity of every step, while trying to maximize the creative possibilities it encapsulates. Young people who are following the suggestions of the cards will end up with many different projects, and will many times start adding their own ideas to the project.

Another example of a creative, open-ended tool is the video game Minecraft [24]. Even though it was not created as an educational game, its nature as a game of construction and free exploration lends itself to many educational goals. In fact, Minecraft is being used by many educators in their classrooms.

Minecraft is a sandbox game. Similar to a child's physical sandbox, a sandbox game is one without a great deal of structure or direction. A player is instead able to choose how they want to interact with the available choices and content provided by the game. Play is derived from open-ended choice, where players have the freedom to create, build, and destroy as they please.


A special feature of Minecraft is the experience of the first game play. Minecraft does not have tutorials, or any contextual help. It provides two game modes: survival and creative. In survival mode, players need to collect resources and use them to defend themselves from nighttime creatures. In the creative mode, on the other hand, players have access to limitless resources, and can use them to tinker and experiment with building and creating. New players are "thrown into" the game, and need to figure out on their own how to move, collect resources and use them. A culture of YouTube video tutorials and Wikipages has developed around Minecraft and offers another source of support for a novice player – they can get acquainted with the game or specific parts of it by using resources created by the community of players.
Combining Puzzles and Projects

Other tools combine features from puzzles and projects. Khan Academy’s introduction to coding [21] and Snap!’s Hour of Code activity [51] are two of these tools. Khan Academy’s tutorials use puzzles to introduce new concepts, and then let learners use them to create a project at the end. The Hour of Code tutorial offered by Snap! is using a “closed-ended” project-based approach.

Khan Academy’s Hour of Code experience is composed of three types of activities – step-by-step tutorials, challenges, where learners work on projects with a step-by-step instruction (the learner chooses some values, colors and sizes, but is guided by the tutorial on what to change), and projects, which offer more open-ended design activities, with a general prompt like “make a snowman” or “make a self portrait”. The activity introduces short tutorials and challenges interchangeably, and ends with a call for an open project. Learners can choose which project to work on, and can save and share it with others in an online community.

To provide support, Khan Academy’s tutorials include textual prompts when a mistake is detected. They also use semi-completed commands, where a few of the values are pre-filled, and the learner only has to add one or two values. Further support is provided via “hints” for the learner, where at any point they can find the command needed to solve the current puzzle, and some examples of how to use it.

While Khan Academy tries to allow learners to customize and make choices in their projects, this is still done in the context of an instructional model. For example, Figure 6 shows what happens when trying to change a different command than the one referred to by the prompt.

![Figure 6 - An example of a Khan Academy coding challenge.](image)
Chapter 2 - Background and Related Work

While the prompt suggests that the learner should "Go CRAZY!" and play with the size of the face, trying to change the size of the mouth raises an error prompt saying "not THAT crazy!"

![Hour of Code](image)

Figure 7 – Snap! Hour of Code tutorial

Snap! is a generative tool, which can support open-ended projects. However, for their Hour of Code activity, they offer a "closed-ended" project [51]. The tutorial guides the learner through a series of levels, building up towards a game, without many opportunities to personalize or change it. Similar to other tutorials, it starts with one character and a small set of coding blocks. Every level exposes a few more coding blocks and sets a new goal. Because there’s no automatic mechanism that checks the solution, learners can decide to shift away from this goal. But even then, it’s hard to personalize and change the project, as the coding blocks and assets are almost specific to the goal. Learners have more control over the project than in previous examples, and they can move to the next level whenever they want to. However, if learners follow the tutorial, they will all have the same project at the end.

Unlike many other tools, the Snap! activity doesn’t incorporate contextual help. Every level has a textual description of the task, and the only way to get support is looking at the correct solution. The last step in the tutorial exposes the full Snap! Interface, where learners can start using all coding blocks to personalize and advance the project they created before.

These last two examples show a few attempts to combine features of project-based and puzzle-based approaches, but the ability to be creative in both is still limited. The next chapters will present a different model that uses features from both approaches.
Chapter 3

Research Design

Research Question
My work is inspired by the core themes described in the last chapter – constructionism and instructionism, scaffolding and agency, interest-based learning and Papert’s microworlds. Drawing on the connections and tensions between these central themes, my work is guided by a central research question:

How can we design constructionist introductions to coding?

In the search for how to tackle this big challenge, I looked back to these core ideas, and integrated them with my primary educational goal – to support young people’s development of computational perspectives, rather than focusing primarily on computational concepts [7]. Through my explorations and efforts to reconcile sometimes opposing values, I refined the tensions and challenges into three main questions that guide my exploration:

- How can we simplify the experience while supporting creativity?
- How can we add scaffolding while keeping learners’ agency?
- How can we provide starting points that spark rather than limit the imagination?

I explore these questions by designing a constructionist introduction to coding – Scratch Microworlds.

Research Design and Methodology
To explore my research question, I used an iterative design process and designed Scratch Microworlds – simplified versions of the Scratch coding editor, where learners can work on interest-based, open-ended projects in a scaffolded environment.

The process of designing, implementing, evaluating and reflecting on Scratch Microworlds was supported by members of the Lifelong Kindergarten group and the Scratch Team. I worked together with other researchers from the team, interns from the Harvard Graduate School of Education and Wellesley College, and community partners in the sites where we conducted testing sessions. Together, we discussed the design, collected observations and analyzed them.

Our process used a design-based research approach from the Learning Sciences [4]. In design-based research, through an iterative process of designing, testing, and analyzing, the design is developed and new theories are generated. An important aspect of design-based research is
that while it is trying to solve a problem within a specific context, the theories generated could have an impact in other contexts as well.

We started each iteration with a design phase, where we addressed specific issues that came up in the last iteration by designing either solutions for them or experimentations that will help us enrich our understanding of the issues. With this design in mind, the next phase was implementation, where we changed the prototype to include those design changes. Then, we evaluated the prototype by conducting testing sessions with children and educators. These sessions provided us with more data and observations, with which we could reflect on the current design and our assumptions about it.

Throughout the process, we were interested in two aspects of the project – one is the design choices and principles that create a successful experience, and the other is looking at the experience of the learners. To explore the design choices, I kept design notes from our discussions, and collected observations of learners using the prototype. To look closer at the experiences of the learners, we also collected the artifacts they produced (Scratch projects), logged data about the patterns of activity (for example, time spent in each microworld) and their feedback. We also interviewed some of the participants, to learn more about their experience and their feedback, asking questions like “what was your favorite part?”, “what was hard?” and “what would you want to try next?”. We also used recordings of their screen activity.

We used the collected data to inform our design choices, to develop a set of design considerations, and to develop an understanding of the experience of the learners. In the next chapters, I’ll use this collection of design choices and the learners’ experience to describe what we learned from this research. This will also demonstrate the advantages and disadvantages of following a constructionist approach to introductory experiences.

Evaluation
To learn more about the different ways learners use microworlds and the effect it has on their learning, we conducted a few sessions of field-testing the microworlds prototypes with educators and children. Approximately 90 educators and 100 children participated in our workshops and sessions. A complete list is available in Table 2.

Most of the children that participated in the sessions were in middle school and had little to no experience with Scratch. Some sessions were drop-in sessions, where children could join and leave whenever they wanted to, and others were conducted in a workshop setting. In the drop-in sessions, children spent anywhere between 5 minutes and 45 minutes with us, and the workshop sessions were 45-90 minutes long. In both cases, we started with a very short introduction of the microworlds, and the participants spent most of the time creating with one or more microworlds, which they could choose based on their interests. Every workshop session ended with a “sharing” part, where participants could share their created projects with others in
The educators that participated in our workshops had a diverse set of experiences – some had extensive Scratch experience and have been teaching with it, while for others this was the first introduction to Scratch or coding in general. Their participation was usually voluntary, and many chose to join the workshop because they wanted to learn more about Scratch and different ways they could use it with their students. Educator workshops focused on getting their feedback and learning about their perceptions towards the idea of Scratch Microworlds. While educators always got a chance to create with the microworlds, the discussion focused mostly on how they could imagine using microworlds with their students and in which ways. We also asked educators about improvements needed, and inquired which other themes would be useful for them.

In the next chapters I’ll present some of the observations we collected and the changes we made to the microworlds based on the data collected.
<table>
<thead>
<tr>
<th>Location</th>
<th>Participants</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idea Hub, MIT Museum (Dec, 2015)</td>
<td>8 children</td>
<td>Drop-in</td>
</tr>
<tr>
<td>Scratch Conference, MIT Media Lab (Aug, 2016)</td>
<td>30 educators and researchers</td>
<td>90-minute workshop</td>
</tr>
<tr>
<td>Digital Media and Learning Conference (Oct, 2016)</td>
<td>30 educators and researchers</td>
<td>90-minute workshop</td>
</tr>
<tr>
<td>Creative Learning Network Event, MIT Media Lab (Nov, 2016)</td>
<td>20 educators</td>
<td>120-minute workshop</td>
</tr>
<tr>
<td>Girls Day, MIT Museum (Nov, 2016)</td>
<td>20 children</td>
<td>Drop-in</td>
</tr>
<tr>
<td>Scratch Day Event, The Computer School, New York City (Dec, 2016)</td>
<td>20 children and their parents</td>
<td>45-minute workshop</td>
</tr>
<tr>
<td>Educators Night, MIT’s Teaching Systems Lab (Feb, 2017)</td>
<td>8 educators</td>
<td>45-minute workshop</td>
</tr>
<tr>
<td>Coding Club, Harvard Ed Portal (Feb, 2017)</td>
<td>8 children</td>
<td>120-minute workshop</td>
</tr>
<tr>
<td>Coding Club, Area Four Youth Center (Mar, 2017)</td>
<td>3 children</td>
<td>60-minute workshop</td>
</tr>
<tr>
<td>Girls Day, MIT Museum (Mar, 2017)</td>
<td>30 children</td>
<td>Drop-in</td>
</tr>
<tr>
<td>The Clubhouse @ Jordan Boys &amp; Girls Club, Chelsea (Mar, 2017)</td>
<td>5 Children</td>
<td>60-minute workshop</td>
</tr>
<tr>
<td>Middle school, Roxbury (Mar, 2017)</td>
<td>24 Children</td>
<td>90-minute workshop</td>
</tr>
</tbody>
</table>

Table 2 – An overview of testing sessions
Chapter 4

What are Scratch Microworlds?

Main Design Features

To create Scratch Microworlds, we looked at the different approaches taken by puzzle-based and project-based learning experiences. We were motivated by the potential of adapting the Scratch experience to include some of the puzzle-based elements. We hoped that it would make Scratch easier for beginners, while maintaining the values of learning by creating. Table 3 summarizes the main features of both these approaches (listed also in Table 1) and shows the features we chose for Scratch Microworlds from each approach.

<table>
<thead>
<tr>
<th></th>
<th>Puzzle-Based</th>
<th>Project-Based</th>
<th>Scratch Microworlds</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Focus</strong></td>
<td>solve puzzle</td>
<td>make project</td>
<td>make project</td>
</tr>
<tr>
<td><strong>Outcome</strong></td>
<td>one right answer</td>
<td>no wrong answers</td>
<td>no wrong answers</td>
</tr>
<tr>
<td><strong>Potential for Exploration</strong></td>
<td>few possibilities</td>
<td>many possibilities</td>
<td>many possibilities</td>
</tr>
<tr>
<td><strong>Potential for Creativity</strong></td>
<td>single solution</td>
<td>diverse projects supported</td>
<td>diverse projects supported</td>
</tr>
<tr>
<td><strong>Number of Coding Blocks</strong></td>
<td>a few blocks</td>
<td>many blocks</td>
<td>a few blocks</td>
</tr>
<tr>
<td><strong>Interface Design</strong></td>
<td>simple</td>
<td>complex</td>
<td>simple</td>
</tr>
<tr>
<td><strong>Scaffolding</strong></td>
<td>scaffolded</td>
<td>not scaffolded</td>
<td>scaffolded</td>
</tr>
<tr>
<td><strong>Learning Pathway</strong></td>
<td>designer-led</td>
<td>learner-led</td>
<td>learner-led</td>
</tr>
</tbody>
</table>

Table 3 – Scratch Microworlds features, inspired by puzzle-based and project-based approaches
Chapter 4 - What are Scratch Microworlds?

The microworlds keep the main design features of Scratch – open-ended, learner-led, project-based approach, where children have many venues for exploration and are able to create many types of projects. However, the microworlds incorporate the ideas of a simple interface, a small number of building blocks and the additional scaffolding from the puzzle-based approach.

This results in simplified versions of the Scratch editor that include a subset of the programming blocks and are focused on specific topics like music, dance, or soccer. They support learning in an open-ended environment, where the learners can build a variety of projects and control their own learning pathway. They are also interest-based, so learners can choose to work on a topic that is personally meaningful to them. For example, a learner who is interested in dance can use a small set of relevant blocks to choreograph a dance, while another can design her own interactive house using a different microworld. To prevent the experience from being overwhelming, each microworld only shows a few programming blocks, and allows the learner to access more blocks and options at their own pace. It also has a reduced set of features, excluding options such as drawing a new sprite (character) or recording a sound.

