A Dataset of Scratch Programs: Scraped, Shaped and Scored

Efthimia Aivaloglou*, Felienne Hermans*, Jesús Moreno-León†, and Gregorio Robles‡
*Delft University of Technology, The Netherlands
{e.aivaloglou, f.f.j.hermans}@tudelft.nl
†Programamos.es & Universidad Rey Juan Carlos, Spain
jesus.moreno@programamos.es
‡Universidad Rey Juan Carlos, Fuenlabrada (Madrid), Spain
grex@gsyc.urjc.es

***Abstract*—Scratch is increasingly popular, both as an introductory programming language and as a research target in the computing education research field. In this paper, we present a dataset of 250K recent Scratch projects from 100K different authors scraped from the Scratch project repository. We processed the projects' source code and metadata to encode them into a database that facilitates querying and further analysis. We further evaluated the projects in terms of programming skills and mastery, and included the project scoring results. The dataset enables the analysis of the source code of Scratch projects, of their quality characteristics, and of the programming skills that their authors exhibit. The dataset can be used for empirical research in software engineering and computing education.**

I. Introduction

Scratch [1] is a block-based programming language developed to serve as a stepping stone for children from 8 to 16 years old to the more advanced world of computer programming. It offers a web-based programming environment that enables creating games and interactive animations. The public repository of Scratch programs contains over 19 million projects and 16 million users.

A number of works in the computing education research field attempt to assess the programming skills that novice programmers develop in the Scratch environment. Some utilize program data collected during specific programming courses (e.g., [2], [3], [4]), while others utilize the dataset made available by the Lifelong Kindergarten Group at the MIT Media Lab [5], which contains data for Scratch projects created until 2012 (e.g., [6], [7], [8]). In addition to identifying indications of learning of programming concepts, static analysis of Scratch programs has also been performed for identifying code smells and bad programming practices (e.g., [9], [10]), and automated quality assessment tools have been proposed (e.g., Hairball [11] and Dr. Scratch [12]).

While Scratch is receiving increasing interest as an introductory programming language, there is no recent dataset of Scratch programs available to the research community. The one made available from the MIT Media Lab concerns projects created using the previous, initial version of the Scratch application, before the introduction of the current Scratch version (Scratch 2) and of the web programming interface in 2013. It is since then that the popularity of Scratch started to increase[1]. Scratch 2 introduced several features relating to important programming concepts[2], like custom blocks (the equivalent to procedure definitions), and therefore that dataset does not include projects utilizing those features.

The goal of this paper is to present an open and timely dataset of recent Scratch programs, along with their metadata, that can facilitate quantitative research in the fields of source code analysis and computing education. The dataset contains 250,000 Scratch projects, from more than 100,000 different users, that were scraped from the Scratch project repository. It is made available as a database[3] which includes, for each Scratch project, its metadata and the program data, along with programming mastery scoring results from the Dr. Scratch quality assessment tool [12].

II. Dataset Construction

A. Data Collection

To collect the data from the web interface of the Scratch project repository, we built a scraping program. The web scraping program starts by reading the Scratch projects page[4] and thus obtains the project identifiers of projects that were most recently shared. Subsequently, it retrieves a JSON file for each of the listed projects[5].

We ran the scraper on March 2nd 2016 for 24 hours and, during that time, it obtained the JSON files for 250,163 projects. Out of those, we failed to parse and further analyze 2,367 projects due to technical difficulties with the JSON files. The scraper and all project files are available[3].

Once we obtained the Scratch projects, we parsed the JSON files according to the specification of the format[6]. This resulted in a list of used blocks per project, within the sprites and the stage of the project. We also cross referenced all blocks

[1]Monthly activity trends can be found at https://scratch.mit.edu/statistics/

[2]The complete list of features introduced in Scratch 2 can be found at https://wiki.scratch.mit.edu/wiki/Scratch_2.0

[3]https://github.com/TUDelftScratchLab/ScratchDataset

[4]https://scratch.mit.edu/explore/projects/all/

[5]For a given project id x, the program's JSON representation can be obtained via https://cdn.projects.scratch.mit.edu/internalapi/project/x/get

[6]http://wiki.scratch.mit.edu/wiki/Scratch_File_Format_(2.0)

with the Scratch wiki to determine their shapes and category. For example, `When Green Flag Clicked` is a *Hat block* from the *Events category*. We included blocks from Scratch extensions, such as the LEGO WeDo extensions, that were found in the dataset.

B. Calculation of Programming Mastery Scores

We analyzed the projects with Dr. Scratch, a tool that statically inspects Scratch projects' source code to assign scores on seven dimensions of computational thinking: abstraction and problem decomposition, logical thinking, synchronization, parallelism, algorithmic notions of flow control, user interactivity and data representation. These dimensions are given a value from 0 to 3. The mastery score is the aggregated punctuation of the seven dimensions, and therefore ranges from 0 to 21.

Dr. Scratch also detects: intra-project software cloning, i.e., repetition of code; the use of custom blocks, which is the way functionality can be reused in Scratch programs; and the use of instances of sprites, a feature labeled in Scratch as clone creation. These values are included in the dataset.

Out of the 250,163 projects, 231,050 were successfully analyzed with Dr. Scratch. This was due to problems with non-official extension blocks and with some non-ASCII characters in sprites names.

C. Importing the Data

All scraped project data and metadata, including the list of used blocks and parameters, were imported in a relational database. We also imported the data on the shapes and the categories of the Scratch blocks. We then used SQL queries for normalizing the data and bringing it in its final schema, outlined in Table I.

III. Dataset Description

A. Data Representation

The projects' data are stored in a relational database. Each of the **`projects`** is identified by its Scratch project ID, stored in field `p_id`, while its author is identified by the `username`[7]. If a project is a remix of another one, the original project can be found in the **`remixes`** table. Table **`grades`** stores the Dr. Scratch results for the programming mastery metrics per project.

The schema of the database, outlined in Table I, reflects the structure of the Scratch programs. Projects contain sprites, which are entities with their own associated code. The code is organized into **`scripts`**, i.e., groups of Scratch code blocks, each script belonging to a sprite named `sprite-name`. In the example in Figure 1 there are three scripts, one initiated when the green flag is clicked, one then the sprite is clicked, and the 'backflip' custom block definition. Those custom blocks are the equivalent of procedure definitions, with their names and arguments stored in the **`procedures`** table.

[7]Each project and author page can be accessed in the Scratch web interface at https://scratch.mit.edu/projects/⟨p_id⟩ and https://scratch.mit.edu/users/⟨username⟩ respectively

Table	Key	Attribute(Description)
`projects`	PK	`p_id` (Scratch project ID)
		`project-name` (name given to project)
		`username` (author's Scratch username)
		`total-views` (project views number)
		`total-remixes` (project remixes number)
		`total-favorites` (total users favoriting)
		`total-loves` (users 'loving' the project)
		`is-remix` (calculated column, true if project is a remix of another one)
`scripts`	PK	`script_id` (auto increment)
	FK	`project_id` (`*...1 projects:p_id`)
		`sprite-type` ([$sprite \| stage \| procDef$])
		`sprite-name` (name of sprite the script is in)
		`script-rank` (project-level ranking of script)
		`coordinates` (x-y location of script in Scratch editor)
		`total-blocks` (calculated column, number of blocks comprising the script)
`procedures`	PK,FK	`script_id` (`1...1 scripts:script_id`)
		`proc-name` (name given to custom block)
		`total-args` (number of input arguments)
`blocks`	PK,FK	`script_id` (`*...1 scripts:script_id`)
	PK	`block-rank` (script-level ranking of block)
		`block-type` (`*...1 blockTypes:b-type`)
		`parameter-1` (value of 1st input parameter)
		...
		`parameter-24` (value 24th input parameter)
`blockTypes`	PK	`b-type` (Scratch name of predefined block)
		`category` (Scratch block category)
		`shape` (One of 8 block shapes in Scratch)
		`is-input` (true if block receives user input)
`remixes`	PK	`remix_id` (`project_id` of created project)
		`from-project_id` (`project_id` of original project)
`grades`	PK,FK	`project_id` (`1...1 projects:p_id`)
		`Abstraction` (Score)
		`Parallelism` (Score)
		`Logic` (Score)
		`Synchronization` (Score)
		`FlowControl` (Score)
		`UserInteractivity` (Score)
		`DataRepresentation` (Score)
		`Mastery` (Dr. Scratch total mastery score)
		`Clones` (number of software clones)
		`CustomBlocks` (number of user defined blocks)
		`InstancesSprites` (true if project uses Scratch clones)