Environment and Content

We use the term Scratch Microworlds to talk about two components, environment and content. While it's possible to look at the design of each separately, it's more valuable to look at them together, as learners experience the combination of both. The environment component refers to the new version of the Scratch editor, which has special features such as the ability to show a small number of blocks and to incrementally expose more. With this one environment, a learner can load many different themes, and their content is the second component. The content component refers to the specific microworld template-projects we designed and implemented. Each template-project is a theme-based collection of assets (visual images for the sprites and background and sounds) and a curated and ordered set of blocks. For example, the dance microworld has a few images of dancers in different poses, a few music segments to choose from, and a subset of blocks that allow tinkering with dance and music. Sometimes it also includes some starter code – blocks that are already set up in the editor when the project is loaded, so that learners can immediately start interacting with it.

Throughout the design process, we iterated on the design of both components. Some iterations focused more on changes to the general environment, which affects all the different microworlds, and some focused on changes to the details of each theme-based template-project.

Design Process and Iterations

The core of this work is the design process of Scratch Microworlds. Before looking more closely at specific features of the design, which will be addressed in part 2, this section provides an overview of the general process. It describes the different prototypes created, and briefly discusses the influence of our observations on the transition between them.
Chapter 4 - What are Scratch Microworlds?

The Scratch Microworlds project is part of the "Coding for All: Interest-Driven Trajectories to Computational Fluency" initiative, funded by a National Science Foundation grant [26]. The project aims to "expand the reach of Scratch in groups that have been traditionally underrepresented in computing" [37]. It builds on children's existing interests and social identities to engage them in Scratch across a broader range of interests.

With this aim in mind, the first prototype for Scratch Microworlds (see Figure 9) focused on creating new pathways into Scratch, especially ones that would engage new audiences. For that reason, it was focused on developing content for microworlds - starting with hip-hop dance and fashion microworlds. The environment was a simplified version of the Scratch 2.0 editor, which did not have a palette of coding blocks, and only offered a small set of blocks scattered around the scripting area. This allowed learners to quickly start connecting the blocks in different ways and change the values in them. The sprites and costumes were also pre-selected, and the ability to add new ones was removed. In addition, the editor was stripped of other non-essential features, such as: the paint and sound editors, and buttons for direct manipulation of the sprites. These microworlds were embedded in a webpage dedicated to the topic – dance or fashion, which included tips on how to get started, an inspirational video, and ways to connect with the Scratch community. To expand the variety of microworlds, we worked with the Scratch Team and Lifelong Kindergarten members, and other educators, on conceptualizing their own ideas for microworlds themes. Out of the many concepts explored, we refined and developed five new themes – music, humor, interactive house, soccer and art.

When we started testing this prototype, mostly with educators, the feedback uncovered some challenges. Not having a palette of blocks made the interface look more cluttered and complicated, and learners were limited by the number of blocks on the scripting area. Combined with a small number of blocks, this increased participants' feeling that they need to use all
blocks in their project, and that there is a right answer they should be looking for. While educators were generally excited about the promise of the microworlds approach and talked about how it could help them introduce Scratch to their students, two additional themes were brought up by the educators – feeling too constrained by the experience, and wanting to create more microworlds on additional themes – ones that will fit the interest topics of the learners they are working with.

Based on this feedback, we made the next version of the microworlds with a blocks palette (see Figure 10). To scaffold the experience of learning to use blocks, only a few blocks were visible in the palette, and we tested a few different ways for the learner to uncover more blocks at their own pace. We also experimented with different ways to provide some basic contextual help, supporting learners in their first steps with Scratch – understanding how to drag and connect blocks.

Two other ideas we experimented with in this version were limiting the microworlds to one sprite only and providing initial functional starter code in some of the microworlds. When we wanted to test this prototype we found out that some technical limitations made it hard for the participants to work with it. The prototype was built using a partial Scratch-like editor usually used for rapid prototyping, and for that reason many features of Scratch were not implemented in it. While we were able to learn from the way people interacted with the prototype, it pushed us to think about a better tool to create the next prototype with.
Chapter 4 - What are Scratch Microworlds?

Figure 10 – Scratch-like editor with scaffolding. Gymnastics microworld with category-based scaffolding (top), and music microworld with a “show more” button (bottom).

Encouraged by the reactions to the scaffolded palette, and concerned with the limited feeling created by the one-sprite projects, the next prototype (see Figure 11) focused on refining the learner experience of uncovering more blocks as well as trying to simplify multi-sprite projects in ways that will make it less confusing. For example, in this prototype, while a few sprites exist, they all have the same coding palette. This iteration also included a big change in infrastructure, when we moved to development on top of the Scratch 3.0 project, which allowed for a better user experience, but created other challenges that were caused by missing features in Scratch 3.0, which was still in development.

For the next and last iteration on the prototype, the focus was mostly on the content. On the content side, we made a series of changes to each of the microworlds, based both on specific feedback we collected on each microworld, and general feedback that applied to all. We added more visual and auditory assets to the projects, replaced some of the coding blocks that had caused confusion, and refined which blocks were shown and in which order. On the
environment side, we added small visual cues that help learners navigate the interface and better support for migrating the projects into a full Scratch editor.

Figure 11 – The last prototype, based on Scratch 3.0 – art (top) and music (bottom).
Part II  Design and Observations

After part 1 introduced the context for the Scratch Microworlds project, and described the iterative design process used to create them, this second part goes into the design process and findings in more depth. In part 2, I discuss the design process using three different perspectives: simplifying while supporting creativity; scaffolding while keeping learners’ agency; providing starting points that spark the imagination. Each one of these chapters describes a core tension that guided our search for a better newcomer experience. Each chapter describes the design decisions we made, our observations of people using the microworlds and how these observations affected the improvement of the microworlds through an iterative design process.
Chapter 5

Simplifying while Supporting Creativity

Design Challenge #1: How can we design an environment that is simple enough for beginners, but allows them to express their creativity and make projects that are meaningful to them?

Learning to use a new environment is easier when it is simple, with few options and choices to make. But such a constrained environment is also usually less engaging and it makes it harder for learners to express their creativity. With Scratch Microworlds, we sought to design a simpler version of Scratch that still supports learners in constructing personally meaningful projects.

While Scratch is easier to learn than most other programming languages, the first experience with it can be quite overwhelming for learners [37]. The default view exposed to learners includes ten different code categories, about 15 coding blocks, and many other features and buttons. In addition, it shows the default sprite, the Scratch Cat, on a white background. Overall, the Scratch editor offers almost 120 coding blocks for children to create with, and about 20 different tools they can use.

Puzzle-based experiences such as Code.org and CodeMonkey use a different approach. They start with a limited set of features available for the learners. However, limiting the features comes hand-in-hand with an overall limited experience.

The difference between these experiences is like the difference between “dipping your toes in the water” and “jumping in feet first”, as was described by one of the graduate students in our Learning Creative Learning class. While we want children to be able to jump into the water of creating with Scratch, some children will feel more comfortable with being able to test the water before jumping right in. To balance the support for creativity with the need for a simpler starting point, we decided to adopt and adapt the simplified interface of the puzzle-based experiences and combine it with the open-ended approach of Scratch.

While creating a limited experience seems to be opposed to a creative one, the two are not mutually exclusive. In fact, if done right, adding more structure to a creative experience can actually support creativity rather than hurt it [6]. With Scratch Microworlds, we actually saw a few examples of children using some of the limitations to expand their creative process. Trying to balance the two aspects is challenging, and in the next sections I'll describe a few of the design considerations and principles we developed throughout the process. We focused on a few areas: reducing the number of coding blocks, simplifying features and tools, simplifying the number of sprites, and supporting tinkering as a pathway to creativity.
Reducing the Number of Coding Blocks

In Scratch, the space of possibilities drawn by the 120 blocks is part of what makes it a success – by combining them in different ways, children can create many different projects, from basic ones to very advanced ones. This allows for “wide walls” and “high ceiling”. The order of the blocks, the fact that they connect only in ways that make syntactic sense, and the visual cues given by their color and shape support a “low floor” as well [40]. However, this vastness can still leave novice learners confused and overwhelmed. To try to address that, we started the design of Scratch Microworlds by thinking of ways to reduce the number of coding blocks.

Many other introductions to coding choose to expose beginners to a small set of blocks, one which usually grows bigger and more advanced with the learner advancing through the levels of the experience. But, their rigid structure, where every level has one predetermined goal, limits the possibilities for learners to be creative and express themselves. While it might be an easier way to get started, we believe it’s harder to keep learners engaged and motivated for long in this way.

With Scratch Microworlds we tried balancing a small number of blocks with keeping as much of the creative freedom as possible. The next subsections detail some of the ways we experimented with to achieve this balance.

General-Purpose vs. High-Level Abstraction Blocks

One of the earliest decisions we had to make was which blocks to use – the available, general-purpose Scratch coding blocks, or new, higher-level abstractions of blocks. For example, a “jump” block that will make a character jump on the screen would be a higher-level abstraction, while getting the same result with the Scratch coding blocks would require connecting a few “move” and “turn” commands or “change y by” blocks. Many puzzle-based experiences use higher-level abstractions, providing blocks that are tailored to specific puzzles, like “catch fish” or “plant flower”. Different levels can have different blocks, to allow the learners to focus only on the specific concepts that are needed for this puzzle. While high-level abstraction can make it easier for the learners to get started, they can also create misconceptions and confusion when they do not create a coherent and consistent vocabulary – like when learners are looking for blocks they used in a previous level but can’t find it in current level. For that reason, there is an advantage to keeping the general-purpose blocks, which allows building a basic vocabulary that could lead to fluency. When learning to use the Scratch blocks, like learning the ABCs, you get the fluency needed to express yourself in many different ways [38]. Keeping the same vocabulary between microworlds and Scratch also makes the transition into the full Scratch editor easier, and reduces the discrepancies between the two experiences.

While we believe there’s value and potential in both approaches, we decided to keep the general-purpose Scratch coding blocks, helping to develop learners’ fluency with using Scratch
and making the transition to Scratch smoother. This is especially important because we see microworlds as a first step, which leads into the full Scratch editor eventually.

Choosing the general-purpose blocks means that we have to find other ways to help learners understand how they can build higher-level functionalities. One way to do that, which will be discussed in more details in the next chapter, is using starter code — pre-assembled coding blocks that are already there when the learner starts their experience. While it allows the learner to use some non-trivial functionality right away, it doesn’t create a “black box” like the higher-level abstractions — the learner can tinker and change the preset code and develop an understanding of how it works.

Selecting Expressive Subsets of Blocks

A second consideration was which subset of blocks will be the optimal one for the microworlds experience. Recognizing that different themes require different “building blocks”, we decided to create a specific subset of blocks for each microworld, based on its theme. While this requires extra design work for each microworld, it allows for a more coherent experience for each theme. This also allows the learners who are exploring different microworlds to explore different blocks and different ways to use them.

With the goal of curating a small subset of blocks, which will still allow diverse results, we started by creating Scratch projects around each of the themes. From these projects, which included many blocks, we created simplified versions, iteratively minimizing the subset of blocks as much as we could. Then, based on feedback and observations from testing, we reflected on the choice of blocks and changed them when needed. Changes included replacing one block with another (For example, “play sound until done” with “play sound”), adding blocks that learners wanted to use, and removing ones that were never used or were confusing. We also changed some of the default values of the blocks, and the order in which they are presented.

We learned that the best blocks to include in a microworld are ones that are intuitive for novice learners, but at the same time maximize the variety of potential projects. For example, blocks that have a visible effect when clicked are easier to understand, and blocks that can be used in many different ways create variety. Blocks that have a dropdown menu are a good example, because they contain many choices in one block. Through our observations we learned that some blocks work better than others. For example, blocks that caused a surprising or unexpected result were very engaging for the learners.

As for the group of blocks chosen, good combinations are ones that can fit together in many interesting and meaningful ways. We tried to choose a variety of blocks, from different categories, and test the space of possibilities they create. For example, with just “play sound” and “wait” blocks you can create a whole musical performance, but there is less you can do with only “change size” and “wait” blocks.
Another factor that affected engagement with the blocks is the specific assets (images and sounds) associated with the project. Because many of the blocks in a microworld use and manipulate the assets, we learned that it is very important to adapt the assets to the blocks and theme as well. For example, the "change color" block doesn't affect black and white pixels, so for it to be interesting, costumes can't be just black and white. Furthermore, for the "play sound" block to be engaging, many sounds should be loaded, and to give the feeling of creating a real dance, a sprite must have many different dance moves (each represented as a costume). This is even more important in a microworld than in other Scratch learning resources, as learners cannot add or change the costumes and sounds within the microworld.