TABLE I
DATABASE SCHEMA: TABLES AND ATTRIBUTES

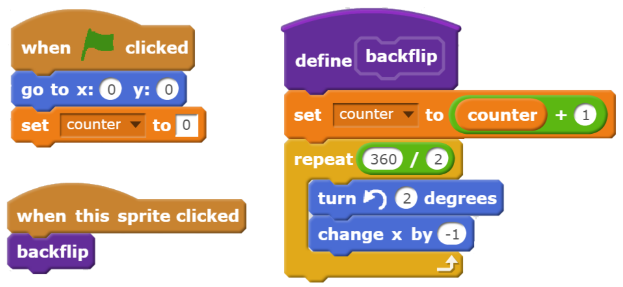

Fig. 1. Example of a Scratch program with three scripts in the same spite

	mean	min	Q1	median	Q3	max
Projects per username	2.3	1	1	1	2	868
Sprites *pp*	5.3	0	1	2	5	525
Scripts *pp*	16.2	0	2	4	11	3,038
Blocks *pp*	144.3	0	9	26	69	34,622
Custom block def. *pp*	0.6	0	0	0	0	372
Total views *pp*	5.8	0	1	1	4	27,993
Total remixes *pp*	0.1	0	0	0	0	306
Total favorites *pp*	0.5	0	0	0	0	2,582
Dr.Scratch mastery *pp*	8.9	0	6	8	11	21

TABLE II
SUMMARY STATISTICS OF THE DATASET (*pp* STANDS FOR PER PROJECT).

Projects	250,163
Non-empty projects	233,491
Projects analyzed by Dr. Scratch	231,050
Remixes of other projects	12,167
Usernames	109,960
Scripts	4,049,356
Blocks	36,085,654
Custom block definitions	204,018

TABLE III
DATASET CONTENTS

The coding elements in the Scratch environment are the **blocks**, which can receive input parameters, like block 'go to x:0 y:0' in the figure. Custom clock invocations are stored in this table just like regular blocks. To cater for the varying number of input parameters that those blocks can have, this table can store up to the maximum number of input parameters found in the dataset, which is 24. The ordering of the blocks in the scripts can be retrieved using the `block-rank` field, which represents a depth-first ranking of the blocks. In the script of the 'backflip' custom block definition in the example, the ranking of the blocks would be 'define backflip', 'set counter to', '+', 'counter', followed by the rest of the blocks. The clocks can be of different shapes and categories. In table **blockTypes** we have stored the encoding of all types of Scratch blocks and of the blocks from Scratch extensions.

B. Dataset Contents

As shown in Table III, the dataset contains the metadata of 250,163 Scratch projects, and the code of 233,491 of those that are non-empty. The projects were created by 109,960 different users. Most of the users, as shown in Table II, have created a single project. However, 40,000 users have created more than one and up to 868 projects in the dataset.

The majority of the projects have been viewed only once, and they have not been favorited or remixed. Around 10,000 of the total projects have been remixed at least once, while there are cases of very popular projects with more than 100 remixes and even one thousand cases with more than 100 views.

Table II also reveals the median project in the dataset: it contains 2 sprites, 4 scripts, 26 blocks, no custom block definitions, and receives a mastery score of 8 out of 21, which is considered a medium level of computational thinking skills development. The table highlights the existence of surprisingly complex projects, with 525 sprites and over 34,000 blocks.

IV. ENABLED RESEARCH AND DATASET EXTENSION

The evaluation of block-based languages in general, and Scratch in particular, as tools for programming education is receiving significant research attention. A number of studies have been carried out during the last years on understanding the programming practices of learners in those environments, on the programming skills they develop, and on the quality of their programs.

The first research direction that this dataset can be utilized for concerns the *assessment of the programming skills that novice programmers develop in the Scratch environment*. The dataset can be used for examining indications of learning of programming concepts and abstractions, as in [10]. Existing works in this research direction include the work by Maloney *et al.* [4], who analyzed 536 Scratch projects created in an after-school clubhouse for blocks that relate to programming concepts and found that within the least utilized ones are Boolean operators and variables. Another study on the internalization of programming concepts with Scratch with 46 students was presented in [2], where it was found that students had problems with initialization, variables and concurrency.

Towards this research direction, the dataset makes the source code of the Scratch programs available for static analysis and can be used for quantitatively evaluating the application of programming concepts and abstractions through the presence of Scratch blocks of the corresponding types.

Another promising research direction relates to the *quality of the programming artifacts developed in the Scratch environment*. The dataset has already been used for quality assessment, for the identification of programming smells [10], and for exploring indications of harmful programming habits [13]. Existing works in this direction include a study in a classroom setting [9], which highlighted two bad programming habits in Scratch, namely bottom-up development and extremely fine-grained programming. The authors connected the later to the reduced use of if-blocks and finite loops and the increased use of infinite loops. Related to smell detection are the Scratch automated quality analysis tools Hairball [11], a static analysis tool that can detect initialization problems and unmatched broadcast and receive blocks, and Dr. Scratch [14], which extends Hairball to detect two bad programming habits: not changing the default object names and duplicating scripts.

This dataset will enable examining code quality and code smells on a large set of Scratch projects. Static analysis of the source code can also be performed for the identification of other types of smells [15] that might be common in the artifacts of novice programmers.

Moving from the software engineering to the computing education research field, a research direction with increased data requirements concerns the *learning progressions of novice programmers*. This includes examining the factors that support learning and improving their programming skills. Existing works in this area have used the dataset of the previous version of Scratch programs, created until 2012, and described in the Introduction. They include the works of Dasgupta et al., who investigated how project remixing relates to the adoption of new computational thinking concepts [8], and Yang et al., who examined the learning patterns of programmers in terms of block use over their first 50 projects [7].

Research towards this direction requires extending the dataset with the complete set of projects for the included users, along with their creation dates. This would enable examining how their programming skills and mastery evolved through time, possibly through remixing the projects of others. An even richer extension of the dataset would be with periodical snapshots of the included projects that are still in development. This would enable reconstructing project versioning information, not available from the Scratch interface, and thus examining how the projects are developed and evolve over time.

Apart from quantitative studies, the dataset can support the design of experiments and field studies, whose material often includes Scratch programs with specific characteristics. For example, examples of Scratch programs exhibiting specific smells have been used in a controlled experiment to determine how the smells affect the understanding and the modification of the programs [16]. This queryable dataset can support finding example programs with characteristics according to the experiment design and the field of the study.

V. Limitations

The only information about the authors of the Scratch projects contained in the dataset is their username. This suffices for creating project portfolios of the authors and for facilitating research in the directions described in the previous section. However, other directions in the computing education research field require richer author data, and especially data on their gender and age. There are, for example, indications that specific programming concepts are better understood after certain ages —Seiter and Foreman [17] analyzed 150 Scratch projects from primary school students and found that design patterns requiring the understanding of parallelism, conditionals and, especially, variables were under-represented by all grades apart from 5th and 6th. The effect of gender and account age were also examined in [6] in relation to the use of programming concepts. However, this dataset does not include information on the gender and the age of authors, and cannot be extended to do so using the current Scratch web interface because this information is not available in the user profile page[7], even though it is provided by users upon registration.

Another limitation of this dataset is that we did not scrape a random sample of Scratch projects, but the most recent ones during the time that we run the scraper. It could be the case that the programming habits of Scratch users are changing over time. However, we counterbalanced that by collecting a large dataset which comprises around 1.3% of all 19 million shared Scratch projects, and is the most recent one to be made available to the research community. Projects in the dataset are the ones that their authors shared publicly when the scraper run. The dataset contains cases of projects that are no longer publicly shared and are thus no longer accessible through the Scratch web interface.