While we were focused on creating the optimal subset of blocks that balanced the theme with creative options, we noticed that many learners didn't follow our imagined themed scenarios, but created their own types of projects. We were excited to see them pushing the limits of the microworlds and expressing their own ideas. A few examples include using the interactive house microworld to make music, the music microworld to create a visual interactive experience where the user controls visual effects, and turning the art microworlds into a simple version of drawing software (see Figure 12 for two examples of such projects). We also noticed learners using the same subset of blocks in many combinations and variations, to create different outcomes. By creating different sequences of blocks, using different modes of interaction and choosing between the different assets, learners were able to create personalized and meaningful projects they were excited about. We were happy to see these results, and believe that this is the result of the careful process of choosing the blocks, and making sure that they allow extensions and modifications.

Number of Blocks

How many coding blocks would create the best initial experience? While it's clear that the 120 blocks Scratch provides could be overwhelming, using a set which is too small could lead learners to misconceptions about what could be achieved with this tool. Furthermore, a small
set of blocks can enhance the feeling that there’s only one right way to use them, and that all of the blocks need to be used.

With our first prototype, which didn’t have a block palette but had the different blocks scattered in the coding area, educators told us that they felt like the microworld was actually a puzzle that needed to be solved. The small number of blocks, and the fact that they were already scattered there led people to believe that they had to use all blocks to create their project (because otherwise it’s not clear why we added them there at all). They also felt very limited in their ability to explore. Even when we added the palette back it was apparent that a very small number of blocks is not as engaging and motivating as a bigger one. With this understanding, we made sure that each microworld offers at least ten different blocks. To help learners navigate their way at the beginning, not all blocks are visible right at the start, but learners can choose to expose them at any point.

While simplifying the blocks was a main part of the effort to balance simplicity and expressiveness, it was not the only one. Next, we turned to look at other aspects of the novice experience that could be simplified.

Simplifying Features and Tools
Similar to the big number of coding blocks, Scratch offers many features and tools for the learners to use – a paint editor where learners can create and change their sprites’ looks, an audio editor, different ways to add new sprites and change their size, and more. Learners use these features to extend, customize, and personalize their projects. This process plays an important part in creating with Scratch. Nevertheless, for the novice learner, the variety of tools and buttons can increase the confusion and make it harder to find their way at the beginning.

In Scratch Microworlds, we removed most of these features and tools, trying to keep a minimal set of basic features. Instead of being able to draw, import, or choose a sprite from a library, each microworld has a pre-loaded set of visuals and sounds for the learner to work with. We also completely removed the ability to code the background and to import or record sounds, and removed other shortcut buttons to different actions like making the sprite bigger or duplicating code stacks.

While this clearly simplified the interface and helped learners navigate to basic functionalities, adding your own assets was the most requested feature by both educators and children who used the microworlds. Some wanted the ability to personalize the project by using their own voice and images, while others were just dissatisfied by the limited choices given by the microworld. For example, while one participant talked about wanting to add their own favorite song to the dance project, another just wanted to have a some beat box sounds in the music microworld. This fits findings from prior research on Scratch, which shows that the ability to personalize is an important motivator for people [47].
Chapter 5 - Simplifying while Supporting Creativity

Some of the benefits of not being able to add their own sprites were subtler. First, we noticed participants were more willing to discard their work and start over when they were not emotionally attached to the sprites. Willingness to throw something away increases the chance that learners will tinker and try different things, because they are not worried about ruining something they care about. Participants also spent more time tinkering with the coding blocks, as they couldn’t tinker with the look of the sprites. A few educators mentioned that one of the reasons they didn’t introduce Scratch to their students is that they sometimes get so caught up with creating their sprite that they spend most of the class time on it.

To address the requests for more customization without adding the ability to add a sprite, we added more default costumes and sounds for learners to choose from. In the last prototype we replaced the default “pop” sound with a list of more than ten different sounds, and added more costumes to many of the sprites. However, this issue is one we’re still experimenting with, trying to find a better compromise that will allow learners more freedom to customize while not adding to their initial confusion.

One way to counterbalance this limiting effect is creating many microworlds for the learner to choose from. Then, instead of choosing their sprites and sounds, they can choose which microworld to use based on the sprites that engage them most. Another way, which we explored only conceptually so far, is to let learners let learners choose from a curated set at the beginning of the experience. After choosing the music microworld, for example, and before the experience starts, the learner will be able to choose their sprite from a small library of musical instruments. In future iterations, we plan to explore those directions and others in search for a better balance.

Simplifying the Sprites

In Scratch, each sprite is a programmable object. It has its own set of visuals and sounds, and its own code. Within the full Scratch editor, working with multiple sprites is a known pain point for beginners, because it requires that they understand when and how to switch context from one sprite to the other.

Limiting the microworlds to one sprite makes them simpler. It reduces the cognitive load, and doesn’t require understanding how to move between the sprites and how to code the interactions between them. However, projects with only one sprite limit the possibilities they provide – there’s less you can do with a guitar than with a whole band. Multiple sprites provide a more engaging starting point and better spark the imagination.

Different iterations of microworlds tackled this issue in different ways. In the first prototype, most microworlds had 2-3 sprites, and each had a different subset of coding blocks. But both educators and children found this confusing. They were not sure how to switch the view to another sprite, and were confused by the need to add blocks to each sprite separately. Upon reflecting on this design choice, we realized that understanding this subtle point is even harder...
Chapter 5 - Simplifying while Supporting Creativity

with this microworld than it is in Scratch. In the full Scratch editor, the learner needs to make an explicit decision to add another sprite and locate where to click to do it. At this point, the learner already knows where to look for the context switch, and when adding a new sprite the context actually switches automatically to the new sprite. In contrast, in a microworld the multiple sprites are pre-loaded which means the learner can’t use any such external cues to understand the underlying model.

As a result of this feedback, the next prototypes focused on single-sprite microworlds. We were challenged to push the boundaries of what can be created with only one sprite, trying to make sure the experience stays engaging and motivating. We used a few strategies, like creating one sprite with different costumes, using themes that are highly attractive for children like Pokémon, and using images of role models like Simone Biles in the gymnastics microworld.

But having only single-sprite microworlds gives way to a misconception of the space of opportunities Scratch provides. For novice users that use Scratch Microworlds as their first experience, the microworlds represent the full scope of possibilities. So when they are only exposed to experiences with one sprite, they might grow to believe that this is all Scratch has to offer. This in turn could affect their ability to imagine what they want to create with Scratch, and lower their level of motivation.

In the last versions of the prototypes we have a combination of some single-sprite and some multi-sprite microworlds. In these versions the blocks palette is the same for all sprites, which addresses some of the confusion expressed before. This, together with some changes in the visual indicators of moving between the sprites, reduces some of the confusion over switching between sprites.

While we still see participants engage with both types of experiences, we believe that presenting them with this combination helps solve the perception problem, because they can see that projects with multiple sprites are possible. Participants choose to use a mixture of the single and multi-sprite experiences. The most popular microworld (art) has only one sprite, and it is followed by another single-sprite microworld (gymnastics), but then two multi-sprite microworlds (music and interactive house). Even within the music category, which has one single-sprite and one multi-sprite microworld, participants are split – the multi-sprite version is more popular, but the single-sprite one gets a lot of use as well. There doesn’t seem to be a difference in participants’ behavior between the two types of microworlds.

Looking at data from field-testing emphasized both sides of this tension. On one hand, participants were really engaged with coding a full musical band or making their house interactive. On the other hand, many participants still found it hard to code a few sprites right away, and did better when there was only one. Future research could focus on finding other ways to make it easier to move between sprites and on making more single-sprite experiences that are engaging and motivating.
Tinkering as a Pathway to Creativity

We designed the microworlds to provide a good starting point for tinkering. The limited set of blocks and features helped children get started by tinkering with the coding blocks. As mentioned in previous sections, we took special care when choosing which blocks go into each microworld and chose blocks that support a variety of behaviors, and ones that can be connected in many different ways to create different things. Unlike many puzzle-based experiences, in Scratch Microworlds there's no step-by-step instruction, and no pre-set goal. Both of these factors allow learners to set their own goal, adapt them, and reassess them.

And indeed, we saw many young people in our workshops starting the process without a clear goal of something they wanted to create. By experimenting with the different blocks and seeing their effects on the sprites, they started developing specific goals and tried to achieve them. If at some point something unexpected caught their attention, they sometimes changed their goal, and started pursuing a different direction. Others spent the whole session exploring, or built a narrative around the project they created by tinkering.

One issue we struggle with, which is also a known issue in Scratch, is how to encourage the learners to test their scripts more often. People tend to create long scripts without trying them out, which makes it harder for them to understand what is happening once they do run it. The art microworlds provides an example for one way to overcome this issue and supporting tinkering. The art microworld starts with a continuous loop, which means when learners add a block into it they could immediately see the effect of the block as it is repeatedly run in the loop. This makes it easier for learners to understand the function of the block, and we believe that it's one of the factors that make this microworld so popular.

Seeing how children and educators created projects with Scratch Microworlds, we were encouraged by the potential of balancing simplicity with support for tinkering and creativity. In the next chapter I'll describe a second area of exploration: adding more scaffolding for the learner throughout the experience, without sacrificing learners' agency.
Chapter 6

Scaffolding While Keeping Learners' Agency

Design Challenge #2: How can Scratch provide scaffolded learning experiences while supporting learners' choice and agency over what they create and how they create it?

To make it easier for learners to start creating with Scratch, with Scratch Microworlds we provide different scaffoldings for the learning process. Similar to many of the puzzle-based tools for coding introduction, we gradually expose the learners to some of the features. We also adopted and adapted other methods from the puzzle-based approaches to support the process. These are both explicit (like contextual textual “tips”, or highlighting different parts of the interface) and implicit (like using starter code as an example which models behavior).

But unlike many of the puzzle-based tools, we designed microworlds to enable learners to freely explore and create. Papert emphasized this aspect of microworlds as “perhaps the most important aspect of them: that they create better and richer conditions for children— and for others, grown-ups as well—to take charge of their own learning” [30]. Research on motivation has also demonstrated the value of supporting learners' autonomy to foster deeper learning [25].

Designing for closer support is sometimes in tension with designing for learner agency [7]. In Scratch Microworlds we tried to balance and find ways to reconcile the two. By adding more scaffolding features we limited some of the control offered to the learners in Scratch, but balanced it by adding features that support agency, and letting learners control the scaffoldings. We also made sure that learners could always make meaningful choices about their learning process and their projects. In the next sections, I’ll describe the features we added to create more scaffolding and our efforts to maintain and improve learner agency.

Scaffolding

Many educational approaches agree that scaffolding can support learning, but each calls for a different type of scaffoldings. As discussed in the background, scaffolding assumes different forms in an instructionist and in a constructionist learning environments, both in-person and in a digital experience. While instructionist scaffolding usually supports learners in progressing along a predetermined path, constructionist scaffolding needs to support their process as they diverge in different directions and while working on open-ended projects.
Focusing on digital coding environments, an instructionist model for support can include designing specific prompts for specific situations, and deciding which concepts and ideas should the learner be exposed to at every step of the way. For example, in puzzle-based experiences, the designer can design 20 levels, each exposing the learner to a specific new idea, with transition between the levels happening only if the learner solved the puzzle, supposedly demonstrating some understanding of the solution. If learners are wrong, textual prompts can guide them to the right answer, as the possible errors and misconceptions can be mapped out in the design phase. Creating support structures that fit the constructionist approach in project-based environments is more challenging, because of the variety of paths learners may choose to follow. When the learner sets the goal, it is harder to plan which type of support will be helpful in which scenarios.

In Scratch Microworlds, we use scaffolding to support learners both within the microworld experience and in their transition into the full Scratch editor. To support both of these processes, we focused our efforts in two areas of the experience: the blocks palette and the coding area.

In the blocks palette, we created mechanisms to gradually expose more blocks. We experimented with different ways to achieve that and took special care to leave as much control as possible with the learner. In the coding area, we experimented with pre-loading coding blocks so that learners will have something to start building upon. We also experimented with different ways of providing contextual help for the process of creating within a microworld. While the last iterations focused less on contextual help, we still believe that it has great potential for making the experience better. While most of these changes could be categorized as a “channeling and focusing” type of scaffolding, the use of starter code and some of the prompts we tested as contextual help are ways of modeling advanced constructs [34].

The Blocks Palette
As the number of coding blocks is one of the most overwhelming features in the Scratch editor, we were looking for ways to allow learners to start with a very small subset and grow it as the learner starts feeling comfortable with it. We discussed the design principle behind reducing the number of blocks in the palette in the last chapter, and here focus on the design’s support for a gradual exposure to the coding blocks.

We started with a prototype that didn’t have a coding palette, and in which all blocks were visible when starting in the coding area. In this prototype, no internal scaffolding was provided, but the microworld itself acted as a scaffolding step into the Scratch editor.