VI. Conclusion

We presented a dataset of recent Scratch programs scraped from the Scratch project repository. It is made available as a database[3] which includes the source code of the Scratch projects, their metadata, and their programming mastery scoring results. The dataset can facilitate research in software engineering and computing education, for topics like the assessment of the programming skills that novice programmers develop, the exploration of their learning progressions, and issues related to the quality of the programs, such as the identification of smells and bad programming habits.

References

[1] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for All," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, Nov. 2009.

[2] O. Meerbaum-Salant, M. Armoni, and M. M. Ben-Ari, "Learning Computer Science Concepts with Scratch," in *Proceedings of ICER 2010*. New York, NY, USA: ACM, 2010, pp. 69–76.

[3] A. Wilson, T. Hainey, and T. Connolly, "Evaluation of computer games developed by primary school children to gauge understanding of programming concepts," in *European Conference on Games Based Learning*. Academic Conferences International Limited, 2012, p. 549.

[4] J. H. Maloney, K. Peppler, Y. Kafai, M. Resnick, and N. Rusk, "Programming by choice: Urban youth learning programming with Scratch," in *Proceedings of the 39th SIGCSE*. ACM, 2008, pp. 367–371.

[5] B. M. Hill and A. Monroy-Hernández, "A longitudinal dataset of five years of public activity in the Scratch online community," *Scientific Data*, vol. 4, pp. 170 002 EP –, 01 2017.

[6] D. A. Fields, M. Giang, and Y. Kafai, "Programming in the wild: Trends in youth computational participation in the online Scratch community," in *Proceedings of the 9th WiPSCE*. ACM, 2014, pp. 2–11.

[7] S. Yang, C. Domeniconi, M. Revelle, M. Sweeney, B. U. Gelman, C. Beckley, and A. Johri, "Uncovering trajectories of informal learning in large online communities of creators," in *Proceedings of the Second ACM Conference on Learning @ Scale*. ACM, 2015, pp. 131–140.

[8] S. Dasgupta, W. Hale, A. Monroy-Hernández, and B. M. Hill, "Remixing as a pathway to Computational Thinking," in *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, ser. CSCW '16. ACM, 2016, pp. 1438–1449.

[9] O. Meerbaum-Salant, M. Armoni, and M. Ben-Ari, "Habits of Programming in Scratch," in *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '11. New York, NY, USA: ACM, 2011, pp. 168–172.

[10] E. Aivaloglou and F. Hermans, "How kids code and how we know: An exploratory study on the Scratch repository," in *Proceedings of ICER 2016*. ACM, 2016, pp. 53–61.

[11] B. Boe, C. Hill, M. Len, G. Dreschler, P. Conrad, and D. Franklin, "Hairball: Lint-inspired Static Analysis of Scratch Projects," in *Proceeding of the 44th SIGCSE*. New York, NY, USA: ACM, 2013, pp. 215–220.

[12] J. Moreno and G. Robles, "Automatic detection of bad programming habits in Scratch: A preliminary study," in *2014 IEEE Frontiers in Education Conference (FIE)*, Oct. 2014, pp. 1–4.

[13] G. Robles, J. Moreno-León, E. Aivaloglou, and F. Hermans, "Software clones in Scratch projects: On the presence of copy-and-paste in Computational Thinking learning," in *Proceedings of the 11th International Workshop on Software Clones*, 2017, pp. 31–37.

[14] J. Moreno-León, G. Robles, and M. Román-González, "Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking," *RED : Revista de Educación a Distancia*, no. 46, pp. 1–23, Jan. 2015. [Online]. Available: https://doaj.org

[15] M. Fowler, *Refactoring: Improving the design of existing code*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[16] F. Hermans and E. Aivaloglou, "Do code smells hamper novice programming? A controlled experiment on Scratch programs," in *2016 IEEE 24th International Conference on Program Comprehension*, 2016, pp. 1–10.

[17] L. Seiter and B. Foreman, "Modeling the learning progressions of Computational Thinking of primary grade students," in *Proceedings of ICER 2013*. ACM, 2013, pp. 59–66.