In the next prototype we wanted to do more, and provide some scaffolding within the microworlds experience itself. One of the reasons for that was our desire to expose the learners to blocks that are more advanced. When there’s no gradual exposure, using advanced blocks creates confusion for the novice learners because they can’t distinguish which blocks will be easier to use first. Gradual exposure allows them to start with basic concepts, and move on to
advanced ones when they are ready for it. For example, we wanted to create a game-starter microworld, but building a game requires using some advanced interactions. We couldn’t make it easy enough in the first prototype, but as we started gradually exposing blocks, it became possible. Then the learners can start by developing the setup for their game, like making the sprites move, and only later add the conditionals needed for game interactions.

To do that, we added an additional button to the palette. The starting setup had only 2-5 blocks and when clicked, this button would expose a few more blocks, in a predetermined order curated to match each microworld. While the function of this button wasn’t always clear for the learners, once it was explained or discovered they liked using it to “get” more blocks to experiment with. Sometimes they would click the button once and try out all of the blocks and sometimes they would click it a few times to expose all available blocks before getting back to building. Learners also expressed the need for a “back” button, that would remove the last set of added blocks, which strengthen our belief that having this smaller set allows the learners to feel more comfortable with creating.

Building on the feedback received, the next prototype included a set of buttons that were easier to understand. In addition to the “show more” button, a “show fewer” button was added and it allowed learners to go back and forth with adding and removing blocks from the palette. The icon was replaced with text, and the buttons were moved from the bottom of the palette to the top. Based on feedback from more testing, we also changed the color of the button to gray, so it would be more neutral and distinguished from the blocks. Many participants were still confused by the buttons and didn’t realize that they could click the “show more” multiple times. To help make it clearer, we added a visual cue for when a button is clickable by graying it out when there were no more blocks to show (or no more blocks to remove). Figure 13 presents

![Figure 13 - Scaffolding with Categories](image)

![Figure 14 - Evolution of the “show more” button, from the early prototype on the left to last one on the right.](image)
examples of the coding palette in the different iterations.

We also iterated on the exact order and grouping of the blocks. At first, every click exposed 3-5 new blocks, but seeing how participants reacted to it we decided to lower the number to 2-3 blocks with every click. We thought about lowering it to only one block at a time because it would be easier to grasp but decided against it for now. The grouping of the blocks is also used as a way to model which blocks would go well together, so by exposing 2 or 3 blocks that can be connected to each other we also offer some implicit suggestion for learners for things they can try next.

However, this scaffolded version of the editor led some participants to a misconception about what they can do with Scratch. Seeing the small number of blocks in the microworld when they opened it, they thought that this is all there is, which made them less motivated to try and quicker to abandon the experience. As the improvements to the interface in the newer versions made it easier for participants to understand that they can get more blocks, this phenomenon became less evident, but didn’t completely disappear.

At the same time we experimented with the “show more” buttons, we also experimented with a different way to organize the gradual exposure in the coding palette. We created another type of scaffolding that was based on micro-goals (see Figure 14). In this prototype, the learner could move between different categories of blocks, each focused on blocks needed to achieve a specific goal like adding sound, or making the sprite dance. Unlike the traditional Scratch blocks categories, here each category could include blocks from different types, as long as they create a coherent set. However, unlike the “show more” option, which from the learner’s point of view exposes a mysterious and almost random set of blocks each time, here they could navigate their project in different ways by choosing to work with different categories. Another advantage of this type of scaffolding is the dual role the categories played, both as ways to break down blocks into groups and as a way to model some of the possibilities one can explore. The category buttons act as visual indicators of different goals one can assume.

For some learners, those goal-oriented categories were very intuitive, while for others they were mostly confusing. For the ones that understood how to work with them, the categories provided inspiration and encouragement to try more new things, and to be more intentional about what they are trying to do. But the ones that were confused were unsure where to find the blocks they were looking for, even after using the same microworld for awhile. One problem with those categories is that the need to keep their names short constrained us in choosing how to represent the action. We were trying to represent a full sentence in a word, and it led to inaccuracies and confusion. Another challenge was coming up with those categories. While it worked great for some microworlds, for others every breakdown of the blocks into micro-goals we tried felt artificial and unhelpful. Also, while we were trying to reduce the number of blocks and features in the interface, these categories actually added a few more buttons.
Chapter 6 - Scaffolding While Keeping Learners' Agency

Based on our experience with building this type of micro-goal scaffolding, as well as the observations from learners, we decided to not move forward with this design at this point. However, I still believe that they could be a powerful feature that will allow learners more agency and more exposure into what’s possible. To achieve that, more iterations on the design are needed and more effort should go into the curation of the goals and the blocks that go in each of them.

Open in Scratch

An important feature of Scratch Microworlds is that they are a pathway into Scratch. As such, designing the possible pathways from the microworlds into Scratch was important. The main pathway is the ability to “step out” of a microworld and transferring one’s project into the full Scratch editor. To support that we added an “open in Scratch” option, which allows learners to move their current project to the full editor with the click of a button, at any point in the process.

This option was added in the second prototype, where clicking the “open in Scratch” button rerouted the project to a new webpage that contained your microworld project as a remix of the starter project. This process was limited in a few ways. Learners had to be signed into Scratch to use it, it took a few seconds to reroute and load, and it was irreversible – once a learner moved to Scratch there was no way back to the microworld. In the next iterations, a much more lightweight way to move between the microworld and Scratch was implemented. It still involves clicking the “open in Scratch” button, but now all it’s doing is changing the view of the same webpage. The coding palette presents the Scratch default categories and blocks and the ability to add new sprites becomes available. It doesn’t take time to load and doesn’t require login in to Scratch. As we learned that learners are sometimes interested in going back to the microworld, we made the action reversible – after clicking the button, its text changes to “return to microworld”, and another click will reverse the changes to the interface.

This led to an interesting behavior we noticed, where learners switched over to Scratch, performed the action they couldn’t do in the microworld (like adding a new sprite or a new block), and then returned to the microworld experience. They sometimes performed this switch many times, and one hypothesis is that they did it because it was easier for them to use the limited palette. After noticing it, we started using it as a facilitation method – when a learner moves to Scratch, adds more sprites and then is not sure what to do, we suggest going back to the microworld to keep working.

Because most participants didn’t find this option on their own, a facilitator or a peer usually told them about it. As facilitators, we offered this option is three different cases – when the learner wanted to code something that was out of the scope of the microworld blocks, wanted to add another sprite, or more generally seemed to need a new challenge. Furthermore, for participants with prior Scratch experience, being able to move to Scratch is what made the microworlds appealing. While many of them treated the microworlds as a challenge of trying to create an interesting project with limited blocks, when they discovered they can get the full set
of features in Scratch they were usually very excited. We saw many advanced Scratchers exploring the microworlds with the same excitement novice learners showed, using it as an opportunity to try out new things in Scratch that they might have never thought of trying.

The question of visibility is still an open one, as most participants do not notice the option to open in Scratch on their own, and need a facilitator or a peer to tell them about it. While ideally the visibility problem could probably be fixed with a better interface design, there are some benefits to “hiding” this option at the beginning. At least for some learners, knowing that there’s an option to move forward might make them spend less time in the microworld experience and make them jump ahead and be confused. One option we are considering, that will highly increase visibility is making the last step of the “show more” button be this switch to Scratch, or at least a prompt that will explain how to move.

Another advantage of this scaffolding into Scratch is the possible connection into the Scratch online community. Once learners move to Scratch, they can start using some of these features, like creating a Scratch account, saving, and sharing their projects with the online community. While this was not part of current prototypes because of the technical challenges of connecting to the online community, I hope that it will be available in the future.

Similarity to Scratch

To further support learners’ transition from the microworlds into Scratch, we tried to keep the design of microworlds as close as possible to the design of Scratch. This meant rejecting most changes that required big changes to the interface and interactions within it, even if they would make it easier for beginners to understand. For example, we thought about adding a new way to add a new sprite, moving the green flag to a different location, and changing the behavior of the “next costume” block to include a short wait. Each of these changes would have implied an inherent change to the way Scratch works, and would have made the transition between the two experiences much more jarring. At the same time we did add or enlarge some selections that were most important, such as choosing a sprite, but in ways that would not interfere with the mental model learners develop of how Scratch works.

Starter Code

“It’s so fine and yet so terrible to stand in front of a blank canvas”

- Paul Cézanne

Starting to paint with a blank canvas can be a terrifying or an exhilarating experience, depending on the person and the context. The same is true for creating with Scratch, where every new project starts with a default sprite, the Scratch cat, and an empty coding area. Learners learn how to use Scratch by either experimenting freely with the blocks or following a learning resource – a facilitator’s demonstration, a tutorial, a physical coding card, or a video, to
name a few. Most of these resources model different ways to connect the coding blocks, give ideas and help with its execution.

A different approach we tried with Scratch Microworlds is using starter code – one or more coding scripts that are pre-assembled in the code editor when the microworld is loaded. While similar to other resources in that it models possible behaviors, it also allows learners to start by manipulating something that is already there, instead of building something from the ground up.

I started experimenting with starter code with the art microworld. As I was trying to adapt an automatic drawing project shared by the ScratchEd team at Harvard and turn it into a microworld, I realized it will be very hard for learners to “tinker their way” into a meaningful project. The art project contained six coding blocks that had to be assembled in a specific way for it to create an exciting effect. Once this basic code is built, the art project allows learners to tinker very easily with a few basic blocks, creating engaging visual effects. So, to create better starting point for learners, I decided to pre-build this basic structure, so learners will start the project when it’s already doing something interesting, and will be able to spend most of their time tinkering in meaningful ways.

Throughout all of our testing sessions, the art microworld was the most popular one, with over 50% of the participants trying it, and many spending most of their time (20-40 minutes) working with it. For many participants this is the most engaging project, and the variety of different projects created with it is very wide. We tried to unpack what makes it so successful and hypothesized that at least part of it is connected to having starter code. What started only to create an easier starting point proved to also help learners see what is possible with Scratch and support their tinkering by creating an immediate feedback loop where every block they added or changed was immediately reflected in the movement of the sprite.

We also experimented with the addition of very few coding blocks as starter code. In many other microworlds, we added just “hat blocks” (e.g., when key pressed, when sprite is clicked) to the coding area. This didn’t make the project do anything interesting when starting, but provided a scaffold to help learners understand the idea of connecting blocks together and creating interactive projects. By adding more than one “hat block” we also hoped to encourage learners to try creating more than one stack of blocks. While we noticed that having the hat blocks helped learners in their first steps, most of them still used only one stack of blocks, ignoring the other one.

Encouraged by the success of the art microworld, in the last iteration of the prototype we added starter code to a few more microworlds. For example, the music microworld now starts with code that makes two instruments play music when the space bar is clicked. In this way, we model some possible ways to make music, but also allow the learner to interact with the project even before starting to use the coding blocks.
While there are many benefits to adding starter code to an experience to make it easier and more engaging for novice users, there are also a few disadvantages to it. A major concern when adding starter code is that learners will play and create with the microworld without understanding how it works—they will treat the code as sort of "magic" and will not try to get to bottom of how and why does it work the way it does. Another risk is that the starter code will push learners to take the project in a specific direction, thus limiting their imagination and agency.

To provide even more scaffolding we considered using the "show more" button to show new pre-built scripts. In this way, we could expose the learner to very simple pre-built scripts at the beginning, but as they choose to move forward they could be exposed to more complicated pre-built scripts that will be hard for them to discover otherwise. While this will probably help them discover and use more complex concepts, it will make both concerns mentioned above even more prevalent—it will limit learners' imagination and possible pathways even more, and will allow them to play with the code even more without understanding how it works. Unwilling to take this risk, we currently decided against this idea but we might want to revisit it in the future.

While our experience with the art microworld and preliminary observations from the other microworlds are promising, more research is needed to unpack the impact of adding starter code to introduction projects, and we plan to continue working on it in the future.

Contextual Help

In puzzle-based experiences, much of the communication with the learner is done via instructional prompts that provide help within the context of the puzzle. These prompts help guide the learner towards the solution by telling them if they are right or wrong, or suggesting how to fix a problem in the code. Because there is a solution to every puzzle, the designer of the experience can effectively design these help prompts, by matching a response to any action that the learners do, guiding them towards the pre-determined goal.

In an open-ended experience, on the other hand, this type of help is much more challenging to design because the learners could have different goals that are not known to the designer.
Figure 16 – Examples of contextual help from Code.org (top), CodeMonkey (bottom right) and Khan Academy (bottom right)

Because there’s no right answer, there’s also no way to evaluate the project to decide if it’s “right” or “wrong”, and if the learner needs help.

Looking at the issues participants needed help with when working with microworlds, three main categories of problems could be recognized:

- **Interface issues** – misunderstandings of an element of the interface. For example, not understanding what a button does, not knowing how to drag and connect blocks, or not being able to delete or duplicate a block.
- **Coding issues** – misunderstandings of the way the coding blocks work. For example, not understanding why switching the costume to the current one doesn’t have a visible effect, or being confused by the fact that running two blocks that cancel each other might look like nothing is happening (like “turn right 15 degrees” and “turn left 15 degrees” without waiting in between).
- **Learning process issues** – issues related to the creative process. Not having ideas for what to do next and not willing to try out new things are two examples.

Contextual prompts are well suited to handle interface issues, like explaining how the basic interactions and buttons. In an instructionist experience, knowing what the learner is trying to do, makes contextual help useful for coding issues and learning process issues as well. But in a constructionist environment, it is very hard to tailor prompts that will be useful for learners when the context of where they are in the process is not known.

Based on this understanding, I decided to focus in the microworlds project on the interface and coding issues. In two of our prototypes, we used instructional prompts to demonstrate the use of the interface. We used animated images to show learners how to drag and connect coding blocks to each other. Because this is simply an aspect of the interface and not a deeper
conceptual idea, there's no reason to leave it for the learners to "explore", and it's better to remove this initial barrier to creation.

The main issue we saw with our animated images was that learners were confused about the fact that this was only an image. They had a tendency to try to drag the blocks out of the animation, which didn't work. This is consistent with the way learners interact with the Tips Window content in Scratch. After seeing this behavior, we decided to seamlessly integrate the prompts into the interface – instead of modeling a click that's somewhere else, highlighting the relevant parts in the interface, or overlaying the animation on top of the blocks palette. We designed prompts for a few basic interactions: dragging and connecting blocks, clicking on a block in the palette to run it, clicking a stack of blocks to run it, and creating interaction with keyboard keys. Technical issues prevented us from implementing many of these designs in the prototypes, but I hope to be able to continue researching their usefulness in the future.

In addition, to handle some of the coding issues, we experimented with designs for small pop-up prompts that warn learners when the way they connect blocks makes it look like nothing is happening. This brings up the delicate balance between letting the learners explore on their own why things are happening and risk frustration over not getting it. I believe a careful design of when to show these prompts, and what should they say will help with this balance.

Similar to Scratch, to support the more meaningful and creative aspects of learning, we use methods that encourage exploration rather than guide learners to the correct answer. For example, we use short videos and animations to spark the imagination of learners, show a range of possibilities, and offer more things to try.

We also experimented with adding more physical and digital cards that can be used by the learners on an as-needed basis. For these cards, we do not try to contextualize them and offer the help to the learner, but offer all of them for the learners to browse through when looking for help.

In the last sections I surveyed the different ways we experimented with adding support and scaffolding for novice learners in Scratch Microworlds. But while offering more support for learners, I wanted to make sure we still let learners be in control of their learning process. In the next section, I discuss how this was achieved.
Chapter 6 - Scaffolding While Keeping Learners’ Agency

Learner Agency
Inspired by Papert’s idea of building learning environments where “the child programs the computer” instead of “the computer programming the child” [31], learner agency is one of the core values behind Scratch, and Scratch Microworlds. But, as providing more support for the learner was another key motivation, we set out to pay special attention to the balance between the two.

Special care was taken in the design of the scaffolding features, as well as in creating new features that increase agency and counterbalance some of the lost agency because of the scaffolds. To achieve that, we created a softer scaffolding structure, which is more aligned with the idea of “constructionist scaffolding”. While some of those considerations were described as part of the design of specific scaffolding features, this section will focus on three guiding principles for the support of agency: learners decide what to create, when to move forward, and what to explore next.

Learners Decide What to Create
A major part of having agency over your learning process is being able to decide what to work on. Scratch Microworlds provide support for that in two ways – choosing a microworld, and choosing what to create in the project.

We designed microworlds to allow learners to easily choose which microworlds to try, how long to spend working with each, and when to move to a different one. We created a homepage, which shows the names and thumbnails for each one of the projects, so that learners could easily see the range of possibilities available. While we can’t promise learners will have this experience, as it depends on the educational context and the educator itself, we hope that by lowering the “cost” of letting learners explore freely, more educators will be likely to use them this way. Microworlds require little instruction, and because they use similar and small sets of blocks, they also require less preparation from the educator. In Scratch, many educators prefer to focus on a specific project, block, or concept in every session, because it is harder for them to support learners that are going off in different directions. With the smaller scope of the microworlds, we hope educators would be encouraged to let their students choose to work on any microworld they want to. When we tested microworlds with educators many were excited about the minimal introduction needed.

This was how we conducted all of our testing sessions, and we saw participants spend anything between 30 seconds and 40 minutes with one microworld. Many of them wanted to explore more than one microworld, and showed different patterns of exploration – some used every possible microworld but for a short time, and others stuck with the same microworld for the whole duration of the engagement. Most participants explored at least 2 or 3 microworlds over a period of 40 minutes.
After choosing a microworld, there are still many choices to make. Because the microworlds projects don't have a pre-defined goal, and they support a big diversity of projects, learners using a microworld can define their own goals.

Learners Decide When to Move Forward

Coding puzzles are usually organized as a series of levels, where the learner follows a predetermined, linear path — every time they solve a puzzle they progress to the next one. While this approach allows for a closer support for the learner with every step, it also prevents the learner from making decisions about when is the right time to move forward. In Scratch Microworlds, we propose a different type of scaffolding — one that is led by the learner. The learner can choose when and how much to move forward.

As we experimented with different ways of scaffolding the blocks palette, from the option to show more blocks every time a button is clicked to the option to move your project to the full Scratch editor, one thing stayed constant — the learner was the one deciding when to click and move. Unlike many coding puzzles, in Scratch Microworlds the program will never tell learners that they are not ready to progress or that they should progress now.

Letting the learner decide when to move allows for different styles of engagement, and we did see different learners use these features in different ways. For some students, the first set of blocks was enough to explore for a long time, carefully experimenting with different values and ways to connect the blocks. Others preferred to start by exposing all available blocks and experiment with all of them at the same time. Many learners chose an intermediate process, and spent a few minutes creating with the first blocks and then exposed another “batch”, two “batches” or all available ones. We also noticed some learners that exposed more blocks, but only until they filled the visible palette. While they can scroll to see more, they either didn’t understand that, or they were just looking for the feeling of having many options.

Figure 18 – The microworlds homepage, featuring 6 starting points.

Figure 19 – codeAcedmy’s example of a level structure
The learners used similar patterns of behavior with the option to move to Scratch. Some participants wanted to explore more than the microworld allowed after a few minutes, but others took 10, 20, or even 40 minutes before wanting to move. However, this finding might be skewed, as most learners didn’t find this option on their own and needed a facilitator or a peer to show them it’s possible. However, I believe that learners would have still presented different patterns of behavior even if they knew about the button beforehand. The time they spend in the microworld might change, but we’ll still see some learners moving right away while for others it would take more time. This is supported by an experiment we did in one of our testing sessions, where I showed everyone how to move into Scratch about 40 minutes into the workshop. Not all students chose to use it right away, some preferred to keep working with the microworld.

We also learned that participants were interested in a way to revert each one of these features that allowed moving forward. Both for exposing more blocks and for the option to move to Scratch, participants were interested in a way to revert this and hide it again. To address their feedback, we moved from a one-button model, which just shows more blocks, to one that has two buttons for “show more” and “show fewer”. We also made “open in Scratch” a reversible action, where one could go back to the microworld experience after moving to Scratch. This allowed learners to explore the next step with less fear – if it was too overwhelming, they could always go back. Learners also used this as a way to add specific blocks or sprites they wanted from the full Scratch editor, while doing most of the work within the microworld environment. While they are obviously aware of the possibility to create with the full set of blocks, they still preferred using the small, but more familiar set of blocks.

Learners Decide What to Explore Next
Within most coding puzzles, the learner cannot make any meaningful choices about which coding blocks to use or what they want to do with them. In contrast, in Scratch Microworlds we strive to provide the learners with many opportunities to make meaningful choices and direct their own learning.

Apart from letting learners choose which microworld to use, we also experimented with supporting them in making meaningful choices on which blocks to use. Limiting the number of blocks definitely limited their choices, so we tried to counterbalance for it by adding new ways to choose. We experimented with two ways of scaffolding the exposure of more blocks.

As mentioned in the section about the blocks palette, we created two different ways to scaffold the addition of more coding blocks. The first allowed learners to expose more blocks by clicking the “show more” button, and the second allowed them to navigate between different sets of blocks based on micro-goals. Both methods let learners guide their learning process. The micro-goals method made it natural for the learners to follow their interests in what they wanted to explore – they choose to use whichever categories were interesting to them. While the “show more” method allows for less goal-oriented control on the learner’s side, learners are still in control. They can choose to use the newly exposed blocks, only use some of them, or not use
any of them and ask for more. This is in contrast to many of the puzzle-based approaches, where one can't move forward and explore more commands before learning how to use the current ones to solve the puzzle. We saw learners use it to create different learning paths for themselves. Only a small number of participants explored the blocks one after the other, in the order they were presented. The rest tinkered with the blocks based on things they wanted to try, picking and choosing between them.

Another way learners' choice is supported is the option to open a microworlds project in the full Scratch editor. This allows learners to explore in different directions than the ones available within the microworld. As soon as they want to explore something that doesn't exist in a microworld, they can move to Scratch and get an infinite amount of possibilities for explorations.

This chapter explored the tension between two core ideas in the design of learning experiences – scaffolding and learner agency. While they are not mutually exclusive, they can be in tension with each other and implementing both requires careful planning. In the next chapter we look at another core idea in learning theory – building on learners interests, and explore the different subtleties and challenges in designing such learning experiences.
Chapter 7

Providing Starting Points that Spark the Imagination

Design Challenge #3: How can we offer entry points into coding that are motivating and meaningful to youth with diverse interests and backgrounds?

Providing opportunities for young people to create projects based on their interests in a core value of Scratch. Its support of a wide range of interests is one of its strongest features, as can be seen from the diversity of projects in the online community. But even if these topics are represented in the community, it’s not always easy for newcomers to use them as starting points. Finding projects that inspire you is not always easy, and when finding them it’s sometimes hard to understand how to change it or create a similar one. Designing specific interest-based starting points for a diverse set of interests can help.

Scratch attracts a diverse range of young people, and we hope that creating more interest-based pathways into Scratch will expand its reach in groups that have been traditionally underrepresented in computing. The microworlds project, as part of the larger Coding for All initiative aims to support a broader range of youth interests, social identities and cultural references [37]. By building on their interests, we hope to engage young people that have previously felt alienated or uninterested in traditional pathways into computing.

For that reason, we focus more on helping children see themselves as interested and capable of creating with code, rather than on them exploring specific scientific concepts. We want to focus on computational perspectives, not just computational concepts [7]. To do that, we decided to make microworlds interest-based and creative, to harness learners’ internal motivation. To support learning through topics that are meaningful to learners, we chose to create many different interest-based microworlds that can support a diversity of projects and encourage tinkeringability and playful explorations.

While the last two chapters introduced many of the design considerations for the general microworlds environment, this chapter will mostly describe our consideration with regards to the content of the microworlds – the projects themselves. It will present the design considerations we used when choosing which themes to use for the microworlds and how we adapted them to create interest-based, tinkerable and playful starting points.
Interest-Based Starting Points

We used a few methods to identify themes that would make good interest-based microworlds. First, to support the goal of engaging youth from groups that are traditionally underrepresented in computing, we started looking for themes that would be relevant for them. One such example is the hip-hop dance microworld, where learners code their own dance routine. This helps them see the connection between choreography and coding, and makes coding more approachable and relevant. Second, we looked for topics that are highly popular in the Scratch community, such as music, fashion, and humor, with the belief that topics that are popular in the community would also be engaging for more young people. Last, we collected ideas from the children and educators we interacted with. In each of our workshops we asked them about other topics they would like to create with, and while we got very different suggestions from different people, some themes repeated many times. Game design, fantasy (dragons, magic, and mythical creatures), and more sports and arts were the most popular requests. In many of the educators' workshops we also worked with educators on designing their own microworlds - based on themes they likes, or thought their students would like.

The first two microworlds, hip-hop dance and fashion were chosen using the first two methods mentioned above. Hip-hop connected with Scratch on a few levels - from the origins of the name Scratch in remixing music, to the idea of creative expression. As members of the Scratch team connected with a local educational organization, Progressive Arts Alliance, they discovered the similarities between coding and dance routines, and decided to focus on hip-hop dance. It was chosen as a topic that interests youth all over the world, but is not represented on Scratch so much. Fashion was chosen as a topic that was already very popular on Scratch, and also connected with many youth interests. Over a few workshops we conducted with educators and members of the Lifelong Kindergarten group, we collected ideas for new microworlds and created five new ones - soccer, music, art, humor and interactive house. In the next iteration we focused on single-sprite microworlds, which meant building new versions of the music and dance microworlds, as well as adding a Pokémon microworld and a gymnastics microworld. We chose these two topics to connect with two cultural trends - the gymnastics microworld features Simone Biles, the American artistic gymnast that just won many medals at the 2016 Olympic Games and Pokémon was very popular due to the Pokémon Go game. Later, in the next iteration, we decide to remove the single-sprite limitation and where only limited by the technical infrastructure we were using. We started using the Scratch 3.0 infrastructure, and because it is still under development some of the features we needed were missing. For example, "say" blocks are not implemented yet, which prevented us from using the humor microworld, as it is based on the sprite telling a joke with say blocks. In this last prototype, we combined microworlds from the two previous prototypes, and currently support six microworlds: gymnastics, art, music A (just a guitar), music B (a whole band), hip-hop dance and interactive house (see Figure 18).
These topics cover only a very small subset of youth interests, which are highly varied. While some popular interests can reach a big group of youth, there’s also a long tail of very specific interests they want to interact with. For example, having a soccer microworld was engaging for many children, but others would have preferred baseball, basketball or golf. To allow every learner to work on a topic that is personally meaningful for them, we need to provide a much wider variety of themes. While we know it will be impossible to cover all topics youth are interested in, we hope that by creating a many varied microworlds we can appeal to most of the children and youth.

We saw large diversity in the microworlds participants chose to use. Within the six microworlds in the last iteration, the art microworld was the most popular, but for the others the distribution was fairly even. In some workshops gymnastics and hip-hop dance were next, while in others music and interactive house were next. It’s interesting to note that small refinements of the microworlds changed how popular a microworld was. While the participants in our workshops enjoyed the available themes, many asked if there were additional microworlds that fit their specific topics of interests.

Even when learners were interested in one of themes, they sometimes wanted to extend the scope of the theme. One young man told us he’s interested in music, but when he tried the music microworld he was disappointed to see there’s no way to add beat box sounds. Another was excited about the hip-hop microworld, but wanted to use different music. To address that, we’re currently looking into other ways to allow learners to customize the microworlds. For example, we can maybe create one sports microworld, but let the learner choose the type of ball and player before starting the experience.

Choosing suitable themes for a microworld is a very important aspect of creating the right experience we’re aiming for, but choosing a theme is not enough. In the next section I’ll introduce a few considerations for how to turn a theme into a microworld.

“Deep” Interest

Building on children’s interests is not a unique strategy to Scratch. Many educational tools tap into children’s interests to motivate their learning. Some use characters from popular media, some use popular topics such as sports or stories to draw children into the experience. But, when discussing interest-based learning, I want to suggest a distinction between “shallow” and “deep” interest-based experience. A “shallow” connection with an interest area consists of taking an existing learning experience and packaging it with popular characters, an appealing story, or themed assets. A “deep” interest-based experience allows the learner to actually engage with the ideas and pursue their interest in the topic. For example, many coding puzzles take the form of navigating a character through a maze to get to a goal. Simply swapping out the characters and narrative but using the same maze format can be seen as a shallow connection with an interest. With microworlds, we try to make sure we are not only packaging the same thing with different assets to create different microworlds, but that each microworld is tailored to give the
learner an opportunity to engage with the ideas. Thus, for example, in the humor microworld, learners are scripting characters to tell jokes, while in the interactive house microworld they are scripting how lamps and other items in a house will respond to interaction.

**Tinkerable Starting Points**

Designing a good interest-based learning experience depends not only on choosing suitable themes, and the right level of connection with it. When choosing themes for microworlds we also placed a priority on starting points that encourage exploration and tinkering. Microworlds work best when children can create something meaningful and personal by tinkering with the coding blocks. For example, it is easy to create many different dances by using just two blocks - “change costume” (which changes the sprite’s appearance) and “wait”, and almost any way you connect a series of these blocks can create an animation of a dance. Other projects, like many types of games, require very specific ways of assembling the blocks to create something meaningful. Trying to create a microworld out of these types of projects can be frustrating and more like a puzzle. These types of projects may be better presented as a tutorial or a remixable starter project than a microworld.

Early in the design process, we thought about creating a sports-related microworld, but found it difficult to represent a soccer game in a simple, tinkerable microworld. We tried to design a simple soccer game, but no matter what we tried it was all either too hard to build without instruction or not engaging enough. Talking about this issue with another graduate student in the group, who played soccer as a kid, helped us make a breakthrough. She suggested looking at soccer practice, rather than a soccer match, for inspiration. Working with her to refine the experience, we had much better success once we stopped trying to recreate a full soccer game and started working on a soccer interactive animation project, where learners animate a soccer ball and a cleat to kick it. While less engaging than a full game, learners still find it fun and engaging and it inspired them to want to learn more about Scratch so that they could build their game.

In the newer version of the microworlds, with the added scaffolding features, new possibilities arise to create an engaging, tinkerable, and interest-based experience. We experimented with a design for a soccer game microworld based on those ideas, but we could not implement it yet because of missing features in the Scratch 3.0 prototype. We hope to create it once this features are implemented, and hope to see children more engaged, and creating with more advanced blocks.
Chapter 7 - Providing Starting Points that Spark the Imagination

Support Diversity of Outcomes

Another important aspect of the microworlds we created was the diversity of projects it can support. In chapter 5 I described the importance of supporting diverse projects for creativity, but it also affects the connection learners have with their projects. When learners have different final projects, they can start treating the project as their own. They can share it with others (offline or online) and feel proud of what they accomplished. Having an audience for work can also motivate improvements and further work [22, 45]. In contrast, in puzzle-based experiences, learners will usually finish the experience with very similar projects, which limits this connection, and makes learners less likely to want to share it with others. While they might be proud of their accomplishments, their projects will be less personally meaningful and lack a sense of ownership.

In Scratch Microworlds, we treat the diversity of projects created as one of the measurements for our success. We chose themes that encourage this diversity, like art and music, and carefully chose coding blocks that allow different behaviors. Looking at the available microworlds, some allow for more diversity, while for others the projects tend to be more similar. We observed different dimensions of diversity:

Figure 20 – Different art projects created by children and educators
Chapter 7 - Providing Starting Points that Spark the Imagination

- Different looks – projects that have different visuals. Either because they use different assets, or because of modifications done using the coding blocks. For example, few learners using the art microworlds used very similar code, but because of the sprites they chose and their placement the final result has different aesthetics.
- Different behavior – projects that have different scripts, and use the coding blocks in different ways. These projects might look very similar at the beginning, but the running project will be very different. For example, two hip-hop dance projects might show the same sprites – a boy and a girl, but while one project is a synchronized dance the other

Figure 21 – Hip-hop and music examples created by children and educators
Chapter 7 - Providing Starting Points that Spark the Imagination

uses visual effects to make the dancers change colors.

- Different project type – projects that have different goal or interaction mode. For example, it’s the difference between an automatic drawing project, and one where a user can control what is being drawn. With the music microworlds, some learners focused on trying to recreate a song they know with the available notes, some created an interactive music project where the user can play different notes or instruments with different interactions, and others created a new tune.

- Different stories – projects vary in the stories learners tell about them. The same behavior could be explained in many ways, and when asked about their projects children give different explanation to what it is doing and way. For example, explaining that the interactive house is “haunted”, “magical” or that “things move when you click the keyboard keys”.

Figures 20-21 will unpack some of this variety with examples.

Figure 20 shows example of four art projects. The top two projects use random numbers - one only uses one costume and the other switches between the available costumes to create a pattern. The bottom right project is a very simple one, only adding one block to the starter code to create a beautiful pattern. The bottom left project is an interactive drawing project, where you can move the sprite with the arrows, and decide when to stamp it on the canvas.

Figure 21 shows various examples from music and hip-hop projects. Some project use very little code while some use a lot. The top right project is a game where music is playing and the user needs to play along with the arrows. The top left makes a sound and changes color with every key press.

Because microworlds are a pathway into Scratch, we envision it also as the first step in building one’s online Scratch portfolio. Ideally, learners will be able to save their microworlds projects, and share them with the Scratch community. To test if learners would be interested in saving and sharing their projects, in our workshops we encouraged them to present their projects to other participants, and asked if they would like an image of their project. Many of the participants liked presenting their projects and explain them to others, and many requested to get the image saved for the project.

Playful Explorations

Two young girls are playing with the art microworld, and created a project where colorful sprites are stamped all over the screen. When a key is pressed, it turns white and starts deleting the colorful marks. They whimsically call it “maid service”.

A teenage boy isn’t very engaged by the microworlds, and is browsing through them. When he surveys the interactive house project, the name of the carpet
sprite catches his eye – “magic carpet”. The name sparks his imagination and he sets off to start coding the carpet to change colors when clicked.

A teenage girl tries out different sounds with the “play sound” block, when she chooses the sound named “hey” a funny voice is heard saying “hey”. She starts laughing, and so does her friend sitting next to her.

These are all examples of the ways participants reacted to playful or unexpected behaviors in the microworlds. We started the microworlds project with a focus on interest-based learning, and while our observations support this approach, they also show the value of playful and unexpected interactions, which lead to very high level of engagement. A surprising behavior, a playful name of a sprite, or adding a randomized piece to a project could be the difference between an engaged learner and a bored one. It leads to a more playful, imaginative, and exploratory play.

Seeing the sense of wonder and excitement that participants presented when they were surprised by the microworlds encouraged us to add more opportunities for this type of interactions. In the last iteration of the prototype, we increased the chances of learners to stumble upon something that will surprise them while exploring. We added many playful sounds, interesting costumes, added some coding blocks, and added blocks for adding random numbers that can be used to create something unexpected.

Because microworlds are introductory experiences, and surprises help get learners engaged and motivated, we want to get learners to experience it as early as possible. We see that when their first steps with the microworlds are playful and exciting, they are much more likely to stay and explore more. Playing with visual effects is a good example of that. Participants found this block very engaging, and many spent a long time exploring the different effects and the way they work on different sprites. They weren’t trying to understand exactly how these effects work, but were more interested in running them and seeing what happens.

But, unexpected behaviors are also harder to debug, so learners do not know how to fix them if something goes wrong. For example, while playing with the visual effect, many participants tried out the “brightness” or “ghost” visual effect, which very quickly made their sprite invisible, and they usually didn’t know how to bring it back. Participants were also sometimes confused by the random number block, and the fact that it doesn’t attach to other blocks, but fits into a variable field in another block. After trying to add it to the stack of blocks and seeing it doesn’t connect, they tended to give up and ignore the random block. While adding randomness to project is a great way to create a playful experience, we need to find a better way to introduce them to learners.
Part III  Reflections and Suggestions

While part II focused on specific aspects of the design of Scratch Microworlds, part III looks back at the whole experience. By taking a few steps back, one can start to connect the ideas into a bigger picture of constructionist introductions to coding. First we take one step back, and look at the Scratch Microworlds experience as a whole, through the stories and feedback of the young people and educators we worked with. Then, taking another step back, chapter 9 discusses some of the lessons we learned, and the implications they could have for the design of other constructionist introductions. Chapter 9 concludes with a discussion of the limitation and future directions for the project.
Chapter 8

Learning and Teaching with Scratch Microworlds

Much of our design was informed by the ways people were using Scratch Microworlds, the projects they created, and their feedback. We tested the microworlds with both young people and educators, and were able to learn a lot from their experiences and their thoughts. In this chapter, I focus on the experience as a whole, as opposed to the more specific observations mentioned throughout the last three chapters. I start with the experiences of children and youth, and continue with a discussion of educators’ experiences and our reflections about them.

Learning with Scratch Microworlds

We designed Scratch Microworlds to support novice learners in Scratch. For that reason, this section will first discuss the newcomer experience with the microworlds by highlighting a few examples. Then, I describe how young people with prior experience in Scratch or other coding activities approached the microworlds. We were lucky to meet and observe many young people with some prior experience, and we learned a lot from them. It encouraged us to think more about the role of interest-based experiences for all learners, and the benefits of the microworlds even for learners with experience.

Newcomers to Scratch

The best way to describe the experience of getting started with Scratch through the microworlds is the story of the last workshop we ran. It demonstrated the value and potential of this approach. The workshop was held in a public middle school, in two groups of 8th graders, each with 12 students with no Scratch or coding experience. It was part of their school day, but presented as a “coding club”. It was the longest session we ran: we spent 90 minutes with each group (Figure 22 shows a few of the projects they created).

We started the workshop with a quick introduction to the microworlds, demonstrating choosing a microworld, clicking on the blocks to see what they do, and dragging and connecting them to each other. Then, participants started working with the microworlds, choosing different ones and moving between them as they wished. They explored many different topics, like making music and art, and with the support of the facilitators were able to create many diverse projects.
About 40 minutes into the workshop, some of the youth were getting restless and losing interest. So we took a short break from creating, in which they shared their projects with others. I then demonstrated moving from a microworld to Scratch, and showed the differences — there are more blocks to try, and the ability to add new sprites. I emphasized that they can choose to continue to work on the project they started, start a new one, or move their current project to Scratch and use these new features.

Some participants chose to keep working on their projects, but within a few minutes many of them moved their project to Scratch, and started experimenting with adding new sprites. The first sprites in the library were letters, and so many participants naturally chose to write a word with the sprites — their name, nickname, or something they like. After adding a few letters, they started trying to animate them in different ways — mostly changing colors and making them move.

While it wasn’t a completely smooth transition, and it took some time to figure out how to get the effect they wanted, they were able to get over these hurdles, and animate the letters as they wished to. Many used the limited coding palette in the microworld, and their experience from making similar things happen in the microworlds.

In 90 minutes, with very little instruction time, they moved from having no idea how to create with Scratch to being able to create projects that were personally meaningful for them and that they were excited to share with their friends and wanted to keep working on. Their natural exploration process took them from the microworlds into an “animate your name” project, similar to the “animate your name” tutorial offered on Scratch. But because these young people had the experience of using a microworld, they were able to build a project without a step-by-step instruction. And they enjoyed the experience — the classroom was filled with chatter, funny sounds and working together.

The youth at this last playtest demonstrated the vision of learning with Scratch Microworlds, and through their experience we could see how our design choices allowed it. Starting with a simple and open-ended experience kept them engaged and excited while they were discovering how things work. The combination of scaffolding and agency allowed them to control the pace of the experience, and helped them progress from novices to Scratchers. Starting with many topics to choose from, and especially ones that they could relate to, increased their motivation to explore and create. Once they learned how to personalize their project even further, with the addition of letters, they were even more motivated.

Some aspects of the activity design supported the success of this workshop. Having a longer session allowed learner to take their time and still get to the transition into Scratch. The
prototype itself was also better than in the first workshops, because we iteratively improved both the environment and the content.

While this workshop provides a great example, the other play testing sessions also allowed us to recognize other trends. Learners were for the most part highly engaged with the microworlds. Even in drop-in context, where there were many other interesting activities around, participants

Figure 22 – Example projects created by youth at the last playtest
chose to stay and create with the microworlds for long periods of time. In fact, in a few cases parents had to repeatedly ask their children to finish up so that they could leave. Some young people stayed for 5 minutes, but more common were the cases where they stayed for 10-20 minutes, with a number of cases where they stayed for over 40 minutes.

Learners created a wide range of projects, using a variety of approaches and ideas. Their projects ranged in every possible dimension – building different code, using different visuals, and even creating different types of projects. By tinkering with the coding blocks and connecting them in different ways, almost all participants were able to create something they liked and were proud of. We observed many calls of “Mom, look what’s it doing!” or “Look, David, I can make magic!” targeted at an educator or a peer. When we asked youth if they want to share their project with others at the end of a workshop, many of them were excited to share, even if they didn’t get very far with their projects. Wanting to share the project with others shows the sense of ownership they developed.

The playfulness of the experience was also apparent. Participants reacted with awe and surprise when something unexpected happened, and with a “cool!” or “ooooo” when something started running. Some microworlds, like the gymnastics one, were even more playful, and both educators and children talked about how they just wanted to create something funny with it.

When asked what did they like about the microworlds, children mentioned liking specific themes, like art or music, liking the assets, or specific blocks like the graphic effects. For the assets, children mostly referred to the costumes – Gobo in the art microworld (referred to as “the cute flame guy” by one participant), Simone Biles costumes in the gymnastics, or using the line costume to draw in the art microworld. Participants’ main complaint was that there were not enough options – they wanted more coding blocks, more music choices, or more costumes. For that reason, most participants were excited to see the option to open the project in the full Scratch editor.

One of the most interesting factors we noticed in our observations is the amount of time needed for instruction before newcomers could start creating. While we didn’t explicitly compare microworlds to other ways of getting started with Scratch, the difference from other workshops we are running was apparent. Usually, a Scratch workshop starts with at least 5-10 minutes of introduction to Scratch and modeling of some things you can do with it. With the microworld, 2-3 minutes were usually enough to get participants started. After this short introduction, most children were able to start tinkering and try out things on their own, with the support of a facilitator when needed. This shows that microworlds are probably easier to get started with.

However, novice learners faced many of the same difficulties encountered when getting started with Scratch. Participants experienced some of the most common pitfalls when getting started with Scratch – misunderstanding the effect of connecting two “next costume” blocks without a
“wait” in between, navigating between different sprites, and the inclination to build long scripts without testing them.

For a few participants, even the microworlds experience was too overwhelming and confusing. We noticed at least two participants, in different workshops, who said they didn’t know what they’re doing. One of them asked for help from a facilitator, which helped her get started and she was later able to play with the blocks and experiment with the values in the blocks by herself. The other, who didn’t ask for help, was exploring different things in the art microworld, and actually created an interesting project, but when her teacher asked her how she is doing she said “I don’t even know what I’m doing”. This points to one of the issues with open-ended activities, where not everyone feel confident enough to “just explore”, and they sometimes prefer a more straightforward introduction.

Intermediate Coders

Many of the youth we met have used Scratch or a different coding experience before, but only for a short engagement, usually during an Hour of Code [18] activity, or a class activity. They were familiar with some basic features of visual programming with coding blocks, but were usually not fluent in creating projects.

These learners expressed positive attitudes towards the microworlds. They were able to start exploring and creating faster than others, and were not bothered by the difference of experience from Scratch or the other tools they know. The Scratch Microworlds experience helped them improve their coding capabilities, and they were less hesitant to explore new things.

The first thing that was surprising to me was the effects (referring to the visual effect block) because I didn’t know that was there, and I was like ‘Oooh, this looks cool.’ There are so many different options that I could try.

- Lily, 6th grader, used Scratch before in a coding club

We’ve used Scratch before, but never seen patterns like this!

- Sydney, 11 years old, used Scratch for an Hour of Code activity

... but here, it is its own fun trying to figure out what to do with limited blocks

- Sara, 12 years old, used Scratch before

They were usually not bothered by the constraints of the microworlds, but were happy to be introduced to the option to move to the full Scratch when they wanted to do more than the microworld allowed. When asked what would improve the microworlds their first suggestion was
adding more blocks, usually offering a specific block or two they wanted to use but couldn’t, like the forever loop.

For some learners, the microworlds changed a previous perception of Scratch, like this next example shows:

  Diana, 13 years old, came to our station in an open event at the MIT Museum with her family, and when she saw we were using Scratch told her mother that she had used Scratch before and that it was stupid. Her younger sister, who is 10 years old, sat down and started creating, and Diana was watching her for a few minutes. Then, Diana sat down at another computer, and started exploring the microworlds herself. She ended up exploring for 30 minutes.

In post-engagement interviews, a few participants in this group mentioned they liked the microworlds better than their prior engagements. Some of the reasons they gave for it were the fact that there are more options in microworlds than their Hour of Code experience, and that one could create their own designs. When asked if they thought this would make a good introduction to Scratch, many agreed, and thought that this could be easier and more fun that their prior activities.

Experienced Scratchers

In the workshops we also met a few advanced Scratchers, who were young people with a more extensive experience in Scratch. They usually used it in school or an after-school program, and spent at least a few hours creating with Scratch. We were surprised to see these participants found the microworlds engaging. The interest-based themes in the microworlds sparked their imagination and helped them get started on a project in new areas that they said were different from others they had created with Scratch. For the advanced Scratchers, the microworlds were less about the limited experience, and more about the variety of themes. They spent much less time than other participants in the simplified and scaffolded experience, and quickly moved their project to the full version of Scratch. However, they kept on building and revising their project on top of the microworld template project, using the provided assets and code, and adding more sprites and code based on their prior knowledge. For these participants, the microworlds acted more as starter projects, which allowed them to combine their prior experiences with a new domain. They created projects that could not have been created with the limited microworld experience, but ones that would have probably not been created without the microworlds. However, Dan, 11 years old, summarized what we heard from a few of the advanced Scratchers by saying he still preferred Scratch because there’s more variety, and you are “able to do more of what you want.”
Teaching with Scratch Microworlds

Educators were another important resource for us in the design process. As mentioned before, we conducted a few workshops with educators, where they used the microworlds themselves and gave us feedback on them. In some of the workshops, we also worked with the educators on creating new microworlds based on new themes they wanted to add. Their feedback and ideas supported the next iterations on the microworlds. Educators went through a similar play experience as the children, but our discussions with them focused more on their thoughts on microworlds as a learning activity, how they might want to use them with their students, and their perception of coding.

Many educators found the idea of Scratch Microworlds promising, and expressed interest in incorporating them into their teaching. They liked the fact that microworlds let learners explore on their own yet within a structured activity, which is easier to handle in a classroom setting than Scratch. For example, one educator mentioned that they like the fact that very little instruction is needed when working with a microworld – this allows the students to just try it out immediately. Another educator referred to the microworlds as “discovery worlds”, where learners are free to experiment and discover how things work by themselves. They also mentioned that this could especially benefit young learners. Here are a few answers educators gave when asked what they liked about the experience, and what was difficult:

"I liked that it was guided in a specific direction, and that they could add more blocks."

“The environment was very rich – there were only 3 blocks, but you can do a lot with them. “

“Fun, artistic, open-ended”

“I think fewer buttons are definitely less intimidating and help encourage students to poke around more”

When asked how they imagine using microworlds in their classrooms, one educator wrote "use it as hour of code, with more freedom and creativity"; others wrote that they would like to use it to introduce coding, especially with younger students and with special education students.

A few of the teachers praised the ease with which newcomers could start using Scratch with microworlds, and pointed out that this opens possibilities not just for the students, but also for the educators themselves. One suggestion was to use microworlds as part of professional development for educators, to introduce educators to Scratch and some of the specific things you can do with it. For one of the participants in an educator workshop, that is exactly what the workshop was:
It was her first time working with Scratch. She quickly picked the humor microworld ("make a joke") and started working. With a little support from the facilitators and her peers, she managed to create a small project that tells a joke, and plays a cheers sound once it's over. She was very excited about her project, and was trying to think about ways to extend it. Then I showed her she could open her project in Scratch and add more sprites. With more excitement, she got back to creating, and was able to add another sprite that interacts with the comedian and made her project very personal. When we asked if someone wants to share their project with the group at the end, she was the first to raise her hand.

This story demonstrates how microworlds could be the starting points for educators as well, where they can build their confidence with Scratch in a more closely supported experience, but when they're ready they can start playing with everything Scratch has to offer.

While educators were generally excited about using microworlds with their students, in each session one or two educators expressed concern that microworlds may be too limiting for students. For example, one was concerned about not having access to drawing tools within the microworlds, as she had seen many students become engaged in Scratch by first drawing their own characters, something that the microworlds do not currently support. A couple other educators expressed concern that the microworlds may limit what learners think is possible to do with Scratch. Their feedback highlights the importance of the option to open any microworlds project in the full Scratch project editor. These two quotes represent the tension many educators expressed:

"I like the constraints initially. I struggled with not having some of the blocks and costumes available."

"Lots of creative possibility with very few tools. Wished I had more tools."

Some of the educators expressed enthusiasm about the potential of creating their own microworlds for their students, or as one educator put it "designing your own playbox". For example, an educator from Brazil used our prototype to create a storytelling microworld with a small set of blocks and two characters. In a follow-up email several weeks later she wrote:

This week I started Scratch classes with 2 new groups of kids (9 and 12 years old) that had never used Scratch before. In the first class, they used a microworld to start exploring and creating with Scratch, and it was an amazing experience – both for the kids and for me, as a teacher. I felt it was a much friendlier first contact in comparison to the previous approach I was using (starting with the "generic" Scratch environment to create small projects)...I saw a lot of diversity in
the stories they created, and they explored the blocks in a much more autonomous way.

Educators were often interested in concept-based microworlds, and not just interest-based. For example, one educator tried to create a microworld to introduce learners to the concept of clones. A couple of other educators suggested creating microworlds to introduce the concept of variables and randomness. This offers an interesting area for future development – investigating ways to make microworlds that focus on a concept but also are still situated in the context of a meaningful project or interest area.

To make it easier to create new microworlds, I hope to build a tool that can be used by our team. But we're still debating if it should be open to everyone, and in which way. On the one hand, seeing what educators imagined and created was very helpful in our design process. Opening it up for educators will also increase the number of themes we can offer to learners, and allow them to design ones that would fit their students better. On the other hand, we learned how hard it is to design a balanced microworld that helps new learners get started, while supporting creativity and tinkering. We worry that opening this option will lead to microworlds that are unbalanced. Unbalanced microworlds could be ones designed like a puzzle, with one right solution, or ones where the choice of blocks and assets doesn't leave the learner with many interesting choices. I hope to find a way to open this option to teachers, while still maintaining the values and strengths of the current microworlds.

Supporting Educators

Many of the features educators were most excited about in our sessions are addressing known issues for educators using Scratch with their students. For example, we hear from many educators that Scratch can be overwhelming for them as well, both as users and as educators that need to support their students in creating with it. Educators describe many challenges in incorporating open-ended learning activities into their classrooms [6]. They try to balance capacity building with opportunities to explore, deal with large class sizes, and need to feel confident in their ability to support students [6].

While Scratch Microworlds still offer an open-ended, exploratory experience, they do it in a much more structured way. They reduce the amount of instruction students need before getting started, and support their learning and progress more closely. We see Scratch Microworlds as a way for educators to negotiate the tension between wanting to let students work on open-ended projects and the limitations discussed above. We hope that it will help educators that do not feel comfortable teaching Scratch, and will give educators a new way to introduce Scratch – one that is well supported, but still open-ended.
Facilitation Tips for Educators

For educators that are interested in introducing Scratch Microworlds to their students, we offer a few facilitation tips that will help plan an activity that matches the spirit of the microworlds. Some of the tips are more specific to microworlds, but most are relevant for any open-ended learning experience:

- Give learners the option to choose between the microworld.
- Allow learners to decide and how much time to spend in a microworld, and when to move to another.
- Trust the tinkering process – while at times it might look like they are just playing without a goal, trust that through it they will discover new things.
- Ask questions instead of giving answers.
- Allow time for learners to share their creations with others.

More good facilitation tips for constructionist experiences can be found in the Family Creative Learning guide [13] and the Learning and Facilitation Frameworks described by the Tinkering Studio [23].

Reflecting on the experiences of young people and educators allows us to keep improving Scratch Microworlds. Encouraged by the general positive feedback and observations, we hope to keep making them better and accessible to more people. In the next chapter, we take a step back and look at what can be learned from our experience and discuss future directions.
Chapter 9

Looking Back, Looking Forward

Guidelines for Designing Microworlds

In this section, I apply what we learned from designing and testing Scratch Microworlds to suggest a set of design guidelines. I hope that these guidelines can be useful for the design of other microworlds, as well as constructionist introductions to other tools. I present our recommendations in four areas: creating from the start; simplicity and creativity; scaffolding and choice; and interest-based experiences.

Creating from the Start

While many introductions to new tools use an instructionist approach, this work presented a successful model for a constructionist introduction to a tool. It is based on the belief that constructionist experiences do not need to wait until after learners already know how to use a tool. A constructionist introduction can support learners’ understanding of a new tool through a process of creating with it. We believe that young people learn better when they’re engaged in constructing personally-meaningful projects, even when they are just getting started, and that this approach increases their motivation and supports their learning.

Recently, some coding tools have added few creative and open-ended levels to their experience, recognizing the value of project-based learning. But these levels are typically available only after the learner finished the rest of the experience. For example, in the Moana Hour of Code activity [54], after navigating the character Moana through many obstacles, in the last level learners create their own dance for three little creatures. In this level learners can create something of their own, use their creativity, and explore playfully. But why can’t that be the first thing they do with code? Scratch Microworlds presents an alternative that is creative and open-ended from the first step.

When supporting creating in a simplified version of a tool, it is important to not lose sight of the transition to the full tool. Focusing on creating the best introduction experience can be in tension with planning for the most coherent way for learners to understand the tool. We chose to address this tension with Scratch Microworlds by leaning towards coherency — trying to keep the microworlds as similar to Scratch as possible in order to help make the transition easier, even at the cost of creating a more complex first experience.
Simplicity and Creativity

As discussed in chapter 5, creating a simple interface helps learners get started with a new experience, but many times it comes at the expense of engagement, making it more likely that learners lose interest and leave. To simplify with a minimal effect on engagement, we suggest carefully evaluating the advantages and disadvantages of the features to be removed. For example, in the microworlds projects we removed many of the buttons, but carefully evaluated the idea of removing two features: the ability to move between sprites and the ability to add a new sprite. We decided for simplicity to remove the ability to add a new sprite, but that it was important to keep the ability to move between different sprites.

While simplifying the environment, we made sure to keep the ability to tinker with the blocks and code, so learners can experiment and explore possibilities and see the results of their actions. By providing a playful, experimental, iterative style of engagement, we allow learners to construct their own understanding and follow their own path. For a more in-depth discussion of ways to support tinkering, see Resnick & Rosenbaum, 2013 [41].

Designing for a variety of projects also supports creativity, even in a simplified interface. When designing a learning experience, avoid designing it with one solution in mind. Make sure that there are many ways to use the building blocks you provide (In the Scratch Microworlds case, these were both the media assets and the coding blocks). Avoid a situation where to create something interesting learners have to configure the building blocks in a specific way. To test it, think about their ability to “tinker their way” into a meaningful project. Pay special attention to situations where an implicit goal leads all learners in the same path, to create the same project – make sure to design experiences that encourage a variety of potential goals.

Another way to enhance learners’ engagement and support their creative process is adding playful elements to the experience. An unexpected outcome, a funny sound, or an interesting pattern can draw learners in and encourage them to explore more. In Scratch Microworlds we added interesting sounds and images, and used coding blocks with surprising behaviors, such as visual effects and random numbers.

Scaffolding and Choice

The balance between providing support for newcomers and allowing them to control their learning process is a main consideration in constructionist learning environments. One signal of this balance is the ability of learners to make meaningful choices. Learners should be able to make as many decisions as possible – both on what to create and explore, and about their learning process. An important emphasis is that decisions have to be meaningful – not just a superficial choice between a boy or girl character or the color of the background, but decisions which have a real impact on the project and the experience. Examples of the ways we incorporated learners’ control into the microworlds project are described in chapter 6.

92
Allowing learners to control their learning process can include: deciding what to work on, when to move forward, and where to move to. When allowing learners to control moving forward, we found that learners needed a way to reverse their action, as they sometimes moved on before they were ready for it. Controlling the project they are working on is another important aspect of agency. We suggest providing ways for learners to create many different types of projects, and providing many options for themes, images and sounds. This can help them develop an emotional connection and ownership over their projects.

We also encourage considering the different ways support is provided through the experience. Giving suggestions instead of answers, encouraging exploration and experimentation, and not assuming that all learners will want to achieve the same things are all useful facilitation methods which can be used in digital learning environments as well.

Interest-Based Experiences

Building on learners' interests is a core idea in constructionism. Chapter 7 discusses many of our experiments and findings about connecting to learners' interests. Matching themes with the interests of learners increases their motivation and allows them to have a deeper connection with the experience. However, many different themes are required to support a large diverse set of learners. When choosing themes, we recommend considering the whole range of learners you want to reach. Pay special attention to groups that are not currently represented and look for themes that would draw them in.

We recommend designing experiences that have a strong connection to the underlying theme chosen. This include choosing relevant components such as images and sounds, but also making sure that the experience allows the learner to really engage with the topic of interest. When designing theme-based experiences, we found different levels of connection between the theme and the project. An example of a weak connection is using different popular characters in different themes, but having the learner do the same thing in all of them (like guiding the character through a maze). A stronger connection would be one where the dancer character dances, the musician makes music, and the builder builds.

Another important aspect of interest-based learning is the "width of the walls" – what is the range of projects the experience supports? To engage a wide range of learners, the experience needs to have "wide walls" and support a large variety of projects. Designing for wide walls could mean offering many different themes, and making sure each theme allows creating different projects. Especially for introductory experiences, we recommend providing a wide range of starting points for the learners to choose from, which might increase their motivation and engagement. We also recommend choosing components that contain many options, but are easy to use. For example, many of our microworlds take advantage of the visual effect block, which allows creating many variations of visual changes. On the other hand, we try to refrain from using blocks with only one use – like changing the size of a sprite.
Limitations

Importance of the Context
Conducting research on educational technologies should always take into account the context of the testing, as an educational experience is not just the technology, but also the people and cultures present [29]. In my work, I tried to avoid thinking about the microworlds in a “technocentric” way, which only looks at the technology [29]. But taking into account all parts of the experience, it becomes harder to make generalization about the technology. We conducted playtests in many different contexts, which are similar to the scenarios in which microworlds might be used. While this diversity helped us see the variety of ways microworlds might be used, and learn more about how and when they might be most successful, it also increased the variance between the testing sessions greatly. What seemed to work great in one workshop was completely unsuccessful in another, and the variance made it harder to know if the cause for it was the change in the prototype, the change in setting, or both.

Single-Meeting Evaluation
Because Scratch Microworlds were designed as an introduction experience, we conducted every testing session with a different group of participants. We met each participant only once, and usually for a short period of time. This allowed us to focus on the first experience of the learners, but prevented us from seeing the effect microworlds might have in the long run. Using a framework of single-meeting sessions meant that we could not evaluate some interesting questions, like the effect on learners’ perception of themselves as digital creators, and their pattern of engagement and retention with Scratch.

Fast Iterations
Due to the short timeline of this work, every iteration of the prototype included many changes, as seen in previous chapters. With every iteration, we “tweaked” many of the parameters of the experience. For example, within the same iteration we changed the subset of block and their number, we changed the looks and location of some of the buttons, and changed and added to the assets available within the microworlds. While it allowed faster improvements, it also made it harder to evaluate the effect of each one of those smaller design decisions.

Facilitation
In all of our testing session, my team and I were facilitating the experience. We designed workshops to include very little instruction, a lot of time for exploration and provided help to learners in a “constructionist” manner – by encouraging them to try different things, support their exploration and help them generate ideas. But that also meant we did not get to see how educators would handle facilitation of the microworlds without us, and how would that affect the experience of the learners.
Future Directions
Reflecting on the iterative design process and the feedback collected raised new questions and opened up new directions for research. I present some of these ideas here, in two parts – further explorations in design and implementation, and further explorations in research. While broken into two sections here, the greatest value will definitely come from exploring both of these directions together.

Design and Implementation

Integrating Microworlds into Scratch 3.0
As the work on Scratch 3.0 continues, Scratch Microworlds could be integrated into the new editor experience. We built the last prototypes on top of Scratch 3.0, but there’s still much work in transforming it from a prototype used to allow for fast iterations to a stable, scalable and coherent part of the Scratch editor. However, the impact of integrating microworlds into the editor could be great, as it will allow millions of people all over the world to use them. In the following months, I hope to work with the Scratch product and engineering teams to identify which parts of the microworlds will be most useful and feasible to integrate, and work on their integration.

Creating More Microworlds
The value of having many different microworlds became clearer through our design process. We collected many ideas for new microworlds from the children and educators we met, and implementing some of those will allow us to support more young learners that are interested in a variety of topics. As reflected in this work, designing new microworlds consists of much more than just choosing the right theme, and by creating a larger set we can also refine our understanding of the qualities of a good microworld and their limitations.

More Support for the Interface
While we experimented with providing contextual help to address some of the basic issues learners had when getting started, we decided not to focus on it at this time. Still, I believe that there is much more to be explored in these areas. Learners need help understanding how to drag and connect blocks, how to run their code, and how to fix common bugs. If we could find a way to provide that support in the microworlds experience, young learners might be able to use it without a facilitator.

Using Starter Code Effectively
Our preliminary findings show that adding starter code to microworlds could be very helpful to learners, both as a place to start from and as a model of how to create code with the blocks. By experimenting with different features of the starter code, we might be able to use it more effectively, balancing the benefits of having a working example with the possible limiting effect
on the imagination. In future microworlds, we plan to experiment with different types of coding stacks, different sizes and different behaviors. Trying out microworlds with starter code that gets added every time the learner asks for more blocks is another area for exploration.

_Easing the Transition to Scratch_

We saw learners get really excited about the option to continue working on their projects in the full version of the Scratch editor. But, most of them did not find the option of doing it themselves. In future iterations, making this option more prominent and coherent might help learners find it.

_Tools for Easier Creation of Microworlds_

Currently, developing a new microworld with the recent prototypes requires manually manipulating JSON files. This leaves only few people capable of changing and adding new microworlds. By developing tools that will make it easier to create new microworlds, we would be able to develop the microworlds more quickly. More importantly, we will be able to include others – educators, children and collaborators in the process of designing new microworlds.

Research

_Microworlds Effect on Long Term Engagement_

As mentioned before, the short engagement we had with participants is one of the limitations of this research. It limited our ability to learn more about the ways microworlds affect their learning trajectories, perceptions and retention. Conducting a longer-term research would allow us to address some of these questions. I’m mostly interested in the ways microworlds might affect the ways children interact with Scratch later on – do they tinker more? Do they explore different types of projects? Is it easier for them to follow a tutorial after using a microworld?

In addition, I would like to use more screen recordings and more interviews with participants. These two methods proves extremely useful in helping us understand how participants use and make sense of Scratch and the microworlds. I would also like to learn more about their self-perceptions towards computational creation and the things that inspire them.

_Comparative Studies_

Comparing the experiences of newcomers using microworlds to other introduction resources will help us further understand the affordances and limitations of Scratch Microworlds. One interesting comparison would be introductions to Scratch using microworlds or other learning resources, like our tutorials or coding cards. Another comparison would be comparing introductions to coding via Scratch Microworlds or via puzzle-based tools. In both studies, I’d like to evaluate learners’ understanding of computational concepts, as well as their use of computational practices, and their computational perspectives [7].
Teaching with Microworlds

Working with educators on designing new microworlds and evaluating current ones helped us learn more about their needs and ideas. As the microworlds get to a more stable state, educators can start using them with their students. This will allow us to observe and investigate how they use microworlds, and what kinds of support will help them do it successfully. If we can provide them with a way to build their own microworlds we could also explore their creations, both to get ideas for new microworlds, and to learn more about the affordances and limitations of them.
Bibliography


[50] Scratch Editor with Tips Window: scratch.mit.edu/create.